Generalized Principal Component Analysis

Modeling & Clustering of High-Dimensional Data

René Vidal (JOHNS HOPKINS UNIVERSITY)
Yi Ma (UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN)
S. Shankar Sastry (UNIVERSITY OF CALIFORNIA AT BERKELEY)

May 20, 2010

This is page ii
Printer: Opaque this

Contents

Chapter 1
Introduction

"The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work."

– John von Neumann

The primary goal of this book is to study how to model a data set that consists of multiple subsets, each one drawn from a different primitive model. In different contexts, such a data set is sometimes referred to as "mixed," or "multi-modal," or "multi-model," or "piecewise," or "heterogeneous," or "hybrid." To unify the terminology, in this book, we will refer to such data as "mixed data" and the model used to fit the data as a "hybrid model." Thus, a hybrid model typically consists of multiple constituent (primitive) models. Modeling mixed data with a hybrid model implies grouping the data into multiple (disjoint) subsets and fitting each subset with one of the constituent models. In the literature, the words "group," or "cluster," or "partition," or "decompose," or "segment" are often used interchangeably. However, in this book, we will use the words "group," or "cluster," or "partition" primarily for the data points,[1] and use the words "decompose" or "segment" for the associated models.[2]

[1]For instance, we may say "group (or cluster or partition) the data into multiple subsets," or "a group (or a cluster) of sample points."

[2]For instance, we may say "decompose (or segment) a hybrid model into its constituent primitive models."

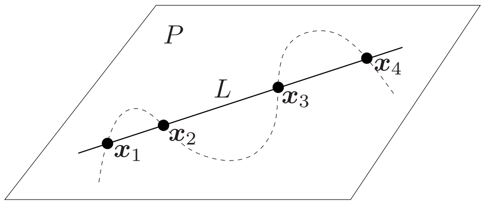


Figure 1.1. Four sample points on a plane are fitted by a straight line. However, they can also be fitted by many other smooth curves, for example the one indicated by the dashed curve.

In this chapter, we give a brief introduction to several concepts involved in modeling mixed data with hybrid models. First, we discuss some basic concepts associated with data modeling in general, such as the choice of the model class. Next, we motivate the problem of modeling mixed data with hybrid models by using several examples from computer vision, image processing, pattern recognition, system identification and system biology. We then give a brief account of geometric, statistical and algebraic methods for estimating hybrid models from data, with an emphasis on the particular case of modeling data with an *arrangement of subspaces*,[3] also called hybrid linear models. We finish the chapter with a discussion about how noise and outliers make the estimation problem extremely difficult, especially when the complexity of the model to be estimated is unknown.

1.1 Modeling Data with a Parametric Model

In scientific studies or engineering practice, one is frequently called upon to infer (or learn) a quantitative model M for a given set of sample points, denoted as $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$. For instance, Figure 1.1 shows a simple example in which one is given a set of four sample points on a two dimensional plane.

Obviously, these points can be fitted perfectly by a (one-dimensional) straight line. The line can then be called a "model" for the given points. The reason for inferring such a model is because it serves many useful purposes. For instance, it can reveal the information encoded in the data or the underlying mechanisms from which the data were generated. Alternatively, it can significantly simplify the representation of the given data set or help to predict effectively future samples.

1.1.1 The Choice of a Model Class

A first important consideration to keep in mind is that inferring the "correct" model for a given data set is an elusive, if not impossible, task. The fundamen-

[3]In this book, we will use interchangeably "a mixture," "a collection," "a union," or "an arrangement" of subspaces or models. But be aware that, in the case of subspaces, the formal terminology in Algebraic Geometry is "an arrangement of subspaces."

tal difficulty is that, if we are not specific about what we mean by a "correct" model, there could easily be many different models that fit the given data set "equally well." For instance, in the example shown in Figure 1.1, any smooth curve that passes through the sample points would seem to be as valid a model as the straight line. Furthermore, if there were noise in the given sample points, then any curve, including the line, passing through the points exactly would unlikely be the "ground truth."

The question now is: in what sense can we say that a model is correct or optimal for a given data set? On the one hand, to make the model inference problem meaningful, we need to impose additional assumptions or restrictions on the class of models considered. That is, we should not be looking for any model that can describe the data. Instead, we should look for a model M^* that is the *best* among a *restricted* class of models $\mathcal{M} = \{M\}$.[4] In fact, the well-known No Free Lunch Theorem[5] in computational learning implies that, in the absence of prior information or preference about the final model, there is no reason to prefer one optimization or learning algorithm over another [**?**]. On the other hand, to make the model inference problem tractable, we need to specify how restricted the class of models needs to be. A common strategy, known as the principle of Occam's Razor[6], is to try to get away with the simplest possible class of models that is *just necessary* to describe the data or solve the problem at hand. More precisely, the model class should be rich enough to contain at least one model that can fit the data to a desired accuracy and yet be restricted enough so that it is relatively simple to find the best model for the given data.

Thus, in engineering practice, the most popular strategy is to start from the simplest class of models, and only increase the complexity of the models when the simpler models become inadequate. For instance, to fit a set of sample points, one may try first the simplest class of models, namely linear models, followed by the class of hybrid (piecewise) linear models, then followed by the class of (piecewise) quadratic models, and finally followed by the class of general topological manifolds. One of the goals of this book is to demonstrate that among them, piecewise linear (and quadratic) models can already achieve an excellent balance between expressiveness and simplicity for many important practical problems.

[4]Or equivalently, we may impose a non-uniform prior distribution over all models.

[5]Or more precisely, the "No Free Lunch Theorem for Search," attributed to Wolpert and Macready (1995), states "*...all algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions. In particular, if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.*"

[6]Occam's (or Ockham's) razor is a principle attributed to the 14th century logician and Franciscan friar; William of Occam: "*Pluralitas non est ponenda sine neccesitate,*" which translates literally as "*entities should not be multiplied unnecessarily.*" In science, this principle is often interpreted as "*when you have two competing theories which make exactly the same predictions, the simpler one is better.*"

1.1.2 Statistical Models versus Geometric Models

In this book, we consider the problem of modeling mixed data under the assumption that the group memberships of the given data points are not known *a priori*. As a consequence, the problem of modeling mixed data falls into the category of *unsupervised learning*. In the literature, almost all unsupervised learning methods fall into one of two categories. The first category of methods model the data as random samples from a probability distribution and tries to learn this distribution from the data. We call such models *statistical models*. The second category of methods models the overall geometric shape of the data set as smooth manifolds or topological spaces.[7] We call such models *geometric models*.

Statistical Learning.

In the statistical paradigm, one assumes that the points $\boldsymbol{x}_i$ in the data set $\boldsymbol{X}$ are drawn independently from a common probability distribution $p(\boldsymbol{x})$. So the task of learning a model from the data becomes one of inferring the most likely probability distribution within a family of distributions of interest (for example the Gaussian distributions). Normally the family of distributions is parameterized and denoted as $\mathcal{M} \doteq \{p(\boldsymbol{x}|\theta) : \theta \in \Theta\}$, where $p(\boldsymbol{x}|\theta)$ is a probability density function parameterized by $\theta \in \Theta$, and Θ is the space of parameters. Consequently, the optimal model $p(\boldsymbol{x}|\theta^*)$ is given by the maximum likelihood (ML) estimate[8]

$$\theta^*_{ML} \doteq \arg\max_{\theta\in\Theta} \prod_{i=1}^{N} p(\boldsymbol{x}_i|\theta). \tag{1.1}$$

If a prior distribution (density) $p(\theta)$ of the parameter θ is also given, then, following the Bayesian rule, the optimal model is given by the maximum a posteriori (MAP) estimate

$$\theta^*_{MAP} \doteq \arg\max_{\theta\in\Theta} \prod_{i=1}^{N} p(\boldsymbol{x}_i|\theta)p(\theta). \tag{1.2}$$

Many effective methods and algorithms have been developed in the statistics and machine learning literature to infer the optimal distribution $p(\boldsymbol{x}|\theta^*)$ or a good approximation of it if the exact solution is computationally prohibitive. A brief review is given in Appendix A. The estimated probability distribution gives a *generative* description of the samples and can be used to generate new samples or predict the outcomes of new observations.

---

[7] Roughly speaking, a smooth manifold is a special topological space that is locally smooth – Euclidean space-like, and has the same dimension everywhere. A general topological space may have singularities and consist of components of different dimensions.

[8] If the true distribution from which the data are drawn is $q(\boldsymbol{x})$, then the maximum likelihood estimate $p(\boldsymbol{x}|\theta^*)$ minimizes the Kullback-Leibler (KL) divergence: $D(q\|p) = \int q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\, d\boldsymbol{x}$ among the given class of distributions, see Appendix A.

Geometric Modeling.

In many practical scenarios, we may not know *a priori* the statistical origins of the data. Also, the amount of data may not be sufficient to determine a unique optimal distribution within a large class of possible distributions. In such cases, we may exploit the fact that the data points are often subject to topological or geometric constraints, e.g., they must lie in a low-dimensional subspace. This implies that the data can only be represented with a probability distribution that is close to being singular.[9]

In general, it is very ineffective to learn such a singular or approximately singular distribution via statistical means [Vapnik, 1995]. Thus, an alternative data-modeling paradigm is to directly learn the overall geometric shape of the given data set. Typical methods include fitting one or more geometric primitives such as points[10], lines, subspaces, surfaces, and manifolds to the data set. For instance, the approach of classical principal component analysis (PCA) is essentially to fit a lower-dimensional subspace, say $S \doteq \text{span}\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_d\}$, to a data set in a high-dimensional space, say $\boldsymbol{X} = \{\boldsymbol{x}_i\} \subset \mathbb{R}^D$. That is, we try to represent the data points as:

$$\boldsymbol{x}_i = y_{i1}\boldsymbol{u}_1 + y_{i2}\boldsymbol{u}_2 + \cdots + y_{id}\boldsymbol{u}_d + \varepsilon_i, \quad \forall\, \boldsymbol{x}_i \in \boldsymbol{X}, \tag{1.3}$$

where $d < D$, $y_{ij} \in \mathbb{R}$, and $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_d \in \mathbb{R}^D$ are unknown model parameters that need to be determined – playing the role of the parameters θ in the foregoing statistical model. The line model in Figure 1.1 can be viewed as an example of PCA for the four points on the plane. In the above equation, the term $\varepsilon_i \in \mathbb{R}^D$ denotes the error between the sample and the model. PCA minimizes the error $\sum_i \|\varepsilon_i\|^2$ for the optimal subspace (see Chapter 2 for details).[11] In general, a geometric model gives an intuitive description of the samples, and it is often preferred to a statistical one as a "first-cut" description of the given data set. Its main purpose is to capture global geometric, topological, or algebraic characteristics of the data set, such as the number of clusters and their dimensions. A geometric model always gives a more compact representation of the original data set, which makes it useful for data compression and dimensionality reduction.

As two competing data-modeling paradigms, the statistical modeling techniques in general are more effective in the high-noise (or high-entropy) regime when the generating distribution is (piecewise) non-singular; while the geometric techniques are more effective in the low-noise (or low-entropy) regime when the underlying geometric space is (piecewise) smooth, at least locally. The two paradigms thus complement each other in many ways. On the one hand, once the overall geometric shape, the clusters and their dimensions are obtained from

[9]Singular distributions are probability distributions concentrated on a set of Lebesgue measure zero. Such distributions are not absolutely continuous with respect to the Lebesgue measure. The Cantor distribution is one example of a singular distribution.

[10]As the means of clusters.

[11]When the errors ε_i are independent samples drawn from a zero-mean Gaussian distribution, the geometric formulation of PCA coincides with the classical statistical formulation [**?**].

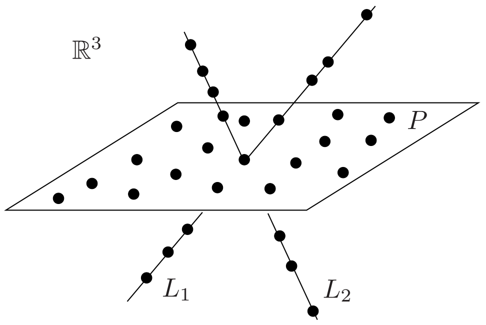


Figure 1.2. A set of sample points in $\mathbb{R}^3$ are well fitted by a hybrid model with two straight lines and a plane.

geometric modeling, one can choose the class of probability distributions more properly for further statistical inference. On the other hand, since samples are often corrupted by noise and sometimes contaminated with outliers, in order to robustly estimate the optimal geometric model, one often resorts to statistical techniques. Thus, although this book puts more emphasis on geometric and algebraic modeling techniques, we will also thoroughly investigate their connection to and combination with various statistical techniques (see Chapter **??**).

1.2 Modeling Mixed Data with a Hybrid Model

As we alluded to earlier, many data sets $\boldsymbol{X}$ cannot be modeled well by a single primitive model M in a pre-chosen or preferred model class $\mathcal{M}$. Nevertheless, it is often the case that if we *group* such a data set $\boldsymbol{X}$ into multiple disjoint subsets:

$$\boldsymbol{X} = \boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \cdots \cup \boldsymbol{X}_n, \quad \text{with } \boldsymbol{X}_l \cap \boldsymbol{X}_m = \emptyset, \text{ for } l \neq m, \tag{1.4}$$

then each subset $\boldsymbol{X}_j$ can be modeled sufficiently well by a model in the chosen model class:

$$M_j^* = \arg\min_{M \in \mathcal{M}} \left[\text{Error}(\boldsymbol{X}_j, M)\right], \quad j = 1, 2, \ldots, n, \tag{1.5}$$

where $\text{Error}(\boldsymbol{X}_j, M)$ represents some measure of the error incurred by using the model M to fit the data set $\boldsymbol{X}_j$. Each model M_j^* is called a *primitive* or a *component* model. Precisely in this sense, we call the data set $\boldsymbol{X}$ *mixed* (with respect to the chosen model class $\mathcal{M}$) and call the collection of primitive models $\{M_j^*\}_{j=1}^n$ a *hybrid model* for $\boldsymbol{X}$. For instance, suppose we are given a set of sample points shown in Figure 1.2. These points obviously cannot be fitted well by any single line, plane or smooth surface in $\mathbb{R}^3$; but once they are grouped into three subsets, each subset can be fitted well by a line or a plane. Note that in this example the topology of the data is indeed “hybrid” – two of the subsets are of dimension one and the other is of dimension two.

1.2.1 Examples of Mixed Data Modeling

The problem of modeling mixed data is quite representative of many data sets that one often encounters in practical applications. To motivate further the importance of modeling mixed data, we give below a few real-world problems that arise in image processing and computer vision. Most of these problems will be revisited later in this book and more detailed and principled solutions will be given.

Image Representation and Segmentation

A first example arises in the context of *image representation and segmentation*. It is commonplace that, in an image, pixels at different regions have significantly different local color/texture profiles (normally in an $N \times N$ window around a pixel). Conventional image representation/compression schemes, such as JPEG or JPEG2000, often ignore such differences and apply the same linear filters or bases (for example the Fourier transform, discrete cosine transform, wavelets, or curvelets) to the entire set of local profiles. Nevertheless, modeling the set of local profiles as a mixed data set allows us to segment the image into different regions and represent each region differently. Each region consists of only those pixels whose local profiles span the same low-dimensional linear subspace.[12] The subspace basis can be viewed as a bank of adaptive filters for the associated image region. Figure 5.2 shows regions of an image segmented by such a hybrid representation. The obtained subspaces (and their bases) normally provide a

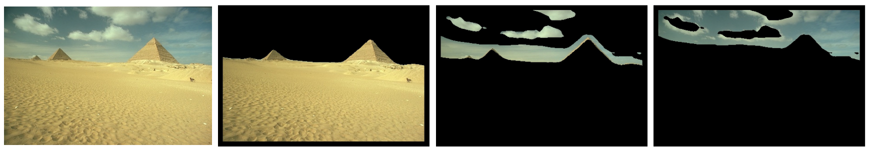

(a) Input image (b) First segment (c) Second segment (d) Third segment

Figure 1.3. Image segmentation based on fitting different linear subspaces (and bases) to regions of different textures. The three segments (or subspaces) correspond to the ground, the clouds, and the sky.

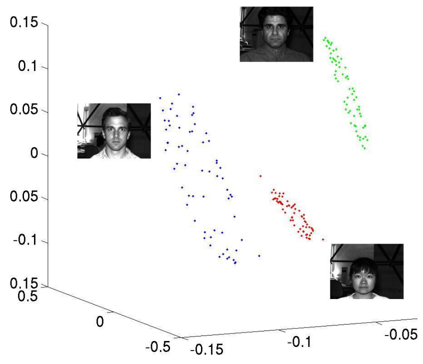
very compact representation of the image, often more compact than any of the aforementioned fixed-basis schemes. Therefore, they are very useful for purposes such as image compression, classification, or retrieval. More details on this image representation/segmentation scheme can be found in Chapter 6.

Face Classification under Varying Illumination

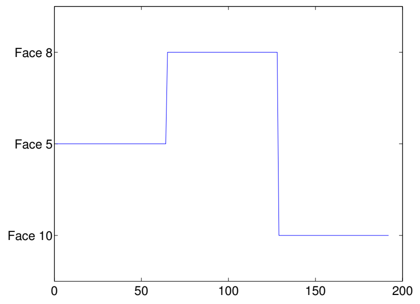
The next example arises in the context of image-based *face classification*. Given a collection of unlabeled images $\{I_i\}_{i=1}^n$ of several different faces taken under

[12]Unlike the previous two examples, there is no rigorous mathematical justification that local profiles from a region of similar texture must span a low-dimensional linear subspace. However, there is strong empirical evidence that a linear subspace is normally a very good approximation.

varying illumination, we would like to classify the images corresponding to the face of the same person. For a Lambertian object, it has been shown that the set of all images taken under all lighting conditions forms a cone in the image space, which can be well approximated by a low-dimensional subspace [**?**]. Since the images of different faces will live in different "illumination subspaces", we can classify the collection of images by estimating a basis for each one of those subspaces. This is obviously another subspace-segmentation problem. In the example shown in Figure 1.4, we use a subset of the Yale Face Database B consisting of $n = 64 \times 3$ frontal views of three faces (subjects 5, 8 and 10) under 64 varying lighting conditions. For computational efficiency, we first down-sample each image to a size of 30×40 pixels. We then project the data onto their first three principal components using PCA, as shown in Figure 1.4(a).[13] By modeling the projected data with a hybrid linear model in $\mathbb{R}^3$, we obtain three affine subspaces of dimension 2, 1, and 1, respectively. Despite the series of down-sampling and projection, the subspaces lead to a perfect classification of the images, as shown in Figure 1.4(b).



(a) Several images of three faces projected onto the first three principal components



(b) Classification of the images according to the three different faces

Figure 1.4. Classifying a subset of the Yale Face Database B consisting of 64 frontal views under varying lighting conditions for subjects 2, 5 and 8.

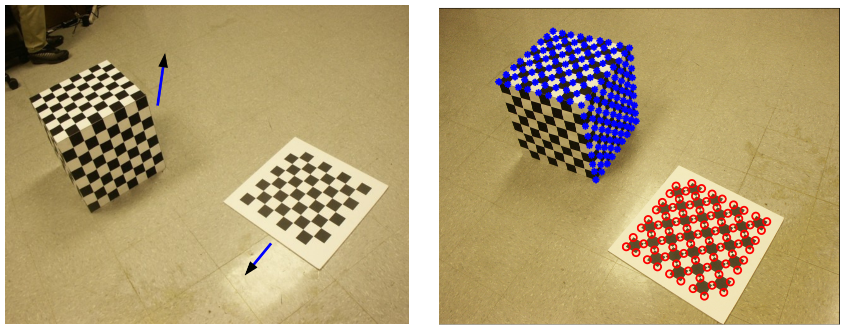
Segmentation of Moving Objects in Video

The next example is the *motion segmentation* problem that arises in the field of computer vision: given a sequence of images of multiple moving objects in a scene, how does one segment the images so that each segment corresponds to only one moving object? This is a very important problem in applications such as motion capture, vision-based navigation, target tracking, surveillance, etc.

One way of solving this problem is to extract a set of feature points in the first image and track these points through the video sequence. As a result one obtains

[13]The legitimacy of the projection process will be addressed in Chapter 4.

a set of point trajectories such that each trajectory corresponds to a different moving object in the video. It is well known from the computer vision literature [**?**, **?**] that feature points from two corresponding views of the same object are related by either linear or quadratic constraints depending on the type of motions and camera projection models (see Chapter 8). Therefore, mathematically, the problem of motion segmentation is equivalent to segmentation of points to different linear subspaces and quadratic surfaces. Figure 1.5 shows two images of a moving checker board and cube. The image on the left shows the starting positions of


(a) First frame of a video sequence containing two moving objects

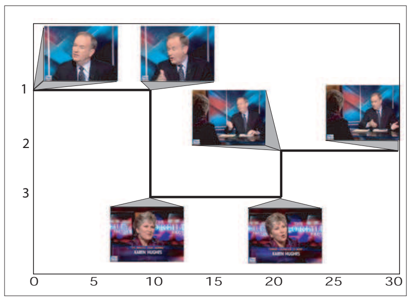
(b) Segmentation of feature points in the second frame according to different 3-D motions

Figure 1.5. Clustering feature points according to different 3-D motions.

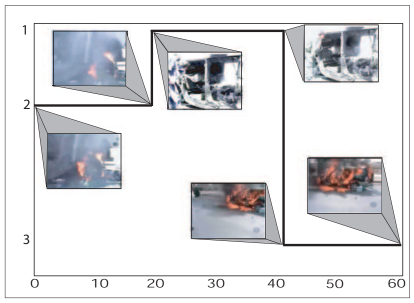
the board and the cube and their directions of motion; and the image on the right shows the final positions. The image on the right also shows the segmentation results obtained by fitting the motion flow of points on the cube and the board with a hybrid linear model. We will describe in detail the motion segmentation method used to achieve this result in Chapters **??** and 8.

Temporal Video Segmentation and Event Detection

Another example arises in the context of *detecting events from video sequences*. A typical video sequence contains multiple activities or events separated in time. For instance, Figure 1.6(a) shows a news sequence where the host is interviewing a guest and the camera is switching between the host, the guest and both of them. The problem is to separate the video sequence into subsequences, so that each subsequence corresponds to one of the three events. For this purpose, we assume that all the frames associated with the same event live in a low-dimensional subspace of the space spanned by all the images in the video, and that different events correspond to different subspaces. The problem of segmenting the video into multiple events is then equivalent to a subspace-segmentation problem. Since the image data live in a very high-dimensional space ($\sim 10^5$, the number of pixels), we first project the image data onto a low-dimensional subspace ($\sim$ 10) using principal component analysis (PCA) and then fit a hybrid linear model to the projected data to identify the different events. Figure 1.6 shows the segmentation results for two video sequences. In both cases, a perfect segmentation is obtained. We will describe in detail the segmentation method used to achieve the these results in Chapter **??**.



(a) Thirty frames of a video sequence of a television show clustered into three groups: host, guest, and both of them



(b) Sixty frames of a news video sequence clustered into three groups: car with a burning wheel, burnt car with people, and burning car

Figure 1.6. Clustering frames of a news video sequence into groups of scenes by modeling each group with a linear subspace.

Identification of Hybrid Dynamical Models

The last example arises in the context of modeling time series data with dynamical models. A popular dynamical model used to analyze a time series $\{y_t \in \mathbb{R}\}_{t\in\mathbb{Z}}$ is the linear auto-regressive (AR) model:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} + \varepsilon_t, \quad \forall t, \in \mathbb{Z}, \tag{1.6}$$

where $\varepsilon_t \in \mathbb{R}$ models the modeling error or noise and it is often assumed to be a white-noise random process. In order to capture more complex dynamics in the data, one can assume that y_t is the output of a piece-wise AR model, where the output at each time instant is drawn from one out of finitely many ARX models. Notice that at each time instant, the vector $\boldsymbol{x}_t = [y_t, y_{t-1}, \ldots, y_{t-n}]^\top$ lies on an n-dimensional hyperplane in $\mathbb{R}^{n+1}$. Therefore, the vectors $\boldsymbol{x}_t$ for all t lie in a collection of hyperplanes. As a consequence, the identification of the parameters of a piece-wise AR model can be viewed as another subspace-segmentation problem. We will discuss this and more general classes of hybrid dynamical models, together with algorithms for identifying the parameters of such models, in Chapters **??** and **??**.

As we see from the foregoing examples, in some cases, one can rigorously show that a given data set belongs to a collection of linear and quadratic surfaces of the same or of possibly different dimensions (motion segmentation example). In many other cases, one can use piecewise linear structures to approximate the data set and obtain a more compact and meaningful geometric representation of the data, including segments, dimensions, and bases (image representation, face classification and video segmentation examples). Subspace (or surface) segmentation is a natural abstraction of all these problems and thus merits systematic investigation. From a practical standpoint, the study will lead to many general and powerful modeling tools that are applicable also to many types of data, such as feature points, images, videos, audio data, dynamic data, genomic data, proteomic data, etc.

1.2.2 Mathematical Representations of Hybrid Models

The examples presented in the previous subsection argue forcefully for the development of modeling and estimation techniques for hybrid models. Obviously, whether the model associated with a given data set is hybrid or not depends on the class of primitive models considered. In this book, the primitives are normally chosen to be simple classes of smooth manifolds or non-singular distributions.

For instance, one may choose the primitive models to be linear subspaces. Then one can use an arrangement of linear subspaces $\{S_i\}_{i=1}^n \subset \mathbb{R}^D$,

$$Z \doteq S_1 \cup S_2 \cup \cdots \cup S_n, \tag{1.7}$$

also called a hybrid linear model, to approximate many nonlinear manifolds or piecewise smooth topological spaces. This is the typical model in Generalized Principal Component Analysis (GPCA) and it is to be studied for most part of this book.

The statistical counterpart of the algebraic model in (1.7) is to assume that the samples points are drawn independently from a mixture of near singular Gaussian distributions $\{p_i(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^D\}_{i=1}^n$ with probability density function

$$q(\boldsymbol{x}) \doteq \pi_1 p_1(\boldsymbol{x}) + \pi_2 p_2(\boldsymbol{x}) + \cdots + \pi_n p_n(\boldsymbol{x}), \tag{1.8}$$

where $\pi_i > 0$ and $\pi_1 + \pi_2 + \cdots + \pi_n = 1$. This is the typical model studied in mixtures of probabilistic principal component analysis (MPPCA) [Tipping and Bishop, 1999a]. A classical way of estimating such a mixture model is the Expectation Maximization (EM) algorithm, which infers the membership of each sample as a hidden random variable (see Appendix A for a review).

In this book, we will study and clarify the similarities and differences between these geometric models and statistical models (see Chapter 4 and 3).

Difficulties with Conventional Data-Modeling Methods.

The reader may have been wondering why not simply enlarge the class of primitive models to include such hybrid models so that we can deal with them by the conventional single-model paradigms for learning distribution- or manifold-like models? If this were the case, then there would be no need of developing special theory and algorithms for hybrid models and thus no need of this book! However, the most compelling reason why we *do* need hybrid models is that smooth manifolds and non-singular distributions *are not rich or flexible enough to describe the structure of many commonly observed data,* as we have seen in the examples from the previous subsection. On the one hand, the underlying topological space of a mixed data set may contain multiple manifolds of different dimensions which will probably intersect with each other, as is the case with a collection of multiple subspaces. Conventional estimation techniques for manifold-like models such as [**?**, **?**] do not apply well to mixtures of manifolds. On the other hand, if one represents a hybrid model with a probability distribution, then the distribution will typically be close to singular. Conventional statistical-learning techniques become rather ineffective in inferring such (close to) singular distributions [Vapnik, 1995].

An alternative approach to modeling mixed data is to first segment the data set into coherent subsets and then model each subset using the classical single-model methods. This is a popular approach adopted by many practitioners in the industry. The fundamental difficulty with this approach is that, without knowing which subset of sample points belongs to which constituent model, there is seemingly a "chicken-and-egg" relationship between data segmentation and model estimation: If the segmentation of the data were known, one could fit a model to each subset of samples using classical model estimation techniques; and conversely, if the constituent models were known, one could easily find the subset of samples that best fit each model. This relationship has been the rationale behind the *iterative* modeling techniques for mixed data, such as the well-known EM and K-means algorithms (see Appendix A). These iterative methods share several drawbacks:

- The iteration needs to start with a good initial guess of the solution; otherwise the iteration is likely to converge to a local minimum.
- Without knowing *a priori* the number of models and the dimension of each model, the algorithm may diverge if it starts with a wrong guess on these key parameters.
- There are cases in which it is difficult to solve the grouping problem correctly, yet it is possible to obtain a good estimate of the models. In such cases a direct estimation of the models without grouping seems more appropriate than one based on incorrectly segmented data.

Hybrid Models as Algebraic Sets.

In this book, instead of manifolds or distributions, we will represent hybrid models mainly as algebraic sets.[14] To see the merit of such a representation, let us consider a simple example where the data corresponding to the ith constituent model belong to a hyperplane of $\mathbb{R}^D$ of the form

$$Z_i = \{\boldsymbol{x} : \boldsymbol{b}_i^\top \boldsymbol{x} = 0\} \quad \text{for} \quad i = 1, \dots, n. \tag{1.9}$$

In other words, the set Z_i is the zero-level set of the polynomial $p_i(\boldsymbol{x}) = \boldsymbol{b}_i^\top \boldsymbol{x}$. Therefore, we can interpret a mixed data set drawn from a union of n hyperplanes

[14] Roughly speaking, an algebraic set is the common zero-level set of a family of algebraic equations, see Appendix B.

as the zero-level set of the polynomial $p(\boldsymbol{x}) = (\boldsymbol{b}_1^\top \boldsymbol{x})(\boldsymbol{b}_2^\top \boldsymbol{x}) \cdots (\boldsymbol{b}_n^\top \boldsymbol{x})$, i.e.,

$$Z \doteq Z_1 \cup Z_2 \cup \cdots \cup Z_n = \big\{\boldsymbol{x} : p_1(\boldsymbol{x})p_2(\boldsymbol{x}) \cdots p_n(\boldsymbol{x}) = 0\big\}. \tag{1.10}$$

This polynomial can be determined from a number of (random) sample points on the algebraic set $\boldsymbol{X} \doteq \{\boldsymbol{x}_j \in Z\}$ using techniques analogous to those used for fitting a circle to three points in $\mathbb{R}^2$. Given the polynomial $p(\boldsymbol{x})$, we can use polynomial factorization techniques to obtain the factors $p_i(\boldsymbol{x}) = \boldsymbol{b}_i^\top \boldsymbol{x}$, and hence the parameters for each constituent model, namely the vector $\boldsymbol{b}_i$ normal to the hyperplane.

This simple example of modeling the data with a union of hyperplanes can be immediately generalized to modeling the data with a union of algebraic varieties.[15] More specifically, let us suppose that the data corresponding to the ith constituent model can be described as the zero-level set of some polynomials in a prime ideal $\mathfrak{p}_i$,[16]

$$Z_i \doteq \{\boldsymbol{x} : p(\boldsymbol{x}) = 0, p \in \mathfrak{p}_i\} \ \subset \ \mathbb{R}^D, \quad i = 1, 2, \ldots, n. \tag{1.11}$$

The (mixed) data from a union of n such models then belong to an algebraic set:[17]

$$\begin{aligned} Z &\doteq Z_1 \cup Z_2 \cup \cdots \cup Z_n \\ &= \big\{\boldsymbol{x} : p_1(\boldsymbol{x})p_2(\boldsymbol{x}) \cdots p_n(\boldsymbol{x}) = 0, \ \ \forall p_i \in \mathfrak{p}_i, \ \ i = 1, 2, \ldots, n\big\}. \end{aligned} \tag{1.12}$$

From a number of (random) sample points on the algebraic set $\boldsymbol{X} \doteq \{\boldsymbol{x}_j \in Z\}$, one can determine the (radical) ideal of polynomials that vanish on the set Z:[18]

$$\boldsymbol{X} \quad \to \quad \mathfrak{q}(Z) \doteq \big\{q(\boldsymbol{x}_j) = 0, \ \ \forall \boldsymbol{x}_j \in Z\big\}. \tag{1.13}$$

Obviously, the ideal $\mathfrak{q}$ is no longer a prime ideal. Nevertheless, once the ideal $\mathfrak{q}$ is obtained, the constituent models $\mathfrak{p}_i$ (or Z_i) can be subsequently retrieved by *decomposing* the ideal $\mathfrak{q}$ into irreducible prime ideals via algebraic means.[19]

$$\mathfrak{q} \quad \to \quad \mathfrak{q} = \mathfrak{p}_1 \cap \mathfrak{p}_2 \cap \cdots \cap \mathfrak{p}_n. \tag{1.14}$$

Clearly, the above representation establishes a natural correspondence between terminologies developed in algebraic geometry and the heuristic languages used in modeling mixed data: the constituent models become algebraic varieties, the hybrid model becomes an algebraic set, the mixed data become samples from an algebraic set, and the estimation of hybrid models becomes the estimation and

[15]An algebraic variety is an irreducible algebraic set. An algebraic set is called irreducible if it cannot be written as the union of two proper algebraic subsets. A subspace is one such example.

[16]A prime ideal is an ideal that cannot be decomposed further as the intersection of two other ideals, see Appendix B. The zero-level set of a prime ideal is an irreducible algebraic set, i.e., an algebraic variety.

[17]Notice that the "union" of algebraic varieties corresponds to the "multiplication" of the polynomials associated with the varieties.

[18]According to Hilbert's Nullstellensatz (see Appendix B), there is a one-to-one correspondence between algebraic sets and radical ideals [Eisenbud, 1996].

[19]For the special case in which the ideal is generated by a single polynomial, the decomposition is equivalent to factoring the polynomial into factors.

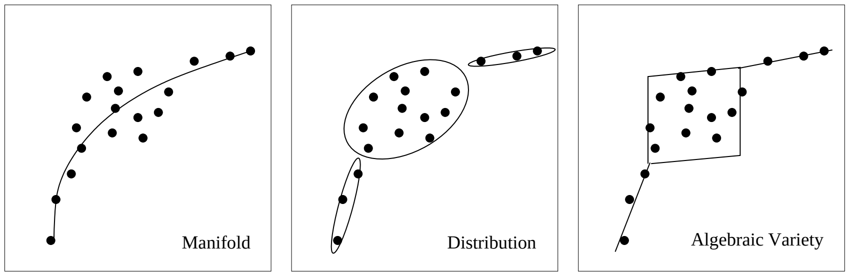


Figure 1.7. Comparison of three representations of the same data set: 1. a (nonlinear) manifold, 2. a (mixed Gaussian) distribution, or 3. a (piecewise linear) algebraic set.

decomposition of a radical ideal. Although this nomenclature may seem abstract and challenging at first, we will see in Chapter 3 how to make this very concrete for the case of an arrangement of subspaces.

Modeling Hybrid Topologies and Degenerate Distributions.

Despite its pure algebraic nature, the above algebraic representation is closely related to and complements well the two aforementioned geometric and statistical data modeling paradigms.

From the geometric viewpoint, unlike a smooth manifold M which sometimes can be implicitly represented as the level set of a single function, an algebraic set Z is the zero-level set of a *family* of polynomials. Because of that, an algebraic set Z allows components with different dimensions as well as singularities that the zero-level set of a single smooth function does not have.

From the statistical viewpoint, one can also view the irreducible components $\{Z_i\}$ of Z as the "means" of a collection of probability distributions $\{p_i(\cdot)\}$ and the overall set Z as the "skeleton" of their mixture $q(\cdot)$. For instance, a piecewise linear structure can be viewed as the skeleton of a mixture of Gaussian distributions (see Figure 7.1). Therefore, hybrid models represented by algebraic sets can be interpreted as a special class of *generative models* such that the random variables have small variance outside the algebraic sets, but large variance inside.

As we will show in this book, if the primitive models are simple models such as linear subspaces (or quadratic surfaces), then in principle, the problem of segmenting mixed data and estimating a hybrid model can be solved *non-iteratively* (see Chapter 3). Moreover, the correct number of models and their dimensions can also be correctly determined via purely algebraic means, at least in the noise-free case (see Chapter 3).

1.2.3 *Noise, Outliers, and Model Selection*

In many real-world applications, the given data samples may be corrupted with noise or contaminated with outliers. Figure 1.8 shows one such example. Unlike the noiseless or low-noise scenario, the problem of finding the "correct" model

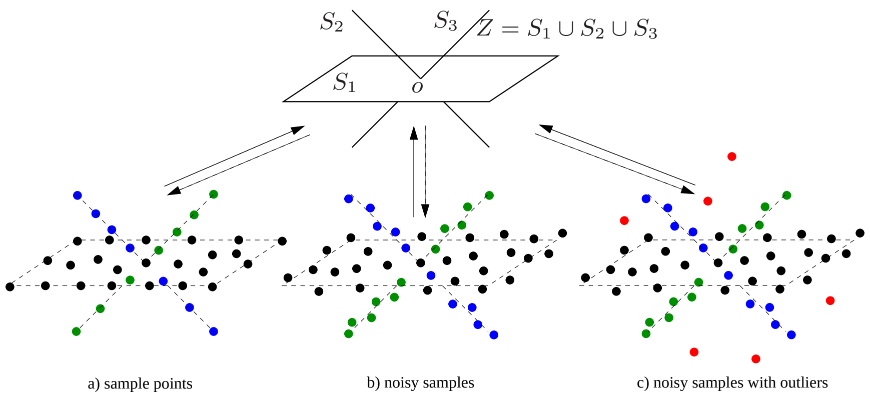


Figure 1.8. Inferring a hybrid linear model Z, consisting of one plane (S_1) and two lines (S_2, S_3), from a set of mixed data, which can be: a) noiseless samples from the plane and lines; b) noisy samples; c) noisy samples with outliers.

becomes much more challenging in presence of a significant amount of noise or outliers. Proper statistical and robust statistical techniques therefore need to be developed for the estimation and segmentation of algebraic sets such as subspace arrangements. These issues will be carefully treated in Chapter **??**.

Another important observation is that, in the presence of noise and outliers, a hybrid linear model is not necessarily the best if it has the highest fidelity to the data. This is especially the case when the number of subspaces and their dimensions are not known *a priori*. In fact, for every point in the data set, one can fit a separate line to it, which results in no modeling error at all. But such a model is not so appealing since it has exactly the same complexity as the original data.

In general, the higher the model complexity, the smaller the modeling error.[20] A good model should strike a balance between the complexity of a model M and its fidelity to the data $\boldsymbol{X}$.[21] Many general model selection criteria have been proposed in the statistics or machine learning literature, including the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), the Minimum Description Length (MDL), and the Minimum Message Length (MML). (See Appendix A for a brief review.) Despite some small differences, these criteria all tradeoff modeling error for model complexity and minimize an objective of the following form:

$$\min_{M \in \mathcal{M}} \quad J(M) \doteq [\alpha \cdot \text{Complexity}(M) + \beta \cdot \text{Error}(\boldsymbol{X}, M)].$$

[20]For example, any function can be approximated arbitrarily well by a piecewise linear function with a sufficient number of pieces.

[21]For instance, the complexity of a model can be measured as the minimum number of bits needed to fully describe the model and the data fidelity can be measured by the distance from the sample points to the model.

In this book, we will introduce a model complexity measure that is specially designed for an arrangement of linear subspaces of arbitrary dimensions, namely the *effective dimension* (see Chapter **??**).

There is yet another fundamental tradeoff that is often exploited for model selection. When the model complexity is too high, the model tends to over-fit the given data, including the noise in it. Such a model does not generalize well in the sense that it would not predict well the outcome of new samples. When the model complexity is too low, the model under-fits the data, which results in a large error in the prediction. Therefore, a good model should minimize the prediction error.

YM: BUG IN THE FIGURE. If the alpha and beta are the same the line should be tangent to the Prediction Error Curve

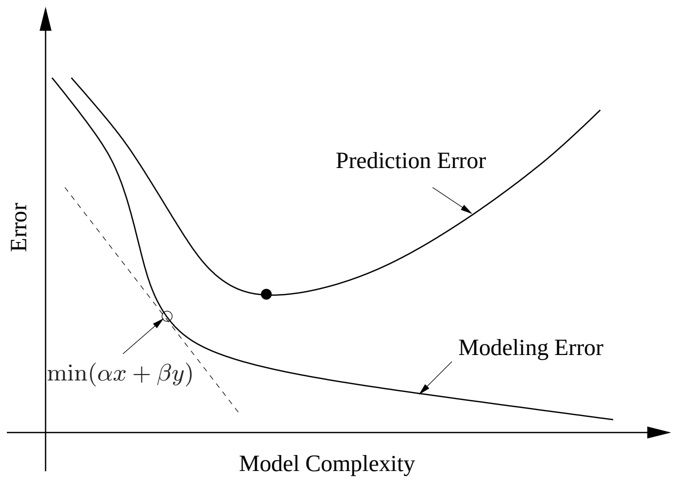


Figure 1.9. Modeling and prediction error versus model complexity.

The relationship between modeling error and prediction error as a function of model complexity is plotted in Figure 1.9. Unfortunately, the "optimal" models obtained by trading off modeling error and prediction error can be different, as illustrated in the figure. In such a case, a choice between the two objectives has to be made. In the unsupervised learning setting, it is often difficult to obtain the prediction error curve;[22] and for purposes such as data compression, the prediction error is of less concern than the modeling error. In these cases, we often choose the tradeoff between the modeling error and the model complexity (see Chapter **??**).

[22]Unless one does cross-validation within the given data set itself.

Part I

Theory, Analysis, and Algorithms

Chapter 2
Data Modeling with a Single Subspace

"Principal component analysis is probably the oldest and best known of the techniques of multivariate analysis."
– I. T. Jolliffe

In this chapter, we give a brief review of principal component analysis (PCA), i.e., the method for finding an optimal (affine) subspace to fit a set of data points. The solution to PCA has been well established in the literature and it has become one of the most useful tools for data modeling, compression, and visualization. We introduce both the statistical and geometric formulation of PCA and establish their equivalence. Specifically, we show that the singular value decomposition (SVD) provides an optimal solution to PCA. We also establish the similarities and differences between PCA and two generative subspace models, namely Factor Analysis (FA) and Probabilistic PCA (PPCA). When the dimension of the subspace is unknown, we introduce some conventional model selection methods to determine the number of principal components. When the data points are incomplete or contain outliers, we review some robust statistical techniques that help resolve these difficulties. Finally, some nonlinear extensions to PCA such as nonlinear PCA and kernel PCA are also reviewed.

2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) refers to the problem of fitting a low-dimensional affine subspace S to a set of points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ in a high-dimensional space $\mathbb{R}^D$, the ambient space. Mathematically, this problem can be formulated as either a statistical problem or a geometric one, and they both lead to the same solution, as we will show in this section.

2.1.1 *A Statistical View of PCA*

Historically, PCA was first formulated in a statistical setting to estimate the principal components of a multivariate random variable $\boldsymbol{x}$ [Pearson, 1901, Hotelling, 1933]. Specifically, given a multivariate random variable $\boldsymbol{x} \in \mathbb{R}^D$ and any integer $d < D$, the d "principal components" of $\boldsymbol{x}$ are defined as the d *uncorrelated* linear components of $\boldsymbol{x}$:

$$y_i = u_i^\top \boldsymbol{x} \ \in \mathbb{R}, \quad u_i \in \mathbb{R}^D, \quad i = 1, 2, \ldots, d, \tag{2.1}$$

such that the variance of y_i is maximized subject to

$$u_i^\top u_i = 1 \quad \text{and} \quad \text{Var}(y_1) \geq \text{Var}(y_2) \geq \cdots \geq \text{Var}(y_d). \tag{2.2}$$

For example, to find the first principal component, y_1, we seek a vector $u_1^* \in \mathbb{R}^D$ such that

$$u_1^* = \arg\max_{u_1 \in \mathbb{R}^D} \text{Var}(u_1^\top \boldsymbol{x}), \quad \text{s.t.} \quad u_1^\top u_1 = 1. \tag{2.3}$$

Without loss of generality, in what follows, we will assume $\boldsymbol{x}$ has zero-mean.

Theorem 2.1 (Principal Components of a Random Variable)**.** *The first d principal components of a multivariate random variable $\boldsymbol{x}$ are given by $y_i = u_i^\top \boldsymbol{x}$, where $\{u_i\}_{i=1}^d$ are the d leading eigenvectors of its covariance matrix $\Sigma_{\boldsymbol{x}} \doteq \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\top]$.*

Proof. Notice that for any $u \in \mathbb{R}^D$,

$$\text{Var}(u^\top \boldsymbol{x}) = \mathbb{E}[(u^\top \boldsymbol{x})^2] = \mathbb{E}[u^\top \boldsymbol{x}\boldsymbol{x}^\top u] = u^\top \Sigma_{\boldsymbol{x}} u. \tag{2.4}$$

Therefore, the optimization in problem in (2.3) for finding the first principal component is equivalent to

$$\max_{u_1 \in \mathbb{R}^D} u_1^\top \Sigma_{\boldsymbol{x}} u_1, \quad \text{s.t.} \quad u_1^\top u_1 = 1. \tag{2.5}$$

In order to solve the above constrained minimization problem, we use the Lagrange multiplier method. The Lagrangian is given by

$$\mathcal{L} = u_1^\top \Sigma_{\boldsymbol{x}} u_1 + \lambda(1 - u_1^\top u_1) \tag{2.6}$$

for some Lagrange multiplier $\lambda \in \mathbb{R}$. The necessary condition for u_1 to be an extrema is

$$\Sigma_{\boldsymbol{x}} u_1 = \lambda u_1, \tag{2.7}$$

and the associated extremum value is $u_1^\top \Sigma_{\boldsymbol{x}} u_1 = \lambda$. It follows that the optimal solution u_1^* is exactly the eigenvector of $\Sigma_{\boldsymbol{x}}$ associated with the largest eigenvalue.

To find the remaining principal components, since $u_1^\top \boldsymbol{x}$ and $u_i^\top \boldsymbol{x}$ $(i > 1)$ need to be uncorrelated, we have

$$\mathbb{E}[(u_1^\top \boldsymbol{x})(u_i^\top \boldsymbol{x})] = \mathbb{E}[u_1^\top \boldsymbol{x}\boldsymbol{x}^\top u_i] = u_1^\top \Sigma_{\boldsymbol{x}} u_i = \lambda_1 u_1^\top u_i = 0. \tag{2.8}$$

That is, $u_2, \ldots, u_d$ are all orthogonal to u_1. More generally, $u_i^\top u_j = 0$ for all $i \neq j = 1, \ldots d$. To find u_2 we define the Lagrangian

$$\mathcal{L} = u_2^\top \Sigma_{\boldsymbol{x}} u_2 + \lambda_2(1 - u_2^\top u_2) + \gamma u_1^\top u_2. \tag{2.9}$$

The necessary condition for u_2 to be an extrema is

$$\Sigma_{\boldsymbol{x}} u_2 + \gamma u_1 = \lambda_2 u_2, \tag{2.10}$$

from which it follows that $u_1^\top \Sigma_{\boldsymbol{x}} u_2 + \gamma u_1^\top u_1 = \lambda_1 u_1^\top u_2 + \gamma = \lambda_2 u_1^\top u_2$, and so $\gamma = 0$. Since the associated extremum value is $u_2^\top \Sigma_{\boldsymbol{x}} u_2 = \lambda_2$, u_2^* is the leading eigenvector of $\Sigma_{\boldsymbol{x}}$ restricted to the orthogonal complement of u_1.[1] Assuming that $\Sigma_{\boldsymbol{x}}$ does not have repeated eigenvalues, u_2^* is the eigenvector of $\Sigma_{\boldsymbol{x}}$ associated with the second largest eigenvalue. Inductively, one can show that u_3, u_4, ..., u_d are the top third, fourth, ..., dth eigenvectors of $\Sigma_{\boldsymbol{x}}$ and that the corresponding eigenvalues give the variance of the principal components, i.e., $\lambda_i = \text{Var}(y_i)$. $\square$

The solution to PCA provided by Theorem 2.1 suggests that we may find the d principal components of $\boldsymbol{x}$ simultaneously, rather than one by one. Specifically, we can define a matrix a random vector $\boldsymbol{y} = [y_1, y_2, \ldots, y_d]^\top \in \mathbb{R}^d$ and a matrix $U_d = [u_1, u_2, \cdots, u_d] \in \mathbb{R}^{D \times d}$. Since $\boldsymbol{y} = U_d^\top \boldsymbol{x}$, we have that

$$\Sigma_{\boldsymbol{y}} = \mathbb{E}(\boldsymbol{y}\boldsymbol{y}^\top) = U_d^\top \mathbb{E}(\boldsymbol{x}\boldsymbol{x}^\top) U_d = U_d^\top \Sigma_{\boldsymbol{x}} U_d. \tag{2.11}$$

Since were are looking for uncorrelated random variables, the matrix Σ_y must be diagonal and the matrix U_d must be orthonormal, i.e., $U_d^\top U_d = I_d$.

Recall that any real, symmetric and positive semi-definite matrix A can be transformed into a diagonal matrix $\Lambda = V^{-1} A V$, where the columns of V are the eigenvectors of A and the diagonal entries of Λ are the corresponding eigenvalues. Recall also that the eigenvalues are real and nonnegative, i.e., $\lambda_i \geq 0$, and that the eigenvectors can be chosen to be orthonormal, so that $V^{-1} = V^\top$. Since the matrix $\Sigma_{\boldsymbol{x}}$ is real, symmetric and positive semi-definite, the equation $\Sigma_{\boldsymbol{y}} = U_d^\top \Sigma_{\boldsymbol{x}} U_d$ suggests that the columns of U_d can be chosen as d eigenvectors of $\Sigma_{\boldsymbol{x}}$ and that the diagonal entries of $\Sigma_{\boldsymbol{y}}$ can be chosen as the corresponding d eigenvalues. Moreover, since our goal is to maximize the variance of each y_i and $\lambda_i = \text{Var}(y_i)$, we conclude that the columns of U_d are the top d eigenvectors of $\Sigma_{\boldsymbol{x}}$ and the entries of $\Sigma_{\boldsymbol{y}}$ are the corresponding top d eigenvalues.

This alternative derivation of PCA allows us to understand what happens when $\Sigma_{\boldsymbol{x}}$ has repeated eigenvalues. When the eigenvalues are different, each eigenvector u_i is unique (up to sign), thus the principal components are unique (up to sign).

[1]The reason for this is that both u_1 and its orthogonal complement $u_1^\perp$ are invariant subspaces of $\Sigma_{\boldsymbol{x}}$.

When an eigenvalue is repeated, $\Sigma_{\boldsymbol{x}}$ still admits a basis of orthonormal eigenvectors. However, the eigenvectors corresponding to the repeated eigenvalue form an eigensubspace and any orthonormal basis for this eigensubspace gives valid principal components. As a consequence, the principal components are not always uniquely defined.

In practice, we may not know the population covariance matrix, $\Sigma_{\boldsymbol{x}}$. Instead, we may be given N i.i.d. samples of $\boldsymbol{x}$, $\{\boldsymbol{x}_i\}_{i=1}^N$. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$ be the sample data matrix. It is well known from statistics that an asymptotically unbiased estimate of $\Sigma_{\boldsymbol{x}}$ is given by

$$\widehat{\Sigma}_N \doteq \frac{1}{N-1}\sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^\top = \frac{1}{N-1}\boldsymbol{X}\boldsymbol{X}^\top. \tag{2.12}$$

We define the d “sample principal components” of $\boldsymbol{x}$ as

$$\hat{y}_i = \hat{u}_i^\top \boldsymbol{x}, \quad i = 1, \ldots, d, \tag{2.13}$$

where $\{\hat{u}_i\}_{i=1}^d$ are the top d eigenvectors of $\widehat{\Sigma}_N$, or equivalently those of $\boldsymbol{X}\boldsymbol{X}^\top$. Notice also that, even though the principal components of $\boldsymbol{x}$ and the sample principal components of $\boldsymbol{x}$ are different notions, under certain assumptions on the distribution of $\boldsymbol{x}$ they can be related to each other. Specifically, one can show that, if $\boldsymbol{x}$ is Gaussian, then every eigenvector $\hat{u}$ of $\widehat{\Sigma}_N$ is an asymptotically unbiased estimate for the corresponding eigenvector u of $\Sigma_{\boldsymbol{x}}$ [Jolliffe, 1986].

2.1.2 A Geometric View of PCA

An alternative geometric view of PCA, which is very much related to the SVD [Beltrami, 1873, Jordan, 1874], seeks to find an (affine) subspace S that fits the given data points $\{\boldsymbol{x}_i\}_{i=1}^N$.

Let us assume for now that the dimension of the subspace d is known. Then every point $\boldsymbol{x}_i$ on a d-dimensional affine subspace in $\mathbb{R}^D$ can be represented as

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + U_d \boldsymbol{y}_i, \quad i = 1, 2, \ldots, N \tag{2.14}$$

where $\boldsymbol{x}_0 \in \mathbb{R}^D$ is a(ny) fixed point in the subspace, U_d is a $D \times d$ matrix whose columns form a basis for the subspace, and $\boldsymbol{y}_i \in \mathbb{R}^d$ is simply the vector of new coordinates of $\boldsymbol{x}_i$ in the subspace.

Notice that there is some redundancy in the above representation due to the arbitrariness in the choice of $\boldsymbol{x}_0$ and U_d. More precisely, for any $\boldsymbol{y}_0 \in \mathbb{R}^d$, we can re-represent $\boldsymbol{x}_i$ as $\boldsymbol{x}_i = (\boldsymbol{x}_0 + U_d \boldsymbol{y}_0) + U_d(\boldsymbol{y}_i - \boldsymbol{y}_0)$. We call this ambiguity the *translational ambiguity*. Also, for any $A \in \mathbb{R}^{d \times d}$ we can re-represent $\boldsymbol{x}_i$ as $\boldsymbol{x}_i = \boldsymbol{x}_0 + (U_d A)(A^{-1}\boldsymbol{y}_i)$. We call this ambiguity the *change of basis ambiguity*. Therefore, we need some additional constraints in order to end up with a unique solution to the problem of finding an affine subspace to for the data.

A common constraint used to resolve the translational ambiguity is to impose that the mean of $\boldsymbol{y}_i$ is zero:[2]

$$\bar{\boldsymbol{y}} \doteq \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{y}_i = \mathbf{0}, \tag{2.15}$$

while a common constraint used to resolve the change of basis ambiguity is to impose that the columns of U_d be orthonormal. This last constraint eliminates the change of basis ambiguity only up to a rotation, because we can still re-represent $\boldsymbol{x}_i$ as $\boldsymbol{x}_i = \boldsymbol{x}_0 + (U_d R)(R^\top \boldsymbol{y}_i)$ for some rotation R in $\mathbb{R}^d$. However, this *rotational ambiguity* can be easily deal with during optimization, as we shall see.

In general the given points are imperfect and have noise. We define the "optimal" affine subspace to be the one that minimizes the sum of squared distances between $\boldsymbol{x}_i$ and its projection onto the subspace $\boldsymbol{x}_0 + U_d\boldsymbol{y}_i$, i.e.,

$$\min_{\boldsymbol{x}_0, U_d, \{\boldsymbol{y}_i\}} \sum_{i=1}^{N}\left\|\boldsymbol{x}_i - \boldsymbol{x}_0 - U_d\boldsymbol{y}_i\right\|^2, \quad \text{s.t.} \;\; U_d^\top U_d = I_d \text{ and } \bar{\boldsymbol{y}} = \mathbf{0}. \tag{2.16}$$

In order to solve this optimization problem, we define the Lagrangian

$$\mathcal{L} = \sum_{i=1}^{N}\left\|\boldsymbol{x}_i - \boldsymbol{x}_0 - U_d\boldsymbol{y}_i\right\|^2 + \gamma^\top \sum_{i=1}^{N}\boldsymbol{y}_i + \mathbf{tr}(\Lambda(I_d - U_d^\top U_d)), \tag{2.17}$$

where $\gamma \in \mathbb{R}^d$ and $\Lambda = \Lambda^\top \in \mathbb{R}^{d\times d}$ are, respectively, a vector and a matrix of Lagrange multipliers.

The necessary condition for $\boldsymbol{x}_0$ to be an extrema is

$$-2\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{x}_0 - U_d\boldsymbol{y}_i) = \mathbf{0} \implies \hat{\boldsymbol{x}}_0 = \bar{\boldsymbol{x}} \doteq \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i. \tag{2.18}$$

The necessary condition for $\boldsymbol{y}_i$ to be an extrema is

$$-2U_d^\top(\boldsymbol{x}_i - \boldsymbol{x}_0 - U_d\boldsymbol{y}_i) + \gamma = \mathbf{0}. \tag{2.19}$$

Summing over i yields $\gamma = 0$, from which we obtain

$$\hat{\boldsymbol{y}}_i = U_d^\top(\boldsymbol{x}_i - \bar{\boldsymbol{x}}). \tag{2.20}$$

The vector $\hat{\boldsymbol{y}}_i \in \mathbb{R}^d$ is simply the coordinates of the projection of $\boldsymbol{x}_i \in \mathbb{R}^D$ onto the subspace S. We may call such $\hat{\boldsymbol{y}}$ the "geometric principal components" of $\boldsymbol{x}$.[3]

[2]In the statistical setting, $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ will be samples of two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Then this constraint is equivalent to setting their means to be zero.

[3]As we will soon see in the next section, the geometric principal components coincide with the sample principal components defined in a statistical sense.

Before optimizing over U_d, we can replace the optimal values for $\boldsymbol{x}_0$ and $\boldsymbol{y}_i$ into the objective function. This leads to the following optimization problem

$$\min_{U_d} \sum_{i=1}^{N} \left\| (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) - U_d U_d^\top (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \right\|^2 \quad \text{s.t.} \quad U_d^\top U_d = I_d. \tag{2.21}$$

Note that this is a restatement of the original problem with the mean $\bar{\boldsymbol{x}}$ subtracted from each of the sample points. Therefore, from now on, we will consider only the case in which the data points have zero mean. If not, simply subtract the mean from each point before computing U_d.

The following theorem gives a constructive solution for finding an optimal $\hat{U}_d$.

Theorem 2.2 (PCA via SVD)**.** *Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ be the matrix formed by stacking the (zero-mean) data points as its column vectors. Let $\boldsymbol{X} = U\Sigma V^\top$ be the SVD of the matrix $\boldsymbol{X}$. Then for any given $d < D$, an optimal solution for U_d is given by the first d columns of U, and an optimal solution for $\boldsymbol{y}_i$ is given by the ith column of the top $d \times N$ submatrix $\Sigma_d V_d^\top$ of $\Sigma V^\top$.*

Proof. Recalling that $\boldsymbol{x}^\top A \boldsymbol{x} = \mathbf{tr}(A\boldsymbol{x}\boldsymbol{x}^\top)$, we can rewrite the least-squares error

$$\sum_{i=1}^{N} \left\| \boldsymbol{x}_i - U_d U_d^\top \boldsymbol{x}_i \right\|^2 = \sum_{i=1}^{N} \boldsymbol{x}_i^\top (I_D - U_d U_d^\top) \boldsymbol{x}_i \tag{2.22}$$

as $\mathbf{tr}((I_D - U_d U_d^\top)\boldsymbol{X}\boldsymbol{X}^\top)$. The first term $\mathbf{tr}(\boldsymbol{X}\boldsymbol{X}^\top)$ does not depend on U_d. Therefore, we can transform the minimization of (2.22) to

$$\max_{U_d} \mathbf{tr}(U_d U_d^\top \boldsymbol{X}\boldsymbol{X}^\top) \quad \text{s.t.} \quad U_d^\top U_d = I_d. \tag{2.23}$$

Since $\mathbf{tr}(AB) = \mathbf{tr}(BA)$, the Lagrangian for this problem can be written as

$$\mathcal{L} = \mathbf{tr}(U_d^\top \boldsymbol{X}\boldsymbol{X}^\top U_d) + \mathbf{tr}((I_d - U_d^\top U_d)\Lambda). \tag{2.24}$$

The conditions for an extrema are given by

$$\boldsymbol{X}\boldsymbol{X}^\top U_d = U_d \Lambda. \tag{2.25}$$

Therefore, $\Lambda = U_d^\top \boldsymbol{X}\boldsymbol{X}^\top U_d$ and the objective function reduces to $\mathbf{tr}(\Lambda)$. Now, recall that U_d is defined only up to a rotation, i.e., $U_d' = U_d R$ is also a valid solution, hence so is $\Lambda' = R\Lambda R^\top$. Since Λ is symmetric, it has an orthogonal matrix of eigenvectors. Thus, if we choose R to be the matrix of eigenvectors of Λ, then Λ' is a diagonal matrix. As a consequence, we can choose Λ to be diagonal without loss of generality. It follows from (2.25) that the columns of U_d must be eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\top$ with the corresponding eigenvalues in the diagonal entries of Λ. Since the goal is to maximize $\mathbf{tr}(\Lambda)$, an optimal solution is given by the top d eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\top$, i.e., the top d singular vectors of $\boldsymbol{X} = U\Sigma V^\top$, which are the first d columns of U. It then follows from (2.20) that $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_N] = U_d^\top \boldsymbol{X} = U_d^\top U\Sigma V^\top = \Sigma_d V_d^\top$. Finally, since $\Lambda = U_d^\top U \Sigma^2 U^\top U_d = \Sigma_d^2$, the optimal least-squares error is given by $\mathbf{tr}(\Sigma^2) - \mathbf{tr}(\Sigma_d^2) = \sum_{i=d+1}^{D} \sigma_i^2$, where σ_i is the ith singular value of $\boldsymbol{X}$. □

According to the theorem, the SVD gives an optimal solution to the PCA problem. The resulting matrix $\hat{U}_d$ (together with the mean $\bar{\boldsymbol{x}}$ if the data is not zero-mean) provides a geometric description of the dominant subspace structure for all the points[4]; and the columns of the matrix $\Sigma_d V_d^\top = [\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_N] \in \mathbb{R}^{d \times N}$, i.e., the principal components, give a more compact representation for the points $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$, as d is typically much smaller than D.

Theorem 2.3 (Equivalence of Geometric and Sample Principal Components)**.** *Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ be the data matrix (with $\bar{\boldsymbol{x}} = 0$). The vectors $\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_d \in \mathbb{R}^D$ associated with the d sample principal components for $\boldsymbol{X}$ are exactly the columns of the matrix $\hat{U}_d \in \mathbb{R}^{D \times d}$ that minimizes the least-squares error (2.22).*

Proof. The proof is simple. Notice that if $\boldsymbol{X}$ has the singular value decomposition $\boldsymbol{X} = U\Sigma V^\top$, then $\boldsymbol{X}\boldsymbol{X}^\top = U\Sigma^2 U^\top$ is the eigenvalue decomposition of $\boldsymbol{X}\boldsymbol{X}^\top$. If Σ is ordered, then the first d columns of U are exactly the leading d eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\top$, which give the d sample principal components. □

Therefore, both the geometric and statistical formulation of PCA lead to exactly the same solutions/estimates of the principal components. The geometric formulation allows us to apply PCA to data even if the statistical nature of the data is unclear; the statistical formulation allows to quantitatively evaluate the quality of the estimates. For instance, for Gaussian random variables, one can derive explicit formulae for the mean and covariance of the estimated principal components. For a more thorough analysis of the statistical properties of PCA, we refer the reader to the classical book [Jolliffe, 1986].

2.1.3 Probabilistic PCA

The PCA model described so far allows us to find a low-dimensional representation $\{\boldsymbol{y}_i \in \mathbb{R}^d\}$ of a set of points $\{\boldsymbol{x}_i \in \mathbb{R}^D\}$, with $d \ll D$. However, the PCA model is not a proper generative model, because the low-dimensional representation $\boldsymbol{y}$ and the error ε are treated as parameters, rather than as random variables. As a consequence, the PCA model cannot be used to generate new samples $\boldsymbol{x}$.

To address this issue, assume that the low-dimensional representation $\boldsymbol{y}$ and the error ε are independent random variables with pdfs $p(\boldsymbol{y})$ and $p(\varepsilon)$, respectively. This allows us to generate a new sample of $\boldsymbol{x}$ from samples of $\boldsymbol{y}$ and ε as

$$\boldsymbol{x} = \boldsymbol{x}_0 + U_d \boldsymbol{y} + \varepsilon. \tag{2.26}$$

Assume that mean and covariance of $\boldsymbol{y}$ are denoted as $\mu_{\boldsymbol{y}}$ and $\Sigma_{\boldsymbol{y}}$, respectively. Assume also that ε is zero mean with covariance Σ_ε. The mean and covariance of

[4]From a statistical standpoint, the column vectors of U_d give the directions in which the data X has the largest variance, hence the name "principal components." See the next section for detail.

the observations are then given by

$$\mu_{\boldsymbol{x}} = \boldsymbol{x}_0 + U_d \mu_{\boldsymbol{y}} \quad \text{and} \quad \Sigma_{\boldsymbol{x}} = U_d \Sigma_{\boldsymbol{y}} U_d^\top + \Sigma_\varepsilon. \tag{2.27}$$

The remainder of the section discusses different methods for estimating the parameters of this model, $\boldsymbol{x}_0$, U_d, μ_y, $\Sigma_{\boldsymbol{y}}$ and Σ_ε, from the mean and covariance of the population, $\mu_{\boldsymbol{x}}$ and $\Sigma_{\boldsymbol{x}}$, or from i.i.d. samples $\{\boldsymbol{x}_i\}_{i=1}^N$.

PPCA from Population Mean and Covariance

Observe that, in general, we cannot recover model parameters from $\mu_{\boldsymbol{x}}$ and $\Sigma_{\boldsymbol{x}}$. For instance, notice that $\boldsymbol{x}_0$ and $\mu_{\boldsymbol{y}}$ cannot be uniquely recovered from $\mu_{\boldsymbol{x}}$. Similarly to what we did in the case of PCA, this issue can be easily resolved by assuming that $\mu_{\boldsymbol{y}} = \mathbf{0}$. This leads to the following estimate of $\boldsymbol{x}_0$

$$\widehat{\boldsymbol{x}}_0 = \mu_{\boldsymbol{x}}, \tag{2.28}$$

which is the same estimate as that of PCA.

Another ambiguity that cannot be resolved in a straightforward manner is that $\Sigma_{\boldsymbol{y}}$ and Σ_ε cannot be uniquely recovered from $\Sigma_{\boldsymbol{x}}$. For instance, $\Sigma_{\boldsymbol{y}} = 0$ and $\Sigma_\varepsilon = \Sigma_{\boldsymbol{x}}$ is a valid solution. However, this solution is not meaningful, because it assigns all the information in $\Sigma_{\boldsymbol{x}}$ to the error, rather than to the low-dimensional representation. Intuitively we would like $\Sigma_{\boldsymbol{y}}$ to capture as much information about $\Sigma_{\boldsymbol{x}}$ as possible. Thus it makes sense for $\Sigma_{\boldsymbol{y}}$ to be full rank and for Σ_ε to be as close to zero as possible. Probabilistic PCA (PPCA) resolves the aforementioned ambiguity by assuming that

1. the low-dimensional representation has unit covariance $\Sigma_{\boldsymbol{y}} = I_d \in \mathbb{R}^{d\times d}$ and
2. the noise covariance matrix $\Sigma_\varepsilon \in \mathbb{R}^{D\times D}$ is isotropic, i.e., $\Sigma_\varepsilon = \sigma^2 I_D$.

These assumptions lead to the following relationship

$$\Sigma_{\boldsymbol{x}} = U_d U_d^\top + \sigma^2 I_D. \tag{2.29}$$

The following theorem allows us to compute the parameters U_d and σ.

Theorem 2.4. *The optimal solution for U_d and σ with the smallest σ is given by*

$$\widehat{U}_d = U_1(\Sigma_1 - \widehat{\sigma}^2 I)^{1/2} \quad \textit{and} \quad \widehat{\sigma}^2 = \frac{1}{D-d}\sum_{i=d+1}^{D} \lambda_i, \tag{2.30}$$

where U_1 is the matrix with the top d eigenvectors of $\Sigma_{\boldsymbol{x}}$, Σ_1 is the matrix with the corresponding d top eigenvalues, and λ_i is the ith eigenvalue of $\Sigma_{\boldsymbol{x}}$.

Proof. Multiplying (2.29) on the right by U_d leads to

$$(\Sigma_{\boldsymbol{x}} - \sigma^2 I_D)U_d = U_d \Lambda. \tag{2.31}$$

Therefore, the columns of U_d must be eigenvectors of $\Sigma_{\boldsymbol{x}} - \sigma^2 I_D$, which are the same as the eigenvectors of $\Sigma_{\boldsymbol{x}}$. Since we want σ to be as small as possible, it makes sense to choose the top d eigenvectors of $\Sigma_{\boldsymbol{x}}$. So see this, let $U_d = U_1\Gamma$, where the columns of $U_1 \in \mathbb{R}^{D \times d}$ are *any* d orthonormal eigenvectors of $\Sigma_{\boldsymbol{x}}$ and $\Gamma \in \mathbb{R}^{d \times d}$ is a diagonal matrix, which scales these eigenvectors so that they satisfy $U_d^\top U_d = \Lambda$. Since $U_1^\top U_1 = I_d$, we obtain $\Gamma^2 = \Lambda = \Sigma_1 - \sigma^2 I_d$, where Σ_1 is a diagonal matrix with the d eigenvalues of $\Sigma_{\boldsymbol{x}}$ corresponding to the d eigenvectors in U_1. Now, recalling that $\Sigma_{\boldsymbol{x}} = U_d U_d^\top + \sigma^2 I_D$ we have that

$$\begin{aligned} \mathbf{tr}(\Sigma_{\boldsymbol{x}}) &= \mathbf{tr}(U_d U_d^\top) + \mathbf{tr}(\sigma^2 I_D) = \mathbf{tr}(U_d^\top U_d) + D\sigma^2 && (2.32) \\ &= \mathbf{tr}(\Lambda) + D\sigma^2 = \mathbf{tr}(\Sigma_1) + (D-d)\sigma^2. && (2.33) \end{aligned}$$

Therefore, the smallest possible σ is obtained when $\mathbf{tr}(\Sigma_1)$ is maximized, which happens if we choose the diagonal entries of Σ_1 to be the top d eigenvalues of $\Sigma_{\boldsymbol{x}}$. □

PPCA by Maximum Likelihood

In general, we may not know the true covariance matrix $\Sigma_{\boldsymbol{x}}$. Instead, we are given samples $\{\boldsymbol{x}_i\}_{i=1}^N$ from which we can estimate the sample covariance matrix $\widehat{\Sigma}_N$. The question is whether the model parameters can be estimated as in the previous section after replacing $\Sigma_{\boldsymbol{x}}$ by $\widehat{\Sigma}_N$. As it turns out, the maximum likelihood estimates of the model parameters can be computed almost as before when $\boldsymbol{y}$ and ε are assumed to be Gaussian random variables.

More specifically, assume that both $\boldsymbol{y}$ and ε are Gaussian random variables $\boldsymbol{y} \sim \mathcal{N}(\mu_{\boldsymbol{y}}, \Sigma_{\boldsymbol{y}})$ and $\varepsilon \sim \mathcal{N}(\mathbf{0}, \Sigma_\varepsilon)$. This implies that $\boldsymbol{x}$ is also Gaussian, because it is a linear combination of Gaussians. Specifically, $\boldsymbol{x} \sim \mathcal{N}(\mu_{\boldsymbol{x}}, \Sigma_{\boldsymbol{x}})$, where $\mu_{\boldsymbol{x}}$ and $\Sigma_{\boldsymbol{x}}$ are given in (2.27). Assume also that $\Sigma_{\boldsymbol{y}} = I_d$ and that $\Sigma_\varepsilon = \sigma^2 I$. The maximum likelihood estimate for $\mu_{\boldsymbol{x}}$ is $\frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i$. The maximum likelihood estimates for U_d and Σ_ε are obtained by maximizing

$$\mathcal{L}(U_d, \Sigma_\varepsilon) = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det(\Sigma_{\boldsymbol{x}}) - \frac{N}{2}\mathbf{tr}(\Sigma_{\boldsymbol{x}}^{-1}\widehat{\Sigma}_{\boldsymbol{x}}) \qquad (2.34)$$

subject to $\Sigma_{\boldsymbol{x}} = U_d U_d^\top + \Sigma_\varepsilon$.

After taking derivatives with respect to U_d, we obtain

$$\frac{\partial \mathcal{L}}{\partial U_d} = -N\Sigma_{\boldsymbol{x}}^{-1}U_d + N\Sigma_{\boldsymbol{x}}^{-1}\widehat{\Sigma}_{\boldsymbol{x}}\Sigma_{\boldsymbol{x}}^{-1}U_d = 0 \implies \widehat{\Sigma}_{\boldsymbol{x}}\Sigma_{\boldsymbol{x}}^{-1}U_d = U_d. \qquad (2.35)$$

One possible solution is $U_d = 0$, which leads to a minimum of the log-likelihood and violates our assumption that U_d should be full rank. Another possible solution is $\Sigma_{\boldsymbol{x}} = \widehat{\Sigma}_{\boldsymbol{x}}$, where the covariance model is exact. This corresponds to the case discussed in the previous section, after replacing $\Sigma_{\boldsymbol{x}}$ by $\widehat{\Sigma}_{\boldsymbol{x}}$. Thus, the model parameters can be computed as before. A third solution is obtained when $U_d \neq 0$ and $\Sigma_{\boldsymbol{x}} \neq \widehat{\Sigma}_{\boldsymbol{x}}$. In this case, we have,

$$\begin{aligned} \Sigma_{\boldsymbol{x}} U_d = U_d(\Lambda + \sigma^2 U_d) &\implies U_d = \Sigma_{\boldsymbol{x}}^{-1} U_d(\Lambda + \sigma^2 I_d) && (2.36) \\ &\implies \widehat{\Sigma}_{\boldsymbol{x}} U_d = U_d(\Lambda + \sigma^2 I_d) && (2.37) \end{aligned}$$

Notice that the last equation is the same as that in (2.31) with $\Sigma_{\boldsymbol{x}}$ replaced by $\widehat{\Sigma}_{\boldsymbol{x}}$. Therefore, the optimal solution is of the form $U_d = U_1(\Sigma_1 - \sigma^2 I)^{1/2}$, where U_1 is a matrix with d eigenvectors of $\widehat{\Sigma}_{\boldsymbol{x}}$ with the corresponding eigenvalues in Σ_1.

Before replacing this solution into (2.34), recall two well known identities, the matrix determinant lemma $\det(A + UV^\top) = \det(I + V^\top A^{-1} U)\det(A)$ and the matrix inversion lemma $(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$. Applying the matrix determinant lemma to $\det(\Sigma_{\boldsymbol{x}})$ leads to

$$|U_d U_d^\top + \sigma^2 I_D| = |I_d + \sigma^{-2} U_d^\top U_d)||\sigma^2 I_D| = |(\Sigma_1/\sigma^2)|\sigma^{2D} = |\Sigma_1|\sigma^{2(D-d)}, \tag{2.38}$$

while applying the matrix inversion lemma to $\Sigma_{\boldsymbol{x}}$ leads to

$$(U_d U_d^\top + \sigma^2 I_D)^{-1} = \frac{I_D}{\sigma^2} - \frac{U_d}{\sigma^2}(I_d + \frac{1}{\sigma^2}U_d^\top U_d)^{-1}\frac{U_d^\top}{\sigma^2} \tag{2.39}$$

$$= \frac{1}{\sigma^2}(I_D - U_d \Lambda^{-1} U_d^\top) = \frac{1}{\sigma^2}(I_D - U_1 U_1^\top) \tag{2.40}$$

Therefore, the log-likelihood can be rewritten as

$$\mathcal{L} = -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\big((D-d)\log\sigma^2 + \log\det(\Sigma_1)\big) \tag{2.41}$$

$$- \frac{N}{2\sigma^2}\,\mathbf{tr}(\widehat{\Sigma}_{\boldsymbol{x}} - U_1 U_1^\top \widehat{\Sigma}_{\boldsymbol{x}}) \tag{2.42}$$

The condition for an extrema in σ^2 is given by

$$\frac{\partial \mathcal{L}}{\partial \sigma^2} = -\frac{N}{2}\frac{D-d}{\sigma^2} + \frac{N}{2\sigma^4}\big(\mathbf{tr}(\widehat{\Sigma}_{\boldsymbol{x}}) - \mathbf{tr}(U_1^\top \widehat{\Sigma}_{\boldsymbol{x}} U_1)\big) = 0. \tag{2.43}$$

Since $\mathbf{tr}(U_1^\top \widehat{\Sigma}_{\boldsymbol{x}} U_1) = \mathbf{tr}(\Sigma_1)$, we conclude that

$$\sigma^2 = \frac{1}{D-d}\big(\mathbf{tr}(\widehat{\Sigma}_{\boldsymbol{x}}) - \mathbf{tr}(\Sigma_1)\big). \tag{2.44}$$

This expression is minimized when $\mathbf{tr}(\Sigma_1)$ is maximized, which happens when Σ_1 is chosen as the matrix with the top d eigenvalues of $\widehat{\Sigma}_{\boldsymbol{x}}$.

In summary, we have shown that the optimal solution to PPCA is given by

$$\widehat{U}_d = U_1(\Sigma_1 - \widehat{\sigma}^2 I)^{1/2} \quad \text{and} \quad \widehat{\sigma}^2 = \frac{1}{D-d}\sum_{i=d+1}^{D}\lambda_i, \tag{2.45}$$

where U_1 is the matrix with the top d eigenvectors of $\widehat{\Sigma}_{\boldsymbol{x}}$, Σ_1 is the matrix with the corresponding d top eigenvalues, and λ_i is the ith eigenvalue of $\widehat{\Sigma}_{\boldsymbol{x}}$.

2.2 Determining the Number of Principal Components

In the above discussions, we have assumed that the dimension of the subspace S (the number of principal components) is given and that all the sample points can be fit with the same geometric or statistical model: a subspace. In this section, we discuss various robustness issues for PCA, such as how to determine the dimension of the subspace from noisy data and how to determine the principal components when the data are contaminated by outliers or incomplete data points.

Notice that the SVD of the noisy data matrix $\boldsymbol{X}$ gives a solution to PCA not only for a particular dimension of the subspace, d, but also for all $d = 1, 2, \ldots, D$. This has an important side-benefit: If the dimension of the subspace S is *not* known or specified a priori, rather than optimizing for both d and S simultaneously, we can easily look at the entire spectrum of solutions for different values of d to decide on the "best" estimate $\hat{d}$ for the dimension of the subspace d given the data $\boldsymbol{X}$.

The problem of determining the optimal dimension $\hat{d}$ is in fact a "model selection" problem. As we discussed in the introduction of the book, the conventional wisdom is to strike a good balance between the *complexity* of the chosen model and the *fidelity* of the data to the model. The dimension d of the subspace S is a natural measure of model complexity, while the least-squares error between the given data $\boldsymbol{X}$ and its projection $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_N]$ onto the subspace S, i.e.,

$$\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_F^2 = \sum_{i=1}^{N} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2, \tag{2.46}$$

is a natural measure of the data fidelity.

As shown in the proof of Theorem 2.2, the optimal least-squares error is given by the sum of the squares of the remaining singular values of $\boldsymbol{X}$, $\sum_{i=d+1}^{D} \sigma_i^2$. Normally, the leading term σ_{d+1}^2 of $\sum_{i=d+1}^{D} \sigma_i^2$ is already a good index of the magnitude of the remaining ones. Thus, one can simply seek for a balance between d and σ_{d+1}^2 by minimizing an objective function of the form:

$$J_1(d) \doteq \alpha \cdot \sigma_{d+1}^2 + \beta \cdot d \tag{2.47}$$

for some proper weights $\alpha, \beta > 0$. A similar criterion that is often used to determine the rank d of a noisy matrix $\boldsymbol{X}$ is:

$$J_2(d) \doteq \frac{\sigma_{d+1}^2}{\sum_{i=1}^{d} \sigma_i^2} + \kappa d, \tag{2.48}$$

where $\kappa > 0$ is a proper weight (see [Kanatani, 2002]).

In general, the ordered singular values of the data matrix $\boldsymbol{X}$ versus the dimension d of the subspace resemble a plot similar to that shown in Figure 2.1. In the statistics literature, this is known as the "Scree graph." We should see a significant drop in the singular values right after the "correct" dimension $\hat{d}$, which is sometimes called the "knee" or "elbow" point of the plot. Such a point is a stable minimum as it optimizes the above objective function (2.47) for a range of values for α and β, or the objective function in (2.48) for a range of values of κ. One can also select the optimal dimension $\hat{d}$ from the Scree graph by specifying a tolerance τ for the fitting error and then using the plot to identify the model that has the lowest dimension and satisfies the given tolerance, as indicated in the figure.

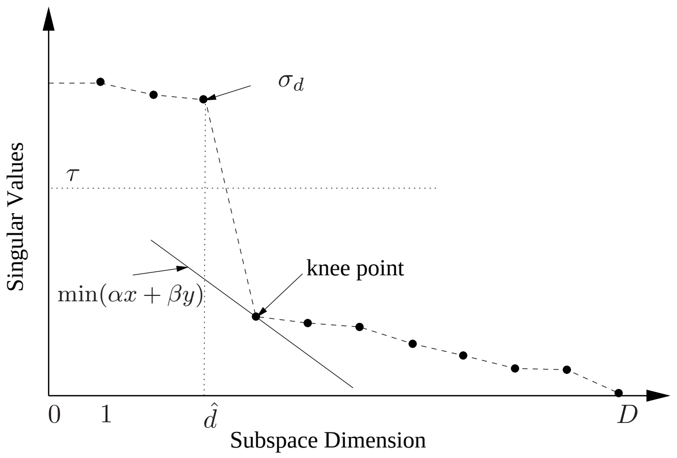


Figure 2.1. Singular value as a function of the dimension of the subspace.

A more principled approach to finding the optimal dimension of the subspace, $\hat{d}$, is to use some of the model selection criteria described in Appendix A. Such criteria rely on a different choice of the model complexity term and provide an automatic way of choosing the parameters α and β or κ. Specifically, the complexity of the model is measured by the number of parameters needed to describe the subspace. Using the Grassmannian coordinates, the dimension of the parameter space for a d-dimensional subspace in $\mathbb{R}^D$ is $Dd - d^2$.[5] With a model parameter space of dimension $Dd - d^2$ and a Gaussian noise model with known variance σ^2, the Bayesian information criterion (BIC) is equivalent to minimizing

$$\mathrm{BIC}(d) \doteq \frac{1}{N}\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_F^2 + (\log N)\frac{(Dd - d^2)}{N}\sigma^2, \tag{2.49}$$

while the Akaike information criterion (AIC) minimizes

$$\mathrm{AIC}(d) \doteq \frac{1}{N}\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_F^2 + 2\frac{(Dd - d^2)}{N}\sigma^2. \tag{2.50}$$

More recently, a geometric version of the Akaike information criterion has been proposed by [Kanatani, 2003]. The Geometric AIC minimizes

$$\text{G-AIC}(d) \doteq \frac{1}{N}\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_F^2 + 2\frac{(Dd - d^2 + Nd)}{N}\sigma^2, \tag{2.51}$$

where the extra term Nd accounts for the number of coordinates needed to represent (the closest projection of) the given N data points in the estimated d-

[5] $Dd - d^2$ is the dimension of the Grassmannian manifold of d-dimensional subspaces in $\mathbb{R}^D$. To specify a subspace, one can use the so-called Grassmannian coordinates which need exactly $Dd - d^2$ entries: starting with a $D \times d$ matrix whose columns form a basis for the subspace, perform column-reduction so that the first $d \times d$ block is the identity matrix. Then, one only needs to give the rest $(D - d) \times d$ entries to specify the subspace.

dimensional subspace. From an information-theoretic viewpoint, the additional Nd coordinates are necessary if we are interested in encoding not only the model but also the data themselves. This is often the case when we use PCA for purposes such as data compression and dimension reduction. The quantity $\frac{(Dd-d^2+Nd)}{N}$ is closely related to the so-called "effective dimension" of the data set defined in Chapter 6, which can be generalized to multiple subspaces.

In some sense, all the above criteria can be loosely referred to as *information-theoretic* model selection criteria, in the sense that most of these criteria can be interpreted as variations to minimizing the optimal code length for both the model and the data with respect to certain class of distributions and coding schemes [Hansen and Yu, 2001].[6] There are many other methods for determining the number of principal componenrs. Interested readers may find more references in [Jolliffe, 1986].

2.3 Robust PCA: Classical Approaches

In the above discussions, we have assumed that all the sample points can be fit with the same statistical or geometric model: a subspace. In practical applications it is often the case that the data points are contaminated not only by noise, but also by outliers. Sometimes it is also the case that some entries of the of the data points are missing. In this section, we discuss classical approaches from robust statistics for dealing with outliers and incomplete data points in the context of PCA.

2.3.1 Dealing with Incomplete Data Points

In practice, it is often the case that some of the given data points are "incomplete." For an incomplete data point $\boldsymbol{x} = [x_1, x_2, \ldots, x_D]^\top$, we mean that some of its entries are missing or unspecified. For instance, if the x_i-entry of $\boldsymbol{x}$ is missing, then $\boldsymbol{x}$ is known only up to a line in $\mathbb{R}^D$:

$$\boldsymbol{x} \ \in \ L \doteq \big\{[x_1, \ldots, x_{i-1}, t, x_{i+1}, \ldots, x_D]^\top, t \in \mathbb{R}\big\}. \qquad (2.52)$$

One should be aware that an incomplete data point is in nature rather different from a noisy data point.[7] In general, such incomplete data points can contain useful information about the model, and in the case of PCA, the principal subspace. For instance, if the principal subspace happens to contain the line L, the principal subspace can be determined from a sufficiently large number of such lines.

[6]Even if one chooses to compare models by their algorithmic complexity, such as the minimum message length (MML) criterion [Wallace and Boulton, 1968] (an extension of the Kolmogrov complexity to model selection), a strong connection with the above information-theoretic criteria, such as MDL, can be readily established via Shannon's optimal coding theory (see [Wallace and Dowe, 1999]).

[7]One can view incomplete data points as a very special type of noisy data points which have infinite uncertainty only in certain directions.

In general, the line L may or may not lie in the principal subspace. We therefore should handle incomplete data points with more care.

A useful observation here is that an incomplete data point $\boldsymbol{x}$ is just as good as any point on the line L. Hence it is natural to choose a representative $\hat{\boldsymbol{x}} \in L$ that is the closest to the principal subspace. If we let the columns of U_d for a basis form an orthonormal basis for the subspace, then the closest point $\boldsymbol{x}^* = [x_1, \dots, x_{i-1}, t^*, x_{i+1}, \dots, x_D]^\top$ on L to the principal subspace can be found by minimizing the following quadratic function in t:

$$t^* = \arg\min_t \big(\boldsymbol{x}^\top (I_D - U_d U_d^\top)\boldsymbol{x}\big). \tag{2.53}$$

This problem has a unique solution as long as the line L is not parallel to the principal subspace, i.e., $e_i \notin \text{span}(U_d)$.

In essence, the above process of finding $\boldsymbol{x}^*$ on the principal subspace is to give a rank-d approximation of the entire data set containing both complete and incomplete data points. Mathematically, under the assumption that the samples $\{\boldsymbol{x}_i\}_{i=1}^N$ are zero-mean, PCA with incomplete data is equivalent to finding a rank-d approximation/factorization of the data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N]$ with incomplete data entries (in a least-squares sense). That is, the goal is to find matrices $U_d \in \mathbb{R}^{D\times d}$ and $\boldsymbol{Y} \in \mathbb{R}^{d\times N}$ that minimize $\|\boldsymbol{X} - U_d\boldsymbol{Y}\|_F^2$. The main issue is that some entries of $\boldsymbol{X}$, $\{x_{ij}\}$, are missing.

Obviously, we cannot expect to always be able to find a solution to this problem. For instance, suppose the first entry is missing from each one of the data points. Then we cannot hope to be able to recover such an entry. Likewise, suppose that all the entries of one data point are missing. While in this case we can find the subspace from all other data points, we cannot recover the low-dimensional representation for that point. Nevertheless, if the missing entries do not follow a specific pattern, we should be able to recover both U_d and $\boldsymbol{Y}$ as long as the number of measurements (known entries of $\boldsymbol{X}$) is sufficiently large relative to the number of unknowns ($D(D-d) + dN$ entries in U_d and $\boldsymbol{Y}$). Intuitively, the smallest the rank of the matrix d the larger the amount of missing information we can tolerate.

In what follows, we discuss a few traditional approaches to PCA with incomplete data. Throughout the exposition, we will make use of a matrix $W \in \mathbb{R}^{D\times N}$ whose entries $\{w_{ij}\}$ encode the locations of the missing information, i.e.,

$$w_{ij} = \begin{cases} 1 & \text{if } x_{ij} \text{ is known} \\ 0 & \text{if } x_{ij} \text{ is missing} \end{cases}. \tag{2.54}$$

We will also make use of the Haddamart product of two matrices $W \odot \boldsymbol{X}$, which is defined as $(W \odot \boldsymbol{X})_{ij} = w_{ij}x_{ij}$.

Incomplete Mean and Covariance

Since the optimal solution to PCA is obtained from the mean and covariance of the data points, a straightforward method for dealing with missing entries is to simply compute the mean and covariance over the missing entries. Specifically, the *incomplete mean* and *incomplete covariance* are given by

$$\hat{\mu}_i = \frac{\sum_{j=1}^N w_{ij} x_{ij}}{\sum_{j=1}^N w_{ij}} \implies \hat{\boldsymbol{x}}_0 = \operatorname{diag}(W\mathbf{1})^{-1}(W \odot \boldsymbol{X})\mathbf{1}, \tag{2.55}$$

$$\hat{\sigma}_{ij} = \frac{\sum_{j=1}^N w_{ij}(x_{ij} - \mu_i)}{\sum_{j=1}^N w_{ij}} \implies \hat{\boldsymbol{x}}_0 = \operatorname{diag}(W\mathbf{1})^{-1}(W \odot \boldsymbol{X})\mathbf{1}, \tag{2.56}$$

$$\tag{2.57}$$

Power Factorization

Power Factorization (PF) is an iterative algorithm for finding a low-rank approximation $U_d\boldsymbol{Y}$ of a matrix $\boldsymbol{X}$ with missing entries (see [Vidal and Hartley, 2004] and references therein for further details). The main idea behind PF is to minimize $\|\boldsymbol{X} - U_d\boldsymbol{Y}\|_F^2$ considering only the known entries of $\boldsymbol{X}$. Given $\boldsymbol{Y}$, the optimal U_d can be computed linearly. Likewise, given U_d, the optimal $\boldsymbol{Y}$ can be computed linearly. The PF algorithm then iterates between these two steps till convergence.

More specifically, the PF algorithm tries to minimize a cost function of the form

$$\|W \odot (\boldsymbol{X} - U_d\boldsymbol{Y})\|_F^2 = \sum_{i=1}^D \sum_{j=1}^N w_{ij}(x_{ij} - u_i^\top \boldsymbol{y}_j)^2. \tag{2.58}$$

Notice that this cost function is the same as that in (2.16), except that the errors $\varepsilon_{ij} = x_{ij} - u_i^\top \boldsymbol{y}_j$ associated with the missing entries ($w_{ij} = 0$) are removed from the cost function.

2.3.2 *Dealing with Outliers*

Another issue that we encounter in practice is that a small portion of the data points does not fit well the same model as the rest of the data. Such points are called *outliers*. Their presence can lead to a completely wrong estimate of the underlying subspace. Therefore, it is very important to develop methods for detecting and eliminating outliers from the given data.

The true nature of outliers can be very elusive. In fact, there is really no unanimous definition for what an outlier is.[8] Outliers could be atypical samples that have an unusually *large influence* on the estimated model parameters. Outliers could also be perfectly valid samples from the same distribution as the rest of the data that happen to be *small-probability* instances. Alternatively, outliers could be samples drawn from a different model, and therefore they will likely *not be consistent* with the model derived from the rest of the data. In principle, however, there is no way to tell which is the case for a particular "outlying" sample point.

In what follows, we discuss a few approaches to dealing with outliers that are particularly related to PCA. We will distinguish between two types of outliers.

[8]For a more thorough exposition of outliers in statistics, we recommend the books of [Barnett and Lewis, 1983, Huber, 1981].

The first kind, which we call *sample outliers*, corresponds to the case where the entire sample data point is an atypical sample. The second kind, which we call *intra-sample outliers*, corresponds to the case where only a few entries of a data point are atypical, while the remaining entries are not. The main distinction to be made is that in the latter case we do not want to discard the entire data point, but only the atypical entries.

Influence-Based Outlier Detection

This approach relies on the assumption that an outlier is an *atypical* sample which has an unusually large influence on the estimated model parameters. This leads to an outlier detection scheme where the influence of a sample is determined by comparing the difference between the model estimated with and without this sample. For instance, for PCA one may use a *sample influence function* to measure the difference:

$$I(\boldsymbol{x}_i, U_d) \doteq \langle \hat{U}_d, \hat{U}_{d(i)} \rangle, \tag{2.59}$$

where $\langle \cdot, \cdot \rangle$ is the largest subspace angle (see Exercise 2.2) between the subspace $\text{span}(\hat{U}_d)$ estimated with the ith sample and the subspace $\text{span}(\hat{U}_{d(i)})$ without the ith sample. The larger the difference, the larger the influence of $\boldsymbol{x}_i$ on the estimate, and the more likely that $\boldsymbol{x}_i$ is an outlier. Thus, we may eliminate a sample $\boldsymbol{x}_i$ as an outlier if

$$I(\boldsymbol{x}_i, U_d) \geq \tau \tag{2.60}$$

for some threshold $\tau > 0$ or if $I(\boldsymbol{x}_i, U_d)$ is relatively large among all the samples.

However, this method does not come without an extra cost. We need to compute the principal components (and hence perform SVD) N times: one time with all the samples together and another $N-1$ times with one sample eliminated from each time. There have been many studies that aim to give a formula that can accurately approximate the sample influence without performing SVD N times. Such a formula is called a *theoretical influence* function. For more detailed discussion of the sample influence for PCA, we refer the interested readers to [Jolliffe, 2002].

Probability-Based Outlier Detection

In this approach a model is fit to *all* the sample points, including potential outliers. Outliers are then detected as the points that correspond to small-probability events or that have large fitting errors with respect to the identified model. A new model is then estimated with the detected outliers removed or down-weighted. This process is then repeated until the estimated model stabilizes.

In the case of PCA, the goal is to find a low-dimensional subspace that best fits a given set of data points $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^N$ by minimizing the least-squares errors

$$\sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{x}_0 - U_d \boldsymbol{y}_i\|^2, \tag{2.61}$$

between a point $\boldsymbol{x}_i$ and its projection onto the subspace $\boldsymbol{x}_0 + U_d \boldsymbol{y}_i$, where $\boldsymbol{x}_0 \in \mathbb{R}^D$ is any point in the subspace, $U_d \in \mathbb{R}^{D \times d}$ is a basis for the subspace, and $\boldsymbol{y}_i \in \mathbb{R}^d$ are the coordinates of the point in the subspace. If there were no outliers, an optimal solution to PCA could be obtained as described in Section 2.1.2, i.e.,

$$\hat{\boldsymbol{x}}_0 = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \quad \text{and} \quad \hat{\boldsymbol{y}}_i = \hat{U}_d^\top (\boldsymbol{x} - \hat{\boldsymbol{x}}_0), \tag{2.62}$$

where $\hat{U}_d$ is a $D \times d$ matrix whose columns are the top d eigenvectors of

$$\widehat{\Sigma}_N = \frac{1}{N-1}\sum_{i=1}^{N} (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0)(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0)^\top. \tag{2.63}$$

If we adopt the guideline that outliers are samples that do not fit the model well or have a small probability with respect to the estimated model, then the outliers are exactly those samples that have a relatively large residual

$$\varepsilon_i^2 = \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0 - \hat{U}_d \hat{\boldsymbol{y}}_i\|^2 \quad \text{or} \quad \varepsilon_i^2 = \boldsymbol{x}_i^\top \widehat{\Sigma}_N^{-1} \boldsymbol{x}_i, \quad i = 1, 2, \dots, N. \tag{2.64}$$

The first error is simply the distance to the subspace, while the second error is the *Mahalanobis distance*,[9] which is obtained when we approximate the probability that a sample $\boldsymbol{x}_i$ comes from this model by a multivariate Gaussian

$$p(\boldsymbol{x}_i; \widehat{\Sigma}_N) = \frac{1}{(2\pi)^{D/2} \det(\widehat{\Sigma}_N)^{1/2}} \exp\Big(-\frac{1}{2}\boldsymbol{x}_i^\top \widehat{\Sigma}_N^{-1} \boldsymbol{x}_i\Big). \tag{2.65}$$

In principle, we could use $p(\boldsymbol{x}_i, \widehat{\Sigma}_N)$ or either residual ε_i to determine if $\boldsymbol{x}_i$ is an outlier. However, the above estimate of the subspace is obtained using all the samples, including the outliers themselves. Therefore, the estimated subspace could be completely wrong and hence the outliers could be incorrectly detected. In order to improve the estimate of the subspace, one can recompute the model parameters after discarding or down-weighting samples that have large residuals. More specifically, let $w_i \in [0, 1]$ be a weight assigned to the ith point such that $w_i \approx 1$ if $\boldsymbol{x}_i$ is an inlier and $w_i \approx 0$ if $\boldsymbol{x}_i$ is an outlier. Then, similarly to (2.16), a new estimate of the subspace can be obtained by minimizing a weighted least-squares error:

$$\sum_{i=1}^{N} w_i \|\boldsymbol{x}_i - \boldsymbol{x}_0 - U_d \boldsymbol{y}_i\|^2 \quad \text{s.t.} \quad U_d^\top U_d = I_d \quad \text{and} \quad \sum_{i=1}^{N} w_i \boldsymbol{y}_i = \boldsymbol{0}. \tag{2.66}$$

[9] In fact, it can be shown that [Ferguson, 1961], if the outliers have a Gaussian distribution of a different covariance matrix $a\Sigma$, then ε_i is a sufficient statistic for the test that maximizes the probability of correct decision about the outlier (in the class of tests that are invariant under linear transformations). Interested reader may want to find out how this distance is equivalent (or related) to the sample influence $\widehat{\Sigma}_N^{(i)} - \widehat{\Sigma}_N$ or the approximate sample influence given in (A.50).

If we follow the same steps as in Section 2.1.2, we can find that an optimal solution to this problem is of the form:

$$\hat{\boldsymbol{x}}_0 = \frac{\sum_{i=1}^N w_i \boldsymbol{x}_i}{\sum_{i=1}^N w_i} \quad \text{and} \quad \hat{\boldsymbol{y}}_i = \hat{U}_d^\top (\boldsymbol{x} - \hat{\boldsymbol{x}}_0), \tag{2.67}$$

where $\hat{U}_d$ is a $D \times d$ matrix whose columns are the top d eigenvectors of

$$\widehat{\Sigma}_N = \frac{\sum_{i=1}^N w_i (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0)(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_0)^\top}{\sum_{i=1}^N w_i - 1}. \tag{2.68}$$

As a consequence, under the least-squares criterion, finding a robust solution to PCA reduces to finding a robust estimate of the sample mean and the sample covariance of the data by properly setting the weights. In what follows, we discuss two main approaches approaches for estimating the weights.

Multivariate trimming (MVT) is a popular robust method for estimating the sample mean and covariance of a set of points. This method assumes discrete weights

$$w_i = \begin{cases} 1 & \text{if } \boldsymbol{x}_i \text{ is an inlier} \\ 0 & \text{if } \boldsymbol{x}_i \text{ is an outlier} \end{cases}, \tag{2.69}$$

and chooses the outliers as a certain percentage of the samples (say 10 percent) that have relatively large residual. This can be done by simply sorting the residuals $\{\varepsilon_i\}$ from the lowest to the highest and then choosing as outliers the desired percentage of samples with the highest residuals. Once the outliers are trimmed out, one can use the remaining samples to re-estimate the subspace as in (2.67)-(2.68). Each time we have a new estimate of the subspace, we can recalculate the residual of every sample and reselect samples that need to be trimmed. We can repeat the above process until a stable estimate of the subspace is obtained. When the percentage of outliers is somewhat known, it usually takes only a few iterations for MTV to converge and the resulting estimate is in general more robust. However, if the percentage is wrongfully specified, MVT may not converge or it may converge to a wrong estimate of the subspace. In general, the ”breakdown point” of MTV, i.e., the proportion of outliers that it can tolerate before giving a completely wrong estimate, depends only on the chosen trimming percentage. In Chapter **??**, we will discuss how MVT can be modified in the context of GPCA when the percentage of outliers is not known.

Maximum Likelihood Type Estimators (M-Estimators) uses continuous weights $w_i = \rho(\varepsilon_i)/\varepsilon_i^2$ for some robust loss function $\rho(\cdot)$. The objective function then becomes

$$\sum_{i=1}^N \rho(\varepsilon_i). \tag{2.70}$$

Many loss functions $\rho(\cdot)$ have been proposed in the statistics literature [Huber, 1981, Barnett and Lewis, 1983]. When $\rho(\varepsilon) = \varepsilon^2$, we obtain the standard least-squares solution, which is not robust. Other robust loss functions include

1. L_1 or total variation loss: $\rho(\varepsilon) = |\varepsilon|$.
2. Cauchy loss: $\rho(\varepsilon) = \varepsilon_0^2 \log(1 + \varepsilon^2/\varepsilon_0^2)$
3. Huber loss [Huber, 1981]: $\rho(\varepsilon) = \begin{cases} \varepsilon^2 & \text{if} \quad |\varepsilon| < \varepsilon_0 \\ 2\varepsilon_0|\varepsilon| - \varepsilon_0^2 & \text{otherwise} \end{cases}$
4. Geman-McClure loss [Geman and McClure, 1987]: $\rho(\varepsilon) = \frac{\varepsilon^2}{\varepsilon^2+b^2}$

where $\varepsilon > 0$ is a parameter.

One way of minimizing (2.70) with respect to the subspace parameters is to initialize all the weights to $w_i = 1, i = 1, \dots, N$. This will give an initial estimate for the subspace which is the same as that given by PCA. Given this initial estimate of the subspace, one may compute the weights as $w_i = \rho(\varepsilon)/\varepsilon^2$ using any the aforementioned robust cost functions. Given these weights, one can reestimate the subspace from (2.67)-(2.68). One can then iterate in between computing the weights given the subspace and computing the subspace given the weights. This iterative process, called iterative re-weighted least squares, converges to a local minima of the cost function (2.70). An alternative method for minimizing (2.70) is to simply do gradient descent. This method may be preferable for loss functions ρ that are differentiable, e.g., the Geman-McClure loss function.

One drawback of the M-estimators is that its breakdown point is inversely proportional to the dimension of the space. Thus, the M-estimators become much less robust when the dimension is high. This makes M-estimators of limited use in the context of GPCA since the dimension of the space is typically very high (≥ 70).

Consensus-Based Outlier Detection

This approach assumes that the outliers are not drawn from the same model as the rest of the data. Hence it makes sense to try to avoid the outliers when we infer the model in the first place. However, without knowing which points are outliers beforehand, how can we avoid them? One idea is to fit a model, instead of to all the data points at the same time, only to a *subset* of the data. This is possible when the number of data points required for a unique solution for the estimate is *much* smaller than that of the given data set. Of course, one should *not* expect that a randomly chosen subset will have no outliers and always lead to a good estimate of the model. Thus, one should try on *many different subsets*:

$$\boldsymbol{X}_1, \boldsymbol{X}_2, \dots, \boldsymbol{X}_n \ \subset \ \boldsymbol{X}. \tag{2.71}$$

The rationale is that if the number of subsets are large enough,[10] one of the trial subsets, say $\boldsymbol{X}_i$, likely contains few or no outliers and hence the resulting model would be the most consistent with the rest of the data points.

In the case of PCA, the minimum number of data points needed to define the model is d for linear subspaces and $d+1$ for affine subspaces. Therefore, each

[10]See Appendix A.5 for details on how large this number needs to be.

subset $\boldsymbol{X}_i$ is formed by randomly sampling d (or $d+1$) data points and fitting a subspace with basis $\widehat{U}_d(\boldsymbol{X}_i)$ to the subset. The subset $\boldsymbol{X}_i$ gives a consistent estimate $\widehat{U}_d(\boldsymbol{X}_i)$ of the subspace if the number of data points that fit the subspace well is large enough. For instance, we may claim that the subset $\boldsymbol{X}_i$ gives a consistent estimate $\widehat{U}_d(\boldsymbol{X}_i)$ if the following criterion is maximized (among all the chosen subsets):

$$\max_i \#\big\{\boldsymbol{x} \in \boldsymbol{X} : \big\|\boldsymbol{x} - \hat{U}_d(\boldsymbol{X}_i)\big\| \leq \tau\big\}, \tag{2.72}$$

where $\#$ is the cardinality of the set and $\tau > 0$ is a chosen error threshold. This scheme is typically called *Random Sample Consensus* (RANSAC) [Fischler and Bolles, 1981], and it normally improves the robustness of the estimate. As a word of caution, in practice, in order to design a successful RANSAC algorithm, one needs to carefully choose a few key parameters: the size of every subset, the number of subsets, and the consensus criterion.[11] There is a vast amount of literature on RANSAC-type algorithms, especially in computer vision. For more details on RANSAC and other related random sampling techniques, the reader is referred to Appendix A.5. In Chapter **??**, we will discuss some limitations of RANSAC in the context of estimating multiple subspaces simultaneously.

2.4 Robust PCA: A Sparse Representation Approach

In this section, we discuss a sparse representation-based approach to dealing with intra-sample outliers in PCA. In this approach, it is assumed that the given data matrix $\boldsymbol{X}$ is generated as the sum of two matrices

$$\boldsymbol{X} = L_0 + E_0. \tag{2.73}$$

The matrix L_0 represents the ideal low-rank matrix, while the matrix E_0 represents the intra-sample outliers. Since many entries of $\boldsymbol{X}$ are not corrupted (otherwise the problem is not well posed), many entries of E_0 should be zero. As a consequence, we can pose the robust PCA problem as one of decomposing a given matrix $\boldsymbol{X}$ as the sum of two matrices $L+E$, where L is of low-rank and E is sparse. This problem can be formulated as

$$\min_{L,E} \quad \text{rank}(L) + \lambda\|E\|_0 \quad \text{s.t.} \quad \boldsymbol{X} = L + E, \tag{2.74}$$

where $\|E\|_0$ is the number of non-zero entries in E and $\lambda > 0$ is a user parameter.

At a first sight, one may think that solving the problem in (2.74) is really impossible. First of all, we have $D \times N$ equations and $2D \times N$ unknowns. Second, it is not clear that we can always decompose a matrix as the sum of a low-rank matrix and a sparse matrix. For instance, if $\boldsymbol{X}_{11} = 1$ and $\boldsymbol{X}_{ij} = 0$ for all $(i,j) \neq (1,1)$,

[11] That is, the criterion that verifies whether each sample is consistent with the model derived from the subset.

then the matrix $\boldsymbol{X}$ is both rank 1 and sparse. Thus, if $\lambda = 1$, we can choose $L = \boldsymbol{X}$ and $E = 0$ or $L = 0$ and $E = \boldsymbol{X}$ as valid solutions. Last, but not least, the cost function to be minimized is non-convex and non-differentiable. Moreover, it is well known that this problem is in general NP hard [**?**].

In what follows, we will show that, under certain conditions on L_0 and E_0, the optimal solution to (2.74) can be found by solving the following convex optimization problem

$$\min_{L,E} \quad \|L\|_* + \lambda\|E\|_1 \quad \text{s.t.} \quad \boldsymbol{X} = L + E, \tag{2.75}$$

where $\|L\|_* = \sum_i \sigma_i(L)$ is the nuclear norm of L, i.e., the sum of its singular values, and $\|E\|_1 = \sum_{i,j} |E_{ij}|$ is the ℓ_1 norm of E considered as a vector. The conditions rely on recent results from compressed sensing, which aim at finding a sparse solution to a linear system $A\boldsymbol{x} = b$. Therefore, we will first review recent results on sparsity and rank minimization before we return to the problem of decomposing a matrix as the sum of a low rank plus a sparse matrix.

2.4.1 *Basis Pursuit*

Let us first consider the simpler problem of finding a solution to the linear system $A\boldsymbol{x} = b$, where $\boldsymbol{x} \in \mathbb{R}^N$, $\boldsymbol{b} \in \mathbb{R}^D$ and $A \in \mathbb{R}^{D\times N}$, with $D < N$. Since this linear system is underdetermined, in general there could be many solutions $\boldsymbol{x}$. A classical approach to finding a unique solution (when a solution exists) is to look for a vector $\boldsymbol{x}$ of minimum ℓ_2 norm, i.e., $\min \|\boldsymbol{x}\|_2$ such that $A\boldsymbol{x} = b$.

An alternative approach is to look for a vector $\boldsymbol{x}$ that is sparse. Specifically, assume that the vector b is generated as $A\boldsymbol{x}_0 = b$, where $\boldsymbol{x}_0$ is a d-sparse vector, i.e., $\|\boldsymbol{x}_0\|_0 = d \ll N$. When the matrix A is such that $\delta_{2d}(A) < 1$, where $\delta_d(A)$ is the smallest number such that for all $\boldsymbol{x}$ with $\|\boldsymbol{x}\|_0 \le d$,

$$(1 - \delta_d(A))\|\boldsymbol{x}\|_2^2 \le \|A\boldsymbol{x}\|_2^2 \le (1 + \delta_d(A))\|\boldsymbol{x}\|_2^2, \tag{2.76}$$

then the $\boldsymbol{x}_0$ is the only d-sparse vector such that $A\boldsymbol{x} = b$.

In order to find $\boldsymbol{x}_0$, we seek a solution to the problem

$$\min \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad A\boldsymbol{x} = b. \tag{2.77}$$

In general, this problem is NP hard. However, when the matrix A satisfies the so-called restricted isometry property $\delta_{2d}(A) < \sqrt{2} - 1$, then the optimal solution to (2.80) can be found by solving the following convex optimization problem

$$\min \|\boldsymbol{x}\|_1 \quad \text{s.t.} \quad A\boldsymbol{x} = b. \tag{2.78}$$

2.4.2 *Rank Minimization and PCA with Missing Data*

Consider now the problem of finding a solution to the matrix linear system $\mathcal{A}(X) = b$, where $X \in \mathbb{R}^{D\times N}$, $b \in \mathbb{R}^K$, $\mathcal{A} : \mathbb{R}^{D\times N} \to \mathbb{R}^K$ is a linear map, and $K < D \times N$. As before, there could be many matrices X that solve the linear system $\mathcal{A}(X) = b$. Assume that there is a matrix X_0 of rank $d \ge 1$ that solves the linear system. When the matrix $\mathcal{A}$ is such that $\delta_{2d}(\mathcal{A}) < 1$, where $\delta_d(\mathcal{A})$ is the smallest number such that for all matrices $X \in \mathbb{R}^{D\times N}$ of rank d

$$(1-\delta_d(\mathcal{A}))\|X\|_F \leq \|\mathcal{A}(X)\|_2 \leq (1+\delta_d(\mathcal{A}))\|X\|_F, \tag{2.79}$$

then X_0 is the only matrix of rank at most d satisfying $\mathcal{A}(X) = b$.

In order to find X_0, we seek a solution to the problem

$$\min \quad \text{rank}(X) \quad \text{s.t.} \quad \mathcal{A}(X) = b. \tag{2.80}$$

In general, this problem is NP hard. However, when the matrix $\mathcal{A}$ is such that $\delta_{5d}(\mathcal{A}) < 1/10$, the optimal solution to (2.80) can be found by solving the following convex problem

$$\min \quad \|X\|_* \quad \text{s.t.} \quad \mathcal{A}(X) = b. \tag{2.81}$$

Observe that, when generalizing from the vector case to the matrix case, the 2-norm of $\boldsymbol{x}$ is replaced by the Frobenius norm of X. Observe also that the Frobenius norm $\|X\|$ is the ℓ_2 norm of the singular values, while the nuclear norm $\|X\|_*$ is the ℓ_1 norm of the singular values.

Observe also that the above rank minimization problem provides a solution to PCA with missing data. Specifically, let $\boldsymbol{X}$ be a given matrix of rank d with missing entries, and recall the definition of the matrix W where $w_{ij} = 1$ if the X_{ij} is known and $w_{ij} = 0$ otherwise. Then, we can find $\boldsymbol{X}$ by solving the problem

$$\min \quad \text{rank}(X) \quad \text{s.t.} \quad W \odot X = W \odot \boldsymbol{X} \tag{2.82}$$

In other words, we seek a matrix X of minimum rank, whose entries coincide with the known entries of $\boldsymbol{X}$. From the results above, we know that if the matrix $\mathcal{A}_W$ defined by the relationship $\mathcal{A}_W(X) = W \odot X$ is such that $\delta_{2d}(\mathcal{A}_W) < 1$, then the missing entries of $\boldsymbol{X}$ are uniquely defined. Moreover, if $\delta_{5d}(\mathcal{A}_W) < 1/10$, we can find the missing entries of $\boldsymbol{X}$ by solving the following convex problem

$$\min \quad \|X\|_* \quad \text{s.t.} \quad W \odot X = W \odot \boldsymbol{X}. \tag{2.83}$$

An additional advantage of this formulation of PCA with missing data is that we do not need to specify the number of principal components in advance: the number of principal components is simply the rank of X and this method searches for the matrix of minimum rank.

2.4.3 *Principal Component Pursuit and Robust PCA*

Let us now return to the original problem of decomposing a matrix $\boldsymbol{X}$ as the sum of a low-rank matrix L_0 plus a sparse matrix E_0. Recall from (2.75) that we wish to find L_0 and E_0 by solving the following optimization problem

$$\min_{L,E} \quad \|L\|_* + \lambda\|E\|_1 \quad \text{s.t.} \quad \boldsymbol{X} = L + E, \tag{2.84}$$

The following theorem gives conditions on the rank of the matrix and the percentage of outliers under which the optimal solution is exactly L_0 and E_0 with overwhelming probability.

Theorem 2.5 (**??**). *Let $\boldsymbol{X} = L_0 + E_0$. Assume that there exists a $\mu > 0$ such that the compact SVD of $L_0 = U\Sigma V^\top$ satisfies*

$$\max_i \|u_i\|^2 \leq \frac{\mu d}{D}, \quad \max_i \|v_i\|^2 \leq \frac{\mu d}{N} \quad \textit{and} \quad \|UV^\top\|_\infty \leq \sqrt{\frac{\mu d}{ND}}, \tag{2.85}$$

where $U = [u_1, u_2, \cdots, u_D]^\top \in \mathbb{R}^{D\times d}$ and $V = [v_1, v_2, \cdots, v_N]^\top \in \mathbb{R}^{N\times d}$. Assume also that the support of E_0 is uniformly distributed among all the sets of cardinality $D \times N$. If

$$\textit{rank}(L_0) \leq \frac{\rho_d \min\{D, N\}}{\mu \log^2\big(\max\{D, N\}\big)} \quad \textit{and} \quad \|E_0\|_0 \leq \rho_s ND. \tag{2.86}$$

Then there is a constant c such that with probability at least $1 - c\max\{N, D\}^{-10}$, the solution (L^*, E^*) to (2.75) with $\lambda = \frac{1}{\sqrt{\max\{N,D\}}}$ is exact, i.e.,

$$L^* = L_0 \quad \textit{and} \quad E^* = E_0. \tag{2.87}$$

Assuming that the conditions of the theorem are satisfied, the next question is how do we actually optimize the cost function in order to find the global minimum.

2.5 Extensions to PCA

2.5.1 Factor Analysis

Factor Analysis (FA) resolves the aforementioned ambiguity by assuming that

1. the low-dimensional representation has unit covariance $\Sigma_{\boldsymbol{y}} = I_d \in \mathbb{R}^{d\times d}$ and
2. the noise covariance matrix $\Sigma_\varepsilon \in \mathbb{R}^{D\times D}$ is diagonal.

These assumptions lead to the following relationship

$$\Sigma_{\boldsymbol{x}} = U_d U_d^\top + \Sigma_\varepsilon, \tag{2.88}$$

from which it follows that the off-diagonal entries of $\Sigma_{\boldsymbol{x}}$ are equal to the off-diagonal entries of $U_d U_d^\top$. As a consequence, even though both FA and PCA try to capture as much information from $\Sigma_{\boldsymbol{x}}$ into $\Sigma_{\boldsymbol{y}}$, the information they attempt to capture is not the same. On the one hand, FA tries to find a matrix U_d such that the covariances are preserved, i.e., the off-diagonal entries of $\Sigma_{\boldsymbol{x}}$. On the other hand, PCA tries to preserve the variances, i.e., the diagonal entries of $\Sigma_{\boldsymbol{x}}$.

The analysis above implies that, in general, the solutions to PCA and FA need not be the same. To see this, simply multiply (2.88) on the right by U_d to obtain

$$(\Sigma_{\boldsymbol{x}} - \Sigma_\varepsilon)U_d = U_d\Lambda, \tag{2.89}$$

where $\Lambda = U_d^\top U_d \succ 0$. Notice that Λ is, without loss of generality, a diagonal matrix. This is because U_d is defined only up to a rotation R_d, thus if Λ is not diagonal, we can replace it by $R_d^\top U_d^\top U_d R_d$, which is diagonal if R_d is chosen as the matrix of eigenvectors of $U_d^\top U_d$. As a consequence the columns of U_d must be eigenvectors of $\Sigma_{\boldsymbol{x}} - \Sigma_\varepsilon$. Such eigenvectors do not generally coincide with the top d eigenvectors of $\Sigma_{\boldsymbol{x}}$, which give the solution to PCA. Moreover, the eigenvectors of $\Sigma_{\boldsymbol{x}} - \Sigma_\varepsilon$ cannot be directly computed without knowing Σ_ε. As a consequence, the solution to FA is often found via the following iterative procedure:

1. Initialize $\Sigma_\varepsilon = 0$.
2. Given Σ_ε, set $U_d = U_1 \Sigma_1^{1/2}$, where the columns of U_1 are the top d eigenvectors of $\Sigma_{\boldsymbol{x}} - \Sigma_\varepsilon$ and Σ_1 is a diagonal matrix whose diagonal entries are the corresponding eigenvalues.
3. Given U_d, set $\Sigma_\varepsilon = \operatorname{diag}(\Sigma_{\boldsymbol{x}} - U_d U_d^\top)$.
4. Go to 2. until convergence.

Notice that the solutions to PCA and FA are initially the same, except for the linear transformation $\Sigma_1^{1/2}$. However, as the iterations proceed, the solutions are generally different.

2.5.2 *Nonlinear and Kernel PCA*

Although PCA offers a rather useful tool to model the linear structure of a given data set, it becomes less effective when the data lies in a nonlinear manifold. In this section, we introduce some basic extensions to PCA which can, to some extent, handle the difficulty with nonlinearity.

2.5.3 *Nonlinear and Kernel PCA*

Nonlinear PCA

The key idea behind nonlinear PCA is that, instead of applying PCA directly to the given data, we can apply it to a transformed version of the data. The rationale is that the structure of the data may become linear after embedding the data into a higher-dimensional space. For example, imagine that the data point (x_1, x_2) lies in a conic of the form

$$c_1 x_1^2 + c_2 x_1 x_2 + c_3 x_2^2 + c_4 = 0. \tag{2.90}$$

If we define the map $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ as

$$(z_1, z_2, z_3) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2), \tag{2.91}$$

then the conic in $\mathbb{R}^2$ transforms into the following affine subspace in $\mathbb{R}^3$

$$c_1 z_1 + \frac{c_2}{\sqrt{2}} z_2 + c_3 z_3 + c_4 = 0. \tag{2.92}$$

Therefore, instead of learning a nonlinear manifold in $\mathbb{R}^2$, we can simply learn an affine manifold in $\mathbb{R}^5$.

More generally, we seek a nonlinear transformation (usually an embedding):

$$\begin{aligned} \phi(\cdot): \mathbb{R}^D &\to \mathbb{R}^M, && (2.93)\\ \boldsymbol{x} &\mapsto \phi(\boldsymbol{x}), && (2.94) \end{aligned}$$

such that the structure of the resulting data $\{\phi(\boldsymbol{x}_i)\}_{i=1}^N$ becomes (significantly more) linear. In machine learning, $\phi(\boldsymbol{x}) \in \mathbb{R}^M$ is called the "feature" of the data point $\boldsymbol{x} \in \mathbb{R}^D$, and the space $\mathbb{R}^M$ is called the "feature space."

Let $\bar{\phi} = \frac{1}{N}\sum_{i=1}^N \phi(\boldsymbol{x}_i)$ be the sample mean in the feature space and define the matrix $\Phi \doteq [\phi(\boldsymbol{x}_1) - \bar{\phi}, \phi(\boldsymbol{x}_2) - \bar{\phi}, \ldots, \phi(\boldsymbol{x}_N) - \bar{\phi}] \in \mathbb{R}^{M \times N}$. The principal components in the feature space are given by the eigenvectors of the sample covariance matrix[12]

$$\Sigma_{\phi(\boldsymbol{x})} \doteq \frac{1}{N-1}\sum_{i=1}^{N}(\phi(\boldsymbol{x}_i) - \bar{\phi})(\phi(\boldsymbol{x}_i) - \bar{\phi})^\top = \frac{1}{N-1}\Phi\Phi^\top \in \mathbb{R}^{M\times M}. \tag{2.95}$$

Let $v_i \in \mathbb{R}^M$, $i = 1, \ldots, M$, be the M eigenvectors, i.e.,

$$\Sigma_{\phi(\boldsymbol{x})} v_i = \lambda_i v_i, \quad i = 1, 2, \ldots, M. \tag{2.96}$$

Then the d "nonlinear principal components" of every data point $\boldsymbol{x}$ are given by

$$y_i \doteq v_i^\top (\phi(\boldsymbol{x}) - \bar{\phi}) \in \mathbb{R}, \quad i = 1, 2, \ldots, d. \tag{2.97}$$

Unfortunately, the map $\phi(\cdot)$ is generally not known beforehand and searching for the proper map is a difficult task. In such cases, the use of nonlinear PCA becomes limited. However, in some practical applications, good candidates for the map $\phi(\cdot)$ can be found from the nature of the problem. In such cases, the map, together with PCA, can be very effective in extracting the overall geometric structure of the data.

Example 2.6 (Veronese Map for an Arrangement of Subspaces). As we will see later in this book, if the data points belong to a union of multiple subspaces, then a natural choice of the transformation $\phi(\cdot)$ is the Veronese map:

$$\begin{aligned} \nu_n(\cdot): \quad \boldsymbol{x} &\mapsto \nu_n(\boldsymbol{x}),\\ (x_1, \ldots, x_D) &\mapsto (x_1^n, x_1^{n-1}x_2, \ldots, x_D^n), \end{aligned}$$

where the monomials are ordered in the degree-lexicographic order. Under such a mapping, the multiple low-dimensional subspaces are mapped into a single subspace in the feature space, which can then be identified via PCA for the features. ■

NLPCA in a High-dimensional Feature Space.

A potential difficulty associated with nonlinear PCA is that the dimension of the feature space, M, can be very high. Thus computing the principal components

[12]In principle, we should use the notation $\hat{\Sigma}_{\phi(\boldsymbol{x})}$ to indicate that it is the estimate of the actual covariance matrix. But for simplicity, we will drop the hat in the sequel and simply use $\Sigma_{\phi(\boldsymbol{x})}$. The same goes for the eigenvectors and the principal components.

in the feature space may become computationally prohibitive. For instance, if we use a Veronese map of degree n, the dimension of the feature space M grows exponentially with the degree. When M exceeds N, the eigenvalue decomposition of $\Phi\Phi^\top \in \mathbb{R}^{M\times M}$ becomes more costly than that of $\Phi^\top\Phi \in \mathbb{R}^{N\times N}$, although the two matrices have the same eigenvalues.

This motivates us to examine whether the computation of PCA in the feature space can be reduced to a computation with the lower-dimensional matrix $\Phi^\top\Phi$. The answer is actually yes. The key is to notice that, despite the dimension of the feature space, every eigenvector $v \in \mathbb{R}^M$ of $\Phi\Phi^\top$ associated with a non-zero eigenvalue is always in the span of the matrix Φ:[13]

$$\Phi\Phi^\top v = \lambda v \quad \Leftrightarrow \quad v = \Phi(\lambda^{-1}\Phi^\top v) \ \in \text{range}(\Phi). \tag{2.98}$$

We define the vector $w \doteq \lambda^{-1}\Phi^\top v \in \mathbb{R}^N$. Obviously $\|w\|^2 = \lambda^{-1}$. It is straightforward to check that w is an eigenvector of $\Phi^\top\Phi$ for the same eigenvalue λ. Once such a w is computed from $\Phi^\top\Phi$, we can recover the corresponding v in the feature space as:

$$v = \Phi w. \tag{2.99}$$

Therefore the d nonlinear principal component of $\boldsymbol{x}$ under the map $\phi(\cdot)$ can be computed as:

$$y_i \doteq v_i^\top(\phi(\boldsymbol{x}) - \bar{\phi}) = w_i^\top\Phi^\top(\phi(\boldsymbol{x}) - \bar{\phi}) \ \in \mathbb{R}, \quad i = 1, \ldots, d, \tag{2.100}$$

where $w_i \in \mathbb{R}^N$ is the ith leading eigenvector of $\Phi^\top\Phi \in \mathbb{R}^{N\times N}$.

Kernel PCA

A very interesting property of the above NLPCA method is that the computation of the nonlinear principal components involves only inner products of the features. More specifically, in order to compute the nonlinear principal components, y_i, we simply need to compute the entries of the matrix $\Phi^\top\Phi$ and the entries of the vectors $\Phi^\top\phi(\boldsymbol{x})$ and $\Phi^\top\bar{\phi} = \frac{1}{N}\sum\Phi^\top\phi(\boldsymbol{x}_i)$, all of which can be obtained from inner products of the form $\phi(\boldsymbol{x})^\top\phi(\boldsymbol{y})$, as we will show next.

Define the "kernel function" of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D$ to be the inner product of their features

$$k(\boldsymbol{x}, \boldsymbol{y}) \doteq \phi(\boldsymbol{x})^\top\phi(\boldsymbol{y}) \ \in \mathbb{R}. \tag{2.101}$$

The so-defined function $k(\cdot,\cdot)$ is a symmetric positive semi-definite function in $\boldsymbol{x}$ and $\boldsymbol{y}$,[14] which can be used to compute the nonlinear principal components as follows. Define a *kernel matrix* $K \in \mathbb{R}^{N\times N}$ as $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The entries of

[13]The remaining $M - N$ eigenvectors of $\Phi\Phi^\top$ are associated with the eigenvalue zero.

[14]A function $k(\boldsymbol{x}, \boldsymbol{y})$ is positive semi-definite if $\int\int_{\mathbb{R}^D} f(\boldsymbol{x})k(\boldsymbol{x}, \boldsymbol{y})f(\boldsymbol{y})\, d\boldsymbol{x}d\boldsymbol{y} \geq 0$ for all square-integrable functions $f(\cdot)$.

the matrix $\mathcal{K} = \Phi^\top \Phi$ can be computed as

$$\mathcal{K}_{ij} = (\Phi^\top \Phi)_{ij} = (\phi(\boldsymbol{x}_i) - \bar{\phi})^\top (\phi(\boldsymbol{x}_j) - \bar{\phi}) \tag{2.102}$$

$$= k_{ij} - \frac{1}{N}\sum_j k_{ij} - \frac{1}{N}\sum_i k_{ij} + \frac{1}{N^2}\sum_i \sum_j k_{ij}, \tag{2.103}$$

or in matrix notation

$$\mathcal{K} = K - \frac{1}{N}K\mathbf{1}\mathbf{1}^\top - \frac{1}{N}\mathbf{1}\mathbf{1}^\top K + \frac{\mathbf{1}^\top K \mathbf{1}}{N^2}\mathbf{1}\mathbf{1}^\top \tag{2.104}$$

$$= (I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top)K(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top). \tag{2.105}$$

The matrix $I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$ is called the centering matrix, since it makes the
The vectors w_i are then eigenvectors of $\mathcal{K}$ associated with its top d eigenvalues. Now, the entries of the vector $\Phi^\top(\phi(\boldsymbol{x}) - \bar{\phi})$ can be computed as

$$(\Phi^\top(\phi(\boldsymbol{x}) - \bar{\phi}))_i = (\phi(\boldsymbol{x}_i) - \bar{\phi})^\top(\phi(\boldsymbol{x}) - \bar{\phi}) \tag{2.106}$$

$$= k(\boldsymbol{x}_i, \boldsymbol{x}) - \frac{1}{N}\sum_j k_{ij} - \frac{1}{N}\sum_i k(\boldsymbol{x}_i, \boldsymbol{x}) + \frac{1}{N^2}\sum_i\sum_j k_{ij}, \tag{2.107}$$

or in vector notation

$$\Phi^\top(\phi(\boldsymbol{x}) - \bar{\phi}) = k_{\boldsymbol{x}} - \frac{1}{N}K\mathbf{1} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top k_{\boldsymbol{x}} + \frac{\mathbf{1}^\top K\mathbf{1}}{N^2}\mathbf{1}, \tag{2.108}$$

where $k_{\boldsymbol{x}} = [k(\boldsymbol{x}_1, \boldsymbol{x}), k(\boldsymbol{x}_2, \boldsymbol{x}), \cdots, k(\boldsymbol{x}_N, \boldsymbol{x})]^\top \in \mathbb{R}^N$. The nonlinear principal components are then given by

$$y_i = w_i^\top k_{\boldsymbol{x}} - \frac{w_i^\top K\mathbf{1}}{N} - \frac{w_i^\top \mathbf{1}\mathbf{1}^\top k_{\boldsymbol{x}}}{N} + \frac{\mathbf{1}^\top K\mathbf{1}}{N^2}w_i^\top \mathbf{1}. \tag{2.109}$$

In the particular case where the data is zero-mean, i.e., $\bar{\phi} = \mathbf{0}$, we simply have

$$\mathcal{K} = K, \quad \mathbf{1}^\top k_{\boldsymbol{x}} = 0, \quad K\mathbf{1} = \mathbf{0} \quad \text{and} \quad y_i = w_i^\top k_{\boldsymbol{x}}, \quad i = 1, \ldots, d. \tag{2.110}$$

If follows from the analysis above that the nonlinear principal components can be computed directly from the kernel function $k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})$. Therefore, we may be able to avoid having to compute $\phi(\boldsymbol{x})$ whenever an expression for the kernel k is known. For instance, in the conic example in (2.91), we have

$$k(\boldsymbol{x}, \boldsymbol{y}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2][y_1^2, \sqrt{2}y_1y_2, y_2^2]^\top = (x_1y_1 + x_2y_2)^2 = (\boldsymbol{x}^\top \boldsymbol{y})^2, \tag{2.111}$$

which can be computed directly in $\mathbb{R}^2$ without need to resort to computing the embedding into $\mathbb{R}^3$.

In general, we do not need to explicitly define and evaluate the map $\phi(\cdot)$. In fact, given any (positive-definite) kernel function, according to a fundamental result in functional analysis, one can in principle decompose the kernel and recover the associated map $\phi(\cdot)$ if one wishes to.

Theorem 2.7 (Mercer's Theorem)**.** *Suppose $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a symmetric real valued function such that for some $C > 0$ and almost every $(\boldsymbol{x}, \boldsymbol{y})$*[15] *we have $|k(\boldsymbol{x}, \boldsymbol{y})| \leq C$. Suppose that the linear operator $\mathcal{L} : L^2(\mathbb{R}^D) \to L^2(\mathbb{R}^D)$,*

$$\mathcal{L}(f)(\boldsymbol{x}) \doteq \int_{\mathbb{R}^D} k(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{y}) d\boldsymbol{y}, \tag{2.112}$$

is positive semi-definite. Let ψ_i be the normalized orthogonal eigenfunctions of $\mathcal{L}$ associated with the eigenvalues $\lambda_i > 0$, sorted in non-increasing order, and let M be the number of nonzero eigenvalues. Then

- *The sequence of eigenvalues is absolutely convergent, i.e., $\sum_{i=1}^{M} |\lambda_i| < \infty$.*
- *The kernel k can be expanded as $k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{M} \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{y})$ for almost all $(\boldsymbol{x}, \boldsymbol{y})$.*

The interested readers may refer to [Mercer, 1909] for a proof of the theorem. It follows from the theorem that, given a positive semi-definite kernel k, we can always associate with it an embedding function ϕ as

$$\phi_i(\boldsymbol{x}) = \sqrt{\lambda_i} \psi_i(\boldsymbol{x}) \quad i = 1, \ldots M. \tag{2.113}$$

Notice that the dimension of the embedding, M, could be rather large, sometimes even infinity. Nevertheless, an important reason for computing with the kernel function is that we do not need to compute the embedding function or the features. Instead, we simply evaluate the dot products $k(\boldsymbol{x}, \boldsymbol{y})$ in the original space $\mathbb{R}^D$.

Example 2.8 (Examples of Kernels). There are several popular choices for the nonlinear kernel function, such as the polynomial kernel and the Gaussian kernel, respectively,

$$k_P(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y})^n \quad \text{and} \quad k_G(\boldsymbol{x}, \boldsymbol{y}) = \exp\big(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2}\big). \tag{2.114}$$

Evaluation of such functions only involves the inner product or the difference between two vectors in the original space $\mathbb{R}^D$. This is much more efficient than evaluating the inner product in the associated feature space, whose dimension for the first kernel grows exponentially with the degree n and for the second kernel is infinite. ■

We summarize our discussion in this section as Algorithm 2.1.

2.5.4 *Locally Linear Embedding*

2.6 Bibliographic Notes

As a matrix decomposition tool, SVD was initially developed independently from PCA in the numerical linear algebra literature, also known as the Erkart and Young decomposition [Eckart and Young, 1936, Hubert et al., 2000]. The result regarding the least-squares optimality of SVD given in Theorem 2.2 can

[15] "Almost every" means except for a set of measure zero.

Algorithm 2.1 (Nonlinear Kernel PCA).

For a given set of zero-mean data points $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$, and a given map $\phi(\boldsymbol{x})$ or a kernel function $k(\boldsymbol{x}, \boldsymbol{y})$ such that $\phi(\mathbf{0}) = \mathbf{0}$ or $k(\mathbf{0}, \mathbf{0}) = 0$,

1. Compute the inner product matrix
$$\Phi^\top \Phi \;=\; \big(\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)\big) \;\text{ or }\; \big(k(\boldsymbol{x}_i, \boldsymbol{x}_j)\big) \;\in \mathbb{R}^{N \times N}; \tag{2.115}$$
2. Compute the eigenvectors $w_i \in \mathbb{R}^N$ of $\Phi^\top \Phi$:
$$\Phi^\top \Phi w_i = \lambda_i w_i, \tag{2.116}$$
and normalize $\|w_i\|^2 = \lambda_i^{-1}$;
3. For any data point $\boldsymbol{x}$, its ith nonlinear principal component is given by
$$y_i \;=\; w_i^\top \Phi^\top \phi(\boldsymbol{x}) \;\text{ or }\; w_i^\top [k(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x})]^\top, \tag{2.117}$$
for $i = 1, 2, \ldots, d$.

---

be traced back to [Householder and Young, 1938, Gabriel, 1978]. While principal components were initially defined exclusively in a statistical sense [Pearson, 1901, Hotelling, 1933], one can show that the algebraic solution given by SVD gives asymptotically unbiased estimates of the true parameters in the case of Gaussian distributions. A more detailed analysis of the statistical properties of PCA can be found in [Jolliffe, 2002].

Note that PCA only infers the principal subspace (or components), but not a probabilistic distribution of the data in the subspace. Probabilistic PCA was developed to infer an explicit probabilistic distribution from the data [Tipping and Bishop, 1999b]. The data is assumed to be independent samples drawn from an unknown distribution, and the problem becomes one of identifying the subspace and the parameters of the distribution in a maximum-likelihood or a maximum-a-posteriori sense. When the underlying noise distribution is Gaussian, the geometric and probabilistic interpretations of PCA coincide [Collins et al., 2001]. However, when the underlying distribution is non Gaussian, the optimal solution to PPCA may no longer be linear. For example, in [Collins et al., 2001] PCA is generalized to arbitrary distributions in the exponential family.

PCA is obviously not applicable to data whose underlying structure is nonlinear. PCA was generalized to principal curves and surfaces by [Hastie, 1984] and [Hastie and Stuetzle, 1989]. A more general approach however is to find a nonlinear embedding map, or equivalently a kernel function, such that the embedded data would lie on a linear subspace. Such methods are referred to as nonlinear kernel PCA [Scholkopf et al., 1998]. Finding such nonlinear maps or kernels is by no means a simple problem. Learning kernels is still an active research topic in the statistical learning community.

2.7 Exercises

Exercise 2.1 (Some Properties of PCA). Let $\boldsymbol{x}$ be a random vector with covariance matrix $\Sigma_{\boldsymbol{x}}$. Consider a linear transformation of $\boldsymbol{x}$:

$$\boldsymbol{y} = W^\top \boldsymbol{x}, \tag{2.118}$$

where $\boldsymbol{y} \in \mathbb{R}^d$ and W is a $D \times d$ orthogonal matrix. Let $\Sigma_{\boldsymbol{y}} = W^\top \Sigma_{\boldsymbol{x}} W$ be the covariance matrix for $\boldsymbol{y}$. Show that

1. The trace of $\Sigma_{\boldsymbol{y}}$ is maximized by $W = U_d$, where U_d consists of the first d (normalized) eigenvectors of $\Sigma_{\boldsymbol{x}}$.
2. The trace of $\Sigma_{\boldsymbol{y}}$ is minimized by $W = \tilde{U}_d$, where $\tilde{U}_d$ consists of the last d (normalized) eigenvectors of $\Sigma_{\boldsymbol{x}}$.

Exercise 2.2 (Subspace Angles). Given two d-dimensional subspaces S_1 and S_2 in $\mathbb{R}^D$, define the largest subspace angle θ_1 between S_1 and S_2 to be the largest possible sharp angle ($< 90^\circ$) formed by any two vectors $u_1, u_2 \in (S_1 \cap S_2)^\perp$ with $u_1 \in \boldsymbol{S}_1$ and $u_2 \in S_2$ respectively. Let $U_1 \in \mathbb{R}^{D\times d}$ be an orthogonal matrix whose columns form a basis for S_1 and similarly U_2 for S_2. Then show that if σ_1 is the smallest non-zero singular value of the matrix $W = U_1^\top U_2$, then we have

$$\cos(\theta_1) = \sigma_1. \tag{2.119}$$

Similarly, one can define the rest of the subspace angles as $\cos(\theta_i) = \sigma_i, i = 2, \ldots, d$ from the rest of the singular values of W.

Exercise 2.3 (Fixed-Rank Approximation of a Matrix). Given an arbitrary full-rank matrix $A \in \mathbb{R}^{m\times n}$, find the matrix $B \in \mathbb{R}^{m\times n}$ with a fixed rank $r < \min\{m, n\}$ such that the Frobenius norm $\|A - B\|_F$ is minimized. The Frobenius norm of a matrix M is defined to be $\|M\|_F^2 = \text{trace}(M^T M)$. (Hint: Use the SVD of A to guess the matrix B and then prove its optimality.)

Exercise 2.4 (Identification of Auto-Regressive Exogeneous (ARX) Systems). A popular model that is often used to analyze a time series $\{y_t\}_{t\in\mathbb{Z}}$ is the linear auto-regressive model:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_n y_{t-n} + \varepsilon_t, \quad \forall t, y_t \in \mathbb{R}, \tag{2.120}$$

where $\varepsilon_t \in \mathbb{R}$ models the modeling error or noise and it is often assumed to be a white-noise random process. Now suppose that you are given the values of y_t for a sufficiently long period of time.

1. Show that in the noise free case, i.e. $\varepsilon_t \equiv 0$, regardless of the initial conditions, the vectors $\boldsymbol{x}_t = [y_t, y_{t-1}, \ldots, y_{t-n}]^T$ for all t lie on an n-dimensional hyperplane in $\mathbb{R}^{n+1}$. What is the normal vector to this hyperplane?
2. Now consider the case with noise. Describe how you may use PCA to identify the unknown model parameters $(a_1, a_2, \ldots, a_n)$?

Exercise 2.5 (Basis for an Image). Given a gray-level image $\boldsymbol{I}$, consider all of its $b \times b$ blocks, denoted as $\{B_i \in \mathbb{R}^{b\times b}\}$. We would like to approximate each block as a superposition of d base blocks, say $\{\hat{B}_j \in \mathbb{R}^{b\times b}\}_{j=1}^{d}$. That is,

$$B_i = \sum_{j=1}^{d} a_{ij}\hat{B}_j + E_i, \tag{2.121}$$

where $E_i \in \mathbb{R}^{b\times b}$ is the possible residual from the approximation. Describe how you can use PCA to identify an optimal set of d base blocks so that the residual is minimized?

In Section 1.2.1, we have seen an example in which a similar process can be applied to an ensemble of face images, where the first $d = 3$ principal components are computed for further classification. In the computer vision literature, the corresponding base images are called "eigen faces."

Exercise 2.6 (Probability of Selecting a Subset of Inliers). Imagine we have 80 samples from a four-dimensional subspace in $\mathbb{R}^5$. However, the samples are contaminated with another 20 samples that are far from the subspace. We want to estimate the subspace from randomly drawn subsets of four samples. In order to draw a subset that only contains inliers with probability 0.95, what is the smallest number of subsets that we need to draw?

Exercise 2.7 (Ranking of Webpages). PCA is actually used to rank webpages on the Internet by many popular search engines. One way to see this is to view the Internet as a directed graph $G = (V, E)$, where every webpage, denoted as p_i, is a node in V, and every hyperlink from p_i to p_j, denoted as e_{ij}, are directed edges in E. We can assign each webpage p_i an "authority" score x_i that indicates how many other webpages point to it and a "hub" score y_i that indicates how many other webpages it points out to. Then, the authority score x_i depends on how many hubs point to p_i and the hub score y_i depends on how many authorities p_i points to. Let L be the adjacent matrix of the graph G (i.e. $L_{ij} = 1$ if $e_{ij} = E$), $\boldsymbol{x}$ the vector of the authority scores and $\boldsymbol{y}$ of the hub scores.

1. Justify that the following relationships hold:
$$\boldsymbol{y}' = L\boldsymbol{x}, \quad \boldsymbol{x}' = L^T\boldsymbol{y}; \quad \boldsymbol{x} = \boldsymbol{x}'/\|\boldsymbol{x}'\|, \quad \boldsymbol{y} = \boldsymbol{y}'/\|\boldsymbol{y}'\|. \tag{2.122}$$
2. Show that $\boldsymbol{x}$ is the eigenvector of L^TL and $\boldsymbol{y}$ is the eigenvector of LL^T associated with the largest eigenvalue (why not the others). Explain how $\boldsymbol{x}$ and $\boldsymbol{y}$ can be computed from the singular value decomposition of L.

In the literature, this is known as the *Hypertext Induced Topic Selection* (HITS) algorithm [Kleinberg, 1999, Ding et al., 2004]. In fact, the same algorithm can also be used to rank any competitive sports such as football teams and chess players.

Exercise 2.8 (Karhunen-Loève Transform). The Karhunen-Loève transform (KLT) can be thought as a generalization of PCA from a (finite-dimensional) random vector $\boldsymbol{x} \in \mathbb{R}^D$ to an (infinite-dimensional) random process $x(t), t \in \mathbb{R}$. When $x(t)$ is a (zero-mean) second-order stationary random process, its auto correlation function is defined to be $K(t,\tau) \doteq E[x(t)x)(\tau)]$ for all $t, \tau \in \mathbb{R}$.

1. Show that $K(t,\tau)$ has a family of orthonormal eigen-functions $\{\phi_i(t)\}_{i=1}^{\infty}$ that are defined as
$$\int K(t,\tau)\phi_i(\tau)\,d\tau = \lambda_i\phi_i(t), \quad i = 1, 2, \ldots. \tag{2.123}$$
(Hint: First show that $K(t,\tau)$ is a positive definite function and then use Mercer's Theorem.)

2. Show that with respect to the eigen-functions, we original random process can be decomposed as

$$x(t) = \sum_{i=1}^{n} x_i \phi_i(t), \tag{2.124}$$

where $\{x_i\}_{i=1}^{\infty}$ are a set of uncorrelated random variables.

Exercise 2.9 (Full Rank of Gaussian RBF Gram Matrices) Suppose that you are given N distinct points $\{\boldsymbol{x}_i\}_{i=1}^{N}$. If $\sigma \neq 0$, then the matrix $K \in \mathbb{R}^{N \times N}$ given by

$$K_{ij} = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \tag{2.125}$$

has full rank.

Chapter 3
Algebraic Methods for Multiple-Subspace Segmentation

"The art of doing mathematics consists in finding that special case which contains all the germs of generality."
– David Hilbert

In this chapter, we consider a generalization of PCA in which the given sample points are drawn from an unknown arrangement of subspaces of unknown and possibly different dimensions. We first present a series of simple examples that demonstrate that the subspace-segmentation problem can be solved non-iteratively via certain algebraic methods. These solutions lead to a general-purpose algebro-geometric algorithm for subspace segmentation. We conveniently refer to the algorithm as Generalized Principal Component Analysis (GPCA). To better isolate the difficulties in the general problem, we will develop the algorithm in two steps. The first step is to develop a basic GPCA algorithm by assuming a known number of subspaces; and in the second step, we deal with an unknown number of subspaces and develop a recursive version of the GPCA algorithm. The algorithms in this chapter will be derived under ideal noise-free conditions and assume no probabilistic model. Nevertheless, the algebraic techniques involved are numerically well-conditioned and the algorithms are designed to tolerate moderate amounts of noise. Dealing with large amounts of noise or even outliers will be the subject of Chapter **??**.

In order to make the material accessible to a larger audience, in this chapter we focus primarily on the development of a (conceptual) algorithm. We leave a more formal study of subspace arrangements and rigorous justifications of all the algebraic facts that support the algorithms of this chapter to Appendix C.

3.1 Problem Formulation of Subspace Segmentation

In mathematics (especially in algebraic geometry), a collection of subspaces is formally known as a subspace arrangement:

Definition 3.1 (Subspace Arrangement)**.** *A* subspace arrangement *is defined as a finite collection of n linear subspaces in R^D: $\mathcal{A} \doteq \{S_1, \ldots, S_n\}$. The union of the subspaces is denoted as $Z_{\mathcal{A}} \doteq S_1 \cup S_2 \cup \cdots \cup S_n$.*

For simplicity, we will use the term "subspace arrangement" to refer to both $\mathcal{A}$ and $Z_{\mathcal{A}}$.

Imagine that we are given a set of sample points drawn from an arrangement of unknown number of subspaces which have unknown and possibly different dimensions. Our goal is to simultaneously estimate these subspaces and segment the points into their corresponding subspaces. Versions of this problem are known in the literature as *subspace clustering*, *multiple eigenspaces* [Leonardis et al., 2002], or *mixtures of principal component analyzers* [Tipping and Bishop, 1999a], etc. To be precise, we will first state the problem that we will study in this book, which we refer to as "multiple-subspace segmentation," or simply as "subspace segmentation," to be suggestive of the problem of fitting multiple (principal) subspaces to the data.

Problem 3.1 (Multiple-Subspace Segmentation).

Given a set of sample points $\boldsymbol{X} = \{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^N$ drawn from $n \geq 1$ distinct linear subspaces $S_j \subset \mathbb{R}^D$ of dimensions $d_j < D$, $j = 1, 2, \ldots, n$, identify each subspace S_j without knowing which sample points belong to which subspace. More specifically, by identifying the subspaces we mean the following:

1. Identifying the number of subspaces n and their dimensions $d_j = \dim(S_j)$;
2. Identifying an orthonormal basis for each subspace S_j (or equivalently a basis for its orthogonal complement $S_j^{\perp}$);
3. Clustering the N points into the subspaces to which they belong.

Notice that in the foregoing problem statement, we have not yet specified the objective for what is an "optimal" solution. We will leave the interpretation of that open for now and will delay the definition until the context is more specific. Although the problem seems to be stated in a purely geometric fashion, it is easy to re-formulate it in a statistical fashion. For instance, we have assumed here that the subspaces do not have to be orthogonal to each other. In a statistical setting, this is essentially equivalent to assuming that these subspaces are not necessarily uncorrelated. Within each subspace, one can also relate all the geometric and statistical notions associated with "principal components" in the classical PCA: The orthonormal basis chosen for each subspace usually corresponds to a decomposition of the random variable into uncorrelated principal components *conditioned* *on the subspace*. In Section 4.2, a detailed analysis and comparison will be given for both points of view.

3.1.1 Projectivization of Affine Subspaces

Note that a linear subspace always passes through the origin but an affine subspace does not. So, would the above problem statement lose any generality by restricting it only to linear subspaces? The answer to this question is no. In fact every proper affine subspace in $\mathbb{R}^D$ can be converted to a proper linear subspace in $\mathbb{R}^{D+1}$ by lifting every point of it through the so-called homogeneous coordinates:

Definition 3.2 (Homogeneous Coordinates)**.** *The homogeneous coordinates of a point* $\boldsymbol{x} = [x_1, x_2, \ldots, x_D]^T \in \mathbb{R}^D$ *are defined as* $[x_1, x_2, \ldots, x_D, 1]^T$.

Given a set of points in an affine subspace, it is easy to prove that their homogeneous coordinates span a linear subspace. More precisely:

Fact 3.3 (Homogeneous Representation of Affine Subspaces)**.** *The homogeneous coordinates of points on a* k*-dimensional affine subspace in* $\mathbb{R}^D$ *span a* $(d+1)$*-dimensional linear subspace in* $\mathbb{R}^{D+1}$*. This representation is one-to-one.*

Figure 3.1 shows an example of the homogeneous representation of three lines in $\mathbb{R}^2$. The points on these lines span three linear subspaces in $\mathbb{R}^3$ which pass through the origin.

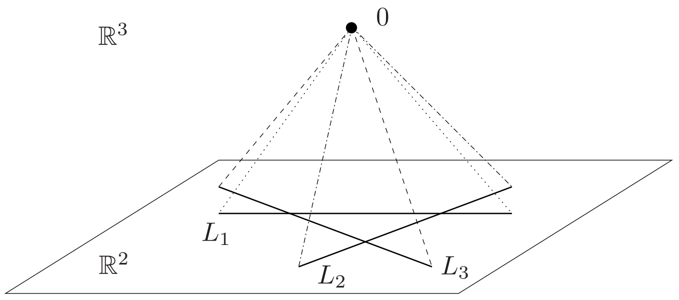


Figure 3.1. Lifting of three (affine) lines in $\mathbb{R}^2$ to three linear subspaces in $\mathbb{R}^3$ via the homogeneous representation.

Definition 3.4 (Central Subspace Arrangements)**.** *We say an arrangement of subspaces is* central *if every subspace passes through the origin, i.e., every subspace is a linear subspace.*

According to this definition, the homogeneous representation of any (affine) subspace arrangement in $\mathbb{R}^D$ gives a central subspace arrangement in $\mathbb{R}^{D+1}$. Therefore, Problem 3.1 does not loss any generality. From now on, we may assume that our data set is drawn from a central subspace arrangement, in which all subspaces are linear, not affine, subspaces, unless otherwise stated. In a statistical setting, this is equivalent to assuming that each subset of samples has zero mean.

3.1.2 Subspace Projection and Minimum Representation

The are many cases in which the given data points live in a very high dimensional space. For instance, in many computer vision problems the dimension of the ambient space D is the number of pixels in an image, which is normally in the range 10^6. In such cases, the complexity of any subspace segmentation solution becomes computationally prohibitive. It is therefore important for us to seek situations in which the dimension of the ambient space can be significantly reduced.

Fortunately, in most practical applications, we are interested in modeling the data by subspaces of relatively small dimensions ($d \ll D$), thus one can avoid dealing with high-dimensional data sets by first projecting them onto a lower-dimensional (sub)space. An example is shown in Figure 3.2, where two lines L_1 and L_2 in $\mathbb{R}^3$ are projected onto a plane P. In this case, segmenting the two lines in the three-dimensional space $\mathbb{R}^3$ is equivalent to segmenting the two projected lines in the two-dimensional plane P.

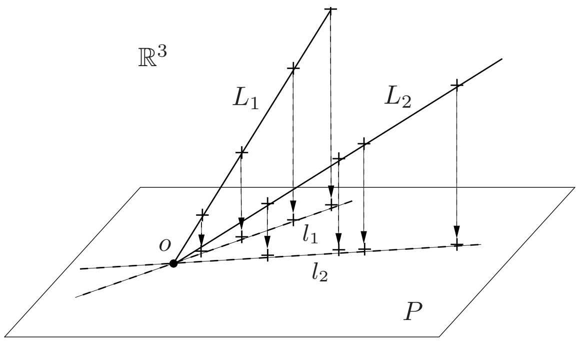


Figure 3.2. Samples on two 1-dimensional subspaces L_1, L_2 in $\mathbb{R}^3$ projected onto a 2-dimensional plane P. The number and separation of the lines is preserved by the projection.

In general, we will distinguish between two different kinds of "projections." The first kind corresponds to the case in which the span of all the subspaces is a proper subspace of the ambient space, i.e., span$(\cup_{j=1}^{n} S_j) \subset \mathbb{R}^D$. In this case, one may simply apply PCA (Chapter 2) to eliminate the redundant dimensions. The second kind corresponds to the case in which the largest dimension of the subspaces, denoted by $d_{\max}$, is strictly less than $D-1$. When $d_{\max}$ is known,[1] one may choose a $(d_{\max}+1)$-dimensional subspace P such that, by projecting $\mathbb{R}^D$ onto this subspace:

$$\pi_P : \boldsymbol{x} \in \mathbb{R}^D \quad \mapsto \quad \boldsymbol{x}' = \pi_P(\boldsymbol{x}) \in P, \tag{3.1}$$

[1]For example, in 3-D motion segmentation from affine cameras, it is known that the subspaces have dimension at most four [Costeira and Kanade, 1998, Kanatani, 2001, Vidal and Hartley, 2004].

the dimension of each original subspace S_j is preserved,[2] and there is a one-to-one correspondence between S_j and its projection – no reduction in the number of subspaces n,[3] as stated in the following theorem.

Theorem 3.5 (Segementation-Preserving Projections)**.** *If a set of vectors $\{\boldsymbol{x}_i\}$ all lie in n linear subspaces of dimensions $\{d_j\}_{j=1}^n$ in $\mathbb{R}^D$, and if π_P represents a linear projection onto a subspace P of dimension D', then the points $\{\pi_P(\boldsymbol{x}_i)\}$ lie in at most n linear subspaces of P of dimensions $\{d'_j \leq d_j\}_{j=1}^n$. Furthermore, if $D > D' > d_{\max}$, then there is an open and dense set of projections that preserve the separation and dimensions of the subspaces.*

Thanks to Theorem 3.5, if we are given a data set $\boldsymbol{X}$ drawn from an arrangement of low-dimensional subspaces in a high-dimensional space, we can first project $\boldsymbol{X}$ onto a generic subspace of dimension $D' = d_{\max} + 1$ and then model the data with a subspace arrangement in the projected subspace, as illustrated by the following sequence of steps:

$$\boldsymbol{X} \subset \mathbb{R}^D \xrightarrow{\pi_P} \boldsymbol{X}' \subset P \longrightarrow \cup_{j=1}^n \pi_P(S_j) \xrightarrow{\pi_P^{-1}} \cup_{j=1}^n S_j. \tag{3.2}$$

However, even though the set of $(d_{\max}+1)$-dimensional subspaces $P \subset \mathbb{R}^D$ that preserve the separation and dimension of the subspaces is an open and dense set, it remains unclear as to what a "good" choice for P is, especially when there is noise in the data. For simplicity, one may randomly select a few projections and choose the one that results in the smallest fitting error. Another alternative is to apply PCA regardless and project the data onto the $(d_{\max}+1)$-dimensional principal subspace.

One solution for choosing P is attributed to [Broomhead and Kirby, 2000]. The technique was originally designed for dimension reduction of differential manifolds.[4] We here adopt it for subspace arrangements. Instead of directly using the original data matrix $\boldsymbol{X}$, we gather the vectors (also called "secants") defined by every pair of points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{X}$

$$\boldsymbol{y}_{ij} \doteq \boldsymbol{x}_i - \boldsymbol{x}_j \quad \in \mathbb{R}^D, \tag{3.3}$$

and construct a matrix consisting of $\boldsymbol{y}_{ij}$ as columns:

$$\boldsymbol{Y} \doteq [\boldsymbol{y}_{12}, \boldsymbol{y}_{13}, \ldots, \boldsymbol{y}_{(N-1)N}] \quad \in \mathbb{R}^{D\times M}, \tag{3.4}$$

[2]This requires that P be transversal to each $S_j^\perp$, i.e., span$\{P, S_j^\perp\} = \mathbb{R}^D$ for every $j = 1, 2, \ldots, n$. Since n is finite, this transversality condition can be easily satisfied. Furthermore, the set of positions for P which violate the transversality condition is only a zero-measure closed set [Hirsch, 1976].

[3]This requires that all $\pi_P(S_j)$ be transversal to each other in P, which is guaranteed if we require P to be transversal to $S_j^\perp \cap S_{j'}^\perp$ for $j, j' = 1, 2, \ldots, n$. All P's which violate this condition form again only a zero-measure set.

[4]That is essentially based on Whitney's classic proof of the fact any differential manifold can be embedded in a Euclidean space.

where $M = (N-1)N/2$. Then the principal components of $\boldsymbol{Y}$ span the subspace in which the distance (and hence the separateness) between the projected points is preserved the most. Therefore, the optimal subspace that maximizes the separateness of the projected points is given by the $d_{\max}+1$ principal components of $\boldsymbol{Y}$. More precisely, if $\boldsymbol{Y} = U\Sigma V^T$ is the SVD of $\boldsymbol{Y}$, then the optimal subspace P is given by the first $d_{\max}+1$ columns of U.

3.2 Introductory Cases of Subspace Segmentation

Notice that, to apply the K-subspaces and EM algorithms, we need to know three things in advance: the number of subspaces, their dimensions, and initial estimates of the bases of the subspaces. In practice, this may not be the situation and many difficulties may arise. The optimizing process in both algorithms is essentially a local iterative descent scheme. If the initial estimates of the bases of the subspaces are far off from the global optimum, the process is likely to converge to a local minimum. More seriously, if the number of subspaces and their dimensions were wrong, the process might never converge or might converge to meaningless solutions. Furthermore, when the number and dimensions of the subspaces are unknown and the samples are noisy (or contaminated by outliers), model selection becomes a much more elusive problem as we have alluded to earlier in the introduction chapter.

In this and next few chapters, we will systematically address these difficulties and aim to arrive at global non-iterative solutions to subspace segmentation that require less or none of the above initial information. Before we delve into the most general case, we first examine, in this section, a few important special cases. The reason is two-fold: Firstly, many practical problems fall into these cases already and the simplified solutions can be directly applied; and secondly, the analysis of these special cases offers some insights into a solution to the general case.

3.2.1 Segmenting Points on a Line

Let us begin with an extremely simple clustering problem: clustering a collection of points $\{x_1, x_2, \ldots, x_N\}$ on the real line $\mathbb{R}$ around a collection of cluster centers $\{\mu_1, \mu_2, \ldots, \mu_n\}$. In spite of its simplicity, this problem shows up in various segmentation problems. For instance, in intensity-based image segmentation, one wants to separate the pixels of an image into different regions, with each region corresponding to a significantly different level of intensity (a one-dimensional quantity). More generally, the point clustering problem is very much at the heart of spectral clustering, a popular technique for clustering data in spaces of any dimension. Furthermore, as we will see throughout this book, the same basic ideas introduced through this simple example can also be applied to clustering points from arrangements of more complex structures such as lines, hyperplanes, subspaces, and even surfaces.

In the sequel, we introduce a not so conventional solution to the point clustering problem. The new formulation that the solution is based on is neither geometric (like K-subspaces) nor statistical (like EM). Instead, the solution is purely *algebraic*.

Let $x \in \mathbb{R}$ be any of the data points. In an ideal situation in which each data point perfectly matches one of the cluster centers, we know that there exists a constant μ_j such that $x = \mu_j$. This means that

$$(x = \mu_1) \vee (x = \mu_2) \vee \cdots \vee (x = \mu_n). \tag{3.5}$$

The "$\vee$" in the preceding equation stands for the logical connective "or." This is equivalent to that x satisfies the following polynomial equation of degree n in x:

$$p_n(x) \doteq (x - \mu_1)(x - \mu_2) \cdots (x - \mu_n) = \sum_{k=0}^{n} c_k x^{n-k} = 0. \tag{3.6}$$

Since the polynomial equation $p_n(x) = 0$ must be satisfied by every data point, we have that

$$\boldsymbol{V}_n \, \boldsymbol{c}_n \doteq \begin{bmatrix} x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ x_2^n & x_2^{n-1} & \cdots & x_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_N^n & x_N^{n-1} & \cdots & x_N & 1 \end{bmatrix} \begin{bmatrix} 1 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = 0, \tag{3.7}$$

where $\boldsymbol{V}_n \in \mathbb{R}^{N\times(n+1)}$ is a matrix of embedded data points, and $\boldsymbol{c}_n \in \mathbb{R}^{n+1}$ is the vector of coefficients of $p_n(x)$.

In order to determine the number of groups n and then the vector of coefficients $\boldsymbol{c}_n$ from (3.7), notice that for n groups there is a unique polynomial of degree n whose roots are the n cluster centers. Since the coefficients of this polynomial must satisfy equation (3.7), in order to have a unique solution we must have that $\text{rank}(\boldsymbol{V}_n) = n$. This rank constraint on $\boldsymbol{V}_n \in \mathbb{R}^{N\times(n+1)}$ enables us to determine the number of groups n as[5]

$$n \doteq \min\{j : \text{rank}(\boldsymbol{V}_j) = j\}. \tag{3.8}$$

Example 3.6 (Two Clusters of Points). The intuition behind this formula is as follows. Consider, for simplicity, the case of $n = 2$ groups, so that $p_n(x) = p_2(x) = (x - \mu_1)(x - \mu_2)$, with $\mu_1 \neq \mu_2$. Then, it is clear that there is no polynomial equation of degree one, $p_1(x) = x - \mu$, that is satisfied by *all* the points. Similarly, there are infinitely many polynomial equations of degree 3 or more that are satisfied by all the points, namely any multiple of $p_2(x)$. Thus the degree $n = 2$ is the only one for which there is a unique polynomial that fits all the points. ■

[5]Notice that the minimum number of points needed is $N \geq n$, which is *linear* in the number of groups. We will see in future chapters that this is no longer the case for more general segmentation problems.

Once the minimum polynomial $p_n(x)$ that fits all the data points is found, we can solve the equation $p_n(x) = 0$ for its n roots. These roots, by definition, are the centers of the clusters. We summarize the overall solution as Algorithm 3.1.

Algorithm 3.1 (Algebraic Point Clustering Algorithm).

Let $\{x_1, x_2, \ldots, x_N\} \subset \mathbb{R}$ be a given collection of $N \geq n$ points clustering around an unknown number n of cluster centers $\{\mu_1, \mu_2, \ldots, \mu_n\}$. The number of groups, the cluster centers and the segmentation of the data can be determined as follows:

1. **Number of Groups**. Let $\boldsymbol{V}_j \in \mathbb{R}^{N \times (j+1)}$ be a matrix containing the last $j+1$ columns of $\boldsymbol{V}_n$. Determine the number of groups as
$$n \doteq \min\{j : \text{rank}(\boldsymbol{V}_j) = j\}.$$
2. **Cluster Centers.** Solve for $\boldsymbol{c}_n$ from $\boldsymbol{V}_n \boldsymbol{c}_n = 0$. Set $p_n(x) = \sum_{k=0}^{n} c_k x^{n-k}$. Find the cluster centers μ_j as the n roots of $p_n(x)$.
3. **Segmentation.** Assign point x_i to cluster $j = \arg\min_{l=1,\ldots,n}(x_i - \mu_l)^2$.

Notice that the above algorithm is described in a purely algebraic fashion and is more of a conceptual than practical algorithm. It does not minimize any geometric errors or maximize any probabilistic likelihood functions. In the presence of noise in the data, one has to implement each step of the algorithm in a numerically more stable and statistically more robust way. For example, with noisy data, the matrix $\boldsymbol{V}_n$ will most likely be of full rank. In this case, the vector of coefficients $\boldsymbol{c}_n$ should be solved in a least-squares sense as the singular-vector of $\boldsymbol{V}_n$ associated with the smallest singular value. It is also possible that the $p_n(x)$ obtained from $\boldsymbol{c}_n$ may have some complex roots, because the constraint that the polynomial must have real roots is never enforced when solving for the coefficients in the least-squares sense.[6] In practice, for well-separated clusters with moderate noise, the roots normally give decent estimates of the cluster centers.

Although clustering points on a line may seem a rather simple problem, it can be easily generalized to the problem of clustering points on a plane (see Exercise 3.1). Furthermore, it is also a key step of a very popular data clustering algorithm: *spectral clustering*. See Exercise 3.2.

3.2.2 *Segmenting Lines on a Plane*

Let us now consider the case of clustering data points to a collection of n lines in $\mathbb{R}^2$ passing through the origin, as illustrated in Figure 3.3. Each one of the lines

[6]However, in some special cases, one can show that this would never occur. For example, when $n = 2$, the least-squares solution for $\boldsymbol{c}_n$ is $c_2 = \text{Var}[x]$, $c_1 = E[x^2]E[x] - E[x^3]$ and $c_0 = E[x^3]E[x] - E[x^2]^2 \leq 0$, hence $c_1^2 - 4c_0c_2 \geq 0$ and the two roots of the polynomial $c_0x^2 + c_1x + c_2$ are always real.

can be represented as:

$$L_j \doteq \{\boldsymbol{x} = [x, y]^T : b_{j1}x + b_{j2}y = 0\}, \quad j = 1, 2, \ldots, n. \tag{3.9}$$

Given a point $\boldsymbol{x} = [x, y]^T$ in one of the lines we must have that

$$(b_{11}x + b_{12}y = 0) \vee \cdots \vee (b_{n1}x + b_{n2}y = 0). \tag{3.10}$$

Therefore, even though each individual line is described with one polynomial equation of degree one (a linear equation), an arrangement of n lines can be described with a polynomial of degree n, namely

$$p_n(\boldsymbol{x}) = (b_{11}x + b_{12}y) \cdots (b_{n1}x + b_{n2}y) = \sum_{k=0}^{n} c_k x^{n-k} y^k = 0. \tag{3.11}$$

An example is shown in Figure 3.3.

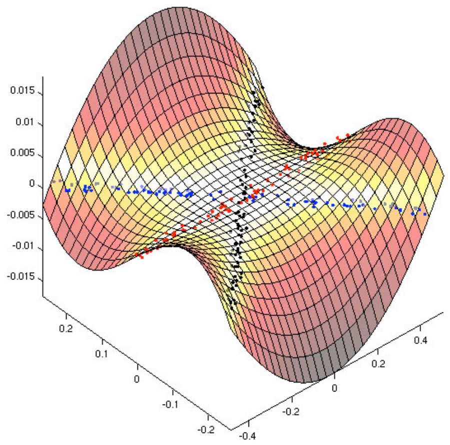


Figure 3.3. A polynomial in two variables whose zero set is three lines in $\mathbb{R}^2$.

The polynomial $p_n(\boldsymbol{x})$ allows us to algebraically eliminate the segmentation of the data at the beginning of the model estimation, because the equation $p_n(\boldsymbol{x}) = 0$ is satisfied by every data point regardless of whether it belongs to L_1, L_2, …, or L_n. Furthermore, even though $p_n(\boldsymbol{x})$ is nonlinear in each data point $\boldsymbol{x} = [x, y]^T$, $p_n(\boldsymbol{x})$ is actually linear in the vector of coefficients $\boldsymbol{c} = [c_0, c_1, \ldots, c_n]^T$. Therefore, given enough data points $\{\boldsymbol{x}_i = [x_i, y_i]^T\}_{i=1}^N$, one can linearly fit this polynomial to the data. Indeed, if n is known, we can obtain the coefficients of $p_n(\boldsymbol{x})$ from solving the equation:

$$\boldsymbol{V}_n \boldsymbol{c}_n = \begin{bmatrix} x_1^n & x_1^{n-1}y_1 & \cdots & x_1 y_1^{n-1} & y_1^n \\ x_2^n & x_2^{n-1}y_2 & \cdots & x_2 y_2^{n-1} & y_2^n \\ \vdots & \vdots & & \vdots & \vdots \\ x_N^n & x_N^{n-1}y_N & \cdots & x_N y_N^{n-1} & y_N^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = 0. \tag{3.12}$$

Similar to the case of points in a line, the above linear system has a unique solution if and only if rank$(\boldsymbol{V}_n) = n$, hence the number of lines is given by

$$n \doteq \min\{j : \text{rank}(\boldsymbol{V}_j) = j\}. \tag{3.13}$$

Given the vector of coefficients $\boldsymbol{c}_n$, we are now interested in estimating the equations of each line from the associated polynomial $p_n(\boldsymbol{x})$. We know each line is determined by its normal vector $\boldsymbol{b}_j = [b_{1j}, b_{2j}]^T$, $j = 1, 2, \ldots, n$. For the sake of simplicity, let us consider the case $n = 2$. A simple calculation shows that the derivative of $p_2(\boldsymbol{x})$ is given by

$$\nabla p_2(\boldsymbol{x}) = (b_{21}x + b_{22}y)\boldsymbol{b}_1 + (b_{11}x + b_{12}y)\boldsymbol{b}_2. \tag{3.14}$$

Therefore, if the point $\boldsymbol{x}$ belongs to L_1, then $(b_{11}x + b_{12}y) = 0$ and hence $\nabla p_2(\boldsymbol{x}) \sim \boldsymbol{b}_1$. Similarly, if $\boldsymbol{x}$ belongs to L_2, then $\nabla p_2(\boldsymbol{x}) \sim \boldsymbol{b}_2$. This means that given any point $\boldsymbol{x}$, without knowing which line contains the point, we can obtain the equation of the line passing through the point by simply evaluating the derivative of $p_2(\boldsymbol{x})$ at $\boldsymbol{x}$. This fact should come at no surprise and is valid for any number of lines n. Therefore, if we are given one point in each line[7] $\{\boldsymbol{y}_j \in L_j\}$, we can determine the normal vectors as $\boldsymbol{b}_j \sim \nabla p_n(\boldsymbol{y}_j)$. We summarize the overall solution for clustering points to multiple lines as Algorithm 3.2.

Algorithm 3.2 (Algebraic Line Segmentation Algorithm).

Let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ be a collection of $N \geq n$ points in $\mathbb{R}^2$ clustering around an unknown number n of lines whose normal vectors are $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_N\}$. The number of lines, the normal vectors, and the segmentation of the data can be determined as follows:

1. **Number of Lines**. Let $\boldsymbol{V}_j$ be defined as in (3.12). Determine the number of groups as
$$n \doteq \min\{j : \text{rank}(\boldsymbol{V}_j) = j\}.$$
2. **Normal Vectors.** Solve for $\boldsymbol{c}_n$ from $\boldsymbol{V}_n\boldsymbol{c}_n = 0$ and set $p_n(x, y) = \sum_{k=0}^{n} c_k x^{n-k} y^k$. Determine the normal vectors as
$$\boldsymbol{b}_j = \frac{\nabla p_n(\boldsymbol{y}_j)}{\|\nabla p_n(\boldsymbol{y}_j)\|} \in \mathbb{R}^2, \qquad j = 1, 2, \ldots, n,$$
where $\boldsymbol{y}_j$ is a point in the jth line.
3. **Segmentation.** Assign point $\boldsymbol{x}_i$ to line $j = \arg\min_{\ell=1,\ldots,n}(\boldsymbol{b}_\ell^T \boldsymbol{x}_i)^2$.

The reader may have realized that the problem of clustering points on a line is very much related to the problem of segmenting lines in the plane. In point clus-

[7]We will discuss how to automatically obtain one point per subspace from the data in the next subsection when we generalize this problem to clustering points on hyperplanes.

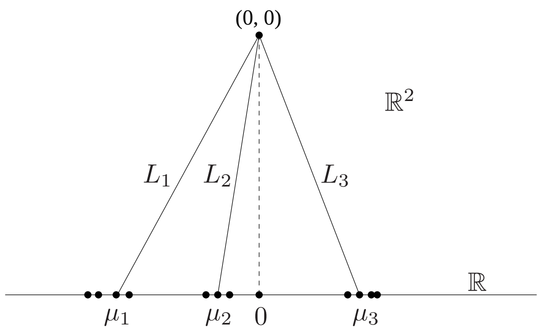


Figure 3.4. Using homogeneous coordinates to convert the point clustering problem into the line segmentation problem.

tering, for each data point x there exists a cluster center μ_j such that $x - \mu_j = 0$. By working in homogeneous coordinates, one can convert it into a line clustering problem: for each data point $\boldsymbol{x} = [x, 1]^T$ there is a line $\boldsymbol{b}_j = [1, -\mu_j]^T$ passing through the point. Figure 3.4 shows an example of how three cluster centers are converted into three lines via homogeneous coordinates. Indeed, notice that if we let $y = 1$ in the matrix $\boldsymbol{V}_n$ in (3.12), we obtain exactly the matrix $\boldsymbol{V}_n$ in (3.7). Therefore, the vector of coefficients $\boldsymbol{c}_n$ is the same for both algorithms and the two polynomials are related as $p_n(x, y) = y^n p_n(x/y)$. Therefore, the point clustering problem can be solved either by polynomial factorization (Algorithm 3.1) or by polynomial differentiation (Algorithm 3.2).

3.2.3 Segmenting Hyperplanes

In this section, we consider another particular case of Problem 3.1 in which all the subspaces are hyperplanes of equal dimension $d_1 = \cdots = d_n = d = D - 1$. This case shows up in a wide variety of segmentation problems in computer vision, including vanishing point detection and motion segmentation. We will discuss these applications in greater detail in later chapters.

We start by noticing that every $(D-1)$-dimensional subspace $S_j \subset \mathbb{R}^D$ can be defined in terms of a nonzero *normal* vector $\boldsymbol{b}_j \in \mathbb{R}^D$ as follows:[8]

$$S_j \doteq \big\{\boldsymbol{x} \in \mathbb{R}^D : \boldsymbol{b}_j^T \boldsymbol{x} \doteq b_{j1}x_1 + b_{j2}x_2 + \cdots + b_{jD}x_D = 0\big\}. \tag{3.15}$$

Therefore, a point $\boldsymbol{x} \in \mathbb{R}^D$ lying in one of the hyperplanes S_j must satisfy the formula:

$$(\boldsymbol{b}_1^T \boldsymbol{x} = 0) \;\vee\; (\boldsymbol{b}_2^T \boldsymbol{x} = 0) \;\vee \cdots \vee\; (\boldsymbol{b}_n^T \boldsymbol{x} = 0), \tag{3.16}$$

[8]Since the subspaces S_j are all different from each other, we assume that the normal vectors $\{\boldsymbol{b}_j\}_{j=1}^n$ are pairwise linearly independent.

which is equivalent to the following homogeneous polynomial of degree n in $\boldsymbol{x}$ with real coefficients:

$$p_n(\boldsymbol{x}) = \prod_{j=1}^{n} (\boldsymbol{b}_j^T \boldsymbol{x}) = \sum c_{n_1,n_2,\ldots,n_D} x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D} = \nu_n(\boldsymbol{x})^T \boldsymbol{c}_n = 0, \quad (3.17)$$

where $c_{n_1,\ldots,n_D} \in \mathbb{R}$ represents the coefficient of monomial $x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$, $\boldsymbol{c}_n$ is the vector of all coefficients, and $\nu_n(\boldsymbol{x})$ is the stack of all possible monomials. The number of linearly independent monomials is $M_n \doteq \binom{D+n-1}{n}$, hence $\boldsymbol{c}_n$ and $\nu_n(\boldsymbol{x})$ are vectors in $\mathbb{R}^{M_n}$.

After applying (3.17) to the given collection of N sample points $\{\boldsymbol{x}_i\}_{i=1}^N$, we obtain the following system of linear equations on the vector of coefficients $\boldsymbol{c}_n$

$$\boldsymbol{V}_n \, \boldsymbol{c}_n \doteq \begin{bmatrix} \nu_n(\boldsymbol{x}_1)^T \\ \nu_n(\boldsymbol{x}_2)^T \\ \vdots \\ \nu_n(\boldsymbol{x}_N)^T \end{bmatrix} \boldsymbol{c}_n = 0 \quad \in \mathbb{R}^N. \quad (3.18)$$

We now study under what conditions we can solve for n and $\boldsymbol{c}_n$ from equation (3.18). To this end, notice that if the number of hyperplanes n was known, we could immediately recover $\boldsymbol{c}_n$ as the eigenvector of $\boldsymbol{V}_n^T \boldsymbol{V}_n$ associated with its smallest eigenvalue. However, since the above linear system (3.18) depends explicitly on the number of hyperplanes n, we cannot estimate $\boldsymbol{c}_n$ directly without knowing n in advance. Recall from Example C.14, the vanishing ideal I of a hyperplane arrangement is always principal, i.e., generated by a single polynomial of degree n. The number of hyperplanes n then coincides with the degree of the first non-trivial homogeneous component I_n of the vanishing ideal. This leads to the following theorem.

Theorem 3.7 (Number of Hyperplanes)**.** *Assume that a collection of $N \geq M_n - 1$ sample points $\{\boldsymbol{x}_i\}_{i=1}^N$ on n different $(D-1)$-dimensional subspaces of $\mathbb{R}^D$ is given. Let $\boldsymbol{V}_j \in \mathbb{R}^{N \times M_j}$ be the matrix defined in (3.18), but computed with polynomials of degree j. If the sample points are in general position and at least $D-1$ points correspond to each hyperplane, then:*

$$\text{rank}(\boldsymbol{V}_j) \begin{cases} = M_j & j < n, \\ = M_j - 1 & j = n, \\ < M_j - 1 & j > n. \end{cases} \quad (3.19)$$

Therefore, the number n of hyperplanes is given by:

$$n = \min\{j : \text{rank}(\boldsymbol{V}_j) = M_j - 1\}. \quad (3.20)$$

In the presence of noise, one cannot directly estimate n from (3.20), because the matrix $\boldsymbol{V}_j$ is always full rank. In this case, one can use the criterion (2.48) given in Chapter 2 to determine the rank.

Theorem 3.7 and the linear system in equation (3.18) allow us to determine the number of hyperplanes n and the vector of coefficients $\boldsymbol{c}_n$, respectively, from sample points $\{\boldsymbol{x}_i\}_{i=1}^N$. The rest of the problem now becomes how to recover the normal vectors $\{\boldsymbol{b}_j\}_{j=1}^n$ from $\boldsymbol{c}_n$. Imagine, for the time being, that we were given a set of n points $\{\boldsymbol{y}_j\}_{j=1}^n$, each one lying in only one of the n hyperplanes, that is $\boldsymbol{y}_j \in S_j$ for $j = 1, 2, \dots, n$. Now let us consider the derivative of $p_n(\boldsymbol{x})$ evaluated at each $\boldsymbol{y}_j$. We have:

$$\nabla p_n(\boldsymbol{x}) = \frac{\partial p_n(\boldsymbol{x})}{\partial \boldsymbol{x}} = \frac{\partial}{\partial \boldsymbol{x}} \prod_{j=1}^{n} (\boldsymbol{b}_j^T \boldsymbol{x}) = \sum_{j=1}^{n} (\boldsymbol{b}_j) \prod_{\ell \neq j} (\boldsymbol{b}_\ell^T \boldsymbol{x}). \tag{3.21}$$

Because $\prod_{\ell \neq m}(\boldsymbol{b}_\ell^T \boldsymbol{y}_j) = 0$ for $j \neq m$, one can obtain each one of the normal vectors as

$$\boldsymbol{b}_j = \frac{\nabla p_n(\boldsymbol{y}_j)}{\|\nabla p_n(\boldsymbol{y}_j)\|}, \quad j = 1, 2, \dots, n. \tag{3.22}$$

Therefore, if we know one point in each one of the hyperplanes, the hyperplane segmentation problem can be solved analytically by simply evaluating the partial derivatives of $p_n(\boldsymbol{x})$ at each one of the points with known labels.

Consider now the case in which we do not know the membership of any of the data points. We now show that one can obtain one point per hyperplane by intersecting a random line with each one of the hyperplanes. To this end, consider a random line $L \doteq \{t\boldsymbol{v} + \boldsymbol{x}_0,\ t \in \mathbb{R}\}$ with direction $\boldsymbol{v}$ and base point $\boldsymbol{x}_0$. We can obtain one point in each hyperplane by intersecting L with the union of all the hyperplanes.[9] Since at the intersection points we must have $p_n(t\boldsymbol{v} + \boldsymbol{x}_0) = 0$, the n points $\{\boldsymbol{y}_j\}_{j=1}^n$ can be obtained as

$$\boldsymbol{y}_j = t_j \boldsymbol{v} + \boldsymbol{x}_0, \quad j = 1, 2, \dots, n, \tag{3.23}$$

where $\{t_j\}_{j=1}^n$ are the roots of the univariate polynomial of degree n

$$q_n(t) = p_n(t\boldsymbol{v} + \boldsymbol{x}_0) = \prod_{j=1}^{n} \left(t\boldsymbol{b}_j^T \boldsymbol{v} + \boldsymbol{b}_j^T \boldsymbol{x}_0\right) = 0. \tag{3.24}$$

We summarize our discussion so far as Algorithm 3.3 for segmenting hyperplanes.

3.3 Subspace Segmentation Knowing the Number of Subspaces

In this section, we derive a general solution to the subspace-segmentation problem (Problem 3.1) in the case in which the number of subspaces n is *known*. However, unlike the special cases we saw in the previous section, the dimensions of the subspaces can be different. In Section 3.3.1, we illustrate the basic ideas of dealing with subspaces of different dimensions via a simple example. Through Sections

[9] Except when the chosen line is parallel to one of the hyperplanes, which corresponds to a zero-measure set of lines.

Algorithm 3.3 (Algebraic Hyperplane Segmentation Algorithm).

Let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$ be a given collection of points clustered around an unknown number n of planes $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$. The number of planes, the normal vectors, and the segmentation of the data can be determined as follows:

1. **Number of Hyperplanes**. Let $\boldsymbol{V}_j$ be defined as in (3.18). Determine the number of groups as
$$n \doteq \min\{j : \text{rank}(\boldsymbol{V}_j) = M_j - 1\}.$$
2. **Normal Vectors.** Solve for $\boldsymbol{c}_n$ from $\boldsymbol{V}_n \boldsymbol{c}_n = 0$ and set $p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$. Choose $\boldsymbol{x}_0$ and $\boldsymbol{v}$ at random and compute the n roots $t_1, t_2, \ldots, t_n \in \mathbb{R}$ of the univariate polynomial $q_n(t) = p_n(t\boldsymbol{v} + \boldsymbol{x}_0)$. Determine the normal vectors as
$$\boldsymbol{b}_j = \frac{\nabla p_n(\boldsymbol{y}_j)}{\|\nabla p_n(\boldsymbol{y}_j)\|}, \qquad j = 1, 2, \ldots, n,$$
where $\boldsymbol{y}_j = \boldsymbol{x}_0 + t_j \boldsymbol{v}$ is a point in the jth hyperplane.
3. **Segmentation.** Assign point $\boldsymbol{x}_i$ to hyperplane $j = \arg\min_{l=1,\ldots,n} (\boldsymbol{b}_l^T \boldsymbol{x}_i)^2$.

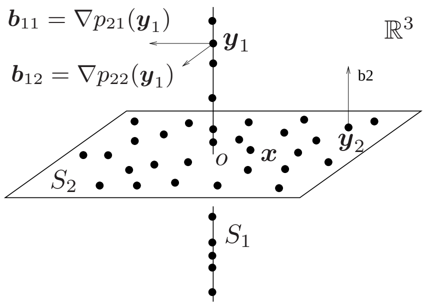

$\boldsymbol{b}_{11} = \nabla p_{21}(\boldsymbol{y}_1)$
$\boldsymbol{b}_{12} = \nabla p_{22}(\boldsymbol{y}_1)$
$\boldsymbol{y}_1$
$\mathbb{R}^3$
b2
$\boldsymbol{y}_2$
o
$\boldsymbol{x}$
S_2
S_1

Figure 3.5. Data samples drawn from a union of one plane and one line (through the origin o) in $\mathbb{R}^3$. The derivatives of the two vanishing polynomials $p_{21}(\boldsymbol{x}) = x_1x_2$ and $p_{22}(\boldsymbol{x}) = x_1x_3$ evaluated at a point $\boldsymbol{y}_1$ in the line give two normal vectors to the line. Similarly, the derivatives at a point $\boldsymbol{y}_2$ in the plane give the normal vector to the plane.

3.3.2-3.3.4, we give detailed derivation and proof for the general case. The final algorithm is summarized in Section 3.3.5.

3.3.1 An Introductory Example

To motivate and highlight the key ideas, in this section we study a simple example of clustering data points lying in subspaces of different dimensions in $\mathbb{R}^3$: a line $S_1 = \{\boldsymbol{x} : x_1 = x_2 = 0\}$ and a plane $S_2 = \{\boldsymbol{x} : x_3 = 0\}$, as shown in Figure 3.5.

We can describe the union of these two subspaces as

$$S_1 \cup S_2 = \{\boldsymbol{x} : (x_1 = x_2 = 0) \vee (x_3 = 0)\} = \{\boldsymbol{x} : (x_1x_3 = 0) \wedge (x_2x_3 = 0)\}.$$

Therefore, even though each individual subspace is described with polynomials of degree one (linear equations), the union of two subspaces is described with two polynomials of degree two, namely $p_{21}(\boldsymbol{x}) = x_1x_3$ and $p_{22}(\boldsymbol{x}) = x_2x_3$. In general, we can represent any two subspaces of $\mathbb{R}^3$ as the set of points satisfying a set of homogeneous polynomials of the form

$$c_1x_1^2 + c_2x_1x_2 + c_3x_1x_3 + c_4x_2^2 + c_5x_2x_3 + c_6x_3^2 = 0. \qquad (3.25)$$

Although these polynomials are nonlinear in each data point $[x_1, x_2, x_3]^T$, they are actually linear in the vector of coefficients $\boldsymbol{c} = [c_1, c_2, \ldots, c_6]^T$. Therefore, given enough data points, one can linearly *fit* these *polynomials* to the *data*.

Given the collection of polynomials that vanish on the data points, we are now interested in estimating a basis for each subspace. In our example, let $P_2(\boldsymbol{x}) = [p_{21}(\boldsymbol{x}),\ \ p_{22}(\boldsymbol{x})]$ and consider the derivatives of $P_2(\boldsymbol{x})$ at two representative points of the two subspaces $\boldsymbol{y}_1 = [0,0,1]^T \in S_1$ and $\boldsymbol{y}_2 = [1,1,0]^T \in S_2$:

$$\nabla P_2(\boldsymbol{x}) = \begin{bmatrix} x_3 & 0 \\ 0 & x_3 \\ x_1 & x_2 \end{bmatrix} \implies \nabla P_2(\boldsymbol{y}_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } \nabla P_2(\boldsymbol{y}_2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}. \qquad (3.26)$$

Then the columns of $\nabla P_2(\boldsymbol{y}_1)$ span the orthogonal complement to the first subspace $S_1^\perp$ and the columns of $\nabla P_2(\boldsymbol{y}_2)$ span the orthogonal complement to the second subspace $S_2^\perp$ (see Figure 3.5). Thus the dimension of the line is given by $d_1 = 3 - \text{rank}(\nabla P_2(\boldsymbol{y}_1)) = 1$ and the dimension of the plane is given by $d_2 = 3 - \text{rank}(\nabla P_2(\boldsymbol{y}_2)) = 2$. Therefore, if we are given one point in each subspace, we can obtain the *subspace bases* and their *dimensions* from the *derivatives of the polynomials* at the given points.

The final question is how to choose one representative point per subspace. With perfect data, we may choose a first point as any of the points in the data set. With noisy data, we may first define a distance from any point in $\mathbb{R}^3$ to the union of the subspaces,[10] and then choose a point in the data set that minimizes this distance. Say we pick $\boldsymbol{y}_2 \in S_2$ as such point. We can then compute the normal vector $\boldsymbol{b}_2 = [0,0,1]^T$ to S_2 from $\nabla P(\boldsymbol{y}_2)$ as above. How do we now pick a second point in S_1 but not in S_2? As it turns out, this can be done by *polynomial division*. We can divide the original polynomials by $\boldsymbol{b}_2^T\boldsymbol{x}$ to obtain new polynomials of degree $n - 1 = 1$:

$$p_{11}(\boldsymbol{x}) = \frac{p_{21}(\boldsymbol{x})}{\boldsymbol{b}_2^T\boldsymbol{x}} = x_1 \quad \text{and} \quad p_{12}(\boldsymbol{x}) = \frac{p_{22}(\boldsymbol{x})}{\boldsymbol{b}_2^T\boldsymbol{x}} = x_2.$$

[10]For example, the squared algebraic distance to $S_1 \cup S_2$ is $p_{21}(\boldsymbol{x})^2 + p_{22}(\boldsymbol{x})^2 = (x_1^2 + x_2^2)x_3^2$.

Since these new polynomials vanish on S_1 but not on S_2, we can use them to define a new distance to S_1 only,[11] and then find a point $\boldsymbol{y}_1$ in S_1 but not in S_2 as the point in the data set that minimizes this distance.

The next sections shows how this simple example can be systematically generalized to multiple subspaces of unknown and possibly different dimensions by *polynomial fitting* (Section 3.3.2), *differentiation* (Section 3.3.3), and *division* (Section 3.3.4).

3.3.2 *Fitting Polynomials to Subspaces*

Now consider a subspace arrangement $\mathcal{A} = \{S_1, S_2, \ldots, S_n\}$ with $\dim(S_j) = d_j, j = 1, 2, \ldots, n$. Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ be a sufficiently large number of sample points in general position drawn from $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$. As we may know from Appendix C, the vanishing ideal $I(Z_{\mathcal{A}})$, i.e., the set of all polynomials that vanish on $Z_{\mathcal{A}}$, is much more complicated than the special cases we studied earlier in this chapter.

Nevertheless, since we assume to know the number of subspaces n, we only have to consider the set of polynomials of degree n that vanish on $Z_{\mathcal{A}}$, i.e., the homogeneous component I_n of $I(Z_{\mathcal{A}})$. As we know from Appendix C, these polynomials uniquely determine $Z_{\mathcal{A}}$. Furthermore, as the result of Corollary C.22, we know that if the subspace arrangement is transversal, I_n is generated by the products of n linear forms that vanish on the n subspaces, respectively. More precisely, suppose the subspace S_j is of dimension d_j and let $k_j = D - d_j$. Let

$$B_j \doteq [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_{k_j}] \quad \in \mathbb{R}^{D \times (k_j)}$$

be a set of base vectors for the orthogonal complement $S_j^{\perp}$ of S_j. The vanishing ideal $I(S_j)$ of S_j is generated by the set of linear forms

$$\{l(\boldsymbol{x}) \doteq \boldsymbol{b}^T \boldsymbol{x},\ \boldsymbol{b} \in B_j\}.$$

Then any polynomial $p_n(\boldsymbol{x}) \in I_n$ can be written as a summation of products of the linear forms

$$p_n(\boldsymbol{x}) = \sum l_1(\boldsymbol{x}) l_2(\boldsymbol{x}) \cdots l_n(\boldsymbol{x}),$$

where $l_j \in I(S_j)$.

Using the Veronese map, each polynomial in I_n can also be written as:

$$p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x}) = \sum c_{n_1, n_2, \ldots, n_D} x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D} = 0, \tag{3.27}$$

where $c_{n_1, n_2, \ldots, n_D} \in \mathbb{R}$ represents the coefficient of the monomial $\boldsymbol{x}^{\boldsymbol{n}} = x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$. Although the polynomial equation is nonlinear in each data point $\boldsymbol{x}$, it is *linear* in the vector of coefficients $\boldsymbol{c}_n$. Indeed, since each polynomial $p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$ must be satisfied by every data point, we have $\boldsymbol{c}_n^T \nu_n(\boldsymbol{x}_i) = 0$

[11] For example, the squared algebraic distance to S_1 is $p_{11}(\boldsymbol{x})^2 + p_{12}(\boldsymbol{x})^2 = x_1^2 + x_2^2$.

for all $i = 1, 2, \ldots, N$. Therefore, the vector of coefficients $\boldsymbol{c}_n$ must satisfy the system of linear equations

$$\boldsymbol{V}_n(D)\ \boldsymbol{c}_n \doteq \begin{bmatrix} \nu_n(\boldsymbol{x}_1)^T \\ \nu_n(\boldsymbol{x}_2)^T \\ \vdots \\ \nu_n(\boldsymbol{x}_N)^T \end{bmatrix} \boldsymbol{c}_n = 0 \ \in \mathbb{R}^N, \tag{3.28}$$

where $\boldsymbol{V}_n(D) \in \mathbb{R}^{N \times M_n(D)}$ is called the *embedded data matrix*.

Clearly, the coefficient vector of every polynomial in I_n is in the null space of the data matrix $\boldsymbol{V}_n(D)$. For every polynomial obtained from the null space of $\boldsymbol{V}_n(D)$ to be in I_n, we need to have

$$\dim(\text{Null}(\boldsymbol{V}_n(D))) = \dim(I_n) \doteq h_I(n),$$

where $h_I(n)$ is the Hilbert function of the ideal $I(Z_{\mathcal{A}})$ (see Appendix C). Or equivalently, the rank of the data matrix $\boldsymbol{V}_n(D)$ needs to satisfy

$$\text{rank}(\boldsymbol{V}_n(D)) = M_n(D) - h_I(n) \tag{3.29}$$

in order that I_n can be exactly recovered from the null space of $\boldsymbol{V}_n(D)$. As a result of the Algebraic Sampling Theory in Appendix B, the above rank condition is typically satisfied with $N \geq (M_n(D) - 1)$ data points in general position.[12] A basis of I_n,

$$I_n = \text{span}\{p_{n\ell}(\boldsymbol{x}),\ \ell = 1, 2, \ldots, h_I(n)\}, \tag{3.30}$$

can be computed from the set of $h_I(n)$ singular vectors of $\boldsymbol{V}_n(D)$ associated with its $h_I(n)$ zero singular values. In the presence of moderate noise, we can still estimate the coefficients of the polynomials in a least-squares sense from the singular vectors associated with the $h_I(n)$ smallest singular values.

As discussed in Sections 2.5.3 and 2.5.3, the basic modeling assumption in NLPCA and KPCA is that there exists an embedding of the data into a higher-dimensional feature space $\boldsymbol{F}$ such that the features live in a linear subspace of $\boldsymbol{F}$. However, there is no general methodology for finding the correct embedding for an arbitrary problem. Equation (3.28) shows that the commonly used polynomial embedding ν_n is the right one to use when the data lives in an arrangement of subspaces, because the embedded data points $\{\nu_n(\boldsymbol{x}_i)\}_{i=1}^N$ indeed live in a subspace of $\mathbb{R}^{M_n(D)}$. Notice that each vector $\boldsymbol{c}_n$ is simply a normal vector to the embedded subspace, as illustrated in Figure 3.6.

3.3.3 Subspaces from Polynomial Differentiation

Given a basis for the set of polynomials representing an arrangement of subspaces, we are now interested in determining a basis and the dimension of each

[12] In particular, it requires at least d_j points from each subspace S_j.

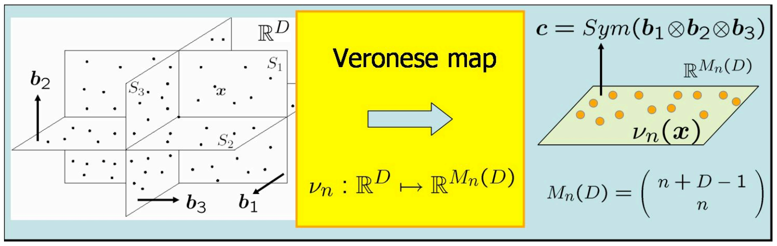

$\mathbb{R}^D$
b_2
S_1
S_3
x
S_2
b_3
b_1
Veronese map
$\nu_n : \mathbb{R}^D \mapsto \mathbb{R}^{M_n(D)}$
$c = Sym(b_1 \otimes b_2 \otimes b_3)$
$\mathbb{R}^{M_n(D)}$
$\nu_n(x)$
$M_n(D) = \binom{n+D-1}{n}$

Figure 3.6. The polynomial embedding maps a union of subspaces of $\mathbb{R}^D$ into a single subspace of $\mathbb{R}^{M_n(D)}$ whose normal vectors $\{\boldsymbol{c}_n\}$ are the coefficients of the polynomials $\{p_n\}$ defining the subspaces. The normal vectors to the embedded subspace $\{\boldsymbol{c}_n\}$ are related to the normal vectors to the original subspaces $\{\boldsymbol{b}_j\}$ via the symmetric tensor product.

subspace. In this section, we show that one can estimate the bases and the dimensions by differentiating all the polynomials $\{p_{n\ell}\}$ obtained from the null space of the embedded data matrix $\boldsymbol{V}_n(D)$.

Let $p_n(\boldsymbol{x})$ be any polynomial in I_n. Since $p_n \in I(Z_{\mathcal{A}}) \subset I(S_j)$, where $I(S_j)$ is generated by linear forms $l(\boldsymbol{x}) = \boldsymbol{b}^T\boldsymbol{x}$ with $\boldsymbol{b} \in S_j^\perp$, p_n is of the form

$$p_n = l_1 g_1 + l_2 g_2 + \cdots + l_{k_j} g_{k_j} \tag{3.31}$$

for $l_1, l_2, \ldots, l_{k_j} \in I(S_j)$ and some polynomials $g_1, g_2, \ldots, g_{k_j}$.[13] The derivative of p_n is

$$\nabla p_n = \sum_{i=1}^{k_j} (g_i \nabla l_i + l_i \nabla g_i) = \sum_{i=1}^{k_j} (g_i \boldsymbol{b}_i + l_i \nabla g_i). \tag{3.32}$$

Let $\boldsymbol{y}_j$ be a point in the subspace S_j but not in any other subspaces in the arrangement $Z_{\mathcal{A}}$. Then $l_i(y_j) = 0, i = 1, 2, \ldots, k_j$. Thus, the derivative of p_n evaluated at $\boldsymbol{y}_j$ is a superposition of the vectors $\boldsymbol{b}_i$:

$$\nabla p_n(\boldsymbol{y}_j) = \sum_{i=1}^{k_j} g_i(\boldsymbol{y}_j)\boldsymbol{b}_i \quad \in S_j^\perp. \tag{3.33}$$

This fact should come at no surprise. The zero set of each polynomial p_n is just a surface in $\mathbb{R}^D$, therefore its derivative at a regular point $\boldsymbol{y}_j \in S_j$, $\nabla p_n(\boldsymbol{y}_j)$, gives a vector orthogonal to the surface. Since an arrangement of subspaces is locally flat, i.e., in a neighborhood of $\boldsymbol{y}_j$ the surface is merely the subspace S_j, then the derivative at $\boldsymbol{y}_j$ lives in the orthogonal complement $S_j^\perp$ of S_j. By evaluating the derivatives of *all* the polynomials in I_n at the same point $\boldsymbol{y}_j$ we obtain a set of normal vectors that span the orthogonal complement of S_j. We summarize the above facts as Theorem 3.8. Figure 3.5 illustrates the theorem for the case of a plane and a line described in Section 3.3.1.

[13]In fact, from discussions in the preceding subsection, we know the polynomials g_i are products of linear forms that vanish on the remaining $n-1$ subspaces.

Theorem 3.8 (Subspace Bases and Dimensions by Polynomial Differentiation)**.** *If the data set $\boldsymbol{X}$ is such that $\dim(\mathit{Null}(\boldsymbol{V}_n(D))) = \dim(I_n) = h_I(n)$ and one generic point $\boldsymbol{y}_j$ is given for each subspace S_j, then we have*

$$S_j^{\perp} = \mathit{span}\Big\{ \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{c}_n^T \nu_n(\boldsymbol{x}) \Big|_{\boldsymbol{x}=\boldsymbol{y}_j}, \ \forall \boldsymbol{c}_n \in \mathit{Null}(\boldsymbol{V}_n(D)) \Big\}. \tag{3.34}$$

Therefore, the dimensions of the subspaces are given by

$$d_j = D - \mathit{rank}\big(\nabla P_n(\boldsymbol{y}_j)\big) \quad \mathit{for} \quad j = 1, 2, \dots, n, \tag{3.35}$$

where $P_n(\boldsymbol{x}) \doteq [p_{n1}(\boldsymbol{x}), \dots, p_{nh_I(n)}(\boldsymbol{x})] \in \mathbb{R}^{1\times h_I(n)}$ is a row of linearly independent polynomials in I_n, and $\nabla P_n(\boldsymbol{x}) \doteq \big[\nabla p_{n1}(\boldsymbol{x}), \dots, \nabla p_{nh_I(n)}(\boldsymbol{x})\big] \in \mathbb{R}^{D\times h_I(n)}$.

Proof. (Sketch only). The fact that the derivatives span the entire normal space is the consequence of the general dimension theory for algebraic varieties [Bochnak et al., 1998, Harris, 1992, Eisenbud, 1996]. For a (transversal) subspace arrangement, one can also prove the theorem by using the fact that polynomials in I_n are generated by the products of n linear forms that vanish on the n subspaces, respectively. □

Given $\boldsymbol{c}_n$, the computation of the derivative of $p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$ can be done algebraically:

$$\nabla p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nabla \nu_n(\boldsymbol{x}) = \boldsymbol{c}_n^T E_n \nu_{n-1}(\boldsymbol{x}),$$

where $E_n \in \mathbb{R}^{M_n(D)\times M_{n-1}(D)}$ is a constant matrix containing only the exponents of the Veronese map $\nu_n(\boldsymbol{x})$. Thus, the computation does *not* involve taking derivatives of the (possibly noisy) data.

3.3.4 Point Selection via Polynomial Division

Theorem 3.8 suggests that one can obtain a basis for each $S_j^{\perp}$ directly from the derivatives of the polynomials representing the union of the subspaces. However, in order to proceed we need to have one point per subspace, i.e., we need to know the vectors $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_n\}$. In this section, we show how to select these n points in the *unsupervised learning scenario* in which we do not know the label for any of the data points.

In Section 3.2.3, we showed that in the case of hyperplanes, one can obtain one point per hyperplanes by intersecting a random line L with the union of all hyperplanes.[14] This solution, however, does not generalize to subspaces of arbitrary dimensions. For instance, in the case of data lying in a line and a plane shown in Figure 3.5, a randomly chosen line L may not intersect the line. Furthermore, because polynomials in the null space of $\boldsymbol{V}_n(D)$ are no longer factorizable, their

[14]This can always be done, except when the chosen line is parallel to one of the subspaces, which corresponds to a zero-measure set of lines.

zero set is no longer a union of hyperplanes, hence the points of intersection with L may not lie in any of the subspaces.

In this section we propose an alternative algorithm for choosing one point per subspace. The idea is that we can always choose a point $\boldsymbol{y}_n$ lying in one of the subspaces, say S_n, by checking that $P_n(\boldsymbol{y}_n) = 0$. Since we are given a set of data points $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ lying in the subspaces, in principle we can choose $\boldsymbol{y}_n$ to be any of the data points. However, in the presence of noise and outliers, a random choice of $\boldsymbol{y}_n$ may be far from the true subspaces. One may be tempted to choose a point in the data set $\boldsymbol{X}$ that minimizes $\|P_n(\boldsymbol{x})\|$, as we did in our introductory example in Section 3.3.1. However, such a choice has the following problems:

1. The value $\|P_n(\boldsymbol{x})\|$ is merely an *algebraic* error, i.e., it does not really represent the *geometric* distance from $\boldsymbol{x}$ to its closest subspace. In principle, finding the geometric distance from $\boldsymbol{x}$ to its closest subspace is a hard problem, because we do not know the normal bases $\{B_1, B_2, \ldots, B_n\}$.

2. Points $\boldsymbol{x}$ lying close to the intersection of two or more subspaces are more likely to be chosen, because two or more factors in $p_n(\boldsymbol{x}) = (\boldsymbol{b}_1^T\boldsymbol{x})(\boldsymbol{b}_2^T\boldsymbol{x})\cdots(\boldsymbol{b}_n^T\boldsymbol{x})$ are approximately zero, which yields a smaller value for $|p_n(\boldsymbol{x})|$. In fact, we can see from (3.33) that for an arbitrary $\boldsymbol{x}$ in the intersection, the vector $\nabla p_n(\boldsymbol{x})$ needs to be a common normal vector to the two or more subspaces. If the subspaces have no common normal vector, then $\|\nabla p_n(\boldsymbol{x})\| = 0$. Thus, one should avoid choosing points close to the intersection, because they typically give very noisy estimates of the normal vectors.

We could avoid these two problems if we could compute the distance from each point to the subspace passing through it. However, we cannot compute such a distance yet because we do not know the subspace bases. The following lemma shows that we can compute a first order approximation to such a distance from P_n and its derivatives.

Lemma 3.9. *Let $\tilde{\boldsymbol{x}}$ be the projection of $\boldsymbol{x} \in \mathbb{R}^D$ onto its closest subspace. The Euclidean distance from $\boldsymbol{x}$ to $\tilde{\boldsymbol{x}}$ is given by*

$$\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| = n\sqrt{P_n(\boldsymbol{x})\big(\nabla P_n(\boldsymbol{x})^T \nabla P_n(\boldsymbol{x})\big)^{\dagger} P_n(\boldsymbol{x})^T} + O\big(\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2\big),$$

where $P_n(\boldsymbol{x}) = [p_{n1}(\boldsymbol{x}), \ldots, p_{nh_I(n)}(\boldsymbol{x})] \in \mathbb{R}^{1\times h_I(n)}$ is a row vector with all the polynomials, $\nabla P_n(\boldsymbol{x}) = \big[\nabla p_{n1}(\boldsymbol{x}), \ldots, \nabla p_{nh_I(n)}(\boldsymbol{x})\big] \in \mathbb{R}^{D\times h_I(n)}$, and $A^\dagger$ is the Moore-Penrose inverse of A.

Proof. The projection $\tilde{\boldsymbol{x}}$ of a point $\boldsymbol{x}$ onto the zero set of the polynomials $\{p_{n\ell}\}_{\ell=1}^{h_I(n)}$ can be obtained as the solution to the following constrained optimization problem

$$\min \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2, \quad \text{s.t.} \quad p_{n\ell}(\tilde{\boldsymbol{x}}) = 0, \quad \ell = 1, 2, \ldots, h_I(n). \tag{3.36}$$

By using Lagrange multipliers $\lambda \in \mathbb{R}^{h_I(n)}$, we can convert this problem into the unconstrained optimization problem

$$\min_{\tilde{\boldsymbol{x}},\lambda} \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 + P_n(\tilde{\boldsymbol{x}})\lambda. \tag{3.37}$$

From the first order conditions with respect to $\tilde{\boldsymbol{x}}$ we have

$$2(\tilde{\boldsymbol{x}} - \boldsymbol{x}) + \nabla P_n(\tilde{\boldsymbol{x}})\lambda = 0. \tag{3.38}$$

After multiplying on the left by $(\nabla P_n(\tilde{\boldsymbol{x}}))^T$ and $(\tilde{\boldsymbol{x}} - \boldsymbol{x})^T$, respectively, we obtain

$$\lambda = 2\big(\nabla P_n(\tilde{\boldsymbol{x}})^T \nabla P_n(\tilde{\boldsymbol{x}})\big)^\dagger \nabla P_n(\tilde{\boldsymbol{x}})^T \boldsymbol{x}, \quad \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 = \frac{1}{2}\boldsymbol{x}^T \nabla P_n(\tilde{\boldsymbol{x}})\lambda, \tag{3.39}$$

where we have used the fact that $(\nabla P_n(\tilde{\boldsymbol{x}}))^T \tilde{\boldsymbol{x}} = 0$. After substituting the first equation into the second, we obtain that the squared distance from $\boldsymbol{x}$ to its closest subspace can be expressed as

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 = \boldsymbol{x}^T \nabla P_n(\tilde{\boldsymbol{x}})\big(\nabla P_n(\tilde{\boldsymbol{x}})^T \nabla P_n(\tilde{\boldsymbol{x}})\big)^\dagger \nabla P_n(\tilde{\boldsymbol{x}})^T \boldsymbol{x}. \tag{3.40}$$

After expanding in Taylor series about $\tilde{\boldsymbol{x}} = \boldsymbol{x}$, and noticing that $\nabla P_n(\boldsymbol{x})^T \boldsymbol{x} = nP_n(\boldsymbol{x})^T$ we obtain

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|^2 \approx n^2 P_n(\boldsymbol{x})\big(\nabla P_n(\boldsymbol{x})^T \nabla P_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T, \tag{3.41}$$

which completes the proof. □

Thanks to Lemma 3.9, we can immediately choose a candidate $\boldsymbol{y}_n$ lying in (close to) one of the subspaces and not in the intersection as

$$\boldsymbol{y}_n = \operatorname*{arg\,min}_{\boldsymbol{x} \in \boldsymbol{X}: \nabla P_n(\boldsymbol{x}) \neq 0} P_n(\boldsymbol{x})\big(\nabla P_n(\boldsymbol{x})^T \nabla P_n(\boldsymbol{x})\big)^\dagger P_n(\boldsymbol{x})^T. \tag{3.42}$$

and compute a basis $B_n \in \mathbb{R}^{D \times (D-d_n)}$ for $S_n^\perp$ by applying PCA to $\nabla P_n(\boldsymbol{y}_n)$.

In order to find a point $\boldsymbol{y}_{n-1}$ lying in (close to) one of the remaining $(n-1)$ subspaces but not in (far from) S_n, we could in principle choose $\boldsymbol{y}_{n-1}$ as in (3.42) after removing the points in S_n from the data set $\boldsymbol{X}$. With noisy data, however, this depends on a threshold and is not very robust. Alternatively, we can find a new set of polynomials $\{p_{(n-1)\ell}(\boldsymbol{x})\}$ defining the algebraic set $\cup_{j=1}^{n-1} S_j$. In the case of hyperplanes, there is only one such polynomial, namely

$$p_{n-1}(\boldsymbol{x}) \doteq (\boldsymbol{b}_1 \boldsymbol{x})(\boldsymbol{b}_2 \boldsymbol{x}) \cdots (\boldsymbol{b}_{n-1}^T \boldsymbol{x}) = \frac{p_n(\boldsymbol{x})}{\boldsymbol{b}_n^T \boldsymbol{x}} = \boldsymbol{c}_{n-1}^T \nu_{n-1}(\boldsymbol{x}).$$

Therefore, we can obtain $p_{n-1}(\boldsymbol{x})$ by *polynomial division*. Notice that dividing $p_n(\boldsymbol{x})$ by $\boldsymbol{b}_n^T \boldsymbol{x}$ is a *linear problem* of the form

$$R_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = \boldsymbol{c}_n, \tag{3.43}$$

where $R_n(\boldsymbol{b}_n) \in \mathbb{R}^{M_n(D) \times M_{n-1}(D)}$. This is because solving for the coefficients of $p_{n-1}(\boldsymbol{x})$ is equivalent to solving the equations $(\boldsymbol{b}_n^T \boldsymbol{x})(\boldsymbol{c}_{n-1}^T \nu_n(\boldsymbol{x})) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$

for all $\boldsymbol{x} \in \mathbb{R}^D$. These equations are obtained by equating the coefficients, and they are linear in $\boldsymbol{c}_{n-1}$, because $\boldsymbol{b}_n$ and $\boldsymbol{c}_n$ are already known.

Example 3.10 If $n = 2$ and $\boldsymbol{b}_2 = [b_1, b_2, b_3]^T$, then the matrix $R_2(\boldsymbol{b}_2)$ is given by

$$R_2(\boldsymbol{b}_2) = \begin{bmatrix} b_1 & b_2 & b_3 & 0 & 0 & 0 \\ 0 & b_1 & 0 & b_2 & b_3 & 0 \\ 0 & 0 & b_1 & 0 & b_2 & b_3 \end{bmatrix}^T \in \mathbb{R}^{6\times 3}.$$

■

In the case of subspaces of arbitrary dimensions we cannot directly divide the entries of the polynomial vector $P_n(\boldsymbol{x})$ by $\boldsymbol{b}_n^T\boldsymbol{x}$ for any column $\boldsymbol{b}_n$ of B_n, because the polynomials $\{p_{n\ell}(\boldsymbol{x})\}$ may not be factorizable. Furthermore, they do not necessarily have the common factor $\boldsymbol{b}_n^T\boldsymbol{x}$. The following theorem resolves this difficulty by showing how to compute the polynomials associated with the remaining subspaces $\cup_{j=1}^{n-1} S_j$.

Theorem 3.11 (Choosing one Point per Subspace by Polynomial Division)**.** *If the data set $\boldsymbol{X}$ is such that $\dim(null(\boldsymbol{V}_n(D))) = \dim(I_n)$, then the set of homogeneous polynomials of degree $(n-1)$ associated with the algebraic set $\cup_{j=1}^{n-1} S_j$ is given by $\{\boldsymbol{c}_{n-1}^T v_{n-1}(\boldsymbol{x})\}$ where the vectors of coefficients $\boldsymbol{c}_{n-1} \in \mathbb{R}^{M_{n-1}(D)}$ must satisfy*

$$\boldsymbol{V}_n(D)R_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = 0, \quad \forall\, \boldsymbol{b}_n \in S_n^\perp. \tag{3.44}$$

Proof. We first show the necessity. That is, any polynomial of degree $n-1$, $\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x})$, that vanishes on $\cup_{j=1}^{n-1} S_j$ satisfies the above equation. Since a point $\boldsymbol{x}$ in the original algebraic set $\cup_{j=1}^{n} S_j$ belongs to either $\cup_{j=1}^{n-1} S_j$ or S_n, we have $\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x}) = 0$ or $\boldsymbol{b}_n^T\boldsymbol{x} = 0$ for all $\boldsymbol{b}_n \in S_n^\perp$. Hence $p_n(\boldsymbol{x}) \doteq (\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x}))(\boldsymbol{b}_n^T\boldsymbol{x}) = 0$, and $p_n(\boldsymbol{x})$ must be a linear combination of polynomials in $P_n(\boldsymbol{x})$. If we denote $p_n(\boldsymbol{x})$ as $\boldsymbol{c}_n^T\nu_n(\boldsymbol{x})$, then the vector of coefficients $\boldsymbol{c}_n$ must be in the null space of $\boldsymbol{V}_n(D)$. From $\boldsymbol{c}_n^T\nu_n(\boldsymbol{x}) = (\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x}))(\boldsymbol{b}_n^T\boldsymbol{x})$, the relationship between $\boldsymbol{c}_n$ and $\boldsymbol{c}_{n-1}$ can be written as $R_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = \boldsymbol{c}_n$. Since $\boldsymbol{V}_n(D)\boldsymbol{c}_n = 0$, $\boldsymbol{c}_{n-1}$ needs to satisfy the following linear system of equations $\boldsymbol{V}_n(D)R_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1} = 0$.

We now show the sufficiency. That is, if $\boldsymbol{c}_{n-1}$ is a solution to (3.44), then $\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x})$ is a homogeneous polynomial of degree $(n-1)$ that vanishes on $\cup_{j=1}^{n-1} S_j$. Since $\boldsymbol{c}_{n-1}$ is a solution to (3.44), then for all $\boldsymbol{b}_n \in S_n^\perp$ we have that $\boldsymbol{c}_n = R_n(\boldsymbol{b}_n)\boldsymbol{c}_{n-1}$ is in the null space of $\boldsymbol{V}_n(D)$. Now, from the construction of $R_n(\boldsymbol{b}_n)$, we also have that $\boldsymbol{c}_n^T\nu_n(\boldsymbol{x}) = (\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x}))(\boldsymbol{b}_n^T\boldsymbol{x})$. Hence, for every $\boldsymbol{x} \in \cup_{j=1}^{n-1} S_j$ but not in S_n, we have $\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x}) = 0$, because there is a $\boldsymbol{b}_n$ such that $\boldsymbol{b}_n^T\boldsymbol{x} \neq 0$. Therefore, $\boldsymbol{c}_{n-1}^T\nu_{n-1}(\boldsymbol{x})$ is a homogeneous polynomial of degree $(n-1)$ that vanishes on $\cup_{j=1}^{n-1} S_j$. □

Thanks to Theorem 3.11, we can obtain a basis $\{p_{(n-1)\ell}(\boldsymbol{x}), \ell = 1, 2, \ldots, h_I(n-1)\}$ for the polynomials vanishing on $\cup_{j=1}^{n-1} S_j$ from the intersection of the null spaces of $\boldsymbol{V}_n(D)R_n(\boldsymbol{b}_n) \in \mathbb{R}^{N\times M_{n-1}(D)}$ for all $\boldsymbol{b}_n \in S_j^\perp$. By evaluating the derivatives of the polynomials $p_{(n-1)\ell}$ we can obtain normal vectors to S_{n-1} and so on. By repeating these process, we can find a basis for each one of the remaining subspaces. The overall subspaces estimation and segmentation process involves polynomial fitting, differentiation, and division.

3.3.5 The Basic Generalized PCA Algorithm

We summarize the results of this section with the following Generalized Principal Component Analysis (GPCA) algorithm for segmenting a known number of subspaces of unknown and possibly different dimensions from sample data points $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$.

Algorithm 3.4 (GPCA: Generalized Principal Component Analysis).

Given a set of samples $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ in $\mathbb{R}^D$, fit n linear subspaces with dimensions $d_1, d_2, \ldots, d_n$:

```
1: Set $\boldsymbol{V}_n(D) \doteq [\nu_n(\boldsymbol{x}_1), \nu_n(\boldsymbol{x}_2), \ldots, \nu_n(\boldsymbol{x}_N)]^T \in \mathbb{R}^{N \times M_n(D)}$.
2: for all $j = n : 1$ do
3:     Solve $\boldsymbol{V}_j(D)\boldsymbol{c} = 0$ to obtain a basis $\{\boldsymbol{c}_{j\ell}\}_{\ell=1}^{h_I(j)}$ of null($\boldsymbol{V}_j(D)$), where
       the number of polynomials $h_I(j)$ is obtained as in Appendix B.
4:     Set $P_j(\boldsymbol{x}) = [p_{j1}(\boldsymbol{x}), p_{j2}(\boldsymbol{x}), \ldots, p_{jh_I(j)}(\boldsymbol{x})] \in \mathbb{R}^{1 \times h_I(j)}$, where
       $p_{j\ell}(\boldsymbol{x}) = \boldsymbol{c}_{j\ell}^T \nu_j(\boldsymbol{x})$ for $\ell = 1, 2, \ldots, h_I(j)$.
5:     Compute
         $\boldsymbol{y}_j = \arg\min_{\boldsymbol{x} \in \boldsymbol{X} : \nabla P_j(\boldsymbol{x}) \neq 0} P_j(\boldsymbol{x}) \big(\nabla P_j(\boldsymbol{x})^T \nabla P_j(\boldsymbol{x})\big)^{\dagger} P_j(\boldsymbol{x})^T,$
         $B_j \doteq [\boldsymbol{b}_{j1}, \boldsymbol{b}_{j2}, \ldots, \boldsymbol{b}_{j(D-d_j)}] = \text{PCA}\big(\nabla P_j(\boldsymbol{y}_j)\big),$
         $\boldsymbol{V}_{j-1}(D) = \boldsymbol{V}_j(D) \left[R_j^T(\boldsymbol{b}_{j1}), R_j^T(\boldsymbol{b}_{j2}), \ldots, R_j^T(\boldsymbol{b}_{j(D-d_j)})\right]^T.$
6: end for
7: for all $i = 1 : N$ do
8:     Assign point $\boldsymbol{x}_i$ to subspace $S_j$ if $j = \arg\min_{\ell=1,2,\ldots,n} \|B_\ell^T \boldsymbol{x}_i\|^2$.
9: end for
```

Avoiding Polynomial Division.

In practice, we may avoid computing P_j for $j < n$ by using a heuristic distance function to choose the points $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n\}$ as follows. Since a point in $\cup_{\ell=j}^n S_\ell$ must satisfy $\|B_j^T \boldsymbol{x}\| \|B_{j+1}^T \boldsymbol{x}\| \cdots \|B_n^T \boldsymbol{x}\| = 0$, we can choose a point $\boldsymbol{y}_{j-1}$ on $\cup_{\ell=1}^{j-1} S_\ell$ as:

$$\boldsymbol{y}_{j-1} = \arg\min_{\boldsymbol{x} \in \boldsymbol{X} : \nabla P_n(\boldsymbol{x}) \neq 0} \frac{\sqrt{P_n(\boldsymbol{x})(\nabla P_n(\boldsymbol{x})^T \nabla P_n(\boldsymbol{x}))^{\dagger} P_n(\boldsymbol{x})^T} + \delta}{\|B_j^T \boldsymbol{x}\| \|B_{j+1}^T \boldsymbol{x}\| \cdots \|B_n^T \boldsymbol{x}\| + \delta}, \tag{3.45}$$

where $\delta > 0$ is a small number chosen to avoid cases in which both the numerator and the denominator are zero (e.g., with perfect data).

3.4 Subspace Segmentation not Knowing the Number of Subspaces

The solution to the subspace-segmentation problem proposed in Section 3.3.5 assumes prior knowledge of the number of subspaces n. In practice, however, the number of subspaces n may not be known beforehand, hence we cannot estimate the polynomials representing the subspaces directly, because the linear system in (3.28) depends explicitly on n.

Earlier in Section 3.2, we have presented some special cases (e.g., arrangements of hyperplanes) for which one can recover the number of subspaces from data. In this section, we show that by exploiting the algebraic structure of the vanishing ideals of subspace arrangements it is possible to simultaneously recover the number of subspaces, together with their dimensions and their bases. As usual, we first examine some subtlety with determining the number of subspaces via two simple examples in Section 3.4.1 and illustrate the key ideas. Section 3.4.2 considers the case of *perfect subspace arrangements* in which all subspaces are of equal dimension $d = d_1 = \cdots = d_n$. We derive a set of rank constraints on the data from which one can estimate the n and d. Section 3.4.3 considers the most general case of subspaces of different dimensions and shows that n and can be computed in a recursive fashion by first fitting subspaces of larger dimensions and then further segmenting these subspaces into subspaces of smaller dimensions.

3.4.1 Introductory Examples

Imagine we are given a set of points $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ lying in two lines in $\mathbb{R}^3$, say

$$S_1 = \{\boldsymbol{x} : x_2 = x_3 = 0\} \quad \text{and} \quad S_2 = \{\boldsymbol{x} : x_1 = x_3 = 0\}. \tag{3.46}$$

If we form the matrix of embedded data points $\boldsymbol{V}_n(D)$ for $n = 1$ and $n = 2$, respectively:

$$\boldsymbol{V}_1(3) = \begin{bmatrix} \vdots & & \vdots \\ x_1 & x_2 & x_3 \\ \vdots & & \vdots \end{bmatrix} \text{ and } \boldsymbol{V}_2(3) = \begin{bmatrix} \vdots & & & & & \vdots \\ x_1^2 & x_1x_2 & x_1x_3 & x_2^2 & x_2x_3 & x_3^2 \\ \vdots & & & & & \vdots \end{bmatrix},$$

we obtain $\text{rank}(\boldsymbol{V}_1(3)) = 2 < 3$ and $\text{rank}(\boldsymbol{V}_2(3)) = 2 < 6$.[15] Therefore, we cannot determine the number of subspaces as the degree n such that the matrix $\boldsymbol{V}_n(D)$ drops rank (as we did in Section 3.2.3 for the case of hyperplanes), because we would obtain $n = 1$ which is not the correct number of subspaces.

How do we determine the correct number of subspaces in this case? As discussed in Section 3.1.2, a linear projection onto a low-dimensional subspace

[15]The reader is encouraged to verify these facts numerically and do the same for the examples in the rest of this section.

preserves the number and dimensions of the subspaces. In our example, if we project the data onto the plane $P = \{\boldsymbol{x} : x_1 + x_2 + x_3 = 0\}$ and then embed the projected data we obtain

$$\boldsymbol{V}_1(2) = \begin{bmatrix} \vdots & \vdots \\ x_1 & x_2 \\ \vdots & \vdots \end{bmatrix} \quad \text{and} \quad \boldsymbol{V}_2(2) = \begin{bmatrix} \vdots & & \vdots \\ x_1^2 & x_1x_2 & x_2^2 \\ \vdots & & \vdots \end{bmatrix}.$$

In this case $\text{rank}(\boldsymbol{V}_1(2)) = 2 \not< 2$, but $\text{rank}(\boldsymbol{V}_2(2)) = 2 < 3$. Therefore, the first time the matrix $\boldsymbol{V}_n(d+1)$ drops rank is when $n = 2$ and $d = 1$. This suggests that, as we will formally show in Section 3.4.2, when the subspaces are of equal dimension one can determine d and n as the minimum values for which there are a projection onto a $d+1$-dimensional subspace such that the matrix $\boldsymbol{V}_n(d+1)$ drops rank.

Unfortunately, the situation is not so simple for subspaces of different dimensions. Imagine now that in addition to the two lines S_1 and S_2 we are also given data points on a plane $S_3 = \{\boldsymbol{x} : x_1 + x_2 = 0\}$ (so that the overall configuration is similar to that shown in Figure 1.2). In this case we have $\text{rank}(\boldsymbol{V}_1(3)) = 3 \not< 3$, $\text{rank}(\boldsymbol{V}_2(3)) = 5 < 6$, and $\text{rank}(\boldsymbol{V}_3(3)) = 6 < 10$. Therefore, if we try to determine the number of subspaces as the degree of the embedding for which the embedded data matrix drops rank we would obtain $n = 2$, which is incorrect again. The reason for this is clear: we can either fit the data with one polynomial of degree $n = 2$, which corresponds to the plane S_3 and the plane P spanned by the two lines, or we can fit the data with four polynomials of degree $n = 3$, which vanish precisely on the two lines S_1, S_2, and the plane S_3.

In cases like this, one needs to resort to a more sophisticated algebraic process to identify the correct number of subspaces. As in the previous example, we can first search for the minimum degree n and dimension d such that $\boldsymbol{V}_n(d+1)$ drops rank. In our example, we obtain $n = 2$ and $d = 2$. By applying the GPCA algorithm to this data set we will partition it into two planes P and S_3. Once the two planes have been estimated, we can reapply the same process to each plane. The plane P will be separated into two lines S_1 and S_2, as described in the previous example, while the plane S_3 will remain unchanged. This recursive process stops when every subspace obtained can no longer be separated into lower-dimensional subspaces. We will a more detailed description of this Section 3.4.3.

3.4.2 Segmenting Subspaces of Equal Dimension

In this section, we derive explicit formulae for the number of subspaces n and their dimensions $\{d_j\}$ in the case of subspaces of equal dimension $d = d_1 = d_2 = \cdots = d_n$. Notice that this is a generalized version to the two-lines example that we discussed in the previous section. In the literature, arrangements of subspaces of equal dimensions are called *pure arrangements*. This type of arrangements are important for a wide range of applications in computer vision [Costeira and Kanade, 1998, Kanatani, 2002, Vidal and Ma, 2004], pattern recognition [Belhumeur et al., 1997, Vasilescu and Terzopoulos, 2002], as well as identification of hybrid linear systems [Overschee and Moor, 1993, Ma and Vidal, 2005].

Theorem 3.12 (Subspaces of Equal Dimension)**.** *Let $\{\boldsymbol{x}_i\}_{i=1}^N$ be a given collection of $N \geq M_n(d+1) - 1$ sample points lying in n different d-dimensional subspaces of $\mathbb{R}^D$. Let $\boldsymbol{V}_j(\ell+1) \in \mathbb{R}^{N \times M_j(\ell+1)}$ be the embedded data matrix defined in (3.28), but computed with the Veronese map ν_j of degree j applied to the data projected onto a generic $(\ell+1)$-dimensional subspace of $\mathbb{R}^D$. If the sample points are in general position and at least d points are drawn from each subspace, then the dimension of the subspaces is given by:*

$$d = \min\{\ell : \exists\, j \geq 1 \text{ such that } rank(\boldsymbol{V}_j(\ell+1)) < M_j(\ell+1)\}, \tag{3.47}$$

and the number of subspaces can be obtained as:

$$n = \min\{j : rank(\boldsymbol{V}_j(d+1)) = M_j(d+1) - 1\}. \tag{3.48}$$

Proof. For simplicity, we divide the proof into the following three cases:

Case 1: d known

Imagine for a moment that d was known, and that we wanted to compute n only. Since d is known, following our analysis in Section 3.1.2, we can first project the data onto a $(d+1)$-dimensional space $P \subset \mathbb{R}^D$ so that they become n d-dimensional hyperplanes in P (see Theorem 3.5). Now compute the matrix $\boldsymbol{V}_j(d+1)$ as in (3.28) by applying the Veronese map of degree $j = 1, 2, \ldots$ to the projected data. From our analysis in Section 3.2.3, there is a unique polynomial of degree n representing the union of the projected subspaces and the coefficients of this polynomial must lie in the null space of $\boldsymbol{V}_n(d+1)$. Thus, given $N \geq M_n(d+1) - 1$ points in general position, with at least d points in each subspace, we have that $\text{rank}(\boldsymbol{V}_n(d+1)) = M_n(d+1) - 1$. Furthermore, there cannot be a polynomial of degree less than n that is satisfied by all the data,[16] hence $\text{rank}(\boldsymbol{V}_j(d+1)) = M_j(d+1)$ for $j < n$. Consequently, if d is known, we can compute n by first projecting the data onto a $(d+1)$-dimensional space and then obtain

$$n = \min\{j : \text{rank}(\boldsymbol{V}_j(d+1)) = M_j(d+1) - 1\}. \tag{3.49}$$

Case 2: n known

Consider now the opposite case in which n is known, but d is unknown. Let $\boldsymbol{V}_n(\ell+1)$ be defined as in (3.28), but computed from the data projected onto a generic $(\ell+1)$-dimensional subspace of $\mathbb{R}^D$. When $\ell < d$, we have a collection of $(\ell+1)$-dimensional subspaces in an $(\ell+1)$-dimensional space, which implies that $\boldsymbol{V}_n(\ell+1)$ must be full rank. If $\ell = d$, then from equation (3.49) we have that $\text{rank}(\boldsymbol{V}_n(\ell+1)) = M_n(\ell+1) - 1$. When $\ell > d$, then equation (3.28) has

[16]This is guaranteed by the algebraic sampling theorem in Appendix B.

more than one solution, thus $\text{rank}(\boldsymbol{V}_n(\ell+1)) < M_n(\ell+1)-1$. Therefore, if n is known, we can compute d as

$$d = \min\{\ell : \text{rank}(\boldsymbol{V}_n(\ell+1)) = M_n(\ell+1)-1\}. \tag{3.50}$$

Case 3: n and d unknown

We are left with the case in which both n and d are unknown. As before, if $\ell < d$ then $\boldsymbol{V}_j(\ell+1)$ is full rank for all j. When $\ell = d$, $\boldsymbol{V}_j(\ell+1)$ is full rank for $j < n$, drops rank by one if $j = n$ and drops rank by more than one if $j > n$. Thus one can set d to be the smallest integer ℓ for which there exist an j such that $\boldsymbol{V}_j(\ell+1)$ drops rank, that is

$$d = \min\{\ell : \exists j \geq 1 \text{ such that } \text{rank}(\boldsymbol{V}_j(\ell+1)) < M_j(\ell+1)\}.$$

Given d one can compute n as in equation (3.49).

□

Therefore, in principle, both n and d can be retrieved if sufficient data points are drawn from the subspaces. The subspace-segmentation problem can be subsequently solved by first projecting the data onto a $(d+1)$-dimensional subspace and then applying the GPCA algorithm (Algorithm 3.4) to the projected data points.

In the presence of noise, one may not be able to estimate d and n from from equations (3.47) and (3.48), respectively, because the matrix $\boldsymbol{V}_j(\ell+1)$ may be full rank for all j and ℓ. As before, we can use the criterion (2.48) of Chapter 2 to determine the rank of $\boldsymbol{V}_j(\ell+1)$. However, in practice this requires to search for up to possibly $(D-1)$ values for d and $\lceil N/(D-1)\rceil$ values for n. In our experience, the rank conditions work well when either d or n are known. There are still many open issues in the problem of finding a good search strategy and model selection criterion for n and k when both of them are unknown. Some of these issues will be discussed in more detail in Chapter **??**

3.4.3 Segmenting Subspaces of Different Dimensions

In this section, we consider the problem of segmenting an unknown number of subspaces of unknown and possibly different dimensions from sample points.

First of all, we notice that the simultaneous recovery of the number and dimensions of the subspaces may be an ill-conditioned problem if we are not clear about what we are looking for. For example, in the extreme cases, one may interpret the sample set $\boldsymbol{X}$ as N 1-dimensional subspaces, with each subspace spanned by each one of the sample points $\boldsymbol{x} \in \boldsymbol{X}$; or one may view the whole $\boldsymbol{X}$ as belonging to one D-dimensional subspace, i.e., $\mathbb{R}^D$ itself.

Although the above two trivial solutions can be easily rejected by imposing some conditions on the solutions,[17] other more difficult ambiguities may also arise

[17]To reject the N-lines solution, one can put a cap on the maximum number of groups n_{max}; and to reject $\mathbb{R}^D$ as the solution, one can simply require that the maximum dimension of every subspace is strictly less than D.

in cases such as that of Figure 1.2 in which two lines and a plane can also be interpreted as two planes. More generally, when the subspaces are of different dimensions one may not be able to determine the number of subspaces directly from the degree of the polynomials fitting the data, because the degree of the polynomial of minimum degree that fits a collection of subspaces is always less than or equal to the number of subspaces.

To resolve the difficulty in determining the number and dimension of subspaces, notice that the *algebraic set* $Z_{\mathcal{A}} = \cup_{j=1}^{n} S_j$ can be decomposed into irreducible subsets S_j's – an irreducible algebraic set is also called a *variety*. The decomposition of Z into $\{S_1, S_2, \ldots, S_n\}$ is always unique. Therefore, as long as we are able to correctly determine from the given sample points the underlying algebraic set $Z_{\mathcal{A}}$ or the associated (radical) ideal $I(Z_{\mathcal{A}})$, in principle the number of subspaces n and their dimensions $\{d_1, d_2, \ldots, d_n\}$ can always be uniquely determined in a purely algebraic fashion. In Figure 1.2, for instance, the first interpretation (2 lines and 1 plane) would be the right one and the second one (2 planes) would be incorrect, because the two lines, which span one of the planes, is not an irreducible algebraic set.

Having established that the problem of subspace segmentation is equivalent to decomposing the algebraic ideal associated with the subspaces, we are left with deriving a computable scheme to achieve the goal.

From every homogeneous component I_i of

$$I(Z_{\mathcal{A}}) = I_m \oplus I_{m+1} \oplus \cdots \oplus I_n \oplus \cdots ,$$

we may compute a subspace arrangement Z_i such that $Z_{\mathcal{A}} \subseteq Z_i$ is a subspace embedding (see Section C.2). For each $i \geq m$, we can evaluate the derivatives of polynomials in I_i on subspace S_j and denote the collection of derivatives as

$$D_{i,j} \doteq \cup_{\boldsymbol{x} \in S_j} \{\nabla f \mid_{\boldsymbol{x}}, \; \forall f \in I_i\}, \quad j = 1, 2, \ldots, n. \tag{3.51}$$

Obviously, we have the following relationship:

$$D_{i,j} \subseteq D_{i+1,j} \subseteq S_j^{\perp}, \quad \forall i \geq m. \tag{3.52}$$

Then for each I_i, we can define a new subspace arrangement as

$$Z_i \doteq D_{i,1}^{\perp} \cup D_{i,2}^{\perp} \cup \cdots \cup D_{i,n}^{\perp}. \tag{3.53}$$

Notice that it is possible that $D_{i,j} = D_{i,j'}$ for different j and j' and Z_i contains less than n subspaces. We summarize the above derivation as the following theorem.

Theorem 3.13 (A Filtration of Subspace Arrangements)**.** *Let* $I(Z_{\mathcal{A}}) = I_m \oplus I_{m+1} \oplus \cdots \oplus I_n \oplus \cdots$ *be the ideal of a subspace arrangement* $Z_{\mathcal{A}}$*. Let* Z_i *be the subspace arrangement defined by the derivatives of* $I_i, i \geq m$ *as above. Then we obtain a filtration of subspace arrangements:*

$$Z_m \supseteq Z_{m+1} \supseteq \cdots \supseteq Z_n = Z_{\mathcal{A}},$$

and each subspace of $Z_{\mathcal{A}}$ *is embedded in one of the subspaces of* Z_i*.*

The above theorem naturally leads to a recursive scheme that allows us to determine the correct number and dimensions of the subspaces in $Z_{\mathcal{A}}$. Specifically, we start with $n = 1$ and increase n until there is at least one polynomial of degree n fitting all the data, i.e., until the matrix $\boldsymbol{V}_n(D)$ drops rank for the first time. For such an n, we can use Algorithm 3.4 to separate the data into n subspaces. Then we can further separate each one of these n groups of points using the same procedure. The stopping criterion for the recursion is when all the groups cannot be further separated or the number of groups n reaches some n_{max}.[18]

3.5 Model Selection for Multiple Subspaces

However, if the data points in the sample set $\boldsymbol{X}$ are corrupted by random noise, the above recursive scheme may fail to return a meaningful solution. In fact, up till now, we have been purposely avoiding a fundamental difficulty in our problem: it is inherently *ambiguous* in fitting multiple subspaces for any given data set, especially if the number of subspaces and their dimensions are not given *a priori*. When the sample points in $\boldsymbol{X}$ are noisy or are in fact drawn from a nonlinear manifold, any multi-subspace model unlikely will fit the data perfectly except for the pathological cases: 1. All points are viewed as in one D-dimensional subspace – the ambient space; 2. Every point is viewed as in an individual one-dimensional subspace. In general, the more the number of planes we use, the higher accuracy may we achieve in fitting any given data set. Thus, a fundamental question we like to address in this section is:

> *Among a class of subspace arrangements, what is the "optimal" model that fits a given data set?*

From a practical viewpoint, we also need to know under what conditions the optimal model exists and is unique, and more importantly, how to compute it efficiently.

In Appendix C, we have seen that in general, any model selection criterion aims to strike a balance between the complexity of the resulting model and the fidelity of the model to the given data. However, its exact form often depends on the class of models of interest as well as how much information is given about the model in advance. If we were to apply any of the model-selection criteria (or their concepts) to subspace arrangements, at least two issues need to be addressed:

1. We need to know how to measure the model complexity of arrangements of subspaces (possibly of different dimensions).

2. As the choice of a subspace arrangement involves both continuous parameters (the subspace bases) and discrete parameters (the number of subspaces

[18]For example, the inequality $M_n(D) \leq N$ imposes a constraint on the maximum possible number of groups n_{max}.

and their dimensions), we need to know how to properly balance the model complexity and the modeling error for subspace arrangements.

In the rest of this section, we provide a specific model selection criterion for subspace arrangements. The most fundamental idea behind the proposed criterion is that the optimal model should lead to the most compact or sparse representation for the data set.

3.5.1 Effective Dimension of Samples of Multiple Subspaces

Definition 3.14 (Effective Dimension)**.** *Given an arrangement of n subspaces $Z_{\mathcal{A}} \doteq \cup_{j=1}^{n} S_j$ in $\mathbb{R}^D$ of dimension $d_j < D$, and N_j sample points $\boldsymbol{X}_j$ drawn from each subspace S_j, the* effective dimension *of the entire set of $N = \sum_{j=1}^{n} N_j$ sample points, $\boldsymbol{X} = \cup_{j=1}^{n} \boldsymbol{X}_j$, is defined to be:*

$$\text{ED}(\boldsymbol{X}, Z_{\mathcal{A}}) \doteq \frac{1}{N}\Big(\sum_{j=1}^{n} d_j(D-d_j) + \sum_{j=1}^{n} N_j d_j\Big). \tag{3.54}$$

We contend that $\text{ED}(\boldsymbol{X}, Z_{\mathcal{A}})$ is the "average" number of (unquantized) real numbers that one needs to assign to $\boldsymbol{X}$ per sample point in order to specify the configurations of the n subspaces and the relative locations of the sample points in the subspaces. In the first term of equation (3.54), $d_j(D-d_j)$ is the total number of real numbers (known as the Grassmannian coordinates[19]) needed to specify a d_j-dimensional subspace S_j in $\mathbb{R}^D$; in the second term of (3.54), $N_j d_j$ is the total number of real numbers needed to specify the d_j coordinates of the N_j sample points in the subspace S_j. In general, if there are more than one subspace in $Z_{\mathcal{A}}$, $\text{ED}(\boldsymbol{X}, Z_{\mathcal{A}})$ can be a rational number, instead of an integer for the conventional dimension.

Notice that we here choose real numbers as the basic "units" for measuring complexity of the model in a similar fashion in the theory of sparse representation. Indeed, if the set of basis vectors of the subspaces are given, the second term of the effective dimension is essentially the sum of ℓ^0 norm of the data points each represented as a linear combination of the bases. In general, the existence of sparse linear representation always relies on the fact that the underlying model is an arrangement of a large number of subspaces. Of course, the compactness of the model can potentially be measured by more accurate units other than real numbers. Binary numbers, or "bits," have traditionally been used in information theory for measuring the complexity of a data set. We will thoroughly examine that direction in the next chapter and will subsequently reveal the relationships among different measures such as ℓ^0 norm, ℓ^1 norm, and (binary) coding length.

[19]Notice that to represent a d-dimensional subspace in a D-dimensional space, we only need to specify a basis of d linearly independent vectors for the subspace. We may stack these vectors as rows of a $d \times D$ matrix. Any nonsingular linear transformation of these vectors span the same subspace. Thus, without loss of generality, we may assume that the matrix is of the normal form $[I_{d\times d}, G]$ where G is a $d \times (D-d)$ matrix consisting of the so-called Grassmannian coordinates.

In the above definition, the effective dimension of $\boldsymbol{X}$ depends on the subspace arrangement $Z_{\mathcal{A}}$. This is because in general, there could be many subspace structures that can fit $\boldsymbol{X}$. For example, we could interpret the whole data set as lying in one D-dimensional subspace and we would obtain an effective dimension D. On the other hand, we could interpret every point in $\boldsymbol{X}$ as lying in a one-dimensional subspace spanned by itself. Then there will be N such one-dimensional subspaces in total and the effective dimension, according to the above formula, will also be D. In general, such interpretations are obviously somewhat redundant. Therefore, we define the *effective dimension* of a given sample set $\boldsymbol{X}$ to be the minimum one among all possible models that can fit the data set:[20]

$$\mathrm{ED}(\boldsymbol{X}) \doteq \min_{Z_{\mathcal{A}}:\boldsymbol{X}\subset Z_{\mathcal{A}}} \mathrm{ED}(\boldsymbol{X}, Z_{\mathcal{A}}). \tag{3.55}$$

Example 3.15 (Effective Dimension of One Plane and Two Lines). Figure 1.2 shows data points drawn from one plane and two lines in $\mathbb{R}^3$. Obviously, the points in the two lines can also be viewed as lying in the plane that is spanned by the two lines. However, that interpretation would result in an increase of the effective dimension since one would need two coordinates to specify a point in a plane, as opposed to one in a line. For instance, suppose there are fifteen points in each line; and thirty points in the plane. When we use two planes to represent the data, the effective dimension is: $\frac{1}{60}(2\times 2\times 3 - 2\times 2^2 + 60\times 2) = 2.07$; when we use one plane and two lines, the effective dimension is reduced to: $\frac{1}{60}(2\times 2\times 3 - 2^2 - 2\times 1 + 30\times 1 + 30\times 2) = 1.6$. In general, if the number of points N is arbitrarily large (say approaching to infinity), depending on the distributions of points on the lines or the plane, the effective dimension can be anything between 1 and 2, the true dimensions of the subspaces. ■

As suggested by the above example, the arrangement of subspaces that lead to the minimum effective dimension normally corresponds to a "natural" and hence compact representation of the data in the sense that it achieves the best compression (or dimension reduction) among all possible multiple-subspace models.

3.5.2 Minimum Effective Dimension of Noisy Samples

In practice, real data are corrupted with noise, hence we do not expect that the optimal model fits the data perfectly. The conventional wisdom is to strike a good balance between the complexity of the chosen model and the data fidelity (to the model). See Appendix A.4 for a more detailed discussion about numerous model selection criteria. To measure the data fidelity, let us denote the projection of each data point $\boldsymbol{x}_i \in \boldsymbol{X}$ to the closest subspace as $\hat{\boldsymbol{x}}_i$ and let $\hat{\boldsymbol{X}} = \{\hat{\boldsymbol{x}}_i\}$. Then, the

[20]The space of subspace arrangements is topologically compact and closed, hence the minimum effective dimension is always achievable and hence well-defined.

total error residual can be measured by:

$$\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|^2 = \sum_{i=1}^{N} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2. \tag{3.56}$$

As all model-selection criteria exercise the same rationale as above, we here adopt the geometric-AIC (GAIC) criterion (2.51)[21] and it leads to the following objective for selecting the optimal multiple-subspace model:

$$Z^*_{\mathcal{A}} = \arg\min_{Z_{\mathcal{A}}:\hat{\boldsymbol{X}}\subset Z_{\mathcal{A}}} \frac{1}{N}\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|^2 + 2\sigma^2 \mathrm{ED}(\hat{\boldsymbol{X}}, Z_{\mathcal{A}}), \tag{3.57}$$

where σ^2 is the noise variance of the data. However, this optimization problem can be very difficult to solve: The variance σ^2 might not be known *a priori* and we need to search for the global minimum in the configuration space of all subspace arrangements, which is not a smooth manifold and has very complicated topological and geometric structures. The resulting computation is typically prohibitive.

To alleviate some of the difficulty, in practice, we may instead minimize the effective dimension subject to a maximum allowable error tolerance. That is, among all the multiple-subspace models that fit the data within a given error bound, we choose the one with the smallest effective dimension. To this end, we define the minimum effective dimension *subject to an error tolerance* τ as:

$$\mathrm{MED}(\boldsymbol{X}, \tau) \doteq \min_{Z_{\mathcal{A}}} \mathrm{ED}(\hat{\boldsymbol{X}}, Z_{\mathcal{A}}) \text{ s.t. } \|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_\infty \le \tau, \tag{3.58}$$

where $\hat{\boldsymbol{X}}$ is the projection of $\boldsymbol{X}$ onto the subspaces in $Z_{\mathcal{A}}$ and the error norm $\|\cdot\|_\infty$ indicates the maximum norm: $\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_\infty = \max_{1\le i\le N} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|$. Based on the above definition, the effective dimension of a data set is then a notion that depends on the error tolerance. In the extreme, if the error tolerance is arbitrarily large, the "optimal" subspace-model for any data set can simply be the (zero-dimensional) origin; if the error tolerance is zero instead, for data with random noise, each sample point needs to be treated as a one-dimensional subspace in $\mathbb{R}^D$ of its own and that brings the effective dimension up close to D.

In many applications, the notion of maximum allowable error tolerance is particularly relevant. For instance, in image representation and compression, the task is often to find a linear or hybrid linear model to fit the imagery data subject to a given peak signal to noise ratio (PSNR).[22] The resulting effective dimension directly corresponds to the number of coefficients needed to store the resulting representation. The smaller the effective dimension is, the more compact or compressed is the final representation. In Chapter 6, we will see exactly how the minimum effective dimension principle is applied to image representation. The

[21] We here adopt the GAIC criterion only to illustrate the basic ideas. In practice, depending on the problem and application, it is possible that other model selection criteria may be more appropriate.

[22] In this context, the noise is the different between the original image and the approximate image (the signal).

same principle can be applied to any situation in which one tries to fit a piecewise linear model to a data set whose structure is nonlinear or unknown.

3.5.3 *The Recursive GPCA Algorithm*

Unlike the geometric AIC (3.57), the MED objective (3.58) is relatively easy to achieve. For instance, the recursive GPCA scheme that we have discussed earlier at the end of Section 3.4.3 can be easily modified to minimize the effective dimension subject to an error tolerance: we allow the recursion to proceeds only if the effective dimension would decrease while the resulting subspaces still fit the data with the given error bound.

To summarize the above discussions, in principle we can use the following algorithm to recursively identify subspaces in an arrangement $Z_{\mathcal{A}}$ from a set of noisy samples $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$.

Algorithm 3.5 (Recursive GPCA).

Given a set of samples $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ in the ambient space $\mathbb{R}^D$, find a set of subspaces that fit $\boldsymbol{X}$ subject to an error $\tau > 0$:

```
1:  for all $k = 1 : n_{max}$ do
2:    Set $\boldsymbol{V}_k(D) \doteq [\nu_k(\boldsymbol{x}_1), \nu_k(\boldsymbol{x}_2), \ldots, \nu_k(\boldsymbol{x}_N)]^T \in \mathbb{R}^{M_k(D)\times N}$.
3:    if rank$(\boldsymbol{V}_k(D)) < M_k(D)$ then
4:      Use the GPCA Algorithm 3.4 to partition $\boldsymbol{X}$ into $k$ subsets $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$.

5:      Apply PCA and fit each $\boldsymbol{X}_j$ with a subspace $S_j$ of dimension $d_j$, subject
        to the error $\tau$. Let $Z = S_1 \cup \cdots \cup S_k$.
6:      if ED$(\boldsymbol{X}, Z) < D$ then
7:        for $j = 1 : k$ do
8:          Apply Recursive GPCA for $\boldsymbol{X}_j$ (with $S_j$ as the ambient space).
9:        end for
10:     else
11:       Break.
12:     end if
13:   else
14:     $k \leftarrow k + 1$.
15:   end if
16: end for
```

Figure 3.7 demonstrates the result of the Recursive GPCA algorithm segmenting synthetic data drawn from two lines (100 points each) and one plane (400 points) in $\mathbb{R}^3$ corrupted with 5% uniform noise (Figure 3.7 top-left). Given a reasonable error tolerance, the algorithm stops after two levels of recursion (Figure 3.7 top-right). Note that the pink line (top-right) or group 4 (bottom-left) is a "ghost" line at the intersection of the original plane and the plane spanned by the two lines.[23] Figure 3.7 bottom-right is the plot of MED of the same data set subject to different levels of error tolerance. As we see, the effective dimension decreases monotonically with the increase of error tolerance.

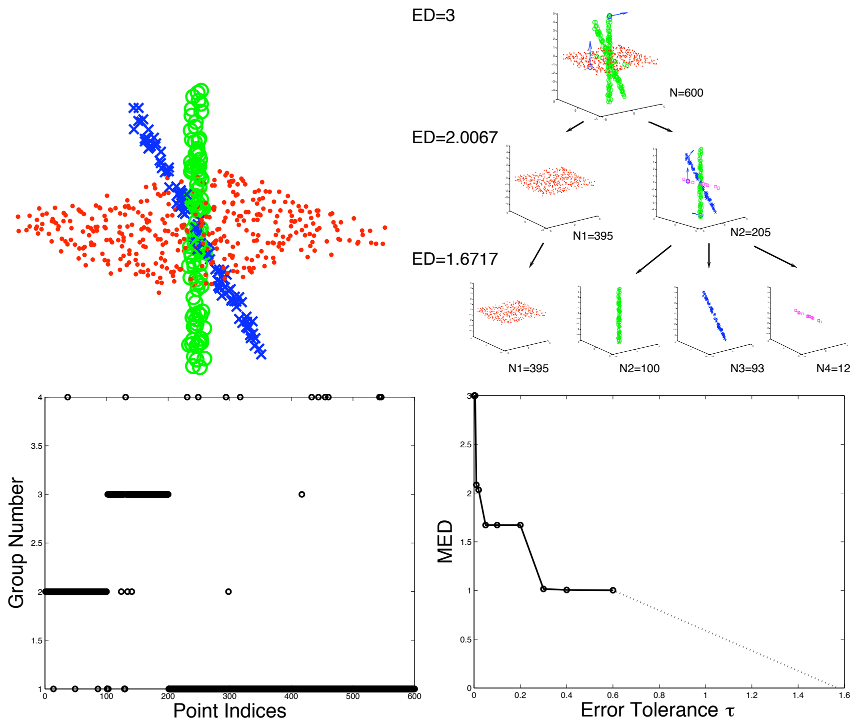


Figure 3.7. Simulation results. Top-left: sample points drawn from two lines and a plane in $\mathbb{R}^3$ with 5% uniform noise; Top-right: the process of recursive segmentation by the Recursive GPCA algorithm 3.5 with the error tolerance $\tau = 0.05$; Bottom-left: group assignment for the points; Bottom-right: plot of MED versus error tolerance.

Be aware that when the data is noisy, it sometimes can be very difficult to determine the correct dimension of the null space of the matrix $\boldsymbol{V}_n(D)$ from its singular-value spectrum. If the dimension is wrongfully determined, it may result in either under-estimating or over-estimating the number of fitting polynomials. In general, if the number of polynomials were under-estimated, the resulting subspaces would over-fit the data;[24] and if the number of polynomials were over-estimated, the resulting subspaces would under-fit the data.

[23]This is exactly what we would have expected since the recursive GPCA first segments the data into two planes. Points on the intersection of the two planes get assigned to either plane depending on the random noise. If needed, the points on the ghost line can be merged with the plane by some simple post-processing.

[24]That is, the dimensions of some of the subspaces estimated could be larger than the true ones.

Obviously, both over-fitting and under-fitting result in incorrect estimates of the subspaces. However, do they necessarily result in equally bad segmentation of the data? The answer is *no*. Between over-fitting and under-fitting, we actually would favor over-fitting. The reason is that, though over-fitting results in subspaces that are larger than the original subspaces, but it is a zero-measure event that any over-estimated subspace contains simultaneously more than one original subspace. Thus, the grouping of the data points may still be correct. For instance, consider the extreme case that we choose only one polynomial that fits the data, then the derivatives of the polynomial, evaluated at one point per subspace, lead to n hyperplanes. Nevertheless, these over-fitting hyperplanes will in general result in a correct grouping of the data points. One can verify this with the introductory example we discussed in Section 3.3.1. Either of the two polynomials $p_{21}(\boldsymbol{x}) = x_1 x_3$ and $p_{22}(\boldsymbol{x}) = x_2 x_3$ leads to two hyperplanes that segment the line and the plane correctly.

3.6 Bibliographic Notes

GPCA Algorithms and Extensions

The difficulty with initialization for the iterative clustering algorithms that we have presented in the previous chapter has motivated the recent development of algebro-geometric approaches to subspace segmentation that do *not* require initialization. [Kanatani, 2001, Boult and Brown, 1991, Costeira and Kanade, 1998] demonstrated that when the subspaces are orthogonal, of equal dimensions, and with trivial intersection, one can use the SVD of the data to define a similarity matrix from which the segmentation of the data can be obtained using spectral clustering techniques. Unfortunately, this method is sensitive to noise in the data, as pointed out in [Kanatani, 2001, Wu et al., 2001], where various improvements are proposed. When the intersection of the subspaces is nontrivial, the segmentation of the data is usually obtained in an *ad-hoc* fashion again using clustering algorithms such as K-means. A basis for each subspace is then obtained by applying PCA to each group. For the special case of two planes in $\mathbb{R}^3$, a geometric solution was developed by [Shizawa and Mase, 1991] in the context of segmentation of 2-D transparent motions. In the case of subspaces of co-dimension one, i.e., *hyperplanes*, an algebraic solution was developed by [Vidal et al., 2003], where the hyperplane clustering problem is shown to be equivalent to homogeneous polynomial factorization.

The GPCA algorithm for the most general case[25] was later developed in [Vidal et al., 2004]; and the decomposition of the polynomial(s) was based on differentiation, a numerically better-conditioned operation. The GPCA algorithm was successfully applied to solve the motion segmentation problem in computer vi-

[25]That is, an arbitrary number of subspaces of arbitrary dimensions.

sion [Vidal and Ma, 2004]. The generalization to arrangements of both linear and quadratic surfaces was first studied by [Rao et al., 2005].

Algebraic Properties of Subspace Arrangements

The importance of using subspace arrangements to model real-world high-dimensional data and the early success of the basic GPCA algorithms had motivated mathematicians to provide a more thorough characterization of subspace arrangements in terms of their vanishing ideals. A complete characterization of the Hilbert functions of the ideals for subspace arrangements was given by [Derksen, 2005], which serves as the theoretical foundation for this chapter. In Appendices B and C, we have sketched the basic algebraic concepts, results, and additional references about subspace arrangements. One may also refer to [Ma et al., 2008] for a comprehensive review on recent developments of this topic.

Effective Dimension and Sparsity

The notion of minimum Effective Dimension was first introduced in the context of recursive GPCA in [Huang et al., 2004]. We now understand that Effective Dimension is essentially a parsity measure in terms of ℓ^0-norm. Incidentally, that is the same year David Donoho published his landmark paper on sparse representation, revealing the remarkable equivalence between ℓ^0 and ℓ^1 minimization. We will have a more detailed discussion about this connection in Section 5.5, after we have examined yet another measure, coding length, for the compactness of a data set for a model.

Robustness and Outlier Rejection

There have been many work on the estimation of polynomials that best fit a given set of noisy samples. In Exercise 3.7, we will study one such approach that works well in the context of GPCA. The approach essentially follows that of [Taubin, 1991].

If there are also outliers in the given sample set, the problem becomes a more difficult robust model estimation problem. There is vast body of literature on robust statistics, see Appendix A.5 for a brief review. Sample influence is always believed to be an important index for detecting outliers. Certain first order approximations of the influence value were developed at roughly the same period as the sample influence function was proposed [Campbell, 1978,Critchley, 1985], when the computational resource was scarcer than it is today. In the literature, formulae that approximate an influence function are referred to as *theoretical influence functions*. Usually, the percentage of outliers can be determined by the influence of the candidate outliers on the model estimated [Hampel et al., 1986].

In the basic GPCA algorithm 3.4, we see that the key is to be able to robustly estimate the covariance of the samples in the lifted space, i.e. the matrix $\boldsymbol{V}_n(D)^T\boldsymbol{V}_n(D)$. Among the class of robust covariance estimators (see Appendix A.5), the multivariate trimming (MVT) method [Gnanadesikan and Kettenring, 1972] has always been one of the most popular for practitioners, probably because of its computational efficiency for high-dimensional data as well as its tolerance of large percentage of outliers. It application to GPCA is posed as Exercise 3.9.

Random sampling techniques such as the least median estimate (LME) [Hampel, 1974, Rousseeuw, 1984] and random sampling consensus (RANSAC) [Fischler and Bolles, 1981] have been widely used in many engineering areas, especially in pattern recognition and computer vision [Stewart, 1999]. They are very effective when the model is relatively simple. For instance, RANSAC is known to be very effective in making the classic PCA robust, i.e. estimating a single subspace in the presence of outliers. However, if there are multiple subspaces, RANSAC is known to work well in the case when the dimension of all the subspaces are the same [**?**]. If the subspace dimensions have different dimensions, a *Monte Carlo* scheme can be used to estimate one subspace at a time [Torr and Davidson, 2003, Schindler and Suter, 2005]. However, the performance degrades very quickly with the increase of the number of subspaces and the percentage of outliers. This has been observed in the careful experimental comparison done by [**?**]. GPCA combined with MVT was shown to perform generally better on most of the simulated data sets.

In the next chapter, we are going to see an entirely new approach to clustering data from multiple subspaces. Rather than fitting a global model to the arrangement or one model for each subspace, the new method forms subspace-like clusters by merging one sample point at a time. As we will see, one distinctive feature of such an agglomerative approach is its striking ability to handle high percentage of outliers, far more robust than the methods we have discussed or exercised so far.

3.7 Exercises

Exercise 3.1 (Clustering Points in a Plane). Describe how Algorithm 3.1 can also be applied to a set of points in the plane $\{\boldsymbol{x}_i \in \mathbb{R}^2\}_{i=1}^N$ that are distributed around a collection of cluster centers $\{\boldsymbol{\mu}_j \in \mathbb{R}^2\}_{j=1}^n$ by interpreting the data points as complex numbers: $\{z \doteq x + y\sqrt{-1} \in \mathbb{C}\}$. In particular, discuss what happens to the coefficients and roots of the fitting polynomial $p_n(z)$.

Exercise 3.2 (Connection of Algebraic Clustering with Spectral Clustering). Spectral clustering is a very popular data clustering method. In spectral clustering, one is given a set of N data points (usually in a multi-dimensional space) and an $N \times N$ pairwise similarity matrix $S = (s_{ij})$. The entries s_{ij} of S measure the likelihood of two points belonging to the same cluster: $s_{ij} \to 1$ when points i and j likely belong to the same group and $s_{ij} \to 0$ when points i and j likely belong to different groups.

1. First examine the special case in which the N data points have two clusters and the similarity matrix S is *ideal*: That is, $s_{ij} = 1$ if and only if points i and j belong to the same cluster and $s_{ij} = 0$ otherwise. What do the eigenvectors of S look like, especially the one(s) that correspond to nonzero eigenvalue(s)? Argue how the entries of the eigenvectors encode information about the membership of the points.
2. Generalize your analysis and conclusions to the case of n clusters.

3. Show how Algorithm 3.1 can be used to cluster the points based on the eigenvector of the similarity matrix. Based on Exercise 3.1, show how to cluster the points by using two eigenvectors simultaneously.

Since many popular image segmentation algorithms are based on spectral clustering (on certain similarity measure between pixels), you may use the above algorithm to improve the segmentation results.

Exercise 3.3 (Level Sets and Normal Vectors). Let $f(\boldsymbol{x}) : \mathbb{R}^D \rightarrow \mathbb{R}$ be a smooth function. For any constant $c \in \mathbb{R}$, the set $S_c \doteq \{\boldsymbol{x} \in \mathbb{R}^D | f(\boldsymbol{x}) = c\}$ is called a level set of the function f. S_c is in general a $D-1$ dimensional submanifold. Show that if $\|\nabla f(\boldsymbol{x})\|$ is nonzero at a point $\boldsymbol{x}_0 \in S_c$, then the gradient $\nabla f(\boldsymbol{x}_0) \in \mathbb{R}^D$ at $\boldsymbol{x}_0$ is orthogonal to any tangent vectors of the level set S_c.

Exercise 3.4 (Hyperplane Embedding from a Single Polynomial). Consider a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n \subset \mathbb{R}^D$. $f(\boldsymbol{x})$ is a polynomial that vanishes on $Z_{\mathcal{A}}$. Show that if we differentiate $f(\boldsymbol{x})$ at points on $Z_{\mathcal{A}}$, we always obtain an arrangement of hyperplanes that contain $Z_{\mathcal{A}}$.

Exercise 3.5 (Multiple GPCA). For each $f = 1, 2, \ldots, F$, let $\{\boldsymbol{x}_{fi} \in \mathbb{R}^D\}_{i=1}^N$ be a collection of N points lying in n hyperplanes with normal vectors $\{\boldsymbol{b}_{fj}\}_{j=1}^n$. Assume that $\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i}, \ldots, \boldsymbol{x}_{Fi}$ correspond to each other, i.e., for each $i = 1, 2, \ldots, N$ there is a $j = 1, 2, \ldots, n$ such that for all $f = 1, 2, \ldots, F$, we have $\boldsymbol{b}_{fj}^\top \boldsymbol{x}_{1i} = 0$. Propose an extension of the GPCA algorithm that computes the normal vectors in such a way that $\boldsymbol{b}_{1j}, \boldsymbol{b}_{2j}, \ldots \boldsymbol{b}_{Fj}$ correspond to each other.

Hint: If $p_{fn}(\boldsymbol{x}) = \boldsymbol{c}_f^\top \nu_n(\boldsymbol{x}) = (\boldsymbol{b}_{f1}^\top \boldsymbol{x})(\boldsymbol{b}_{f2}^\top \boldsymbol{x}) \cdots (\boldsymbol{b}_{fn}^\top \boldsymbol{x})$ *and the ith set of points* $\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i}, \ldots, \boldsymbol{x}_{Fi}$ *corresponds to the jth group of hyperplanes, then* $\boldsymbol{b}_{fj} \sim \nabla p_{fn}(\boldsymbol{x}_{fi})$.

Exercise 3.6 Implement the basic GPCA Algorithm 3.4 and test the algorithm for different subspace arrangements with different levels of noise.

Exercise 3.7 (Estimating Vanishing Polynomials). In the next two exercises, we study two ways of estimating the vanishing polynomials of a subspace arrangement from noisy samples. Since the data are noisy, a sample point $\boldsymbol{x}$ is only close to the zero set of the fitting polynomials $P(\boldsymbol{x}) = [p_1(\boldsymbol{x}), p_2(\boldsymbol{x}), \ldots, p_m(\boldsymbol{x})]^T$. Let $\hat{\boldsymbol{x}}$ be the closest point to $\boldsymbol{x}$ on the zero set of $P(\boldsymbol{x})$.

1. Show that the approximate square distance from $\boldsymbol{x}$ to $\hat{\boldsymbol{x}}$ is given by

$$\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 \approx P(\boldsymbol{x})^T \left(DP(\boldsymbol{x})DP(\boldsymbol{x})^T\right)^\dagger P(\boldsymbol{x}). \tag{3.59}$$

   This distance is known as the *Sampson distance*. From this to conclude that, given a set of sample points $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, in order to minimize the mean square fitting error, $\frac{1}{N}\sum_{i=1}^N \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2$, we can approximately minimize the average Sampson distance

$$\frac{1}{N}\sum_{i=1}^N P(\boldsymbol{x}_i)^T \left(DP(\boldsymbol{x}_i)DP(\boldsymbol{x}_i)^T\right)^\dagger P(\boldsymbol{x}_i) \tag{3.60}$$

2. However, since for any non-singular matrix $M \in \mathbb{R}^{m \times m}$, $\tilde{P}(\boldsymbol{x}) = MP(\boldsymbol{x})$ define the same zero set. Show that, in order to reduce this redundancy, we can normalize

the following matrix to an identity:

$$\frac{1}{N}\sum_{i=1}^{N} DP(\boldsymbol{x}_i)DP(\boldsymbol{x}_i)^T = I_{m\times m}. \tag{3.61}$$

Thus, the problem of minimizing the average Sampson distance now becomes a constrained optimization problem:

$$\begin{array}{rcl} P^* & = & \arg\min_P \frac{1}{N}\sum_{i=1}^{N} P(\boldsymbol{x}_i)^T\big(DP(\boldsymbol{x}_i)DP(\boldsymbol{x}_i)^T\big)^{\dagger}P(\boldsymbol{x}_i), \\ & & \text{subject to } \ \frac{1}{N}\sum_{i=1}^{N} DP(\boldsymbol{x}_i)DP(\boldsymbol{x}_i)^T = I_{m\times m}. \end{array} \tag{3.62}$$

3. Since the average of $DP(\boldsymbol{x}_i)DP(\boldsymbol{x}_i)^T$ is an identity, we can approximate each by an identity too. Then, the above problem becomes:

$$\begin{array}{rcl} P^* & = & \arg\min_P \frac{1}{N}\sum_{i=1}^{N} \|P(\boldsymbol{x}_i)\|^2, \\ & & \text{subject to } \ \frac{1}{N}\sum_{i=1}^{N} DP(\boldsymbol{x}_i)DP(\boldsymbol{x}_i)^T = I_{m\times m}. \end{array} \tag{3.63}$$

Now show that the vector of coefficients of each polynomial in P^* is a generalized eigenvector for a properly defined pair of matrices W and B. That is, they are solutions $\boldsymbol{c}_i^*$ to the following equation:

$$W\boldsymbol{c}_i^* = \lambda_i B\boldsymbol{c}_i^*, \quad i = 1, 2, \ldots, m. \tag{3.64}$$

Exercise 3.8 (Fisher Discriminant Analysis for Subspaces). We now illustrate how concepts from discriminant analysis can be adopted to estimate better fitting polynomials. We use an arrangement of hyperplanes to demonstrate the basic ideas. In this case, the fitting polynomial as the form:

$$p(\boldsymbol{x}) = \prod_{j=1}^{n}\big(\boldsymbol{b}_j^T\boldsymbol{x}\big) = \boldsymbol{c}^T\nu_n(\boldsymbol{x}) = 0 \tag{3.65}$$

with n the number of (different) hyperplanes and $\boldsymbol{b}_j$ the normal vector to the jth plane. In this case, it is very easy to find the coefficient vector $\boldsymbol{c}$ as the kernel of the data matrix $\boldsymbol{V}_n(D)$ is only one-dimensional.

1. In the presence of noise, it is likely that $p(\boldsymbol{x}) \neq 0$, but we would like to find the coefficient vector $\boldsymbol{c}$ that minimizes the following average least-square fitting error $\frac{1}{N}\sum_{i=1}^{N}|p(\boldsymbol{x}_i)|^2$. Show that the solution $\boldsymbol{c}^*$ is the eigenvector associated with the smallest eigenvalue of the matrix:

$$W \doteq \Big(\frac{1}{N}\boldsymbol{V}_n(D)^T\boldsymbol{V}_n(D)\Big). \tag{3.66}$$

In the spirit of discriminant analysis, the matrix W will be called the *within-subspace scatter matrix.*

2. Let us examine the derivative of the polynomial at each of the data samples. Let $\boldsymbol{x}_1 \in S_1$. Show that the norm of the derivative $\nabla p(\boldsymbol{x}_1)$ is

$$\|\nabla p(\boldsymbol{x}_1)\|^2 = \Big|\Big(\prod_{j=2}^{n}\boldsymbol{b}_j^T\boldsymbol{x}_1\Big)\Big|^2. \tag{3.67}$$

Thus, the average of the quantity $\|\nabla p(\boldsymbol{x}_1)\|^2$ over all $\boldsymbol{x}_1$ in S_1 gives a good measure of "distance" from S_1 to $\bigcup_{j=2}^{n} S_j$, the union of the other subspaces. For the segmentation purpose, we would like to find the coefficient vector $\boldsymbol{c}$ that maximizes the following quantity:

$$\max \frac{1}{N}\sum_{i=1}^{N} \|\nabla p(\boldsymbol{x}_i)\|^2 = \boldsymbol{c}^T \Big(\frac{1}{N}\sum_{i=1}^{N} \nabla \nu_n(\boldsymbol{x}_i)\nabla \nu_n(\boldsymbol{x}_i)^T\Big)\boldsymbol{c} \doteq \boldsymbol{c}^T B\boldsymbol{c}. \quad (3.68)$$

In the spirit of discriminant analysis, we will call B the *between-subspace scatter matrix*.

3. Therefore, we would like to seek a fitting polynomial that simultaneously minimizes the polynomial evaluated at each of the samples while maximizing the norm of the derivative at each point. This can be achieve by minimizing the ratio of these two metrics:

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \frac{\boldsymbol{c}^T W\boldsymbol{c}}{\boldsymbol{c}^T B\boldsymbol{c}}. \quad (3.69)$$

Show that the solution to this problem is given by the generalized eigenvector $\boldsymbol{c}$ that is associated with the smallest generalized eigenvalue λ of (W, B):

$$W\boldsymbol{c} = \lambda B\boldsymbol{c}. \quad (3.70)$$

In the case when B is non-singular, $\boldsymbol{c}$ is simply the eigenvector of $B^{-1}W$ associated with the smallest eigenvalue.

Exercise 3.9 (Robust Estimation of Fitting Polynomials). We know that samples from an arrangement of n subspaces, their Veronese lifting all lie on a single subspace $\operatorname{span}(\boldsymbol{V}_n(D))$. The coefficients of the fitting polynomials are simply the null space of $\boldsymbol{V}_n(D)$. If there is noise, the lifted samples approximately span a subspace and the coefficients of the fitting polynomials are eigenvectors associated with the small eigenvalues of $\boldsymbol{V}_n(D)^T\boldsymbol{V}_n(D)$. However, if there are outliers, the lifted samples together no longer span a subspace. Notice that this is the same situation that robust statistical techniques such as multivariate trimming (MVT) are designed to deal with. See Appendix A.5 for more details. In this exercise, show how to combine MVT with GPCA so that the resulting algorithm will be robust to outliers. Implement your scheme and find out the highest percentage of outliers that the algorithm can handle (for various subspace arrangements).

Chapter 4

Iterative Methods for Multiple-Subspace Segmentation

"Statistics in the hands of an engineer are like a lamppost to a drunk – they're used more for support than illumination."
– A.E. Housman

We will first review some basic concepts and existing iterative algorithms for clustering multivariate data, i.e. the K-means algorithm and the Expectation Maximization (EM) algorithm. We then give a clear formulation of the problem in which the clusters are subspaces and introduce the basic notation for representing both linear and affine subspaces. We then customize the two algorithms so as to segment a known number of subspaces with known dimensions. We point out the advantages and disadvantages of these algorithms, particularly their sensitivity to initialization.

4.1 Statistical Methods for Data Clustering

In clustering analysis, the basic assumption is that the given data points $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^D$ are grouped into a number of clusters $n \leq N$ such that the "distance" (or "dissimilarity") among points in the same group is significantly smaller than those between clusters. Thus the outcome of clustering analysis is a map:

$$c(\cdot): \ i \in \{1, 2, \ldots, N\} \quad \mapsto \quad j = c(i) \in \{1, 2, \ldots, n\} \tag{4.1}$$

that assigns each point $\boldsymbol{x}_i$ to one of the n clusters. Obviously, the outcome of the clustering very much depends on what the chosen measure of distance is.

If the notion of distance is not clearly specified, the clustering problem can be ill-defined. The following example shows some of the reasons.

Example 4.1 (No Invariant Clustering by the Euclidean Distance). If we always choose the Euclidean distance, then the clustering result *cannot* be invariant under an arbitrary linear transformation of the data points – usually representing a change of coordinates. That is, if we replace $\boldsymbol{x}_i$ with $\boldsymbol{x}_i' = A\boldsymbol{x}_i$ for some non-singular matrix $A \in \mathbb{R}^{D\times D}$, then the clustering of $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{x}_i'\}$ will in general be different. This is easy to see with a simple example. Suppose we need to cluster the $N = 4$ points in $\mathbb{R}^2$ as follows

$$\boldsymbol{x}_1 = [1, 10]^T, \quad \boldsymbol{x}_2 = [-1, 10]^T, \quad \boldsymbol{x}_3 = [1, -10]^T, \quad \boldsymbol{x}_4 = [-1, -10]^T$$

into $n = 2$ clusters. The two clusters are obviously $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ and $\{\boldsymbol{x}_3, \boldsymbol{x}_4\}$. Now consider two linear transformations A_1 and $A_2 \in \mathbb{R}^{2\times 2}$:

$$A_1 = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -10 \\ 10 & 0 \end{bmatrix}.$$

Applying the two maps to the original set of points, we obtain two new sets of points $\{\boldsymbol{x}_i' = A_1\boldsymbol{x}_i\}$ and $\{\boldsymbol{x}_i'' = A_2\boldsymbol{x}_i\}$, respectively:

$$\begin{aligned} &\boldsymbol{x}_1' = [100, 10]^T,\ \boldsymbol{x}_2' = [-100, 10]^T,\ \boldsymbol{x}_3' = [100, -10]^T,\ \boldsymbol{x}_4' = [-100, -10]^T; \\ &\boldsymbol{x}_1'' = [-100, 10]^T,\ \boldsymbol{x}_2'' = [-100, -10]^T,\ \boldsymbol{x}_3'' = [100, 10]^T,\ \boldsymbol{x}_4'' = [100, -10]^T. \end{aligned}$$

As a set $\{\boldsymbol{x}_i'\}$ is the same as $\{\boldsymbol{x}_i''\}$. However, the two clusters are $\{\boldsymbol{x}_1', \boldsymbol{x}_3'\}$ and $\{\boldsymbol{x}_2', \boldsymbol{x}_4'\}$ for the first set; and $\{\boldsymbol{x}_1'', \boldsymbol{x}_2''\}$ and $\{\boldsymbol{x}_3'', \boldsymbol{x}_4''\}$ for the latter. In fact, regardless of the choice of objective or method, it is always the case that the clustering result for one of the two new sets will be different from that for the original set. ■

From the above example, we see that in order for the clustering result to be invariant under a linear transformation, instead of always using the Euclidean distance, one should properly adjust the distance measure after each linear transformation of the data. To be more precise, let the length of a vector $\boldsymbol{x} \in \mathbb{R}^D$ be measured by

$$\|\boldsymbol{x}\|_\Sigma^2 \doteq \boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x} \tag{4.2}$$

for some positive-definite symmetric matrix $\Sigma \in \mathbb{R}^{D\times D}$. Notice that $\Sigma = I_{D\times D}$ corresponds to the Euclidean length. Then after a linear transformation, $\boldsymbol{x}' = A\boldsymbol{x}$ for some $D \times D$ matrix A, the "induced" length of $\boldsymbol{x}'$ is defined to be

$$\|\boldsymbol{x}'\|_{\Sigma'}^2 = (\boldsymbol{x}')^T (\Sigma')^{-1} \boldsymbol{x}' = (\boldsymbol{x}')^T (A\Sigma A^T)^{-1} \boldsymbol{x}' = \boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x}. \tag{4.3}$$

Thus, the induced length remains the same after the transformation.

Notice that the relationship between Σ and $\Sigma' = A\Sigma A^T$ is just like that between the covariance matrices of two random vectors related by a linear transformation A. Thus, the change of distance measure is equivalent to the assumption that the original data $\{\boldsymbol{x}_i\}$ are drawn from some probabilistic distribution. In the context of data clustering, it is natural to further assume that the distribution itself is a mixture of n (Gaussian) distributions with different means and covariances:[1]

$$p_j(\boldsymbol{x}) \ \sim \ \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j), \quad j = 1, 2, \ldots, n. \tag{4.4}$$

Thus, the clustering problem becomes a statistical model estimation problem and can be solved via statistical methods. We introduce below two such methods that are based on two different estimation (and optimization) paradigms: 1. Minimax estimate; 2. Maximum-likelihood estimate. In this section, we illustrate the basic ideas using mixtures of Gaussians; but a discussion on more general cases can be found in Appendix C.

4.1.1 K-Means

With respect to the above statistical model, a natural measure of the distance between a sample point and the mean of a cluster is the Mahanalobis distance:

$$d(\boldsymbol{x}_i, \boldsymbol{\mu}_j) \doteq \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2_{\Sigma_j}, \tag{4.5}$$

which is proportional to the (negative) log-likelihood of the sample. The map $c^*(\cdot)$ that represents an optimal clustering of the data $\{\boldsymbol{x}_i\}$ minimizes the following "within-cluster scatter":

$$\min_{c(\cdot)} \ w(c) \doteq \frac{1}{N} \sum_{j=1}^{n} \sum_{c(i)=j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2_{\Sigma_j}. \tag{4.6}$$

That is, $w(c)$ is a measure of the average distance of all the sample points to their respective cluster means. Notice that the minimum value of $w(c)$ decreases with the increase of the number n of clusters. In the extreme case $n = N$, i.e., each point is a cluster itself, we have $w(c) = 0$. Therefore, before conducting clustering analysis, it is very important to know the correct value of n. We will discuss methods to determine n in later chapters; in this chapter, we always assume the correct cluster number n is known.

In the above objective $w(c)$ (4.6), $c(\cdot)$, $\{\boldsymbol{\mu}_j\}$, and $\{\Sigma_j\}$ are all unknown. The problem is how to find the optimal $c^*(\cdot)$, $\boldsymbol{\mu}_j^*$ and Σ_j^* so that $w(c)$ is minimized. Unfortunately, there is no closed-form solution to the optimal estimates. The main difficulty is that the objective (4.6) is hybrid – it is a combination of minimization on the continuous variables $\{\boldsymbol{\mu}_j, \Sigma_j\}$ and the discrete variable $c(i)$. Conventional nonlinear optimization techniques, such as gradient descent, do not directly apply to this case. Hence special optimization schemes have to be developed.

Notice that for $w(c)$ to be minimum, it is necessary that each point $\boldsymbol{x}_i$ is assigned to the cluster whose mean is the closest to $\boldsymbol{x}_i$. That is, given $\{\boldsymbol{\mu}_j, \Sigma_j\}$, we have

$$c(i) = \arg\min_j \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2_{\Sigma_j}. \tag{4.7}$$

[1] From the viewpoint of subspaces, here we try to fit the data with *multiple* zero-dimensional affine spaces (or points) – one point (the mean) for each cluster. Later in this Chapter, we will see how to generalize the cluster means from points to arbitrary (affine) subspaces.

Also, from the samples that belong to each cluster, we can obtain unbiased estimates of the mean and covariance of the cluster:

$$\hat{\boldsymbol{\mu}}_j \doteq \frac{1}{N_j}\sum_{c(i)=j} \boldsymbol{x}_i \ \in \mathbb{R}^D, \quad \hat{\Sigma}_j \doteq \frac{1}{N_j-1}\sum_{c(i)=j}(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j)^T \ \in \mathbb{R}^{D\times D}, \tag{4.8}$$

where N_j is the number of points that are assigned to cluster j by the map $c(\cdot)$.

The above discussions have suggested the following two-step iterative process for minimizing $w(c)$.

Suppose that some initial estimates $\{\hat{\boldsymbol{\mu}}_j^{(0)}, \hat{\Sigma}_j^{(0)}\}$ of the means are available. Then we can easily minimize the objective (4.6) for $c(i)$. That is, for each cluster with the mean $\hat{\boldsymbol{\mu}}_j^{(0)}$ and covariance $\hat{\Sigma}_j^{(0)}$, we obtain the subset of points $\boldsymbol{X}_j^{(0)}$ that are closer to $\boldsymbol{\mu}_j$ than to any other means. The data set $\boldsymbol{X}$ is therefore segmented into n clusters

$$\boldsymbol{X} = \boldsymbol{X}_1^{(0)} \cup \boldsymbol{X}_2^{(0)} \cup \cdots \cup \boldsymbol{X}_n^{(0)}, \tag{4.9}$$

and we further require $\boldsymbol{X}_j^{(0)} \cap \boldsymbol{X}_{j'}^{(0)} = \emptyset$ for $j \neq j'$.[2] In this way we obtain an estimate of the map $c^{(0)}(\cdot)$.

Knowing the membership of each point $\boldsymbol{x}_i$ from the above segmentation, the objective (4.6) can be rewritten as:

$$\sum_{j=1}^{n}\left(\min_{\boldsymbol{\mu}_j, \Sigma_j} \sum_{c^{(0)}(i)=j} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2_{\Sigma_j}\right). \tag{4.10}$$

Notice that the solution to the minimization inside the bracket is an new set of estimates of the mean and covariance:

$$\hat{\boldsymbol{\mu}}_j^{(1)} = \frac{1}{N_j}\sum_{c^{(0)}(i)=j} \boldsymbol{x}_i, \quad \hat{\Sigma}_j^{(1)} = \frac{1}{N_j-1}\sum_{c^{(0)}(i)=j}\big(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j^{(1)}\big)\big(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j^{(1)}\big)^T.$$

These new means and covariances give a new value of the objective no larger than that given by the initial estimates $\big\{\hat{\boldsymbol{\mu}}_j^{(0)}, \hat{\Sigma}_j^{(0)}\big\}$.

We can further reduce the objective by re-classifying each data point $\boldsymbol{x}_i$ to its closest mean according to the new estimates $\big\{\hat{\boldsymbol{\mu}}_j^{(1)}, \hat{\Sigma}_j^{(1)}\big\}$. In this way, we obtain a new segmentation $\boldsymbol{X} = \boldsymbol{X}_1^{(1)} \cup \boldsymbol{X}_2^{(1)} \cup \cdots \cup \boldsymbol{X}_n^{(1)}$. If we keep iterating between the above two steps, the objective will keep decreasing until its value stabilizes to a (local) equilibrium and the segmentation no longer changes. This minimization process is referred to as the K-means algorithm in the statistical-learning literature. We summarize the algorithm as Algorithm 4.1.

Notice that Algorithm 4.1 can be significantly simplified if the Gaussian distributions are all *isotropic*, i.e., $\Sigma_j = \sigma_j^2 I$ for some $\sigma_j^2 \in \mathbb{R}_+$, or all covariance

[2]If a point $\boldsymbol{x} \in \boldsymbol{X}$ has the same minimal distance to more than one cluster, then we assign it arbitrarily to one of them.

Algorithm 4.1 (K-Means).

Given a set of sample points $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, the number of clusters n, initialize the means and covariances of the clusters with a set of initial values $\hat{\boldsymbol{\mu}}_j^{(0)} \in \mathbb{R}^D, \hat{\Sigma}_j^{(0)} \in \mathbb{R}^{D\times D}, j = 1, 2, \ldots, n$.
Let $m = 0$.

1. **Segmentation:** For each point $\boldsymbol{x}_i \in \boldsymbol{X}$, assign it to $\boldsymbol{X}_j^{(m)}$ if
$$j = c(i) = \operatorname*{argmin}_{\ell=1,2,\ldots,n} \|\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_\ell^{(m)}\|^2_{\Sigma_\ell^{(m)}}. \tag{4.11}$$
If the above cost function is minimized by more than one mean, assign the point arbitrarily to one of them.

2. **Estimation:** Obtain new estimates for the n cluster means and covariances:
$$\begin{aligned}\hat{\boldsymbol{\mu}}_j^{(m+1)} &= \frac{1}{N_j}\sum_{c^{(m)}(i)=j} \boldsymbol{x}_i,\\ \hat{\Sigma}_j^{(m+1)} &= \frac{1}{N_j - 1}\sum_{c^{(m)}(i)=j} \big(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j^{(m+1)}\big)\big(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j^{(m+1)}\big)^T. \end{aligned}\tag{4.12}$$

Let $m \leftarrow m+1$, and repeat Steps 1 and 2 until the segmentation does not change.

matrices are equal to the identity matrix $\Sigma_j \equiv I$. In the latter case, one essentially adopts the Euclidean distance between the sample points and the cluster means. This special case is often referred to also as the "K-means" algorithm in the literature.

4.1.2 Expectation Maximization (EM)

The K-means algorithm essentially relies on the minimax estimation paradigm in statistics (see Appendix C) and it does not need to assume how exactly the n component distributions are mixed. The Expectation Maximization (EM) algorithm [Dempster et al., 1977] to be introduced below, however, relies on the maximum-likelihood estimation paradigm (see Appendix C) and it does need an explicit model for the mixed distribution. Instead of minimizing the modeling error in a least-distance sense, the EM algorithm estimates the model parameters and the segmentation of the data in a maximum-likelihood (ML) sense. As we shall soon see, the EM algorithm, though derived from a different set of assumptions, principles, and objectives, has an overall structure that resembles very much that of the K-means algorithm.[3]

[3]This resemblance however should not be mistaken as excuses to confuse these two algorithms. The solutions given by these two algorithms will be close but different in general.

A Probabilistic Model for a Mixed Distribution

The EM algorithm is based on the assumption that the given data points $\{\boldsymbol{x}_i\}_{i=1}^N$ are independent samples from a (mixed) probabilistic distribution. In the context of clustering analysis, it is reasonable to assume that $\boldsymbol{x}_i$ are samples drawn from multiple "component" distributions and each component distribution is centered around a mean. To model from which component distribution a sample $\boldsymbol{x}$ is actually drawn, we can associate a latent discrete random variable $z \in \mathbb{R}$ to each data point $\boldsymbol{x}$, such that each discrete random variable $z_i = j$ if the point $\boldsymbol{x}_i$ is drawn from the jth component, $i = 1, 2, \ldots, N$. Then the random vector

$$(\boldsymbol{x}, z) \quad \in \mathbb{R}^D \times \mathbb{Z}_+ \tag{4.13}$$

completely describes the random event that the point $\boldsymbol{x}$ is drawn from a component distribution indicated by the value of z.

Typically, one assumes that the random variable z is subject to a multinomial (marginal) distribution, i.e.,

$$p(z = j) = \pi_j \geq 0, \quad \text{s.t. } \pi_1 + \pi_2 + \cdots + \pi_n = 1. \tag{4.14}$$

Each component distribution is then modeled as a conditional distribution $p(\boldsymbol{x}|z)$ of $\boldsymbol{x}$ given z. A popular choice for the component distribution is a multivariate Gaussian distribution: $p(\boldsymbol{x}|z = j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$, in which $\boldsymbol{\mu}_j$ is the mean and Σ_j is the covariance of the jth cluster.

The Maximum-Likelihood Estimation

In the model, the parameters $\theta \doteq \{\boldsymbol{\mu}_j, \Sigma_j, \pi_j\}_{j=1}^n$ are unknown and they need to be inferred from the samples of $\boldsymbol{x}$. The marginal distribution of $\boldsymbol{x}$ given the parameters is called the likelihood function, and is given by

$$p(\boldsymbol{x}|\theta) \quad = \quad \sum_{z=1}^{n} p(\boldsymbol{x}|z, \theta)p(z|\theta) = \sum_{j=1}^{n} \pi_j p(\boldsymbol{x}|z = j, \theta). \tag{4.15}$$

Notice that $p(\boldsymbol{x}|\theta)$ is a "mixture" of n distributions $p(\boldsymbol{x}|z = j, \theta), j = 1, 2, \ldots, n$ that is exactly of the form (1.8) introduced in Chapter 1.

Given N *i.i.d.* samples $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ from the distribution, the optimal estimates of the parameters $\hat{\theta}_{ML}$ are given by maximizing the log-likelihood function

$$l(\boldsymbol{X}; \theta) \quad \doteq \quad \sum_{i=1}^{N} \log p(\boldsymbol{x}_i|\theta). \tag{4.16}$$

In the statistical learning literature, this objective is often referred to as the *incomplete log-likelihood function* – "incomplete" compared to the complete log-likelihood function to be introduced later. However, maximizing the incomplete log-likelihood with respect to the parameters θ is typically very difficult, because this is a very high-dimensional nonlinear optimization problem. This is the motivation for the *expectation maximization* (EM) process which utilizes the latent random variable z introduced earlier to attempt to simplify the maximization process.

Derivation of the Expectation and Maximization

First notice $p(\boldsymbol{x}|\theta) = p(\boldsymbol{x}, z|\theta)/p(z|\boldsymbol{x}, \theta)$ and $\sum_j p(z = j|\boldsymbol{x}, \theta) = 1$. We can rewrite the (incomplete) log-likelihood function as

$$l(\boldsymbol{X};\theta) = \sum_{i=1}^{N}\sum_{j=1}^{n} p(z_i = j|\boldsymbol{x}_i, \theta) \log \frac{p(\boldsymbol{x}_i, z_i = j|\theta)}{p(z_i = j|\boldsymbol{x}_i, \theta)} \tag{4.17}$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{n} p(z_i = j|\boldsymbol{x}_i, \theta) \log p(\boldsymbol{x}_i, z_i = j|\theta) \tag{4.18}$$

$$- \sum_{i=1}^{N}\sum_{j=1}^{n} p(z_i = j|\boldsymbol{x}_i, \theta) \log p(z_i = j|\boldsymbol{x}_i, \theta). \tag{4.19}$$

The first term (4.18) is called the *expected complete log-likelihood function* in the statistical learning literature;[4] and the second term (4.19) is the *conditional entropy*[5] of z_i given $\boldsymbol{x}_i$ and θ. Hence, the maximum-likelihood estimation is equivalent to maximizing the expected log-likelihood and at the same time minimizing the conditional entropy of z_i.

Given each $\boldsymbol{x}_i$, we can define a new function $w_{ij}(\theta) \doteq p(z_i = j|\boldsymbol{x}_i, \theta)$. By replacing $\boldsymbol{w}(\theta) = \{w_{ij}(\theta)\}$ into the incomplete log-likelihood, we can view $l(\boldsymbol{X};\theta)$ as a new function

$$l(\boldsymbol{X};\theta) \doteq g(\boldsymbol{w}(\theta), \theta). \tag{4.20}$$

Instead of directly maximizing the $l(\boldsymbol{X};\theta)$ with respect to θ, we may maximize $g(\boldsymbol{w}(\theta), \theta)$ in a "hill-climbing" style by iterating between the following two steps:

Step 1. partially maximizing $g(\boldsymbol{w}(\theta), \theta)$ with respect to $\boldsymbol{w}(\theta)$ with θ (the second argument) fixed;

Step 2. partially maximizing $g(\boldsymbol{w}(\theta), \theta)$ with respect to the second θ with $\boldsymbol{w}(\theta)$ fixed (to the value obtained from Step 1.)

Notice that at each step the value of $g(\boldsymbol{w}(\theta), \theta)$ does not decrease, so neither does that of $l(\boldsymbol{X};\theta)$. When the iteration converges to a stationary point θ^*, it must be a (local) extremum for the function $l(\boldsymbol{X};\theta)$. To see this, examine the equation

$$\frac{dl(\boldsymbol{X};\theta)}{d\theta} = \frac{\partial g(\boldsymbol{w},\theta)}{\partial \boldsymbol{w}}\frac{\partial \boldsymbol{w}(\theta)}{\partial \theta} + \frac{\partial g(\boldsymbol{w},\theta)}{\partial \theta}. \tag{4.21}$$

[4]That is, it is the expected value of the complete log-likelihood $\log p(\boldsymbol{x}, z|\theta)$ of the "complete" random vector $(\boldsymbol{x}, z)$ with respect to the distribution of $(z|\boldsymbol{x}, \theta)$.

[5]The entropy of a (discrete) random variable z is defined to be $H(z) \doteq \sum_j p(z = j) \log p(z = j)$.

Since θ^* must be a stationary point for each step, we have $\frac{\partial g(\boldsymbol{w},\theta)}{\partial \boldsymbol{w}}\Big|_{\theta^*} = 0$ and $\frac{\partial g(\boldsymbol{w},\theta)}{\partial \theta}\Big|_{\theta^*} = 0$. Therefore, $\frac{dl(\boldsymbol{X};\theta)}{d\theta}\Big|_{\theta^*} = 0$.

As we have alluded to earlier, the main reason for choosing this alternative maximization is that, for the log-likelihood function of a mixture of distributions, each of these two steps of maximizing g are much easier to compute than directly maximizing the original log-likelihood function. In fact, for Gaussian distributions, one can often find closed-form solutions to each step.

E-Step: Expected Membership of Samples. To find the optimal $\hat{\boldsymbol{w}} = \{\hat{w}_{ij}\}$ that maximize $g(\boldsymbol{w},\theta)$, we need to maximize the function

$$\max_{\boldsymbol{w}} g(\boldsymbol{w},\theta) = \sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\log p(\boldsymbol{x}_i, z_i = j|\theta) - \sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\log w_{ij} \tag{4.22}$$

with respect to $\boldsymbol{w}$ subject to the constraints $\sum_j w_{ij} = 1$ for every i. For this purpose, we have the following statement.

Proposition 4.2 (Expected Membership)**.** *The optimal $\hat{\boldsymbol{w}}$ that partially maximizes $g(\boldsymbol{w},\theta)$ is given by:*

$$\hat{w}_{ij} = \frac{\pi_j p(\boldsymbol{x}_i|z_i = j,\theta)}{\sum_{\ell=1}^{n}\pi_\ell p(\boldsymbol{x}_i|z_i = \ell,\theta)}. \tag{4.23}$$

Proof. Using the Lagrange multipliers method, we differentiate the objective function

$$\sum_{i=1}^{N}\sum_{j=1}^{n}\Big(w_{ij}\log p(\boldsymbol{x}_i, z_i = j|\theta) - w_{ij}\log w_{ij}\Big) + \sum_{i=1}^{N}\lambda_i\Big(\sum_{j=1}^{n} w_{ij} - 1\Big). \tag{4.24}$$

with respect to w_{ij} and set the derivatives to zero. We obtain the necessary conditions for extrema:

$$\log p(\boldsymbol{x}_i, z_i = j|\theta) - \log w_{ij} - 1 + \lambda_i = 0 \tag{4.25}$$

for every i and j. Solving for w_{ij} from this equation, we obtain:

$$w_{ij} = e^{\lambda_i - 1} p(\boldsymbol{x}_i, z_i = j|\theta). \tag{4.26}$$

Since $\sum_j w_{ij} = 1$, we have $e^{\lambda_i-1} = \big(\sum_\ell p(\boldsymbol{x}_i, z_i = \ell|\theta)\big)^{-1}$. In addition,

$$p(\boldsymbol{x}_i, z_i = j|\theta) \;=\; p(\boldsymbol{x}_i|z_i = j,\theta)p(z_i = j|\theta) \;=\; \pi_j p(\boldsymbol{x}_i|z_i = j,\theta).$$

We hence have the claim of the proposition. □

M-Step: Maximize the Expected Complete Log-Likelihood. Now we consider the second step in which we fix $\boldsymbol{w}$ and maximize $g(\boldsymbol{w},\theta)$ with respect to θ. This means we fix $w_{ij} = p(z_i = j|\boldsymbol{x}_i,\theta)$ in the expression of $l(\boldsymbol{X};\theta)$. The second term (4.19) of $l(\boldsymbol{X};\theta)$ is therefore fixed as far as this step is concerned. Hence it is equivalent to maximizing the first term (4.18), the so-called expected complete log-likelihood:

$$L(\boldsymbol{X};\theta) \doteq \sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\log\big(\pi_j p(\boldsymbol{x}_i|z_i=j,\theta)\big). \tag{4.27}$$

For many common choices of the distributions $p(\boldsymbol{x}|z=j,\theta)$, we can find closed-form solutions to maximize $L(\boldsymbol{X};\theta)$.

For simplicity, in the clustering analysis, we may assume that each cluster is an isotropic normal distribution, i.e., $p(\boldsymbol{x}|z=j,\theta)=\mathcal{N}(\boldsymbol{\mu}_j,\sigma_j^2 I)$. Maximizing $L(\boldsymbol{X};\theta)$ is then equivalent to maximizing the function

$$\sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\left(\log\pi_j - D\log\sigma_j - \frac{\|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|^2}{\sigma_j^2}\right), \tag{4.28}$$

where we have omitted terms that depend on only the fixed w_{ij} and constants. The goal of maximization is to find the parameters $\hat{\theta}=\{(\hat{\boldsymbol{\mu}}_j,\hat{\sigma}_j,\hat{\pi}_j)\}_{j=1}^n$ that maximize the above expression. Since $\sum_{j=1}^n \pi_j = 1$, this is a constrained optimization problem, which can be solved in closed-form using the Lagrange-multiplier method. We here give below the formulae but leave the derivation to the reader as an exercise (see Exercise 4.2):

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{i=1}^N w_{ij}\boldsymbol{x}_i}{\sum_{i=1}^N w_{ij}}, \quad \hat{\sigma}_j^2 = \frac{\sum_{i=1}^N w_{ij}\|\boldsymbol{x}_i-\hat{\boldsymbol{\mu}}_j\|^2}{D\sum_{i=1}^N w_{ij}}, \quad \hat{\pi}_j = \frac{\sum_{i=1}^N w_{ij}}{N}. \tag{4.29}$$

We summarize the above results as Algorithm 4.2.

Instead of using a deterministic map to assign each point $\boldsymbol{x}_i$ to a cluster (as in the K-means algorithm 4.1, where $j=c(i)$), the EM algorithm assigns the point $\boldsymbol{x}_i$ "softly" to each cluster according to a set of probabilities $\{w_{ij}\}$ (that are subject to $\sum_{j=1}^n w_{ij}=1$). Subsequently, the number of points N_j in the jth cluster is expected to be $\sum_{i=1}^N w_{ij}$; the ratio $\frac{N_j}{N}$ is expected as $\frac{\sum_{i=1}^N w_{ij}}{N}$; and the means $\boldsymbol{\mu}_j$ in (4.12) are replaced by an expected version in (4.31). In general, if the variances σ_j are significantly smaller than the distances between the means $\boldsymbol{\mu}_j$, the K-means and EM algorithms give similar clustering results.

From the above derivation, each step of the EM algorithm increases the log-likelihood function $l(\boldsymbol{X};\theta)$. However, beware that a stationary value θ^* that the algorithm converges to is not necessarily the global maximum (if the global maximum exists at all). Furthermore, for distributions as simple as a mixture of Gaussian distributions, the global maximum may not even exist! We illustrate this via the following example.

Example 4.3 (ML Estimate of Two Mixed Gaussians [Vapnik, 1995]). Consider a distribution $p(x), x\in\mathbb{R}$ that is a mixture of two Gaussian (normal) distributions:

$$p(x,\mu,\sigma) = \frac{1}{2\sigma\sqrt{2\pi}}\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} + \frac{1}{2\sqrt{2\pi}}\exp\left\{-\frac{x^2}{2}\right\}, \tag{4.32}$$

where $\theta=(\mu,\sigma)$ are unknown.

Algorithm 4.2 (Expectation Maximization).

Given a set of sample points $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^D$ drawn from n (isotropic) Gaussian clusters $\mathcal{N}(\boldsymbol{\mu}_j, \sigma_j^2 I), j = 1, 2, \ldots, n$, initialize the parameters $\theta = \{\boldsymbol{\mu}_j, \sigma_j, \pi_j\}$ with a set of vectors $\hat{\boldsymbol{\mu}}_j^{(0)} \in \mathbb{R}^D$ and scalars $\hat{\sigma}_j^{(0)}, \hat{\pi}_j^{(0)} \in \mathbb{R}$.
Let $m = 0$.

1. **Expectation:** Using the current estimate for the parameters $\hat{\theta}^{(m)} = \{\hat{\boldsymbol{\mu}}_j^{(m)}, \hat{\sigma}_j^{(m)}, \hat{\pi}_j^{(m)}\}$, compute the estimate of w_{ij} as

$$w_{ij}^{(m)} = p(z_i = j | \boldsymbol{x}_i, \hat{\theta}^{(m)}) = \frac{\hat{\pi}_j^{(m)} p(\boldsymbol{x}_i | z_i = j, \hat{\theta}^{(m)})}{\sum_{\ell=1}^n \hat{\pi}_\ell^{(m)} p(\boldsymbol{x}_i | z_i = \ell, \hat{\theta}^{(m)})}, \tag{4.30}$$

   where $p(\boldsymbol{x} | z = j, \theta)$ is given in (4.39).

2. **Maximization:** Using the estimated $w_{ij}^{(m)}$, update the estimates for the parameters $\hat{\boldsymbol{\mu}}_j$, $\hat{\sigma}_j$ as:

$$\hat{\boldsymbol{\mu}}_j^{(m+1)} = \frac{\sum_{i=1}^N w_{ij}^{(m)} \boldsymbol{x}_i}{\sum_{i=1}^N w_{ij}^{(m)}}, \quad \left(\hat{\sigma}_j^{(m+1)}\right)^2 = \frac{\sum_{i=1}^N w_{ij}^{(m)} \left\| \boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j^{(m+1)} \right\|^2}{D \sum_{i=1}^N w_{ij}^{(m)}}, \tag{4.31}$$

   and update $\hat{\pi}_j$ as $\hat{\pi}_j^{(m+1)} = \frac{\sum_{i=1}^N w_{ij}^{(m)}}{N}$.

Let $m \leftarrow m + 1$, and repeat Steps 1 and 2 until the update in the parameters is small enough.

Then for any data $\boldsymbol{X} = \{x_1, x_2, \ldots, x_N\}$ and for any given constant $A > 0$, there exists a small σ_0 such that for $\mu = x_1$ the log-likelihood will exceed A (regardless of the true μ, σ):

$$\begin{aligned} l(\boldsymbol{X}; \theta)\big|_{\mu = x_1, \sigma = \sigma_0} &= \sum_{i=1}^N \ln p(x_i \mid \mu = x_1, \sigma = \sigma_0) \\ &> \ln\left(\frac{1}{2\sigma_0\sqrt{2\pi}}\right) + \sum_{i=2}^N \ln\left(\frac{1}{2\sqrt{2\pi}} \exp\left\{-\frac{x_i^2}{2}\right\}\right) \\ &= -\ln \sigma_0 - \sum_{i=1}^N \frac{x_i^2}{2} - N \ln 2\sqrt{2\pi} > A. \end{aligned}$$

Therefore, the maximum of the log-likelihood does not exist, and the ML objective does not provide a solution to estimating the unknown parameters. In fact, in this case, the true parameter corresponds to the largest (finite) local maximum of the log-likelihood. ∎

From the simple example, we can conclude that the ML method only applies to very restrictive set of densities.[6] If we insist using it for mixtures of Gaussians, we have to rule out the situations that the variance can be arbitrarily small, i.e., $\sigma_0 \rightarrow$

[6]For instance, a class of density functions that are bounded by a common finite value from above.

0. Fortunately, in practice, the EM algorithm typically tends to avoid such singular directions and is able to converge to a local maximum that represents the true parameters if a reasonable initialization is given. However, this leads to another potential problem: What if the distributions to be estimated are indeed close to being singular? This is unfortunately the case with subspace-like distributions.[7] Thus, singular distributions like subspaces require special treatment.

Also notice that the above K-means and EM algorithms are derived mainly for isotropic Gaussian distributions. In practice, a cluster is rarely isotropic. For instance, as we have seen in PCA, a cluster can be a set of points sampled from a principal subspace. For the above reasons, in the next two sections of this chapter (Section 3.1 and 4.2), we will extend the basic ideas of K-means and EM to the case in which clusters are subspaces.

4.2 Subspace-Segmentation Algorithms

In this section, we generalize the K-means and EM algorithms to estimate arrangements of principal subspaces and cluster points into subspaces. They can both be viewed as certain extension of PCA to multiple principal subspaces. Both algorithms assume that the number of subspaces n and their dimensions $d_j, j = 1, 2, \dots, n$ are known. They estimate a basis for each subspace and the segmentation of the data by optimizing certain objective functions, namely the least-squares error in the geometric setting or the log-likelihood in the statistical setting. Since the optimal solution is normally not available in closed-form, the optimization problem is solved by iterating between the segmentation of the data points and the estimation of the subspace bases, starting from an initial guess for the subspace bases.

The following sections give a detailed description of both algorithms tailored to Problem 3.1. The goal is to reveal the similarity and difference between these two algorithms as well as their advantages and disadvantages.

4.2.1 K-Subspaces

If the number of subspaces n and their dimensions $d_j, j = 1, 2, \dots, n$ are known, then the problem of fitting multiple subspaces to the data is to find orthogonal matrices $U_j, j = 1, 2, \dots, n$ of dimension $D \times d_j$ such that

$$\forall i \ \exists j \text{ such that } \ \boldsymbol{x}_i = U_j \boldsymbol{y}_i, \tag{4.33}$$

where $i \in \{1, 2, \dots, N\}$ and $j \in \{1, 2, \dots, n\}$. Once the assignment map $c(i) = j$ is found for each point $\boldsymbol{x}_i$, $\boldsymbol{y}_i$ is simply given by $\boldsymbol{y}_i = U_{c(i)}^T \boldsymbol{x}_i$. When $\boldsymbol{x}_i$ is

[7] A subspace-like distribution is one that has large variance inside the subspace but very small (close to singular) variance in directions orthogonal to the subspace.

at the intersection of two subspaces, the solution for $c(i)$ and therefore $\boldsymbol{y}_i$ is not unique. In this case, we arbitrarily choose one of the possible solutions.

In case the given points are corrupted by noise, we expect that the model parameters be found in a least-squares sense by minimizing the modeling error between $\boldsymbol{x}_i$ and its closest projection onto the subspaces:

$$\min_{\{U_j\}} \sum_{i=1}^{N} \min_j \left\| \boldsymbol{x}_i - U_j U_j^T \boldsymbol{x}_i \right\|^2, \tag{4.34}$$

where U_j is a $D \times d_j$ orthogonal matrix that represents a basis for the jth subspace $S_j, j = 1, 2, \ldots, n$. Unfortunately, unlike PCA, there is no constructive solution to the above minimization problem. The main difficulty is that the foregoing objective of (4.34) is hybrid – it is a combination of minimization on the continuous variables $\{U_j\}$ and the discrete variable j. Conventional nonlinear optimization techniques, such as gradient descent, do not directly apply to this case. Hence special optimization schemes have to be developed. For that purpose, we need to examine more closely the relationships between the two minimizations in the above objective function.

Suppose that some initial estimates $\hat{U}_1^{(0)}, \hat{U}_2^{(0)}, \ldots, \hat{U}_n^{(0)}$ of the subspaces are available. Then we can easily minimize the objective (4.34) for j. That is, for each subspace S_j defined by $\hat{U}_j^{(0)}$, we obtain the subset of points $\boldsymbol{X}_j^{(0)}$ that are closer to S_j than to any other subspace. The data set $\boldsymbol{X}$ is therefore segmented into n groups

$$\boldsymbol{X} = \boldsymbol{X}_1^{(0)} \cup \boldsymbol{X}_2^{(0)} \cup \cdots \cup \boldsymbol{X}_n^{(0)}, \tag{4.35}$$

and we further require $\boldsymbol{X}_i^{(0)} \cap \boldsymbol{X}_j^{(0)} = \emptyset$ for $i \neq j$.[8]

Knowing the membership of each point $\boldsymbol{x}_i$ from the above segmentation, the objective (4.34) can be rewritten as:

$$\sum_{j=1}^{n} \left(\min_{U_j} \sum_{\boldsymbol{x}_i \in \boldsymbol{X}_j^{(0)}} \left\| \boldsymbol{x}_i - U_j U_j^T \boldsymbol{x}_i \right\|^2 \right). \tag{4.36}$$

Notice that the minimization inside the bracket is exactly the same as the minimization in (2.22). Consequently, we have solved this problem in Theorem 2.2 for PCA. We can therefore apply PCA to each group of points $\{\boldsymbol{X}_j^{(0)}\}$ to obtain new estimates for the bases $\{\hat{U}_j^{(1)}\}$. Such estimates give a modeling error no larger than the error given by the initial estimates $\{\hat{U}_j^{(0)}\}$.

We can further reduce the modeling error by re-assigning each data point $\boldsymbol{x}_i$ to its closest subspace according to the new estimates $\{\hat{U}_j^{(1)}\}$. In this way, we obtain a new segmentation $\boldsymbol{X} = \boldsymbol{X}_1^{(1)} \cup \boldsymbol{X}_2^{(1)} \cup \cdots \cup \boldsymbol{X}_n^{(1)}$. If we keep iterating

[8]If a point $\boldsymbol{x} \in \boldsymbol{X}$ has the same minimal distance to more than one subspace, then we assign it to an arbitrary subspace.

between the above two steps, the modeling error will keep decreasing until its value stabilizes to a (local) equilibrium and the segmentation no longer changes. This minimization process is in essence an extension of the K-means algorithm to subspaces. We summarize the algorithm as Algorithm 4.3.

Algorithm 4.3 (K-Subspaces: K-Means for Subspace Segmentation).

Given a set of noisy sample points $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ drawn from n subspaces with the dimensions $d_j, j = 1, 2, \ldots, n$, initialize the bases of the subspaces with a set of orthogonal matrices $\hat{U}_j^{(0)} \in \mathbb{R}^{D \times d_j}$.
Let $m = 0$.

1. **Segmentation:** For each point $\boldsymbol{x}_i \in \boldsymbol{X}$, assign it to $\boldsymbol{X}_j^{(m)}$ if

$$j = \arg \min_{\ell=1,\ldots,n} \left\| \boldsymbol{x}_i - \hat{U}_\ell^{(m)} \big(\hat{U}_\ell^{(m)}\big)^T \boldsymbol{x}_i \right\|^2.$$

   If the above cost function is minimized by more than one subspace, assign the point arbitrarily to one of them.

2. **Estimation:** Apply PCA to each subset $\boldsymbol{X}_j^{(m)}$ using Theorem 2.2 and obtain new estimates for the subspace bases

$$\hat{U}_j^{(m+1)} = \text{PCA}\big(\boldsymbol{X}_j^{(m)}\big), \quad j = 1, 2, \ldots, n.$$

Let $m \leftarrow m+1$, and repeat Steps 1 and 2 until the segmentation does not change.

4.2.2 *Expectation Maximization for Subspaces*

To apply the EM method in Section 4.1.2 to subspaces, we need to assume a statistical model for the data. Following the general setting in Section 4.1.2, it is reasonable to assume that the data points $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ are samples drawn from multiple component distributions and each component distribution is centered around a subspace. To model from which component distribution a sample $\boldsymbol{x}$ is actually drawn, we again associate a latent discrete random variable $z \in \mathbb{R}$ to every data point $\boldsymbol{x}$, where each discrete random variable $z_i = j$ if the point $\boldsymbol{x}_i$ is drawn from the jth component, $i = 1, 2, \ldots, N$.

To model the fact that each component distribution has a principal subspace, say spanned by the columns of an orthogonal matrix $U_j \in \mathbb{R}^{D \times d_j}$, we may assume that the jth component distribution is a special Gaussian distribution determined by the following equation:

$$\boldsymbol{x} = U_j \boldsymbol{y} + B_j \boldsymbol{s}, \tag{4.37}$$

where the orthogonal matrix $B_j \in \mathbb{R}^{D \times (D-d_j)}$ is the orthogonal complement to the orthogonal matrix $U_j \in \mathbb{R}^{D \times d_j}$, and $\boldsymbol{y} \sim \mathcal{N}(0, \sigma_{\boldsymbol{y}}^2 I)$ and $\boldsymbol{s} \sim \mathcal{N}(0, \sigma_j^2 I)$. If we further assume that $\boldsymbol{y}$ and $\boldsymbol{s}$ are independent random variables, then we have

$$\Sigma_j^{-1} = \sigma_{\boldsymbol{y}}^{-2} U_j U_j^T + \sigma_j^{-2} B_j B_j^T. \tag{4.38}$$

The term $B_j\boldsymbol{s}$ models the projection error of $\boldsymbol{x}$ onto the subspace spanned by U_j. For $\boldsymbol{x}$ to be close to the subspace, one may assume $\sigma_j^2 \ll \sigma_{\boldsymbol{y}}^2$. Therefore, when $\sigma_{\boldsymbol{y}}^2 \to \infty$, we have $\Sigma_j^{-1} \to \sigma_j^{-2} B_j B_j^T$. In the limiting case, one essentially assumes a uniform distribution for $\boldsymbol{y}$ inside the subspace. The uniform assumption suggests that we do not care much about the distribution of the data inside the subspace – it is the subspace itself in which we are interested. Technically, this assumption also helps eliminate additional parameters so that the ML method may better avoid the difficulty shown in Example 4.3. In practice, this assumption is approximately valid as long as the variance of the data inside the subspace is significantly larger than that outside the subspace.

Therefore, in the sequel, we will adopt the limiting case as our probabilistic model for the derivation of the EM algorithm and derive closed-form formulae for the two steps of the EM algorithm. More precisely, we assume the distributions are

$$p(\boldsymbol{x}|z=j) \doteq \frac{1}{(2\pi\sigma_j^2)^{(D-d_j)/2}} \exp\left(-\frac{\boldsymbol{x}^T B_j B_j^T \boldsymbol{x}}{2\sigma_j^2}\right). \tag{4.39}$$

In the model, the parameters $\theta \doteq \{B_j, \sigma_j, \pi_j\}_{j=1}^n$ are unknowns and they need to be inferred from the samples of $\boldsymbol{x}$. The likelihood function (which is given by the marginal distribution of $\boldsymbol{x}$ given the parameters) is

$$\begin{aligned} p(\boldsymbol{x}|\theta) &= \sum_{z=1}^n p(\boldsymbol{x}|z,\theta)p(z|\theta) \\ &= \sum_{j=1}^n \frac{\pi_j}{(2\pi\sigma_j^2)^{(D-d_j)/2}} \exp\left(-\frac{\boldsymbol{x}^T B_j B_j^T \boldsymbol{x}}{2\sigma_j^2}\right). \end{aligned} \tag{4.40}$$

Then given the N samples $\boldsymbol{X} = \{\boldsymbol{x}_i\}$, estimates of the parameters $\hat{\theta}_{ML}$ are given by maximizing the log-likelihood function

$$l(\boldsymbol{X};\theta) \doteq \sum_{i=1}^N \log p(\boldsymbol{x}_i|\theta) \tag{4.41}$$

$$= \sum_{i=1}^N \log\left[\sum_{j=1}^n \frac{\pi_j}{(2\pi\sigma_j^2)^{(D-d_j)/2}} \exp\left(-\frac{\boldsymbol{x}_i^T B_j B_j^T \boldsymbol{x}_i}{2\sigma_j^2}\right)\right] \tag{4.42}$$

Again, this is in general a difficult high-dimensional optimization problem. Thus, we can apply the Expectation Maximization method introduced in Section 4.1.2. All the analysis in Section 4.1.2 directly applies to this new log-likelihood function except that in the M-Step, under the new probabilistic model, the new expected complete log-likelihood $L(\boldsymbol{X};\theta)$ becomes

$$\sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\left(\log \pi_j - (D-d_j)\log\sigma_j - \frac{\|B_j^T\boldsymbol{x}_i\|^2}{2\sigma_j^2}\right), \tag{4.43}$$

where, as before, we have omitted terms that depend on only the fixed w_{ij} and constants. The goal now is to find the parameters $\hat{\theta} = \{(\hat{B}_j, \hat{\sigma}_j, \hat{\pi}_j)\}_{j=1}^n$ that maximize the above expected complete log-likelihood. Since $B_j^T B_j = I$ and $\sum_{j=1}^n \pi_j = 1$, this is again a constrained optimization problem, whose solutions are given by the following proposition.

Proposition 4.4 (Maximum of the Expected Complete Log-Likelihood)**.** *The parameters* $\hat{\theta} = \{\hat{B}_j, \hat{\sigma}_j, \hat{\pi}_j\}_{j=1}^n$ *that maximize the expected complete log-likelihood function (4.43) are:* $\hat{B}_j$ *are exactly the eigenvectors associated with the smallest* $D-d_j$ *eigenvalues of the weighted sample covariance matrix* $\hat{\Sigma}_j \doteq \sum_{i=1}^N w_{ij}\boldsymbol{x}_i\boldsymbol{x}_i^T$*, and* π_j *and* σ_j^2 *are*

$$\hat{\pi}_j = \frac{\sum_{i=1}^N w_{ij}}{N}, \quad \hat{\sigma}_j^2 = \frac{\sum_{i=1}^N w_{ij}\|\hat{B}_j^T\boldsymbol{x}^i\|^2}{(D-d_j)\sum_{i=1}^N w_{ij}}. \tag{4.44}$$

Proof. The part of objective function associated with the bases $\{B_j\}$ can be rewritten as

$$\sum_{i=1}^{N}\sum_{j=1}^{n} -w_{ij}\frac{\|B_j^T\boldsymbol{x}_i\|^2}{2\sigma_j^2} = \sum_{j=1}^{n} -\text{trace}\left(\frac{B_j^T\hat{\Sigma}_j B_j}{2\sigma_j^2}\right), \tag{4.45}$$

where $\hat{\Sigma}_j = \sum_{i=1}^N w_{ij}\boldsymbol{x}_i\boldsymbol{x}_i^T$. Differentiating the Lagrangian associated with B_j and setting the derivatives to zero, we obtain the necessary conditions for extrema:

$$\sum_{j=1}^{n} -\text{trace}\left(\frac{B_j^T\hat{\Sigma}_j B_j}{2\sigma_j^2}\right) + \text{trace}\big(\Lambda_j(B_j^T B_j - I)\big) \quad \Rightarrow \quad \hat{\Sigma}_j B_j = 2\sigma_j^2 B_j\Lambda_j,$$

where Λ_j is a matrix of Lagrangian multipliers. Since $B_j^T B_j = I$, the objective function for B_j becomes $-\sum_{j=1}^n \text{trace}(\Lambda_j)$. Thus $\hat{B}_j$ can be obtained as the matrix whose columns are the eigenvectors of $\hat{\Sigma}_j$ associated with the $(D-d_j)$ smallest eigenvalues.

From the Lagrangian associated with the mixing proportions $\{\pi_j\}$, we have

$$\min \sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\log(\pi_j) + \lambda\Big(1 - \sum_{j=1}^{n}\pi_j\Big) \quad \Rightarrow \quad \hat{\pi}_j = \frac{\sum_{i=1}^N w_{ij}}{N}. \tag{4.46}$$

Finally, after taking the derivative of the expected log-likelihood with respect to σ_j and setting it to zero, we obtain

$$\hat{\sigma}_j^2 = \frac{\sum_{i=1}^N w_{ij}\|\hat{B}_j^T\boldsymbol{x}^i\|^2}{(D-d_j)\sum_{i=1}^N w_{ij}}. \tag{4.47}$$

□

We summarize the above results as Algorithm 4.4.

Algorithm 4.4 (EM for Subspace Segmentation).

Given a set of sample points $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^D$, the number of subspaces n and the dimensions d_j, initialize the parameters $\theta = \{B_j, \sigma_j, \pi_j\}$ with a set of initial orthogonal matrices $\hat{B}_j^{(0)} \in \mathbb{R}^{D\times(D-d_j)}$ and scalars $\hat{\sigma}_j^{(0)}, \hat{\pi}_j^{(0)}, j = 1, 2, \ldots, n$. Let $m = 0$.

1. **Expectation:** Using the current estimate for the parameters $\hat{\theta}^{(m)} = \{\hat{B}_j^{(m)}, \hat{\sigma}_j^{(m)}, \hat{\pi}_j^{(m)}\}$, compute the estimate of w_{ij} as

$$w_{ij}^{(m)} = p(z_i = j|\boldsymbol{x}_i, \hat{\theta}^{(m)}) = \frac{\hat{\pi}_j^{(m)} p(\boldsymbol{x}_i|z_i = j, \hat{\theta}^{(m)})}{\sum_{\ell=1}^n \hat{\pi}_\ell^{(m)} p(\boldsymbol{x}_i|z_i = \ell, \hat{\theta}^{(m)})}, \tag{4.48}$$

   where $p(\boldsymbol{x}|z = j, \theta)$ is given in (4.39).

2. **Maximization:** Using the estimated $w_{ij}^{(m)}$, compute $\hat{B}_j^{(m+1)}$ as the eigenvectors associated with the smallest $D - d_j$ eigenvalues of the matrix $\hat{\Sigma}_j^{(m)} \doteq \sum_{i=1}^N w_{ij}^{(m)} \boldsymbol{x}_i \boldsymbol{x}_i^T$, and update $\hat{\pi}_j$ and $\hat{\sigma}_j$ as:

$$\hat{\pi}_j^{(m+1)} = \frac{\sum_{i=1}^N w_{ij}^{(m)}}{N}, \quad \left(\hat{\sigma}_j^{(m+1)}\right)^2 = \frac{\sum_{i=1}^N w_{ij}^{(m)} \left\|\left(\hat{B}_j^{(m+1)}\right)^T \boldsymbol{x}_i\right\|^2}{(D - d_j)\sum_{i=1}^N w_{ij}^{(m)}}. \tag{4.49}$$

Let $m \leftarrow m + 1$, and repeat Steps 1 and 2 until the update in the parameters is small enough.

4.2.3 *Relationships between K-Subspaces and EM*

As we have seen in the above, both K-subspaces and EM are algorithms that can be used to analyze arrangements of principal subspaces and fit multiple subspaces to a given set of data points. Both algorithms optimize their objectives via an iterative scheme. The overall structure of the two algorithms is also very much similar: the "Segmentation" step in K-subspaces is replaced by the "Expectation" step in EM; and "Estimation" by "Maximization".

In addition to the structural similarity, there are also subtle technical relationships between the two steps of K-subspaces and EM. To see this, let us further assume that in the EM algorithm, the noise has the same variance for all the subspaces (i.e., $\sigma = \sigma_1 = \cdots = \sigma_n$). According to equation (4.45), the EM algorithm updates the estimates for the subspaces in the "Maximization" step by minimizing the objective function:

$$\min_{\{B_j\}} \sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\left\|B_j^T \boldsymbol{x}_i\right\|^2 = \min_{\{U_j\}} \sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\left\|\boldsymbol{x}_i - U_j U_j^T \boldsymbol{x}_i\right\|^2, \tag{4.50}$$

where the equality is due to the identity $B_j B_j^T = I - U_j U_j^T$. For EM, the weights w_{ij} are computed from the "Expectation" step as the expected membership of $\boldsymbol{x}_i$ in the subspaces j according to the equation (4.23), and w_{ij} in general take continuous values between 0 and 1. For K-subspaces, however, w_{ij} is a discrete variable and it is computed in the "Segmentation" step as (see Algorithm 4.3):

$$w_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_{\ell=1,\ldots,n} \|B_\ell^T \boldsymbol{x}_i\|^2, \\ 0 & \text{otherwise.} \end{cases} \tag{4.51}$$

Then the objective function (4.50) can be rewritten as:

$$\min_{\{U_j\}} \sum_{i=1}^{N}\sum_{j=1}^{n} w_{ij}\left\|\boldsymbol{x}_i - U_j U_j^T \boldsymbol{x}_i\right\|^2 = \min_{\{U_j\}} \sum_{i=1}^{N} \min_j \left\|\boldsymbol{x}_i - U_j U_j^T \boldsymbol{x}_i\right\|^2, \tag{4.52}$$

which is exactly the same objective function (4.34) for K-subspaces. This is also the reason why both K-subspaces and EM rely on the eigenvalue decomposition (or singular value decomposition) of the sample covariance matrix to estimate the basis for each subspace.

Based on the above analysis, the only conceptual difference between the K-subspaces and EM algorithm is: At each iteration, the K-subspaces algorithm gives a "definite" assignment of every data point into one of the subspaces; but the EM algorithm views the membership as a random variable and uses its expected value to give a "probabilistic" assignment of the data point. Because of this difference, for the same set of data points, the "subspaces" found by using K-subspaces and EM will in general be different, although normally the difference is expected to be small. A precise quantitative characterization of the difference between the solutions by K-subspaces and EM remains an open question. Also because of this difference, the K-subspaces algorithm is less dependent on the correct knowledge of the dimension of each subspace: As long as the initial subspaces may segment the data well enough, both the basis and the dimension of each subspace can be updated at the Estimation step. However, the EM algorithm, at least for the version we presented above, depends explicitly on correct knowledge in both the number of subspaces and their dimensions. In addition, both algorithms require a good initialization so that they are more likely to converge to the optimal solution (e.g., the global maximum of the log likelihood) when the iteration stabilizes. In the next chapter, we will show how these difficulties can be resolved by a new algebraic method for identifying arrangements of principal subspaces.

4.3 Relationships between GPCA, K-Subspaces, and EM

In Section 4.2.3, we have discussed the relationships between K-subspaces and EM. In this section, we reveal their relationships with GPCA through the special case of hyperplane arrangements. Let $\boldsymbol{b}_j$ be the normal vectors to an arrangement of hyperplanes S_j, $j = 1, 2, \ldots, n$, respectively.

We know from Chapter 4 that, under reasonable assumptions, both the K-subspaces and the EM methods minimize an objective of the form

$$\min_{\{\boldsymbol{b}_j\}} \sum_{i=1}^{N} \sum_{j=1}^{n} w_{ij} \left\| \boldsymbol{b}_j^T \boldsymbol{x}_i \right\|^2. \tag{4.53}$$

In the case of K-subspaces, w_{ij} is a "hard" assignment of $\boldsymbol{x}_i$ to the subspaces: $w_{ij} = 1$ only if $\boldsymbol{x}_i \in S_j$ and 0 otherwise. The above objective function becomes exactly the geometric modeling error. In the case of EM, $w_{ij} \in [0, 1]$ is the probability of the latent random variable $z_i = j$ given $\boldsymbol{x}_i$. Then w_{ij} plays the role as a "soft" assignment of $\boldsymbol{x}_i$ to group j.

Following the same line of reasoning, we can replace w_{ij} with an even "softer" assignment of membership:

$$w_{ij} \doteq \frac{1}{n} \prod_{l \neq j} \left\| \boldsymbol{b}_l^T \boldsymbol{x}_i \right\|^2 \ \in \mathbb{R}. \tag{4.54}$$

Notice that, in general, the value of w_{ij} is large when $\boldsymbol{x}_i$ belongs to (or is close to) S_j, and the value is small when $\boldsymbol{x}_i$ belongs to (or is close to) any other subspace. With this choice of w_{ij}, the objective function becomes

$$\min_{\{\boldsymbol{b}_j\}} \sum_{i=1}^{N} \sum_{j=1}^{n} \Big(\frac{1}{n} \prod_{l \neq j} \left\| \boldsymbol{b}_l^T \boldsymbol{x}_i \right\|^2 \Big) \left\| \boldsymbol{b}_j^T \boldsymbol{x}_i \right\|^2 = \sum_{i=1}^{N} \prod_{j=1}^{n} \left\| \boldsymbol{b}_j^T \boldsymbol{x}_i \right\|^2. \tag{4.55}$$

This is exactly the objective function that all the algebraic methods are based upon. To see this, notice that

$$\sum_{i=1}^{N} \prod_{j=1}^{n} \left\| \boldsymbol{b}_j^T \boldsymbol{x}_i \right\|^2 = \sum_{i=1}^{N} p_n(\boldsymbol{x}_i)^2 = \sum_{i=1}^{N} \left(\boldsymbol{c}_n^T \nu_n(\boldsymbol{x}_i) \right)^2. \tag{4.56}$$

Not so surprisingly, we end up with a "least-squares like" formulation in terms of the embedded data $\nu_n(\boldsymbol{x})$ and the coefficient vector $\boldsymbol{c}_n$. Notice that the above objective function can be rewritten as

$$\sum_{i=1}^{N} \left(\boldsymbol{c}_n^T \nu_n(\boldsymbol{x}_i) \right)^2 = \left\| \boldsymbol{V}_n(D) \boldsymbol{c}_n \right\|^2. \tag{4.57}$$

The least-squares solution of $\boldsymbol{c}_n$ is exactly given by the eigenvector associated with the smallest eigenvalue of the matrix $\boldsymbol{V}_n(D)$.

The K-subspaces or EM methods minimizes its objective iteratively using $\boldsymbol{b}_j$ computed in the previous iteration. However, one key observation in the GPCA algorithm is that the derivative of the vanishing polynomial $p_n(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x})$ at the sample points provide information about the normal vectors $\boldsymbol{b}_j$. Therefore, the GPCA algorithm does not require initialization and iteration but still achieves a goal similar to that of K-subspaces or EM.

4.4 Bibliographic Notes

When the data points lie on an arrangement of subspaces, the modeling problem was initially treated as "chicken-and-egg" and tackled with iterative methods, such as the K-means and EM algorithms. The basic ideas of K-means clustering goes back to [Lloyd, 1957, Forgy, 1965, Jancey, 1966, MacQueen, 1967]. Its probabilistic counterpart, the Expectation Maximization (EM) algorithm is due to [Dempster et al., 1977]. See Appendix A for a more general review. For a more thorough and complete exposition of EM, one may refer to [Neal and Hinton, 1998] or the book of [McLanchlan and Krishnan, 1997].

In [Tipping and Bishop, 1999a], the classical PCA has been extended to the mixtures of probabilistic PCA, and the maximum-likelihood solution was recommended to be found by the EM algorithm too. The classical K-means algorithm was also extended to the case of subspaces, called K-subspace [Ho et al., 2003]. Some other algorithms such as the subspace growing and the subspace selection algorithm [Leonardis et al., 2002] were also proposed in different contexts. Unfortunately, as we have alluded to above, iterative methods are sensitive to initialization, hence they may not converge to the global optimum. This has severely limited the performance and generality of such methods in solving practical problems in computer vision or image processing [Shi and Malik, 1998, Torr et al., 2001]. Thus, in the next chapter, we will change the tools a little bit and seek for alternative solutions to the subspace segmentation problem.

4.5 Exercises

Exercise 4.1 (K-Means for Image Segmentation). K-means is a very useful and simple algorithm for many practical problems that require clustering multivariate data. In this exercise, implement the K-means algorithm 4.1 and apply it to the segmentation of color (RGB) images. Play with the number of segments and the choice of the window size (i.e., instead of using only the RGB values at the pixel, use also the RGB values of a window of surrounding pixels).

Exercise 4.2 (Maximizing the Expected Log-Likelihood of Gaussians). Show that the formulae given in equation (4.29) are the solutions for maximizing the expected log-likelihood $L(\boldsymbol{X};\theta)$ (4.27) for isotropic Gaussian distributions $p(\boldsymbol{x}|z = j, \theta) = \mathcal{N}(\boldsymbol{\mu}_j \sigma_j^2 I)$.

Exercise 4.3 (Two Subspaces in General Position). Consider two linear subspaces of dimension d_1 and d_2 respectively in $\mathbb{R}^D$. We say they are in general position if an arbitrary (small) perturbation of the position of the subspaces does not change the dimension of their intersection. Show that two subspaces are in general position if and only if

$$\dim(S_1 \cap S_2) = \min\{d_1 + d_2 - D, ; 0\}. \tag{4.58}$$

Exercise 4.4 (Segmenting Three Planes in $\mathbb{R}^3$). Customize and implement (in MATLAB) the K-subspaces algorithm 4.3 and the EM-algorithm 4.4 for the purpose of segmenting three planes in $\mathbb{R}^3$. Randomly generate three subspaces and draw a number of (say uniformly distributed) sample points on the planes. Use the algorithms to segment the samples. Play with the level of noise (added to the samples) and the number of random initializations of the algorithm.

Chapter 5
Agglomerative Methods for Multiple-Subspace Segmentation

"The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work."
– John von Neumann

So far, we have introduced several methods for estimation and segmentation of multiple subspaces, utilizing either statistical or algebraic techniques. A common assumption behind these approaches is that a good estimate of the underlying mixture models is necessary for the segmentation of the data. The goodness of the segmentation relies on how good the estimate is. In the literature, there is yet another spectrum of data clustering algorithms that follow different set of criteria and do not explicitly cast the segmentation problem as model estimation, those as known as the *agglomerative methods* [**?**]. In this chapter, we introduce a new agglomerative algorithm for clustering mixed data drawn from multiple subspaces. We use this algorithm to establish some fundamental connections between the statistical and algebraic approaches studied separately in earlier chapters, or more specifically, between (almost degenerate) Gaussian distributions and (linear) subspaces.

In addition, the rationale behind this new algorithm will offer new perspectives on some fundamental issues that have not been addressed by either of the previous approaches:

1. In what sense the segmented data is better than the original data set, or in other words, how should we measure the "gain" or "loss" of segmentation?
2. For a given data set, how do we automatically determine the optimal number of subspaces, as well as their dimensions?

To answer either question, we need a measure for the segmented data (and the resulting model) so as to tell one segmentation is better or worse than another. Towards the end of the previous chapter, we have introduced the notion of "effective dimension" as a measure of goodness for the segmented data and the resulting model. Essentially, it uses the total number of real coefficients needed to represent both the data and the model. One may argue that such a measure is rather coarse: For instance, it does not depend on the magnitude of the coefficients.

Thus, in this chapter, we study a more accurate measure which is supposed to be more physically meaningful: the number of binary bits needed to represent the segmented data. Therefore, a better segmentation means a shorter length of binary code for storing the data set on a digital computer. In fact, this is closely related to using the maximum likelihood (ML) criterion for estimating a model for the data. To see this, consider a set of data $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N)$ drawn from a mixture of distributions: $p(\boldsymbol{x}|\theta, \pi) \doteq \sum_{j=1}^{n} \pi_j p_j(\boldsymbol{x}|\theta_j)$. The maximum likelihood (ML) estimate is:

$$(\hat{\theta}, \hat{\pi})_{ML} = \arg\max_{\theta,\pi} \sum_{i=1}^{N} \log p(\boldsymbol{x}_i|\theta, \pi). \tag{5.1}$$

The ML criterion is equivalent to minimizing the negated log-likelihood: $\sum_i -\log p(\boldsymbol{x}_i|\theta, \pi)$, which is the expected coding length required to store the data using the optimal Shannon coding scheme for the distribution $p(\boldsymbol{x}|\hat{\theta}, \hat{\pi})$ [Cover and Thomas, 1991]. By law of large numbers, we know the quantity $\frac{1}{N}\sum_i -\log p(\boldsymbol{x}_i|\theta, \pi)$ converges to the *entropy* of the distribution

$$H(\boldsymbol{x}|\theta, \pi) \doteq \int -p(\boldsymbol{x}|\theta, \pi) \log p(\boldsymbol{x}|\theta, \pi)\, d\boldsymbol{x}. \tag{5.2}$$

However, the optimal coding length, or entropy, requires precise knowledge of the distribution $p(\cdot)$ while in reality only a finite number of samples are directly accessible. As we have seen in previous chapters, inferring the (mixture) distribution from the samples can be a rather difficult problem itself because we typically have to deal with a nonlinear and singular maximum likelihood function. Thus, the goal of this chapter is to seek for a good "surrogate" for the optimal Shannon coding length for the mixed data.

This leads to the question "what constitutes as a good surrogate?" Clearly, a good surrogate should be easily computable directly from the given sample data. The coding length given by the surrogate should be a close approximation to the minimum coding length required for the given sample data and converges asymptotically to the optimal coding length when the number of samples goes to infinity. Many non-parametric approximations of the distribution from the samples can be used to construct such a surrogate. Here we actually can do much better since we are dealing with subspaces or Gaussians, not an arbitrary distribution. For data drawn from this special class of models, we can obtain a close-form formula that gives an accurate estimate for the coding length.

Once we are able to accurately evaluate the coding length of any data set, we can decide whether a particular segmentation (or clustering) of the data set leads to a shorter coding length. The optimal segmentation of the data is the one that minimizes the overall coding length. Thus, data segmentation becomes the result of data compression. In this chapter, we will show how the overall coding length can be minimized in a simple agglomerative fashion, which leads to an extremely efficient and robust algorithm for segmenting mixtures of linear subspaces or Gaussians.

5.1 Basic Ideas and Algorithm

In this section, we give a self-contained summary of the main ideas and algorithm and leave more detailed mathematical analysis and justification to Section 5.2 and 5.3. Readers who are interested only in the algorithm and experiments may bypass those two sections and skip to Section 5.4 without any loss of continuity.

5.1.1 *Lossy Coding of Multivariate Data*

A lossy coding scheme maps a set of vectors $V = (v_1, v_2, \ldots, v_N) \in \mathbb{R}^{D\times N}$ to a sequence of binary bits, such that the original vectors can be recovered up to an allowable distortion $\mathbb{E}[\|v_i - \hat{v}_i\|^2] \le \varepsilon^2$. The length of the encoded sequence is denoted as the function $L(V) : \mathbb{R}^{D\times N} \to \mathbb{R}_+$.

In general, the coding scheme and the associated $L(\cdot)$ function can be chosen to be optimal for any family of distributions of interest. In the case where the data are i.i.d. samples from a zero-mean[1] multivariate Gaussian distribution $\mathcal{N}(0, \Sigma)$, the function $R = \frac{1}{2}\log_2 \det(I + \frac{D}{\varepsilon^2}\Sigma)$ provides a good approximation to the optimal rate-distortion function [Cover and Thomas, 1991].[2] As $\hat{\Sigma} = \frac{1}{N}VV^T$ is

[1]For simplicity, in the main text, we will derive and present our main results with the zero-mean assumption. However, all the formulas, results, and algorithms can be readily extended to the nonzero mean case, as shown in Appendix 5.B.

[2]Strictly speaking, the rate-distortion function for the Gaussian source $\mathcal{N}(0, \Sigma)$ is $R = \frac{1}{2}\log_2 \det\left(\frac{D}{\varepsilon^2}\Sigma\right)$ when $\frac{\varepsilon^2}{D}$ is smaller than the smallest eigenvalue of Σ. Thus the approximation is good only when the distortion ε is relatively small. However, when $\frac{\varepsilon^2}{D}$ is larger than some eigenvalues of Σ, the rate distortion function becomes more complicated [Cover and Thomas, 1991]. Nevertheless, the approximate formula $R = \frac{1}{2}\log_2 \det(I + \frac{D}{\varepsilon^2}\Sigma)$ can be viewed as the rate-distortion of the "regularized" source that works for all range of ε. Furthermore, as we will show in Appendix 5.A, the same formula gives a tight upper bound of the coding rate for any finite number of samples.

an estimate of the covariance Σ, the average number of bits needed per vector is:

$$R(V) \doteq \frac{1}{2}\log_2 \det\big(I + \frac{D}{\varepsilon^2 N} V V^T\big). \tag{5.3}$$

For readers who are less familiar with the rate-distortion theory, we will give an intuitive explanation of this formula in Section 5.2.

Representing the m vectors of V therefore requires $N \cdot R(V)$ bits. Since the optimal codebook is adaptive to the data V, we must also represent it with an additional $D \cdot R(V)$ bits[3], yielding an overall coding length of

$$L(V) \doteq (N + D) \cdot R(V) = \frac{N+D}{2}\log_2 \det\big(I + \frac{D}{\varepsilon^2 N} V V^T\big). \tag{5.4}$$

We will study the properties of this function in Section 5.2. For purposes of segmentation, it suffices to note that in addition to being (approximately) asymptotically optimal for Gaussian data, $L(V)$ also provides a tight bound on the number of bits needed to code a finite number of vectors when the underlying distribution is a degenerate or non-degenerate Gaussian (see Appendix 5.A for a proof).

5.1.2 *Segmentation via Data Compression*

Given a set of samples, $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{D\times N}$, one can always view them as drawn from a single Gaussian source and code $\boldsymbol{X}$ subject to distortion ε^2 using $L(W)$ bits. However, if the samples are drawn from a mixture of Gaussian distributions or subspaces, it may be more efficient to encode $\boldsymbol{X}$ as the union of multiple (disjoint) groups: $\boldsymbol{X} = \boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \cdots \cup \boldsymbol{X}_n$. If each group is coded separately, the total number of bits needed is

$$L^s(\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n) \doteq \sum_{i=1}^{n} L(\boldsymbol{X}_i) + |\boldsymbol{X}_i|\big(-\log_2(|\boldsymbol{X}_i|/N)\big), \tag{5.5}$$

where $|\boldsymbol{X}_i|$ indicates the cardinality (i.e. number of vectors) of the group $\boldsymbol{X}_i$. In the above expression, the term $\sum_{i=1}^{n} |\boldsymbol{X}_i|\big(-\log_2(|\boldsymbol{X}_i|/N)\big)$ is the number of bits needed to code (losslessly) the membership of the N samples in the n groups (e.g. using the Huffman coding [Cover and Thomas, 1991]).[4]

Then, given a fixed coding scheme with its associated coding length function $L(\cdot)$, an optimal segmentation is one which minimizes the segmented coding length, $L^s(\cdot)$, over all possible partitions of $\boldsymbol{X}$. Moreover, we will see that due to the properties of the rate-distortion function (5.3) for Gaussian data, softening the objective function (5.5) by allowing probabilistic (or fuzzy) segmentation

[3]This can be viewed as the cost of coding the D principal axes of the data covariance $\frac{1}{N}VV^T$. A more detailed explanation of $L(V)$ is given in Section 5.2.

[4]Here we assume that the ordering of the samples is random and entropy coding is the best we can do to code the membership. However, if the samples are ordered such that nearby samples more likely belong to the same group (e.g., in segmenting pixels of an image), the second term can and should be replaced by a tighter estimate.

does *not* further reduce the (expected) overall coding length (see Theorem 5.3 of Section 5.3).

Notice that the above objective (5.5) is a function of the distortion ε. In principle, one may add a "penalty" term, such as $ND \cdot \log \varepsilon$ to the overall coding length[5] L^s so as to determine the optimal distortion ε^*. The resulting objective $\min_\varepsilon L^s + ND \cdot \log \varepsilon$ will then correspond to an optimal coding length that only depends on the data. Nevertheless, very often we leave ε as a free parameter to be set by the user. In practice, this allows the user to potentially obtain hierarchical segmentation of the data at different scales of quantization. We will thoroughly examine how the value of ε affects the final segmentation through experiments in Section 5.4.

5.1.3 Minimizing the Coding Length

Finding the global minimum of the overall coding length L^s over all partitions of the dataset is a daunting combinatorial optimization problem, intractable for large data sets. Nevertheless, the coding length can be effectively minimized in a steepest descent fashion, as outlined in Algorithm 5.1. The minimization proceeds in a "bottom-up" fashion: initially, every sample is treated as its own group. At each iteration, two groups S_1 and S_2 are chosen so that merging them results in the greatest decrease in the coding length. The algorithm terminates when the coding length cannot be further reduced by merging any pair of groups.[6] A simple implementation which maintains a table containing $L^s(S_i \cup S_j)$ for all i, j requires $O(N^3 + N^2 D^3)$ time, where N is the number of samples and D the dimension of the space.

Algorithm 5.1 (Pairwise Steepest Descent of Coding Length).

```
1: Input: the data $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{D \times N}$ and a distortion $\varepsilon^2 > 0$.
2: initialize $\mathcal{S} = \{S_i = \{\boldsymbol{x}_i\} \mid \boldsymbol{x}_i \in \boldsymbol{X}\}$.
3: while $|\mathcal{S}| > 1$ do
4:     choose two distinct sets $S_i, S_j \in \mathcal{S}$ such that $L^s(S_i \cup S_j) - L^s(S_i, S_j)$
       is minimal.
5:     if $L^s(S_i \cup S_j) - L^s(S_i, S_j) \geq 0$ then break;
6:     else $\mathcal{S} \leftarrow \big(\mathcal{S} \setminus \{S_i, S_j\}\big) \cup \{S_i \cup S_j\}$.
7: end
8: Output: $\mathcal{S}$
```

Extensive simulations and experiments demonstrate that this algorithm is consistently and remarkably effective in segmenting data that are a mixture of

[5]This particular penalty term is justified by noticing that $ND \cdot \log \varepsilon$ is (within an additive constant) the number of bits required to code the residual $\boldsymbol{x} - \hat{\boldsymbol{x}}$ upto (very small) distortion $\delta \ll \varepsilon$.

[6]In the supplementary material, we have included a video showing the convergence of this algorithm on data drawn from mixtures of subspaces in $\mathbb{R}^3$.

Gaussians or subspaces (see Section 5.4). It tolerates significant amounts of outliers, and automatically determines the corresponding number of groups at any given distortion. As a greedy descent scheme, the algorithm does not guarantee to always find the globally optimal segmentation for any given $(\boldsymbol{X}, \varepsilon)$.[7] From our experience, we found that the main factor affecting the global convergence of the algorithm seems to be the density of the samples relative to the distortion ε^2. In Section 5.4 we will give strong empirical evidences for the convergence of the algorithm over a wide range of ε.

Notice that the greedy merging process in Algorithm 5.1 is similar in spirit to classical agglomerative clustering methods, especially Ward's method [**?**]. However, whereas Ward's method assumes isotropic Gaussians, our coding-based approach is capable of segmenting Gaussians with arbitrary covariance, including nearly degenerate distributions. Classical agglomerative approaches have been shown to be inappropriate for such situations [?]. In this sense, the change in coding length provides a principled means of measuring similarity between arbitrary Gaussians. Our approach also demonstrates significant robustness to uniform outliers, another situation in which linkage algorithms [**?**] fail.

5.2 Lossy Coding of Multivariate Data

In this section, we give a more detailed justification of the coding rate/length functions introduced in the previous section. In the next section, we provide a more thorough analysis of the compression-based approach to data segmentation. Readers who are less concerned with technical details may skip these two sections at first read, without much loss of continuity.

If the given data $\boldsymbol{x}_i \in \mathbb{R}^D$ are i.i.d. samples of a random vector $\boldsymbol{x}$ with the probabilistic distribution $p(\boldsymbol{x})$, the optimal coding scheme and the optimal coding rate of such a random vector $\boldsymbol{x}$ have been well characterized in *information theory* (see [Cover and Thomas, 1991] and references therein). However, here we are dealing with a finite set of vectors $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$. Such a data set can be viewed as a non-parametric distribution itself – each vector $\boldsymbol{x}_i$ in $\boldsymbol{X}$ occurs with an equal probability $1/N$. The optimal coding scheme for the distribution $p(\boldsymbol{x})$ is no longer optimal for $\boldsymbol{X}$ and the formula for the coding length no longer accurate. Nevertheless, some of the basic ideas of deriving the optimal coding rate can still be extended to the non-parametric setting. In this section, borrowing ideas from information theory, we derive a tight bound of the coding length or rate for the given data $\boldsymbol{X}$. In Appendix 5.A, we give an alternative derivation of the bound. Although both approaches essentially arrive at the same estimate, they

[7] However, it may be possible to improve the convergence by using more complicated split-and-merge strategies [**?**]. In addition, due to Theorem 1 of Section 5.3, the globally (asymptotically) optimal segmentation can also be computed via concave optimization [**?**], at the cost of potentially exponential computation time.

together reveal that the derived coding length/rate function holds under different conditions:

1. The derivation in this section shows that for small ε the formula for $R(\boldsymbol{X})$ gives a good approximation to the (asymptotically) optimal rate-distortion function of a Gaussian source.

2. The derivation in Appendix 5.A shows that the same coding length/rate formula works for any finite set of vectors $\boldsymbol{X}$ that span a subspace.

5.2.1 *The Rate Distortion Function*

For simplicity, we here assume that the given data are zero mean, i.e. $\mu \doteq \frac{1}{N}\sum_i \boldsymbol{x}_i = 0$. The reader may refer to Appendix 5.B for the case in which the mean is not zero. Let ε^2 be the squared error allowable for encoding every vector $\boldsymbol{x}_i$. That is, if $\hat{\boldsymbol{x}}_i$ is an approximation of $\boldsymbol{x}_i$, we allow $\mathbb{E}[\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2] \leq \varepsilon^2$. In other words, on average, the allowable squared error for each entry of $\boldsymbol{x}_i$ is ε^2/D.

The solution to coding the vectors in $\boldsymbol{X}$, subject to the mean squared error ε^2, can be explained by *sphere packing*, which is normally adopted in information theory [Cover and Thomas, 1991]. Here we are allowed to perturb each vector $\boldsymbol{x}_i \in \boldsymbol{X}$ within a sphere of radius ε in $\mathbb{R}^D$. In other words, we are allowed to distort each entry of $\boldsymbol{x}_i$ with an (independent) random variable of variance ε^2/D. Without loss of generality, we may model the error as an independent additive Gaussian noise:

$$\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{z}_i, \quad \text{with} \quad \boldsymbol{z}_i \sim \mathcal{N}\Big(0, \frac{\varepsilon^2}{D}I\Big). \tag{5.6}$$

Then the covariance matrix of the vectors $\{\hat{\boldsymbol{x}}_i\}$ is:

$$\hat{\Sigma} \doteq \mathbb{E}\Big[\frac{1}{N}\sum_{i=1}^{N}\hat{\boldsymbol{x}}_i\hat{\boldsymbol{x}}_i^T\Big] = \frac{\varepsilon^2}{D}I + \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T \ \in \mathbb{R}^{D\times D}. \tag{5.7}$$

The volume of the region spanned by these vectors is proportional to (the square root of the determinant of the covariance matrix):

$$\text{vol}(\hat{\boldsymbol{X}}) \propto \sqrt{\det\Big(\frac{\varepsilon^2}{D}I + \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T\Big)}.$$

Similarly, the volume spanned by each random vector $\boldsymbol{z}_i$ is proportional to

$$\text{vol}(\boldsymbol{z}) \propto \sqrt{\det\Big(\frac{\varepsilon^2}{D}I\Big)}.$$

In order to encode each vector, we can partition the region spanned by all the vectors into non-overlapping spheres of radius ε. When the volume of the region $\text{vol}(\hat{\boldsymbol{X}})$ is significantly larger than the volume of the sphere, the total number of spheres that we can pack into the region is approximately equal to

$$\#\text{of spheres } = \text{vol}(\hat{\boldsymbol{X}})/\text{vol}(\boldsymbol{z}). \tag{5.8}$$

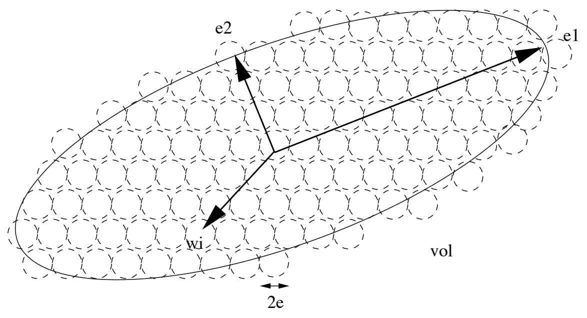


Figure 5.1. Coding of a set of vectors in a region in $\mathbb{R}^D$ with an accuracy up to ε^2. To know the vector $\boldsymbol{x}_i$, we only need to know the label of the corresponding sphere. $e_1, e_2 \in \mathbb{R}^D$ represent the singular vectors of the matrix $\hat{\boldsymbol{X}}$ and $\sigma_1, \sigma_2 \in \mathbb{R}$ the singular values.

Thus, to know each vector $\boldsymbol{x}_i$ with an accuracy up to ε^2, we only need to specify which sphere $\boldsymbol{x}_i$ is in (see Figure 5.1). If we use binary numbers to label all the spheres in the region of interest, the number of bits needed is

$$\begin{aligned} R(\boldsymbol{X}) &\doteq \log_2(\#\text{of spheres}) = \log_2\big(\text{vol}(\hat{\boldsymbol{X}})/\text{vol}(\boldsymbol{z})\big) \\ &= \frac{1}{2}\log_2 \det\big(I + \frac{D}{N\varepsilon^2}\boldsymbol{X}\boldsymbol{X}^T\big), \end{aligned} \tag{5.9}$$

where the last equality uses the fact $\det(A)/\det(B) = \det(B^{-1}A)$.

If the samples $\boldsymbol{x}_i$ are drawn from a Gaussian source $\mathcal{N}(0, \Sigma)$, then $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T$ converges to the covariance Σ of the Gaussian source. Thus, we have $R(\boldsymbol{X}) \to \frac{1}{2}\log_2\det\big(I + \frac{D}{\varepsilon^2}\Sigma\big)$ as $N \to \infty$. When $\frac{\varepsilon^2}{D} \leq \lambda_{min}(\Sigma)$, the optimal rate-distortion for a parallel i.i.d. $\mathcal{N}(0,\Sigma)$ source is $\frac{1}{2}\log_2\det\big(\frac{D}{\varepsilon^2}\Sigma\big)$, to which (5.9) provides a good approximation. In general, the optimal rate-distortion is a complicated formula given by reverse-waterfilling on the eigenvalues of Σ (see Theorem 13.3.3 of [Cover and Thomas, 1991]). The approximation (5.9) provides an upper bound which holds for all ε, and is tight when ε is small relative to the eigenvalues of the covariance.

The formula for $R(\boldsymbol{X})$ can also be viewed as the rate-distortion of the source $\boldsymbol{X}$ regularized by a noise of variance $\frac{\varepsilon^2}{D}$ as in equation (5.6). The covariance $\hat{\Sigma}$ of the perturbed vectors $\hat{\boldsymbol{x}}_i$ always satisfies $\frac{\varepsilon^2}{D} \leq \lambda_{min}(\hat{\Sigma})$, allowing for a simple, analytic expression for the rate distortion for all range of ε. This regularized rate-distortion has the further advantage of agreeing with the bound for the coding length of finitely many vectors that span a subspace, derived in Appendix 5.A.[8].

Notice that the formula for $R(\boldsymbol{X})$ is accurate only in the asymptotic sense, i.e., when we are dealing with a large number of samples and the error ε is small (relative to the magnitude of the data $\boldsymbol{X}$). We want to emphasize that the above derivation of the coding rate does not give an actual coding scheme. The con-

[8]In addition, this formula resembles the channel capacity of an MIMO Gaussian channel. The interested reader may refer to [**?**].

struction of efficient coding schemes which achieve the optimal rate-distortion bound is itself a difficult problem (see, for example, [**?**] and references therein). However, for the purpose of measuring the quality of segmentation and compression, all that matters is that *in principle* a scheme attaining the optimal rate $R(\boldsymbol{X})$ exists.

5.2.2 *The Coding Length Function*

Given the coding rate $R(\boldsymbol{X})$, the total number of bits needed to encode the N vectors in $\boldsymbol{X}$ is

$$N \cdot R(\boldsymbol{X}) = \frac{N}{2} \log_2 \det \big(I + \frac{D}{N\varepsilon^2} \boldsymbol{X}\boldsymbol{X}^T\big). \tag{5.10}$$

From the communication point of view, $N \cdot R(\boldsymbol{X})$ bits are already sufficient as both the transmitter and the receiver share the same code book – that is they both know the region spanned by $\boldsymbol{X}$ in $\mathbb{R}^D$. However, from the data representation or compression point of view, we need more bits to represent the code book itself. This is equivalent to specifying all the principal axes of the region spanned by the data, i.e. the singular values/vectors of $\boldsymbol{X}$, see Figure 5.1. As the number of principal axes is D, we need $D \cdot R(\boldsymbol{X})$ additional bits to encode them. Therefore, the total number of bits needed to encode the N vectors in $\boldsymbol{X} \subset \mathbb{R}^D$ subject to the squared error ε^2 is[9]

$$L(\boldsymbol{X}) \doteq (N + D)R(\boldsymbol{X}) = \frac{N + D}{2} \log_2 \det \big(I + \frac{D}{N\varepsilon^2} \boldsymbol{X}\boldsymbol{X}^T\big). \tag{5.11}$$

Appendix 5.A provides an alternative derivation of the same coding length function $L(\boldsymbol{X})$, as an upper bound for a finite number of samples. If the data $\boldsymbol{X}$ have a non-zero mean, we need more bits to encode the mean too. See in Appendix 5.B how the coding length function should be properly modified in that case.

5.2.3 *Properties of the Coding Length Function*

Commutative Property.

Since $\boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{D\times N}$ and $\boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}^{N\times N}$ have the same non-zero eigenvalues, the coding length function can also be expressed as:

$$\begin{aligned} L(\boldsymbol{X}) &= \frac{N + D}{2} \log_2 \det \big(I + \frac{D}{N\varepsilon^2} \boldsymbol{X}\boldsymbol{X}^T\big) \\ &= \frac{N + D}{2} \log_2 \det \big(I + \frac{D}{N\varepsilon^2} \boldsymbol{X}^T\boldsymbol{X}\big). \end{aligned}$$

[9]Compared to the MDL criterion (2.49), if the term $N \cdot R(\boldsymbol{X})$ corresponds to the coding length for the data, the term $D \cdot R(\boldsymbol{X})$ then corresponds to the coding length for the model parameter θ.

Thus, if $D \ll N$, the second expression will be less costly for computing the coding length. The matrix $\boldsymbol{X}^T\boldsymbol{X}$, which depends only on the inner products between pairs of data vectors, is known in the statistical learning literature as the *kernel matrix*. This property suggests that the ideas and the algorithm presented in Section 5.1 can be readily extended to segment data sets that have *nonlinear* structures, by choosing a proper kernel function.

Invariant Property.

Notice that in the zero-mean case, the coding length function $L(\boldsymbol{X})$ is invariant under an orthogonal transformation of the data $\boldsymbol{X}$. That is, for any orthogonal matrix $U \in O(D)$ or $V \in O(N)$, we have

$$L(U\boldsymbol{X}) = L(\boldsymbol{X}) = L(\boldsymbol{X}V). \tag{5.12}$$

In other words, the length function depends only on the singular values of $\boldsymbol{X}$ (or eigenvalues of $\boldsymbol{X}\boldsymbol{X}^T$). This equality suggests that one may choose any orthonormal basis (e.g., Fourier, wavelets) to represent and encode the data and the number of bits needed should always be the same. This agrees with the fact that the chosen coding length (or rate) is optimal for a Gaussian source. However, if the data are non-Gaussian or nonlinear, a proper transformation can still be useful for compressing the data.[10] Here we are essentially seeking a partition, rather than a transformation, of the non-Gaussian (or nonlinear) data set, such that each subset is sufficiently Gaussian (or subspace-like) and hence cannot be compressed any further, either by (orthogonal) transformation or segmentation.

5.3 Coding Length of Segmented Data

Now suppose we have partitioned the set of N vectors $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ into n non-overlapping groups $\boldsymbol{X} = \boldsymbol{X}_1 \cup \boldsymbol{X}_2 \cup \cdots \cup \boldsymbol{X}_n$. Then the total number of bits needed to encode the segmented data is $L^s(\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n) = \sum_{i=1}^n L(\boldsymbol{X}_i) + |\boldsymbol{X}_i|\big(-\log_2(|\boldsymbol{X}_i|/N)\big)$. Here the superscript "$s$" is used to indicate the coding length after segmentation.

5.3.1 *Segmentation and Compression*

To better understand under what conditions a set of data should or should not be segmented so that the overall coding length/rate becomes smaller, we here provide two representative examples. In the examples, we want to study whether a data set should be partitioned into two subsets of an equal number of vectors: $\boldsymbol{X}_1, \boldsymbol{X}_2 \in \mathbb{R}^{D\times N}$. To simplify the analysis, we assume $N \gg D$ so that we can ignore the asymptotically insignificant terms in the coding length/rate function.

[10] For a more thorough discussion on why some transformations (such as wavelets) are useful for data compression, the reader may refer to [**?**].

Example 5.1 [Uncorrelated Subsets] Notice that in general, we have

$$\begin{aligned} L(\boldsymbol{X}_1) + L(\boldsymbol{X}_2) &= \frac{N}{2}\log_2 \det\big(I + \frac{D}{N\varepsilon^2}\boldsymbol{X}_1\boldsymbol{X}_1^T\big) + \frac{N}{2}\log_2 \det\big(I + \frac{D}{N\varepsilon^2}\boldsymbol{X}_2\boldsymbol{X}_2^T\big) \\ &\leq \frac{2N}{2}\log_2 \det\big(I + \frac{D}{2N\varepsilon^2}(\boldsymbol{X}_1\boldsymbol{X}_1^T + \boldsymbol{X}_2\boldsymbol{X}_2^T)\big) = L(\boldsymbol{X}_1 \cup \boldsymbol{X}_2), \end{aligned}$$

where the inequality is from the concavity of the function $\log_2 \det(\cdot)$ (see Theorem 7.6.7 of [**?**]). Thus, if the difference $L(\boldsymbol{X}_1 \cup \boldsymbol{X}_2) - \big(L(\boldsymbol{X}_1) + L(\boldsymbol{X}_2)\big)$ is large, the overhead needed to encode the membership of the segmented data (here one bit per vector) becomes insignificant. If we further assume that $\boldsymbol{X}_2$ is a rotated version of $\boldsymbol{X}_1$, i.e. $\boldsymbol{X}_2 = U\boldsymbol{X}_1$ for some $U \in O(D)$, one can show that the difference $L(\boldsymbol{X}_1 \cup \boldsymbol{X}_2) - \big(L(\boldsymbol{X}_1) + L(\boldsymbol{X}_2)\big)$ is (approximately) maximized when $\boldsymbol{X}_2$ becomes orthogonal to $\boldsymbol{X}_1$. We call two groups $\boldsymbol{X}_1, \boldsymbol{X}_2$ *uncorrelated* if $\boldsymbol{X}_1^T\boldsymbol{X}_2 = 0$. Thus, segmenting the data into uncorrelated groups typically reduces the overall coding length. From the viewpoint of sphere packing, Figure 5.2 explains the reason. ■

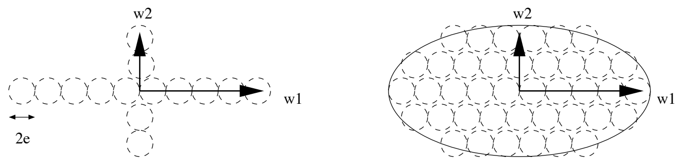


Figure 5.2. The number of spheres (code words) of two different schemes for coding two orthogonal vectors. Left: encoding the two vectors separately; Right: encoding the two vectors together.

Example 5.2 [Strongly Correlated Subsets] We say two groups $\boldsymbol{X}_1, \boldsymbol{X}_2$ are strongly correlated if they span the same subspace in $\mathbb{R}^D$. Or somewhat equivalently, we may assume that $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ have approximately the same covariance $\boldsymbol{X}_2\boldsymbol{X}_2^T \approx \boldsymbol{X}_1\boldsymbol{X}_1^T$. Thus we have

$$\begin{aligned} L(\boldsymbol{X}_1) + L(\boldsymbol{X}_2) &= \frac{N}{2}\log_2 \det\big(I + \frac{D}{N\varepsilon^2}\boldsymbol{X}_1\boldsymbol{X}_1^T\big) + \frac{N}{2}\log_2 \det\big(I + \frac{D}{N\varepsilon^2}\boldsymbol{X}_2\boldsymbol{X}_2^T\big) \\ &\approx \frac{2N}{2}\log_2 \det\big(I + \frac{D}{2N\varepsilon^2}(\boldsymbol{X}_1\boldsymbol{X}_1^T + \boldsymbol{X}_2\boldsymbol{X}_2^T)\big) = L(\boldsymbol{X}_1 \cup \boldsymbol{X}_2). \end{aligned}$$

Since $L^s(\boldsymbol{X}_1, \boldsymbol{X}_2) = L(\boldsymbol{X}_1) + L(\boldsymbol{X}_2) + H(|\boldsymbol{X}_1|, |\boldsymbol{X}_2|)$, the overhead needed to encode the membership becomes significant and the segmented data require more bits than the unsegmented. ■

5.3.2 *Optimality of Deterministic Segmentation*

So far, we have only considered partitioning the data $\boldsymbol{X}$ into n non-overlapping groups. That is, each vector is assigned to a group with probability either 0 or 1. We call such a segmentation "deterministic." In this section, we examine an important question: *Is there a probabilistic segmentation of the data that can achieve an even lower coding rate?* That is, we consider a more general class of segmentations in which we assign each vector $\boldsymbol{x}_i$ to the group j according to a probability $\pi_{ij} \in [0, 1]$, with $\sum_{j=1}^{n} \pi_{ij} = 1$ for all $i = 1, 2, \ldots, N$.

To facilitate counting the expected coding length of such (probabilistically) segmented data, we introduce a matrix Π_j that collects the membership of the N vectors in group j:

$$\Pi_j \doteq \begin{bmatrix} \pi_{1j} & 0 & \cdots & 0 \\ 0 & \pi_{2j} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \pi_{Nj} \end{bmatrix} \in \mathbb{R}^{N \times N}. \tag{5.13}$$

These matrices satisfy the constraint: $\sum_{j=1}^{n} \Pi_j = I_{N\times N}, \Pi_j \succeq 0$.

Obviously, the jth group has an expected number of $\textbf{tr}(\Pi_j)$ vectors and the expected covariance is $\frac{1}{\textbf{tr}(\Pi_j)}\boldsymbol{X}\Pi_j\boldsymbol{X}^T$. If viewed as a Gaussian source, the coding rate of the jth group is bounded by: $R(\boldsymbol{X}_j) \doteq \frac{1}{2}\log_2\det\big(I + \frac{D}{\textbf{tr}(\Pi_j)\varepsilon^2}\boldsymbol{X}\Pi_j\boldsymbol{X}^T\big)$. If for each vector $\boldsymbol{x}_i$, we code it using the coding scheme for the jth group with probability π_{ij}, then the expected total number of bits required to encode the data W according to the segmentation $\Pi = \{\Pi_j\}$ is bounded by[11]

$$\begin{aligned} L^s(\boldsymbol{X}, \Pi) \quad &\doteq \quad \sum_{j=1}^{n} \frac{\textbf{tr}(\Pi_j) + D}{2} \log_2 \det\big(I + \frac{D}{\textbf{tr}(\Pi_j)\varepsilon^2}\boldsymbol{X}\Pi_j\boldsymbol{X}^T\big) \\ &+ \quad \textbf{tr}(\Pi_j)\Big(-\log_2 \frac{\textbf{tr}(\Pi_j)}{N}\Big). \end{aligned} \tag{5.14}$$

Similarly, the expected number of bits needed to encode each vector is bounded by

$$\begin{aligned} R^s(\boldsymbol{X}, \Pi) \quad &\doteq \quad \frac{1}{N} L^s(\boldsymbol{X}, \Pi) \\ &= \quad \sum_{j=1}^{n} \frac{\textbf{tr}(\Pi_j)}{N}\Big(R(\boldsymbol{X}_j) - \log_2 \frac{\textbf{tr}(\Pi_j)}{N}\Big) + \frac{D}{N}R(\boldsymbol{X}_j). \end{aligned} \tag{5.15}$$

Thus, one may consider that the optimal segmentation Π^* is the global minimum of the expected overall coding length $L^s(\boldsymbol{X}, \Pi)$, or equivalently the average coding rate $R^s(\boldsymbol{X}, \Pi)$. To some extent, one can view the minimum value of $R^s(\boldsymbol{X}, \Pi)$ as a good approximation to the actual entropy of the given data set $\boldsymbol{X}$.[12]

Notice that the second term in the expression of $R^s(\boldsymbol{X}, \Pi)$, $\frac{D}{N}R(\boldsymbol{X}_j)$, is insignificant when the number of samples is large $N \gg D$. Nevertheless, this term, as well as the term that encodes the membership of the vectors, gives a *tight* bound

[11]Strictly speaking, the formula is an upper bound for the expected coding length because $L^s(\boldsymbol{X}, \Pi)$ is essentially a concave function of the group assignment Π (see the proof of Theorem 5.3). Hence, $L^s(\boldsymbol{X}, \mathbb{E}[\Pi]) \geq \mathbb{E}[L^s(\boldsymbol{X}, \Pi)]$ (using that $f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$ for concave functions).

[12]Especially when the data $\boldsymbol{X}$ indeed consist of a mixture of subsets and each group is a typical set of samples from a (almost degenerate) Gaussian distribution.

on the coding length even for small sets of samples. This essentially allows us to find the optimal segmentation in a bottom-up manner by merging small subsets of samples, which is effectively harnessed by the greedy algorithm introduced in Section 5.1. That said, for the rest of this section, we examine more carefully the asymptotic properties of the coding length/rate function.

The first term in the expression of $R^s(\boldsymbol{X}, \Pi)$ is the only part that matters asymptotically (i.e. when the number of vectors in each group goes to infinity) and we denote it as:

$$\begin{aligned} R^{s,\infty}(\boldsymbol{X}, \Pi) \quad &\doteq \quad \sum_{j=1}^{n} \frac{\mathbf{tr}(\Pi_j)}{2N} \log_2 \det \big(I + \frac{D}{\varepsilon^2\, \mathbf{tr}(\Pi_j)} \boldsymbol{X} \Pi_j \boldsymbol{X}^T\big) \\ &- \quad \frac{\mathbf{tr}(\Pi_j)}{N} \log_2 \Big(\frac{\mathbf{tr}(\Pi_j)}{N}\Big). \end{aligned} \tag{5.16}$$

Thus, the global minimum of $R^{s,\infty}(\boldsymbol{X}, \Pi)$ determines the optimal segmentation when the sample size is large.

Theorem 5.3. *The asymptotic part $R^{s,\infty}(\boldsymbol{X}, \Pi)$ of the rate distortion function $R^s(\boldsymbol{X}, \Pi)$ is a concave function of Π in the convex domain $\Omega \doteq \{\Pi : \sum_{j=1}^{n} \Pi_j = I, \Pi_j \succeq 0\}$.*

Proof. Let $\mathcal{S}$ be the set of all $N \times N$ non-negative definite symmetric matrices. We will show that $R^{s,\infty}(\boldsymbol{X}, \Pi)$ is concave as a function from $\mathcal{S}^n \to \mathbb{R}$, and so is it when restricted to the domain of interest, $\Omega \subset \mathcal{S}^n$.

First consider the second term of $R^{s,\infty}(\boldsymbol{X}, \Pi)$. Notice that $\sum_{j=1}^{n} \mathbf{tr}(\Pi_j) = N$ is a constant. So we only need to show the concavity of the function $g(P) \doteq -\,\mathbf{tr}(P) \log_2 \mathbf{tr}(P)$ for $P \in \mathcal{S}$. The function, $f(x) = -x \log_2 x$ is concave, and $g(P) = f(\mathbf{tr}(P))$. So for $\lambda \in [0, 1]$,

$$\begin{aligned} g(\lambda P_1 + (1-\lambda) P_2) &= f(\lambda\, \mathbf{tr}(P_1) + (1-\lambda)\, \mathbf{tr}(P_2)) \\ \geq \quad \lambda f(\mathbf{tr}(P_1)) &+ (1-\lambda) f(\mathbf{tr}(P_2)) = \lambda g(P_1) + (1-\lambda) g(P_2). \end{aligned}$$

Thus, $g(P)$ is concave in P.

Now consider the first term of $R^{s,\infty}(\boldsymbol{X}, \Pi)$. Let

$$h(\Pi_j) \doteq \mathbf{tr}(\Pi_j) \log_2 \det \big(I + \frac{D}{\varepsilon^2\, \mathbf{tr}(\Pi_j)} \boldsymbol{X} \Pi_j \boldsymbol{X}^T\big).$$

It is well-known in information theory that the function $q(P) \doteq \log_2 \det(P)$ is concave for $P \in \mathcal{S}$ and $P \succ 0$ (see Theorem 7.6.7 of [?]). Now define $r : \mathcal{S} \to \mathbb{R}$ to be

$$r(\Pi_j) \doteq \log_2 \det(I + \alpha \boldsymbol{X} \Pi_j \boldsymbol{X}^T) = q(I + \alpha \boldsymbol{X} \Pi_j \boldsymbol{X}^T).$$

Since r is just the concave function q composed with an affine transformation $\Pi_j \mapsto I + \alpha \boldsymbol{X} \Pi_j \boldsymbol{X}^T$, r is concave (see Section 3.2.3 of [**?**]). Let $\psi : \mathcal{S} \times \mathbb{R}_+ \to \mathbb{R}$ as

$$\psi(\Pi_j, t) \doteq t \cdot \log_2 \det \big(I + \frac{N}{\varepsilon^2 t} \boldsymbol{X} \Pi_j \boldsymbol{X}^T\big) = t \cdot r\big(\frac{1}{t} \Pi_j\big).$$

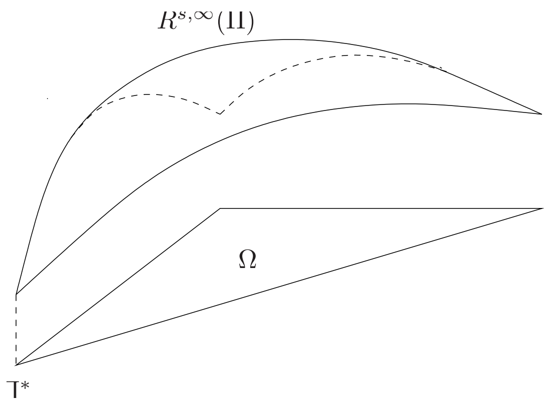


Figure 5.3. The function $R^{s,\infty}(\boldsymbol{X},\Pi)$ is a concave function of Π over a convex domain Ω, which is in fact a polytope in the space $\mathbb{R}^{nN}$. The minimal coding length is achieved at a vertex Π^* of the polytope.

According to Theorem 3.2.6 of [?], ψ is concave. Notice that $H \doteq \{(\Pi_j,t) : t = \mathbf{tr}(\Pi_j)\}$ is a linear subspace in the product space of $\mathbb{R}$ and the space of all symmetric matrices. So, $H \cap (\mathcal{S} \times \mathbb{R}_+)$ is a convex set, and the desired function, $h(\Pi_j) = \psi(\Pi_j, \mathbf{tr}(\Pi_j))$, is just the restriction of ψ to this convex set. Thus, h is concave.

Since $R^{s,\infty}(\boldsymbol{X},\Pi)$ is a sum of concave functions in Π_j, it is concave as a function from $\mathcal{S}^n$ to $\mathbb{R}$, and so is its restriction to the convex set Ω in $\mathcal{S}^n$. □

Since $R^{s,\infty}(\boldsymbol{X},\Pi)$ is concave, its global minimum Π^* is always reached at the boundary, or more precisely, at a vertex of the convex domain Ω, as shown in Figure 5.3. At the vertex of Ω, the entries π_{ij} of Π^* are either 0s or 1s. It means that even if we allow soft assignment of each point to the n groups according to any probabilistic distribution, the optimal solution with the minimal coding length can always be approximately achieved by assigning each point to one of the groups with probability one! This is the reason why Algorithm 5.1 does not consider any probabilistic segmentation.

Another implication of the above theorem is that the problem of minimizing the coding length is essentially a concave optimization problem. Many effective concave optimization algorithms can be adopted to find the globally optimal segmentation, such as the simplex algorithm [?]. However, such generic concave optimization algorithms typically have high (potentially exponential) complexity. In the next section, we will show with extensive simulations and experiments that the greedy algorithm proposed in Section 5.1 is already effective in minimizing the coding length.

Interestingly, in multiple-channel communications, the goal is instead to *maximize* the channel capacity, which has very much the same formula as the coding rate function [?]. The above theorem suggests that a higher channel capacity may be achieved inside the convex domain Ω, i.e. by probabilistically assigning the transmitters into certain number of groups. As the coding rate function is concave, the maximal channel capacity can be very easily computed via convex optimization [?].

5.4 Simulation and Experimental Results

In this section, we conduct simulations on a variety of challenging data sets to examine the effectiveness of the proposed coding length function as well the performance of the steepest descent algorithm. In the end, we will also demonstrate some experimental results of applying the algorithm to segment imagery and bioinformatic data.

5.4.1 Simulations on Synthetic Data

Segmentation of Linear Subspaces of Different Dimensions.

We first demonstrate the ability of the algorithm to segment noisy samples drawn from a mixture of linear subspaces of different dimensions. For every d-dimensional subspace, $d \times 100$ samples are drawn uniformly from a ball of diameter 1 lying on the subspace. Each sample is corrupted with independent Gaussian noise of standard deviation $\varepsilon_0 = 0.04$. For Algorithm 5.1, we set $\varepsilon = \varepsilon_0$. We compare the results of Algorithm 5.1 with the expectation maximization (EM) algorithm for mixture of factor analyzers [**?**], followed by an ML classification step. We have modified [?] slightly to allow it to work for mixture of factor analyzers with different dimensions. To avoid the model selection issue, which we postpone to subsection *6)*, we provided the EM algorithm with the correct number and dimensions of the subspaces. Figure 5.4 shows one representative result of Algorithm 5.1. Table 5.1 summarizes the comparison of results on several configurations tested.

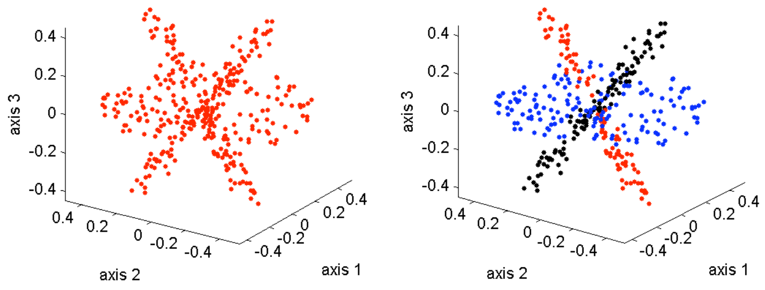


Figure 5.4. The computed segmentation for $(2, 1, 1)$ in $\mathbb{R}^3$ is displayed. Left: Input data with noise; Right: Output segmented data.

Subspace dimensions	Identified dimensions	Classification (%) (Algorithm 5.1)	Classification (%) (EM)
$(2,1,1)$ in $\mathbb{R}^3$	$2,1,1$	96.62	39.33
$(2,2,1)$ in $\mathbb{R}^3$	$2,2,1$	90.00	68.98
$(4,2,2,1)$ in $\mathbb{R}^5$	$4,2,2,1$	98.53	43.36
$(6,3,1)$ in $\mathbb{R}^7$	$6,3,1$	99.77	66.16
$(7,5,2,1,1)$ in $\mathbb{R}^8$	$7,5,2,1,1$	98.04	42.29

Table 5.1. Simulation results for data drawn from mixtures of noisy linear subspaces. Classification percentages are averaged over 25 trials. Our algorithm correctly identifies the number and dimension of the subspaces in all 25 trials, for all configurations. Far right column: results using EM for mixture of factor analyzers with different dimensions [?] with random initialization.

In each case, the algorithm stops at the correct number of groups, and the dimensions of the segments $\boldsymbol{X}_i$ match those of the generating subspaces.[13] The correctness of the segmentation is further corroborated by the high percentage of points correctly classified (by comparing the segments with the *a priori* groups). For all five configurations, the average percentage of samples assigned to the correct group was at least 90.0%. The main cause of classification error is points which lie near the intersection of multiple subspaces. Due to noise, it may actually be more efficient to code such points according to the optimal coding scheme for one of the other subspaces. In all cases, Algorithm 5.1 dramatically outperforms EM (for mixture of factor analyzers), despite requiring no knowledge of the subspace dimensions.

Subspace dimensions	$(2,1,1)$ in $\mathbb{R}^3$	$(2,2,1)$ in $\mathbb{R}^3$	$(4,2,2,1)$ in $\mathbb{R}^5$	$(6,3,1)$ in $\mathbb{R}^7$	$(7,5,2,1,1)$ in $\mathbb{R}^8$
$\log_{10}\frac{\varepsilon_{max}}{\varepsilon_{min}}$	2.5	1.75	2.0	2.0	.75

Table 5.2. The size of the range of $\log\varepsilon$ for which the Algorithm 5.1 converges to the correct number and dimension of groups, for each of the arrangements considered in Figure 5.4.

Since in practice, ε_0 is not known, it is important to investigate the sensitivity of the results to the choice of ε. for each of the examples in Figure 5.4, Table 5.2 gives the range of ε for which Algorithm 5.1 converges to the a-priori number and dimension of subspaces. Notice that for each of the configurations considered, there exists a significant range of ε for which the greedy algorithm converges.

Global Convergence.

Empirically, we find that Algorithm 5.1 does not suffer many of the difficulties with local minima that plague iterative clustering algorithms (e.g. K-means) and

[13]The dimension of each segment $\boldsymbol{X}_i$ is identified using principal component analysis (PCA) by thresholding the singular values of $\boldsymbol{X}_i$ with respect to ε.

parameter estimation algorithms (e.g. EM). The convergence appears to depend mostly on the density of the samples relative to the distortion ε. For example, if the number of samples is fixed at $N = 1200$, and the data are drawn from three $\lceil \frac{D}{2} \rceil$-dimensional subspaces in $\mathbb{R}^D$, the algorithm converges to the correct solution for $D = 2$ upto $D = 56$. Here, we choose $\varepsilon = \varepsilon_0 = 0.008$. Beyond $D = 56$, the algorithm fails to converge to the three *a priori* subspaces as the samples have become too sparse. For $D > 56$, the computed segmentation gives a higher coding length than the *a priori* segmentation.

The same observation occurs for subspaces with different dimensions. For example, we randomly draw 800 noisy $(\varepsilon_0 = 0.14)$ samples from four subspaces of dimension 20, 15, 15, 10 in $\mathbb{R}^{40}$. The results of the greedy algorithm at different distortion ε are shown in Figure 5.5. As we see from the results, when the distortion ε is very small, the greedy algorithm does not necessarily converge to the optimal coding length. Nevertheless, the number of groups, 4, is still identified correctly by the algorithm when ε becomes relatively large.

As described in Section 5.1, ε can potentially be chosen automatically by minimizing $L^s + ND \cdot \log \varepsilon$, where the second term approximates (upto a constant) the number of bits needed to code the residual. The green curve in Figure 5.5 shows the value of this penalized coding length. Notice that its minimum falls very near the true $\log \varepsilon$. We observe similar results for other simulated examples: the penalty term is generally effective in selecting a relevant ε.

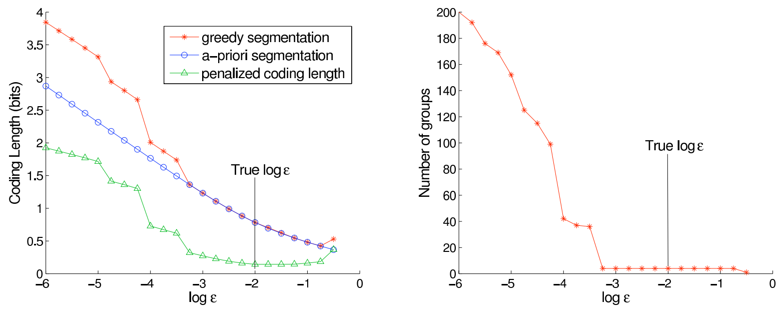


Figure 5.5. Left: the coding length found by the greedy algorithm (the red curve) compared to the ground truth (the blue curve) for data drawn from four linear subspaces of dimension 20, 15, 15, 10 in $\mathbb{R}^{40}$. The green curve shows the penalized coding length $L^s + ND \cdot \log \varepsilon$. Right: the number of groups found by the greedy algorithm – it converges to the correct number 4 when the distortion is relatively large.

Robustness to Outliers.

We test the robustness of Algorithm 5.1 to outliers on the easily visualized example of two lines and a plane in $\mathbb{R}^3$. $N = 158$ samples are drawn uniformly from a 2-D disc of diameter 1. 100 samples are drawn uniformly from each of the two line segments of length 1. The additive noise level is $\varepsilon_0 = 0.03$. The data set is

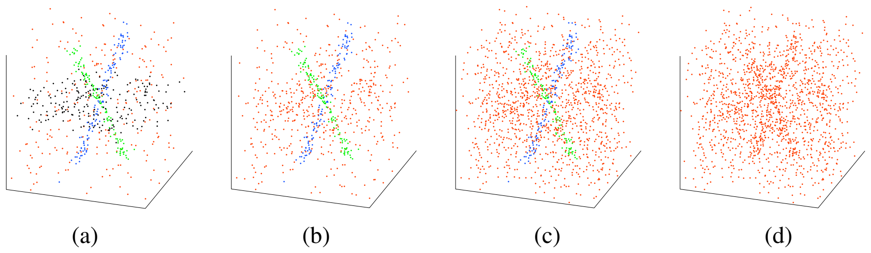


Figure 5.6. Segmentation results for data drawn from three linear subspaces, corrupted with various numbers of outliers, N_o. (a) $N_o = 300$ (45.6% outliers). (b) $N_o = 400$ (52.8% outliers). (c) $N_o = 1100$ (75.4% outliers). (d) $N_o = 1200$ (77.0% outliers).

contaminated with N_o outliers, whose three coordinates are uniformly distributed on $[-0.5, 0.5]$.

As the number of outliers increases, the segmentation exhibits several distinct phases. For $N_o \leq 300$ (45.6% outliers), the algorithm always finds the correct segmentation. The outliers are merged into a single (three-dimensional) group. From $N_o = 400$ (52.8% outliers) upto $N_o = 1100$ (75.4% outliers), the two lines are correctly identified, but samples on the plane are merged with the outliers. For $N_o = 1200$ (77.4% outliers) and higher, all of the data samples are merged into one group, as the distribution of data has become essentially random in the ambient space. Figure 5.6 shows the results for $N_o = 300, 400, 1100, 1200$. Notice that the effect of adding the outliers resembles the effect of ice (the lines and the plane) being melted away by warm water. This suggests a similarity between the artificial process of data clustering and the physical process of phase transition.

Number of Segments versus Distortion Level.

Figure 5.7 shows how the number of segments changes as ε varies. $N = 358$ points are drawn from two lines and a plane, as in the previous experiment, and then perturbed with noise of standard deviation $\varepsilon_0 = 0.05$. Notice that the number of groups experiences distinct phases, with abrupt transitions around several critical values of ε. For sufficiently small ε, each data point is grouped by itself. However, as ε increases, the cost of coding the group membership begins to dominate, and all the points are grouped together in a single three-dimensional subspace (the ambient space). Around the true noise level, ε_0, there is another stable phase, corresponding to the three *a priori* subspaces. Finally, as ε becomes large, the number of segments reverts to 1, as it becomes most efficient to represent the points using a single zero-dimensional subspace (the origin).

This behavior contrasts with the phase transition discussed in [**?**]. There, the number of segments increases monotonically throughout the simulated annealing process. Because our formulation allows the dimension of the segments to vary, the number of segments does not decrease monotonically with ε. Notice, however, that the phase corresponding to the "correct" (*a priori*) segmentation is stable over several orders of magnitude of the parameter ε. This is important since in practice the true noise level ε_0 is usually unknown.

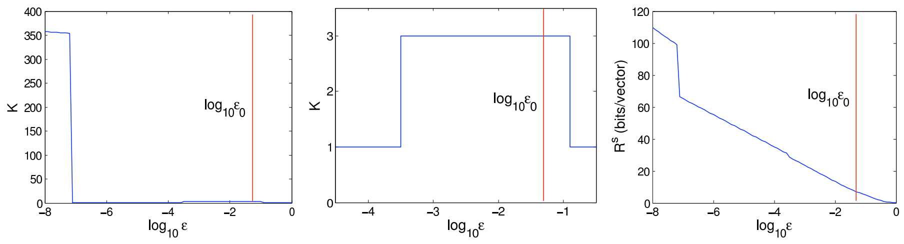


Figure 5.7. The effect of varying ε, with $\varepsilon_0 = 0.05$. Left: number of groups, n, versus $\log(\varepsilon)$. Center: detail of n versus $\log(\varepsilon)$ around $\log(\varepsilon_0)$. Right: the coding rate (bits per vector) versus $\log(\varepsilon)$.

Another interesting thing to notice is that the coding rate $R^s(\boldsymbol{X})$ in many regions is mostly a linear function of $-\log_{10}\varepsilon$: $R^s(\boldsymbol{X}) \approx -\beta \log_{10}\varepsilon + \alpha$, for some constants $\alpha, \beta > 0$, which is a typical characteristic of the rate-distortion function of Gaussians. For this data set, the algorithm takes about 10 seconds to run in Matlab on a 1.6GHz PC.

Segmentation of Affine Subspaces.

Appendix 5.B shows how the coding length function should be properly modified in the case when the data are not zero-mean. Here, we show how the modified algorithm works for affine subspaces. $N = 358$ samples are drawn from three linear subspaces in $\mathbb{R}^3$ and their centers are translated to $[2.1, 2.2, 2]^T$, $[2.4, 1.9, 2.1]^T$, $[1.9, 2.5, 1.9]^T$.

Figure 5.8 shows the segmentation results at different noise level, with the distortion level chosen as $\varepsilon = \varepsilon_0$. For $10^{-7} < \varepsilon < 0.1$, the algorithm always identifies the correct number of subspaces with $\varepsilon = \varepsilon_0$. When $\varepsilon \leq 10^{-7}$, the density of the samples within the subspace becomes more important than the distortion orthogonal to the subspace, and the algorithm no longer converges with $\varepsilon = \varepsilon_0$. However, for such small distortion, there always exists a large stable phase (with respect to changing ε) corresponding to the correct number of subspaces, $n = 3$. When $\varepsilon_0 > 0.1$, the algorithm starts to fail and merge the data samples into one or two groups.

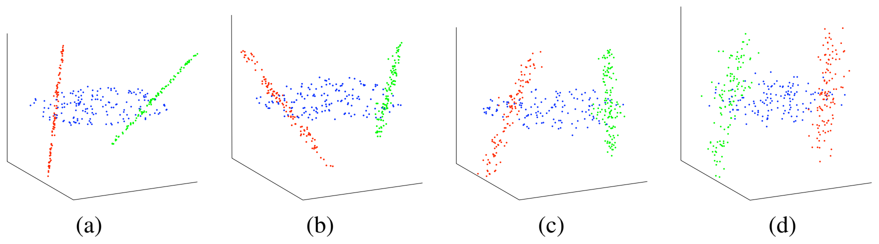


Figure 5.8. The segmentation results for data drawn from 3 affine subspaces at different noise level ε_0. The ε in the algorithm is chosen to be $\varepsilon = \varepsilon_0$. (a) $\varepsilon_0 = 0.01$, (b) $\varepsilon_0 = 0.03$, (c) $\varepsilon_0 = 0.05$, (d) $\varepsilon_0 = 0.08$.

We now fix the Gaussian noise at $\varepsilon_0 = 0.02$, and add N_o outliers whose three coordinates are uniformly distributed in the range of $[1.5, 2.5]$, which is the same as the range of the inliers. When the number of outliers is $\leq N_o = 200$ (35.8% outliers), the algorithm finds the correct segmentation, and all the outlying samples are segmented into one group. From $N_o = 300$ (45.6% outliers) to $N_o = 700$ (66.2% outliers), the algorithm still identifies the two lines and one plane. However, the outliers above and below the plane are clustered into two separate groups. For more than $N_o = 800$ (69.1% outliers), the algorithm identifies the two lines, but samples from the plane are merged with the outliers into one group. Figure 5.9 shows the segmentation results for $N_o = 200, 300, 700, 800$, respectively.

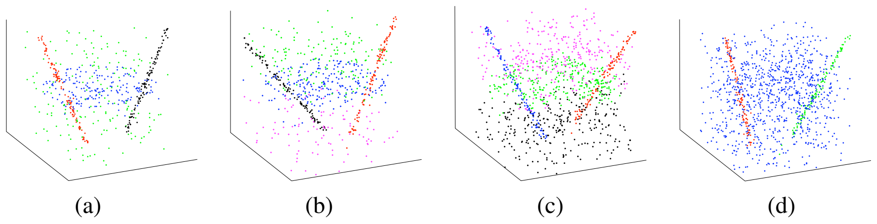


Figure 5.9. The segmentation results for data drawn from 3 affine subspaces with different number of outliers N_o. The ε in the algorithm is $\varepsilon = \varepsilon_0 = 0.02$. (a) $N_o = 200$ (35.8% outliers), (b) $N_o = 300$ (45.6% outliers), (c) $N_o = 700$ (66.2% outliers), (d) $N_o = 800$ (69.1% outliers).

Model Selection for Affine Subspaces and Nonzero-Mean Gaussians.

We compare the performance of Algorithm 5.1 to that of [**?**] and [**?**] on mixed data drawn from affine subspaces and non-zero mean Gaussians. We test the algorithms' performance over multiple trials for three different types of data distribution. The first is three affine subspaces: two lines and one plane, with noise standard deviation $\varepsilon_0 = 0.01$ and no outliers. Samples are drawn as in the previous examples. The means of the three groups are fixed (as in the previous examples), but the orientations of the two lines are chosen randomly. The second distribution tested is three affine subspaces: two planes and one line, with 158 points drawn from each plane and 100 from the line, again with $\varepsilon_0 = 0.01$. The orientations of one plane and of the line are chosen randomly. The final distribution tested is a mixture of $n = 3$ full-rank Gaussians in $\mathbb{R}^2$, with means $[2, 0]$, $[0, 0]$, $[0, 2]$ and covariance diag$(2, 0.2)$ (this is Figure 3 of [?]). $N = 900$ points are sampled (with equal probability) from the three Gaussians.

For the two subspace examples, we run Algorithm 5.1 with $\varepsilon = \varepsilon_0 = 0.01$. For the third example, we set $\varepsilon = 0.2$. We repeat each trial 50 times. Figure 5.10 shows a histogram of the number of groups arrived at by the three algorithms. For all algorithms, all of the segmentations with $n = 3$ are essentially correct (classification error $< 4\%$). However, for degenerate, or subspace-like data (Figure 5.10(a) and Figure 5.10(b)), Algorithm 5.1 was the most likely to converge to the a-priori group number. For full-rank Gaussians (Figure 5.10(c)), Algorithm

5.1 performs quite well, but is outperformed by [?], which finds the correct segmentation in all 50 trials. The failures of Algorithm 5.1 occur because the greedy descent converges to a local minimum of the coding length, rather than the global minimum.

Please note that [?] was not explicitly designed for degenerate distributions, whereas [?] was not designed for full-rank distributions. Also note that the samples in this experiment were drawn from a uniform distribution. The performance of all three algorithms improves when the generating distribution is indeed Gaussian. The main implication of the comparison, is therefore that Algorithm 5.1 succeeds under a wide range of conditions, and requires one to make less assumptions on the underlying data distribution.

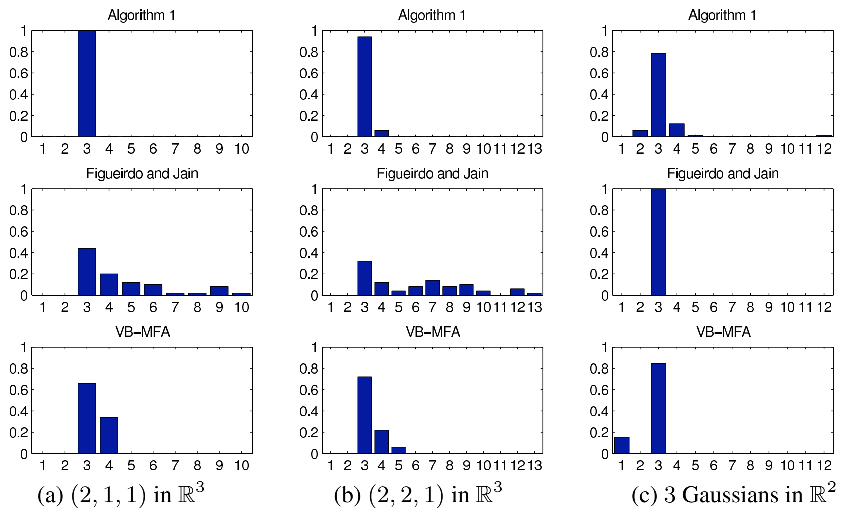


Figure 5.10. Frequency of occurrence for various n in 50 trials. Top row: Algorithm 5.1. Middle row: [?]. Bottom row: [?]. The left and center columns show results for randomly generated arrangements of affine subspaces. The right column shows results for datasets generated from three full-rank Gaussians, as in [?]. For all cases, the correct number of groups is $n = 3$.

5.4.2 *Experiments on Real Data*

As an example for practical applications, in this section, we test the proposed clustering algorithm on some real data such as images and microarray data. The goal is to demonstrate that this algorithm is capable of finding visually appealing structures in real data. However, we emphasize that it does not provide a complete solution to either of these practical problems. Such a solution usually entails a significant amount of domain-specific knowledge and engineering. Nevertheless, from these preliminary results with images and microarray data, we believe that the method presented in this chapter provides a generic solution for segmenting

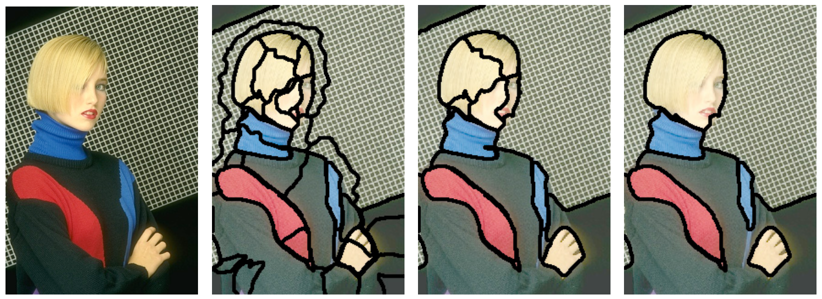


Figure 5.11. Hierarchical segmentation. Left: original image; Middle Left: $\varepsilon = 0.005$; Middle Right: $\varepsilon = 0.02$; Right: $\varepsilon = 0.05$.

mixed data that is simple and effective enough to be easily customized for a broad range of practical problems.[14]

Segmentation of Natural Images

The lossy compression based clustering Algorithm 5.1 has shed some new light on image segmentation, where degeneracy is typically introduced from using a common feature representation for different textures. In Chapter **??**, we will have detailed discussions on how to apply the algorithm to segmenting natural images. As we will see, the new clustering algorithm obtains good (unsupervised) image segmentation results even using features as simple as fixed-size Gaussian windows. Figure 5.11 shows the results of an image segmented using 5×5 windows under different levels of distortion ε. Readers who are interested in the subject of image segmentation please see Chapter **??** for more details and experimental results.

Clustering of Microarray Data

Figure 5.12 shows the result of applying Algorithm 5.1 to gene expression data. The dataset[15] consists of 13,872 vectors in $\mathbb{R}^{19}$, each of which describes the expression level of a single gene at different time points during an experiment on anthrax sporulation. A random subset of 600 vectors is visualized in figure 5.12(a). Here, rows correspond to genes and columns to time points. We cluster these vectors without any preprocessing, using Algorithm 5.1 with $\varepsilon = 1$. The algorithm finds three distinct clusters, which are displayed in figure 5.12(b) by reordering the rows.

Figure 5.13 shows clustering results on two additional gene expression datasets[16]. The first consists of 8,448 vectors in $\mathbb{R}^5$, describing the expression

[14] We have also tested our algorithm on other mixed data such as speech and handwritten digits. The results are equally encouraging.

[15] GDS930, available at http://www.ncbi.nlm.nih.gov/projects/geo.

[16] GDS34 (left) and GDS1316 (right), also available at http://www.ncbi.nlm.nih.gov/projects/geo.

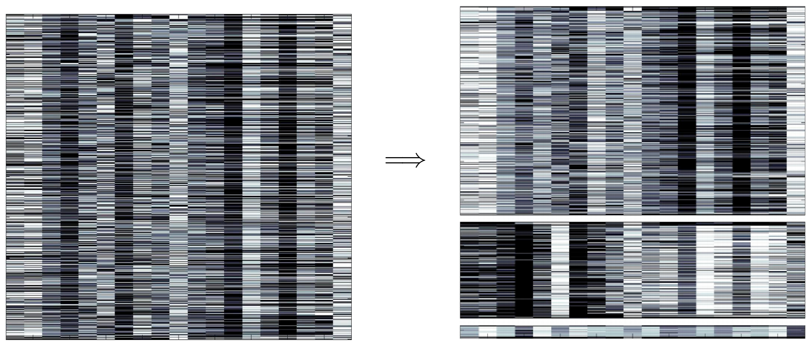


Figure 5.12. Segmentation of microarray data. Left: raw data. Each row represents the expression level of a single gene. Right: Three distinct clusters are found, visualized by reordering the rows.

levels of yeast genes at 5 different time points during a heat shock experiment. Figure 5.13(a) shows expression levels for a randomly selected subset of 1,200 genes. We cluster these vectors using Algorithm 5.1, with $\varepsilon = 0.1$. Our algorithm discovers a number of visually coherent clusters, shown in Figure 5.13(b). The second dataset consists of 45,101 vectors in $\mathbb{R}^{10}$, each of which corresponds to the expression level of a single gene under varying experimental conditions (this experiment investigated Down Syndrome-related leukemias). We run Algorithm 5.1 with $\varepsilon = 1$ on a subset of 800 of these vectors (shown in Figure 5.13). Three large, distinct clusters emerge, visualized in Figure 5.13(d) by reordering the rows of the data.

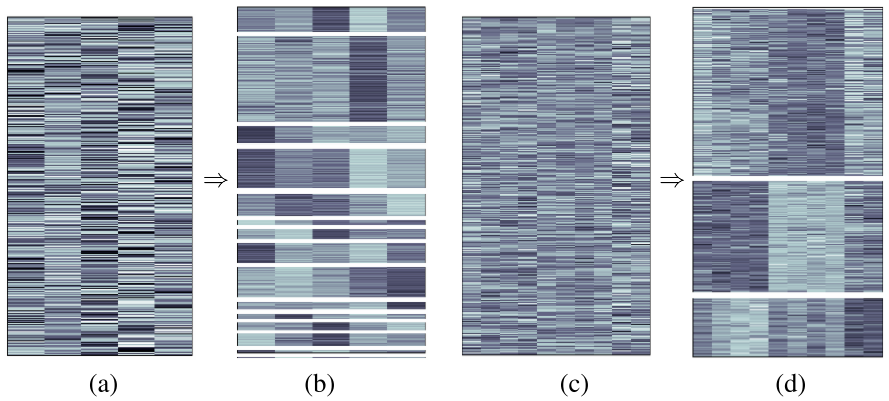


Figure 5.13. Results on two microarray datasets. (a) raw yeast data. (b) segmentation, visualized by reordering rows. The algorithm discovers a number of distinct clusters of varying size. (c) raw leukemia data. (d) segmentation. Three clusters are found.

5.5 Coding Length, Effective Dimension, and Sparse Representation

In the previous sections, we have seen that the existence of underlying subspaces in real data essentially enables a more compact or compressible representation; inversely, the process to find the most compact or compressed representation (either algebraic or statistical) for the data automatically leads to correct clustering of the data into multiple subspaces.

In Chapter 3, we have introduced the notion of minimum effective dimension to measure compactness of the subspace arrangement that best fits the given data set. Earlier in this chapter, we saw that minimum coding length provides another, arguably more direct, measure of compactness for the data set. It is therefore natural to ask how these two seemingly different measures are related to each other and more importantly, under what conditions they would give (approximately) the same solution to the subspace segmentation problem.

To reveal the connections between effective dimension and coding length, let us assume that an (orthonormal) basis B_j is known for each subspace S_j in the arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$. We can collect all the bases into a single matrix $A = [B_1, B_2, \ldots, B_n]$. Now consider using A to represent each point $\boldsymbol{x}_i$ in the data set $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$:

$$\boldsymbol{x}_i = A\alpha_i, \quad \forall i = 1, \ldots, N. \tag{5.17}$$

If the sum of the dimensions of all the subspaces $m \doteq d_1 + \cdots + d_n$ is larger than D, the above equation is under-determined and the solution for $\alpha_i \in \mathbb{R}^m$ is not unique. Among all the α_i's that satisfy the above linear equation, normally the correct solution for α_i has the fewest nonzero entries. That is, it gives the *sparsest* linear representation for $\boldsymbol{x}_i$ and minimizes the following objective:

$$(\ell^0) \qquad \min \|\alpha_i\|_0, \quad \text{subject to} \quad \boldsymbol{x}_i = A\alpha_i, \tag{5.18}$$

where $\|\cdot\|_0$ is the 0-norm that counts the number of nonzero entries in a vector. Under mild conditions (such as the dimension of each subspace is low enough), the sparsest α_i only have nonzero entries for base vectors of the subspace to which $\boldsymbol{x}_i$ belongs. So if $\boldsymbol{x}_i \in S_k$, we have $\|\alpha_i\|_0 \leq d_k$. Thus, we can measure how well A represents the data set $\boldsymbol{X}$ by the smallest number of nonzero coefficients:

$$\sum_{i=1}^{N} \|\alpha_i\|_0 = \sum_{i=1}^{N}\sum_{j=1}^{m} \|\alpha_{ij}\|_0. \tag{5.19}$$

Notice that this is precisely the second term of the effective dimension (3.55) (without the first term which is the overhead needed to describe the bases).

Although finding the sparsest solution α_i to an under-determined system of linear equations is an NP-hard problem, the recent work of [**?**] has show that if the solution α_i is sparse enough, it can be found precisely by minimizing the ℓ^1 norm instead:

$$(\ell^1) \qquad \min \|\alpha_i\|_1, \quad \text{subject to} \quad \boldsymbol{x}_i = A\alpha_i. \tag{5.20}$$

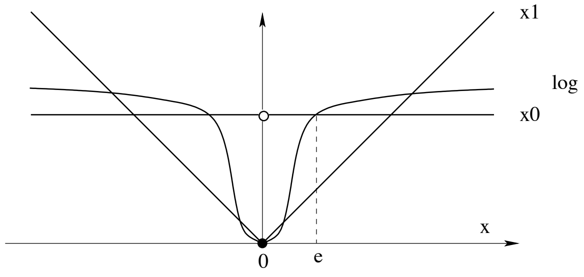


Figure 5.14. Comparison of three functions that measure the compactness of the representation.

This is a convex optimization problem that can be solved efficient via linear programming.

One can view that the ℓ^1-norm $\|x\|_1$ for $x \in \mathbb{R}$ is a continuous "convexification" of the ℓ^0-norm $\|x\|_0$. The convex function $\|\cdot\|_1$ reaches minimum at the same location as the nonconvex $\|\cdot\|_0$. However, the ℓ^1-norm $\|x\|_1$ is not a differentiable function art $x = 0$ whereas the regularized logarithmic function $\log(1 + \frac{x^2}{\varepsilon^2})$ is. It reaches minimum at the same location $x = 0$ as both $\|x\|_0$ and $\|x\|_1$ and approximates $\|x\|_0$ better as $\varepsilon \to 0$. Thus, we can expect to find approximately the same sparse solution α_i by solving the following nonlinear programming problem:

$$\text{(log)} \qquad \min \sum_{j=1}^{m} \frac{1}{2} \log\left(1 + \frac{\alpha_{ij}^2}{\varepsilon^2}\right), \quad \text{subject to} \quad \boldsymbol{x}_i = A\alpha_i. \tag{5.21}$$

Although the logarithmic function is not convex, it is a smooth function that strikes a good balance between $\|x\|_0$ and $\|x\|_1$ in the neighborhood of the minimum $x = 0$, as shown in Figure 5.14. Indeed, it has been observed empirically in the literature that minimizing the logarithmic function above can indeed further improve the algorithm's ability to recover sparse solutions over the ℓ^1-minimization [**?**].

In our context, the regularized logarithmic value is exactly the number of bits needed to encode any coefficient. According to our derivation in Appendix 5.A, the sum of the logarithmic value of the coefficients

$$\sum_{i=1}^{N} \sum_{j=1}^{m} \frac{1}{2} \log\left(1 + \frac{\alpha_{ij}^2}{\varepsilon^2}\right) \tag{5.22}$$

is precisely the total lossy coding length $L^s(\cdot)$ (5.5), only without the overhead for describing the bases and the segmentation. Since the minimum coding length is a close approximation to the entropy of the data, seeking the sparsest representation is essentially minimizing the entropy of the final representation.

From the above discussion, we now understand that minimizing either the effective dimension or the lossy coding length is simply to enforce that the resulting subspace arrangement gives the sparsest representation for the given data set. The theory of sparse representation ensures that when the relative dimension of each subspace is low enough, correctly minimizing either objective function should lead to (approximately) the same solution. However, the reader might have noticed that the effective dimension is easier to compute with a global algebraic model. Thus, it is used in the recursive GPCA Algorithm 3.5 to decide whether or not to "split" a large subspace into multiple smaller ones. The coding length is however easier to compute directly from the data and does not require an explicit model (although the tightness of the coding length function relies on the assumption that the data are Gaussian or linear). Thus, it is used in the agglomerative clustering Algorithm 5.1 to decide whether or not to "merge" data points into subspaces.

5.5.1 *Compressed Sensing and Clustering*

The relationship between the sparse measure (ℓ^0-norm) and the coding length is not simply that they approximate each other. There is a much more fundamental connection. In the theory of compressed sensing, a fundamental question is how to efficiently and accurately recover a sparse vector $\alpha_0 \in \mathbb{R}^m$ from a number of linear measurements, say $\boldsymbol{x} = A\alpha_0, \in \mathbb{R}^n$.

Clearly if α_0 is an arbitrary vector in $\mathbb{R}^m$, in general one must need $n \geq m$ linearly independent measurements in order to uniquely recover α_0 from $\boldsymbol{x} = A\alpha$. The theory of compressed sensing has revealed that if α_0 is sufficiently sparse, say has less than d non-zero entries, it can be efficiently and uniquely recovered from about $O(d \log m)$ linear measurements. In fact, this is an overwhelming phenomenon in the sense that these linear measurements can be randomly constructed. For instance, A can be a random matrix whose entries are drawn from independent Gaussian random variables.

In our context, the set of all d-sparse vectors in $\mathbb{R}^m$ form a special subspace arrangement Z_d: it contains all $\binom{m}{d}$ subspaces, each spanned by a set of d coordinates. Obviously any set of samples drawn from Z_d has an effective dimension d.

Now, let us consider the problem of retrieving a vector $\alpha_0 \in Z_d$ from the lossy data compression viewpoint. We may construct α_0 by randomly select d entries and then set their values as independent zero-mean Gaussians variables $r_1, \ldots, r_d$ with variance $E[r_i^2] = \sigma^2 = 1/d$. In doing so, we essentially impose that the expected length of α_0 be $E[\|\alpha_0\|^2] = 1$. This does not lose any generality as the equation $\boldsymbol{x} = A\alpha$ is linear in α.

In the context of lossy data compression, we expect to recover α_0 up to a prescribed quantization error ε. That is, the difference between the recovered $\hat{\alpha}_0$ and α_0 is: $\|\alpha_0 - \hat{\alpha}_0\|_2^2 \leq \varepsilon^2$. As α_0 is a vector in $\mathbb{R}^m$, the quantization error allowed for each entry is therefore $\varepsilon' = \frac{\varepsilon}{\sqrt{m}}$.

The expected lossy coding length for all such α_0 is:

$$\begin{aligned} E\left[\sum_{i=1}^{m}\frac{1}{2}\log\left(1+\frac{\alpha_{0i}^2}{\varepsilon'^2}\right)\right] &= E\left[\sum_{i=1}^{d}\frac{1}{2}\log\left(1+\frac{r_i^2}{\varepsilon'^2}\right)\right] \\ \leq \sum_{i=1}^{d}\frac{1}{2}\log\left(1+\frac{E[r_i^2]}{\varepsilon'^2}\right) &\leq \frac{1}{2}d\log\left(1+\frac{m}{d\varepsilon^2}\right) \\ \leq cd\log(m/d) & \end{aligned} \tag{5.23}$$

for some constant $c > 0$. In fact, in the case of independent Gaussians, the above bound is actually tight. That is, to specify a d-sparse vector α_0 in $\mathbb{R}^m$ (subject to an error ε, the average number of binary bits needed is about

$$H \doteq cd\log(m/d).$$

This is the entropy of the set of all d-sparse vectors in consideration. We should not expect to distinguish any two vectors in this set by any means at all if these vectors are described with less that $H \approx O(d\log m/d)$ bits.

Notice that the codebook that maps the vector α to the binary code could be rather complicated and hard to compute in general.[17] A surprising news from compressed sensing is that if $d \ll m$, α can actually be retrieved very efficiently, via linear programming, from essentially the same order, $O(d\log m/d)$, of any (real-valued) linear measurements [**?**]! As revealed by the work of Donoho and Tanner, the fundamental reason why this is even possible has a lot to do with an amazing fact associated with high-dimensional polytopes: The set of all d sparse vectors in $\mathbb{R}^m$ span the d skeleton of the standard cross polytope, the unit ℓ^1 ball, of $\mathbb{R}^m$. As it turns out, the d skeleton is preserved, with almost probability one, under any random projection onto an n-dimensional space as long as n is large enough:

$$n \geq 2d\log(m/n). \tag{5.24}$$

The inequality holds for the regime where n and d are asymptotically in proportion to each other. In particular, if $n/d = const.$, we see that the two bounds $cd\log(m/d)$ and $2d\log(m/n)$ are essentially the same. In other words, one has good chance to recover any d-sparse vector α_0 from $\boldsymbol{x} = A\alpha$ for any choice of $A \in \mathbb{R}^{n\times m}$, where $n \sim O(d\log m/n)$. The way to recover α is to find out which d-face $\boldsymbol{x}$ lies on the projection of the cross polytope via A. This can be done precisely by solving the following ℓ^1-minimization problem:

$$\min \|\alpha\|_1, \quad \text{subject to} \quad \boldsymbol{x} = A\alpha. \tag{5.25}$$

[17] In fact, for a general distribution, it is an extremely difficult problem to construct a code book that exactly achieves the rate-distortion function of the distribution.

5.5.2 *Dictionary Learning and Clustering*

In the discussion above, we have assumed that the set of bases are given for all the subspaces and they are put together as a dictionary $A = [B_1, B_2, \ldots, B_n]$ so that each data point in $\boldsymbol{X}$ has a sufficiently sparse representation with respect to this dictionary. Thus, conceptually, clustering can be thought as the problem of finding a dictionary A so that the points in $\boldsymbol{X}$ all have very sparse representations with respect to A. This is obviously a very challenging problem.

So another way to formulate the subspace segmentation problem is to find a special "factorization" of the data matrix $\boldsymbol{X}$:

$$\boldsymbol{X} = AC, \tag{5.26}$$

where A is a minimal dictionary (consisting of bases of the subspaces) and C is a matrix whose columns are expected to be very sparse. By A being minimal, we mean its size is the smallest possible while still guaranteeing the coefficient matrix C being sufficiently sparse. Notice that without sparsity requirement on C, the minimal dictionary A can simply be chosen to be the standard basis of $\mathbb{R}^D : A = I$. Also, if we only ask C to be the sparsest possible, we can simply choose A to be the data matrix itself: $A = \boldsymbol{X}$. Then, each data point $\boldsymbol{x}$ can be represented by exactly one element, $\boldsymbol{x}$ itself, in A.

Thus, we see that the subspace segmentation problem cannot be trivially reduced to a matrix factorization problem. In general, it needs proper joint regularization on the size of matrix A and the sparsity of C. Here we analyze the special case when the dimension of A is known, say the number and dimensions of all the subspaces are known in advance. That is, we know A is a $D \times m$ matrix with $m = d_1 + d_2 + \cdots + d_n$. In this case, we have to find a factorization $\boldsymbol{X} = AC$ such that C is the sparsest.

Since either A or C is unknown, this problem is still very difficult to solve directly. One popular approach to find a matrix factorization is through iteration between the two unknown factors: For any fixed A, we could try to find the sparsest C such that $\boldsymbol{X} = AC$; for a fixed C, we could find the best A that satisfies $\boldsymbol{X} = AC$. In the iteration, since either A or C might not be the actual solution, we do not expect the equation $\boldsymbol{X} = AC$ to hold exactly. Hence, we can look for the best approximation instead. The above iterative scheme can then be summarized as the following algorithm for learning the dictionary A from a given dataset $\boldsymbol{X}$.

In the literature, such a block coordinate descent algorithm is also known as the "Method of Directions" (MOD) [**?**]. Other variations to this methods include the popular K-SVD algorithm [**?**], which only differs from Algorithm 5.2 in how $A_{(k)}$ is updated at each iteration.

As an iterative algorithm, in general there is no guarantee that the above scheme will always converge to the correct set of base vectors for the subspaces. Indeed, from our experience, although the above scheme in general is able to find a better dictionary that improves the joint sparsity of the data points in $\boldsymbol{X}$, it does not always converge the globally optimal solution – the set of bases for the subspaces. Seeking better algorithms for learning sparse dictionary remains an open problem

Algorithm 5.2 (Iterative Learning of Sparse Dictionary).

```
1: **Input:** the data $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m) \in \mathbb{R}^{D \times N}$ and an error tolerance $\tau > 0$.
2: Initialize $k = 0$ and $A_{(0)} \in \mathbb{R}^{D \times N}$.
3: **while** error $> \tau$ **do**
4:         $k = k + 1$.
5:         For each $\boldsymbol{x}_i \in \boldsymbol{X}$, find $c_i \in \mathbb{R}^N$ as the solution to the problem:
               $c_i \quad = \quad \arg\min \|c\|_1 \quad \text{subject to} \quad \boldsymbol{x}_i = A_{(k-1)} c.$
6:         Let $C_{(k)} = [c_1, \dots, c_m]$ and update the dictionary as:
           $A_{(k)} \quad = \quad \arg\min_A \|\boldsymbol{X} - AC_{(k)}\|_F^2 = \boldsymbol{X} C_{(k)}^T \big(C_{(k)} C_{(k)}^T\big)^{-1}.$
7:         error $= \|\boldsymbol{X} - A_{(k)} C_{(k)}\|_F^2$.
8: **end**
9: **Output:** $A_{(k)}$ and $C_{(k)}$.
```

in machine learning. It has become a very active research topic recently due to its importance for problems in computer vision and pattern recognition.

5.6 Bibliographic Notes

Compression-based Clustering

Analysis in this chapter has revealed strong connections between data segmentation and data compression: the correct segmentation is associated with the actual entropy of the data. Compression as a principle has been proposed for data clustering before, e.g., [**?**] introducing an inter-point normalized compression distance that works for various data types. However, here the coding length gives a measure of distance between different *subsets* of the data. However, here we require the data to be real-valued.

The simulations and experiments have suggested potential connections with certain phase transition phenomena that often appear in statistical physics. From a theoretical standpoint, it would be highly desirable to obtain analytical conditions on the critical values of the distortion and the outlier density that can explain and predict the phase transition behaviors. So far, only the case with zero-dimensional subspaces, i.e. vector quantization (VQ), has been well characterized [?].

Relations to Other Metrics

It has been shown that computing the lossy ML estimate is approximately (up to first order, asymptotically) equivalent to minimizing the coding rate of the data subject to a distortion ε [**?**]:

$$\hat{\theta}_{LML} = \arg\min_{\theta,\pi} R(\hat{p}(\boldsymbol{X}), \theta, \varepsilon), \tag{5.27}$$

where $\hat{p}(\boldsymbol{X})$ is the empirical estimate of the probabilistic distribution from a set of sample data $\boldsymbol{X} = \{\boldsymbol{x}_i\}$. From the dimension-reduction perspective, one would attempt to directly minimizing the dimension, or rank$(\boldsymbol{X}_i)$, of each subset. It is well-known that the ε-regularized $\log\det(\cdot)$ is a good continuous surrogate for the discrete-valued rank function [**?**]. Techniques from compressed sensing have recently shown that when the rank is sufficiently low, the minimum of such surrogates coincides with the minimum-rank solution [**?**]. From the data-compression viewpoint, one might be more interested in an accurate estimate of the total volume of the data set. Notice that $\det(\boldsymbol{X}_j\boldsymbol{X}_j^T + \varepsilon^2 I)$ is an ε-regularized volume of the subset $\boldsymbol{X}_j$, which is well-defined even if $\boldsymbol{X}_j$ lie on a proper subspace. Thus, minimizing the lossy coding length function L^s indeed unifies and generalizes other statistical or geometric metrics popular for data compression and clustering.

Improving the Agglomerative Algorithm

There are many possible ways to further improve the efficiency or convergence of the proposed greedy algorithm. For instance, one may adopt more advanced split-and-merge strategies (such as those in [?]) or random techniques (such as [**?**]) to improve the speed of the algorithm. It is possible though that in the future, one can even develop more efficient and effective algorithms to minimize the coding length function, entirely different from the agglomerative approach proposed in this chapter.

Extensions to Classification

The good performance of the Agglomerative Algorithm 5.1 can be partly justified from a classification perspective. Notice that at line 4 of Algorithm 5.1, the algorithm is essentially dealing with a *classification* problem. For each sample point $\boldsymbol{x} \in \mathbb{R}^n$ that has not been merged into one of the existing clusters, $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k$, the algorithm compares how many additional bits are needed to encode it with each one of the clusters:

$$\delta L(\boldsymbol{x}, j) = L(\boldsymbol{X}_j \cup \{\boldsymbol{x}\}) - L(\boldsymbol{X}_j) + L(j), \tag{5.28}$$

where the last term is the cost of losslessly coding the label y for $\boldsymbol{x}$ as $y = j$. Thus, the algorithm simply assigns $\boldsymbol{x}$ to the cluster that minimizes the number of additional bits needed to code $(\boldsymbol{x}, \hat{y})$, which we refer to as the minimum incremental coding length (MICL) criterion:

$$\hat{y}(\boldsymbol{x}) \doteq \arg\min_{y=1,\ldots,k} \delta L(\boldsymbol{x}, y). \tag{5.29}$$

Somewhat surprisingly, this seemingly naive rule followed by the merging process at each step is in fact nearly optimal! Or more precisely, classification based on the MICL criterion is, asymptotically, equivalent to a regularized version of maximum a posterior (MAP) classifier, see [**?**] for a proof.

5.7 Exercises

Exercise 5.1 (Large Quantization Error). Show that when $\varepsilon \longrightarrow \infty$, the coding length function L^s reaches minimum when all sample points are merged into one group.

Exercise 5.2 (Small Quantization Error: Zero Mean Case). Now we characterize what happens when $\varepsilon \longrightarrow 0$. Here we assume all clusters are zero mean and consider the coding length function (5.4).

1. Show that for two sample points $\boldsymbol{x}_1, \boldsymbol{x}_2$ that are linearly independent, as $\varepsilon \longrightarrow 0$, $L(\{\boldsymbol{x}_1, \boldsymbol{x}_2\}) > L^s(\{\boldsymbol{x}_1\}, \{\boldsymbol{x}_2\})$.
2. Now for an arbitrary set $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, suppose $N > D^2$, then as $\varepsilon \longrightarrow 0$, grouping all points together will result in a shorter coding length than leaving each point as its own group.
3. Show that under the same conditions as the previous question, if the data are drawn from some non-singular distribution, then for sufficiently small ε, with probability 1, the minimum coding length is achieved by merging all samples into one group.

The first two facts show that for extremely small ε, the agglomerative Algorithm 5.1 will mostly likely get stuck in a local minimum as it does not merge any pair of points at all. This is what we have seen in the simulations. The third fact shows that for a generic distribution, with the sample size fixed, for an extremely small ε, assigning all samples into one group is actually the optimal solution. Notice that none of this invalidate the proposed algorithm as it is expected to work in the regime where the sample density is comparable to ε. But they do suggest that when the data are rather under-sampled, one should modify Algorithm 5.4 to better impose the global subspace structures.

Exercise 5.3 (Small Quantization Error: Affine Case). Show that if we use the coding length for the nonzero mean (affine) case (5.42), given in the Appendix 5.B, then as $\varepsilon \longrightarrow 0$, keeping each point separate gives a smaller coding length than grouping all points together. This is opposite to the zero mean case.

5.A Lossy Coding Length for Subspace-Like Data

In Section 5.2, we have shown that in principle, one can construct a coding scheme for a given set of data $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) \in \mathbb{R}^{D \times N}$ such that the average number of bits needed to encode each vector is bounded by

$$R(\boldsymbol{X}) = \frac{1}{2} \log_2 \det \big(I + \frac{D}{N\varepsilon^2} \boldsymbol{X}\boldsymbol{X}^T\big), \tag{5.30}$$

as if $\boldsymbol{X}$ is drawn from a multivariate Gaussian distribution of covariance $\Sigma = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^T$. However, we do not know in the non-parametric setting (i.e. with finite number of samples), whether the above coding length is still of any good. In this appendix, we provide a constructive proof that $L(\boldsymbol{X}) = (N + D)R(\boldsymbol{X})$ indeed gives a tight upper bound for the number of bits needed to encode $\boldsymbol{X}$. One interesting feature of the construction is that the coding scheme apparently relies on coding the subspace spanned by the vectors (i.e., the singular vectors) and the coordinates of the vectors with respect to the subspace. Thus geometrically, minimizing the coding length (via segmentation) is essentially to reduce the dimension of each subset of the data and the variance of each subset within each subspace.

Consider the singular value decomposition (SVD) of the data matrix $\boldsymbol{X} = U\Sigma V^T$. Let $B = (b_{ij}) = \Sigma V^T$. The column vectors of $U = (u_{ij})$ form a basis for the subspace spanned by vectors in W, and the column vectors of B are the coordinates of the vectors with respect to this basis.

For coding purpose, we store the approximated matrices $U + \delta U$ and $B + \delta B$. The matrix $\boldsymbol{X}$ can be recovered as

$$\boldsymbol{X} + \delta\boldsymbol{X} \doteq (U + \delta U)(B + \delta B) = UB + \delta UB + U\delta B + \delta U\delta B. \tag{5.31}$$

Then $\delta\boldsymbol{X} \approx \delta UB + U\delta B$ as entries of $\delta U\delta B$ are negligible when ε is small (relative to the data $\boldsymbol{X}$). The squared error introduced to the entries of $\boldsymbol{X}$ are

$$\begin{aligned}\sum_{i,j} \delta\boldsymbol{x}_{ij}^2 &= \mathbf{tr}\big(\delta W\delta W^T\big)\\ &\approx \mathbf{tr}\big(U\delta B\delta B^T U^T + \delta UBB^T\delta U^T + \delta UB\delta B^T U^T + U\delta BB^T\delta U^T\big).\end{aligned}$$

We may further assume that the coding errors δU and δB are zero-mean independent random variables. Using the fact that $\mathbf{tr}(AB) = \mathbf{tr}(BA)$, the expected squared error becomes

$$\mathbb{E}(\mathbf{tr}\big(\delta\boldsymbol{X}\delta\boldsymbol{X}^T)\big) = \mathbb{E}\big(\mathbf{tr}(\delta B\delta B^T)\big) + \mathbb{E}\big(\mathbf{tr}(\Sigma^2\delta U^T\delta U)\big).$$

Now, let us encode each entry b_{ij} with a precision $\varepsilon' = \frac{\varepsilon}{\sqrt{D}}$ and u_{ij} with a precision $\varepsilon''_j = \frac{\varepsilon\sqrt{N}}{\sqrt{\lambda_j D}}$, where λ_j is the jth eigenvalue of $\boldsymbol{X}\boldsymbol{X}^T$.[18] This is equivalent to assume that the error δb_{ij} is uniformly distributed in the interval $\big[-\frac{\varepsilon}{\sqrt{D}}, \frac{\varepsilon}{\sqrt{D}}\big]$ and δu_{ij} is uniformly distributed in the interval $\big[-\frac{\varepsilon\sqrt{N}}{\sqrt{\lambda_j D}}, \frac{\varepsilon\sqrt{N}}{\sqrt{\lambda_j D}}\big]$. Under such a coding precision, it is easy to verify that

$$\mathbb{E}\big(\mathbf{tr}(\delta\boldsymbol{X}\delta\boldsymbol{X}^T)\big) \le \frac{2\varepsilon^2 N}{3} < \varepsilon^2 N. \tag{5.32}$$

Then the mean squared error per vector in $\boldsymbol{X}$ is

$$\frac{1}{N}\mathbb{E}\big(\mathbf{tr}(\delta\boldsymbol{X}\delta\boldsymbol{X}^T)\big) < \varepsilon^2. \tag{5.33}$$

The number of bits to store the coordinates b_{ij} with precision $\varepsilon' = \frac{\varepsilon}{\sqrt{D}}$ is

$$\begin{aligned}&\sum_{i=1}^{D}\sum_{j=1}^{N}\frac{1}{2}\log_2\Big(1 + (\frac{b_{ij}}{\varepsilon'})^2\Big) = \frac{1}{2}\sum_{i=1}^{D}\sum_{j=1}^{N}\log_2\Big(1 + \frac{b_{ij}^2 D}{\varepsilon^2}\Big)\\ \le\ &\frac{N}{2}\sum_{i=1}^{D}\log_2\Big(1 + \frac{D\sum_{j=1}^{N} b_{ij}^2}{N\varepsilon^2}\Big) = \frac{N}{2}\sum_{i=1}^{D}\log_2\Big(1 + \frac{D\lambda_i}{N\varepsilon^2}\Big).\end{aligned}$$

[18]Notice that ε''_j normally does not increase with the number of vectors N, because λ_j increases proportionally to N.

In the above inequality, we have used the concavity of the $\log$ function:

$$\frac{\log(1+a_1)+\cdots+\log(1+a_n)}{n} \leq \log\Big(1+\frac{a_1+\cdots+a_n}{n}\Big) \tag{5.34}$$

for nonnegative real numbers $a_1, a_2, \ldots, a_n \geq 0$.

Similarly, the number of bits to store the entries of the singular vectors u_{ij} with precision $\varepsilon'' = \frac{\varepsilon\sqrt{N}}{\sqrt{\lambda_i}D}$ is

$$\begin{aligned}\sum_{i=1}^{D}\sum_{j=1}^{D}\frac{1}{2}\log_2\Big(1+\big(\frac{u_{ij}}{\varepsilon''}\big)^2\Big) &= \frac{1}{2}\sum_{i=1}^{D}\sum_{j=1}^{D}\log_2\Big(1+\frac{u_{ij}^2D^2\lambda_j}{N\varepsilon^2}\Big)\\ \leq \quad \frac{D}{2}\sum_{j=1}^{D}\log_2\Big(1+\frac{D^2\lambda_j\sum_{i=1}^{D}u_{ij}^2}{N\varepsilon^2}\Big) &= \frac{D}{2}\sum_{j=1}^{D}\log_2\Big(1+\frac{D\lambda_j}{N\varepsilon^2}\Big).\end{aligned}$$

Thus, for U and B together, we need a total of

$$L(\boldsymbol{X}) = \frac{N+D}{2}\sum_{i=1}^{D}\log_2\Big(1+\frac{D\lambda_i}{N\varepsilon^2}\Big) = \frac{N+D}{2}\log_2\det\big(I+\frac{D}{N\varepsilon^2}\boldsymbol{X}\boldsymbol{X}^T\big). \tag{5.35}$$

We thus have proved the statement given in the beginning of this section: $L(\boldsymbol{X}) = (N+D)\cdot R(\boldsymbol{X})$ gives a good upper bound on the number of bits needed to encode $\boldsymbol{X}$.

5.B Nonzero Mean Case

In the above analysis, we have assumed that the given vectors $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ are zero-mean. In general, these vectors may have a non-zero mean. In other words, the points represented by these vectors may lie in an affine subspace, instead of a linear subspace.

In case $\boldsymbol{X}$ is not zero mean, let $\mu \doteq \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i \in \mathbb{R}^D$ and define the matrix

$$V \doteq \mu \cdot 1_{1\times N} = (\mu, \mu, \ldots, \mu) \in \mathbb{R}^{D\times N}. \tag{5.36}$$

Then $\bar{\boldsymbol{X}} \doteq \boldsymbol{X} - V$ is a matrix whose column vectors have zero mean. We may apply the same coding scheme in the previous section to $\bar{\boldsymbol{X}}$.

Let $\bar{\boldsymbol{X}} = U\Sigma V^T \doteq UB$ be the singular value decomposition of $\bar{\boldsymbol{X}}$. Let $\delta U, \delta B, \delta\mu$ be the error in coding U, B, μ, respectively. Then the error induced on the matrix $\boldsymbol{X}$ is

$$\delta\boldsymbol{X} = \delta\mu \cdot 1_{1\times N} + U\delta B + \delta U B. \tag{5.37}$$

Assuming that $\delta U, \delta B, \delta\mu$ are zero-mean independent random variables, the expected total squared error is

$$\mathbb{E}\big(\mathbf{tr}(\delta\boldsymbol{X}\delta\boldsymbol{X}^T)\big) = N\mathbb{E}(\delta\mu^T\delta\mu) + \mathbb{E}\big(\mathbf{tr}(\delta B\delta B^T)\big) + \mathbb{E}\big(\mathbf{tr}(\Sigma\delta U^T\delta U)\big). \tag{5.38}$$

We encode entries of B and U with the same precision as before. We encode each entry μ_i of the mean vector μ with the precision $\varepsilon' = \frac{\varepsilon}{\sqrt{D}}$ and assume that the error $\delta\mu_i$ is a uniform distribution in the interval $\big[-\frac{\varepsilon}{\sqrt{D}}, \frac{\varepsilon}{\sqrt{D}}\big]$. Then we have $N\mathbb{E}(\delta\mu^T\delta\mu) = \frac{N\varepsilon^2}{3}$. Using equation (5.32) for the zero-mean case, the total squared error satisfies

$$\mathbb{E}\big(\mathbf{tr}(\delta\boldsymbol{X}\delta\boldsymbol{X}^T)\big) \leq \frac{N\varepsilon^2}{3} + \frac{2N\varepsilon^2}{3} = N\varepsilon^2. \tag{5.39}$$

Then the mean squared error per vector in W is still bounded by ε^2:

$$\frac{1}{N}\mathbb{E}\big(\mathbf{tr}(\delta W\delta W^T)\big) \leq \varepsilon^2. \tag{5.40}$$

Now in addition to the $L(\bar{\boldsymbol{X}})$ bits needed to encode U and B, the number of bits needed to encode the mean vector μ with precision $\varepsilon' = \frac{\varepsilon}{\sqrt{D}}$ is

$$\sum_{i=1}^{D}\frac{1}{2}\log_2\Big(1+\big(\frac{\mu_i}{\varepsilon'}\big)^2\Big) = \frac{1}{2}\sum_{i=1}^{D}\log_2\Big(1+\frac{D\mu_i^2}{\varepsilon^2}\Big) \leq \frac{D}{2}\log_2\Big(1+\frac{\mu^T\mu}{\varepsilon^2}\Big), \tag{5.41}$$

where the last inequality follows from the inequality (5.34).

Thus, the total number bits needed to store $\boldsymbol{X}$ is

$$L(\boldsymbol{X}) = \frac{N+D}{2}\log_2\det\big(I + \frac{D}{N\varepsilon^2}\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^T\big) + \frac{D}{2}\log_2\Big(1+\frac{\mu^T\mu}{\varepsilon^2}\Big). \tag{5.42}$$

Notice that if $\boldsymbol{X}$ is actually zero-mean, we have $\mu = 0$, $\bar{\boldsymbol{X}} = \boldsymbol{X}$, and the above expression for $L(\boldsymbol{X})$ is exactly the same as before.

Part II

Applications in Image Processing & Computer Vision

Chapter 6
Image Representation

In this chapter, we demonstrate why subspace arrangements can be a very useful class of models for image processing and how the subspace-segmentation techniques may facilitate many important image processing tasks, such as image representation (compression), segmentation, and classification.

6.1 Image Representation as a GPCA Problem

Researchers in image processing and computer vision have long sought for efficient and sparse representations of images. Except for a few image representations such as fractal-based approaches [Fisher, 1995], most existing sparse image representations use an effective linear transformation so that the energy of the (transformed) image will be concentrated in the coefficients of a small set of bases of the transformation. Computing such a representation is typically the first step of subsequent (lossy) compression of the image.[1] The result can also be used for other purposes such as image segmentation,[2] classification, and object recognition.

Most of the popular methods for obtaining a sparse representation of images can be roughly classified into two categories.

1. *Fixed-Basis Linear Transformations.* Methods of the first category seek to transform all images using a *pre-fixed* linear transformation. Each image

[1]Which involves further quantization and entropy-coding of the so-obtained representation.
[2]As we will study in the next section.

is then represented as a superposition of a set of basis functions (specified by the transformation). These methods essentially all evolved from the classical Fourier Transform. One variation of the (discrete) Fourier Transform, the Discrete Cosine Transform (DCT), serves as the core of the JPEG standard [Wallace, 1991]. Due to the Gibbs' phenomenon, DCT is poor at approximating discontinuities in the imagery signal. Wavelets [DeVore et al., 1992, Donoho et al., 1998, Mallat, 1999, Shapiro, 1993] have been developed to remedy this problem and have been shown to be optimal for representing 1-D signals with discontinuities. JPEG-2000 adopted wavelets as its standard. However, because wavelet transforms only deal with 1-D discontinuities, they are not well-suited to represent 2-D singularities along edges or contours. Anisotropic bases such as wedgelets [Donoho, 1999], curvelets [**?**], countourlets [Do and Vetterli, 2002] and bandlets [LePennec and Mallat, 2005] have been proposed explicitly to capture different 2-D discontinuities. These x-lets have been shown to be (approximately) optimal for representing objects with singularities along C^2-smooth edges.[3] However, natural images, especially images that have complex textures and patterns, do not consist solely of discontinuities along C^2-smooth edges. This is probably the reason why these edge-based methods do not seem to outperform (separable) wavelets on complex images. More generally, one should not expect that a (fixed) "gold-standard" transformation would work optimally for all images (and signals) in the world. Furthermore, conventional image (or signal) processing methods are developed primarily for gray-scale images. For color images or other multiple-valued images, one has to apply them to each value separately (e.g., one color channel at a time). The strong correlation that is normally present among the multiple values or colors is unfortunately ignored.

2. *Adaptive Transformations & Hybrid Models* Methods of the second category aim to identify the optimal (or approximately optimal) representation that is *adaptive* to specific statistics or structures of each image.[4] The Karhunen-Loève transform (KLT) or principal component analysis (PCA) [Effros and Chou, 1995] identifies the optimal principal subspace from the statistical correlation of the imagery data and represents the image as a superposition of the basis of the subspace. In theory, PCA provides the optimal linear sparse representation assuming that the imagery data satisfy a uni-modal distribution. However in reality, this assumption is rarely true. Natural images typically exhibit multi-modal statistics as they usually contain many heterogeneous regions with significantly different geometric

[3]Here, "optimality" means that the transformation achieves the optimal asymptotic for approximating the class of functions considered [DeVore, 1998].

[4]Here, unlike in the case of prefixed transformations, "optimality" means the representation obtained is the optimal one within the class of models considered, in the sense that it minimizes certain discrepancy between the model and the data.

structures or statistical characteristics (e.g. Figure 6.2). Heterogeneous data can be better-represented using a mixture of parametric models, one for each homogeneous subset. Such a mixture of models is often referred to as a *hybrid model*. Vector quantization (VQ) [Gersho and Gray, 1992] is a special hybrid model that assumes the imagery data are clustered around many different centers. From the dimension reduction point of view, VQ represents the imagery data with many 0-dimensional (affine) subspaces. This model typically leads to an excessive number of clusters or subspaces.[5] The primal sketch model [Guo et al., 2003] is another hybrid model which represents the high entropy parts of images with Markov random fields [Zhu et al., 1998, Wu et al., 2000] and the low entropy parts with sketches. The result is also some kind of a "sparse" representation of the image as superposition of the random fields and sketches. However, the primary goal of primal sketch is not to authentically represent and approximate the original image. It is meant to capture the (stochastic) generative model that produces the image (as random samples). Therefore, this type of models are more suited for image parsing, recognition, and synthesis than approximation and compression. In addition, finding the sketches and estimating the parameters of the random fields are computationally expensive and therefore less appealing for developing efficient image representation and compression schemes.

In this chapter, we would like to show how to combine the benefits of PCA and VQ by representing an image with multiple (affine) subspaces – one subspace for one image segment. The dimension and basis of each subspace are pertinent to the characteristics of the image segment it represents. We call this a *hybrid linear model* and will show that it strikes a good balance between simplicity and expressiveness for representing natural images.

A Multi-Scale Hybrid Linear Model for Lossy Image Representation.

One other important characteristic of natural images is that they are comprised of structures at many different (spatial) scales. Many existing frequency-domain techniques harness this characteristic [**?**]. For instance, wavelets, curvelets, and fractals have all demonstrated effectiveness in decomposing the original imagery signal into multiple scales (or subbands). As the result of such a *multi-scale* decomposition, the structures of the image at different scales (e.g., low v.s. high frequency/entropy) become better exposed and hence can be more compactly represented. The availability of multi-scale structures also significantly reduces the size and dimension of the problem and hence reduces the overall computational complexity.

[5]Be aware that compared to methods in the first category, representations in the second category typically need additional memory to store the information about the resulting model itself, e.g., the basis of the subspace in PCA, the cluster means in VQ.

Therefore, in this chapter we introduce a new approach to image representation by combining the hybrid paradigm and the multi-scale paradigm. The result is a *multi-scale hybrid linear model* which is based on an extremely simple concept: Given an image, at each scale level of its down-sample pyramid, fit the (residual) image by a (multiple-subspace) hybrid linear model. Compared to the single-scale hybrid linear model, the multi-scale scheme can reduce not only the size of the resulting representation but also the overall computational cost. Surprisingly, as we will demonstrate, such a simple scheme is able to generate representations for natural images that are more compact, even with the overhead needed to store the model, than most state-of-the-art representations, including DCT, PCA, and wavelets.

6.2 Image Representation with Hybrid Linear Models

6.2.1 *Linear versus Hybrid Linear Models*

In this section we introduce and examine the hybrid linear model for image representation. The relationship between hybrid linear models across different spatial scales will be discussed in Section 6.2.2.

An image $\boldsymbol{I}$ with width W, height H, and c color channels resides in a very high-dimensional space $\mathbb{R}^{W\times H\times c}$. We may first reduce the dimension by dividing the image into a set of non-overlapping b by b blocks.[6] Each b by b block is then stacked into a vector $\boldsymbol{x} \in \mathbb{R}^D$, where $D = b^2c$ is the dimension of the ambient space. For example, if $c = 3$ and $b = 2$, then $D = 12$. In this way, the image $\boldsymbol{I}$ is converted to a set of vectors $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^N$, where $N = WH/b^2$ is the total number of vectors.

Borrowing ideas from existing unsupervised learning paradigms, it is tempting to assume the imagery data $\{\boldsymbol{x}_i\}$ are random samples from a (non-singular) probability distribution or noisy samples from a smooth manifold. As the distribution or manifold can be very complicated, a common approach is to infer a best approximation within a simpler class of models for the distributions or manifolds. The "optimal" model is then the one that minimizes certain distance to the true model. Different choices of model classes and distance measures have led to many different learning algorithms developed in machine learning, pattern recognition, computer vision, and image processing. The most commonly adopted distance measure, for image compression, is the Mean Square Error (MSE) between the original image $\boldsymbol{I}$ and approximated image $\hat{\boldsymbol{I}}$,

$$\varepsilon_I^2 = \frac{1}{WHc}\|\hat{\boldsymbol{I}} - \boldsymbol{I}\|^2. \tag{6.1}$$

Since we will be approximating the (block) vectors $\{\boldsymbol{x}_i\}$ rather than the image pixels, in the following derivation, it is more convenient for us to define the Mean

[6] Therefore, b needs to be a common divisor of W and H.

Square Error (MSE) *per vector* which is different from ε_I^2 by a scale,

$$\varepsilon^2 = \frac{1}{N}\sum_{i=1}^{N}\|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2 = \frac{b^2}{WH}\sum_{i=1}^{N}\|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2 = \frac{b^2}{WH}\|\hat{\boldsymbol{I}} - \boldsymbol{I}\|^2 = (b^2c)\varepsilon_I^2. \tag{6.2}$$

The Peak Signal to Noise Ratio (PSNR) of the approximated image is defined as,[7]

$$\text{PSNR} \doteq -10 \log \varepsilon_I^2 = -10 \log \frac{\varepsilon^2}{b^2c}. \tag{6.3}$$

Linear Models.

If we assume that the vectors $\boldsymbol{x}$ are drawn from an anisotropic Gaussian distribution or a linear subspace, the optimal model subject to a given PSNR can be inferred by Principal Component Analysis (PCA) [Pearson, 1901, Hotelling, 1933, Jolliffe, 2002] or equivalently the Karhunen-Loève Transform (KLT) [Effros and Chou, 1995]. The effectiveness of such a linear model relies on the assumption that, although D can be large, all the vectors $\boldsymbol{x}$ may lie in a subspace of a much lower dimension in the ambient space $\mathbb{R}^D$. Figure 6.1 illustrates this assumption.

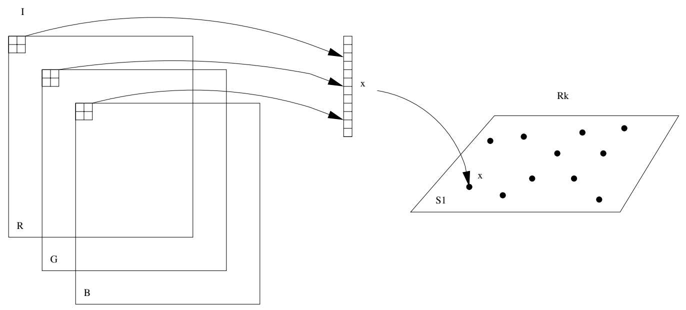


Figure 6.1. In a linear model, the imagery data vectors $\{\boldsymbol{x}_i \in \mathbb{R}^D\}$ reside in an (affine) subspace S of dimension $d \ll D$.

Let $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i$ be the mean of the imagery data vectors, and $\boldsymbol{X} = [\boldsymbol{x}_1 - \bar{\boldsymbol{x}}, \boldsymbol{x}_2 - \bar{\boldsymbol{x}}, \ \ldots\ , \boldsymbol{x}_N - \bar{\boldsymbol{x}}] = U\Sigma V^T$ be the SVD of the mean-subtracted data matrix $\boldsymbol{X}$. Then all the vectors $\boldsymbol{x}_i$ can be represented as a linear superposition: $\boldsymbol{x}_i = \bar{\boldsymbol{x}} + \sum_{j=1}^{D}\alpha_i^j\phi_j, i = 1, ..., N$, where $\{\phi_j\}_{j=1}^D$ are just the columns of the matrix U.

The matrix $\Sigma = \text{diag}(\sigma_1, \sigma_2,\ \ldots\ , \sigma_D)$ contains the ordered singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_D$. It is well known that the optimal linear representation of

[7]The peak value of the imagery data is normalized into 1.

$\boldsymbol{x}_i$ subject to the MSE ε^2 is obtained by keeping the first d (principal) components

$$\hat{\boldsymbol{x}}_i \doteq \bar{\boldsymbol{x}} + \sum_{k=1}^{d} \alpha_i^k \phi_k, \quad i = 1, ..., N, \tag{6.4}$$

where d is chosen to be

$$d = \min(k), \quad \text{s.t.} \quad \frac{1}{N} \sum_{i=k+1}^{D} \sigma_i^2 \leq \varepsilon^2. \tag{6.5}$$

The model complexity of the linear model, denoted as Ω, is the total number of coefficients needed for representing the model $\{\alpha_i^k, \phi_k, \bar{\boldsymbol{x}}\}$ and subsequently a lossy approximation $\hat{\boldsymbol{I}}$ of the image $\boldsymbol{I}$. It is given by

$$\Omega(N, d) \doteq Nd + d(D - d + 1), \tag{6.6}$$

where the first term is the number of coefficients $\{\alpha_i^k\}$ to represent $\{\hat{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}\}_{i=1}^N$ with respect to the basis $\Phi = \{\phi_k\}_{k=1}^d$ and the second term is the number of Grassmannian coordinates[8] needed for representing the basis Φ and the mean vector $\bar{\boldsymbol{x}}$. The second term is often called *overhead*.[9] Notice that the original set of vectors $\{\boldsymbol{x}_i\}$ contain ND coordinate entries. If $\Omega \ll ND$, the new representation, although lossy, is more compact. The search for such a compact representation is at the heart of any (lossy) image compression method. When the image $\boldsymbol{I}$ is large and the block size b is small, N will be much larger than D so that the overhead will be much smaller than the first term. However, in order to compare fairly with other methods, in the subsequent discussions and experiments, we always count the total number of coefficients needed for the representation, including the overhead.

Hybrid Linear Models.

The linear model is very efficient when the target manifold or distribution function is indeed unimodal. However, if the image $\boldsymbol{I}$ contains several heterogeneous regions $\{\boldsymbol{I}_j\}_{j=1}^n$, the data vectors $\boldsymbol{x}_i$ can be samples from a collection of subspaces of possibly different dimensions or from a mixture of multiple (Gaussian) distributions. Figure 6.2 shows the first three principal components of the data vector $\boldsymbol{x}_i$ (as dots in $\mathbb{R}^3$) of an image. Note the clear multi-modal characteristic in the data.

Suppose that a natural image $\boldsymbol{I}$ can be segmented into n disjoint regions $\boldsymbol{I} = \cup_{j=1}^n \boldsymbol{I}_j$ with $\boldsymbol{I}_j \cap \boldsymbol{I}_{j'} = \emptyset$ for $j \neq j'$. In each region $\boldsymbol{I}_j$, we may assume the

[8]Notice that to represent a d-dimensional subspace in a D-dimensional space, we only need to specify a basis of d linearly independent vectors for the subspace. We may stack these vectors as rows of a $d \times D$ matrix. Any nonsingular linear transformation of these vectors span the same subspace. Thus, without loss of generality, we may assume that the matrix is of the normal form $[I_{d\times d}, G]$ where G is a $d \times (D - d)$ matrix consisting of the so-called Grassmannian coordinates.

[9]Notice that if one uses a pre-chosen basis such as discrete Fourier transform, discrete cosine transform (JPEG), and wavelets (JPEG-2000), there is no such overhead.

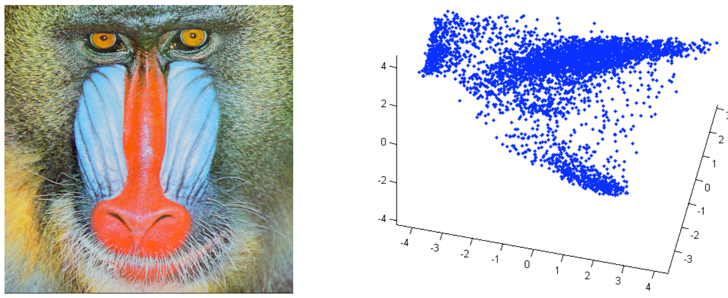


Figure 6.2. Left: The baboon image. Right: The coordinates of each dot are the first three principal components of the vectors $\boldsymbol{x}_i$. There is a clear multi-modal structure in the data.

linear model (6.4) is valid for the subset of vectors $\{\boldsymbol{x}_{j,i}\}_{i=1}^{N_j}$ in $\boldsymbol{I}_j$:

$$\hat{\boldsymbol{x}}_{j,i} = \bar{\boldsymbol{x}}_j + \sum_{k=1}^{d_j} \alpha_i^k \phi_{j,k}, \quad i = 1, ..., N_j. \tag{6.7}$$

Intuitively, the hybrid linear model can be illustrated by Figure 6.3.

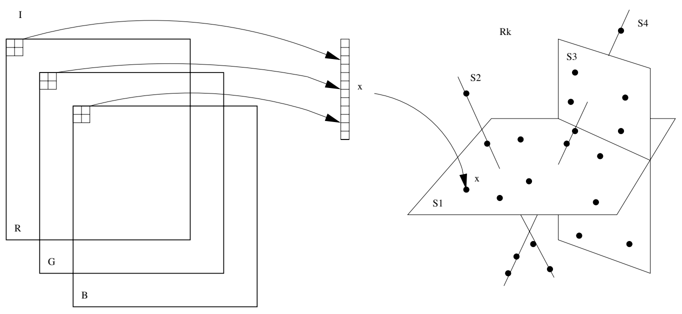


Figure 6.3. In hybrid linear models, the imagery data vectors $\{\boldsymbol{x}_i\}$ reside in multiple (affine) subspaces which may have different dimensions.

As in the linear model, the dimension d_j of each subspace is determined by a common desired MSE ε^2 using equation (6.5). The model complexity, i.e., the total number of coefficients needed to represent the hybrid linear model $\{\phi_{j,k}, \hat{\boldsymbol{x}}_{j,i}\}$ is[10]

$$\Omega = \Omega(N_1, d_1) + \cdots + \Omega(N_n, d_n) = \sum_{j=1}^{n} \big(N_j d_j + d_j(D - d_j + 1)\big). \tag{6.8}$$

[10]We also need a very small number of binary bits to store the membership of the vectors. But those extra bits are insignificant comparing to Ω and often can be ignored.

Notice that Ω is similar to the effective dimension (ED) of the hybrid linear representation defined in [**?**]. Thus, finding a representation that minimizes Ω is the same as minimizing the effective dimension of the imagery data set.[11]

Instead, if we model the union of all the vectors $\cup_{j=1}^{n}\{\boldsymbol{x}_{j,i}\}_{i=1}^{N_j}$ with a single subspace (subject to the same MSE), the dimension of the subspace in general needs to be $d = \min\{d_1 + \cdots + d_n, D\}$. It is easy to verify from the definition (6.6) that under reasonable conditions (e.g., n is bounded from being too large), we have

$$\Omega(N, d) \ > \ \Omega(N_1, d_1) + \cdots + \Omega(N_n, d_n). \tag{6.9}$$

Thus, if a hybrid linear model can be identified for an image, the resulting representation will in general be much more compressed than that with a single linear or affine subspace. This will also be verified by experiments on real images in Section 6.2.3.

However, such a hybrid linear model alone is not able to generate a representation that is as compact as that by other competitive methods such as wavelets. There are at least two aspects in which the above model can be further improved. Firstly, we need to further reduce the negative effect of overhead by incorporating a pre-projection of the data onto a lower dimensional space. Secondly, we need to implement the hybrid linear model in a multi-scale fashion. We will discuss the former aspect in the remainder of this section and leave the issues with multi-scale implementation to the next section.

Dimension Reduction via Projection.

In the complexity of the hybrid linear model (6.8), the first term is always smaller than that of the linear model (6.6) because $d_j \leq d$ for all j and $\sum_{j=1}^{n} N_j = N$. The second overhead term however can be larger than in that of the linear model (6.6) because the bases of multiple subspaces now must be stored. We here propose a method to further reduce the overhead by separating the estimation of the hybrid model into two steps.

In the first step, we may project the data vectors $\{\boldsymbol{x}_i\}$ onto a lower-dimensional subspace (e.g., via PCA) so as to reduce the dimension of the ambient space from D to D'. The justification for such a subspace projection has been discussed earlier in Section 3.1.2. Here, the dimension D' is chosen to achieve an MSE $\frac{1}{2}\varepsilon^2$. The data vectors in the lower ambient space $\mathbb{R}^{D'}$ are denoted as $\{\boldsymbol{x}'_i\}$. In the second step, we identify a hybrid linear model for $\{\boldsymbol{x}'_i\}$ within the lower-dimension ambient space $\mathbb{R}^{D'}$. In each subspace, we determine the dimension d_j subject to the MSE $\frac{1}{2}\varepsilon^2$. The two steps combined achieve an overall MSE ε^2, but they can

[11] In fact, the minimal Ω can also be associated to the Kolmogorov entropy or to the minimum description length (MDL) of the imagery data.

actually reduce the total model complexity to

$$\Omega = \sum_{j=1}^{n} \big(N_j d_j + d_j(D' - d_j + 1)\big) + D(D'+1). \tag{6.10}$$

This Ω will be smaller than the Ω in equation (6.8) because D' is smaller than D. The reduction of the ambient space will also make the identification of the hybrid linear model (say by GPCA) much faster.

If the number of subspaces, n, is given, algorithms like GPCA or EM can always find a segmentation. The basis $\{\phi_{j,k}\}$ and dimension d_j of each subspace are determined by the desired MSE ε^2. As n increases, the dimension of the subspaces may decrease, but the overhead required to store the bases may increase. The optimal n^* therefore can be found recursively by minimizing Ω for different n's, as shown in Figure 6.4. From our experience, we found that n is typically in the

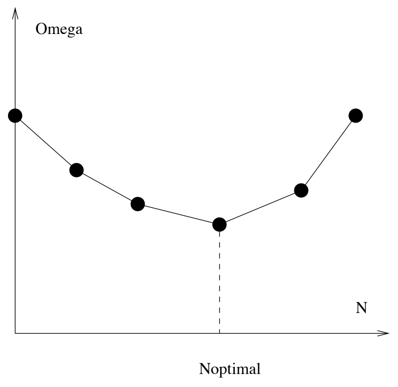


Figure 6.4. The optimal n^* can be found by minimizing Ω with respect to n.

range from 2 to 6 for natural images, especially in a multi-scale implementation that we will introduce next.

Algorithm 6.1 describes the pseudocode for estimating the hybrid linear model of an image $\boldsymbol{I}$, in which the *SubspaceSegmentation*$(\cdot)$ function is implemented (for the experiments in this chapter) using the GPCA algorithm given in earlier chapters. But it can also be implemented using EM or other subspace segmentation methods.

Example 6.1 (A Hybrid Linear Model for the Gray-Scale Barbara Image). Figure 6.5 and Figure 6.6 show intuitively a hybrid linear model identified for the 8×8 blocks of the standard 512×512 gray-scale Barbara image. The total number of blocks is $N = 4,096$. The GPCA algorithm identifies three subspaces for these blocks (for a given error tolerance), as shown in Figure 6.5. Figure 6.6 displays the three sets of bases for the three subspaces identified, respectively. It is worth noting that these bases are very consistent with the textures of the image blocks in the respective groups. ■

Algorithm 6.1 (Hybrid Linear Model Estimation).

```
1: function $\hat{\boldsymbol{I}} = $ HybridLinearModel($\boldsymbol{I}, \varepsilon^2$)
2: $\{\boldsymbol{x}_i\} = $ StackImageIntoVectors($\boldsymbol{I}$);
3: $\{\boldsymbol{x}'_i\}, \{\phi_k\}, \{\alpha_i^k\} = $ PCA($\{\boldsymbol{x}_i - \bar{\boldsymbol{x}}\}, \frac{1}{2}\varepsilon^2$);
4: for each possible $n$ do
5:     $\{\boldsymbol{x}'_{j,i}\} = $ SubspaceSegmentation($\{\boldsymbol{x}'_i\}, n$);
6:     $\{\hat{\boldsymbol{x}'}_{j,i}\}, \{\phi_{j,k}\}, \{\alpha_{j,i}^k\} = $ PCA($\{\boldsymbol{x}'_{j,i} - \bar{\boldsymbol{x}}'_j\}, \frac{1}{2}\varepsilon^2$);
7:     compute $\Omega_n$;
8: end for
9: $\Omega_{opt} = \min(\Omega_n)$;
10: $\hat{\boldsymbol{I}} = $ UnstackVectorsIntoImage($\{\hat{\boldsymbol{x}'}_{j,i}\}$ with $\Omega_{opt}$);
11: output $\{\alpha_i^k\}, \{\phi_k\}, \bar{\boldsymbol{x}}, \{\alpha_{j,i}^k\}, \{\phi_{j,k}\}, \{\bar{\boldsymbol{x}}'_j\}$ with $\Omega_{opt}$;
12: return $\hat{\boldsymbol{I}}$.
```

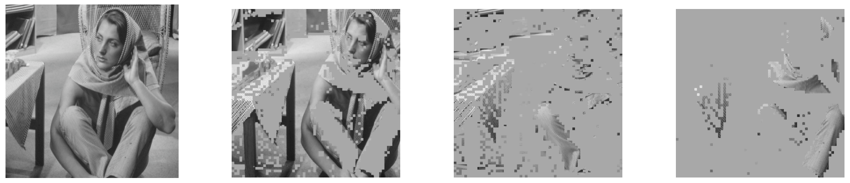


Figure 6.5. The segmentation of the 4,096 image blocks from the Barbara image. The image (left) is segmented into three groups (right three). Roughly speaking, the first subspace contains mostly image blocks with homogeneous textures; the second and third subspaces contain blocks with textures of different spatial orientations and frequencies.

6.2.2 *Multi-Scale Hybrid Linear Models*

There are at least several reasons why the above hybrid linear model needs further improvement. Firstly, the hybrid linear model treats low frequency/entropy regions of the image in the same way as the high frequency/entropy regions, which is inefficient. Secondly, by treating all blocks the same, the hybrid linear model fails to exploit stronger correlations that typically exist among adjacent image blocks.[12] Finally, estimating the hybrid linear model is computationally expensive when the image is large. For example, we use 2 by 2 blocks, a 512 by 512 color image will have $M = 65,536$ data vectors in $\mathbb{R}^{12}$. Estimating a hybrid linear model for such a huge number of vectors is difficult (if not impossible) on a regular PC. In this section, we introduce a multi-scale hybrid linear representation which is able to resolve the above issues.

[12]For instance, if we take all the b by b blocks and scramble them arbitrarily, the scrambled image would be fit equally well by the same hybrid linear model for the original image.

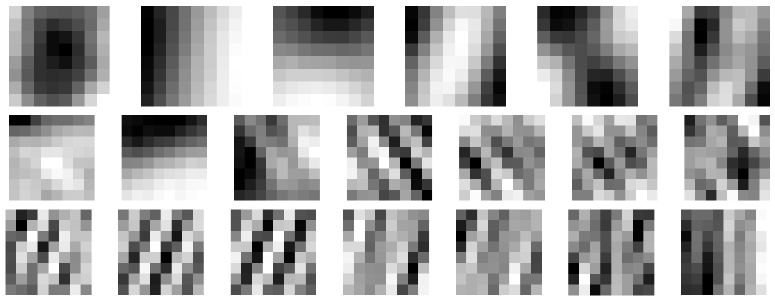


Figure 6.6. The three sets of bases for the three subspaces (of blocks) shown in Figure 6.5, respectively. One row for one subspace and the number of base vectors (blocks) is the dimension of the subspace.

The basic ideas of multi-scale representations such as the Laplacian pyramid [?] have been exploited for image compression for decades (e.g., wavelets, subband coding). A multi-scale method will give a more compact representation because it encodes low frequency/entropy parts and high frequency/entropy parts separately. The low frenquecy/entropy parts are invariant after low-pass filtering and down-sampling, and can therefore be extracted from the much smaller down-sampled image. Only the high frenquecy/entropy parts need to be represented at a level of higher resolution. Furthermore, the stronger correlations among adjacent image blocks will be captured in the down-sampled images because every four images blocks are merged into one block in the down-sampled image. At each level, the number of imagery data vectors is one fourth of that at one level above. Thus, the computational cost can also be reduced.

We now introduce a multi-scale implementation of the hybrid linear model. We use the subscript l to indicate the level in the pyramid of down-sampled images.[13] The finest level (the original image) is indicated by $l = 0$. The larger is l, the coarser is the down-sampled image. We denote the highest level to be $l = L$.

Pyramid of Down-Sampled Images.

First, the level-l image $\boldsymbol{I}_l$ passes a low-pass filter F_1 (averaging or Gaussian filter, etc) and is down-sampled by 2 to get a coarser version image $\boldsymbol{I}_{l+1}$:

$$\boldsymbol{I}_{l+1} \doteq \mathrm{F}_1(\boldsymbol{I}_l) \downarrow 2, \quad l = 0, ..., L-1. \tag{6.11}$$

The coarsest level-L image $\boldsymbol{I}_L$ is approximated by $\hat{\boldsymbol{I}}_L$ using a hybrid linear model with the MSE ε_L^2. The number of coefficients needed for the approximation is Ω_L.

Pyramid of Residual Images.

At all other levels l, $l = 0, ..., L-1$, we do *not* need to approximate the down-sampled image $\boldsymbol{I}_l$ because it has been roughly approximated by the image at

[13]This is not to be confused with the subscript j used to indicate different segments of an image.

level-$(l+1)$ upsampled by 2. We only need to approximate the residual of this level, denoted as $\boldsymbol{I}'_l$:

$$\boldsymbol{I}'_l \doteq \boldsymbol{I}_l - \mathrm{F}_2(\hat{\boldsymbol{I}}_{l+1}) \uparrow 2, \quad l = 0, ..., L-1, \tag{6.12}$$

where the F_2 is an interpolation filter. Each of these residual images $\boldsymbol{I}'_l$, $l = 0, ..., L-1$ is approximated by $\hat{\boldsymbol{I}}'_l$ using a hybrid linear model with the MSE ε_l^2. The number of coefficients needed for the approximation is Ω_l, for each $l = 0, ..., L-1$.

Pyramid of Approximated Images.

The approximated image at the level-l is denoted as $\hat{\boldsymbol{I}}_l$:

$$\hat{\boldsymbol{I}}_l \doteq \hat{\boldsymbol{I}}'_l + \mathrm{F}_2(\hat{\boldsymbol{I}}_{l+1}) \uparrow 2, \quad l = 0, ..., L-1. \tag{6.13}$$

Figure 6.7 shows the structure of a three-level ($L=2$) approximation of the image

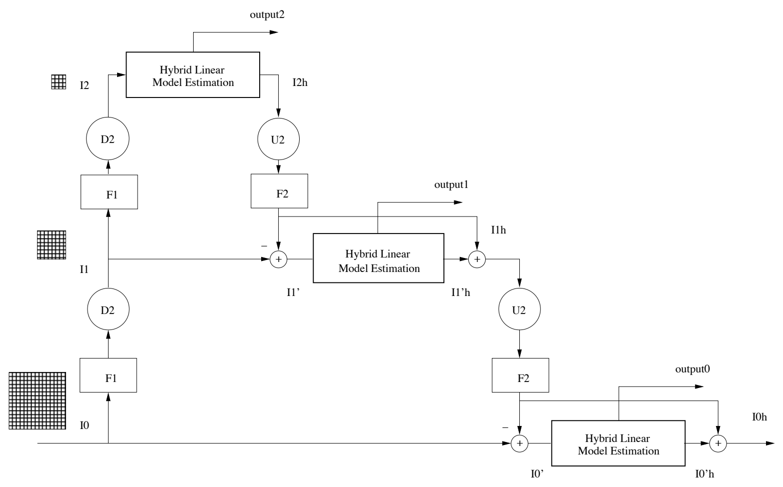


Figure 6.7. Laplacian pyramid of the multi-scale hybrid linear model.

$\boldsymbol{I}$. Only the hybrid linear models for $\hat{\boldsymbol{I}}_2$, $\hat{\boldsymbol{I}}'_1$, and $\hat{\boldsymbol{I}}'_0$, which are approximations for $\boldsymbol{I}_2$, $\boldsymbol{I}'_1$, and $\boldsymbol{I}'_0$, respectively, are needed for the final representation of the image. Figure 6.8 shows the $\boldsymbol{I}_2$, $\boldsymbol{I}'_1$, and $\boldsymbol{I}'_0$ for the baboon image.

The total number of coefficients needed for the representation will be

$$\Omega = \sum_{l=0}^{L} \Omega_l. \tag{6.14}$$

MSE Threshold at Different Scale Levels.

The MSE thresholds at different levels should be different but related because the up-sampling by 2 will enlarge 1 pixel at level-$(l+1)$ into 4 pixels at level-l. If the

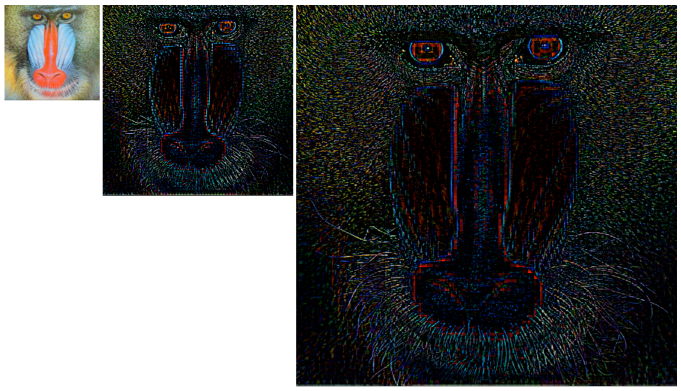


Figure 6.8. Multi-scale representation of the Baboon image. Left: The coarsest level image $\boldsymbol{I}_2$. Middle: The residual image $\boldsymbol{I}'_1$. Right: The residual image $\boldsymbol{I}'_0$. The data at each level are modeled as the hybrid linear models. The contrast of the middle and right images has been adjusted so that they are visible.

MSE of the level-$(l+1)$ is ε^2_{l+1}, the MSE of the level-l after the up-sampling will become $4\varepsilon^2_{l+1}$. So the MSE thresholds of level-$(l+1)$ and level-l are related as

$$\varepsilon^2_{l+1} = \frac{1}{4}\varepsilon^2_l, \quad l = 0, ..., L-1. \tag{6.15}$$

Usually, the user will only give the desired MSE for the approximation of original image which is ε^2. So we have

$$\varepsilon^2_l = \frac{1}{4^l}\varepsilon^2, \quad l = 0, ..., L. \tag{6.16}$$

Vector Energy Constraint at Each Level.

At each level-l, $l = 0, ..., L-1$, not all the vectors of the residual need to be approximated. We only need to approximate the (block) vectors $\{\boldsymbol{x}_i\}$ of the residual image $\boldsymbol{I}'_l$ that satisfy the following constraint:

$$\|\boldsymbol{x}'_i\|^2 > \varepsilon^2_l. \tag{6.17}$$

In practice, the energy of most of the residual vectors is close to zero. Only a small portion of the vectors at each level-l need to be modeled (e.g. Figure 6.9). This property of the multi-scale scheme not only significantly reduces the overall representation complexity Ω but also reduces the overall computational cost as the number of data vectors processed at each level is much less than those of the original image. In addition, for a single hybrid linear model, when the image size increases, the computational cost will increase in proportion to the square of the image size. In the multi-scale model, if the image size increases, we can correspondingly increase the number of levels and the complexity increases only linearly in proportion to the image size.

The overall process of estimating the multi-scale hybrid linear model can be written as the recursive pseudocode in Algorithm 6.2.

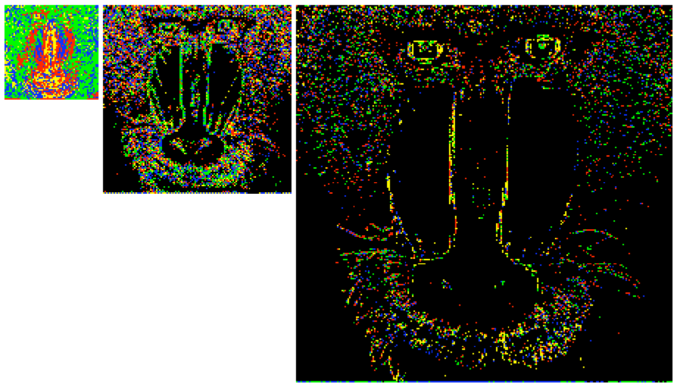


Figure 6.9. The segmentation of (residual) vectors at the three levels—different subspaces are denoted by different colors. The black regions correspond to data vectors whose energy is below the MSE threshold ε_l^2 in equation (6.17).

Algorithm 6.2 (Multi-Scale Hybrid Linear Model Estimation).

```
1:  function $\hat{\boldsymbol{I}}$ = MultiscaleModel($\boldsymbol{I}, level, \varepsilon^2$)
2:  if $level <$ MAXLEVEL then
3:      $\boldsymbol{I}_{down}$ = Downsample($\mathrm{F}_1(\boldsymbol{I})$);
4:      $\hat{\boldsymbol{I}}_{nextlevel}$ = MultiscaleModel($\boldsymbol{I}_{down}, level + 1, \frac{1}{4}\varepsilon^2$);
5:  end if
6:  if $level =$ MAXLEVEL then
7:      $\boldsymbol{I}' = \boldsymbol{I}$;
8:  else
9:      $\boldsymbol{I}_{up}$ = $\mathrm{F}_2$(Upsample($\hat{\boldsymbol{I}}_{nextlevel}$));
10:     $\boldsymbol{I}' = \boldsymbol{I} - \boldsymbol{I}_{up}$;
11: end if
12: $\hat{\boldsymbol{I}}'$ = HybridLinearModel($\boldsymbol{I}', \varepsilon^2$);
13: return $\boldsymbol{I}_{up} + \boldsymbol{I}'$.
```

6.2.3 *Experiments and Comparisons*

Comparison of Different Lossy Representations.

The first experiment is conducted on two standard images commonly used to compare image compression schemes: the 480×320 hill image and the 512×512 baboon image shown in Figure 6.10. We choose these two images because they are representative of two different types of images. The hill image contains large low frequency/entropy regions and the baboon image contains mostly high frequency/entropy regions. The size of the blocks b is chosen to be 2 and the level of the pyramid is 3 – we will test the effect of changing these parameters in

Figure 6.10. Testing images: the hill image (480×320) and the baboon image (512×512).

subsequent experiments. In Figure 6.11, the results of the multi-scale hybrid linear model are compared with several other commonly used image representations including DCT, PCA/KLT, single-scale hybrid linear model and Level-3 (Daubechies) biorthogonal 4.4 wavelets (adopted by JPEG-2000). The x-axis of the figures is the ratio of coefficients (including the overhead) kept for the representation, which is defined as,

$$\eta = \frac{\Omega}{WHc}. \tag{6.18}$$

The y-axis is the PSNR of the approximated image defined in equation (6.3). The

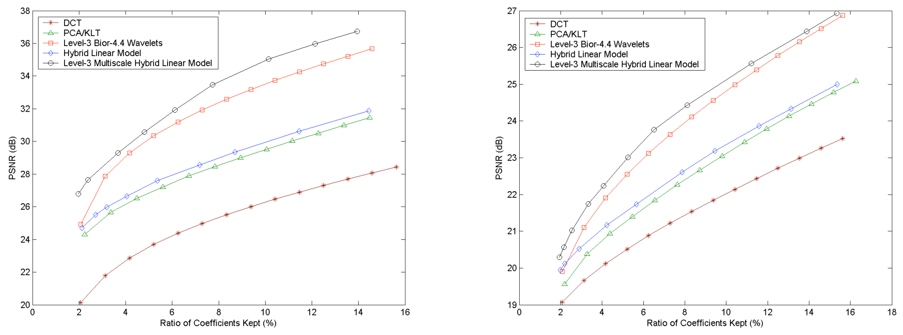


Figure 6.11. Left: Comparison of several image representations for the hill image. Right: Comparison for the baboon image. The multi-scale hybrid linear model achieves the best PSNR among all the methods for both images.

multi-scale hybrid linear model achieves the best PSNR among all the methods for both images. Figure 6.12 shows the two recovered images using the same amount of coefficients for the hybrid linear model and the wavelets. Notice that in the area around the whiskers of the baboon, the hybrid linear model preserves the detail of the textures better than the wavelets. But the multiscale hybrid linear model produces a slight block effect in the smooth regions.

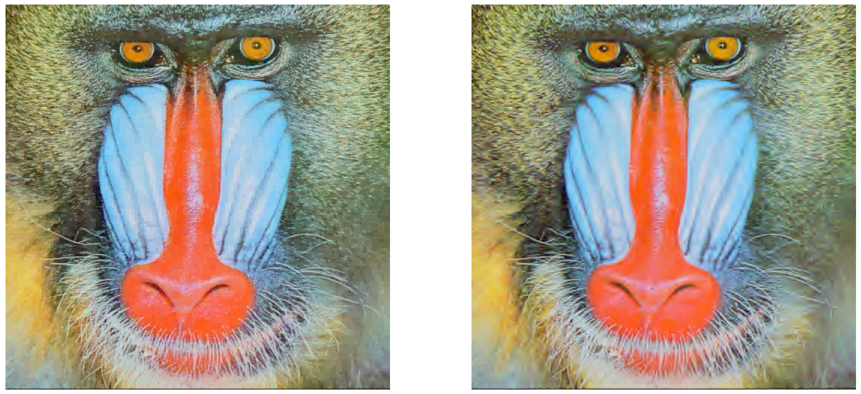


Figure 6.12. Left: The baboon image recovered from the multi-scale hybrid linear model using 7.5% coefficients of the original image. (PSNR=24.64). Right: The baboon image recovered from wavelets using the same amount of coefficients. (PSNR=23.94).

We have tested the algorithms on a wide range of images. We will summarize the observations in Section 6.2.4.

Effect of the Number of Scale Levels.

The second experiment shown in Figure 6.13 compares the multi-scale hybrid

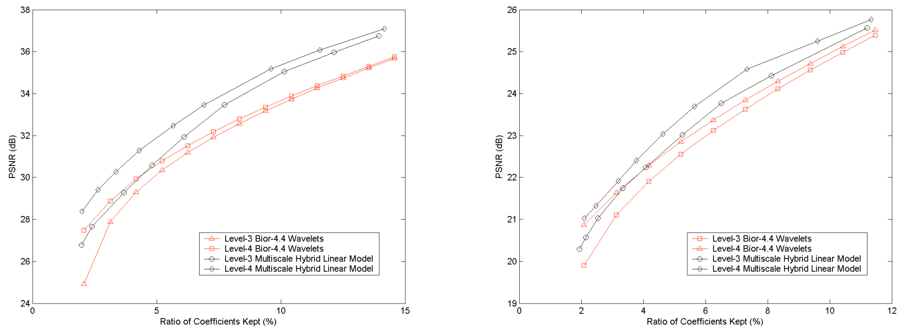


Figure 6.13. Top: Comparison of the multi-scale hybrid linear model with wavelets for level-3 and level-4 for the hill image. Bottom: The same comparison for the baboon image. The performance increases while the number of levels increases from 3 to 4.

linear representation with wavelets for different number of levels. It is conducted on the hill and baboon image with 2 by 2 blocks. The performance increases while the number of levels is increased from 3 to 4. But if we keep increasing the number of levels to 5, the level-5 curves of both wavelets and our method (which are not shown in the figures) coincide with the level-4 curves. The performance cannot improve any more because the down-sampled images in the fifth level are so small that it is hard to be further compressed. Only when the image is large, can we use more levels of down-sampling to achieve a more compressed representation.

Effect of the Block Size.

The third experiment shown in Figure 6.14 compares the multi-scale hybrid lin-

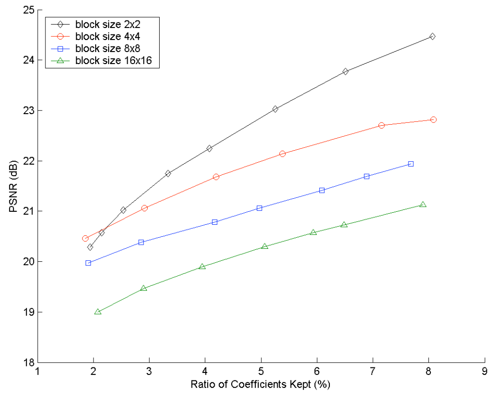


Figure 6.14. Comparison of the multi-scale hybrid linear model with different block sizes: 16, 8, 4, 2. The performance increases while the size of blocks decreases.

ear models with different block sizes from 2×2 to 16×16. The dimension of the ambient space of the data vectors $\boldsymbol{x}$ ranges from 12 to 192 accordingly. The testing image is the baboon image and the number of down-sampling levels is 3. For large blocks, the number of data vectors is small but the dimension of the subspaces is large. So the overhead would be large and seriously degrade the performance. Also the block effect will be more obvious when the block size is large. This experiment shows that 2 is the optimal block size, which also happens to be compatible with the simplest down-sampling scheme.

6.2.4 *Limitations*

We have tested the multi-scale hybrid linear model on a wide range of images, with some representative ones shown in Figure 6.15. From our experiments and experience, we observe that the multi-scale hybrid linear model is more suitable than wavelets for representing images with multiple high frequency/entropy regions, such as those with sharp 2-D edges and rich of textures. Wavelets are prone to blur sharp 2-D edges but better at representing low frequency/entropy regions. This probably explains why the hybrid linear model performs slightly worse than wavelets for the Lena and the monarch – the backgrounds of those two images are out of focus so that they do not contain much high frequency/entropy content.

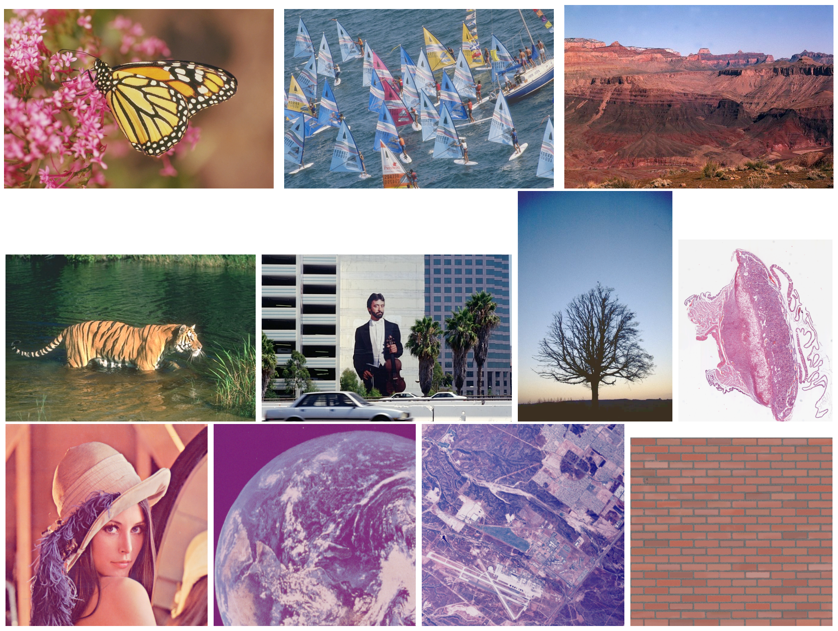


Figure 6.15. A few standard testing images. From the top-left to the bottom-right: monarch (768×512), sail (768×512), canyon (752×512), tiger (480×320), street (480×320), tree (512×768), tissue (microscopic) (1408×1664), Lena (512×512), earth (satellite) (512×512), urban (aerial) (512×512), bricks (696×648). The multi-scale hybrid linear model out-performs wavelets except for the Lena and the monarch.

Another limitation of the hybrid-linear model is that it does not perform well on gray-scale images (e.g., the Barbara image, Figure 6.5). For a gray-scale image, the dimension D of a 2 by 2 block is only 4. Such a low dimension is not adequate for any further dimension reduction. If we use a larger block size, say 8 by 8, the block effect will also degrade the performance.

Unlike pre-fixed transformations such as wavelets, our method involves identifying the subspaces and their bases. Computationally, it is more costly. With unoptimized MATLAB codes, the overall model estimation takes 30 seconds to 3 minutes on a Pentium 4 1.8GHz PC depending on the image size and the desired PSNR. The smaller the PSNR, the shorter the running time because the number of blocks needed to be coded in higher levels will be less.

6.3 Multi-Scale Hybrid Linear Models in Wavelet Domain

From the discussion in the previous section, we have noticed that wavelets can achieve a better representation for smooth regions and avoid the block artifacts.

Therefore, in this section, we will combine the hybrid linear model with the wavelet approach to build multi-scale hybrid linear models in the wavelet domain. For readers who are not familiar with wavelets, we recommend the books of [Vetterli and Kovacevic, 1995].

6.3.1 Imagery Data Vectors in Wavelet Domain

In the wavelet domain, an image is typically transformed into an octave tree of subbands by certain separable wavelets. At each level, the LH, HL, HH subbands contain the information about high frequency edges and the LL subband is further decomposed into subbands at the next level. Figure 6.16 shows the octave tree

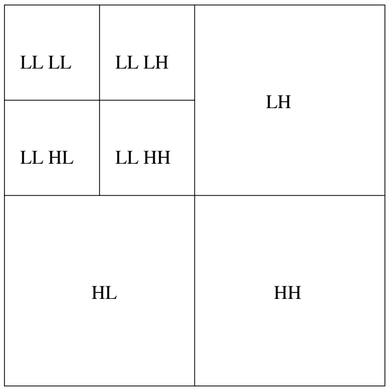


Figure 6.16. The subbands of a level-2 wavelet decomposition.

structure of a level-2 wavelet decomposition. As shown in the Figure 6.17, the

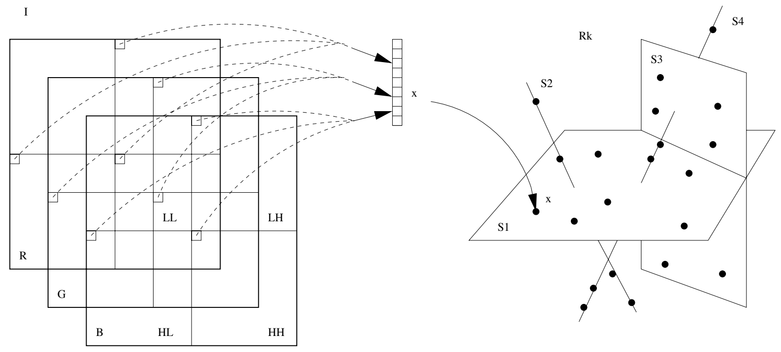


Figure 6.17. The construction of imagery data vectors in the wavelet domain. These data vectors are assumed to reside in multiple (affine) subspaces which may have different dimensions.

vectors $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^M$ are constructed by stacking the corresponding wavelet coefficients in the LH, HL, HH subbands. The dimension of the vectors is $D = 3c$ because there are c color channels. One of the reasons for this choice of vectors is because for edges along the same direction, these coefficients are linearly related and reside in a lower dimensional subspace. To see this, let us first assume that the color along an edge is constant. If the edge is along the horizontal, vertical or diagonal direction, there will be an edge in the coefficients in the LH, HL, or HH subband, respectively. The other two subbands will be zero. So the dimension of the imagery data vectors associated with such an edge will be 1. If the edge is not exactly in one of these three directions, there will be an edge in the coefficients of all the three subbands. For example, if the direction of the edge is between the horizontal and diagonal, the amplitude of the coefficients in the LH and HH subbands will be large. The coefficients in the HL subband will be insignificant relative to the coefficients in the other two subbands. So the dimension of the data vectors associated with this edge is approximately 2 (subject to a small error ε^2). If the color along an edge is changing, the dimension the subspace will be higher but generally lower than the ordinal dimension $D = 3c$. Notice that the above scheme is only one of many possible ways in which one may construct the imagery data vector in the wavelet domain. For instance, one may construct the vector using coefficients across different scales. It remains an open question whether such new constructions may lead to even more efficient representations than the one presented here.

6.3.2 Estimation of Hybrid Linear Models in Wavelet Domain

In the wavelet domain, there is no need to build a down-sampling pyramid. The multi-level wavelet decomposition already gives a multi-scale structure in the wavelet domain. For example, Figure 6.18 shows the octave three structure of

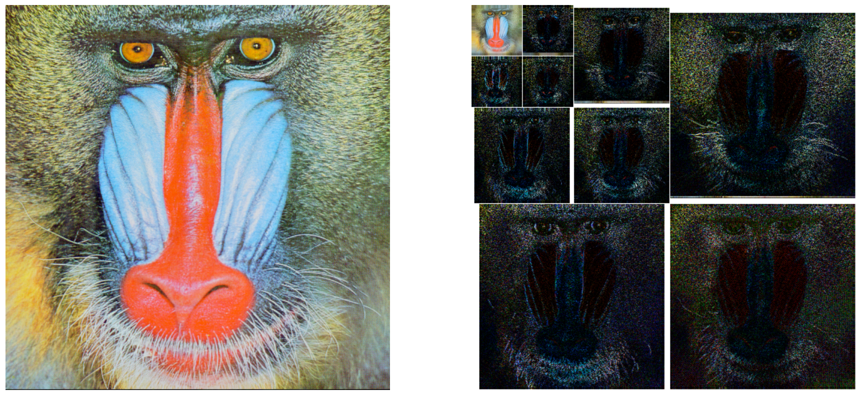


Figure 6.18. The subbands of level-3 bior-4.4 wavelet decomposition of the baboon image.

a level-3 bior-4.4 wavelet transformation of the baboon image. At each level, we may construct the imagery data vectors in the wavelet domain according to the previous section. A hybrid linear model will be identified for the so-obtained vectors at each level. Figure 6.19 shows the segmentation results using the hybrid linear model at three scale levels for the baboon image.

Vector Energy Constraint at Each Level. In the nonlinear wavelet approximation, the coefficients which are below an error threshold will be ignored. Similarly in our model, not all the vectors of the imagery data vectors need to be modeled and approximated. We only need to approximate the (coefficient) vectors $\{\boldsymbol{x}_i\}$ that satisfy the following constraint:

$$\|\boldsymbol{x}_i\|^2 > \varepsilon^2. \tag{6.19}$$

Notice that here we do not need to scale the error tolerance at different levels because the wavelet basis is orthonormal by construction. In practice, the energy of most of the vectors is close to zero. Only a small portion of the vectors at each level need to be modeled (e.g. Figure 6.19).

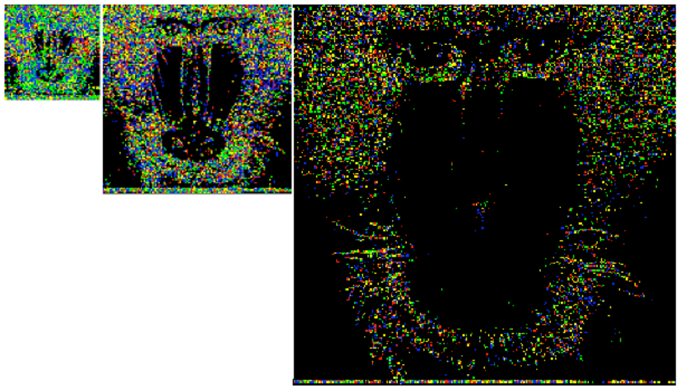


Figure 6.19. The segmentation of data vectors constructed from the three subbands at each level—different subspaces are denoted by different colors. The black regions correspond to data vectors whose energy is below the MSE threshold ε^2 in equation (6.19).

The overall process of estimating the multi-scale hybrid linear model in the wavelet domain can be summarized as the pseudocode in Algorithm 6.3.

6.3.3 Comparison with Other Lossy Representations

In this section, in order to obtain a fair comparison, the experimental setting is the same as that of the spatial domain in the previous section. The experiment is conducted on the same two standard images – the 480×320 hill image and the 512×512 baboon image shown in Figure 6.10.

The number of levels of the model is also chosen to be 3. In Figure 6.20, the results are compared with several other commonly used image representations including DCT, PCA/KLT, single-scale hybrid linear model and Level-3 biorthogonal 4.4 wavelets (JPEG 2000) as well as the multi-scale hybrid linear model in

Algorithm 6.3 **(Multi-Scale Hybrid Linear Model: Wavelet Domain).**

```
1: function $\hat{\boldsymbol{I}} = \text{MultiscaleModel}(\boldsymbol{I}, level, \varepsilon^2)$
2: $\tilde{\boldsymbol{I}} = \text{WaveletTransform}(\boldsymbol{I}, level)$;
3: for each $level$ do
4:     $\hat{\tilde{\boldsymbol{I}}}_{level} = \text{HybridLinearModel}(\tilde{\boldsymbol{I}}_{level}, \varepsilon^2)$;
5: end for
6: $\hat{\boldsymbol{I}} = \text{InverseWaveletTransform}(\hat{\tilde{\boldsymbol{I}}}, level)$;
7: return $\hat{\boldsymbol{I}}$.
```

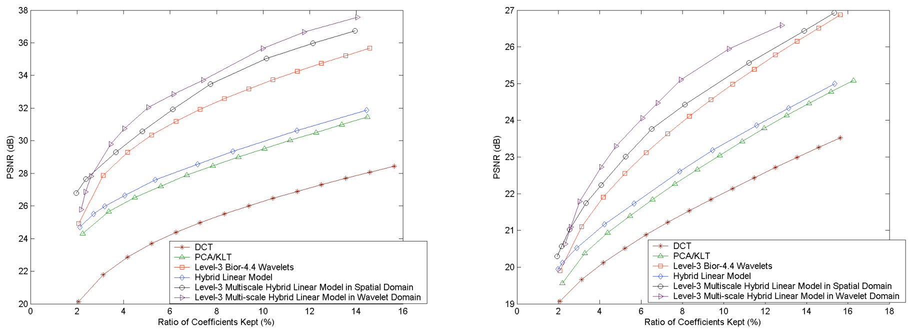


Figure 6.20. Top: Comparison of several image representations for the hill image. Bottom: Comparison for the baboon image. The multi-scale hybrid linear model in the wavelet domain achieves better PSNR than that in the spatial domain.

the spatial domain. The multi-scale hybrid linear model in the wavelet domain achieves better PSNR than that in the spatial domain. Figure 6.21 shows the three recovered images using the same amount of coefficients for wavelets, the hybrid linear model in the spatial domain, and that in the wavelet domain, respectively. Figure 6.22 shows the visual comparison with the enlarged bottom-right cornners of the images in Figure 6.21.

Notice that in the area around the baboon's whiskers, the wavelets blur both the whiskers and the subtle details in the background. The multi-scale hybrid linear model (in the spatial domain) preserves the sharp edges around the whiskers but generates slight block artifacts in the relatively smooth background area. The multi-scale hybrid linear model in the wavelet domain successfully eliminates the block artifacts, keeps the sharp edges around the whiskers, and preserves more details than the wavelets in the background. Among the three methods, the multi-scale hybrid linear model in the wavelet domain achieves not only the highest PSNR, but also produces the best visual effect.

As we know from the previous section, the multi-scale hybrid linear model in the spatial domain performs slightly worse than the wavelets for the Lena and

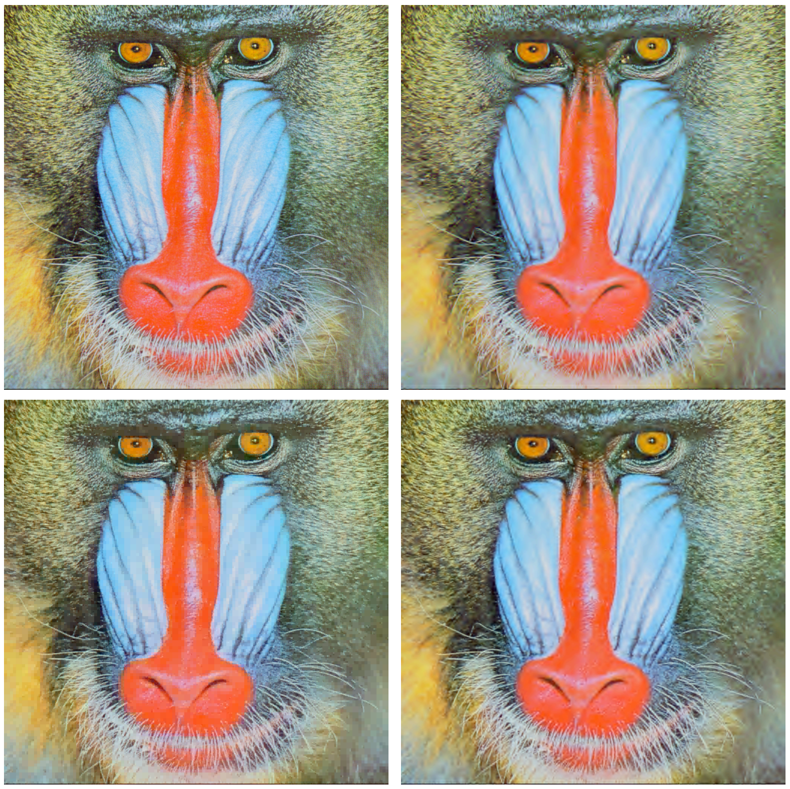


Figure 6.21. Visual comparison of three representations for the baboon image approximated with 7.5% coefficients. Top-left: The original image. Top-right: The level-3 biorthogonal 4.4 wavelets (PSNR=23.94). Bottom-left: The level-3 multi-scale hybrid linear model in the spatial domain (PSNR=24.64). Bottom-right: The level-3 multi-scale hybrid linear model in the wavelet domain (PSNR=24.88).

monarch images (Figure 6.15). Nevertheless, in the wavelet domain, the multi-scale hybrid linear model can generate very competitive results, as shown in Figure 6.23. The multi-scale hybrid linear model in the wavelet domain achieves better PSNR than the wavelets for the monarch image. For the Lena image, the comparison is mixed and merits further investigation.

6.3.4 Limitations

The above hybrid linear model (in the wavelet domain) does not produce very competitive results for gray-scale images as the dimension of the vector is merely 3 and there is little room for further reduction. For gray-scale images, one may have to choose a slightly larger window in the wavelet domain or to construct the vector using wavelet coefficients across different scales. A thorough investigation of all the possible cases is beyond the scope of this book. The purpose here is

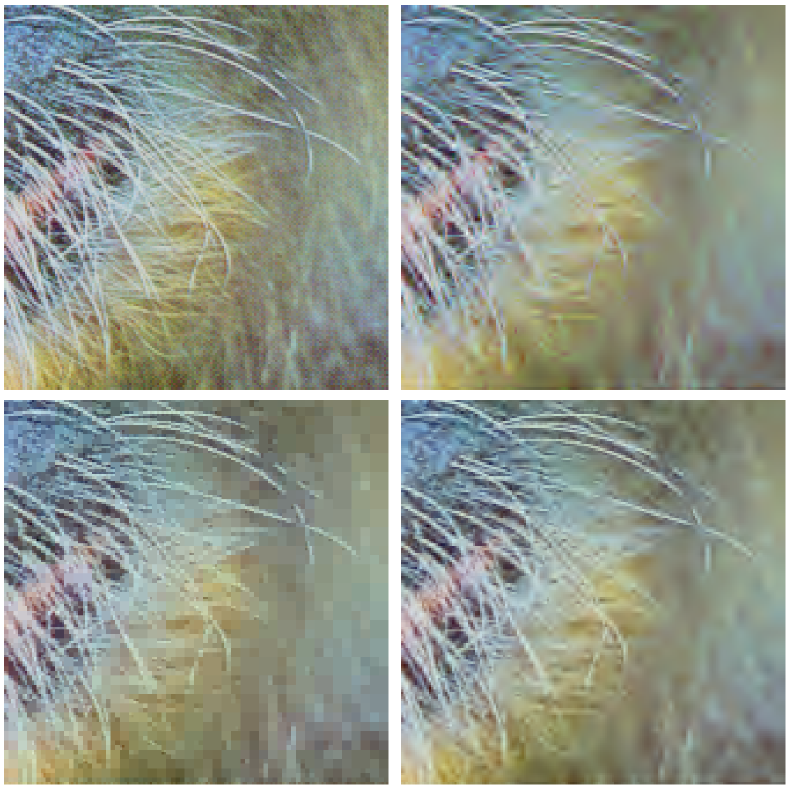
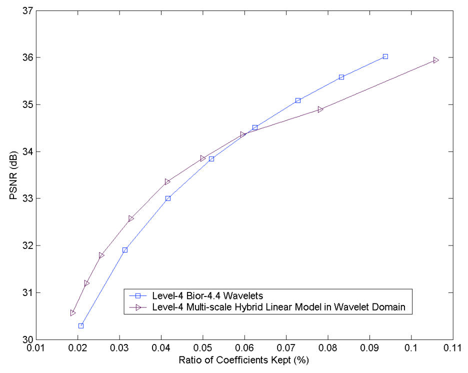


Figure 6.22. Enlarged bottom-right corner of the images in Figure 6.21. Top-left: The original image. Top-right: The level-3 biorthogonal 4.4 wavelets. Bottom-left: The level-3 multi-scale hybrid linear model in the spatial domain. Bottom-right: the level-3 multi-scale hybrid linear model in the wavelet domain.

to demonstrate (using arguably the simplest cases) the vast potential of a new spectrum of image representations suggested by combining subspace methods with conventional image representation/approximation schemes. The quest for the more efficient and more compact representations for natural images without doubt will continue as long as the nature of natural images remains a mystery and the mathematical models that we use to represent and approximate images improve.

6.4 Bibliographic Notes

There is a vast amount of literature on finding adaptive bases (or transforms) for signals. Adaptive wavelet transform and adapted wavelet packets have been extensively studied [Coifman and Wickerhauser, 1992, Ramchandran et al., 1996, Meyer, 2002, Meyer, 2000, Ramchandran and Vetterli, 1993, Delsarte et al., 1992, Pavlovic et al., 1998]. The idea is to search for an optimal transform from


PSNR (dB)
Ratio of Coefficients Kept (%)
Level-4 Bior-4.4 Wavelets
Level-4 Multi-scale Hybrid Linear Model in Wavelet Domain

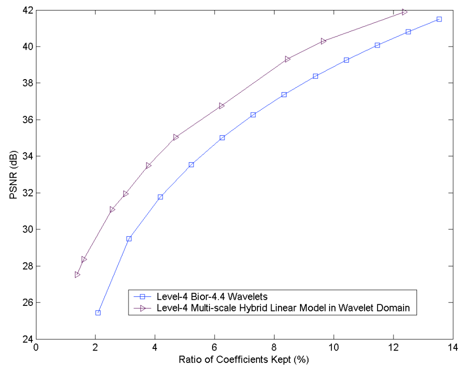

PSNR (dB)
Ratio of Coefficients Kept (%)
Level-4 Bior-4.4 Wavelets
Level-4 Multi-scale Hybrid Linear Model in Wavelet Domain

Figure 6.23. Top: Comparison of multi-scale hybrid linear model in the wavelet domain with wavelets for the Lena image. Bottom: Comparison of multi-scale hybrid linear model in the wavelet domain with wavelets for the Monarch image. The multi-scale hybrid linear model in the wavelet domain achieves better PSNR than wavelets for a wide range of PSNR for these two images.

a limited (although large) set of possible transforms. Another approach is to find some universal optimal transform based on the signals [**?**,Rabiee et al., 1996, Delsarte et al., 1992, Pavlovic et al., 1998]. Spatially adapted bases have also been developed such as [Chen et al., 2003, Sikora and Makai, 1995, Muresan and Parks, 2003].

The notion of hybrid linear model for image representation is also closely related to the *sparse component analysis*. In [Olshausen and D.J.Field, 1996], the authors have identified a set of non-orthogonal base vectors for natural images such that the representation of the image is sparse (i.e., only a few components are needed to represent each image block). In the work of [Donoho and Elad, 2002, Donoho, 1995, Donoho, 1998, Chen et al., 1998, Elad and Bruckstein, 2002, Feuer and Nemirovski, 2003, Donoho and Elad, 2003, Starck et al., 2003, Elad and Bruckstein, 2001], the main goal is to find a mixture of models such that the signals can be decomposed into multiple models and the overall representation of the signals is sparse.

Chapter 7
Image Segmentation

Natural-image segmentation is one of the classical problems in computer vision. It is widely accepted that a good segmentation should group image pixels into regions whose statistical characteristics (of the color or texture) are homogeneous or stationary, and whose boundaries are simple and spatially accurate [**?**]. Nevertheless, from a statistical viewpoint, natural-image segmentation is an *inherently ambiguous* problem for at least the following two technical reasons[1]:

1. The statistical characteristics of local features (e.g., color, texture, edge, contour) of natural images usually do not show the same level of homogeneity or saliency at the same spatial or quantization scale. This is not only the case for different natural images, but also often the case for different regions within the same image. Thus, one should not expect the segmentation result to be unique [**?**], and instead should seek a hierarchy of segmentations at multiple scales.

2. Even after accounting for variations due to scale, different regions or textures may still have different intrinsic complexities, making it a difficult statistical problem to determine the correct number of segments and their model dimensions. For instance, if we use Gaussian distributions to model the features of different textures, the Gaussian for a simple texture obviously has a higher degree of degeneracy (or a lower dimension) than that for a complex texture.

[1]It is arguably true that human perception of an image is itself ambiguous. However, here we are only concerned with ambiguities in computational image segmentation.

In the literature, many statistical models and methods, supervised or unsupervised, have been proposed to address some of these difficulties. Here we are mainly interested in *unsupervised* image segmentation. Popular methods in this category include *feature-based* Mean-Shift [**?**], *graph-based* methods [**?**, **?**], *region-based* split-and-merge techniques [**?**, **?**], and global optimization approaches based on either energy functions [**?**] or minimum description length [**?**].

Although the reported performance of image segmentation algorithms has improved significantly over the years, these improvements have come partly at the price of ever more sophisticated feature selection processes, more complex statistical models, and more costly optimization techniques. In this chapter, however, we aim to show that using texture features as simple as fixed-size Gaussian windows, with the choice of a likely more relevant class of statistical models and the associated clustering algorithm, one can achieve equally good, if not better, segmentation results as many of the sophisticated statistical models and optimization methods.

7.1 Image Segmentation as a GPCA Problem

Extant image segmentation methods segment features based on their cluster centers (e.g., K-Means) or density modes (e.g., Mean-Shift). They typically work well for low-level segmentation using low-dimensional features that have blob-like distributions (as shown in Figure 7.1 left). But for mid-level segmentation

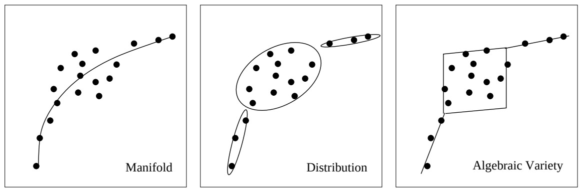


Figure 7.1. Mixture of regular (left) or degenerate (right) Gaussians.

using texture features extracted at a larger spatial scale, we normally choose a feature space whose dimension is high enough that the structures of all textures in the image can be *genuinely represented*.[2] Such a representation unavoidably has redundancy for individual textures: The cluster of features associated with one texture typically lies in a low-dimensional submanifold or subspace whose dimension reflects the complexity of the texture (Figure 7.1 right). Thus, the distribution of texture features in a natural image can be well modeled as a mixture of almost degenerate Gaussians or subspaces that may have different dimensions

[2]Here a genuine representation means that we can recover every texture with sufficient accuracy from the representation.

(see Figure 7.1 right), one for each image segment. Properly harnessed, such low-dimensional structures can be much more informative for distinguishing textures than the cluster means.

In chapter 5, we have revealed a fundamental relationship between data clustering and compression. We have derived an effective clustering algorithm for mixtures of subspaces. By minimizing the overall coding length of the given mixed data subject to a given distortion, the algorithm automatically merges the data points into a number of subspace-like clusters. Thus, we propose to measure the "goodness" of image segmentation in terms of a coding-theoretic criterion: the *optimal* segmentation is the one that minimizes the coding length of the features. It is debatable whether this is how humans segment images. Coding length is an objective measure while human segmentation is highly subjective – much prior knowledge is incorporated in the process. Later we will quantitatively evaluate the extent to which the segmentation results emulate those of humans, in fair comparison with other unsupervised image segmentation techniques. Be aware that, although we here have adopted the data compression paradigm for image segmentation, it is different from compressing the image *per se*. Instead of pixel values, we compress and segment texture features extracted from the image values.

The lossy data compression paradigm of chapter 5 has another attractive feature for image segmentation. Varying the distortion provides a simple but effective means of considering textural information at different *quantization* scales.[3] Compressing the image features with different distortions, we naturally obtain a hierarchy of segmentations: the smaller the distortion, the more refined the segmentation is (see Figure 7.6 for an example). In a way, the distortion also plays an important role in image segmentation as a measure of the *saliency* of the segments in an image: First, how small the distortion needs to be in order for certain regions to be segmented from the background, and second, how much we can change the distortion without significantly altering the segmentation (see Figure 7.6 again). Thus, lossy compression offers a convenient framework for diagnosing the statistics of a natural image at different quantization scales.

This chapter is organized as follows: Section 7.2 discusses in detail how to customize the lossy compression-based clustering algorithm of chapter 5 for image segmentation. Particularly, we discuss how to adaptively select the distortion threshold to achieve good segmentation. Section 7.3 gives experimental results on the Berkeley segmentation database, and compares to other existing algorithms and human segmentation. The software package is available at the website: `http://www.eecs.berkeley.edu/~yang/lossy_segmentation/`.

[3]However, we do not consider varying spatial scale as we will always choose a fixed-size window as the feature vector. Nevertheless, as we will demonstrate, excellent segmentation can already be obtained.

7.2 Image Segmentation via Lossy Compression

In this section, we describe how the lossy compression-based method in Section 5.2 is applied to segment natural images. We first discuss what features we use to represent textures and why. We then describe how a *low-level segmentation* is applied to partition an image into many small homogeneous patches, known as *superpixels*. The superpixels are used to initialize the *mid-level texture-based segmentation*, which minimizes the total coding length of all the texture features by repeatedly merging adjacent segments, subject to a distortion ε^2. Finally, we study several simple heuristics for choosing a good ε for each image.

7.2.1 Constructing Feature Vectors

We choose to represent a 3-channel RGB color image in terms of the $L^*a^*b^*$ color metric, which was specially designed to best approximate perceptually uniform color spaces.[4] While the dependence of the three coordinates on the traditional RGB metric is nonlinear [?], the $L^*a^*b^*$ metric better facilitates representing texture via mixtures of Gaussians. Perceptual uniformity renders the allowable distortion ε^2 meaningful in terms of human perception of color differences, tightening the link between lossy coding and our intuitive notion of image segmentation.

In the literature, there have been two major types of features used to capture local textures. The first type considers responses of a 2D filter bank as texture features [**?**, **?**]. The second directly uses a $w \times w$ cut-off window around each pixel and stacks the color values inside the window into a vector [**?**, **?**]. Each texture window is usually smoothed by convolving with a 2D Gaussian kernel before stacking. Figure 7.2 illustrates this process.

Figure 7.2. The construction of texture features: A $w \times w$ window of each of the three $L^*a^*b^*$ channels is convoluted with a Gaussian and then all channels are stacked into a single vector v.

We have experimented with using simple window features, as well as two classical filter banks, the Leung-Malik set [?] and the Schmid set [**?**], in conjunction with our clustering algorithm. We found the difference in the segmentation result is small despite the fact that filter-bank features are more computationally expensive as they involve convolutions of the image with large number of filters. One likely reason for the similar performance is that the compression-base clustering algorithm is capable of automatically harnessing the low-dimensional linear structures of the features, despite noise and outliers (see Figure **??** and additional evidence in [**?**]).

[4]Equivalently, one can also use the $L^*u^*v^*$ metric.

For simplicity, we here choose to use the window features. We find that a 7×7 window provides satisfactory results, although other similar sizes also work well.[5] Finally, to reduce the computational cost, we project the feature vectors into an 8-dimensional space by PCA. This operation preserves all linear structures of dimension less than 8 in the feature space. Experimentally, we found an 8-dimensional space to be sufficient for most natural image textures.

7.2.2 *Initialization with Superpixels*

Given the feature vectors extracted from an image, one "naive" approach would be to directly apply Algorithm 5.1, and segment the pixels based on the grouping of the feature vectors. Figure 7.3 shows one such result. Notice that the resulting segmentation merges pixels near the strong edges into a single segment. This should be expected from the compression perspective, since windows across the boundary of two segments have significantly different structures from the (homogeneous) textures within those segments [?]. However, such a segmentation does not agree well with human perception.

Figure 7.3. Two segmentation results of the left original using Algorithm 5.1 with different ε's. Notice that the pixels near the boundaries of segments are not grouped correctly.

In order to group edge pixels appropriately, we preprocess an image with a low-level segmentation based on local cues such as color and edges. That is, we oversegment the image into (usually several hundred) small, homogeneous regions, known as *superpixels*. This preprocessing step has been generally recommended for all region merging algorithms in [?]. Such low-level segmentation can be effectively computed using K-Means or Normalized-Cuts (NCuts) [?] with a conservative homogeneity threshold. Here we use the publicly available superpixel code [**?**].

[5] We did not test window sizes larger than 9 pixels, as the current MATLAB implementation cannot store all such texture vectors from a typical 320×240 color image. However, this problem can be alleviated by sampling a subset of the texture features from an image.

Since the superpixel segmentation respects strong edges in an image (see Figure 7.4 middle), it does not suffer from the misassignment of edge pixels seen in Figure 7.3. All feature vectors associated with pixels in each superpixel are initialized as one segment, forcing the subsequent merging process to group boundary pixels together with the interior pixels. An additional benefit from the superpixel preprocessing is a significant reduction in the computational cost. Using superpixel segments as initial grouping, the algorithm only needs to search amongst several hundred of superpixels for the optimal pair to merge, instead of searching amongst all feature vectors (the number of vectors is on the order of tens of thousands).

To further reduce the computational cost, one may consider using only a portion of the feature vectors associated with each superpixel. For instance, feature vectors at the boundary of a superpixel represent a combination of textures from two adjacent superpixels, and their distribution can be rather complicated compared to the distribution of the feature vectors in the interior of the superpixel. Thus, one may use only feature vectors from the interior of each superpixel.[6] Our experiments show that, under the same distortion parameter ε, this modification tends to partition an image into smaller texture segments. This phenomenon will be discussed in more detail in Section 7.3.3. For clarity, all segmentation results presented in this chapter will use both interior and boundary feature vectors of every superpixel unless stated otherwise.

7.2.3 Enforcing Connected Segments

Notice that in the definition of the overall coding length function (**??**), we use the Huffman coding length to upper bound the number of bits required to encode the membership of the feature vectors. This obviously over-estimates the coding length since it does not take into account the fact that in natural images, adjacent pixels have higher probability that belong to the same segment.

In order to enforce that the resulting segmentation respects spatial continuity and consists of only connected segments, we impose an additional constraint that two segments S_i and S_j can be merged together only if they are spatially adjacent in the 2D image. To this end, we need to construct and maintain a *region adjacency graph* (RAG) G in the clustering process. RAG is popularly used in other *merge-and-split* type segmentation methods [**?**]. We represent the RAG using an adjacency list $G\{i\}$ for each segment S_i. Index j is in the set $G\{i\}$ if the segment S_j is a neighbor of S_i. At each iteration, the algorithm searches for a pair of adjacent segments S_i and S_j which lead to maximal decrease in the total coding length. Note, however, that in some applications such as image compression, disconnected regions may be allowed to be grouped as the same segment. In this case, one can simply discard the adjacency constraint in our implementation.

[6]If a superpixel only consists of boundary pixels, these pixels are used anyway.

Figure 7.4 shows an example of the two-step segmentation process. For this image, we find that all feature vectors approximately lie in a 6D subspace in the 8D feature space (i.e. the first 8 principal components of the Gaussian windows). Furthermore, feature vectors of each segment can be well modeled as a 1D to 4D subspace. Figure 7.5 plots the singluar values of two representative segments. This validates our initial assumption that the distributions of texture features are typically (close to) degenerate.

Figure 7.4. The segmentation pipeline. Left: Original. Middle: Superpixels obtained from low-level over-segmentation. Right: Segments obtained by minimizing the coding length with $\varepsilon = 0.2$.

Figure 7.5. Singluar values of the feature vectors drawn respectively from two image segments in Figure 7.4 right: one is on the women's cloth and the other is the background.

7.2.4 Choosing the Distortion

As discussed in the introduction, the distortion ε effectively sets the quantization scale at which we segment an image. Figure 7.6 shows the segmentation of several images under different values of ε. As the figure suggests, a single ε will not give good performance across a widely varying data set such as the Berkeley image segmentation database. Differences in the contrast of the foreground and background, lighting conditions, and image category cause the distribution of the texture features to vary significantly from image to image

Figure 7.6. Segmentation results under different ε. Left: Originals. Middle left: $\varepsilon = 0.1$. Middle right: $\varepsilon = 0.2$. Right: $\varepsilon = 0.4$.

There are several ways to adaptively choose ε to achieve good segmentation for each image. For example, if a desired number of segments is known *a priori*, we can search a range of ε values for the one that gives the desired number of segments. When such information is not available *a priori*, as is the case for image segmentation, a formal way in information theory to estimate the distortion parameter is to minimize a cost function such as the following one:

$$\varepsilon^* \doteq \arg\min_{\varepsilon \in \mathcal{E}} \{L^s(V, \varepsilon) + \lambda N D \log_2(\varepsilon)\}, \tag{7.1}$$

where λ is a parameter provided by the user that weights the two terms. Notice that the first term $L^s(V, \varepsilon)$ decreases as ε increases, as opposed to the second term $ND\log_2(\varepsilon)$. Hence, the expression essentially seeks a balance between the coding length of the data and the complexity of the model measured as $ND\log_2(\varepsilon)$. It is studied in [?] that (7.1) can accurately recover the true value of ε for the simulated Gaussian mixture models by simply setting $\lambda = 1$. However, when applied to image segmentation on real natural images, the so-estimated ε^* tends to over-segment the images. One reason for this discrepancy between simulation and experiment is that the noise associated with different texture segments can have different covariance.

Here, we choose to adaptively select the distortion ε by stipulating that feature distributions in adjacent texture regions must be sufficiently dissimilar. In the literature, the similarity measure between two texture distributions has been extensively studied. In information theory, the *Kullback-Leibler* (KL) divergence measures the relative entropy between two arbitrary distribution functions $p(x)$ and $q(x)$ [Cover and Thomas, 1991]:

$$d_{KL} = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{7.2}$$

However, the KL divergence is ill-posed for distributions functions $p(x)$ and $q(x)$ that have different supports, where $q(x)$ may be equal to zero as the denominator in the log function. Unfortunately, this is often the case to compare two degenerate distributions (e.g., , texture vectors from images).

In computer vision, the heuristic Earth Mover's Distance (EMD) is a metric to measure the similarity of two image distributions [**?**, **?**]. Levina and Bickel [**?**] further show that EMD is equivalent to the Mallows distance in statistics, which has a closed-form expression for two Gaussian distributions $N(\theta_1, \Sigma_1)$ and $N(\theta_2, \Sigma_2)$ [**?**]:

$$d_M(N(\theta_1, \Sigma_1), N(\theta_2, \Sigma_2))^2 = (\theta_1 - \theta_2)^T(\theta_1 - \theta_2) + \boldsymbol{tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{\frac{1}{2}}). \tag{7.3}$$

Finally, as a reasonable approximation to the Mallows distance, one can measure the similarity of $N(\theta_1, \Sigma_1)$ and $N(\theta_2, \Sigma_2)$ using their mean vectors:

$$d_m(N(\theta_1, \Sigma_1), N(\theta_2, \Sigma_2))^2 = (\theta_1 - \theta_2)^T(\theta_1 - \theta_2). \tag{7.4}$$

In our experiment, we tested both the Mallows distance d_M and the mean distance d_m to measure the similarity between texture segments. For a given ε, the minimal distance $d(\varepsilon)$ of an image is calculated between all pairs of adjacent segments after the compression-based merging. The selection process gradually increases the value of ε (from a list of candidate values) until the minimal distance $d(\varepsilon)$ is larger than a preselected threshold γ:

$$\varepsilon^* = \min\{\varepsilon : d(\varepsilon) \geq \gamma\}. \tag{7.5}$$

The final segmentation result then gives the most refined segmentation which satisfies the above constraint. We note that increasing ε typically causes the number of segments to decrease and results in a shorter coding length. We may therefore use the segmentation computed with a smaller ε to initialize the merging process with a larger ε, allowing us to search for the optimal ε more efficiently. The experiment shows that both d_M and d_m give very similar segmentation results.

It may seem that we have merely replaced one free parameter, ε, with another, γ. This replacement has two strong advantages, however. Experimentally we find that even with a single fixed value of γ the algorithm can effectively adapt to all image categories in the Berkeley database, and achieve segmentation results that are consistent with human perception. Furthermore, the appropriate γ can be estimated empirically from human segmentations, whereas ε cannot. This heuristic thresholding method is similar in spirit to several robust techniques in computer vision for estimating mixture models, e.g., , the Hough transform and RANSAC.

The complete segmentation process is specified as Algorithm 7.1. In terms of speed, on a typical 3GHz Intel PC, the MATLAB implementation of the CTM algorithm on a 320×240 color image takes about two minutes to preprocess superpixels, and less than one minute to minimize the coding length of the features.

7.3 Experiments

In this section, we demonstrate the segmentation results of Algorithm 7.1 (CTM) on natural images in the Berkeley segmentation database [**?**], which also contains benchmark segmentation results obtained from human subjects.

7.3.1 Visual Verification

We first *visually* verify the segmentation results on the Berkeley database. Representative segmentation results of the CTM algorithm with $\gamma = 0.1$ and $\gamma = 0.2$ are shown in Figures 7.8 – 7.13. For better visual evaluation, we have partitioned the database into six different image categories, each of which consists of images that are more relevant, namely, *Landscape* (Figure 7.8), *Ocean* (Figure 7.9), *Urban* (Figure 7.10), *Animals* (Figure 7.11), *People* (Figure 7.12), and *Objects* (Figure 7.13). Smaller γ's tend to generate more segments and oversegment the images, and larger γ's tend to generate less segments and hence undersegment the images. The benchmarking MATLAB script and the complete

Algorithm 7.1 (CTM: Compression-based Texture Merging).

```
Input: Image $I \in \mathbb{R}^{H \times W \times 3}$ in $L^*a^*b^*$ metric, reduced dimension $D$, window size $w$, distortion range $\mathcal{E}$, and minimum mean distance $\gamma$.
1: Partition $I$ into superpixels $S_1, \ldots, S_K$. For pixel $p_i \in S_j$, initialize its label $l_i = j$.
2: Construct RAG $G\{1\}, \ldots, G\{K\}$ for the $K$ segments $S_1, \ldots, S_K$.
3: Sample $w \times w$ windows, and stack the resulting values into a feature vector $v_i \in \mathbb{R}^{3w^2}$.
4: Replace $v_i$ with their first $D$ principal components.
5: for all $\varepsilon \in \mathcal{E}$ in ascending order do
6:   for all initial segments $S_i$, $i = 1, \ldots, K$ do
7:     Compute $L^s(S_i, \varepsilon)$.
8:     for all $j \in G\{i\}$ do
9:       $U_{ij} \doteq L^s(S_i, \varepsilon) + L^s(S_j, \varepsilon) - L^s(S_i \cup S_j, \varepsilon)$
10:    end for
11:  end for
12:  while $U_{ij} \doteq \max\{U\} > 0$ do
13:    Merge $S_i$ and $S_j$. Update arrays $l$, $G$, $L$, and $U$.
14:    Segment number $K \leftarrow K - 1$.
15:  end while
16:  if $\gamma \leq \min_{i,j \in G(i)}\{d(S_i, S_j, \varepsilon)\}$ then
17:    break.
18:  end if
19: end for
Output: Final pixel labels $l_1, \ldots, l_{H \times W}$.
```

visualization results of the Berkeley image segmentation database are also available for download at our website: http://www.eecs.berkeley.edu/~yang/lossy_segmentation/.

7.3.2 *Quantitative Verification*

We now compare *quantitatively* CTM against three unsupervised algorithms that have been made available publicly: Mean-Shift [?], NCuts [?], and Felzenszwalb and Huttenlocker (FH) [?]. The comparison is based on four quantitative performance measures:

1. The Probabilistic Rand Index (PRI) [**?**] counts the fraction of pairs of pixels whose labellings are consistent between the computed segmentation and the ground truth, averaging across multiple ground truth segmentations to account for scale variation in human perception.

2. The Variation of Information (VoI) metric [**?**] defines the distance between two segmentations as the average conditional entropy of one segmentation

given the other, and thus roughly measures the amount of randomness in one segmentation which cannot be explained by the other.

3. The Global Consistency Error (GCE) [?] measures the extent to which one segmentation can be viewed as a refinement of the other. Segmentations which are related in this manner are considered to be consistent, since they could represent the same natural image segmented at different scales.

4. The Boundary Displacement Error (BDE) [**?**] measures the average displacement error of boundary pixels between two segmented images. Particularly, it defines the error of one boundary pixel as the distance between the pixel and the closest pixel in the other boundary image.

Since all methods are unsupervised, we use both the training and testing images for the evaluation. Due to memory issues with the NCuts implementation in MATLAB, all images are normalized to have the longest side equal to 320 pixels. We ran Mean-Shift [?] with parameter settings (h_s, h_r) chosen at regular intervals of $[7, 16] \times [3, 23]$, and found that on the Berkeley database, $(h_s, h_r) = (13, 19)$ gives a good overall tradeoff between the above quantitative measures. We therefore use this parameter choice for our comparison. For NCuts [?], we choose the number of segments $K = 20$ to agree with the average number of segments from the human subjects. For the FH algorithm, we choose the Gaussian smoothing parameter $\sigma = 0.5$, the threshold value $k = 500$, and the minimal region size to be 200 pixels, as suggested by the authors [?].

Table 7.1 gives the quantitative comparion of CTM against the other three algorithms on the Berkeley segmentation benchmark. In the experiment, three γ values were tested for CTM, namely, $\gamma = 0.1, 0.15, 0.2$. The mean distance d_m defined in equation (7.4) was used to measure the texture similarity between adjacent segments. The distortion range $\mathcal{E}$ for the ε value was between 0.01 and 0.5, which are relative scales in terms of normalized texture vectors.

Table 7.1. Average performance on the Berkeley Database (bold indicates best of all the algorithms). PRI ranges between $[0, 1]$, higher is better. VoI ranges between $[0, \infty)$, lower is better. GCE ranges between $[0, 1]$, lower is better. BDE ranges between $[0, \infty)$ in the unit of pixel, lower is better.

	PRI	VoI	GCE	BDE
Humans	0.8754	1.1040	0.0797	4.994
CTM$_{\gamma=0.1}$	0.7561	2.4640	**0.1767**	**9.4211**
CTM$_{\gamma=0.15}$	0.7627	2.2035	0.1846	9.4902
CTM$_{\gamma=0.2}$	0.7617	**2.0236**	0.1877	9.8962
Mean-Shift	0.7550	2.477	0.2598	9.7001
NCuts	0.7229	2.9329	0.2182	9.6038
FH	**0.7841**	2.6647	0.1895	9.9497

Table 7.1 shows that quantitatively, CTM outperforms Mean-Shift, NCuts, and FH in terms of most indices: At $\gamma = 0.15$, CTM is better than Mean-Shift and

NCuts in terms of all four indices; and for all chosen γ's, CTM is better than FH except for the PRI index. It is perhaps not surprising that CTM significantly outperforms the other three algorithms in terms of the VoI index, since we are optimizing an information-theoretic criterion. These numbers show that minimizing the coding length leads to segmentation that is closer to human segmentation. This suggests that perhaps human perception also approximately minimizes some measure of the compactness of the representation.

One may also interpret the results in terms of the differences among the four segmentation indices. The GCE and BDE indices, penalize under-segmentation more heavily than over-segmentation. In particular, GCE does not penalize over-segmentation at all, i.e., , the highest score is achieved by assigning each pixel as an individual segment. As a result, CTM$_{\gamma=0.01}$ has returned the best GCE and BDE values among all the results in Table 7.1, but its VoI value is one of the worst in the table. From our experience (also shown in Figures 7.8 – 7.13), PRI and VoI seem to be more correlated with human segmentation in term of visual perception.

To summarize both visual and quantitative comparison, we notice that on one hand, if we tune the algorithms to give the visually best match with human segmentation, none of the algorithms or parameters is a clear winner in terms of all four indices; on the other hand, none of the indices seems to be a better indicator of human segmentation than others, which suggests that human segmentation uses much more comprehensive cues. Nevertheless, the extensive visual demonstration and quantitative comparison does serve to validate our hypotheses that the distribution of texture features of natural images can be well approximated by a mixture of (possibly degenerate) Gaussians. As a result, the compression-based clustering algorithm becomes a powerful tool that exploits the redundancy and degeneracy of the distribution for good texture segmentation.

7.3.3 Difficulties and Possible Extensions

To fairly assess an image segmentation algorithm, we also need to investigate examples for which the algorithm has failed to produce good results. In this subsection, we will show a handful of such examples from the Berkeley database, and discuss several possible extensions to the CTM algorithm to further improve the segmentation results.

A particular category that CTM has trouble with is a set of images of animals with very severe camouflage. Figure 7.7 shows some representative examples. For these examples, it is difficult for CTM to segment the animal from the background even with very small distortion ε. Comparing with Figure 7.6, human figures often endure a larger ε, as human complexion and clothes stand out from the (man-made) surroundings. Thus, in a way, the distortion ε can be interpreted as a measure for how "salient" an object is in an image and how much "attention" is needed to segment the object. The effectiveness of certain camouflage can be measured by the coding length of the texture in question together with the texture of the background.

(a) Orig- i- nal im- ages

(b) Seg- men- ta- tion re- sults with CTM$_{+,\gamma=0.1}$

(c) Seg- men- ta- tion re- sults with CTM$_{-,\gamma=0.1}$

Figure 7.7. Segmentation results on certain animal images. CTM$_{+}$ represents the CTM algorithm applied to all texture vectors including those at the boundaries. CTM$_{-}$ represents the same algorithm without sampling the texture vectors at the boundaries.

In order to extract severely camouflaged animals from their surroundings, a straightforward extension of the CTM algorithm is to exclude texture vectors at the boundaries of the superpixels. A texture vector, say the Gaussian window, at the boundary contains pixels from the two adjacent superpixels that share the common boundary. By excluding these texture vectors, the set of texture vectors from the superpixel become more homogeneous. Hence, the compression-based algorithm can more effectively distinguish the texture of the animal from that of the background. This variation of CTM is denoted as CTM$_{-}$ while the original version is denoted as CTM$_{+}$. Figure 7.7 demonstrates the improvement of the CTM$_{-}$ algorithm on these images. But notice that it still failed to segment out the body of the crocodile from the background; in this case the camouflage is effective enough to fool even human eyes.

We also observe another limitation of CTM from the results in Figure 7.7. As an example, for the second image of Figure 7.7, the algorithm needs to use a relatively small ε to extract the segment of the leopard from the background. Nevertheless, under the the same ε, the background textures are oversegmented. At a fixed γ, the CTM algorithm searches for the best distortion parameter ε value to code the feature vectors of the entire image, despite the fact that these textures may have different noise variances.

A possible solution to this problem is to assign different ε values to different image regions. Given a set of training images that are segmented by a human subject, one can learn the distribution of ε of all the textures. Then, given a new image, one needs to infer the appropriate ε to use for different regions in a Bayesian fashion.

Such an extension may give more relevant segmentation results for several important applications, such as salient object detection. For instance, saliency is arguably a subjective notion, as people have their own preference to which region of an image is the most salient one. We have shown through extensive experiments in this chapter that whether an image region can be easily segmented out from its surroundings is closely related to the distortion allowed in the lossy coding. Therefore, it is possible to learn a compression-based saliency detector through a set of examples. The segmentation results will most likely resemble the results of the individual human subject who has provided the training examples.

7.4 Bibliographic Notes

In this chapter, we have proposed that texture features of a natural image should be modeled as a mixture of possibly degenerate distributions. We have introduced a lossy compression-based clustering algorithm, which is particularly effective for segmenting degenerate Gaussian distributions. We have shown that the algorithm can be customized to successfully segment natural images by harnessing the natural low-dimensional structures that are present in raw texture features such as Gaussian windows.

In addition, the lossy compression-based approach allows us to introduce the distortion as a useful parameter so that we can obtain a hierarchy of segmentations of an image at multiple quantization scales. We have proposed a simple heuristic criterion to adaptively determine the distortion for each image if one wants to match the segmentation with that of humans.

In this chapter, we have studied only unsupervised segmentation of natural images. However, the proposed framework can also be extended to supervised scenarios. We believe that it is of great importance better understand how humans segment natural images from the lossy data compression perspective. Such an understanding would lead to new insights into a wide range of important problems in computer vision such as salient object detection and segmentation, perceptual organization, and image understanding and annotation. These are some of the challenging problems left open for future investigation.

One may refer to [**?**] for a review on the topic of image segmentation. Recent developments in image segmentation have mainly focused on the problem of how to integrate textural information at different scales. For example, one can use more sophisticated *region-growing* or *split-and-merge* techniques [?, **?**, **?**, **?**] to partition inhomogeneous regions; or one can use *Markov random fields* to model textures or other image cues [**?**, ?, ?]. For a more detailed survey of these methods, the reader is referred to [?, ?, ?].

Our method bears resemblance to some global optimization approaches, such as using region merging techniques to minimize the MDL cost function [?].

Image Segmentation.

Image segmentation based on local color and texture information extracted from various filter banks has been studied extensively in the computer vision literature (see e.g., [**?**, **?**, **?**]). In this chapter, we directly used the unfiltered pixel values of the image. Our segmentation is a byproduct of the global fitting of a hybrid linear model for the entire image. Since the image compression standard JPEG-2000 and the video compression standard MPEG-4 have started to incorporate texture segmentation [**?**], we expect that the method introduced in this chapter will be useful for developing new image processing techniques that can be beneficial to these new standards.

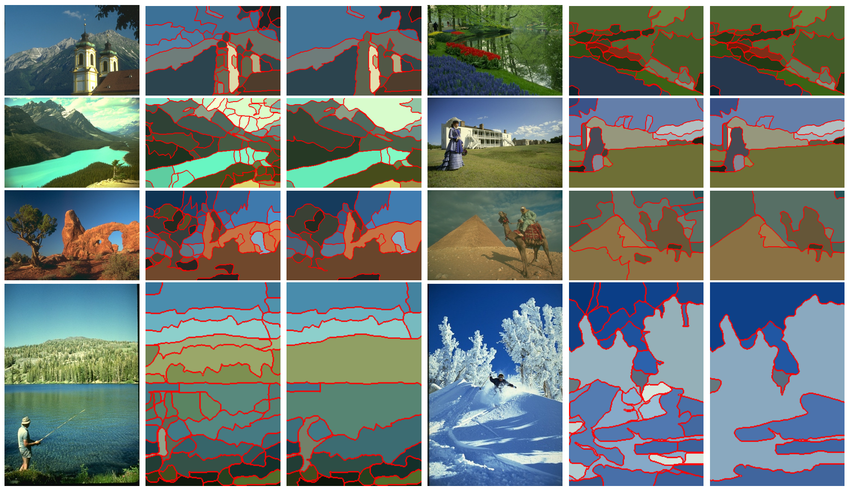


Figure 7.8. Examples in Category Landscape. Left: Original. Middle: CTM$_{\gamma=0.1}$. Right: CTM$_{\gamma=0.2}$.

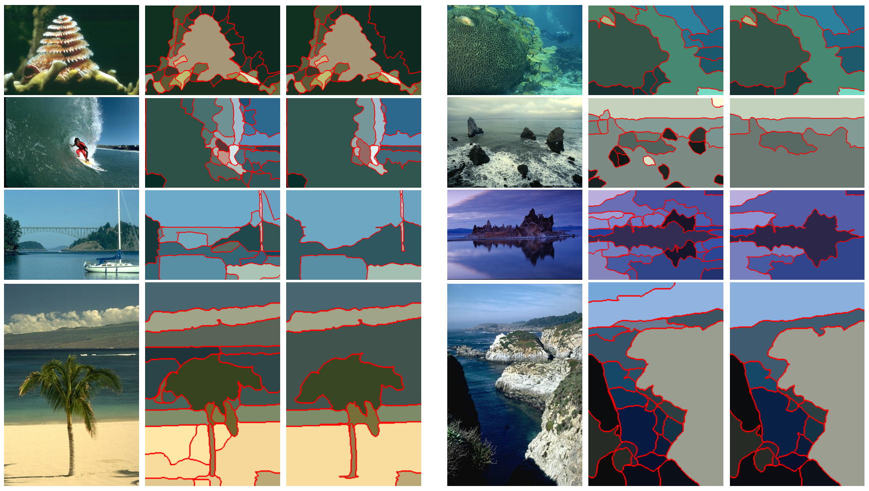


Figure 7.9. Examples in Category Ocean. Left: Original. Middle: CTM$_{\gamma=0.1}$. Right: CTM$_{\gamma=0.2}$.

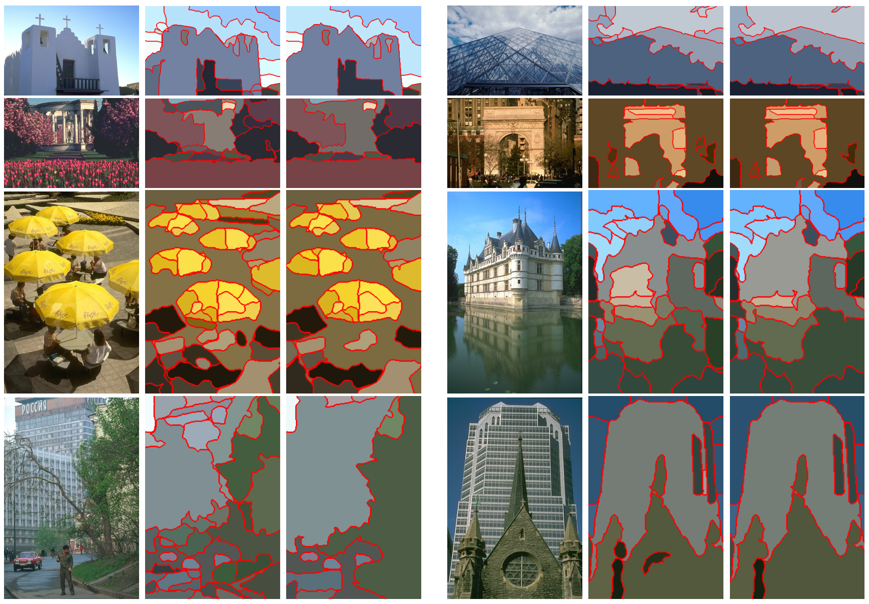


Figure 7.10. Examples in Category Urban. Left: Original. Middle: CTM$_{\gamma=0.1}$. Right: CTM$_{\gamma=0.2}$.

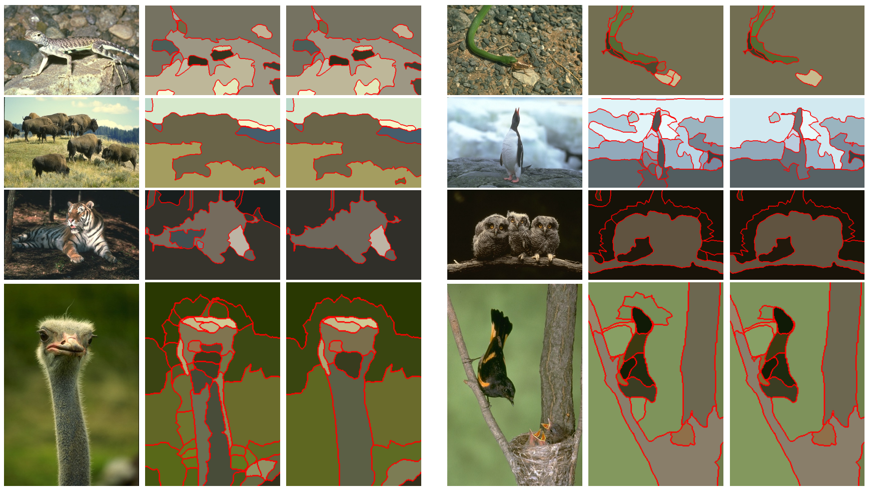


Figure 7.11. Examples in Category Animals. Left: Original. Middle: CTM$_{\gamma=0.1}$. Right: CTM$_{\gamma=0.2}$.

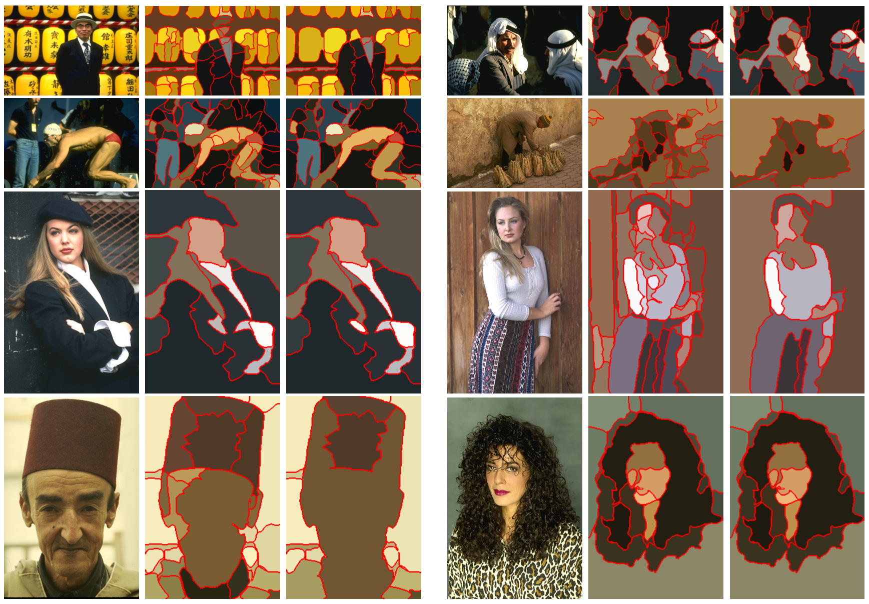


Figure 7.12. Examples in Category People. Left: Original. Middle: CTM$_{\gamma=0.1}$. Right: CTM$_{\gamma=0.2}$.

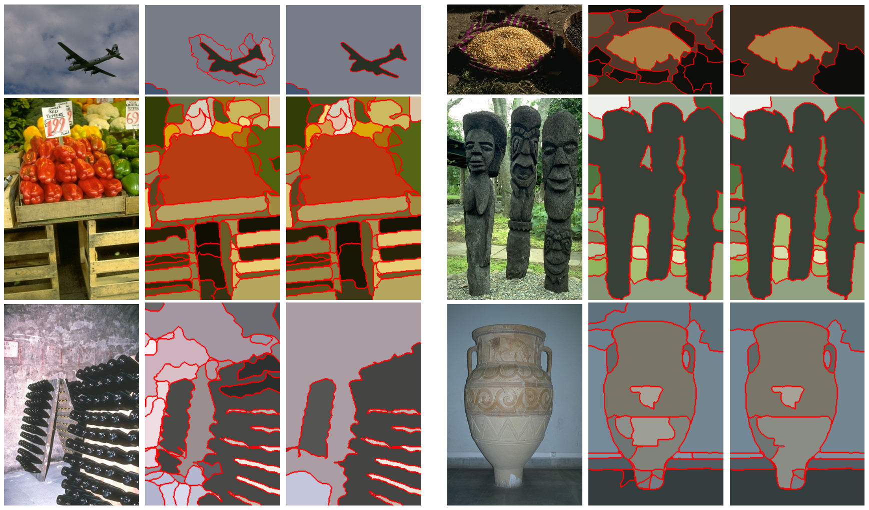


Figure 7.13. Examples in Category Objects. Left: Original. Middle: CTM$_{\gamma=0.1}$. Right: CTM$_{\gamma=0.2}$.

Chapter 8

3-D Motion Segmentation from Point Correspondences

A classic problem in visual motion analysis is to estimate a motion model for a set of 2-D feature points as they move in a video sequence. When the scene is *static*, i.e., when either the camera or a single object move, the problem of fitting a 3-D model compatible with the structure and motion of the scene is well understood [**?**, ?]. For instance, it is well-known that two perspective views of a scene are related by the *epipolar constraint* [**?**] and that multiple views are related by the *multilinear constraints* [**?**]. These constraints can be used to estimate a motion model for the scene using linear techniques such as the eight-point algorithm and its generalizations.

However, these techniques can not deal with *dynamic scenes* in which both the camera and an unknown number of objects with unknown 3-D structure move independently. In principle, one could model such scenes with a collection of 2-D motion models and segment them using the 2-D motion segmentation techniques developed in the previous chapter. However, because of depth discontinuities, perspective effects, etc, 2-D techniques would tend to interpret a single 3-D motion as multiple 2-D motions, which would result in over segmentation of the scene.

In this chapter, we develop techniques for segmentation of 3-D motion models. In particular, we consider the segmentation of three types of models of increasing complexity: linear, bilinear and trilinear. The segmentation of linear models shows up in motion segmentation from multiple affine views, and can be solved using the GPCA algorithm presented in Chapter 3. The segmentation of bilinear and trilinear models shows up in motion segmentation from point correspondences in two and three perspective views, respectively, and will require the development of extensions of GPCA to certain classes of bilinear and trilinear surfaces.

8.1 The Motion Estimation Problem

Before delving into the details of segmentation of multiple 3-D motion models, we present a brief overview of the classical 3-D motion estimation problem from point correspondences. Sections 8.1.2 and 8.1.3 review the two-view geometry of non-planar and planar scenes, respectively, and Section 8.1.4 reviews the three view geometry of non-planar scenes. We refer the readers to [?, ?] for further details.

8.1.1 Rigid-Body Motions and Camera Projection Models

Consider a video sequence taken by a moving camera observing a static scene. We assume that the camera is moving rigidly, so that its pose at frame $f = 1, \dots, F$ can be expressed as $(R_f, T_f) \in SE(3)$, where $R_f \in SO(3)$ is the camera rotation and $T_f \in \mathbb{R}^3$ is the camera translation. Without loss of generality, we assume that the first camera frame coincides with the world frame, so that $(R_1, T_1) = (I, \mathbf{0})$.

Consider a generic point p, with coordinates $\boldsymbol{X}_1 = (X_1, Y_1, Z_1)^\top \in \mathbb{R}^3$ relative to the world reference frame. As illustrated in Figure 8.1, the coordinates $\boldsymbol{X}_f = (X_f, Y_f, Z_f)^\top$ of the same point p relative to the fth camera frame are given the rigid-body transformation (R_f, T_f) of $\boldsymbol{X}_1$:

$$\boldsymbol{X}_f = R_f \boldsymbol{X}_1 + T_f \quad \in \mathbb{R}^3. \tag{8.1}$$

Adopting the pinhole camera model shown in Figure 8.2 with focal length $d(f)$, the point p with coordinates $\boldsymbol{X}_f$ is projected onto the image plane at the point

$$\begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix} = \frac{1}{Z_f} \begin{bmatrix} d(f) & 0 & 0 \\ 0 & d(f) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_f \\ Y_f \\ Z_f \end{bmatrix}. \tag{8.2}$$

The projection model (8.2) is specified relative to a very particular reference frame centered at the optical center with one axis aligned with the optical axis. In

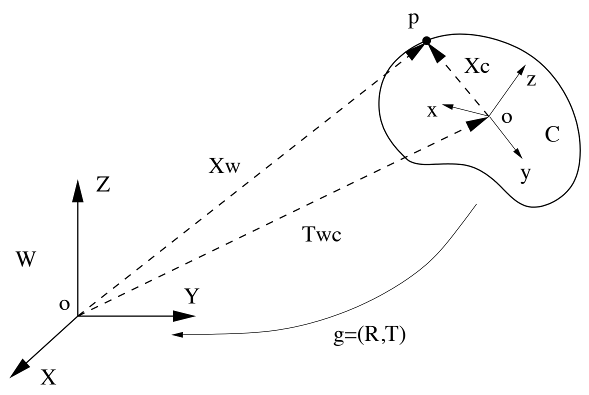


Figure 8.1. A rigid-body motion between a moving frame C and a world frame W.

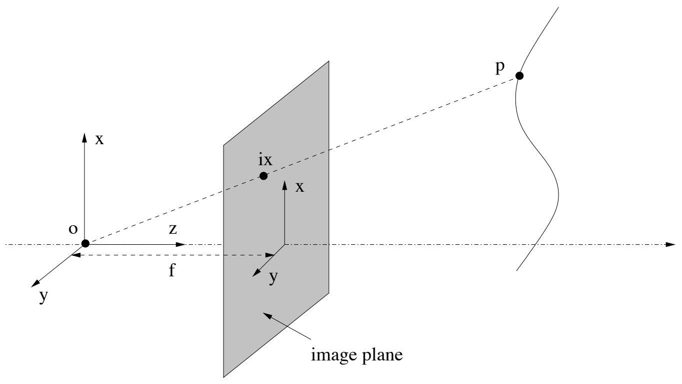


Figure 8.2. Frontal pinhole imaging model: the image of a 3-D point p is the point $\boldsymbol{x}$ at the intersection of the ray going through the optical center o and the image plane at a distance $d(f)$ in front of the optical center.

practice, when one captures digital images the measurements are obtained in pixel coordinates, which are related to the image coordinates by the transformation

$$\boldsymbol{x}_f \doteq \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix}, \tag{8.3}$$

where (s_x, s_y) is a scale factor, $\boldsymbol{s}_\theta$ is a skew factor and (o_x, o_y) is a translation so that the origin is in the upper-left corner of the image.

Combining the camera motion model (8.1), the camera projection model (8.2) and the camera calibration model (8.3), leads to the following *camera model*:

$$\lambda_f \boldsymbol{x}_f = \underbrace{\begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d(f) & 0 & 0 \\ 0 & d(f) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{K_f \in \mathbb{R}^{3\times 3}} \begin{bmatrix} R_f & T_f \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix}, \tag{8.4}$$

where $\lambda_f = Z_f$ and K_f are, respectively, the *depth* of the point and the *camera calibration matrix* in the fth frame. When $K_f = I$, we say that the camera is calibrated. We call the 3×4 matrix $\Pi_f = K_f[R_f\ T_f]$ the *projection matrix*.

8.1.2 The Fundamental Matrix

Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be images of point p in the first and second frames of a video sequence consisting of $F = 2$ frames. As illustrated in Figure 8.3, the vectors $\boldsymbol{X}_2$, T_2 and $R_2\boldsymbol{X}_1$ must be coplanar, hence their triple product must be zero, i.e.,

$$\boldsymbol{X}_2 \cdot (T_2 \times R_2 \boldsymbol{X}_1) = 0 \iff \boldsymbol{X}_2^\top \widehat{T_2} R_2 \boldsymbol{X}_1 = 0. \tag{8.5}$$

where $\widehat{T_2} \in so(3)$ is a skew-symmetric matrix generating the cross product by T_2.

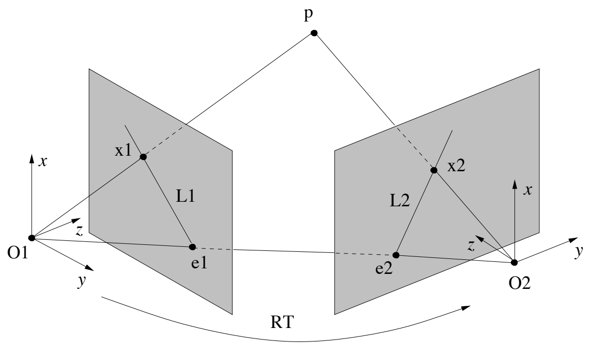


Figure 8.3. Epipolar geometry: Two projections $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^3$ of a 3-D point $\boldsymbol{X}$ from two vantage points. The relative Euclidean transformation between the two vantage points is given by $(R, T) \in SE(3)$. The intersections of the line (o_1, o_2) with each image plane are called *epipoles* and are denoted as $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. The intersections of the plane (o_1, o_2, p) with the two image planes are called *epipolar lines* and are denoted $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$.

It follows from equation (8.4) that $\lambda_1 \boldsymbol{x}_1 = K_1 \boldsymbol{X}_1$ and $\lambda_2 \boldsymbol{x}_2 = K_2 \boldsymbol{X}_2$. Therefore, the following *epipolar constraint* [?] must be satisfied by the relative camera motion (R_2, T_2) and the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$

$$\boldsymbol{x}_2^\top K_2^{-\top} \widehat{T_2} R_2 K_1^{-1} \boldsymbol{x}_1 = 0 \iff \boldsymbol{x}_2^\top F \boldsymbol{x}_1 = 0. \tag{8.6}$$

The matrix $F = K_2^{-\top} \widehat{T_2} R_2 K_1^{-1} \in \mathbb{R}^{3\times 3}$ is called the *fundamental matrix* and is defined up to an indeterminate scale. By construction, F is a rank-2 matrix having $\boldsymbol{e}_1 = R_2^\top K_1 T_2$ and $\boldsymbol{e}_2 = K_2 T_2$ as its right and left null spaces. The vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are known as the *epipoles* in the first and second view, respectively.

Since there are 9 unknowns in the fundamental matrix F (up to a scale), one can linearly solve for F from the epipolar constraint (8.6) from $N \geq 8$ point correspondences $\{(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})\}_{i=1}^N$ in general configuration. Given some additional knowledge about the camera calibration K_1 and K_2, one can solve for the camera motion (R_2, T_2) from F using the *eight-point algorithm* [?].

8.1.3 The Homography Matrix

The motion estimation scheme described in the previous subsection assumes that the displacement of the camera between the two views is nonzero, i.e., $T_2 \neq \mathbf{0}$, otherwise the fundamental matrix $F = K_2^{-\top} \widehat{T_2} R_2 K_1^{-1}$ would be zero. Furthermore, it also requires that the 3-D points be in general configuration, otherwise one cannot uniquely recover F from the epipolar constraint [?]. The latter case occurs, for example, when the 3-D points lie in a plane $\boldsymbol{N}^\top \boldsymbol{X}_1 = d$, where $\boldsymbol{N} \in \mathbb{S}^2$ is the normal to the plane and d is the distance from the plane to the origin of the first view. It follows from the equations $\boldsymbol{X}_2 = R_2 \boldsymbol{X}_1 + T_2$, $\boldsymbol{N}^\top \boldsymbol{X}_1 = d$, $\lambda_1 \boldsymbol{x}_1 = K_1 \boldsymbol{X}_1$ and $\lambda_2 \boldsymbol{x}_2 = K_2 \boldsymbol{X}_2$ that the following *homography constraint* holds

$$\boldsymbol{X}_2 = \left(R_2 + \frac{1}{d}T_2\boldsymbol{N}^\top\right)\boldsymbol{X}_1 \implies \boldsymbol{x}_2 \sim K_2\left(R_2 + \frac{1}{d}T_2\boldsymbol{N}^\top\right)K_1^{-1}\boldsymbol{x}_1 \quad (8.7)$$

The matrix $H = K_2(R_2 + \frac{1}{d}T\mathbf{N}^\top)K_1^{-1}$ is called the *homography matrix* and is, in general, defined up to an indeterminate scale. Notice that the homography constraint $\boldsymbol{x}_2 \sim H\boldsymbol{x}_1$ also holds for non-planar scenes undergoing pure rotation. In this case we simply have $H = K_2R_2K_1^{-1}$. Since there are 9 unknowns in the homographt matrix H (up to a scale), one can linearly solve for H from the homography constraint (8.7) from $N \geq 8$ point correspondences $\{(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})\}_{i=1}^N$. Given some additional knowledge about the camera calibration K_1 and K_2, one can solve for the camera motion (R_2, T_2) from F using linear methods.

8.1.4 The Trifocal Tensor

Let $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2 \leftrightarrow \boldsymbol{x}_3$ be a point correspondence in three perspective views with 3×4 camera matrices

$$\Pi_1 = [K_1 \; 0], \; \Pi_2 = [K_2R_2 \; \boldsymbol{e}_2] \text{ and } \Pi_3 = [K_3R_3 \; \boldsymbol{e}_3], \quad (8.8)$$

where $\boldsymbol{e}_2 \in \mathbb{P}^2$ and $\boldsymbol{e}_3 \in \mathbb{P}^2$ are the epipoles in the 2nd and 3rd views, respectively. Let $\boldsymbol{\ell}_2$ be any line passing through $\boldsymbol{x}_2$, i.e., $\boldsymbol{\ell}_2^\top\boldsymbol{x}_2 = 0$, and $\boldsymbol{\ell}_3$ be any line passing through $\boldsymbol{x}_3$, i.e., $\boldsymbol{\ell}_3^\top\boldsymbol{x}_3 = 0$. Then, the multiple view matrix [**?**]

$$\begin{bmatrix} \boldsymbol{\ell}_2^\top K_2R_2\boldsymbol{x}_1 & \boldsymbol{\ell}_2^\top\boldsymbol{e}_2 \\ \boldsymbol{\ell}_3^\top K_3R_3\boldsymbol{x}_1 & \boldsymbol{\ell}_3^\top\boldsymbol{e}_3 \end{bmatrix} \in \mathbb{R}^{2\times 2} \quad (8.9)$$

must have rank 1, hence its determinant must be zero, i.e.,

$$\boldsymbol{\ell}_2^\top(K_2R_2\boldsymbol{x}_1\boldsymbol{e}_3^\top - \boldsymbol{e}_2\boldsymbol{x}_1^\top R_3^\top K_3^\top)\boldsymbol{\ell}_3 = 0. \quad (8.10)$$

This is the well-known point-line-line *trilinear constraint* among the three views [?], which we will denote as

$$T(\boldsymbol{x}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3) = \sum_{p,q,r} T_{pqr}x_{1p}\ell_{2q}\ell_{3r} = 0 \quad (8.11)$$

where $T \in \mathbb{R}^{3\times3\times3}$ is the so-called *trifocal tensor*.

Computing the trifocal tensor

Since there are 27 unknowns in the trifocal tensor T (up to a scale factor), one can linearly solve for T from the trilinear constraint (8.11) given at least 26 point-line-line correspondences. However, if we are given point-point-point correspondences, then for each point in the 2nd view $\boldsymbol{x}_2$, we can obtain two lines $\boldsymbol{\ell}_{21}$ and $\boldsymbol{\ell}_{22}$ passing through $\boldsymbol{x}_2$, and similarly for the 3rd view. Therefore, each point correspondence gives $2^2 = 4$ linearly independent equations on T and we only need 7 point correspondences to linearly estimate T.

Computing epipoles, epipolar lines and camera matrices

Given the trifocal tensor T, it is well known how to compute the epipolar lines in the 2nd and 3rd views of a point $\boldsymbol{x}_1$ in the 1st view [?]. Specifically, notice from (8.11) that the matrix

$$(K_2 R_2 \boldsymbol{x}_1 \boldsymbol{e}_3^\top - \boldsymbol{e}_2 \boldsymbol{x}_1^\top R_3^\top K_3^\top) \in \mathbb{R}^{3\times 3} \tag{8.12}$$

has rank 2. In fact its left null-space is $\boldsymbol{\ell}_2(\boldsymbol{x}_1) = \boldsymbol{e}_2 \times K_2 R_2 \boldsymbol{x}_1$ and its right null-space is $\boldsymbol{\ell}_3(\boldsymbol{x}_1) = \boldsymbol{e}_3 \times K_3 R_3 \boldsymbol{x}_1$, i.e., the epipolar lines of $\boldsymbol{x}_1$ in the second and third views, respectively.

The epipoles in the second and third views $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ must lie in the epipolar lines in the second and third views, $\{\boldsymbol{\ell}_2(\boldsymbol{x}_{1i})\}_{i=1}^N$ and $\{\boldsymbol{\ell}_3(\boldsymbol{x}_{1i})\}_{i=1}^N$, respectively. Thus we can obtain the epipoles from

$$\boldsymbol{e}_2^\top [\boldsymbol{\ell}_2(\boldsymbol{x}_{11}), \ldots, \boldsymbol{\ell}_2(\boldsymbol{x}_{1N})] = 0 \text{ and } \boldsymbol{e}_3^\top [\boldsymbol{\ell}_3(\boldsymbol{x}_{11}), \ldots, \boldsymbol{\ell}_3(\boldsymbol{x}_{1N})] = 0. \tag{8.13}$$

Clearly, we only need 2 epipolar lines to determine the epipoles, hence we do not need to compute the epipolar lines for all points in the first view. However, it is better to use more than two lines in the presence of noise.

Finally, given T, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$, one can solve for the camera matrices Π_1, Π_2 and Π_3 using linear techniques [?].

8.2 The Motion Segmentation Problem

Consider a moving camera with pose $g_0(t) \in SE(3)$ at time t observing a scene containing n moving objects with poses $\{g_j(t) \in SE(3)\}_{j=1}^n$ at time t. The motion of object j relative to the camera between the zeroth and fth frame is given by $(R_{fj}, T_{fj}) = g_j(f) g_0(f)^{-1} g_0(0) g_j(0)^{-1} \in SE(3)$. Let $\{\boldsymbol{X}_i \in \mathbb{R}^3\}_{i=1}^N$ be a collection of points in 3-D space lying in the n moving objects. The projection of a point $\boldsymbol{X}_i$ lying in the jth object onto the fth camera frame is given by

$$\boldsymbol{x}_{fi} = \pi_f(R_{jf} \boldsymbol{X}_i + T_{jf}), \tag{8.14}$$

where $\pi_f : \mathbb{R}^3 \mapsto I$ is the camera projection model (orthographic, perspective, etc.). In this chapter, we will consider the following problem.

Problem 8.1 (3D Motion Segmentation from Point Correspondences)

Given N image points $\{\boldsymbol{x}_{fi}\}_{i=1,\ldots,N}^{f=1,\ldots,F}$ taken from F views of a motion sequence related by a collection of n 3-D motion models, estimate the number of motion models n and their parameters $\{\mathcal{M}_j\}_{j=1}^n$ without knowing which measurements correspond to which motion model.

In some cases, the camera model is such that the 3-D motion models are linear on the image measurements, thus Problem 8.1 is a direct application of GPCA. In other cases, the motion models are more complex, e.g., bilinear or trilinear. We develop extensions of GPCA to deal with such classes of segmentation problems.

8.3 Segmentation of Linear Motion Models

In this section, we consider the 3-D motion segmentation problem (Problem 8.1) in cases in which the projection model is such that the resulting 3-D motion model is *linear* in the image measurements. In particular, we consider the segmentation of rigid-body motions from point correspondences in multiple affine views and show that the motion segmentation problem boils down to segmenting low-dimensional subspaces of a high-dimensional space.

8.3.1 *The Affine Motion Subspaces*

Let $\{\boldsymbol{x}_{fi} \in \mathbb{R}^2\}_{i=1,\dots,N}^{f=1,\dots,F}$ be the images of N 3-D points $\{\boldsymbol{X}_i \in \mathbb{P}^3\}_{i=1}^N$ seen by a rigidly moving camera in F frames. Under the affine projection model, which generalizes orthographic, weak perspective, and paraperspective projection [?], the images satisfy the equation

$$\boldsymbol{x}_{fi} = A_f \boldsymbol{X}_i, \tag{8.15}$$

where $A_f = K_f \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} [R_f \;\; T_f] \in \mathbb{R}^{2\times 4}$ is the so-called *affine camera matrix* at frame f and depends on the pose of the camera relative to the world $(R_f, T_f) \in SE(3)$ and the internal camera calibration parameters $K_f \in SL(2)$.

When the set of points $\{\boldsymbol{X}_i\}_{i=1}^N$ all correspond to a single rigidly moving object, we can stack of all the image measurements $\{\boldsymbol{x}_{fi}\}$ into a $2F \times N$ matrix W, which can be decomposed into a *motion matrix* M and *structure matrix* S as

$$\begin{gathered} W = MS \\ \begin{bmatrix} \boldsymbol{x}_{11} & \cdots & \boldsymbol{x}_{1N} \\ \vdots & & \vdots \\ \boldsymbol{x}_{F1} & \cdots & \boldsymbol{x}_{FN} \end{bmatrix}_{2F\times N} = \begin{bmatrix} A_1 \\ \vdots \\ A_F \end{bmatrix}_{2F\times 4} \begin{bmatrix} \boldsymbol{X}_1 & \cdots & \boldsymbol{X}_N \end{bmatrix}_{4\times N}. \end{gathered} \tag{8.16}$$

It follows from equation (8.16) that rank$(W) \leq 4$. In addition, notice that the two rows of each A_f are linear combinations of the first two rows of a rotation matrix R_f, hence rank$(W) \geq$ rank$(A_f) \geq 2$. Therefore, the 2-D point trajectories of 3-D points lying in a single rigidly moving object (the columns of the data matrix W) live in a subspace of $\mathbb{R}^{2F}$ of dimension $d = 2$, 3 or 4.[1]

Consider now the case in which the set of points $\{\boldsymbol{X}_i\}_{i=1}^N$ corresponds to n rigid objects moving independently. It follows from our analysis in the previous section that, if we knew the segmentation of the feature points, then we could write the measurement matrix as $W = [W_1, W_2, \dots, W_n]$, where the columns of $W_j \in \mathbb{R}^{2F\times N_j}$ are the N_j measurements associated with the jth moving object, so that $\sum_{j=1}^n N_j = N$. It also follows from our analysis in the previous section

[1]This rank constraint was derived in [?], and was used to propose the first multi-frame algorithm for estimating the motion of an affine camera observing a static scene.

that each measurement matrix W_j satisfies

$$W_j = M_j S_j \quad j = 1, \ldots, n, \tag{8.17}$$

where $M_j \in \mathbb{R}^{2F \times 4}$ and $S_j \in \mathbb{R}^{4 \times N_i}$ are, respectively, the motion and structure matrices associated with the jth moving object.

8.3.2 *Segmenting Affine Motion Subspaces*

In reality, the segmentation of the feature points is *unknown*, and so the measurement matrix is given by $W = [W_1, W_2, \ldots, W_n]P$, where $P \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix. Nevertheless, the columns of W still live in a union of n motion subspaces $\{S_j \subset \mathbb{R}^{2F}\}_{j=1}^n$ of dimensions $d_j \in \{2, 3, 4\}$ for $j = 1, \ldots, n$.

It is worth noting that, under certain additional assumptions, the problem of segmenting the motion subspaces can be solved using a simpler algorithm that depends only on the SVD of the data matrix $W = U\Sigma V^\top$. For example, when the motion subspaces are fully dimensional, i.e., $\dim(S_j) = 4$, and fully independent, i.e.,

$$\dim(S_1 \cup S_2 \cup \cdots \cup S_n) = \dim(S_1) + \dim(S_2) + \cdots + \dim(S_n),$$

or equivalently $S_j \cap S_k = \{\mathbf{0}\}$ for all $j \neq k$, one can apply the Costeira and Kanade (CK) algorithm [**?**] to segment the n motion subspaces. The CK algorithm is based on thresholding the entries of the so-called *shape interaction* matrix

$$Q = V(1:4)V(1:4)^\top \quad \in \mathbb{R}^{N \times N}, \tag{8.18}$$

where $V(1:4)$ contains the first four columns of V. The matrix Q has the property that [Kanatani, 2001]

$$Q_{ij} = 0 \quad \text{if} \quad i \text{ and } j \text{ correspond to different motions.} \tag{8.19}$$

This property has been the basis for most existing motion segmentation algorithms, such as [?, Kanatani, 2001, Kanatani, 2002, **?**, Wu et al., 2001].

In general, however, the motions need not be fully free. For example, the motion of ground automobiles relative to a camera is constrained to be planar, which reduces the dimension of the motion subspaces to $d = 3$. In addition, the motion subspaces may be partially dependent, i.e., $\max\{\dim(S_j), \dim(S_k)\} < \dim(S_j \cup S_k) < \dim(S_j) + \dim(S_k)$ or equivalently $S_j \cap S_k \neq \{\mathbf{0}\}$, $S_j \cap S_k \neq S_j$ and $S_j \cap S_k \neq S_k$, which happens for instance when two objects move with the same rotation but different translation relative to the camera. As reported in [**?**, **?**, Vidal and Hartley, 2004], most existing motion segmentation algorithms show poor performance in the presence of degenerate or partially dependent motions, because they cannot deal with subspaces that have nontrivial intersection and have different dimensions.

Nevertheless, the GPCA algorithm discussed in Chapter 3 does not impose any restriction on either the intersection or the dimensionality of the subspaces, hence it can deal with all the spectrum of affine motions: from two-dimensional and partially dependent to four-dimensional and fully independent. Segmentation of 3-D motions from point correspondences in multiple affine views is equivalent to segmenting subspaces of $\mathbb{R}^{2F}$ of dimensions $d_1, \ldots, d_n \leq d_{\max} = 4$. As discussed in Chapter 3, we can solve this problem by applying GPCA to the $2F$-dimensional point trajectories projected onto a subspace of dimension $D = d_{\max} + 1 = 5$ in $\mathbb{R}^{2F}$. That is, if $W = U\Sigma V^\top$ is the SVD of the data matrix, then we can solve the motion segmentation problem by applying GPCA (Algorithm 3.4) to the first 5 columns of V.

8.3.3 Experimental Results

We tested the GPCA algorithm on three different sequences shown in Figure 8.4. The data for these sequences consist of point correspondences in multiple views, which are available at http://www.suri.it.okayama-u.ac.jp/data.html. Sequence A consists of 30 frames of an outdoor sequence taken by a moving camera tracking a car moving in front of a parking lot. Sequence B consists of 17 frames of an outdoor sequence taken by a moving camera tracking a car moving in front of a building. Sequence C consists of 100 frames of an indoor sequence taken by a moving camera tracking a person moving his head.

For all sequences, we first projected the point trajectories onto a 5-dimensional subspace of $\mathbb{R}^{2F}$, where F is the number of frames in the sequence. We assumed that the motion subspaces are 4-dimensional, so that the motion segmentation problem is reduced to segmenting 4-dimensional hyperplanes in $\mathbb{R}^5$. The number of motions is correctly estimated from (3.20) as $n = 2$. We used the criterion (2.48) with $\kappa \in [2, 20]\,10^{-7}$ to determine the rank of the embedded data matrix.

As shown in Table 8.1, GPCA gives a percentage of correct classification of 100.0% for all three sequences. The table also shows results reported in [?] from existing *multiframe* algorithms for motion segmentation. The only algorithm having a comparable performance to GPCA is Kanatani's multi-stage optimization algorithm, which is based on solving a series of EM-like iterative optimization problems, at the expense of a significant increase in computation.

8.4 Segmentation of Bilinear Motion Models

In this section, we consider the 3-D motion segmentation problem (Problem 8.1) in cases in which the projection model is such that the resulting 3-D motion model is *bilinear* in the image measurements. In particular, we consider the segmentation of rigid-body motions from point correspondences in two perspective views of nonplanar (Section 8.4.1) and planar (Section 8.4.2) scenes. In both cases, we show that the motion segmentation problem can be solved using extensions of GPCA to certain classes of bilinear surfaces.

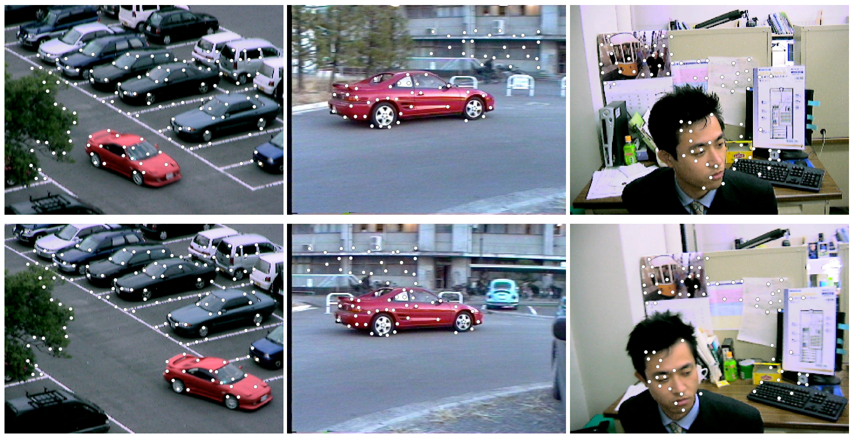


Figure 8.4. Segmenting the point correspondences of sequences A (left), B (center) and C (right) in [?] by clustering subspaces in $\mathbb{R}^5$. First row: first frame of the sequence with point correspondences superimposed. Second row: last frame of the sequence with point correspondences superimposed.

Table 8.1. Classification rates given by various subspace segmentation algorithms for sequences A, B, C in [?].

Sequence	A	B	C
Number of points	136	63	73
Number of frames	30	17	100
Costeira-Kanade	60.3%	71.3%	58.8%
Ichimura	92.6%	80.1%	68.3%
Kanatani: subspace separation	59.3%	99.5%	98.9%
Kanatani: affine subspace separation	81.8%	99.7%	67.5%
Kanatani: multi-stage optimization	100.0%	100.0%	100.0%
GPCA	100.0%	100.0%	100.0%

8.4.1 Segmenting Fundamental Matrices

In this subsection, we consider the problem of segmenting n 3-D rigid-body motions $\{(R_j, T_j) \in SE(3)\}_{j=1}^n$ from point correspondences in two perspective views. We assume that the 3-D scene is nonplanar and that the individual translations T_i are all nonzero. In this case, the motion of the objects relative to the camera between the two views can be modeled as a mixture of fundamental matrices $\{F_j\}_{j=1}^n$. In order for the problem to be well posed, we assume that the fundamental matrices are different from each other (up to a scale factor).

The multibody epipolar constraint and the multibody fundamental matrix

As shown in Section 8.1.2, if $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is an image pair associated with any the n moving objects, then exists a fundamental matrix F_j such that $\boldsymbol{x}_2^\top F_j \boldsymbol{x}_1 = 0$.

Therefore, the following *multibody epipolar constraint* must be satisfied by the number of independent motions n, the fundamental matrices $\{F_j\}_{j=1}^n$ and the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, regardless of which motion is associated with the image pair

$$p_n(\boldsymbol{x}_1, \boldsymbol{x}_2) \doteq \prod_{j=1}^{n} \left(\boldsymbol{x}_2^\top F_j \boldsymbol{x}_1\right) = 0. \tag{8.20}$$

The multibody epipolar constraint (8.20) and the multibody brightness constancy constraint (MBCC) for affine motions (**??**) are both homogeneous polynomials of degree n that factor as a product of n bilinear forms. Therefore, as shown in Theorem **??**, the multibody epipolar constraint can be written in bilinear form as

$$\prod_{j=1}^{n} \left(\boldsymbol{x}_2^\top F_j \boldsymbol{x}_1\right) = \nu_n(\boldsymbol{x}_2)^\top \mathcal{F} \nu_n(\boldsymbol{x}_1) = 0. \tag{8.21}$$

We call the matrix $\mathcal{F} \in \mathbb{R}^{M_n(3)\times M_n(3)}$ the *multibody fundamental matrix* as it is a natural generalization of the fundamental matrix to the case of multiple moving objects. Also, since equation (8.21) clearly resembles the bilinear form of the epipolar constraint for a single rigid-body motion, we will refer to both equations (8.20) and (8.21) as the *multibody epipolar constraint* from now on.

Although the multibody fundamental matrix $\mathcal{F}$ seems a complicated mixture of all the individual fundamental matrices $F_1, \ldots, F_n$, it is still possible to recover all the individual fundamental matrices from $\mathcal{F}$, under some mild conditions (e.g., the F_j's are different). The rest of the section is devoted to providing a constructive proof for this. We first show how to recover n and $\mathcal{F}$ from data, and then show how to recover $\{F_j\}_{j=1}^n$ from $\mathcal{F}$.

Estimating the number of motions n and the multibody fundamental matrix $\mathcal{F}$

Both the MBCC for affine motions (**??**) and the multibody epipolar constraint (8.21) are bilinear expressions on the embedded image measurements and linear expressions on the multibody motion parameters. Therefore, given $N \geq M_n(3)^2 - 1 \sim O(n^4)$ generic point correspondences $\{(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})\}_{i=1}^N$, we can solve for the stack of the columns of the multibody fundamental matrix $\mathcal{F}$, $\text{vec}(\mathcal{F}) \in \mathbb{R}^{M_n(3)^2}$, from the linear system (see equation (**??**))

$$\boldsymbol{V}_n^{\mathcal{F}} \text{vec}(\mathcal{F}) \doteq \left[\nu_n(\boldsymbol{x}_{11}) \otimes \nu_n(\boldsymbol{x}_{21}), \ldots, \nu_n(\boldsymbol{x}_{1N}) \otimes \nu_n(\boldsymbol{x}_{2N})\right]^\top \text{vec}(\mathcal{F}) = \boldsymbol{0}, \tag{8.22}$$

and for the number of independent motions n from (see Theorem **??**)

$$n \doteq \min\{j : \text{rank}(\boldsymbol{V}_j^{\mathcal{F}}) = M_j(3)^2 - 1\}. \tag{8.23}$$

Factorization of the multibody fundamental matrix

Given the multibody fundamental matrix $\mathcal{F}$ and the number of independent motions n, we now show how to recover the fundamental matrices and the segmentation of the image points. We first show that the gradients of the multibody epipolar constraint at the point correspondences lie in a collection of hyperplanes in $\mathbb{R}^3$ whose normal vectors are the n epipoles. Therefore, one can apply GPCA to these gradients in order to obtain the epipoles as well as the segmentation of the data. Once the data has been segmented, computing a fundamental matrix for each group is a linear problem.

1. Given an image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ associated with the jth motion, its *epipolar line* in the second view (see Figure 8.3) is defined as $\boldsymbol{\ell}_j \doteq F_j \boldsymbol{x}_1 \in \mathbb{P}^2$. Since $\boldsymbol{x}_2^\top F_j \boldsymbol{x}_1 = 0$, we can compute $\boldsymbol{\ell}_j$ as the partial derivative of the multibody epipolar constraint with respect to $\boldsymbol{x}_2$ evaluated at $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, because
$$\frac{\partial}{\partial \boldsymbol{x}_2}\big(\nu_n(\boldsymbol{x}_2)^\top \mathcal{F} \nu_n(\boldsymbol{x}_1)\big) = \sum_{j=1}^{n} \prod_{k \neq j} (\boldsymbol{x}_2^\top F_k \boldsymbol{x}_1)(F_j \boldsymbol{x}_1) \tag{8.24}$$
$$= \prod_{k \neq j} (\boldsymbol{x}_2^\top F_k \boldsymbol{x}_1)(F_j \boldsymbol{x}_1) \sim \boldsymbol{\ell}_j. \tag{8.25}$$
Therefore, given a set of point correspondences $\{(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})\}_{i=1}^N$, we can compute its associated set of epipolar lines $\{\boldsymbol{\ell}_i\}_{i=1}^N$ as the gradient of the multibody epipolar constraint at the correspondences.

2. Given an epipolar line $\boldsymbol{\ell}$ associated with the jth motion, there exists an epipole $\boldsymbol{e}_j$ such that $\boldsymbol{e}_j^\top \boldsymbol{\ell} = \boldsymbol{e}_j^\top F_j \boldsymbol{x}_1 = 0$, because $\boldsymbol{e}_j$ is the left null space of F_j. Therefore, the set of N epipolar lines can be interpreted as a set of points in $\mathbb{R}^3$ lying in n hyperplanes with normal vectors $\{\boldsymbol{e}_j\}_{j=1}^n$. We can apply the GPCA algorithm (Algorithm 3.4) to estimate the n epipoles $\{\boldsymbol{e}_j\}_{j=1}^n$ up to a scale factor. If the n epipoles are different,[2] we can immediately segment the data into n groups by assigning the image pair $(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})$ to group j if
$$j = \arg \min_{k=1,\dots,n} (\boldsymbol{e}_k^\top \boldsymbol{\ell}_i)^2 \tag{8.26}$$

3. Once the data has been clustered, solving for the fundamental matrix F_j from the epipolar constraint $\boldsymbol{x}_2^\top F_j \boldsymbol{x}_1 = 0$ is a linear problem of the form (see equation (8.22))
$$\big[w_{1j}\boldsymbol{x}_{11} \otimes \boldsymbol{x}_{21}, w_{2j}, \dots, w_{Nj}\boldsymbol{x}_{1N} \otimes \boldsymbol{x}_{2N}\big]^\top \operatorname{vec}(F_i) = \mathbf{0}, \tag{8.27}$$
where $w_{ij} = 1$ if the ith point belongs to the jth group, and $w_{ij} = 0$ otherwise.

Algorithm 8.1 summarizes the algorithm for segmenting n fundamental matrices. Table 8.2 gives the minimum number of point correspondences required by the algorithm as a function of the number of motions.

[2]This is not a strong assumption. If two individual fundamental matrices share the same (left) epipoles, one can consider the right epipoles (in the first image frame) instead, because it is extremely rare that two motions give rise to the same left and right epipoles. In fact, this happens only when the rotation axes of the two motions are equal to each other and parallel to the translation direction [**?**].

Algorithm 8.1 (Segmentation of Fundamental Matrices).

Given two perspective views $\{(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})\}_{i=1}^N$ of a set of N 3-D points undergoing n different rigid-body motions, recover the number of independent motions n, the fundamental matrix F_j associated with each motion, and the motion model associated with each image pair as follows:

1: **Number of motions:** Form the embedded data matrix of degree $j \geq 1$, $\boldsymbol{V}_j^{\mathcal{F}} \in \mathbb{R}^{N \times M_j(3)^2}$, as in (8.22). Compute the number of independent motions n from (8.23) or else from

$$n = \arg\min_{j \geq 1} \frac{\sigma^2_{M_j(3)^2}(\boldsymbol{V}_j^{\mathcal{F}})}{\sum_{k=1}^{M_j(3)^2-1} \sigma_k^2(\boldsymbol{V}_j^{\mathcal{F}})} + \mu M_j(3)^2. \tag{8.28}$$

2: **Multibody fundamental matrix:** Compute the multibody fundamental matrix $\mathcal{F}$ as the least-squares solution to the linear system $\boldsymbol{V}_n^{\mathcal{F}} \text{vec}(\mathcal{F}) = \mathbf{0}$ in (8.22), where $\boldsymbol{V}_n^{\mathcal{F}}$ is computed using the Veronese map ν_n of degree n.

3: **Epipolar lines:** Compute the epipolar lines $\{\boldsymbol{\ell}_i\}_{i=1}^N$ in the second view associated with each image pair $\{(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})\}_{i=1}^N$ as the gradient of the multibody epipolar constraint with respect to $\boldsymbol{x}_2$ evaluated at each image pair.

4: **Epipoles:** Apply GPCA to the epipolar lines $\{\boldsymbol{\ell}_i\}_{i=1}^N$ to obtain the individual epipoles $\{\boldsymbol{e}_j\}_{j=1}^n$.

5: **Feature segmentation:** Assign image pair $(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})$ to motion $j = \arg\min_{k=1,\ldots,n} (\boldsymbol{e}_k^T \boldsymbol{\ell}_i)^2$.

6: **Fundamental matrices:** Obtain the individual fundamental matrices $\{F_j\}_{j=1}^n$ by applying the eight-point algorithm to each group.

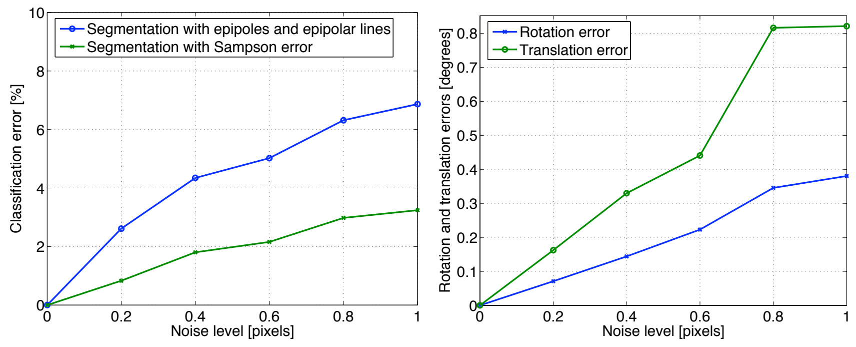
Simulation and experimental results

We first test Algorithm 8.1 on synthetic data. We randomly pick $n = 2$ collections of $N = 100n$ feature points and apply a different (randomly chosen) rigid body motion $(R_i, T_i) \in SE(3)$, with $R_i \in SO(3)$ the rotation and $T_i \in \mathbb{R}^3$ the translation. We add zero-mean Gaussian noise with standard deviation (STD) from 0 to 1 pixels to the images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The image size is 1000×1000 pixels. We run 1000 trials for each noise level. For each trial, the classification error is computed as the percentage of misclassified points, and the error between the true motions $\{(R_i, T_i)\}_{i=1}^n$ and their estimates $\{(\hat{R}_i, \hat{T}_i)\}_{i=1}^n$ are computed as

$$\text{Rotation error} = \frac{1}{n} \sum_{i=1}^n \text{acos}\Big(\frac{\text{trace}(R_i \hat{R}_i^T) - 1}{2}\Big) \qquad \text{(degrees)}.$$

$$\text{Translation error} = \frac{1}{n} \sum_{i=1}^n \text{acos}\Big(\frac{T_i^T \hat{T}_i}{\|T_i\| \|\hat{T}_i\|}\Big) \qquad \text{(degrees)}.$$

Figure 8.5 plots the mean classification error (%), the rotation error (degrees) and the translation error (degrees) as a function of noise. In all trials the number of motions was correctly estimated from equation (8.23) as $n = 2$.[3] The mean classification error is less than 7% using an assignment based on epipoles and epipolar lines, and can be reduced to about 3.25% using an assignment based on the Sampson error. The rotation error is less than $0.38°$ and the translation error is less than $0.83°$.



(a) Classification error (b) Motion estimation error

Figure 8.5. Percentage of misclassification and error in the estimation of rotation and translation (in degrees) as a function of the level of noise for $n = 2$ moving objects.

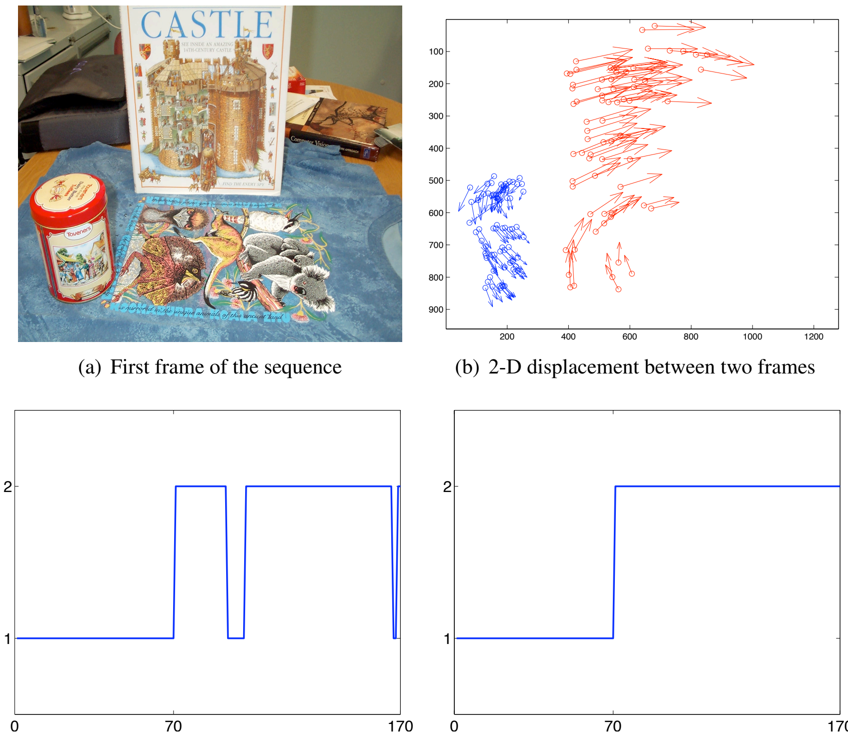
We also tested the proposed approach by segmenting a real sequence in which a moving camera observes a can moving in front of a static background consisting of a T-shirt and a book. We manually extracted a total of $N = 170$ correspondences: 70 for the can and 100 for the background. For comparison purposes, we estimated the ground truth motion (R_i, T_i) by applying the eight-point algorithm to manually segmented correspondences.

Figure 8.6 shows the first frame of the sequence as well as the relative displacement of the correspondences between the two frames. We applied Algorithm 8.1 to estimate the number of motions as $n = 2$.[4] We obtained a misclassification error of 5.88% when the clustering is obtained using epipolar lines and epipoles only. We used this segmentation to obtain the motion parameters for each group. The error in rotation was $0.07°$ for the background and $4.12°$ for the can. The error in translation was $0.21°$ for the background and $4.51°$ for the can. Given the motion parameters for each group, we re-clustered the features using the Sampson error (**??**). The misclassification error reduced to 0%.

8.4.2 *Segmenting Homography Matrices*

The motion segmentation scheme described in the previous subsection assumes that the displacement of each object between the two views relative to the camera

[3]We use $\kappa = 5 \times 10^{-3}$ in equation (2.48) for computing the number of motions.
[4]We use $\kappa = 5 \times 10^{-3}$ in equation (2.48) for computing the number of motions.



(a) First frame of the sequence (b) 2-D displacement between two frames

(c) Segmentation based on epipolar lines (d) Segmentation based on Sampson error

Figure 8.6. Segmenting two frames of a video sequence with two different rigid-body motions – the can and the background. (a) First frame of the sequence. (b) 2-D displacements of the 170 correspondences from the first view ('o') to the second ('$\rightarrow$'). (c) Segmentation of the 170 correspondences using epipoles and epipolar lines. (d) Segmentation of the 170 correspondences using Sampson distance.

is nonzero, i.e., $T \neq 0$, otherwise the individual fundamental matrices $F = \widehat{T}R$ would be zero. Furthermore, it also requires that the 3-D points be in general configuration, otherwise one cannot uniquely recover each fundamental matrix from its epipolar constraint. The latter case occurs, for example, in the case of planar structures, i.e., when the 3-D points lie in a plane [?].

Both in the case of purely rotating objects (relative to the camera) or in the case of a planar 3-D structure, the motion (R, T) between the two views $\boldsymbol{x}_1 \in \mathbb{P}^2$ and $\boldsymbol{x}_2 \in \mathbb{P}^2$ is described by a homography matrix $H \in \mathbb{R}^{3\times3}$. If $N \in \mathbb{R}^3$ is the (unit) normal to the plane and d is the distance from the plane to the origin, the homography matrix $H = R + \frac{1}{d}TN^\top$ is such that [?]

$$\boldsymbol{x}_2 \sim H\boldsymbol{x}_1 = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \boldsymbol{x}_1. \tag{8.29}$$

When the camera calibration $K \in \mathbb{R}^{3\times 3}$ is also unknown, the homography matrix is written as $H = K(R + \frac{1}{d}TN^\top)K^{-1}$.

The multibody homography

Consider now a scene that can be modeled with n different homographies $\{H_j\}_{j=1}^n$. Note that the n homographies do *not* necessarily correspond to n different rigid-body motions, as one rigidly moving object could consists of two or more planes whose motion is described by two or more homographies. Therefore, the n homographies can represent anything from 1 up to n rigid-body motions.

It is evident from the form of equation (8.29) that in order to eliminate the segmentation of the data we cannot take the product of all the equations, as we did with the epipolar constraints, because in this case we have two linearly independent equations per image pair. In Chapter 3 we dealt with this issue by using multiple polynomials to represent multiple subspaces of co-dimension more than one. In the case of homographies, one can avoid using multiple polynomials by exploiting properties of the cross product in $\mathbb{R}^3$, as we show now.

We start by noticing that if $\boldsymbol{\ell}$ is a line passing through $\boldsymbol{x}_2$, i.e., $\boldsymbol{\ell}$ is such that $\boldsymbol{\ell}^\top \boldsymbol{x}_2 = 0$, then it follows from (8.29) that there is a motion i such that $\boldsymbol{\ell}^\top H_j \boldsymbol{x}_1 = 0$. Therefore, the following *multibody homography constraint* must hold

$$p_n(\boldsymbol{x}_1, \boldsymbol{\ell}) = \prod_{j=1}^n (\boldsymbol{\ell}^\top H_j \boldsymbol{x}_1) = \nu_n(\boldsymbol{\ell})^\top \mathcal{H} \nu_n(\boldsymbol{x}_1) = 0. \tag{8.30}$$

We call the matrix $\mathcal{H} \in \mathbb{R}^{M_n(3)\times M_n(3)}$ the *multibody homography*. Notice that $\mathcal{H}$ plays the analogous role of the multibody affine matrix $\mathcal{A}$, or the multibody fundamental matrix $\mathcal{F}$.

Computing the number of homographies n and the multibody homography $\mathcal{H}$

In order to compute n and $\mathcal{H}$ from a given a set of point P correspondences $\{(\boldsymbol{x}_{1p}, \boldsymbol{x}_{2p})\}_{p=1}^P$, notice that given an image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, we can generate two linearly independent lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ passing through $\boldsymbol{x}_2$. This may lead us to conclude that each correspondence gives two linearly independent constraints on the entries of $\mathcal{H}$. In reality, each correspondence gives $n+1$ constraints on $\mathcal{H}$.

To see this, notice that equation (8.30) must hold for any line passing through $\boldsymbol{x}_2$. Since the family of lines $\alpha\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2$ passes through $\boldsymbol{x}_2$ for all $\alpha \in \mathbb{R}$, we have

$$q(\alpha) = \prod_{j=1}^n ((\alpha\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2)^\top H_j \boldsymbol{x}_1) = \nu_n(\alpha\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2)^\top \mathcal{H} \nu_n(\boldsymbol{x}_1) = 0. \tag{8.31}$$

One can show that (see Exercise 8.2)

$$\nu_n(\alpha\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2) = \sum_{j=0}^n \alpha^j f_j(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2), \tag{8.32}$$

where $f_j(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2) \in \mathbb{R}^{M_n(3)}$ a polynomial of degree i in $\boldsymbol{\ell}_1$ and $(n-i)$ in $\boldsymbol{\ell}_2$ for $j = 0, \ldots, n$. Therefore, $q(\alpha)$ is a polynomial of degree n in α such that $\boldsymbol{q}(\alpha) = 0$

for all α, and so each one of its $n+1$ coefficients must be zero, i.e.,

$$f_j(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2)^\top \mathcal{H} \nu_n(\boldsymbol{x}_1) = 0, \qquad j = 0, \dots, n. \tag{8.33}$$

This gives $n+1$ constraints per point correspondence on the entries of $\mathcal{H}$. Therefore, given $P \geq \frac{M_n(3)^3 - 1}{n+1} \sim O(n^3)$ generic point correspondences $\{(\boldsymbol{x}_{1p}, \boldsymbol{x}_{2p})\}_{p=1}^P$, we can solve for the stack of the columns of the multibody homography $\mathcal{H}$, $\boldsymbol{h} \in \mathbb{R}^{M_n(3)^2}$, from the linear system

$$\boldsymbol{V}_n^{\mathcal{H}} \boldsymbol{h} = \boldsymbol{0}, \tag{8.34}$$

where the ith row of $\boldsymbol{V}_n^{\mathcal{H}} \in \mathbb{R}^{P(n+1) \times M_n(3)^2}$ is given by $\nu_n(\boldsymbol{x}_1) \otimes f_j(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2)$.

Finally, similarly to (**??**) and (8.23), the number of homographies is given by

$$n \doteq \min\{j : \text{rank}(\boldsymbol{V}_j^{\mathcal{H}}) = M_j(3)^2 - 1\}. \tag{8.35}$$

Factorization of the multibody homography

Once $\mathcal{H}$ is known, computing the homographies $\{H_j\}_{j=1}^n$ is equivalent to factorizing the multibody homography constraint (8.30) into a product of n bilinear factors. In general, solving such a factorization problem is difficult, unless some structure about the matrices H_j is known. In the case of fundamental matrices discussed in Section 8.4.1, we exploited the fact that epipoles are the left null spaces of the fundamental matrices. In the case of affine matrices discussed in Section **??**, we exploited the fact that the 3rd row of each affine matrix is known.

Unfortunately, homographies are usually full rank and none of their rows are known. Hence, the factorization of $\mathcal{H}$ is more challenging than the factorization of the multibody affine matrix $\mathcal{A}$ or the multibody fundamental matrix $\mathcal{F}$. In fact, we will show that the factorization of $\mathcal{H}$ requires a combination of the methods for factorizing $\mathcal{A}$ and $\mathcal{F}$, according to the following four steps:

1. Compute derivatives of $p_n(\boldsymbol{x}_1, \boldsymbol{\ell})$ with respect to $\boldsymbol{x}_1$ to obtain linear combinations of the rows of each H_j.
2. Obtain three vectors orthogonal to the three pairs of rows of each H_j by solving three hyperplane clustering problems.
3. Obtain the rows of each H_j up to a scale factor from the cross products of these vectors.
4. Solve linearly for the unknown scales of the rows of H_j from the homography constraint.

For step 1, notice from (**??**) that if the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is associated with the ith motion and $\boldsymbol{\ell}$ is any line passing through $\boldsymbol{x}_2$, then the derivative of $p_n(\boldsymbol{x}_1, \boldsymbol{\ell})$ with respect to $\boldsymbol{x}_1$ at $(\boldsymbol{x}_1, \boldsymbol{\ell})$ gives $\boldsymbol{\ell}^\top H_j$. Thus, by properly choosing $\boldsymbol{\ell}$ we can get different linear combinations of the rows of $H_j = [\boldsymbol{h}_{j1}\, \boldsymbol{h}_{j2}\, \boldsymbol{h}_{j3}]^\top$. If we choose $\boldsymbol{\ell}_{12} = (y_2, -x_2, 0)$, $\boldsymbol{\ell}_{23} = (0, 1, -y_2)$ and $\boldsymbol{\ell}_{31} = (1, 0, -x_2)$ as three lines passing through $\boldsymbol{x}_2 = (x_2, y_2, 1)$, we can compute the vectors

$$\boldsymbol{g}_{12} \sim y_2 \boldsymbol{h}_{j1} - x_2 \boldsymbol{h}_{j2}, \quad \boldsymbol{g}_{23} \sim \boldsymbol{h}_{j2} - y_2 \boldsymbol{h}_{j3} \quad \text{and} \quad \boldsymbol{g}_{31} \sim \boldsymbol{h}_{j1} - x_1 \boldsymbol{h}_{j3} \tag{8.36}$$

from the derivatives of p_n at $(\boldsymbol{x}_1, \boldsymbol{\ell}_{12})$, $(\boldsymbol{x}_1, \boldsymbol{\ell}_{23})$ and $(\boldsymbol{x}_1, \boldsymbol{\ell}_{31})$, respectively.

For step 2, notice that $\boldsymbol{g}_{12}$ lives in the plane spanned by $\boldsymbol{h}_{j1}$ and $\boldsymbol{h}_{j2}$, whose normal vector is $\boldsymbol{b}_{12j} = \boldsymbol{h}_{j1} \times \boldsymbol{h}_{j2}$. Therefore, if we evaluate $\boldsymbol{g}_{12}$ at all the given point correspondences, we obtain a set of N vectors $\{\boldsymbol{g}_{12i}\}_{i=1}^N$ lying in a union of n planes with normal vectors $\{\boldsymbol{b}_{12j}\}_{j=1}^n$. Similarly, the vectors $\{\boldsymbol{g}_{23i}\}_{i=1}^N$ and $\{\boldsymbol{g}_{31i}\}_{i=1}^N$ lie in a union of n planes with normal vectors $\{\boldsymbol{b}_{23j} = \boldsymbol{h}_{j2} \times \boldsymbol{h}_{j3}\}_{j=1}^n$ and $\{\boldsymbol{b}_{31j} = \boldsymbol{h}_{j3} \times \boldsymbol{h}_{j1}\}_{j=1}^n$, respectively. In principle we can obtain the vectors $\{\boldsymbol{b}_{12j}\}_{j=1}^n$, $\{\boldsymbol{b}_{23j}\}_{j=1}^n$ and $\{\boldsymbol{b}_{31j}\}_{j=1}^n$ by applying the GPCA algorithm (Algorithm 3.4) to each one of the three sets of points $\{\boldsymbol{g}_{12i}\}$, $\{\boldsymbol{g}_{23i}\}$ and $\{\boldsymbol{g}_{31i}\}$ separately. However, if we do so we would not know which $\boldsymbol{b}_{12j}$ correspond to which $\boldsymbol{b}_{13j}$. Exercise 3.5 shows how to resolve this difficulty by exploiting the fact that the correspondences among the data points $\{\boldsymbol{g}_{12i}\}$, $\{\boldsymbol{g}_{23i}\}$ and $\{\boldsymbol{g}_{31i}\}$ are known, because these three vectors are computed as a triplet associated with the same point correspondence $(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})$. The main idea is to compute three polynomials p_{12}, p_{23} and p_{31} fitting the points $\{\boldsymbol{g}_{12i}\}$, $\{\boldsymbol{g}_{23i}\}$ and $\{\boldsymbol{g}_{31i}\}$, respectively, and then obtain the normal vector for the jth group from the gradients of these polynomials. In order for the normal vectors $\boldsymbol{b}_{12i}$, $\boldsymbol{b}_{23i}$ and $\boldsymbol{b}_{31i}$ to correspond, we can choose the points at which the gradients by minimizing the sum of the squared distances to all hyperplanes, i.e., we compute the normal vectors as

$$\boldsymbol{b}_{12j} \sim \nabla p_{12}(\boldsymbol{g}_{12i_j}), \quad \boldsymbol{b}_{23j} \sim \nabla p_{23}(\boldsymbol{g}_{23i_j}) \quad \text{and} \quad \boldsymbol{b}_{31j} \sim \nabla p_{31}(\boldsymbol{g}_{31i_j}), \tag{8.37}$$

where

$$i_j = \operatorname*{arg\,min}_{j=1,\dots,N} \frac{\frac{|p_{12}(\boldsymbol{g}_{12j})|}{\|\nabla p_{12}(\boldsymbol{g}_{12j})\|}}{\prod_{i+1}^{k=n} |\boldsymbol{b}_{12k}^T \boldsymbol{g}_{12j}|} + \frac{\frac{|p_{23}(\boldsymbol{g}_{23j})|}{\|\nabla p_{23}(\boldsymbol{g}_{23j})\|}}{\prod_{i+1}^{k=n} |\boldsymbol{b}_{23k}^T \boldsymbol{g}_{23j}|} + \frac{\frac{|p_{31}(\boldsymbol{g}_{31j})|}{\|\nabla p_{31}(\boldsymbol{g}_{31j})\|}}{\prod_{i+1}^{k=n} |\boldsymbol{b}_{31k}^\top \boldsymbol{g}_{31j}|}. \tag{8.38}$$

For step 3, given $\boldsymbol{b}_{12} = \boldsymbol{h}_{i1} \times \boldsymbol{h}_{i2}$, $\boldsymbol{b}_{23} = \boldsymbol{h}_{i2} \times \boldsymbol{h}_{i3}$ and $\boldsymbol{b}_{31} = \boldsymbol{h}_{i3} \times \boldsymbol{h}_{i1}$, we can immediately obtain the rows of H_j up to a scale factor as

$$\boldsymbol{h}_{i1} \sim \tilde{\boldsymbol{h}}_{i1} \doteq \boldsymbol{b}_{12} \times \boldsymbol{b}_{31}, \boldsymbol{h}_{i2} \sim \tilde{\boldsymbol{h}}_{i2} \doteq \boldsymbol{b}_{23} \times \boldsymbol{b}_{12}, \boldsymbol{h}_{i3} \sim \tilde{\boldsymbol{h}}_{i3} \doteq \boldsymbol{b}_{31} \times \boldsymbol{b}_{23}. \tag{8.39}$$

For step 4, we know that $\boldsymbol{x}_2 \sim H_j \boldsymbol{x}_1$. Therefore, we can obtain the n homograhies as

$$H_j = \begin{bmatrix} \frac{x_2}{\tilde{\boldsymbol{h}}_{j1}^\top \boldsymbol{x}_1} \tilde{\boldsymbol{h}}_{j1} & \frac{y_2}{\tilde{\boldsymbol{h}}_{j2}^\top \boldsymbol{x}_1} \tilde{\boldsymbol{h}}_{j2} & \frac{1}{\tilde{\boldsymbol{h}}_{j3}^\top \boldsymbol{x}_1} \tilde{\boldsymbol{h}}_{j3} \end{bmatrix}^\top, \quad j = 1, \dots, n. \tag{8.40}$$

Algorithm 8.2 summarizes the algorithm for segmenting homography matrices. Table 8.2 gives the minimum number of point correspondences required by the algorithm as a function of the number of motions.

Experimental results

We applied Algorithm 8.2 to segment two frames of a 2048×1536 video sequence, shown in Figure 8.7(a)-(b), with two moving objects – a cube and a checkerboard. Notice that although there are only *two* rigid-body motions, the scene contains *three* different homographies, each one associated with each one of the three visible planar structures. Furthermore, notice that the top side of the

Algorithm 8.2 (Segmentation of Homography Matrices).

Given two perspective views $\{(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})\}_{i=1}^{N}$ of a set of N 3-D points whose motion can be modeled with n homography matrices $\{H_j\}_{j=1}^{n}$, recover the number of independent motions n, the homography matrix $\{H_j\}_{j=1}^{n}$ associated with each motion, and the motion model associated with each image pair as follows:

1: **Number of motions:** Compute two lines $(\boldsymbol{\ell}_{21i}, \boldsymbol{\ell}_{22i})$ passing through $\boldsymbol{x}_{2i}$. Form the embedded data matrix of degree $j \geq 1$, $\boldsymbol{V}_i^{\mathcal{H}} \in \mathbb{R}^{N(j+1) \times M_n(3)}$, as in (8.34). Compute the number of motions from (8.35), or else from

$$n = \arg\min_{j \geq 1} \frac{\sigma^2_{M_j(3)^2}(\boldsymbol{V}_j^{\mathcal{H}})}{\sum_{k=1}^{M_j(3)^2-1} \sigma_k^2(\boldsymbol{V}_j^{\mathcal{H}})} + \mu M_j(3)^2. \tag{8.41}$$

2: **Multibody homography matrix:** Compute the multibody homography matrix $\mathcal{H}$ as the least-squares solution to the linear system $\boldsymbol{V}_n^{\mathcal{H}} \mathrm{vec}(\mathcal{H}) = \boldsymbol{0}$ in (8.34), and let

$$p_n(\boldsymbol{x}_1, \boldsymbol{\ell}) = \nu_n(\boldsymbol{\ell})^\top \mathcal{H} \nu_n(\boldsymbol{x}_1).$$

3: **Homography matrices:**

1. For all $i = 1, \ldots, N$, let $\boldsymbol{\ell}_{12i} = (y_{2i}, -x_{2i}, 0)^\top$, $\boldsymbol{\ell}_{23i} = (0, 1, y_{2i})^\top$ and $\boldsymbol{\ell}_{31i} = (1, 0, -x_{2i})^\top$ be three lines passing through $\boldsymbol{x}_{2i} = (x_{2i}, y_{2i}, 1)^\top$. Compute a linear combination of rows 1 & 2, 2 & 3, and 3 & 1 of the homography matrix at each point correspondence as

$$\boldsymbol{g}_{12i} = \frac{\partial p_n}{\partial \boldsymbol{x}_1}(\boldsymbol{x}_{1i}, \boldsymbol{\ell}_{12i}); \; \boldsymbol{g}_{23i} = \frac{\partial p_n}{\partial \boldsymbol{x}_1}(\boldsymbol{x}_{1i}, \boldsymbol{\ell}_{23i}); \; \boldsymbol{g}_{31i} = \frac{\partial p_n}{\partial \boldsymbol{x}_1}(\boldsymbol{x}_{1i}, \boldsymbol{\ell}_{31i}).$$

2. Solve for the coefficients $\boldsymbol{c}_{jk}$ of the polynomials $p_{12}(\boldsymbol{g}) = \boldsymbol{c}_{12}^\top \nu_n(\boldsymbol{g})$, $p_{23}(\boldsymbol{g}) = \boldsymbol{c}_{23}^\top \nu_n(\boldsymbol{g})$ and $p_{31}(\boldsymbol{g}) = \boldsymbol{c}_{31}^\top \nu_n(\boldsymbol{g})$, from the linear system

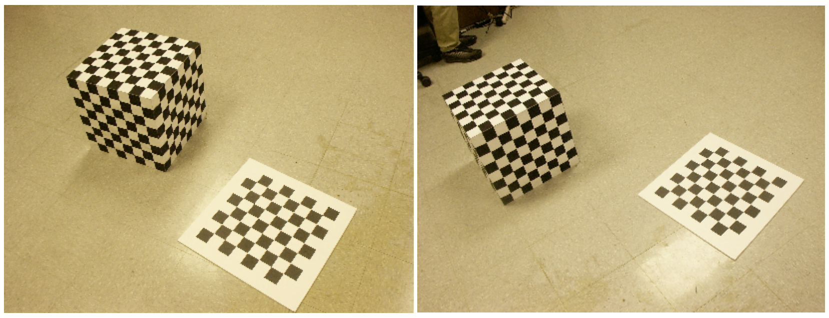
$$[\nu_n(\boldsymbol{g}_{jk1}), \nu_n(\boldsymbol{g}_{jk2}), \ldots, \nu_n(\boldsymbol{g}_{jkN})]^\top \boldsymbol{c}_{jk} = \boldsymbol{0}.$$

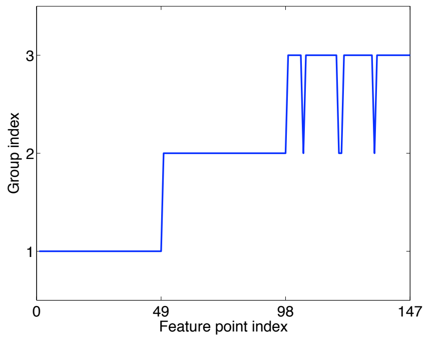
3. Compute the homography matrices from the cross products of the gradients of p_{12}, p_{23} and p_{31} as follows:

```
for all $j = n : 1$ do
    $i_j = \arg\min_{j=1,\ldots,N} \frac{\frac{|p_{12}(\boldsymbol{g}_{12j})|}{\|\nabla p_{12}(\boldsymbol{g}_{12j})\|}}{\prod_{j+1}^{k=n} |\boldsymbol{b}_{12k}^\top \boldsymbol{g}_{12j}|} + \frac{\frac{|p_{23}(\boldsymbol{g}_{23j})|}{\|\nabla p_{23}(\boldsymbol{g}_{23j})\|}}{\prod_{j+1}^{k=n} |\boldsymbol{b}_{23k}^T \boldsymbol{g}_{23j}|} + \frac{\frac{|p_{31}(\boldsymbol{g}_{31j})|}{\|\nabla p_{31}(\boldsymbol{g}_{31j})\|}}{\prod_{j+1}^{k=n} |\boldsymbol{b}_{31k}^\top \boldsymbol{g}_{31j}|};$
    $\boldsymbol{b}_{12j} = \nabla p_{12}(\boldsymbol{g}_{12i_j}); \quad \boldsymbol{b}_{23j} = \nabla p_{23}(\boldsymbol{g}_{23i_j}); \quad \boldsymbol{b}_{31j} = \nabla p_{31}(\boldsymbol{g}_{31i_j});$
    $H_j = \left[ \frac{x_{2i_j}(\boldsymbol{b}_{12j} \times \boldsymbol{b}_{31j})}{\det([\boldsymbol{b}_{12j} \; \boldsymbol{b}_{31j} \; \boldsymbol{x}_{1i_j}])} \quad \frac{y_{2i_j}(\boldsymbol{b}_{23j} \times \boldsymbol{b}_{12j})}{\det([\boldsymbol{b}_{23j} \; \boldsymbol{b}_{12j} \; \boldsymbol{x}_{1i_j}])} \quad \frac{(\boldsymbol{b}_{31j} \times \boldsymbol{b}_{23j})}{\det([\boldsymbol{b}_{31j} \; \boldsymbol{b}_{23j} \; \boldsymbol{x}_{1i_j}])} \right]^\top.$
end for
```

4: **Feature segmentation:** Assign the image pair $(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i})$ to group j if $j = \arg\min_{k=1,\ldots,n} \|\boldsymbol{x}_2 - \frac{H_k \boldsymbol{x}_1}{e_3^\top H_k \boldsymbol{x}_1}\|^2$.

cube and the checkerboard have approximately the same normals. We manually tracked a total of $N = 147$ features: 98 in the cube (49 in each of the two visible sides) and 49 in the checkerboard. We applied Algorithm 8.2 to segment the image data and obtained a 97% of correct classification, as shown in Figure 8.7(c).


(a) First frame (b) Second frame



(c) Feature segmentation

Figure 8.7. Segmenting two different rigid-body motions, a cube and a plane, according to three different homography models corresponding to the three planes in the scene.

We then added zero-mean Gaussian noise with standard deviation between 0 and 1 pixels to the features, after rectifying the features in the second view in order to simulate the noise free case. Figure 8.7(c) shows the mean percentage of correct classification for 1000 trials per level of noise. The percentage of correct classification of our algorithm is between 80% and 100%, which gives a very good initial estimate for any of the existing iterative/optimization/EM based motion segmentation schemes.

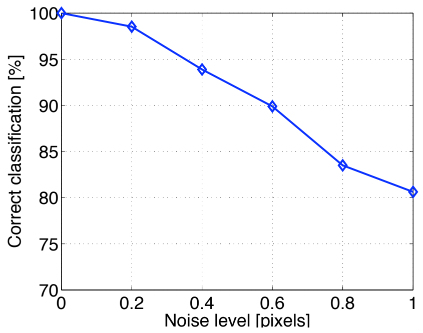


Figure 8.8. Percentage of correct classification as a function of noise synthetically added to the point correspondences of the scene in Figure 8.7.

8.5 Segmentation of Trilinear Motion Models

In this section, we consider the problem of segmenting n 3-D rigid-body motions $\{(R_j, T_j) \in SE(3)\}_{j=1}^n$ from point correspondences in three perspective views. As shown in Section 8.1.4, in this case the motion of the each object relative to the camera among the three views can be modeled as a mixture of trifocal tensors $\{\mathcal{T}_j \in \mathbb{R}^{3\times3\times3}\}_{j=1}^n$ relating a point, a line and a line in the first, second and third views. To avoid degenerate situations, we assume that the 3-D scene is nonplanar and that the trifocal tensors are different from each other (up to a scale factor).

8.5.1 The Multibody Trifocal Tensor

The multibody trilinear constraint and the multibody trifocal tensor

Let $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2 \leftrightarrow \boldsymbol{x}_3$ be an arbitrary point correspondence. Then, there exists a trifocal tensor $\mathcal{T}_j$ that relates the point in the first view $\boldsymbol{x}_1 = (x_{11}, x_{12}, x_{13})^\top$, a line in the second view $\boldsymbol{\ell}_2 = (\ell_{21}, \ell_{22}, \ell_{23})^\top$ passing through $\boldsymbol{x}_2$ and a line in the third view $\boldsymbol{\ell}_3 = (\ell_{31}, \ell_{32}, \ell_{33})^\top$ passing through $\boldsymbol{x}_3$ via the *trilinear constraint*:

$$\mathcal{T}_j(\boldsymbol{x}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3) = \sum_{p,q,r} \mathcal{T}_{j,pqr} x_{1p} \ell_{2q} \ell_{3r} = 0. \tag{8.42}$$

Therefore, regardless of which motion is associated with the correspondence, the following constraint must be satisfied by the number of independent motions n, the trifocal tensors $\{\mathcal{T}_j\}_{j=1}^n$ and the point-line-line correspondence $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{\ell}_2 \leftrightarrow \boldsymbol{\ell}_3$

$$\prod_{j=1}^{n} \mathcal{T}_j(\boldsymbol{x}_1, \boldsymbol{\ell}_2, \boldsymbol{\ell}_3) = 0. \tag{8.43}$$

This multibody constraint is a homogeneous polynomial of degree n in each of $\boldsymbol{x}_1$, $\boldsymbol{\ell}_2$ or $\boldsymbol{\ell}_3$. Thus, we can write it as a sum of monomials of degree n in each of $\boldsymbol{x}_1$, $\boldsymbol{\ell}_2$ and $\boldsymbol{\ell}_3$. By collecting the coefficients of these monomial in a 3-dimensional tensor $\mathcal{T} \in \mathbb{R}^{M_n(3)\times M_n(3)\times M_n(3)}$, we can write the constraint (8.43) as

$$\mathcal{T}(\nu_n(\boldsymbol{x}_1), \nu_n(\boldsymbol{\ell}_2), \nu_n(\boldsymbol{\ell}_3)) = 0. \tag{8.44}$$

We call the array $\mathcal{T}$ the *multibody trifocal tensor* and equation (8.44) the *multibody trilinear constraint*, as they are natural generalizations of the trifocal tensor and the trilinear constraint, respectively, valid for $n = 1$.

Computing the number of motions and the multibody trifocal tensor

Notice that, although (8.44) has degree n in the entries of $\boldsymbol{x}_1$, $\boldsymbol{\ell}_2$ and $\boldsymbol{\ell}_3$, it is in fact *linear* in the entries of $\nu_n(\boldsymbol{x})$, $\nu_n(\boldsymbol{\ell}_2)$ and $\nu_n(\boldsymbol{\ell}_3)$. Hence, given a point-line-line correspondence $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{\ell}_2 \leftrightarrow \boldsymbol{\ell}_3$, we can compute the entries of the vectors $\nu_n(\boldsymbol{x})$, $\nu_n(\boldsymbol{\ell}_2)$ and $\nu_n(\boldsymbol{\ell}_3)$ and use the multibody trilinear constraint (8.44) to obtain a linear relationship in the entries of $\mathcal{T}$. Therefore, we may estimate $\mathcal{T}$ linearly from $M_n(3)^3 - 1 \sim O(n^6)$ point-line-line correspondences. That is 26 correspondences for one motion, 215 for two motions, 999 for three motions, etc.

Fortunately, as in the case of $n = 1$ motion, one may significantly reduce the data requirements by working with point-point-point correspondences $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2 \leftrightarrow \boldsymbol{x}_3$. Since each point in the second view $\boldsymbol{x}_2$ gives two independent lines $\boldsymbol{\ell}_{21}$ and $\boldsymbol{\ell}_{22}$ passing through it and each point in the third view $\boldsymbol{x}_3$ gives two independent lines $\boldsymbol{\ell}_{31}$ and $\boldsymbol{\ell}_{32}$ passing through it, a naive calculation would give $2^2 = 4$ constraints per point-point-point correspondence. However, due to the algebraic properties of the Veronese map, each correspondence provides in general $(n+1)^2$ independent constraints on the multibody trifocal tensor.

To see this, remember from Section 8.1.4 that the trilinear constraint is satisfied by *all* lines $\boldsymbol{\ell}_2 = \alpha\boldsymbol{\ell}_{21} + \boldsymbol{\ell}_{22}$ and $\boldsymbol{\ell}_3 = \beta\boldsymbol{\ell}_{31} + \boldsymbol{\ell}_{32}$ passing through $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$, respectively. Therefore, for all $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$ we must have

$$\prod_{j=1}^{n} T_j(\boldsymbol{x}_1, \alpha\boldsymbol{\ell}_{21} + \boldsymbol{\ell}_{22}, \beta\boldsymbol{\ell}_{31} + \boldsymbol{\ell}_{32}) = 0. \tag{8.45}$$

The above equation, viewed as a function of α, is a polynomial of degree n, hence its $n+1$ coefficients must be zero. Each coefficient is in turn a polynomial of degree n in β, whose $n+1$ coefficients must be zero. Therefore, each point-point-point correspondence gives $(n+1)^2$ constraints on the multibody trifocal tensor $\mathcal{T}$, and we need only $(M_n(3)^3 - 1)/(n+1)^2 \sim O(n^4)$ point-point-point correspondences to estimate $\mathcal{T}$. That is 7, 24 and 63 correspondences for one, two and three motions, respectively. This reduction on the number of required correspondences represents a significant improvement, not only with respect to the case of point-line-line correspondences, as explained above, but also with respect to the case of two perspective views. As discussed in Section 8.4.1, one needs $M_n(3)^2 - 1$ point-point correspondences for linearly estimating the multibody fundamental matrix $\mathcal{F}$, i.e., 8, 35 and 99 correspondences for one, two and three motions, respectively, as shown in Table 8.2.

Given a correspondence $\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2 \leftrightarrow \boldsymbol{x}_3$, we generate the $(n+1)^2$ linear equations in the entries of $\mathcal{T}$ by choosing $\boldsymbol{\ell}_{21}$, $\boldsymbol{\ell}_{22}$, $\boldsymbol{\ell}_{31}$ and $\boldsymbol{\ell}_{32}$ passing through

$\boldsymbol{x}_2$ and $\boldsymbol{x}_3$, respectively, and then computing the coefficient of $\alpha^i\beta^j$ in (8.45). As shown in Exercise 8.3, these $(n+1)^2$ coefficients are given by

$$\begin{aligned}\mathcal{T}(\nu_n(\boldsymbol{x}_1), &f_j(\boldsymbol{\ell}_{21},\boldsymbol{\ell}_{22}), f_k(\boldsymbol{\ell}_{31},\boldsymbol{\ell}_{32})) \\ &= \big(\nu_n(\boldsymbol{x}_1)\otimes f_j(\boldsymbol{\ell}_{21},\boldsymbol{\ell}_{22})\otimes f_k(\boldsymbol{\ell}_{31},\boldsymbol{\ell}_{32})\big)^\top \mathrm{vec}(\mathcal{T}),\ \ j,k=1,\ldots,n,\end{aligned}$$

where $\mathrm{vec}(\mathcal{T})\in\mathbb{R}^{M_n(3)^3}$ is the stack of all the entries of $\mathcal{T}$ and f_j is defined in (8.32). Therefore, we can solve for $\mathcal{T}$ from the linear system

$$\boldsymbol{V}_n^{\mathcal{T}}\mathrm{vec}(\mathcal{T}) = \boldsymbol{0}, \tag{8.46}$$

where the rows of the matrix $\boldsymbol{V}_n^{\mathcal{T}}\in\mathbb{R}^{P(n+1)^2\times M_n(3)^3}$ are of the form $\nu_n(\boldsymbol{x}_{1i})\otimes f_j(\boldsymbol{\ell}_{21i},\boldsymbol{\ell}_{22i})\otimes f_k(\boldsymbol{\ell}_{31i},\boldsymbol{\ell}_{32i})$, for $j,k=1,\ldots,n$ and $i=1,\ldots,N$.

Finally, similarly to (**??**), (8.23) and (8.35), the number of trifocal tensors can be computed from

$$n \doteq \min\{j : \mathrm{rank}(\boldsymbol{V}_j^{\mathcal{T}}) = M_j(3)^3-1\}. \tag{8.47}$$

8.5.2 Segmenting Trifocal Tensors

Computing the epipolar lines

Given the trifocal tensor T, it is well known how to compute the epipolar lines $\boldsymbol{\ell}_2(\boldsymbol{x}_1)$ and $\boldsymbol{\ell}_3(\boldsymbol{x}_1)$ in the 2nd and 3rd views associated with a point $\boldsymbol{x}_1$ in the 1st view. For example, as shown in Section 8.1.4, the epipolar line in the second view $\boldsymbol{\ell}_2(\boldsymbol{x}_1)$ must satisfy the relationship

$$\forall \boldsymbol{\ell}_3\in\mathbb{R}^3\ \ T(\boldsymbol{x}_1,\boldsymbol{\ell}_2(\boldsymbol{x}_1),\boldsymbol{\ell}_3)=0. \tag{8.48}$$

In the case of multiple motions, we are faced with the more challenging problem of computing the epipolar lines $\boldsymbol{\ell}_2(\boldsymbol{x}_1)$ and $\boldsymbol{\ell}_3(\boldsymbol{x}_1)$ without knowing the individual trifocal tensors $\{T_j\}_{j=1}^n$ or the segmentation of the correspondences. The question is then how to compute such epipolar lines from the multibody trifocal tensor $\mathcal{T}$. To this end, notice that with each point in the first view $\boldsymbol{x}_1$ we can associate n epipolar lines corresponding to the n motions between the 1st and 2nd views. If $\boldsymbol{\ell}_2(\boldsymbol{x}_1)$ is an epipolar line of $\boldsymbol{x}_1$ according to the jth motion, then from equation (8.48) we have that for all $\boldsymbol{\ell}_3\in\mathbb{R}^3$, $T_j(\boldsymbol{x}_1,\boldsymbol{\ell}_2(\boldsymbol{x}_1),\boldsymbol{\ell}_3)=0$. This implies that

$$\forall \boldsymbol{\ell}_3\in\mathbb{R}^3 \prod_{j=1}^n T_j(\boldsymbol{x}_1,\boldsymbol{\ell}_2(\boldsymbol{x}_1),\boldsymbol{\ell}_3) = \mathcal{T}(\nu_n(\boldsymbol{x}_1),\nu_n(\boldsymbol{\ell}_2(\boldsymbol{x}_1)),\nu_n(\boldsymbol{\ell}_3)=0. \tag{8.49}$$

As this equation holds for *any* of the n epipolar lines, the question of determining *the* epipolar line of a point $\boldsymbol{x}_1$ is not well posed as such, because *the* epipolar line depends on which of the n motions the point $\boldsymbol{x}_1$ belongs to, which cannot be determined without additional information. We therefore pose the question a little differently, and suppose that we know the point $\boldsymbol{x}_2$ in the second view corresponding to $\boldsymbol{x}_1$. Since the epipolar line $\boldsymbol{\ell}_2(\boldsymbol{x}_1)$ must of course pass through $\boldsymbol{x}_2$, we can parameterize it as

$$\boldsymbol{\ell}_2(\boldsymbol{x}_1) = \alpha\boldsymbol{\ell}_{21} + \boldsymbol{\ell}_{22}, \tag{8.50}$$

where, as before, $\boldsymbol{\ell}_{21}$ and $\boldsymbol{\ell}_{22}$ are two different lines passing through $\boldsymbol{x}_2$. Replacing (8.50) in equation (8.49) gives

$$\forall \boldsymbol{\ell}_3 \in \mathbb{R}^3 \quad \mathcal{T}(\nu_n(\boldsymbol{x}_1), \nu_n(\alpha\boldsymbol{\ell}_{21} + \boldsymbol{\ell}_{22}), \nu_n(\boldsymbol{\ell}_3)) = 0. \tag{8.51}$$

As $\boldsymbol{\ell}_3$ ranges over all of $\mathbb{R}^3$, this gives a total of up to $M_n(3)$ linearly independent equations. Each one of such equations is a polynomial of degree n in α. These polynomials must have a common root α^* for which all the polynomials vanish. The epipolar line of $\boldsymbol{x}_1$ in the second view is then $\boldsymbol{\ell}_2(\boldsymbol{x}_1) = \alpha^*\boldsymbol{\ell}_{21} + \boldsymbol{\ell}_{22}$. The epipolar line of $\boldsymbol{x}_1$ in the third view can be obtained in an analogous fashion.

We may apply this process to all N correspondences $\{\boldsymbol{x}_{1i} \leftrightarrow \boldsymbol{x}_{2i} \leftrightarrow \boldsymbol{x}_{3i}\}_{i=1}^N$ to obtain the set of all N epipolar lines in the second and third views, $\{\boldsymbol{\ell}_2(\boldsymbol{x}_{1i})\}_{i=1}^N$ and $\{\boldsymbol{\ell}_3(\boldsymbol{x}_{1i})\}_{i=1}^N$, according to their individual motion models. Notice, again, that this is done from the multibody trifocal tensor $\mathcal{T}$ only, without knowing the individual trifocal tensors or the segmentation of the correspondences.

Computing the epipoles

As shown in Section 8.1.4, in the case of one rigid-body motion, the epipoles in the second and third views, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$, can be computed from the epipolar lines in the second and third views, $\{\boldsymbol{\ell}_2(\boldsymbol{x}_{1i})\}_{i=1}^N$ and $\{\boldsymbol{\ell}_3(\boldsymbol{x}_{1i})\}_{i=1}^N$, respectively, by solving the linear systems

$$\boldsymbol{e}_2^\top[\boldsymbol{\ell}_2(\boldsymbol{x}_{11}), \ldots, \boldsymbol{\ell}_2(\boldsymbol{x}_{1N})] = \boldsymbol{0}^\top \text{ and } \boldsymbol{e}_3^\top[\boldsymbol{\ell}_3(\boldsymbol{x}_{11}), \ldots, \boldsymbol{\ell}_3(\boldsymbol{x}_{1N})] = \boldsymbol{0}^\top. \tag{8.52}$$

In the case of n motions there exist n epipole pairs, $\{(\boldsymbol{e}_{2j}, \boldsymbol{e}_{3j})\}_{j=1}^n$, where $\boldsymbol{e}_{2j}$ and $\boldsymbol{e}_{3j}$ are epipoles in the second and third views corresponding to the jth motion. Given a set of correspondences $\{\boldsymbol{x}_{2i} \leftrightarrow \boldsymbol{x}_{3i} \leftrightarrow \boldsymbol{x}_{3i}\}$ we may compute the multibody trifocal tensor $\mathcal{T}$ and determine the epipolar lines $\boldsymbol{\ell}_2(\boldsymbol{x}_{1i})$ and $\boldsymbol{\ell}_3(\boldsymbol{x}_{1i})$ associated with each correspondence by the method described in the previous subsection. Then, for each pair of epipolar lines $\big(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}), \boldsymbol{\ell}_3(\boldsymbol{x}_{1i})\big)$ there exists an epipole pair $(\boldsymbol{e}_{2j}, \boldsymbol{e}_{3j})$ such that

$$\boldsymbol{e}_{2j}^\top\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}) = 0 \quad \text{and} \quad \boldsymbol{e}_{3j}^\top\boldsymbol{\ell}_3(\boldsymbol{x}_{1i}) = 0. \tag{8.53}$$

Therefore, the problem of finding the epipole pairs $\{(\boldsymbol{e}_{2j}, \boldsymbol{e}_{3j})\}_{j=1}^n$ is equivalent to one of segmenting two sets of points lying in two collections of hyperplanes whose normal vectors are the epipole pairs.

Exercise 3.5 provides a solution to this problem based on a simple extension of the GPCA (Algorithm 3.4). The first step is to fit two polynomials,

$$\begin{aligned} p_2(\boldsymbol{\ell}_2) &= \prod_{j=1}^n (\boldsymbol{e}_{2j}^\top\boldsymbol{\ell}_2) = \boldsymbol{c}_2^\top\nu_n(\boldsymbol{\ell}_2) = 0, \\ p_3(\boldsymbol{\ell}_3) &= \prod_{j=1}^n (\boldsymbol{e}_{3j}^\top\boldsymbol{\ell}_3) = \boldsymbol{c}_3^\top\nu_n(\boldsymbol{\ell}_3) = 0, \end{aligned} \tag{8.54}$$

to the epipolar lines $\{\boldsymbol{\ell}_2(\boldsymbol{x}_{1i})\}_{i=1}^N$ and $\{\boldsymbol{\ell}_3(\boldsymbol{x}_{1i})\}_{i=1}^N$, respectively. Similarly to (8.52), we may find the coefficients of p_2 and p_3 by solving the linear systems

$$\begin{aligned} \boldsymbol{c}_2^\top[\nu_n(\boldsymbol{\ell}_2(\boldsymbol{x}_{11})),\dots,\nu_n(\boldsymbol{\ell}_2(\boldsymbol{x}_{1N}))] &= \mathbf{0}^\top, \\ \boldsymbol{c}_3^\top[\nu_n(\boldsymbol{\ell}_3(\boldsymbol{x}_{11})),\dots,\nu_n(\boldsymbol{\ell}_3(\boldsymbol{x}_{1N}))] &= \mathbf{0}^\top. \end{aligned} \tag{8.55}$$

The second step is to compute the epipoles as the gradients of these polynomials at a collection of n pairs of epipolar lines $\{(\boldsymbol{\ell}_{2j},\boldsymbol{\ell}_{3j})\}_{j=1}^n$ passing through each one of the epipole pairs, i.e.,

$$\boldsymbol{e}_{2j}\sim\nabla p_2(\boldsymbol{\ell}_{2j}) \quad \text{and} \quad \boldsymbol{e}_{3j}\sim\nabla p_3(\boldsymbol{\ell}_{3j}), \quad j=1,\dots,n. \tag{8.56}$$

The pairs of epipolar lines $\{(\boldsymbol{\ell}_{2j},\boldsymbol{\ell}_{3j})\}_{j=1}^n$ are chosen as the n pairs of epipolar lines in $\{(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}),\boldsymbol{\ell}_3(\boldsymbol{x}_{1i}))\}_{i=1}^N$ that minimize a certain distance to their respective epipoles. More specifically, for $j=n,\dots,2,1$ set $\boldsymbol{\ell}_{2j}=\boldsymbol{\ell}_2(\boldsymbol{x}_{1i_j})$ and $\boldsymbol{\ell}_{2j}=\boldsymbol{\ell}_2(\boldsymbol{x}_{1i_j})$, where

$$i_j = \arg\min_{i=1,\dots,N} \frac{\frac{p_2(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))^2}{\|\nabla p_2(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))\|^2}}{\prod_{k=j+1}^n(\boldsymbol{e}_{2k}^\top\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))^2} + \frac{\frac{p_3(\boldsymbol{\ell}_3(\boldsymbol{x}_{1i}))^2}{\|\nabla p_3(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))\|^2}}{\prod_{k=j+1}^n(\boldsymbol{e}_{3k}^\top\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))^2}. \tag{8.57}$$

Computing the trifocal tensors

Once the n epipole pairs $\{\boldsymbol{e}_{2j},\boldsymbol{e}_{3j})\}_{j=1}^n$ have been computed, we can segment the data into n groups by assigning the image pair $(\boldsymbol{x}_{1i},\boldsymbol{x}_{2i})$ to group j if

$$j = \arg\min_{k=1,\dots n} (\boldsymbol{e}_{2k}^\top\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))^2 + (\boldsymbol{e}_{3k}^\top\boldsymbol{\ell}_3(\boldsymbol{x}_{1i}))^2 \tag{8.58}$$

provided that the n epipoles pairs are different. Given the segmentation of the correspondences, one may obtain trifocal tensors, fundamental matrices and camera matrices using the algebraic methods described in Section 8.1.4.

Algorithm summary

Algorithm 8.3 summarizes the main steps of the algorithm for segmenting trifocal tensors described in this section. Table 8.2 gives the minimum number of point correspondences required by the algorithm as a function of the number of motions.

Table 8.2. Number of correspondences required to linearly solve for the different multibody motion models.

	Correspondence	1	2	3
Multibody fundamental matrix $\mathcal{F}$	point-point	8	35	99
Multibody homography matrix $\mathcal{H}$	point-point	4	12	25
Multibody trifocal tensor $\mathcal{T}$	point-point-point	7	24	63
Multibody trifocal tensor $\mathcal{T}$	point-line-line	26	215	999

Algorithm 8.3 (Segmentation of Trifocal Tensors).

Given a set of points $\{(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i}, \boldsymbol{x}_{3i})\}_{i=1}^{N}$ corresponding to N points undergoing n different rigid-body motions relative to a moving perspective camera, recover the number of independent motions n, the trifocal tensors $\{T_j\}_{j=1}^{n}$ associated with each motion, and the motion associated with each correspondence as follows:

1: **Number of motions:** Compute two lines $(\boldsymbol{\ell}_{21i}, \boldsymbol{\ell}_{22i})$ passing through $\boldsymbol{x}_{2i}$, and two lines $(\boldsymbol{\ell}_{31i}, \boldsymbol{\ell}_{32i})$ passing through $\boldsymbol{x}_{3i}$. Form the embedded data matrix of degree $j = 1, \ldots, n$, $\boldsymbol{V}_j^{\mathcal{T}} \in \mathbb{R}^{N(j+1)^2 \times M_j(3)^3}$, as defined in (8.46). Compute the number of independent motions n from

$$n = \underset{j \geq 1}{\arg\min} \frac{\sigma^2_{M_j(3)^3}(\boldsymbol{V}_j^{\mathcal{T}})}{\sum_{k=1}^{M_j(3)^3-1} \sigma_k^2(\boldsymbol{V}_j^{\mathcal{T}})} + \mu M_j(3)^3. \tag{8.59}$$

2: **Multibody trifocal tensor:** Compute the multibody trifocal tensor $\mathcal{T}$ as the least-squares solution to the linear system $\boldsymbol{V}_n^{\mathcal{T}} \mathrm{vec}(\mathcal{T}) = \mathbf{0}$ in (8.46).

3: **Epipolar lines:** For all $i = 1, \ldots, N$, compute the epipolar lines of $\boldsymbol{x}_{1i}$ in the second and third views, $\boldsymbol{\ell}_2(\boldsymbol{x}_{1i})$ and $\boldsymbol{\ell}_3(\boldsymbol{x}_{1i})$, as follows:

   1. Let $q_k(\alpha) = \sum_{j=0}^{n} \mathcal{T}(\nu_n(\boldsymbol{x}_{1i}), f_j(\boldsymbol{\ell}_{21i}, \boldsymbol{\ell}_{22i}), e_k)\alpha^j$, for all $k = 1, \ldots, M_n(3)$. Compute the common root α^* of these $M_n(3)$ polynomials as the value of α that minimizes $q(\alpha) = \sum_{k=1}^{M_n(3)} q_k(\alpha)^2$. The epipolar line of $\boldsymbol{x}_{1i}$ in the second view is $\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}) = \alpha^* \boldsymbol{\ell}_{21i} + \boldsymbol{\ell}_{22i}$.
   2. Compute the epipolar line of $\boldsymbol{x}_{1i}$ in the third view as $\boldsymbol{\ell}_3(\boldsymbol{x}_{1i}) = \beta^* \boldsymbol{\ell}_{31i} + \boldsymbol{\ell}_{32i}$, where β^* is the common roots of the polynomials $q_k(\beta) = \sum_{j=0}^{n} \mathcal{T}(\nu_n(\boldsymbol{x}_{1i}), e_k, f_j(\boldsymbol{\ell}_{31i}, \boldsymbol{\ell}_{32i}))\beta^j$.

4: **Epipoles:** Given a set of epipolar lines $\{(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}), \boldsymbol{\ell}_3(\boldsymbol{x}_{1i}))\}_{i=1}^{N}$,

   1. Compute the multibody epipoles $\boldsymbol{c}_2 \in \mathbb{R}^{M_n(3)}$ and $\boldsymbol{c}_3 \in \mathbb{R}^{M_n(3)}$ from (8.55), and let $p_2(\boldsymbol{\ell}_2) = \boldsymbol{c}_2^\top \nu_n(\boldsymbol{\ell}_2)$ and $p_3(\boldsymbol{\ell}_3) = \boldsymbol{c}_3^\top \nu_n(\boldsymbol{\ell}_3)$.
   2. Compute the epipole pairs from the gradients of p_2 and p_3 as follows:
      for all $j = n : 1$ **do**

$$i_j = \underset{i=1,\ldots,N}{\arg\min} \frac{\frac{|p_2(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))|}{\|\nabla p_2(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))\|}}{\prod_{k=j+1}^{n} |\boldsymbol{e}_{2k}^\top \boldsymbol{\ell}_2(\boldsymbol{x}_{1i})|} + \frac{\frac{|p_3(\boldsymbol{\ell}_3(\boldsymbol{x}_{1i}))|}{\|\nabla p_3(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))\|}}{\prod_{k=j+1}^{n} |\boldsymbol{e}_{3k}^\top \boldsymbol{\ell}_2(\boldsymbol{x}_{1i})|};$$

$$\boldsymbol{e}_{2j} \sim \nabla p_2(\boldsymbol{\ell}_2(\boldsymbol{x}_{1i_j})) \qquad \text{and} \qquad \boldsymbol{e}_{3j} \sim \nabla p_3(\boldsymbol{\ell}_3(\boldsymbol{x}_{1i_j})).$$

      end for

5: **Feature segmentation:** Assign point correspondence $(\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i}, \boldsymbol{x}_{3i})$ to motion $j = \arg\min_{k=1,\ldots,n} (\boldsymbol{e}_{2k}^\top \boldsymbol{\ell}_2(\boldsymbol{x}_{1i}))^2 + (\boldsymbol{e}_{3k}^\top \boldsymbol{\ell}_3(\boldsymbol{x}_{1i}))^2$.

6: **Trifocal tensors:** Obtain the individual trifocal tensors $\{T_j\}_{j=1}^{n}$ from the trilinear constraint for each group.

8.6 Bibliographical Notes

3-D motion estimation and segmentation has been an active topic of research in the computer vision community over the past few years. Earlier work [**?**] solves this problem by first clustering the features corresponding to the same motion using e.g., K-means or spectral clustering, and then estimating a single motion model for each group. This can also be done in a probabilistic framework [**?**] in which a maximum-likelihood estimate of the parameters of each motion model is sought by alternating between feature clustering and single-body motion estimation using the Expectation Maximization (EM) algorithm. However, the convergence of EM to the global maximum depends strongly on initialization [Torr et al., 2001].

In order to deal with the initialization problem of EM-like approaches, recent work has concentrated on the study of the geometry of dynamic scenes, including the analysis of multiple points moving linearly with constant speed [**?**, **?**] or in a conic section [**?**], multiple points moving in a plane [**?**], multiple translating planes [**?**], self-calibration from multiple motions [**?**, **?**], multiple moving objects seen by an affine camera [Boult and Brown, 1991, ?, Kanatani, 2001, Wu et al., 2001, ?, ?, ?, Vidal and Hartley, 2004], and two-object segmentation from two perspective views [**?**]. The case of multiple moving objects seen by two perspective views was recently studied in [**?**, **?**, Vidal and Ma, 2004, ?], and has been extended to three perspective views via the so-called *multibody trifocal tensor* [**?**]. Such works have been the basis for the material presented in this chapter. Recent extensions omnidirectional cameras can be found in [**?**, **?**].

8.7 Exercises

Exercise 8.1 Motion Segmentation from Optical Flow in Multiple Perspective Views. Let $\Omega_f = (\omega_{1f}, \omega_{2f}, \omega_{3f})^\top$ and $V_f = (v_{1f}, v_{2f}, v_{3f})^\top$ be, respectively, the rotational and translational velocities of one a moving object relative to the camera at frame $f = 1, \ldots, F$. Under the perspective projection model, the projection of point $\boldsymbol{X}_p = (X_p, Y_p, Z_p, 1)^\top \in \mathbb{P}^3$ on the zeroth frame is $(x_p, y_p)^\top = (X_p, Y_p)^\top / Z_p$. Show that the optical flow $\boldsymbol{u}_{fp} \in \mathbb{R}^2$ of point p in the fth frame relative to the zeroth is:

$$\boldsymbol{u}_{fp} = \begin{bmatrix} x_p y_p & -(1+x_p^2) & -y_p & 1/Z_p & 0 & x_p/Z_p \\ (1+y_p^2) & -x_p y_p & x_p & 0 & 1/Z_p & y_p/Z_p \end{bmatrix} \begin{bmatrix} \Omega_f \\ V_f \end{bmatrix}.$$

Given measurements for the optical flow $\{\boldsymbol{u}_{fp}\}$ of P pixels in F frames, define the matrix of image measurements as

$$W = \begin{bmatrix} \boldsymbol{u}_{11} & \cdots & \boldsymbol{u}_{1P} \\ \vdots & & \vdots \\ \boldsymbol{u}_{F1} & \cdots & \boldsymbol{u}_{FP} \end{bmatrix}_{2F \times P} \tag{8.60}$$

Show that W can be factored into its motion and structure components as $W = MS^\top$, where

$$M = \begin{bmatrix} \omega_{11} & \omega_{21} & -\omega_{31} & 0 & 0 & v_{11} & v_{31} & 0 \\ -\omega_{21} & 0 & 0 & \omega_{11} & \omega_{31} & v_{21} & 0 & v_{31} \\ \vdots & & & & & & & \vdots \\ \omega_{1F} & \omega_{2F} & -\omega_{3F} & 0 & 0 & v_{1F} & v_{3F} & 0 \\ -\omega_{2F} & 0 & 0 & \omega_{1F} & \omega_{3F} & v_{2F} & 0 & v_{3F} \end{bmatrix}_{2F\times 8} \tag{8.61}$$

$$S = \begin{bmatrix} x_1y_1 & z_1 - x_1^2 & y_1 & y_1^2 - z_1 & x_1 & \frac{1}{Z_1} & \frac{x_1}{Z_1} & \frac{y_1}{Z_1} \\ \vdots & & & & & & & \vdots \\ x_Py_P & z_P - x_P^2 & y_P & y_P^2 - z_P & x_P & \frac{1}{Z_P} & \frac{x_P}{Z_P} & \frac{y_P}{Z_P} \end{bmatrix}_{P\times 8}.$$

Exercise 8.2 Show that for all $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2 \in \mathbb{R}^3$ and $\alpha \in \mathbb{R}$

$$\nu_n(\alpha\boldsymbol{\ell}_1 + \boldsymbol{\ell}_2) = \sum_{i=0}^{n} \alpha^i f_i(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2), \tag{8.62}$$

where $f_i(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2) \in \mathbb{R}^{M_n(3)}$ is a bi-homogeneous polynomial of degree i in $\boldsymbol{\ell}_1$ and $(n-i)$ in $\boldsymbol{\ell}_2$ for $i = 0, \ldots, n$.

Exercise 8.3 Show that

$$\begin{aligned} &\mathcal{T}(\nu_n(\boldsymbol{x}_1), \nu_n(\alpha\boldsymbol{\ell}_{21} + \boldsymbol{\ell}_{22}), \nu_n(\beta\boldsymbol{\ell}_{31} + \boldsymbol{\ell}_{32})) \\ &\qquad = \mathcal{T}(\nu_n(\boldsymbol{x}_1), \sum_{i=0}^{n} \alpha^i f_i(\boldsymbol{\ell}_{21}, \boldsymbol{\ell}_{22}), \sum_{j=0}^{n} \beta^j f_j(\boldsymbol{\ell}_{31}, \boldsymbol{\ell}_{32})) \\ &\qquad = \sum_{i=0}^{n}\sum_{j=0}^{n} \alpha^i\beta^j \mathcal{T}(\nu_n(\boldsymbol{x}_1), f_i(\boldsymbol{\ell}_{21}, \boldsymbol{\ell}_{22}), f_j(\boldsymbol{\ell}_{31}, \boldsymbol{\ell}_{32})) \end{aligned} \tag{8.63}$$

Exercise 8.4 In this exercise and next few exercises, we investigate yet another algebraic technique that allows us to segment multiple homographies. We may interpret the second image $\boldsymbol{x}_2 \in \mathbb{P}^2$ as a point in $\mathbb{CP}$ by considering the first two coordinates in $\boldsymbol{x}_2$ as a complex number and appending a one to it. However, we still think of $\boldsymbol{x}_1$ as a point in $\mathbb{P}^2$. With this interpretation, we can rewrite (8.29) as

$$\boldsymbol{x}_2 \sim H\boldsymbol{x}_1 \doteq \begin{bmatrix} h_{11} + h_{21}\sqrt{-1} & h_{12} + h_{22}\sqrt{-1} & h_{13} + h_{23}\sqrt{-1} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \boldsymbol{x}_1, \tag{8.64}$$

where $H \in \mathbb{C}^{2\times 3}$ now represents a *complex homography*[5]. Let $\boldsymbol{w}_2$ be the vector in $\mathbb{CP}$ perpendicular to $\boldsymbol{x}_2$, i.e., if $\boldsymbol{x}_2 = [z, 1]^T$ then $\boldsymbol{w}_2 = [1, -z]^T$. Then we can rewrite (8.64) as the following *complex* bilinear constraint

$$\boldsymbol{w}_2^\top H \boldsymbol{x}_1 = 0, \tag{8.65}$$

which we call the *complex homography constraint*. We can therefore interpret the motion segmentation problem as one in which we are given image data $\{\boldsymbol{x}_1^j \in \mathbb{P}^2\}_{j=1}^N$ and $\{\boldsymbol{w}_2^j \in$

[5]Strictly speaking, we embed each real homography matrix into an affine complex matrix.

$\mathbb{CP}\}_{j=1}^{N}$ generated by a collection of n complex homographies $\{H_i \in \mathbb{C}^{2\times 3}\}_{i=1}^{n}$. Then each image pair $(\boldsymbol{x}_1, \boldsymbol{w}_2)$ has to satisfy the *multibody homography constraint*

$$p(\boldsymbol{x}_1, \boldsymbol{w}_2) = \prod_{i=1}^{n} (\boldsymbol{w}_2^\top H_i \boldsymbol{x}_1) = \nu_n(\boldsymbol{w}_2)^\top \mathcal{H} \nu_n(\boldsymbol{x}_1) = 0, \tag{8.66}$$

regardless of which one of the n complex homographies is associated with the image pair. We call the matrix $\mathcal{H} \in \mathbb{C}^{M_n(2)\times M_n(3)}$ the *multibody homography*. Now, since the multibody homography constraint (8.66) is linear in the multibody homography $\mathcal{H}$, we can linearly solve for $\mathcal{H}$ from (8.66) given $N \geq M_n(2)M_n(3) - (M_n(3)+1)/2 \sim O(n^3)$ image pairs in general position[6] with at least 4 pairs per moving object.

1. Show that given a pair of images $(\boldsymbol{x}_1, \boldsymbol{w}_2)$ associated with the ith homography H_i, the partial derivative $\frac{\partial p(\boldsymbol{x}_1, \boldsymbol{w}_2)}{\partial \boldsymbol{x}_1}$ is perpendicular to the (right) null space of H_i.
2. Use the above fact and show how to use GPCA to obtain the null spaces of all H_i simultaneously from the derivatives.
3. Use the above fact to contrive a scheme to segment (image pairs associated with) the n homographies.

Exercise 8.5 There is yet another way to retrieve individual H_i from $\mathcal{H}$ without segmenting the image pairs first. The right null space of the complex homography matrix H_i defined in the previous exercise is called the complex epipole of H_i, denoted as $\boldsymbol{e}_i$.

1. Show that, once $\boldsymbol{e}_i$ is known, the partial derivative $\left.\frac{\partial p(\boldsymbol{x}_1, \boldsymbol{w}_2)}{\partial \boldsymbol{x}_1}\right|_{(\boldsymbol{e}_i, \boldsymbol{w}_2)}$ is a linear combination of the rows of H_i (up to scale).
2. Show that, by properly evaluating the derivative at different values of $\boldsymbol{w}_2$, one can retrieve H_i.

Exercise 8.6 In the above exercises, we have implicitly assumed that the complex epipoles are different. Show that, under mild conditions, e.g., the third rows of each H_i are different, the null spaces of the corresponding complex homographies are indeed different for different real homographies.[7]

Exercise 8.7 A homography is typically of the form $H = R + T\pi^\top$ where π is the plane normal and (R, T) are the rotation and translation of the camera. If the homographies come from different planes (different π) undergoing the same rigid-body motion, show that the epipoles of the corresponding complex homographies are different too.

Exercise 8.8 (Trifocal tensors from second order derivatives of the multibody trilinear constraint) Let $\boldsymbol{x}$ be an arbitrary point in $\mathbb{P}^2$ (not necessarily a point in the first view) and let $(\boldsymbol{e}_i', \boldsymbol{e}_i''$ be the ith epipole pair.

[6]The multibody homography constraint gives two equations per image pair, and there are $(M_n(2)-1)M_n(3)$ complex entries in $\mathcal{H}$ and $M_n(3)$ real entries (the last row).

[7]In fact, one can further show that the set of complex homographies that share the same null space is a five-dimensional subset (hence a zero-measure subset) of all real homography matrices. Furthermore, one can complexify any other two rows of H instead of the first two. As long as two homography matrices are different, one of the complexifications will give different complex epipoles.

1. Given $\boldsymbol{e}_i'$, propose a method to compute the epipolar line of $\boldsymbol{x}$ in the second view $\boldsymbol{\ell}_{i\boldsymbol{x}}'$ according to the i^{th} motion. Show also how to compute the epipolar line of $\boldsymbol{x}$ in the third view $\boldsymbol{\ell}_{i\boldsymbol{x}}''$, given $\boldsymbol{e}_i''$.

2. Show that the slices of the trifocal tensor T_i can be expressed in terms of the second derivative of the multibody epipolar constraint, as follows:

$$\left.\frac{\partial^2(\widetilde{\boldsymbol{x}}\widetilde{\boldsymbol{\ell}'}\widetilde{\boldsymbol{\ell}''}\mathcal{T})}{\partial\boldsymbol{\ell}'\partial\boldsymbol{\ell}''}\right|_{(\boldsymbol{x},\boldsymbol{\ell}_{i\boldsymbol{x}}',\boldsymbol{\ell}_{i\boldsymbol{x}}'')} = M_{i\boldsymbol{x}} \sim \boldsymbol{x}T_i \in \mathbb{R}^{3\times 3}. \tag{8.67}$$

Part III

Appendices

Appendix A
Basic Facts from Mathematical Statistics

"A knowledge of statistics is like a knowledge of foreign languages or of algebra; it may prove of use at any time under any circumstances."
– A. L. Bowley

In the practice of science and engineering, data are often modeled as samples of a random variable (or vector) drawn from a certain probability distribution. Mathematical statistics then deals with the problem how to infer the underlying distribution from the given samples. To render the problem tractable, we typically assume that the unknown distribution belongs to certain parametric family (e.g., Gaussian), and the problem becomes how to estimate the parameters of the distribution from the samples.

In this appendix, we provide a brief review of some of the relevant concepts and results from mathematical statistics used in this book. The review is not meant to be exhaustive, but rather to make the book self-contained for readers who already have basic knowledge in probability theory and statistics. If one is looking for a more formal and thorough introduction to mathematical statistics, we recommend the classic books of [Wilks, 1962] or [Bickel and Doksum, 2000].

A.1 Estimation of Parametric Models

Let $\boldsymbol{x}$ be a random variable or vector. For simplicity, we assume the distribution of $\boldsymbol{x}$ has a density $p(\boldsymbol{x}, \theta)$, where the parameter (vector) $\theta = [\theta_1, \theta_2, \ldots, \theta_d]^\top \in \mathbb{R}^d$, once known, uniquely determines the density function $p(\cdot, \theta)$. Now suppose

$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ are a set of samples of $\boldsymbol{x}$ independently drawn according to the density $p(\boldsymbol{x}, \theta)$. That is, $\boldsymbol{X}$ has the density

$$p(\boldsymbol{X}, \theta) = \prod_{i=1}^{N} p(\boldsymbol{x}_i, \theta). \tag{A.1}$$

We call any real or vector-valued function of the samples $\boldsymbol{X}$ a *statistic* and denote it by $T(\boldsymbol{X})$. The goal here is to properly choose the function $T(\cdot)$ so that it gives a "good" estimate for the true parameter θ.

Definition A.1 (Sufficient Statistic)**.** *A statistic $T(\boldsymbol{X})$ is said to be* sufficient *for θ if, and only if, the conditional distribution of $\boldsymbol{X}$ given $T(\boldsymbol{X})$ does not depend on θ.*

That is, $p(\boldsymbol{X}, \theta | T(\boldsymbol{X}))$ no longer depends on θ. Thus, the original samples $\boldsymbol{X}$ do not contain any more information about θ than $T(\boldsymbol{X})$.

Theorem A.2 (Factorization Theorem)**.** *A statistic $T(\boldsymbol{X})$ is sufficient for θ if, and only if, there exists a function $g(t, \theta)$ and a function $h(\boldsymbol{X})$ such that*

$$p(\boldsymbol{X}, \theta) = g(T(\boldsymbol{X}), \theta) h(\boldsymbol{X}). \tag{A.2}$$

A popular measure of "goodness" of a statistic $T(\boldsymbol{X}) \in \mathbb{R}^d$ as an estimate of $\theta \in \mathbb{R}^d$ is the mean squared error between $T(\boldsymbol{X})$ and θ:

$$R(\theta, T) = \mathbb{E}[\|T(\boldsymbol{X}) - \theta\|^2]. \tag{A.3}$$

The choice of this measure is not just for convenience: When the sample size N is large, the distribution of many estimates converges to a normal distribution with θ as the mean. Then R is the variance of the estimates. In some literature, such a function is also referred to as the "risk function," hence the capital letter "R."

We may rewrite the expression $R(\theta, T)$ as follows:

$$\begin{aligned} R(\theta, T) &= \mathbb{E}[\|T(\boldsymbol{X}) - \mathbb{E}[T(\boldsymbol{X})] + \mathbb{E}[T(\boldsymbol{X})] - \theta\|^2] \\ &= \mathbb{E}[\|T(\boldsymbol{X}) - \mathbb{E}[T(\boldsymbol{X})]\|^2] + \|\mathbb{E}[T(\boldsymbol{X})] - \theta\|^2 \\ &\doteq \text{Var}(T(\boldsymbol{X})) + \boldsymbol{b}^2(\theta, T), \end{aligned} \tag{A.4}$$

where $\boldsymbol{b}(\theta, T) = E[T(\boldsymbol{X})] - \theta$ is called the *bias* of the estimate $T(\boldsymbol{X})$, and $\text{Var}(T(\boldsymbol{X})) \in \mathbb{R}$ is the trace of the covariance matrix

$$\text{Cov}(T(\boldsymbol{X})) \doteq \mathbb{E}[T(\boldsymbol{X}) T(\boldsymbol{X})^\top] \in \mathbb{R}^{d \times d}. \tag{A.5}$$

We refer to $\text{Var}(T(\boldsymbol{X}))$ as the "variance" of $T(\boldsymbol{X})$. Thus, a good estimate is one that has both small bias and small variance.

Unfortunately, there is no such thing as a universally optimal estimate that gives a smaller error R than any other estimates for all θ. For instance, if the true parameter is θ_0, for the estimate $S(\boldsymbol{X}) = \theta_0$, it achieves the smallest possible error $R(\theta, S) = 0$. Thus, the universally optimal estimate, say T, would have to have $R(\theta_0, T) = 0$ too. As θ_0 can be arbitrary, then T has to estimate every potential parameter θ perfectly, which is impossible except for trivial cases. One can view this as a manifestation of the so-called *No Free Lunch Theorem* known in learning theory: Without any prior knowledge in θ, we can only expect a statistical estimate to be better than others most of the time, but we can never expect it to be the best *all the time*. Thus, in the future, whenever we claim some estimate is "optimal," it will be in the restricted sense that it is optimal within a special class of estimates considered (e.g., unbiased estimates).

Define the *Fisher information matrix* to be

$$I(\theta) \doteq \mathbb{E}\Big[\big(\frac{\partial}{\partial\theta}\log p(\boldsymbol{X},\theta)\big)\big(\frac{\partial}{\partial\theta}\log p(\boldsymbol{X},\theta)\big)^\top\Big] \in \mathbb{R}^{d\times d}. \tag{A.6}$$

Let $\psi(\theta) \doteq \mathbb{E}[T(\boldsymbol{X})] = [\psi_1(\theta), \psi_2(\theta, \dots, \psi_d(\theta)]^\top$ and define:

$$\frac{\partial\psi(\theta)}{\partial\theta} \doteq \begin{bmatrix} \frac{\partial\psi_1(\theta)}{\partial\theta_1} & \frac{\partial\psi_1(\theta)}{\partial\theta_2} & \cdots & \frac{\partial\psi_1(\theta)}{\partial\theta_d} \\ \frac{\partial\psi_2(\theta)}{\partial\theta_1} & \frac{\partial\psi_2(\theta)}{\partial\theta_2} & \cdots & \frac{\partial\psi_2(\theta)}{\partial\theta_d} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial\psi_d(\theta)}{\partial\theta_1} & \frac{\partial\psi_d(\theta)}{\partial\theta_2} & \cdots & \frac{\partial\psi_d(\theta)}{\partial\theta_d} \end{bmatrix} \in \mathbb{R}^{d\times d}. \tag{A.7}$$

Theorem A.3 (Information Inequality)**.** *Under reasonable conditions, we have that for all θ, $\psi(\theta)$ is differentiable and*

$$Cov(T(\boldsymbol{X})) \geq \frac{\partial\psi(\theta)}{\partial\theta} I(\theta)^{-1}\Big(\frac{\partial\psi(\theta)}{\partial\theta}\Big)^T, \tag{A.8}$$

where the inequality is between semi-positive definite symmetric matrices.

For unbiased estimate $\psi(\theta) = \theta$, we have $\psi'(\theta) = I$. The information inequality can be thought of as giving a lower bound for the variance of any unbiased estimate: $\mathrm{Cov}(T(\boldsymbol{X})) \geq I(\theta)^{-1}$, which is often referred to as the *Cramér-Rao lower bound.*

As $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ are i.i.d. samples from the distribution $p(\boldsymbol{x}, \theta)$, we define $I_1(\theta) \doteq E\big[\frac{\partial}{\partial\theta}\log p(\boldsymbol{x}_1,\theta)(\frac{\partial}{\partial\theta}\log p(\boldsymbol{x}_1,\theta))^T\big] \in \mathbb{R}^{d\times d}$. Then, we have

$$I(\theta) = N I_1(\theta). \tag{A.9}$$

The Cramér-Rao lower bound can be rewritten as $\mathrm{Cov}(T(\boldsymbol{X})) \geq \frac{1}{N} I_1(\theta)^{-1}$.

A.1.1 Uniformly Minimum Variance Unbiased Estimates

As we have mentioned earlier, to make the model estimation problem well-conditioned, one must restrict the class of estimates. For instance, we may require the estimate $T(\boldsymbol{X})$ needs to be unbiased, i.e., $b(\theta, T) = 0$. Then the problem of finding the best unbiased estimate becomes

$$\min_{T(\cdot)} R(\theta, T) = \mathrm{Var}(T(\boldsymbol{X})) \quad \text{s.t.} \quad E[T(\boldsymbol{X})] = \theta. \tag{A.10}$$

The optimal T^* is then called the *uniformly minimum variance unbiased* (UMVU) estimate. Such a T^* often exists and in the absence of knowledge about θ, it seems to be the best estimate one can hope to obtain.

Definition A.4 (Complete Statistic)**.** *A statistic T is said to be* complete *if the only real function $g(\cdot)$ which satisfies $E[g(T)] = 0$ for all θ is the function $g(T) \equiv 0$.*

Starting with a sufficient and complete statistic $T(\boldsymbol{X})$, the following theorem simplifies the computation of the UMVU estimate:

Theorem A.5 (Lehmann-Scheffé)**.** *If $T(\boldsymbol{X})$ is a complete sufficient statistic and $S(\boldsymbol{X})$ is any unbiased estimate of θ, then $T^*(\boldsymbol{X}) = \mathbb{E}[S(\boldsymbol{X})|T(\boldsymbol{X})]$ is an UMVU estimate of θ. If further Var$(T^*(\boldsymbol{X})) < \infty$ for all θ, $T^*(\boldsymbol{X})$ is the unique UMVU estimate.*

Even so, the UMVU estimate is often too difficult to compute in practice. Furthermore, the property of unbiasedness is not invariant under functional transformation: if $T(\boldsymbol{X})$ is an unbiased estimate for θ, $g(T(\boldsymbol{X}))$ is in general not an unbiased estimate for $g(\theta)$. To have the functional invariant property, we often resort to the so-called Maximum Likelihood estimate.

A.1.2 Maximum Likelihood Estimates

If the N samples $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ are independently drawn from the same distribution $p(\boldsymbol{x}, \theta)$, their joint distribution has the density $p(\boldsymbol{X}, \theta) = \prod_{i=1}^N p(\boldsymbol{x}_i, \theta)$.

Consider $p(\boldsymbol{X}, \theta)$ as a function of θ with $\boldsymbol{X}$ fixed. We call this function the *likelihood function*, denoted as $L(\theta, \boldsymbol{X}) = p(\boldsymbol{X}, \theta)$. The *maximum likelihood (ML) estimate* of θ is given by the solution to the following optimization problem:

$$\hat{\theta}_N = \arg\max_{\theta} \left(L(\theta, \boldsymbol{X}) = p(\boldsymbol{X}, \theta) = \prod_{i=1}^{N} p(\boldsymbol{x}_i, \theta) \right). \tag{A.11}$$

As $\hat{\theta}_N$ maximizes the likelihood function $L(\theta, \boldsymbol{X})$, a necessary condition for optimality is that

$$\frac{\partial L(\theta, \boldsymbol{X})}{\partial \theta}\bigg|_{\hat{\theta}_N} = 0. \tag{A.12}$$

It is easy to see that the ML estimate is invariant under functional transformations. That is, if $\hat{\theta}_N$ is an ML estimate of θ, then $g(\hat{\theta}_N)$ is an ML estimate of $g(\theta)$.

Since the logarithmic function is monotonic, we may choose to maximize the log likelihood function instead:

$$\hat{\theta}_N = \arg\max_{\theta} \left(\log(L(\theta, \boldsymbol{X})) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i, \theta) \right), \tag{A.13}$$

which often turns out to be more convenient to use in practice. The ML estimate is a more popular choice than the UMVU estimate because its existence is easier to establish and is usually easier to compute than the UMVU estimate. Furthermore, when the sample size is large, the ML estimate is asymptotically optimal for a wide variety of parametric models. Thus, both UMVU and ML estimates give essentially the same answer in a way that we explain in more detail.

A.1.3 Estimates from a Large Number of Samples

Definition A.6 (Consistency)**.** *An estimate $\hat{\theta}_N$ of θ is said to be* consistent *if, and only if,*

$$P\big[\|\hat{\theta}_N - \theta\| \geq \varepsilon\big] \;\to\; 0 \tag{A.14}$$

for all $\varepsilon > 0$ as $N \to \infty$.

In other words, $\hat{\theta}_N$ is consistent if it converges in probability to θ.

Definition A.7 (Asymptotic Unbiasedness)**.** *Let $\mu_N = \mathbb{E}[\hat{\theta}_N] \in \mathbb{R}^d$ and $\Sigma_N = Cov(\hat{\theta}_N) \in \mathbb{R}^{d \times d}$. We say that $\hat{\theta}$ is* asymptotically unbiased *if as $N \to \infty$*

$$\sqrt{N}(\mu_N - \theta) \to 0, \quad \textit{and} \quad N\Sigma_N \to \Sigma > 0 \tag{A.15}$$

for some positive-definite symmetric matrix $\Sigma \in \mathbb{R}^{d \times d}$.

It is easy to see that asymptotic unbiasedness is a stronger property than consistency. That is, an estimate can be consistent but asymptotically biased. In addition, most "reasonable" estimates $\hat{\theta}_N$ (e.g., the ML estimate) are often asymptotically normally distributed with mean μ_N and covariance matrix Σ_N due to the law of large numbers. Therefore, the asymptotical distribution of an asymptotically unbiased estimate is uniquely characterized by the parameters θ and Σ.

Between any two asymptotically unbiased estimates, say $\hat{\theta}_N^{(1)}$ and $\hat{\theta}_N^{(2)}$, their relative *asymptotic efficiency* of $\hat{\theta}_N^{(1)}$ to $\hat{\theta}_N^{(2)}$ is defined to be the ratio

$$e\big(\hat{\theta}_N^{(1)}, \hat{\theta}_N^{(2)}\big) \doteq \frac{\det(\Sigma^{(2)})}{\det(\Sigma^{(1)})}, \tag{A.16}$$

where $\Sigma^{(i)} = \lim_{N\to\infty} N\mathrm{Cov}\big(\hat{\theta}_N^{(i)}\big)$, for $i = 1, 2$. The larger the efficiency ratio e, the smaller the asymptotic variance of $\hat{\theta}^{(1)}$, relative to that of $\hat{\theta}^{(2)}$. Thus, $\hat{\theta}^{(1)}$ gives a more accurate or "sharper" estimate for θ, although both $\hat{\theta}^{(1)}$ and $\hat{\theta}^{(2)}$ are asymptotically unbiased.

Nevertheless, according to Theorem A.3, an estimate cannot be arbitrarily more efficient than others. That is, for any asymptotically unbiased estimate $\hat{\theta}_N$, using (A.9) and (A.15), its covariance matrix is bounded asymptotically from below by the Cramér-Rao bound:

$$\lim_{N\to\infty} N\Sigma_N = \Sigma \geq I_1(\theta)^{-1}. \tag{A.17}$$

Definition A.8 (Asymptotic Efficiency)**.** *An estimate $\hat{\theta}_N$ is said to be asymptotically efficient if it is asymptotically normal and it achieves equality in the Cramér-Rao bound* (A.17)*.*

Obviously, an asymptotically efficient estimate has efficiency $e \geq 1$ with respect to any other asymptotically unbiased estimates that satisfy (A.17).

Asymptotic efficiency is a desirable property for an estimate and it is sometimes referred to as asymptotic optimality. It often can be shown that UMVU estimates are asymptotically efficient. We also have that:

Proposition A.9. *In general, the maximum likelihood estimate is asymptotically efficient.*

Proof. We here outline the basic ideas for a "proof," which can also be used to establish for other estimates their asymptotic unbiasedness or efficiency with respect to the ML estimate. Define the function

$$\psi(\boldsymbol{x}, \theta) \doteq \frac{\partial}{\partial \theta} \log p(\boldsymbol{x}, \theta) \in \mathbb{R}^d. \tag{A.18}$$

Assume that the maximum likelihood estimate $\hat{\theta}_N$ exists. It satisfies the equation

$$\frac{\partial L(\theta, \boldsymbol{X})}{\partial \theta}\Big|_{\hat{\theta}_N} = \sum_{i=1}^{N} \psi(\boldsymbol{x}_i, \hat{\theta}_N) = 0. \tag{A.19}$$

By the mean value theorem,

$$\sum_{i=1}^{N} \psi(\boldsymbol{x}_i, \hat{\theta}_N) - \sum_{i=1}^{N} \psi(\boldsymbol{x}_i, \theta) = \Big[\sum_{i=1}^{N} \frac{\partial \psi(\boldsymbol{x}_i, \theta_N^*)}{\partial \theta}\Big]\Big(\hat{\theta}_N - \theta\Big), \tag{A.20}$$

where θ_N^* is a point between θ and $\hat{\theta}_N$. Using (A.19),

$$\sqrt{N}\big(\hat{\theta}_N - \theta\big) = \Big[\frac{1}{N}\sum_{i=1}^{N} \frac{\partial \psi(\boldsymbol{x}_i, \theta_N^*)}{\partial \theta}\Big]^{-1}\Big(-N^{-\frac{1}{2}}\sum_{i=1}^{N} \psi(\boldsymbol{x}_i, \theta)\Big). \tag{A.21}$$

Under suitable conditions, $\hat{\theta}_N$ is consistent, and by the law of large numbers, $\frac{1}{N}\sum_{i=1}^{N} \frac{\partial \psi(\boldsymbol{x}_i, \theta_N^*)}{\partial \theta}$ behaves like $\frac{1}{N}\sum_{i=1}^{N} \frac{\partial \psi(\boldsymbol{x}_i, \theta)}{\partial \theta}$ which converges to

$$\begin{aligned}
\mathbb{E}\Big[\frac{\partial \psi(\boldsymbol{x}_1, \theta)}{\partial \theta}\Big] &= \mathbb{E}\Big[\frac{\partial^2}{\partial \theta^2} \log p(\boldsymbol{x}_1, \theta)\Big] \\
&= -E\Big[\frac{\partial}{\partial \theta} \log p(\boldsymbol{x}_1, \theta)\big(\frac{\partial}{\partial \theta} \log p(\boldsymbol{x}_1, \theta)\big)^T\Big] = -I_1(\theta).
\end{aligned}$$

It is easy to show that $\psi(\boldsymbol{x}, \theta)$ is zero-mean and thus, by the central limit theorem, the right-hand side of (A.21) converges to a normal distribution with zero mean and variance $I_1(\theta)^{-1}$. That is, the asymptotic variance of the ML estimate reaches the Carmér-Rao lower bound. □

When the sample size is large, one can appeal to the law of large numbers to derive an information-theoretic justification for the ML estimate, which can be somewhat more revealing. Notice that maximizing the log likelihood function is equivalent to minimizing the following objective function:

$$\min_{\theta} H(\theta, N) \doteq \frac{1}{N}\sum_{i=1}^{N} -\log p(\boldsymbol{x}_i, \theta). \tag{A.22}$$

In information theory, the quantity $-\log p(\boldsymbol{x}, \theta)$ is associated with the number of bits required to represent a random event $\boldsymbol{x}$ that has the probability $p(\boldsymbol{x}, \theta)$ [Cover and Thomas, 1991]. When the sample size N is large, due to the law of large numbers, the quantity $H(\theta, N)$ converges to

$$\lim_{N\to\infty} H(\theta, N) = H(\theta) = \mathbb{E}[-\log p(\boldsymbol{x}, \theta)] = \int \big(-\log p(\boldsymbol{x}, \theta)\big) p(\boldsymbol{x}, \theta_0)\, d\boldsymbol{x}, \tag{A.23}$$

where $p(\boldsymbol{x}, \theta_0)$ is the true distribution. Notice that the above quantity is a measure similar to the notion of "entropy": $H(\theta)$ is asymptotically the average code length of the sample set $\{\boldsymbol{x}_i\}$ when we assume that it is of the distribution $p(\boldsymbol{x}, \theta)$ while $\boldsymbol{x}$ is actually drawn according to $p(\boldsymbol{x}, \theta_0)$. Thus, the goal of ML estimate is to find the $\hat{\theta}$ that minimizes the empirical entropy of the given sample set. This is obviously a smart thing to do as such estimate $\hat{\theta}$ gives the most compact representation of the given sample data if an optimal coding scheme is adopted [Cover and Thomas, 1991]. We refer to this as the "minimum entropy principle."

Notice also that the $\hat{\theta}$ that minimizes $\int \big(-\log p(\boldsymbol{x}, \theta)\big) p(\boldsymbol{x}, \theta_0)\, d\boldsymbol{x}$ is the same as that minimizing the so-called *Kullback-Leibler (KL) divergence* between the two distributions $p(\boldsymbol{x}, \theta_0)$ and $p(\boldsymbol{x}, \theta)$, i.e.,

$$D\big(p(\boldsymbol{x}, \theta_0) \| p(\boldsymbol{x}, \theta)\big) \doteq \int \Big(\log \frac{p(\boldsymbol{x}, \theta_0)}{p(\boldsymbol{x}, \theta)}\Big) p(\boldsymbol{x}, \theta_0)\, d\boldsymbol{x}, \tag{A.24}$$

One may show that under general conditions, the KL divergence is always nonnegative and becomes zero if and only if $\theta = \theta_0$. In essence, when the sample size is large, the ML objective is equivalent to minimizing the KL divergence.

However, the ML estimate is known to have very bad performance in some models even with a large number of samples. This is particularly the case when the models have many redundant parameters or the distributions are degenerate. Furthermore, both UMVU and ML estimates are not the optimal estimates in a Bayesian[1] or minimax[2] sense. For instance, the ML estimate can be viewed as a special Bayesian estimate only when the parameter θ is uniformly distributed.

In this book, the concepts introduced in this section can help us understand under what assumptions on the distribution of the data, the estimates given by the GPCA algorithms can be asymptotically unbiased (hence consistent), or asymptotically efficient.

A.2 Expectation Maximization

In many practical situations, one is required to estimate a statistical model with only part of the random states being observable and the rest being "missing," or

[1]A bayesian estimate T^* is the solution to the following problem $\min_T \int R(\theta, T)\pi(\theta)\, d\theta$ for a given prior distribution $\pi(\theta)$ of θ. That is, T^* is the best estimate in terms of its average risk.

[2]A minimax estimate T^* is the solution to the problem $\min_T \max_\theta R(\theta, T)$. That is, T^* is the best estimate according to its worst performance. Of course, such a T^* does not have to always exist or be easier to compute than the ML estimate.

"hidden," or "latent," or "unobserved." For instance, suppose that two random vectors $(\boldsymbol{x}, \boldsymbol{z})$ have a joint distribution (density) $p(\boldsymbol{x}, \boldsymbol{z}, \theta)$ but only samples of $\boldsymbol{x}$ are observable and $\boldsymbol{z}$ is not available. Our goal is, as before, to find an optimal estimate $\hat{\theta}$ for θ from the observations.

As samples of $\boldsymbol{z}$ are not available, there is no way one can find the maximum likelihood estimate of θ from the *complete log likelihood function*:

$$\max_{\theta} L_c(\theta, \boldsymbol{X}, \boldsymbol{Z}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i, \boldsymbol{z}_i, \theta). \tag{A.25}$$

Thus, it makes sense to use only the marginal distribution of $\boldsymbol{x}$: $p(\boldsymbol{x}, \theta) = \int p(\boldsymbol{x}, \boldsymbol{z}, \theta)\, d\boldsymbol{z}$ and find the maximum likelihood estimate from

$$\max_{\theta} L(\theta, \boldsymbol{X}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i, \theta), \tag{A.26}$$

which, in this context, is often referred to as the *incomplete log likelihood function* in the statistical literature. The problem is now reduced to a standard ML estimation problem and one can adopt any appropriate optimization method (say conjugate gradient) to find the maximum. It seems that there is no need of involving $\boldsymbol{z}$ at all.

An alternative approach to maximize $L(\theta, \boldsymbol{X})$ is to use the available data of $\boldsymbol{x}$ to estimate the values $\hat{\boldsymbol{z}}$ of the latent variables, and then search for the ML estimate $\hat{\theta}$ from the complete log likelihood $L_c(\theta, \boldsymbol{X}, \hat{\boldsymbol{Z}})$. There are several reasons why this often turns out to be a better idea. First, for some models $p(\boldsymbol{x}, \boldsymbol{z}, \theta)$, marginalizing $\boldsymbol{z}$ out can be difficult to do or that could destroy good structures in the models. The alternative approach may better harness these structures. Second, directly maximizing $L(\theta, \boldsymbol{X})$ may turn out to be a very difficult optimization problem (e.g., high-dimension, many local minima), the introduction of intermediate latent variables $\boldsymbol{z}$ actually makes the optimization easier (as we will see later). Third, in some applications, it is desired to obtain an estimate of the unobservables $\boldsymbol{z}$ from the observables $\boldsymbol{x}$. The alternative approach can simultaneously estimate both θ and $\boldsymbol{z}$. Be aware that regardless of the introduction of the latent variables $\boldsymbol{z}$ or not, as far as the parameter θ is concerned, the ultimate objective has always been to maximize the objective function $\max_{\theta} L(\theta, \boldsymbol{X})$.

Using the following identities

$$\forall \boldsymbol{z} \;\; p(\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, \boldsymbol{z}, \theta)}{p(\boldsymbol{z}|\boldsymbol{x}, \theta)} \quad \text{and} \quad \int p(\boldsymbol{z}|\boldsymbol{x}, \theta)\, d\boldsymbol{z} = 1, \tag{A.27}$$

we have

$$\begin{aligned} L(\theta, \boldsymbol{X}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i, \theta) &= \sum_{i=1}^{N} \int p(\boldsymbol{z}|\boldsymbol{x}_i, \theta) \log \frac{p(\boldsymbol{x}_i, \boldsymbol{z}, \theta)}{p(\boldsymbol{z}|\boldsymbol{x}_i, \theta)}\, d\boldsymbol{z} \\ = \sum_{i=1}^{N} \int \big[p(\boldsymbol{z}|\boldsymbol{x}_i, \theta) \log p(\boldsymbol{x}_i, \boldsymbol{z}, \theta) &- p(\boldsymbol{z}|\boldsymbol{x}_i, \theta) \log p(\boldsymbol{z}|\boldsymbol{x}_i, \theta) \big]\, d\boldsymbol{z}. \end{aligned} \tag{A.28}$$

Although the last expression seems more complicated than the original log likelihood $L(\theta, \boldsymbol{X})$, it reveals that the likelihood is a function of the *a posterior* probability $w_i(\boldsymbol{z}) \doteq p(\boldsymbol{z}|\boldsymbol{x}_i, \theta)$. The *a posterior* distribution of $\boldsymbol{z}$ gives us the best estimate of $\boldsymbol{z}$ given $\boldsymbol{x}_i$ and θ. In turn, we can update the parameter θ based on the estimate of $\boldsymbol{z}$. This leads to the well-known Expectation and Maximization (EM) algorithm for optimizing the log likelihood $L(\theta, \boldsymbol{X})$:

Step 1 (Expectation): For fixed θ^k and every $i = 1, 2, \ldots, N$,

$$w_i^{k+1}(\boldsymbol{z}) = \arg\max_{w_i} \big[w_i(\boldsymbol{z}) \log p(\boldsymbol{x}_i, \boldsymbol{z}, \theta^k) - w_i(\boldsymbol{z}) \log w_i(\boldsymbol{z})\big].$$

Step 2 (Maximization): For fixed w_i^{k+1},

$$\theta^{k+1} = \arg\max_{\theta} \sum_{i=1}^{N} \int w_i^{k+1}(\boldsymbol{z}) \log p(\boldsymbol{x}_i, \boldsymbol{z}, \theta)\, d\boldsymbol{z}.$$

The Maximization step does not involve the second term in (A.28) because it is constant with w_i fixed. The Expectation step is decomposed to every i because the *a posterior* $w_i(\boldsymbol{z})$ depends only on $\boldsymbol{x}_i$. It is important to know that each step of the EM algorithm is in general a much simpler optimization problem than directly maximizing the log likelihood $L(\theta, \boldsymbol{X})$ as the sum $\sum_{i=1}^{N} \log p(\boldsymbol{x}_i, \theta)$. For many popular models (e.g., mixtures of Gaussians), one might even be able to find closed-form formulae for both steps (see Chapter 3).

Notice that the EM algorithm is an *iterative* algorithm. Like gradient ascent, it is essentially a hill-climbing algorithm that each iteration increases the value of the log likelihood.

Proposition A.10. *The Expectation Maximization process converges to one of the stationary points (extrema) of the (log) likelihood function* $L(\theta, \boldsymbol{X})$.

Proof. We here give a sketch of the basic ideas of the proof. Notice that the *a posterior* w_i defined above depend on both $\boldsymbol{z}$ and the parameter θ. By substituting $\boldsymbol{w} = \{w_i\}$ into the incomplete log-likelihood, we can view $L(\theta, \boldsymbol{X})$ as

$$L(\theta, \boldsymbol{X}) \doteq g(\boldsymbol{w}, \theta) \tag{A.29}$$

for some function $g(\cdot)$. Instead of directly maximizing the $L(\theta, \boldsymbol{X})$ with respect to θ, the EM algorithm maximizes the functional $g(\boldsymbol{w}(\theta), \theta)$ in a "hill-climbing" style by iterating between the following two steps:

E Step: partially maximizing $g(\boldsymbol{w}, \theta)$ with respect to $\boldsymbol{w}$ with the second variable θ fixed;

M Step: partially maximizing $g(\boldsymbol{w}, \theta)$ with respect to the second variable θ with $\boldsymbol{w}$ fixed.

Notice that at each step the value of $g(\boldsymbol{w}, \theta)$ does not decrease, so does $L(\theta, \boldsymbol{X})$. When both steps become stationary and no longer increase the value, the process reaches a (local) extremum θ^* of the function $L(\theta, \boldsymbol{X})$. To see this, examine the equation[3]

$$\frac{dL(\theta, \boldsymbol{X})}{d\theta} = \frac{\partial g(\boldsymbol{w}, \theta)}{\partial \boldsymbol{w}} \frac{\partial \boldsymbol{w}}{\partial \theta} + \frac{\partial g(\boldsymbol{w}, \theta)}{\partial \theta}. \tag{A.30}$$

Since at θ^*, each step is stationary, we have $\frac{\partial g(\boldsymbol{w},\theta)}{\partial \boldsymbol{w}} = 0$ and $\frac{\partial g(\boldsymbol{w},\theta)}{\partial \theta} = 0$. Therefore, $\frac{dL(\theta,\boldsymbol{X})}{d\theta}\Big|_{\theta^*} = 0$. □

For a more thorough exposition and complete proof of the convergence of the EM algorithm, one may refer to the book of [McLanchlan and Krishnan, 1997]. However, for the EM algorithm to converge to the maximum-likelihood estimate (usually the global maximum) of $L(\theta, \boldsymbol{X})$, a good initialization is crucial.

A.3 Estimation of Mixture Models

A.3.1 Maximum-Likelihood Estimates

The EM algorithm is often used for estimating a mixture model. By that, we mean the data $\boldsymbol{x}$ is sampled from a distribution which is a superposition of multiple distributions:

$$p(\boldsymbol{x}, \theta) = \pi_1 p_1(\boldsymbol{x}, \theta) + \pi_2 p_2(\boldsymbol{x}, \theta) + \cdots + \pi_n p_n(\boldsymbol{x}, \theta). \tag{A.31}$$

Such a distribution can be easily interpreted as the marginal distribution of a model with a latent random variable $\boldsymbol{z}$ that takes discrete values in $\{1, 2, \ldots, n\}$:

$$\begin{aligned} p(\boldsymbol{x}, \theta) &= \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}, \theta) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z}, \theta) p(\boldsymbol{z}, \theta) \\ &= p(\boldsymbol{x}|\boldsymbol{z} = 1, \theta) p(\boldsymbol{z} = 1, \theta) + \cdots + p(\boldsymbol{x}|\boldsymbol{z} = n, \theta) p(\boldsymbol{z} = n, \theta) \end{aligned}$$

with $p(\boldsymbol{z} = j, \theta) = \pi_j > 0, j = 1, 2, \ldots, n$. Obviously, one can use the EM algorithm to estimate the mixture model, with the mixing weights π_j as part of the unknown model parameters.

Once the model parameters are estimated from the EM algorithm, for a given sample point $\boldsymbol{x}_i$, its "membership" $c(i) \in \{1, 2, \ldots, n\}$, i.e., the component distribution from which $\boldsymbol{x}_i$ is most likely drawn, can be determined by the Bayesian rule from its *a posterior* probability:

$$c(i) = \arg\max_j p(\boldsymbol{z} = j \mid \boldsymbol{x}_i) = \frac{p_j(\boldsymbol{x}_i)}{\pi_1 p_1(\boldsymbol{x}_i) + \cdots + \pi_n p_n(\boldsymbol{x}_i)}. \tag{A.32}$$

[3]Here the "derivative" with respect to $\boldsymbol{w}$ is formal as $\boldsymbol{w}$ is in general a function if $\boldsymbol{z}$ is a continuous random variable. To make the proof here rigorous, one needs to resort to the calculus of variation. For a more careful proof of the convergence of the EM algorithm, one should refer to [McLanchlan and Krishnan, 1997].

A.3.2 Minimax Estimates

Obviously, for the mixture model (A.31), we need to estimate both the distribution parameters θ and the unknown mixing weights π_j. This increases the dimension of the optimization problem that needs to be solved. In practice, we often seek for alternative estimates of the mixture model which do not depend on the mixing weights. Such estimates may no longer be optimal with respect to the above mixture model (A.31) but can be much easier to compute than the ML estimate.

If the mixing weights are not known or not of any interest, the membership of a given sample $\boldsymbol{x}_i$ can be directly determined by the component distribution that returns the highest likelihood: $c(i) = \arg\max_j p_j(\boldsymbol{x}_i) = \arg\min_j -\log p_j(\boldsymbol{x}_i)$.

Therefore, the parameters of the distributions p_j can be estimated by solving the following optimization problem:

$$\min_{\theta} \sum_{i=1}^{N} \big(\min_{j} -\log p_j(\boldsymbol{x}_i, \theta) \big). \tag{A.33}$$

One may interpret the above objective as the follows: For each sample, we find the component distribution for which $\boldsymbol{x}_i$ achieves the highest likelihood; once we have decided to "assign" $\boldsymbol{x}_i$ to the distribution $p_j(\boldsymbol{x}, \theta)$, it takes $-\log p_j(\boldsymbol{x}_i, \theta)$ bits to encode $\boldsymbol{x}_i$. Thus, the above objective function is equivalent to minimize the sum of coding length given the membership of all the samples.

A straightforward way to solve the above optimization problem is to iterate between the following two steps:

Step 1: For fixed θ^k and every $i = 1, 2, \ldots, N$,

$$c^{k+1}(i) = \arg\max_{j} \log p_j(\boldsymbol{x}_i, \theta). \tag{A.34}$$

Step 2: With all $c^{k+1}(i)$ known,

$$\theta^{k+1} = \arg\min_{\theta} \sum_{i=1}^{N} \big(-\log p_{c^{k+1}(i)}(\boldsymbol{x}_i, \theta) \big). \tag{A.35}$$

Notice that the two steps resemble the two steps of the EM algorithm introduced earlier. The difference is that here each sample $\boldsymbol{x}_i$ is assigned to only one of the n groups while in the EM algorithm the hidden variable $\boldsymbol{z}_i$ gives a probabilistic assignment of $\boldsymbol{x}_i$ to the n groups. In fact, the well-known *K-means* algorithm for clustering (see Chapter 2) is essentially based upon the above iteration.

A.4 Model Selection Criteria

So far, we have studied how to solve the following problem: Given N independent samples $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ drawn from a distribution $p(\boldsymbol{x}, \theta)$, where $p(\boldsymbol{x}, \theta)$ belongs to a family of distributions indexed by the parameter θ, how to obtain the (approximate) optimal estimate θ^* of the model parameter. In doing so, we have assumed that the function $p(\boldsymbol{x}, \theta)$ depends smoothly on the parameter θ.

In practice, however, we may not know exactly to which family of distributions the model belongs to. We might only know it belongs to several possible families, $p(\boldsymbol{x}, \theta(m))$, where m is a (discrete) index for the model families. For instance, in the context of GPCA, we try to fit multiple subspaces to a given set of data. However, the number of subspaces and their exact dimensions are sometimes not known or given *a priori*. Thus, determining the number of subspaces and their dimensions is now part of the model estimation problem. Notice that the number of subspaces and their dimensions are discrete variables as opposed to the continuous parameters (e.g., the subspace bases) needed to specify each subspace.

The problem of determining both the model type m and its parameter $\theta(m)$ is conventionally referred to as a *model selection* problem (as opposed to parameter estimation). Many important model-selection criteria have been developed in the statistics community and the algorithmic complexity community for general classes of models. These criteria include

- Akaike Information Criterion (AIC) [Akaike, 1977] (also known as the C_p statistics [Mallows, 1973]) and Geometric AIC (G-AIC) [Kanatani, 2003],
- Bayesian Information Criterion (BIC) (also known as the Schwartz criterion),
- Minimum Description Length (MDL) [Rissanen, 1978] and Minimum Message Length (MML) [Wallace and Boulton, 1968].

Although these criteria are originally motivated and derived from different viewpoints (or in different contexts), they all share a common characteristic: The optimal model should be the one that strikes a good *balance* between the model complexity (typically depends on the dimension of the parameter space) and the data fidelity to the chosen model (typically measured as the sum of squared errors). In fact, some of the criteria are essentially equivalent to each other despite their different origins: To a large extent, the BIC is equivalent to MDL; and the AIC is equivalent to the C_p statistics. Even so, it is impossible to give a detailed review here of all the model selection criteria.

In what follows, we give a brief review of the AIC and the BIC to illustrate the key ideas behind model selection. In Chapter 5, we will further discuss how to modify the AIC in the context of GPCA.

A.4.1 Akaike Information Criterion

Given N independent sample points $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ drawn from a distribution $p(\boldsymbol{x}, \theta_0)$, recall that the maximum-likelihood estimate $\hat{\theta}_N$ of the parameter θ is the one that maximizes the log-likelihood function $L(\theta, \boldsymbol{X}) = \sum_{i=1}^N \log p(\boldsymbol{x}_i, \theta)$.

The *Akaike information criterion* (AIC) for model selection is motivated from an information-theoretic viewpoint. In this approach, the quality of the obtained model is measured by the average code length used by the optimal coding scheme of $p(\boldsymbol{x}, \hat{\theta}_N)$ for a random variable with actual distribution $p(\boldsymbol{x}, \theta_0)$, i.e.,

$$\mathbb{E}[-\log p(\boldsymbol{x}, \hat{\theta}_N)] = \int \big(-\log p(\boldsymbol{x}, \hat{\theta}_N)\big) p(\boldsymbol{x}, \theta_0)\, d\boldsymbol{x}. \tag{A.36}$$

The AIC relies on an approximation to the above expected log-likelihood loss that holds asymptotically as $N \to \infty$:

$$2E[-\log p(\boldsymbol{x}, \hat{\theta}_N)] \approx -\frac{2}{N} L(\hat{\theta}_N, \boldsymbol{X}) + 2\frac{d}{N} \doteq \text{AIC}, \tag{A.37}$$

where d is the number of free parameters for the class of models of interest.

For Gaussian noise models with variance σ^2, we have

$$L(\hat{\theta}_N, \boldsymbol{X}) = -\frac{1}{2\sigma^2} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2,$$

where $\hat{\boldsymbol{x}}_i$ is the best estimate of $\boldsymbol{x}_i$ given the model $p(\boldsymbol{x}, \hat{\theta}_N)$. Thus, if σ^2 is known (or approximated by the empirical sample variance), minimizing the AIC is equivalent to minimizing the so-called C_p statistic:

$$C_p = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2 + 2\frac{d}{N}\sigma^2, \tag{A.38}$$

where the first term is obviously the mean squared error (a measure of data fidelity) and the second term depends linearly on the dimension of the parameter space (a measure of the complexity of the model).

Now consider multiple classes of models whose parameter spaces are of different dimensions. Let us denote the dimension of model class m as $d(m)$. Then the AIC selects the model class m^* that minimizes the following objective function:

$$\text{AIC}(m) = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2 + 2\frac{d(m)}{N}\sigma^2. \tag{A.39}$$

A.4.2 Bayesian Information Criterion

The *Bayesian information criterion* (BIC) for model selection is motivated from a Bayesian inference viewpoint. In this approach, we assume a *prior* distribution of the model $p(\theta|m)$ and wish to choose the model class m^* that maximizes the *posterior* probability $p(m|\boldsymbol{X})$. Using Bayes rule, this is equivalent to maximizing

$$p(m|\boldsymbol{X}) \propto p(m) \cdot p(\boldsymbol{X}|m) = p(m) \cdot \int p(\boldsymbol{X}|\theta, m) p(\theta|m)\, d\theta. \tag{A.40}$$

If we assume that each model class is equally probable, this further reduces to maximizing the likelihood $p(\boldsymbol{X}|m)$ among all the model classes. This is equivalent to minimizing the negative log-likelihood $-2\log p(\boldsymbol{X}|m)$. With certain approximations, one can show that for general distributions the following relationship holds asymptotically as $N \to \infty$:

$$\mathrm{BIC}(m) \doteq -2\log p(\boldsymbol{X}|m) = -2L(\boldsymbol{X}, \hat{\theta}_N) + (\log N)d(m) \tag{A.41}$$

$$= \frac{N}{\sigma^2}\Big[\frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2 + (\log N)\frac{d(m)}{N}\sigma^2\Big]. \tag{A.42}$$

As before, $\hat{\theta}_N$ is the maximum-likelihood estimate of θ given m, $d(m)$ is the number of parameters for class m and σ^2 is the variance of a Gaussian noise model.

Notice that when N and σ are known, the BIC is very similar to the AIC except that the factor 2 in front of the second term in the AIC is replaced by $\log N$ in the BIC. Because we normally have $N \gg e^2$, the BIC penalizes complex models much more than the AIC does. Thus, the BIC tends to choose simpler models. In general, no model selection criterion is always better than others under all circumstances and the best criterion depends on the purpose of the model. From our experience, the AIC tends to provide more satisfactory results for estimation of subspaces. That makes it more favorable in the context of GPCA.

A.5 Robust Statistical Methods

For all the model estimation and selection techniques discussed above, we have always assumed that the given data samples $\{\boldsymbol{x}_i\}_{i=1}^N$ are independent samples drawn from the same distribution $p(\boldsymbol{x}, \theta_0)$. By an appeal to the law of large numbers (or the central limit theorems), the asymptotic optimality of the estimate normally does not depends the particular set of samples given.[4]

However, in many practical situations, the validity of the given data as independent samples of the model becomes questionable. Sometimes, the given data can be corrupted by or mixed with samples of different (probabilistic) nature; or it can simply be the case that the given data are not a typical set of i.i.d. samples from the distribution in question.

For the purpose of model estimation, these seemingly different interpretations are actually equivalent: We need to somehow infer the correct model while *accommodating* an atypical set of samples of the distribution (or the model). Obviously, this is an impossible task unless we impose some restrictions on how "atypical" the samples are. It is customary to assume that only a portion of the samples are somehow different from or inconsistent with the rest of the data. Those samples are often referred to as "outliers" and they may have significant effect on the model inferred from the data.

[4]The fact that almost all sets of i.i.d sample are "typical" or "representative" of the given distribution has been at the heart of the development of Shannon's information theory.

Unfortunately, despite centuries of interest and study[5], there is no universally agreed definition of what an outlier is, especially for multivariate data. Roughly speaking, most definitions (or tests) for an outlier are based on one of the following guidelines:

1. The outliers are a set of samples that have relatively *large influence* on the estimated model parameters. A measure of influence is normally the difference between the model estimated with or without the sample in question.

2. The outliers are a set of *small-probability* samples with respect to the distribution in question. The given data set is therefore an atypical set if such small-probability samples constitute a significant portion of the data.

3. The outliers are a set of samples that are *not consistent* with (the model inferred from) the remainder of the data. A measure of inconsistency is normally the error residual of the sample in question with respect to the model.

Nevertheless, as we will soon see, for popular distributions such as Gaussian, they all lead to more or less equivalent ways of detecting or accommodating outliers. However, under different conditions, different approaches that follow each of the above guidelines may give rise to solutions that can be more convenient and efficient than others.

A.5.1 Influence-Based Outlier Detection

When we try to estimate the parameter of the distribution $p(\boldsymbol{x}, \theta)$ from a set of samples $\{\boldsymbol{x}_i\}_{i=1}^N$, every sample $\boldsymbol{x}_i$ might have uneven effect on the estimated parameter $\hat{\theta}_N$. The samples that have relatively large effect are called *influential samples* and they can be regarded as outliers.

To measure the influence of a particular sample $\boldsymbol{x}_i$, we may compare the difference between the parameter $\hat{\theta}_N$ estimated from all the N samples and the parameter $\hat{\theta}_N^{(i)}$ estimated from all but the ith sample. Without loss of generality, we here consider the maximum-likelihood estimate of the model:

$$\hat{\theta}_N = \arg\max_{\theta} \sum_{j=1}^{N} \log p(\boldsymbol{x}_i, \theta), \tag{A.43}$$

$$\hat{\theta}_N^{(i)} = \arg\max_{\theta} \sum_{j \neq i} \log p(\boldsymbol{x}_i, \theta), \tag{A.44}$$

[5]Earliest documented discussions among astronomers about outliers or “erroneous observations” date back to mid 18th century. See [Barnett and Lewis, 1983, Huber, 1981, Bickel, 1976] for a more thorough exposition of the studies of outliers in statistics.

and measure the influence of $\boldsymbol{x}_i$ on the estimation of θ by the difference

$$I(\boldsymbol{x}_i;\theta) \doteq \hat{\theta}_N - \hat{\theta}_N^{(i)}. \tag{A.45}$$

Assume that $p(\boldsymbol{x},\theta)$ is analytical in θ. Then we have

$$f(\theta) \doteq \sum_{j=1}^{N} \frac{1}{p(\boldsymbol{x}_i,\theta)} \frac{\partial p(\boldsymbol{x}_i,\theta)}{\partial\theta}\Big|_{\theta=\hat{\theta}_N} = 0, \tag{A.46}$$

$$f(\theta)^{(i)} \doteq \sum_{j\neq i} \frac{1}{p(\boldsymbol{x}_i,\theta)} \frac{\partial p(\boldsymbol{x}_i,\theta)}{\partial\theta}\Big|_{\theta=\hat{\theta}_N^{(i)}} = 0. \tag{A.47}$$

If we now evaluate the function $f(\theta)$ at $\theta = \hat{\theta}_N^{(i)}$ using the Taylor series of $f(\theta)$ $\theta = \hat{\theta}_N$ we obtain:

$$f(\hat{\theta}_N^{(i)}) = f(\hat{\theta}_N) + f'(\hat{\theta}_N)(\hat{\theta}_N^{(i)} - \hat{\theta}_N) + o(\|\hat{\theta}_N - \hat{\theta}_N^{(i)}\|). \tag{A.48}$$

Since we have $f(\hat{\theta}_N) = 0$ and $f^{(i)}(\hat{\theta}_N^{(i)}) = 0$, the difference in the estimate caused by the ith sample is

$$\hat{\theta}_N^{(i)} - \hat{\theta}_N \approx \big(f'(\hat{\theta}_N)\big)^{\dagger} \Big[\frac{1}{p(\boldsymbol{x}_i,\hat{\theta}_N^{(i)})} \frac{\partial p(\boldsymbol{x}_i,\hat{\theta}_N^{(i)})}{\partial\theta}\Big]. \tag{A.49}$$

Notice that in the expression on the right hand side, the factor $\big(f'(\hat{\theta}_N)\big)^{\dagger}$ is common for all samples.

Proposition A.11 (Approximate Sample Influence)**.** *The difference between the ML estimate $\hat{\theta}_N$ from N samples and the ML estimate $\hat{\theta}_N^{(i)}$ without the ith sample $\boldsymbol{x}_i$ depends approximately linearly on the quantity:*

$$d(\boldsymbol{x}_i;\theta) \doteq \frac{1}{p(\boldsymbol{x}_i,\hat{\theta}_N^{(i)})} \frac{\partial p(\boldsymbol{x}_i,\hat{\theta}_N^{(i)})}{\partial\theta}. \tag{A.50}$$

In the special case when $p(\boldsymbol{x},\theta)$ is the Gaussian distribution $\mathcal{N}(\mu,\sigma^2)$ with σ^2 known, the above equation gives the influence of the ith sample on the estimate of μ:

$$\hat{\mu}_N^{(i)} - \hat{\mu}_N \approx \alpha(\boldsymbol{x}_i - \hat{\mu}_N^{(i)}), \tag{A.51}$$

where α is some constant depending on σ. That is, the sample influence is very much proportional to the distance between the sample and the mean estimated without the sample; or equivalently, the smaller the probability of a sample is with respect to the estimated (Gaussian) distribution, the larger is its influence on the estimated mean. Therefore, the three guidelines for defining outliers become very much equivalent for a Gaussian distribution.

In general, to evaluate the influence of all the samples, one needs to compute the estimate of the model for $N + 1$ times. That is reasonable to do only if each estimate is not so costly to compute. In light of this drawback, some first order approximations of the influence values were developed at roughly the same period as the sample influence function was proposed [Campbell, 1978, Critchley, 1985], when the computational resources were scarcer than they are today. In robust statistics, formulae that approximate an influence function are referred to as *theoretical influence functions*. One such formula for the influence function of PCA can be found in [Jolliffe, 2002].

A.5.2 Probability-Based Outlier Detection

In general, we assume that the data are drawn from a zero-mean[6] multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{x}})$. Ideally, the principal d-dimensional subspace is spanned by the first d eigenvectors of the covariance matrix $\Sigma_{\boldsymbol{x}}$. Thus, in order to improve the robustness of PCA in the presence of outliers, we essentially seek for a robust estimate of $\Sigma_{\boldsymbol{x}}$.

If there were no outliers, the maximum likelihood estimate of $\Sigma_{\boldsymbol{x}}$ would be given by $\widehat{\Sigma}_N = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i\boldsymbol{x}_i^\top \in \mathbb{R}^{D\times D}$. Therefore, we could approximate the probability that a sample $\boldsymbol{x}_i$ comes from this Gaussian model by

$$p(\boldsymbol{x}_i; \widehat{\Sigma}_N) = \frac{1}{(2\pi)^{D/2}\det(\widehat{\Sigma}_N)^{1/2}} \exp\big(-\frac{1}{2}\boldsymbol{x}_i^\top \widehat{\Sigma}_N^{-1}\boldsymbol{x}_i\big). \tag{A.52}$$

If we adopt the guideline that outliers are samples that have a small probability with respect to the estimated model, then the outliers are exactly those samples that have a relatively large residual:

$$\varepsilon_i = \boldsymbol{x}_i^\top \widehat{\Sigma}_N^{-1}\boldsymbol{x}_i, \quad i = 1, 2, \ldots, N, \tag{A.53}$$

also known as the *Mahalanobis distance*.[7] In terms of the principal components $\boldsymbol{y} = U_d^\top \boldsymbol{x}$, the Mahalanobis distance can also be written as

$$\varepsilon_i = \boldsymbol{y}_i^\top \Sigma_d^{-1}\boldsymbol{y}_i = \sum_{j=1}^d \frac{y_{ij}^2}{\sigma_j^2}, \tag{A.54}$$

where $\Sigma_d \in \mathbb{R}^{d\times d}$ is a diagonal matrix whose jth diagonal entry, σ_j^2, is the jth eigenvalue of $\widehat{\Sigma}_{\boldsymbol{x}}$, or equivalently σ_j is the jth singular value of $\frac{1}{\sqrt{N}}\boldsymbol{X}$.

In principle, we could use $p(\boldsymbol{x}_i, \widehat{\Sigma}_N)$ or ε_i to determine if $\boldsymbol{x}_i$ is an outlier. However, the above estimate of the covariance matrix $\Sigma_{\boldsymbol{x}}$ is obtained using all the samples, including the outliers themselves. Therefore, if $\widehat{\Sigma}_N$ is very different from

[6] We here are only interested in how to robustly estimate the covariance, or "scale," of the distribution. In case the mean, or "location," of the distribution is not known, a separate robust procedure can be employed to determine the mean before the covariance, see [Barnett and Lewis, 1983].

[7] In fact, it can be shown that [Ferguson, 1961], if the outliers have a Gaussian distribution of a different covariance matrix $a\Sigma$, then ε_i is a sufficient statistic for the test that maximizes the probability of correct decision about the outlier (in the class of tests that are invariant under linear transformations). Interested reader may want to find out how this distance is equivalent (or related) to the sample influence $\widehat{\Sigma}_N^{(i)} - \widehat{\Sigma}_N$ or the approximate sample influence given in (A.50).

$\Sigma_{\boldsymbol{x}}$, the outliers could be incorrectly detected. In order to improve the estimate of $\Sigma_{\boldsymbol{x}}$, one can recompute $\widehat{\Sigma}_N$ by discarding or down-weighting samples that have low probability or large Mahalanobis distance. Let $w_i \in [0,1]$ be a weight assigned to the ith point such that $w_i \approx 1$ if $\boldsymbol{x}_i$ is an inlier and $w_i \approx 0$ if $\boldsymbol{x}_i$ is an outlier. Then a new estimate of $\Sigma_{\boldsymbol{x}}$ can be obtained as:

$$\widehat{\Sigma}_N = \frac{w_1^2 \boldsymbol{x}_1 \boldsymbol{x}_1^\top + w_2^2 \boldsymbol{x}_2 \boldsymbol{x}_2^\top + \cdots + w_N^2 \boldsymbol{x}_N \boldsymbol{x}_N^\top}{w_1^2 + w_2^2 + \cdots + w_N^2 - 1}. \tag{A.55}$$

Maximum Likelihood Type Estimators (M-Estimators).

If $w(\varepsilon) \equiv \varepsilon$, the above expression gives the original estimate (**??**) of the covariance matrix. Or, if we want to simply discard all samples with a Mahalanobis distance larger than certain threshold $\varepsilon_0 > 0$, we can choose the following weight function:

$$w(\varepsilon) = \begin{cases} \varepsilon, & \text{for } \varepsilon \leq \varepsilon_0, \\ 0, & \text{for } \varepsilon > \varepsilon_0. \end{cases} \tag{A.56}$$

Nevertheless, under the assumption that the distribution is elliptically symmetric and is contaminated by an associated normal distribution, the following weight function gives a more robust estimate of the covariance matrix [Hampel, 1974, Campbell, 1980]:

$$w(\varepsilon) = \begin{cases} \varepsilon, & \text{for } \varepsilon \leq \varepsilon_0, \\ \varepsilon_0 \exp[-\frac{1}{2a}(\varepsilon - \varepsilon_0)^2] & \text{for } \varepsilon > \varepsilon_0, \end{cases} \tag{A.57}$$

with $\varepsilon_0 = \sqrt{p+b}$ for some suitable choice of positive values for a and b and p denotes the dimension of the space.

Notice that calculating the robust estimate $\hat{\Sigma}_N$ in term of (A.55) is not easy because the weights w_i also depend on the resulting $\hat{\Sigma}_N$. There is no surprise that many known algorithms are based on Monte Carlo [Maronna, 1976, Campbell, 1980].

Many other weight functions have also been proposed in the statistics literature. They serve as the basis for a class of robust estimators, known as *M-estimators* (maximum-likelihood type estimators) [Huber, 1981, Barnett and Lewis, 1983]. Nevertheless, most M-estimators differ only in how the samples are down-weighted but no one seems to dominate others in terms of performance in all circumstances.

Multivariate Trimming (MVT).

One drawback of the M-estimators is that its "breakdown point" is inversely proportional to the dimension of the space. The breakdown point is an important measure of robustness of any estimator: Roughly speaking, it is the smallest proportion of contamination that the estimator can tolerate (or does not diverge). Thus, the M-estimators become much less robust when the dimension is high. This makes M-estimators of limited use in the context of GPCA since the dimension of the space is typically very high (≥ 70).

One way to resolve this problem is to modify the M-estimators by simply trimming out a percentage of the samples with relatively large Mahalanobis distance and then use the remaining samples to re-estimate the covariance matrix. Then each time we have a new estimate of the covariance matrix, we can recalculate the Mahalanobis distance of every sample and reselect samples that need to be trimmed. We can repeat the above process until a stable estimate of the covariance matrix is obtained. This iterative scheme is known as *multivariate trimming* (MVT) – another popular robust estimator. By construction, the breakdown point of MVT does not depend on the dimension of the problem and only depends on the chosen trimming percentage.

When the percentage of outliers is somehow known, it is relatively easy to determine how many samples need to be trimmed. It usually takes only a few iterations for the iteration to converge. However, if the percentage is wrongfully specified, the MVT is known to have trouble to converge or it may converge to a wrong estimate of the covariance matrix. In Chapter **??**, we will discuss in the context of GPCA, how MVT can be modified when the percentage is not known.

A.5.3 Random Sampling-Based Outlier Detection

When the outliers constitute of a large portion (up to 50% or even more than 50%) of the data set, the (ML) estimate $\hat{\theta}_N$ obtained from all the samples can be so severely corrupted that the sample influence and the Mahalanobis distance computed based on it become useless in discriminating outliers from valid samples.[8] This motivates estimating the model parameter θ using only a (randomly sampled) small subset of the samples to begin with. Least median of squares (LMS) and random sample consensus (RANSAC) are two such methods and we now give a brief discussion below.

Least Median Estimation

If we know that only less than half of the samples are potential outliers, it is then reasonable to use only half of the samples to estimate the model parameter. But which half of the samples? We know the maximum-likelihood estimate minimizes the sum of negative log-likelihoods:

$$\hat{\theta}_N = \arg\min_{\theta} \sum_{i=1}^{N} -\log p(\boldsymbol{x}_i, \theta). \tag{A.58}$$

As outliers are the ones of small probability hence large negative log-likelihood, we can order the values of the negative log-likelihood and eliminate from the

[8]Thus, the iterative process is likely to converge to a local minimum other than the true model parameter. Sometimes, it can even be the case that the role of inliers and outliers are exchanged with respect to the converged estimate.

above objective half of the samples that have relatively larger values:

$$\begin{aligned}\hat{\theta}_{N/2} &= \arg\min_{\theta} \sum_j -\log p(\boldsymbol{x}_j, \theta), \quad \text{where} \\ -\log p(\boldsymbol{x}_j, \theta) &\leq \text{median}_{\boldsymbol{x}_i \in \boldsymbol{X}} -\log p(\boldsymbol{x}_i, \theta). \end{aligned} \tag{A.59}$$

A popular approximation to the above objective is to simply minimize the median value of the negative log-likelihood:

$$\hat{\theta}_M \doteq \arg\min_{\theta} \text{median}_{\boldsymbol{x}_i \in \boldsymbol{X}} -\log p(\boldsymbol{x}_i, \theta). \tag{A.60}$$

We call $\hat{\theta}_M$ the *least median estimate*. In the case of Gaussian noise model, $-\log p(\boldsymbol{x}_i, \theta)$ is proportional to the squared error:

$$-\log p(\boldsymbol{x}_i, \theta) \propto \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2.$$

For this reason, the estimate $\hat{\theta}_M$ is more often known as the *least median of squares* (LMS) estimate[9].

However, without knowing θ, it is impossible to order the log-likelihoods or the squared errors, let alone to compute the median. A typical method to resolve this difficulty is to *randomly sample* a number of small subsets of the data:

$$\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_m \subset \boldsymbol{X}, \tag{A.61}$$

where each subset $\boldsymbol{X}_j$ is independently drawn and contains $k \ll N$ samples. So, if p is the fraction of valid samples (the "inliers"), then with probability $q = 1 - (1 - p^k)^m$, one of the above subsets will contain only valid samples. In other words, if we want to be of probability q that one of the selected subsets contains only valid samples, we need to randomly sample at least

$$m \geq \frac{\log(1-q)}{\log(1-p^k)} \tag{A.62}$$

subsets of k samples.

Using each subset $\boldsymbol{X}_j$, we can compute an estimate $\hat{\theta}_j$ of the model and use the estimate to compute the median for the remaining $N-k$ samples in $\boldsymbol{X} \setminus \boldsymbol{X}_j$:

$$\hat{M}_j \doteq \text{median}_{\boldsymbol{x}_i \in \boldsymbol{X} \setminus \boldsymbol{X}_j} -\log p(\boldsymbol{x}_i, \hat{\theta}_j). \tag{A.63}$$

Then the least median estimate $\hat{\theta}_M$ is approximated by the $\hat{\theta}_{j^*}$ that gives the smallest median $\hat{M}_{j^*} = \min_j \hat{M}_j$.

In the case of Gaussian noise model, based on the order statistics of squared errors, we can use the median statistic to obtain an (asymptotically unbiased) estimate of the variance, or scale, of the error as follows:

$$\hat{\sigma} = \frac{N+5}{N\Phi^{-1}(0.5 + p/2)} \sqrt{\text{median}_{\boldsymbol{x}_i \in \boldsymbol{X}} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2}, \tag{A.64}$$

[9]The importance of median for robust estimation were pointed out first in the article of [Hampel, 1974].

where $p = 0.5$ for the median statistic. One then can use $\hat{\sigma}$ to find "good" samples in $\boldsymbol{X}$ whose squared errors are less than $\lambda\sigma^2$ for some chosen constant λ (normally less than 5). Using such good samples, we can recompute a more efficient (ML) estimate $\hat{\theta}$ of the model.

Random Sample Consensus (RANSAC)

In theory, the breakdown point of the least median estimate is up to 50% outliers. In many practical situations however, there might be more than half outlying samples in the data. Random Sample Consensus (RANSAC) [Fischler and Bolles, 1981] is a method that is designed to work for such highly contaminated data.

In many aspects, RANSAC is actually very much similar to LMS. The main difference is that instead of looking at the median statistic,[10] RANSAC try to find, among all the estimates $\{\hat{\theta}_j\}$ obtained from the subsets $\{\boldsymbol{X}_j\}$, the one that maximizes the number of samples that have error residual (measured either by the negative log-likelihood or the squared error) smaller than a pre-specified error tolerance:

$$\hat{\theta}_{j^*} \doteq \arg\max_{\hat{\theta}_j} \#\{\boldsymbol{x}_i \in \boldsymbol{X} : -\log(\boldsymbol{x}_i, \hat{\theta}_j) \leq \tau\}. \tag{A.65}$$

In other words, $\hat{\theta}_{j^*}$ achieves the highest "consensus" among all the sample estimates $\{\hat{\theta}_j\}$, hence the name "random sample consensus" (RANSAC). To improve the efficiency of the estimate, we can recompute an ML estimate $\hat{\theta}$ of the model from all the samples that are consistent with $\hat{\theta}_{j^*}$.

Notice that for RANSAC, one needs to specify the error tolerance τ *a priori*. In other words, RANSAC requires to know the variance σ^2 of the error *a priori*, while LMS normally does not. There have been a few variations of RANSAC in the literature that relax this requirement. We here do not elaborate on them and interested readers may refer to [Steward, 1999] and references therein.

However, in the context of GPCA, the random sampling techniques have not been so effective. The reason is largely because the number of subsets needed grows prohibitively high when the dimension of the model is large or the model is a mixture model such as an arrangement of subspaces. Other complications may also arise when dealing with a mixture model. We will give a more detailed account of these complications in Chapter 5.

[10] which becomes meaningless when the fraction of outliers is over 50%.

Appendix B
Basic Facts from Algebraic Geometry

"Algebra is but written geometry; geometry is but drawn algebra."
– Sophie Germain

As a centuries-old practice in science and engineering, people often fit polynomials to a given set of data points. In this book, we often use the set of zeros of (multivariate) polynomials to model a given data set. In mathematics, polynomials and their zero sets are studied in Algebraic Geometry, with Hilbert's Nullstellensatz establishing the basic link between Algebra (polynomials) and Geometry (the zero set of polynomials, a geometric object). In order to make this book self-contained, in this appendix, we review some of the basic notions and facts that are frequently used in this book. For a more systematic introduction to this topic, the reader may refer to the classic texts of Lang [Lang, 1993] and Eisenbud [Eisenbud, 1996].

B.1 Polynomial Ring

Consider a D-dimensional vector space over a field R (of characteristic 0), denoted by R^D, where R is usually the field of real numbers $\mathbb{R}$ or the field of complex numbers $\mathbb{C}$.

Let $R[\boldsymbol{x}] = [x_1, x_2, \ldots, x_D]$ be the set of all polynomials of D variables $x_1, x_2, \ldots, x_D$. Then $R[\boldsymbol{x}]$ is a *commutative ring* with two basic operations: "summation" and "multiplication" of polynomials. The elements of R are called *scalars* or *constants*. A *monomial* is a product of the variables; its degree is the number of the variables (counting repeats). A monomial of degree n is of the form $\boldsymbol{x}^{\boldsymbol{n}} = x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$ with $0 \leq n_j \leq n$ and $n_1 + n_2 + \cdots + n_D = n$. There are a total of

$$M_n(D) \doteq \binom{D+n-1}{n} = \binom{D+n-1}{D-1}$$

different degree-n monomials.

Definition B.1 (Veronese Map)**.** *For given n and D, the* Veronese map *of degree n, denoted as $\nu_n : R^D \to R^{M_n(D)}$, is defined as:*

$$\nu_n : \ [x_1, \ldots, x_D]^T \mapsto [\ldots, \boldsymbol{x}^{\boldsymbol{n}}, \ldots]^T, \tag{B.1}$$

where $\boldsymbol{x}^{\boldsymbol{n}}$ are degree-n monomials of the form $x_1^{n_1} x_2^{n_2} \cdots x_D^{n_D}$ with $\boldsymbol{n} = (n_1, n_2, \ldots, n_D)$ chosen in the degree-lexicographic order.

Example B.2 (The Veronese Map of Degree 2 in 3 Variables). If $\boldsymbol{x} = [x_1, x_2, x_3]^T \in R^3$, the Veronese map of degree 2 is given by:

$$\nu_2(\boldsymbol{x}) = [x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2]^T \in R^6.$$

■

In the context of Kernel methods (Chapter 2), the Veronese map is usually referred to as the polynomial embedding and the ambient space $R^{M_n(D)}$ is called the *feature* space.

A *term* is a scalar multiplying a monomial. A polynomial $p(\boldsymbol{x})$ is said to be *homogeneous* if all its terms have the same degree. Sometimes, the word *form* is used to mean a homogeneous polynomial. Every homogeneous polynomial $p(\boldsymbol{x})$ of degree n can be written as:

$$p(\boldsymbol{x}) = \boldsymbol{c}_n^T \nu_n(\boldsymbol{x}) = \sum c_{n_1,\ldots,n_D} x_1^{n_1} \cdots x_D^{n_D}, \tag{B.2}$$

where $c_{n_1,\ldots,n_D} \in R$ are the coefficients associated with the monomials $\boldsymbol{x}^{\boldsymbol{n}} = x_1^{n_1} \cdots x_D^{n_D}$.

In this book, we are primarily interested in the *algebra* of homogeneous polynomials with D variables.[1] Because of that, we view R^D as a projective space – the set of one-dimensional subspaces (meaning lines through the origin). Any one-dimensional subspace, say a line L, can be represented by a point $[a_1, a_2, \ldots, a_D]^T \neq [0, 0, \ldots, 0]^T$ on the line. The result is a projective $(D-1)$-space over R which can be regarded as the D-tuples $[a_1, a_2, \ldots, a_D]^T$ of elements of R, modulo the equivalence relation $[a_1, a_2, \ldots, a_D]^T \sim [ba_1, ba_2, \ldots, ba_D]^T$ for all $b \neq 0$ in R.

If $p(x_1, x_2, \ldots, x_D)$ is a homogeneous polynomial of degree n, then for $b \in R$ we have

$$p(ba_1, ba_2, \ldots, ba_D) = b^n p(a_1, a_2, \ldots, a_D). \tag{B.3}$$

[1]For algebra of polynomials defined on R^D as an affine space, the reader may refer to [Lang, 1993].

Therefore, whether $p(a_1, a_2, \dots, a_D) = 0$ or not on a line L does not depend on the representative point chosen on the line L.

We may view $R[\boldsymbol{x}]$ as a *graded ring* which can be decomposed as

$$R[\boldsymbol{x}] = \bigoplus_{i=0}^{\infty} R_i = R_0 \oplus R_1 \oplus \cdots \oplus R_n \oplus \cdots , \tag{B.4}$$

where R_i consists of all polynomials of degree i. In particular, $R_0 = R$ is the set of nonzero scalars (or constants). It is convention (and convenient) to define the degree of the zero element, 0, in R to be infinite or -1. R_1 is the set of all homogeneous polynomials of degree one, i.e., the set of 1-forms,

$$R_1 \doteq \big\{ b_1 x_1 + b_2 x_2 + \cdots + b_D x_D : \ [b_1, b_2, \dots, b_D]^T \in R^D \big\}. \tag{B.5}$$

Obviously, the dimension of R_1 as a vector space is also D. R_1 can also be viewed as the dual space $(R^D)^*$ of R^D. For convenience, we also define the following two sets

$$\begin{aligned} R_{\leq m} &\doteq \bigoplus_{i=0}^{m} R_i = R_0 \oplus R_1 \oplus \cdots \oplus R_m, \\ R_{\geq m} &\doteq \bigoplus_{i=m}^{\infty} R_i = R_m \oplus R_{m+1} \oplus \cdots , \end{aligned}$$

which are the set of polynomials of degree up to degree m and those of degree higher and equal to m, respectively.

B.2 Ideals and Algebraic Sets

Definition B.3 (Ideal)**.** *An* ideal *in the (commutative) polynomial ring $R[\boldsymbol{x}]$ is an additive subgroup I (with respect to the summation of polynomials) such that if $p(\boldsymbol{x}) \in I$ and $q(\boldsymbol{x}) \in R[\boldsymbol{x}]$, then $p(\boldsymbol{x})q(\boldsymbol{x}) \in I$.*

From the definition, it is easy to verify that if I, J are two ideals of $R[\boldsymbol{x}]$, their intersection $K = I \cap J$ is also an ideal. The previously defined set $R_{\geq m}$ is an ideal for every m. In particular, $R_{\geq 1}$ is the so-called *irrelevant ideal*, sometimes denoted by R_+.

An ideal is said to be *generated* by a subset $\mathcal{G} \subset I$ if every element $p(\boldsymbol{x}) \in I$ can be written in the form

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} q_i(\boldsymbol{x}) g_i(\boldsymbol{x}), \quad \text{with } \ q_i(\boldsymbol{x}) \in R[\boldsymbol{x}] \ \text{ and } \ g_i(\boldsymbol{x}) \in \mathcal{G}. \tag{B.6}$$

We write $(\mathcal{G})$ for the ideal generated by a subset $\mathcal{G} \subset R[\boldsymbol{x}]$; if $\mathcal{G}$ contains only a finite number of elements $\{g_1, \dots, g_k\}$, we usually write $(g_1, \dots, g_k)$ in place of $(\mathcal{G})$. An ideal I is *principal* if it can be generated by one element (i.e., $I =$

$p(\boldsymbol{x})R[\boldsymbol{x}]$ for some polynomial $p(\boldsymbol{x})$). Given two ideals I and J, the ideal that is generated by the product of elements in I and J

$$\{f(\boldsymbol{x})g(\boldsymbol{x}),\ f(\boldsymbol{x}) \in I, g(\boldsymbol{x}) \in J\}$$

is called the *product ideal*, denoted as IJ.

An ideal I of the polynomial ring $R[\boldsymbol{x}]$ is *prime* if $I \neq R[\boldsymbol{x}]$ and if $p(\boldsymbol{x}), q(\boldsymbol{x}) \in R[\boldsymbol{x}]$ and $p(\boldsymbol{x})q(\boldsymbol{x}) \in I$ implies that $p(\boldsymbol{x}) \in I$ or $q(\boldsymbol{x}) \in I$. If I is prime, then for any ideals J, K with $JK \subseteq I$ we have $J \subseteq I$ or $K \subseteq I$.

A polynomial $p(\boldsymbol{x})$ is said to be *prime* or irreducible if $p(\boldsymbol{x})$ generates a prime ideal. Equivalently, if $p(\boldsymbol{x})$ is irreducible if $p(\boldsymbol{x})$ is not a nonzero scalar and whenever $p(\boldsymbol{x}) = f(\boldsymbol{x})g(\boldsymbol{x})$, then one of $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ is a nonzero scalar.

Definition B.4 (Homogeneous Ideal)**.** *A homogeneous ideal of $R[\boldsymbol{x}]$ is an ideal that is generated by homogeneous polynomials.*

Note that the sum of two homogeneous polynomials of different degrees is no longer a homogeneous polynomial. Thus, a homogeneous ideal contains nonhomogeneous polynomials too.

Definition B.5 (Algebraic Set)**.** *Given a set of homogeneous polynomials $J \subset R[\boldsymbol{x}]$, we may define a corresponding (projective) algebraic set $Z(J)$ as a subset of R^D to be*

$$Z(J) \doteq \{[a_1, a_2, \ldots, a_D]^T \in R^D | f(a_1, a_2, \ldots, a_D) = 0, \forall f \in J\}. \quad \text{(B.7)}$$

If we view algebraic sets as the closed sets of R^D, this assigns a topology to the space R^D, which is called the *Zariski topology*.[2]

If $X = Z(J)$ is an algebraic set, an algebraic subset $Y \subset X$ is a set of the form $Y = Z(K)$ (where K is a set of homogeneous polynomials) that happens to be contained in X. A nonempty algebraic set is said to be *irreducible* if it is not the union of two nonempty smaller algebraic subsets. We call irreducible algebraic sets as *algebraic varieties*. For instance, any subspace of R^D is an irreducible algebraic variety.

There is an inverse construction of algebraic sets. Given any subset $X \subseteq R^D$, we define the *vanishing ideal of* X to be the set of all polynomials that vanish on X:

$$I(X) \doteq \{f(\boldsymbol{x}) \in R[\boldsymbol{x}] | f(a_1, a_2, \ldots, a_n) = 0, \forall [a_1, a_2, \ldots, a_n]^T \in X\}. \quad \text{(B.8)}$$

One can easily verify that $I(X)$ is an ideal. Treating two polynomials as equivalent if they agree at all the points of X, we get the *coordinate ring* $A(X)$ of X as the quotient $R[\boldsymbol{x}]/I(X)$.

Now, consider a set of homogeneous polynomials $J \subset R[\boldsymbol{x}]$ (which is not necessarily an ideal) and a subset $X \subset R^D$ (which is not necessarily an algebraic set.)

[2]This is because the intersection of any algebraic sets is an algebraic set; and the union of finitely many algebraic sets is also an algebraic set.

Proposition B.6. *The following facts are true:*

1. *$I(Z(J))$ is an ideal that contains J;*
2. *$Z(I(X))$ is an algebraic set that contains X.*

Proposition B.7. *If X is an algebraic set and $I(X)$ is the ideal of X, then X is irreducible if and only if I is a prime ideal.*

Proof. If X is irreducible and $f(\boldsymbol{x})g(\boldsymbol{x}) \in I$, since $Z(\{I, f(\boldsymbol{x})\}) \cup Z(\{I, g(\boldsymbol{x})\}) = X$, then either $X = Z(\{I, f(\boldsymbol{x})\})$ or $X = Z(\{I, g(x)\})$. That is, either $f(\boldsymbol{x})$ or $g(\boldsymbol{x})$ vanishes on X and is in I. Conversely, suppose $X = X_1 \cup X_2$. If both X_1 and X_2 are algebraic sets strictly smaller than X, then there exist polynomials $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ that vanish on X_1 and X_2 respectively, but not on X. Since the product $f_1(\boldsymbol{x})f_2(\boldsymbol{x})$ vanishes on X, we have $f_1(\boldsymbol{x})f_2(\boldsymbol{x}) \in I$ but neither $f_1(\boldsymbol{x})$ or $f_2(\boldsymbol{x})$ is in I. So I is not prime. □

B.3 Algebra and Geometry: Hilbert's Nullstellensatz

In practice, we often use an algebraic set to model a given set of data points and the (ideal of) polynomials that vanish on the set provide a natural parametric model for the data. One question that is of particular importance in this context is: Is there an one-to-one correspondence between ideals and algebraic sets? This is in general not true as the ideals $I = (f^2(\boldsymbol{x}))$ and $J = (f(\boldsymbol{x}))$ both vanish on the same algebraic set as the zero-set of the polynomial $f(\boldsymbol{x})$. Fortunately, this turns out to be essentially the only case that prevents the one-to-one correspondence between ideals and algebraic sets.

Definition B.8 (Radical Ideal)**.** *Given a (homogeneous) ideal I of $R[\boldsymbol{x}]$, the (homogeneous) radical ideal of I is defined to be*

$$\mathit{rad}(I) \doteq \{f(\boldsymbol{x}) \in R[\boldsymbol{x}] | f(\boldsymbol{x})^m \in I \textit{ for some integer } m\}. \tag{B.9}$$

We leave it to the reader to verify that $\mathrm{rad}(I)$ is indeed an ideal and furthermore, if I is homogeneous, so is $\mathrm{rad}(I)$.

Hilbert proved in 1893 the following important theorem that establishes one of the fundamental results in algebraic geometry:

Theorem B.9 (Nullstellensatz)**.** *Let R be an algebraically closed field (e.g., $R = \mathbb{C}$). If $I \subset R[\boldsymbol{x}]$ is an (homogeneous) ideal, then*

$$I(Z(I)) = \mathit{rad}(I). \tag{B.10}$$

Thus, the correspondences $I \mapsto Z(I)$ and $X \mapsto I(X)$ induce a one-to-one correspondence between the collection of (projective) algebraic sets of R^D and (homogeneous) radical ideals of $R[\boldsymbol{x}]$.

One may find up to five different proofs for this theorem in [Eisenbud, 1996].[3] The importance of the Nullstellensatz cannot be exaggerated. It is a natural extension of Gauss' fundamental theorem of algebra[4] to multivariate polynomials. One of the remarkable consequences of the Nullstellensatz is that it identifies a geometric object (algebraic sets) with an algebraic object (radical ideals).

In our context, we often assume our data points are drawn from an algebraic set and use the set of vanishing polynomials as a parametric model for the data. Hilbert's Nullstellensatz guarantees such a model for the data is well-defined and unique. To some extent, when we fit vanishing polynomials to the data, we are essentially inferring the underlying algebraic set. In the next section, we will discuss how to extend Hilbert's Nullstellensatz to the practical situation in which we only have finitely many sample points from an algebraic set.

B.4 Algebraic Sampling Theory

We often face a common mathematical problem: How to identify a (projective) algebraic set $Z \subseteq R^D$ from a finite, though maybe very large, number of sample points in Z? In general, the algebraic set Z is not necessarily irreducible[5] and the ideal $I(Z)$ is not necessarily prime.

From an algebraic viewpoint, it is impossible to recover a continuous algebraic set Z from a finite number of discrete sample points. To see this, note that the set of all polynomials that vanish on one (projective) point $\boldsymbol{z}$ is a submaximal ideal[6] $\mathfrak{m}$ in the (homogeneous) polynomial ring $R[\boldsymbol{z}]$. The set of polynomials that vanish on a set of sample points $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_i\} \subseteq Z$ is the intersection

$$\mathfrak{a}_i \doteq \mathfrak{m}_1 \cap \mathfrak{m}_2 \cap \cdots \cap \mathfrak{m}_i, \tag{B.11}$$

which is a radical ideal that is typically much larger than $I(Z)$.

Thus, some additional assumptions must be imposed on the algebraic set in order to make the problem of inferring $I(Z)$ from the samples well-defined. Typically, we assume that the ideal $I(Z)$ of the algebraic set Z in question is generated by a set of (homogeneous) polynomials whose degrees are bounded by a relatively small n. That is,

$$\begin{aligned} I(Z) &\doteq \big(f_1, f_2, \ldots, f_s\big) \quad \text{s.t.} \quad \deg(f_j) \leq n, \\ Z(I) &\doteq \big\{\boldsymbol{z} \in R^D \mid f_i(\boldsymbol{z}) = 0, i = 1, 2, \ldots, s\big\}. \end{aligned}$$

[3] Strictly speaking, for homogeneous ideals, for the one-to-one correspondence to be exact, one should only consider proper radical ideals.

[4] Every degree-n polynomial in one variable has exactly n roots in an algebraically closed field such as $\mathbb{C}$ (counting repeats).

[5] For instance, it is often the case that Z is the union of many subspaces or algebraic surfaces.

[6] The ideal of a point in the affine space is a maximal ideal; and the ideal of a point in the projective space is called a submaximal ideal. They both are "maximal" in the sense that they cannot be a subideal of any other homogeneous ideal of the polynomial ring.

We are interested in retrieving $I(Z)$ uniquely from a set of sample points $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_i\} \subseteq Z$. In general, $I(Z)$ is always a proper subideal of $\mathfrak{a}_i$, regardless of how large i is. However, the information about $I(Z)$ can still be retrieved from $\mathfrak{a}_i$ in the following sense.

Theorem B.10 (Sampling of an Algebraic Set)**.** *Consider a nonempty set $Z \subseteq R^D$ whose vanishing ideal $I(Z)$ is generated by polynomials in $R_{\leq n}$. Then there is a finite sequence $F_N = \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_N\}$ such that the subspace $I(F_N) \cap R_{\leq n}$ generates $I(Z)$.*

Proof. Let $I_{\leq n} = I(Z) \cap R_{\leq n}$. This vector space generates $I(Z)$. Let $\mathfrak{a}_0 = R[\boldsymbol{x}] = I(\emptyset)$. Let $\mathfrak{b}_0 = \mathfrak{a}_0 \cap R_{\leq n}$ and let $A_0 = (\mathfrak{b}_0)$, the ideal generated by polynomials in $\mathfrak{a}_0$ of degree less than or equal to n. Since $1 \in R[\boldsymbol{x}] \cap R_{\leq n}$ is the generator of this ideal, we have $A_0 = R[\boldsymbol{x}]$. Since $Z \neq \emptyset$, then $A_0 \neq I(Z)$. Set $N = 1$ and pick a point $\boldsymbol{z}_1 \in Z$. Then $1(\boldsymbol{z}_1) \neq 0$ (1 is the function that assigns 1 to every point of Z.). Let $\mathfrak{a}_1$ be the ideal that vanishes on $\{\boldsymbol{z}_1\}$ and define $\mathfrak{b}_1 = \mathfrak{a}_1 \cap R_{\leq n}$. Further let $A_1 = (\mathfrak{b}_1)$.[7] Since $I(Z) \subseteq \mathfrak{a}_1$, it follows that $I_{\leq n} \subseteq \mathfrak{b}_1$. If $A_1 = I(Z)$, then we are done. Suppose then that $I(Z) \subset A_1$.

Let us do the induction at this point. Suppose we have found a finite sequence $F_N = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_N\} \subset Z$ with

$$\begin{aligned} I(F_N) &= \mathfrak{a}_N && \text{(B.12)}\\ \mathfrak{b}_N &= \mathfrak{a}_N \cap R_{\leq n} && \text{(B.13)}\\ A_N &= (\mathfrak{b}_N) && \text{(B.14)}\\ \mathfrak{b}_0 \supset \mathfrak{b}_1 \supset &\cdots \supset \mathfrak{b}_N \supseteq I_{\leq n}. && \text{(B.15)} \end{aligned}$$

It follows that $I_{\leq n} \subseteq \mathfrak{b}_N$ and that $I(Z) \subseteq A_N$. If equality holds here, then we are done. If not, then there is a function $g \in \mathfrak{b}_N$ not in $I(Z)$ and an element $\boldsymbol{z}_{N+1} \in Z$ for which $g(\boldsymbol{z}_{N+1}) \neq 0$. Set $F_{N+1} = \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_N, \boldsymbol{z}_{N+1}\}$. Then one gets $\mathfrak{a}_{N+1}, \mathfrak{b}_{N+1}, A_{N+1}$ as before with

$$\mathfrak{b}_0 \supset \mathfrak{b}_1 \supset \cdots \supset \mathfrak{b}_N \supset \mathfrak{b}_{N+1} \supseteq I_{\leq n}. \tag{B.16}$$

We obtain a descending chain of subspaces of the vector space $R_{\leq n}$. This chain must stabilize, since the vector space is finite dimensional. Hence there is an N for which $\mathfrak{b}_N = I_{\leq n}$ and we are done. □

We point out that in the above proof, no clear bound on the total number N of points needed is given.[8] Nevertheless, from the proof of the theorem, the set of finite sequences of samples that satisfy the theorem is an open set. This is of great

[7] Here we are using the convention that (S) is the ideal generated by the set S. Recall also that the ring $R[\boldsymbol{x}]$ is *noetherian* by the Hilbert basis theorem and so all ideals in the ring are finitely generated [Lang, 1993].

[8] However, loose bounds can be easily obtained from the dimension of $R_{\leq n}$ as a vector space. In fact, in the algorithm, we implicitly used the dimension of $R_{\leq n}$ as a bound for N.

practical importance: With probability one, the vanishing ideal of an algebraic set can be correctly determined from a randomly chosen sequence of samples.

Example B.11 (A Hyperplane in $\mathbb{R}^3$). Consider a plane $P = \{\boldsymbol{z} \in R^3 : f(\boldsymbol{z}) = az_1 + bz_2 + cz_3 = 0\}$. Given any two points in general position in the plane P, $f(\boldsymbol{x}) = ax_1 + bx_2 + cx_3$ will be the only (homogeneous) polynomial of degree 1 that fits the two points. In terms of the notation introduced earlier, we have $I(P) = (\mathfrak{a}_2 \cap R_{\leq 1})$. ■

Example B.12 (Zero Polynomial). When $Z = R^D$, the only polynomial that vanishes on Z is the zero polynomial, i.e., $I(Z) = (0)$. Since the zero polynomial is regarded to be of degree -1, we have $(\mathfrak{a}_N \cap R_{\leq n}) = \emptyset$ for any given n (and large enough N). ■

The above theorem can be viewed as a first step towards an algebraic analogy to the well-known Nyquist-Shannon sampling theory in signal processing, which stipulates that a continuous signal with a limited frequency bandwidth Ω can be uniquely determined from a sequence of discrete samples with a sampling rate higher than 2Ω. Here a signal is replaced by an algebraic set and the frequency bandwidth is replaced by the bound on the degree of polynomials. It has been widely practiced in engineering that a curve or surface described by polynomial equations can be recovered from a sufficient number of sample points in general configuration, a procedure often loosely referred to as "polynomial fitting." However, the algebraic basis for this is often not clarified and the conditions for the uniqueness of the solution are usually not well characterized or specified. This problem certainly merits further investigation.

B.5 Decomposition of Ideals and Algebraic Sets

Modeling a data set as an algebraic set does not stop at obtaining its vanishing ideal (and polynomials). The ultimate goal is to extract all the internal geometric or algebraic structures of the algebraic set. For instance, if an algebraic set consists of multiple subspaces, called a subspace arrangement, we need to know how to derive from its vanishing ideal the number of subspaces, their dimensions, and a basis of each subspace.

Thus, given an algebraic set X or equivalently its vanishing ideal $I(X)$, we want to decompose or segment it into a union of subsets each of which can no longer be further decomposed. As we have mentioned earlier, an algebraic set that cannot be decomposed into smaller algebraic sets is called irreducible. As one of the fundamental finiteness theorem of algebraic geometry, we have:

Theorem B.13. *An algebraic set can have only finitely many irreducible components. That is, for some n,*

$$X = X_1 \cup X_2 \cup \cdots \cup X_n, \tag{B.17}$$

where $X_1, X_2, \ldots, X_n$ are irreducible algebraic varieties.

Proof. The proof is essentially based on the fact that the polynomial ring $R[\boldsymbol{x}]$ is Noetherian (i.e., finitely generated), and there are only finitely many prime ideals containing $I(X)$ that are minimal with respect to inclusion (See [Eisenbud, 1996]). □

The vanishing ideal $I(X_i)$ of each irreducible algebraic variety X_i must be a prime ideal that is minimal over the radical ideal $I(X)$ – there is no prime subideal of $I(X_i)$ that includes $I(X)$. The ideal $I(X)$ is precisely the intersection of all the minimal prime ideals:

$$I(X) = I(X_1) \cap I(X_2) \cap \cdots \cap I(X_n). \tag{B.18}$$

This intersection is called a *minimal primary decomposition* of the radical ideal $I(X)$. Thus the primary decomposition of a radical ideal is closely related to the notion of "segmenting" or "decomposing" an algebraic set into multiple irreducible algebraic varieties: If we know how to decompose the ideal, we can easily find the irreducible algebraic variety corresponding to each primary component.

We are particularly interested in a special class of algebraic sets known as subspace arrangements. One of the goals of generalized principal component analysis (GPCA) is to decompose a subspace arrangement into individual (irreducible) subspaces (see Chapter 3). In Appendix C, we will further study the algebraic properties of subspace arrangements.

B.6 Hilbert Function, Polynomial, and Series

Finally, we introduce an important invariant of algebraic sets, given by the Hilbert function. Knowing the values of Hilbert function can be very useful in the identification of subspace arrangements, especially the number of subspaces and their dimensions.

Given a (projective) algebraic set Z and its vanishing ideal $I(Z)$, We can grade the ideal by degree as

$$I(Z) = I_0(Z) \oplus I_1(Z) \oplus \cdots \oplus I_i(Z) \oplus \cdots . \tag{B.19}$$

The *Hilbert function* of Z is defined to be

$$h_I(i) \doteq \dim(I_i(Z)). \tag{B.20}$$

Notice that $h_I(i)$ is exactly the number of linearly independent polynomials of degree i that vanish on Z. In this book, we also refer to h_I as the Hilbert function of the algebraic set Z.[9]

[9]In the literature, however, the Hilbert function of an algebraic set Z is sometimes defined to be the dimension of the homogeneous components of the coordinate ring $A(Z) \doteq R[\boldsymbol{x}]/I(Z)$ of Z, which is the codimension of $I_i(Z)$ as a subspace in R_i.

The *Hilbert series*, also known as the Poincaré series, of the ideal I is defined to be the power series[10]

$$\mathcal{H}(I,t) \doteq \sum_{i=0}^{\infty} h_I(i)t^i = h_I(0) + h_I(1)t + h_I(2)t^2 + \cdots . \tag{B.21}$$

Thus, given $\mathcal{H}(I,t)$, we know all the values of the Hilbert function h_I from its coefficients.

Example B.14 (Hilbert Series of the Polynomial Ring). The Hilbert series of the polynomial ring $R[\boldsymbol{x}] = \mathbb{R}[x_1, x_2, \ldots, x_D]$ is

$$\mathcal{H}(R[\boldsymbol{x}],t) = \sum_{i=0}^{\infty} \dim(R_i)t^i = \sum_{i=0}^{\infty} \tbinom{D+i-1}{i} t^i = \frac{1}{(1-t)^D}. \tag{B.22}$$

One can easily verify the correctness of the formula with the special case $D = 1$. Obviously, the coefficients of the Hilbert series of any ideal (as a subset of $R[\boldsymbol{x}]$) are bounded by those of $\mathcal{H}(R[\boldsymbol{x}],t)$ and hence the Hilbert series converges. ■

Example B.15 (Hilbert Series of a Subspace). The above formula can be easily generalized to the vanishing ideal of a subspace S of dimension d in $\mathbb{R}^D$. Let the co-dimension of the subspace be $c = D - d$. We have

$$\mathcal{H}(I(S),t) = \left(\frac{1}{(1-t)^c} - 1\right) \cdot \left(\frac{1}{(1-t)^{D-c}}\right) = \frac{1-(1-t)^c}{(1-t)^D}. \tag{B.23}$$

■

The following theorem, also due to Hilbert, reveals that the values of the Hilbert function of an ideal have some remarkable properties:

Theorem B.16 (Hilbert Polynomial)**.** *Let $I(Z)$ be the vanishing ideal of an algebraic set Z over $R[x_1, \ldots, x_D]$, then the values of its Hilbert function $h_I(i)$ agree, for large i, with those of a polynomial of degree $\leq D$. This polynomial, denoted as $H_I(i)$, is called the* Hilbert polynomial *of $I(Z)$.*

Then in the above example, for the polynomial ring, the Hilbert function itself is obviously a polynomial in i

$$H_R(i) = h_R(i) = \tbinom{D+i-1}{i} = \frac{1}{(D-1)!}(D+i-1)(D+i-2)\cdots(i+1).$$

However, for a general ideal I (of an algebraic set), it is not necessarily true that all values of its Hilbert function h_I agree with those of its Hilbert polynomial H_I. They might agree only when i is large enough. Thus, for a given algebraic set (or ideal), it would be interesting to know how large i needs to be in order for the Hilbert function to coincide with a polynomial. As we will see in Appendix B, for subspace arrangements, there is a very elegant answer to this question. One can

[10]In general, the Hilbert series can be defined for any finitely-generated graded module $E = \bigoplus_{i=1}^{\infty} E_i$ using any Euler-Poincaré $\mathbb{Z}$-valued function $h_E(\cdot)$ as $\mathcal{H}(E,t) \doteq \sum_{i=0}^{\infty} h_E(i)t^i$ [Lang, 1993]. Here, for $E = I$, we choose $h_I(i) = \dim(I_i)$.

even derive closed-form formulae for the Hilbert polynomials. These results are very important and useful for Generalized Principal Component Analysis, both conceptually and computationally.

Appendix C
Algebraic Properties of Subspace Arrangements

"He who seeks for methods without having a definite problem in mind seeks in the most part in vain."

– David Hilbert

In this book, the main problem that we study is how to segment a collection of data points drawn from a subspace arrangement $\mathcal{A} = \{S_1, S_2, \ldots, S_n\}$, formally introduced in Chapter 4.[1] $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$ is the union of all the subspaces. $Z_{\mathcal{A}}$ can be naturally described as the zero set of a set of polynomials, which makes it an *algebraic set*. The solution to the above problem typically relies on inferring the subspace arrangement $Z_{\mathcal{A}}$ from the data points. Thus, knowing the algebraic properties of $Z_{\mathcal{A}}$ may significantly facilitate this task.

Although subspace arrangements seem to be a very simple class of algebraic sets, a full characterization of their algebraic properties is a surprisingly difficult, if not impossible, task. Subspace arrangements have been a centuries-old subject that still actively interweaves many mathematical fields: algebraic geometry and topology, combinatorics and complexity theory, graph and lattice theory, etc. Although the results are extremely rich and deep, in fact only a few special classes of subspace arrangements have been well characterized.

In this appendix, we examine some important concepts and properties of subspace arrangements that are closely related to the subspace-segmentation problem. The purpose of this appendix is two-fold: 1. to provide a rigorous jus-

[1]Unless stated otherwise, the subspace arrangement considered will always be a central arrangement, as in Definition 3.4.

tification for the GPCA algorithms derived in the book, especially Chapter 3; 2. to introduce important properties of subspace arrangements, which may suggest potential improvements of the algorithms. For readers who are interested only in the basic GPCA algorithms and their applications, this appendix can be skipped at first read.

C.1 Ideals of Subspace Arrangements

Vanishing Ideal of a Subspace.

A d-dimensional subspace S can be defined by $k = D - d$ linearly independent linear forms $\{l_1, l_2, \ldots, l_k\}$:

$$S \doteq \{\boldsymbol{x} \in R^D : l_i(\boldsymbol{x}) = 0,\ i = 1, 2, \ldots, k = D - d\}, \tag{C.1}$$

where l_i is of the form $l_i(\boldsymbol{x}) = a_{i1}x_1 + a_{i2}x_2 + \cdots a_{iD}x_D$ with $a_{ij} \in R$. Let S^* denote the space of all linear forms that vanish on S, then $\dim(S^*) \doteq k = D - d$. The subspace S is also called the zero set of S^*, i.e., points in the ambient space that vanish on all polynomials in S^*, which is denoted as $Z(S^*)$. We define

$$I(S) \doteq \{p \in R[\boldsymbol{x}] : p(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \in S\}. \tag{C.2}$$

Clearly, $I(S)$ is an ideal generated by linear forms in S^*, and it contains polynomials of all degrees that vanish on the subspace S. Every polynomial $p(\boldsymbol{x})$ in $I(S)$ can be written as a superposition:

$$p = l_1 h_1 + l_2 h_2 + \cdots + l_k h_k \tag{C.3}$$

for some polynomials $h_1, h_2, \ldots, h_k \in R[\boldsymbol{x}]$. Furthermore, $I(S)$ is a prime ideal.[2]

Vanishing Ideal of a Subspace Arrangement.

Given a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$, its vanishing ideal is

$$I(Z_{\mathcal{A}}) = I(S_1) \cap I(S_2) \cap \cdots \cap I(S_n). \tag{C.4}$$

The ideal $I(Z_{\mathcal{A}})$ can be graded by the degree of the polynomial

$$I(Z_{\mathcal{A}}) = I_m(Z_{\mathcal{A}}) \oplus I_{m+1}(Z_{\mathcal{A}}) \oplus \cdots \oplus I_i(Z_{\mathcal{A}}) \oplus \cdots . \tag{C.5}$$

Each $I_i(Z_{\mathcal{A}})$ is a vector space that consists of forms of degree i in $I(Z_{\mathcal{A}})$, and $m \geq 1$ is the least degree of the polynomials in $I(Z_{\mathcal{A}})$. Notice that forms that vanish on $Z_{\mathcal{A}}$ may have degrees strictly less than n. One example is an arrangement of two lines and one plane in $\mathbb{R}^3$. Since any two lines lie on a plane, the arrangement can be embedded into a hyperplane arrangement of two planes, and

[2]It is a prime ideal because for any product $p_1 p_2 \in I(S)$, either $p_1 \in I(S)$ or $p_2 \in I(S)$.

there exist forms of second degree that vanish on the union of the three subspaces. The dimension of $I_i(Z_{\mathcal{A}})$ is known as the Hilbert function $h_I(i)$ of $Z_{\mathcal{A}}$.

Example C.1 (Boolean Arrangement). The Boolean arrangement is the collection of coordinate hyperplanes $H_j \doteq \{\boldsymbol{x} : x_j = 0\}, 1 \leq j \leq D$. The vanishing ideal of the Boolean arrangement is generated by a single polynomial $p(\boldsymbol{x}) = x_1 x_2 \cdots x_D$ of degree D. ■

Example C.2 (Braid Arrangement). The Braid arrangement is the collection of hyperplanes $H_{jk} \doteq \{\boldsymbol{x} : x_j - x_k = 0\}, 1 \leq j \neq k \leq D$. Similarly, the vanishing ideal the Braid arrangement is generated by a single polynomial $p(\boldsymbol{x}) = \prod_{1 \leq j < k \leq D}(x_j - x_k)$. ■

Theorem C.3 (Regularity of Subspace Arrangements)**.** *The vanishing ideal $I(Z_{\mathcal{A}})$ of a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$ is n-regular. This implies that $I(Z)$ has a set of generators with degree $\leq n$.*

Proof. For the concept of n-regularity and the proof of the above statement, please refer to [Derksen, 2005] and references therein. □

Due to the above theorem, the subspace arrangement $Z_{\mathcal{A}}$ is uniquely determined as the zero set of all polynomials of degree up to n in its vanishing ideal, i.e., as the zero set of polynomials in

$$Z_{\mathcal{A}} = Z(I_{(n)}),$$

where $I_{(n)} \doteq I_0 \oplus I_1 \oplus \cdots \oplus I_n$.

Product Ideal of a Subspace Arrangement

Let $J(Z_{\mathcal{A}})$ be the ideal generated by the products of linear forms

$$\{l_1 \cdot l_2 \cdots l_n, \quad \forall l_j \in S_j^*, j = 1, \ldots, n\}.$$

Or equivalently, we can define $J(Z_{\mathcal{A}})$ to be the product of the n ideals $I(S_1), I(S_2), \ldots, I(S_n)$:

$$J(Z_{\mathcal{A}}) \doteq I(S_1) \cdot I(S_2) \cdots I(S_n).$$

Then, the *product ideal* $J(Z_{\mathcal{A}})$ is a subideal of $I(Z_{\mathcal{A}})$. Nevertheless, the two ideals share the same zero set:

$$Z_{\mathcal{A}} = Z(J) = Z(I). \tag{C.6}$$

By definition I is the largest ideal that vanishes on $Z_{\mathcal{A}}$. I is in fact the *radical ideal* of the product ideal J, i.e., $I = \text{rad}(J)$. We may also grade the ideal $J(Z_{\mathcal{A}})$ by the degree

$$J(Z_{\mathcal{A}}) = J_n(Z_{\mathcal{A}}) \oplus J_{n+1}(Z_{\mathcal{A}}) \oplus \cdots \oplus J_i(Z_{\mathcal{A}}) \oplus \cdots . \tag{C.7}$$

Notice that, unlike I, the lowest degree of polynomials in J always starts from n, the number of subspaces. The Hilbert function of J is denoted as $h_J(i) = \dim(J_i(Z_{\mathcal{A}}))$. As we will soon see, the Hilbert functions (or polynomials, or series) of the product ideal J and the vanishing ideal I have very interesting and important relationships.

C.2 Subspace Embedding and PL-Generated Ideals

Let $Z_{\mathcal{A}}$ be a central subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$. Let $Z_{\mathcal{A}'} = S'_1 \cup S'_2 \cup \cdots \cup S'_{n'}$ be another (central) subspace arrangement. If we have $Z_{\mathcal{A}} \subseteq Z_{\mathcal{A}'}$, then it is necessary that for all $S_j \subset Z_{\mathcal{A}}$ there exists $S'_{j'} \subset Z_{\mathcal{A}'}$ such that $S_j \subseteq S'_{j'}$. If so, we call

$$Z_{\mathcal{A}} \subseteq Z_{\mathcal{A}'}$$

a *subspace embedding*. Beware that it is possible $n' < n$ for a subspace embedding as more than one subspace S_j of $Z_{\mathcal{A}}$ may belong to the same subspace $S_{j'}$ of $Z_{\mathcal{A}'}$. The subspace arrangements in Theorem 3.13 are examples of subspace embedding. If $Z_{\mathcal{A}'}$ happens to be a hyperplane arrangement, we call the embedding a *hyperplane embedding*.

Is the zero-set of each homogeneous component of $I(Z_{\mathcal{A}})$, in particular $I_m(Z_{\mathcal{A}})$, a subspace embedding of $Z_{\mathcal{A}}$? Unfortunately, this is not true as counter examples can be easily constructed.

Example C.4 (Five Lines in $\mathbb{R}^3$). Consider five points in $\mathbb{P}^2$ (or equivalently, five lines in $\mathbb{R}^3$) The Veronese embedding of order two of a point $\boldsymbol{x} = [x_1, x_2, x_3] \in \mathbb{R}^3$ is $[x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2] \in \mathbb{R}^6$. For five points in general position, the matrix $\boldsymbol{V}_2 = [\nu_2(\boldsymbol{x}_1), \nu_2(\boldsymbol{x}_2), \ldots \nu_2(\boldsymbol{x}_5)]$ is of rank 5. Let $\boldsymbol{c}^T$ be the only vector in the left null space of $\boldsymbol{V}_2$: $\boldsymbol{c}^T\boldsymbol{V}_2 = 0$. Then $p(\boldsymbol{x}) = \boldsymbol{c}^T\nu_2(\boldsymbol{x})$ is in general an irreducible quadratic polynomial. Thus, the zero-set of $I_2(Z_{\mathcal{A}}) = p(\boldsymbol{x})$ is not a subspace arrangement but an (irreducible) cone in $\mathbb{R}^3$. ■

Nevertheless, the following statement allows us to retrieve a subspace embedding from any polynomials in the vanishing ideal $I(Z_{\mathcal{A}})$.

Theorem C.5 (Hyperplane Embedding via Differentiation)**.** *For every polynomial p in the vanishing ideal $I(Z_{\mathcal{A}})$ of a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$ and n points $\{\boldsymbol{x}_j \in S_j\}_{j=1}^n$ in general position, the union of the hyperplanes $\cup_{j=1}^n H_j = \{\boldsymbol{x} : Dp(\boldsymbol{x}_j)^T\boldsymbol{x} = 0\}$ is a hyperplane embedding of the subspace arrangement.*

Proof. The proof is based on the simple fact that the derivative (gradient) $\nabla f(\boldsymbol{x})$ of any smooth function $f(\boldsymbol{x})$ is orthogonal to (the tangent space of) its level set $f(\boldsymbol{x}) = c$. □

In the above statement, if we replace p with a collection of polynomials in the vanishing ideal, their derivatives give a subspace embedding in a similar fashion as the hyperplane embedding. When the collection contains all the generators of the vanishing ideal, the subspace embedding becomes tight – the resulting subspace arrangement coincides with the original one. This property has been used in the development of GPCA algorithms in Chapter 3.

Another concept that is closely related to subspace embedding is a *pl-generated ideal*.

Definition C.6 (pl-Generated Ideals)**.** *An ideal is said to be* pl-generated *if it is generated by products of linear forms.*

If the ideal of a subspace arrangement $Z_{\mathcal{A}}$ is pl-generated, then the zero-set of every generator gives a hyperplane embedding of $Z_{\mathcal{A}}$.

Example C.7 (Hyperplane Arrangements). If $Z_{\mathcal{A}}$ is a hyperplane arrangement, $I(Z_{\mathcal{A}})$ is always pl-generated as it is generated by a single polynomial of the form:[3]

$$p(\boldsymbol{x}) = (\boldsymbol{b}_1^T\boldsymbol{x})(\boldsymbol{b}_2^T\boldsymbol{x})\cdots(\boldsymbol{b}_n^T\boldsymbol{x}), \tag{C.8}$$

where $\boldsymbol{b}_i \in R^D$ are the normal vectors to the hyperplanes. ■

Obviously, the vanishing ideal $I(S)$ of a single subspace S is always pl-generated. The following example shows that this is also true for an arrangement of two subspaces.

Example C.8 (Two Subspaces). Let us show that for an arrangement $Z_{\mathcal{A}}$ of two subspaces, $I(Z_{\mathcal{A}})$ is always pl-generated. Let $Z_{\mathcal{A}} = S_1 \cup S_2$ and define $U^* \doteq S_1^* \cap S_2^*$ and $V^* \doteq S_1^* \setminus U^*, W^* \doteq S_2^* \setminus U^*$. Let $(u_1, u_2, \ldots, u_k)$ be a basis for U^*, $(v_1, v_2, \ldots, v_l)$ a basis for V^*, and $(w_1, w_2, \ldots, w_m)$ a basis for W^*. Then obviously $I(Z_{\mathcal{A}}) = I(S_1) \cap I(S_2)$ is generated by $(u_1, \ldots, u_k, v_1w_1, v_1w_2, \ldots, v_lw_m)$. ■

Now consider an arrangement of n subspaces: $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$. By its definition, the product ideal $J(Z_{\mathcal{A}})$ is always pl-generated. Now, is the vanishing ideal $I(Z_{\mathcal{A}})$ always pl-generated? Unfortunately, this is not true. Below are some counterexamples.

Example C.9 (Lines in $\mathbb{R}^3$ [?]). For a central arrangement $Z_{\mathcal{A}}$ of r lines in general position in $\mathbb{R}^3$, $I(Z_{\mathcal{A}})$ is not pl-generated when $r = 5$ or $r > 6$. Example C.4 gives a proof for the case with $r = 5$. ■

Example C.10 (Planes in $\mathbb{R}^4$ [?]). For a central arrangement $Z_{\mathcal{A}}$ of r planes in general position in $\mathbb{R}^4$, $I(Z_{\mathcal{A}})$ is not pl-generated for all $r > 2$. ■

However, can each homogeneous component $I_i(Z_{\mathcal{A}})$ be "pl-generated" when i is large enough? For instance, can it be that $I_n = J_n = S_1^* \cdot S_2^* \cdots S_n^*$? This is in general not true for an arbitrary arrangement and below is a counterexample.

Example C.11 (Three Subspaces in $\mathbb{R}^5$ – due to R. Fossum). Consider $R[\boldsymbol{x}] = \mathbb{R}[x_1, \ldots, x_5]$ and an arrangement $Z_{\mathcal{A}}$ of three three-dimensional subspaces in $\mathbb{R}^5$ whose vanishing ideals are given by, respectively:

$$I(S_1) = (x_1, x_2), \quad I(S_2) = (x_3, x_4), \quad I(S_3) = ((x_1 + x_3), (x_2 + x_4)).$$

Denote their intersection as $I = I(S_1) \cap I(S_2) \cap I(S_3)$. The intersection contains the element

$$x_1x_4 - x_2x_3 = (x_1 + x_3)x_4 - (x_2 + x_4)x_3 = x_1(x_2 + x_4) - x_2(x_1 + x_3).$$

Then any element $(x_1x_4 - x_2x_3)l(x_1, \ldots, x_5)$ with l a linear form is in $I_3(Z_{\mathcal{A}})$, the homogeneous component of elements of degree three. In particular, $(x_1x_4 - x_2x_3)x_5$ is in $I_3(Z_{\mathcal{A}})$. However, it is easy to check that this element cannot be written in the form

$$\sum_i (a_ix_1 + b_ix_2)(c_ix_2 + d_ix_4)(e_i(x_1 + x_3) + f_i(x_2 + x_4))$$

[3] In algebra, an ideal which is generated by a single generator is called a principal ideal.

for any $a_i, b_i, c_i, d_i, e_i, f_i \in \mathbb{R}$. Thus, $I_3(Z_{\mathcal{A}})$ is not spanned by $S_1^* \cdot S_2^* \cdot S_3^*$. ■

However, notice that the subspaces in the above example are not in "general position" – their intersections are not of the minimum possible dimension. Could $I_n = J_n = S_1^* \cdot S_2^* \cdots S_n^*$ be instead true for n subspaces if they are in general position? The answer is yes. In fact, we can say more than that. As we will see in the next section, from the Hilbert functions of I and J, we actually have

$$I_i = J_i, \quad \forall i \geq n$$

if $S_1, S_2, \dots, S_n$ are "transversal" (i.e., all intersections are of minimum possible dimension). In other words, J_i could differ from I_i only for $i < n$.

C.3 Hilbert Functions of Subspace Arrangements

In this section, we study the Hilbert functions of subspace arrangements defined in Section B.6. We first discuss a few reasons why in the context of generalized principal component analysis, it is very important to know the values of the Hilbert function for the vanishing ideal I or the product ideal J of a subspace arrangement. We then examine the values of the Hilbert function for a few special examples. Finally, we give a complete characterization of the Hilbert function, the Hilbert polynomial, and the Hilbert series of a general subspace arrangement. In particular, we give a closed-form formula for the Hilbert polynomial of the vanishing ideal and the product ideal of the subspace arrangement.

C.3.1 Relationships between the Hilbert Function and GPCA

In general, for a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$ in general position, the values of the Hilbert function $h_I(i)$ of its vanishing ideal $I(Z_{\mathcal{A}})$ are invariant under a continuous change of the positions of the subspaces. They depend only on the dimensions of the subspaces $d_1, d_2, \dots, d_n$ or their co-dimensions $c_i = D - d_i, i = 1, 2, \dots, n$. Thus, the Hilbert function gives a rich set of invariants of subspace arrangements. In the context of GPCA, such invariants can help to determine the type of the subspace arrangement, such as the number of subspaces and their individual dimensions from a given set of (possibly noisy) sample points.

To see this, consider a sufficiently large number of sample points in general position are drawn from the subspaces $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\} \subset Z_{\mathcal{A}}$, let the embedded data matrix (via the Veronese map of degree i) to be

$$\boldsymbol{V}_i \doteq [\nu_i(\boldsymbol{x}_1), \nu_i(\boldsymbol{x}_2), \dots, \nu_i(\boldsymbol{x}_N)]^T. \tag{C.9}$$

According to the Algebraic Sampling Theorem of Appendix B, the dimension of Null$(\boldsymbol{V}_i)$ is exactly the number of linearly independent polynomials of degree i that vanish on $Z_{\mathcal{A}}$. That is, the following relation holds

$$\dim(\text{Null}(\boldsymbol{V}_i)) = h_I(i) \tag{C.10}$$

or equivalently,

$$\operatorname{rank}(\boldsymbol{V}_i) = \dim(R_i) - h_I(i). \tag{C.11}$$

Thus, if we know the Hilbert function for different subspace arrangements in advance, we can determine from the rank of the data matrix from which subspace arrangement the sample data points are drawn. The following example illustrates the basic idea.

Example C.12 (Three Subspaces in $\mathbb{R}^3$). Suppose that we only know our data are drawn from an arrangement of three subspaces in $\mathbb{R}^3$. There are in total four different types of such arrangements, shown in Figure C.1. The values of their corresponding Hilbert function are listed in Table C.1. Given a sufficiently large number N of sample points from one of the

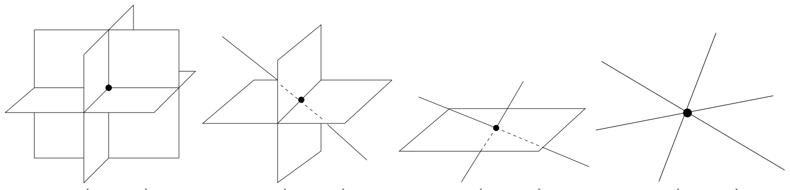


Figure C.1. Four configurations of three subspaces in $\mathbb{R}^3$. The numbers are the co-dimensions (c_1, c_2, c_3) of the subspaces.

c_1	c_2	c_3	$h_{I(Z_\mathcal{A})}(1)$	$h_{I(Z_\mathcal{A})}(2)$	$h_{I(Z_\mathcal{A})}(3)$
1	1	1	0	0	1
1	1	2	0	0	2
1	2	2	0	1	4
2	2	2	0	3	7

Table C.1. Values of the Hilbert function of the four arrangements (assuming the subspaces are in general position).

above subspace arrangements, the rank of the embedded data matrix $\boldsymbol{V}_3 \in \mathbb{R}^{N\times 10}$ can be, instead of any value between 1 and 10, only $10 - h_I(3) = 9, 8, 6, 3$, which correspond to the only four possible configurations of three subspaces in $\mathbb{R}^3$: three planes, two planes and one line, one plane and two lines, or three lines, respectively, as shown in Figure C.1.

This suggests that, given the dimensions of individual subspaces, we may know the rank of the embedded data matrix. Conversely, given the rank of the embedded data matrix, we can determine to a large extent the possible dimensions of the individual subspaces. Therefore, knowing the values of the Hilbert function will help us to at least rule out in advance impossible rank values for the embedded data matrix or the impossible subspace dimensions. This is particularly useful when the data is corrupted by noise so that there is ambiguity in determining the rank of the embedded data matrix or the dimensions of the subspaces. ■

The next example illustrates how the values of Hilbert function can help determine the correct number of subspaces.

Example C.13 (Over-Fit Hyperplane Arrangements in $\mathbb{R}^5$). Consider a dataset sampled from a number of hyperplanes in general position in $\mathbb{R}^5$. Suppose we only know that the number of the hyperplanes is at most 4, and we embed the data via the degree-4 Veronese map anyway. Table C.2 gives the possible values of the Hilbert function for an arrangement of 4, 3, 2, 1 hyperplanes in $\mathbb{R}^5$, respectively. Here we use the convention that an empty set has co-dimension 5 in $\mathbb{R}^5$.

c_1	c_2	c_3	c_4	$h_{I(Z_{\mathcal{A}})}(4)$	rank$(\boldsymbol{V}_4)$
1	1	1	1	1	69
1	1	1	5	5	65
1	1	5	5	15	55
1	5	5	5	35	35

Table C.2. Values of the Hilbert function of (codimension-1) hyperplane arrangements in $\mathbb{R}^5$.

The first row shows that if the number of hyperplanes is exactly equal to the degree of the Veronese map, then $h_I(4) = 1$, i.e., the data matrix $\boldsymbol{V}_4$ has a rank-1 null space. The following rows show the values of $h_I(4)$ when the number of hyperplanes is $n = 3, 2, 1$, respectively. If the rank of the matrix $\boldsymbol{V}_4$ matches any of these values, we know exactly the number of hyperplanes in the arrangement. Figure C.2 shows a super-imposed plot of the singular values of $\boldsymbol{V}_4$ for samples points drawn from $n = 1, 2, 3, 4$ hyperplanes in $\mathbb{R}^5$, respectively.

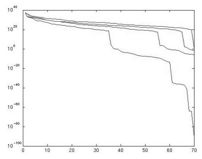


Figure C.2. A super-imposed semi-log plot of the singular values of the embedded data matrix $\boldsymbol{V}_4$ for $n = 1, 2, 3, 4$ hyperplanes in $\mathbb{R}^5$, respectively. The rank drops at $35, 55, 65, 69$, which confirm the theoretical values of the Hilbert function.

Thus, in general, knowing the values of $h_I(i)$ even for $i > n$ may significantly help determine the correct number of subspaces in case the degree i of the Veronese map used for constructing the data matrix $\boldsymbol{V}_i$ is strictly higher than the number n of non-trivial subspaces in the arrangement. ■

The above examples show merely a few cases in which the values of Hilbert function may facilitate solving the GPCA problem. In Chapter **??**, we will see how the Hilbert function can help to improve the performance of GPCA. It now remains as a question how to compute the values of Hilbert function for arbitrary subspace arrangements.

Mathematically, we are interested in finding closed-form formulae, if exist at all, for the Hilbert function (or the Hilbert polynomial, or the Hilbert series) of the subspace arrangements. As we will soon show, if the subspace arrangements are transversal (i.e., any intersection of subset of the subspaces has the smallest possible dimension), we are able to show that the Hilbert function (of both I and J) agrees with the Hilbert polynomial (of both I and J) with $i \geq n$; and a closed-form formula for the Hilbert polynomial is known (and will be given later). However, no general formula is known for the Hilbert function (or series) of I, especially for the values $h_I(i)$ with $i < n$. For those values, one can still compute them in advance numerically based on the identity

$$h_I(i) = \dim(\text{Null}(\boldsymbol{V}_i)) \tag{C.12}$$

from a sufficient set of samples on the subspace arrangements. The values for each type of arrangements need to be computed only once, and the results can be stored in a table such as Table C.1 for each ambient space dimension D and number of subspaces n. We may later query these tables to retrieve information about the subspace arrangements and exploit relations among these values for different practical purposes.

However, computing the values of h_I numerically can be very expensive, especially when the dimension of the space (or the subspaces) is high. In order to densely sample the high-dimensional subspaces, the number of samples grows exponentially with the number of subspaces and their dimensions. Actually the MATLAB package that we are using runs out of the memory limit of 2GB for computing the table for the case $D = 12$ and $n = 6$.

Fortunately, for most applications in image processing, or computer vision, or systems identification, it is typically sufficient to know the values of $h_I(i)$ up to $n = 10$ and $D = 12$. For instance, for most images, the first $D = 12$ principal components already keep up to 99% of the total energy of the image, which is more than sufficient for any subsequent representation or compression purposes. Furthermore, if one chooses to use two by two blocks to represent a color image, then each block becomes one data point of dimension $2 \times 2 \times 3 = 12$. The number of segments sought for an image is typically less than ten. In system identification, the dimensions of the subspaces correspond to the orders of the systems and they are typically less than 10.

C.3.2 Special Cases of the Hilbert Function

Before we study the Hilbert function for general subspace arrangements in the next section, we here give a few special cases for which we have computed certain values of the Hilbert function.

Example C.14 (Hyperplane Arrangements). Consider $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \ldots \cup S_n \subset \mathbb{R}^D$ with each S_i a hyperplane. The subspaces S_i are of co-dimension 1, i.e., $c_1 = c_2 = \cdots = c_n = 1$. Then we have $h_I(n) = 1$, which is consistent with the fact there is exactly one (factorable) polynomial of degree n that fits n hyperplanes. Furthermore, $h_I(i) = 0$ for all $i < n$ and

$$h_I(n+i) = \binom{D+i-1}{i}, \quad \forall i \geq 1.$$

We can generalize the case of hyperplanes to the following example. ■

Example C.15 (Subspaces Whose Duals Have No Intersection). Consider a subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \ldots \cup S_n \subset \mathbb{R}^D$ with $S_i^* \cap S_j^* = 0$ for all $i \neq j$. In other words, if the co-dimensions of $S_1, S_2, \ldots, S_n$ are $c_1, c_2, \ldots, c_n$, respectively, we have $c_1 + c_2 + \cdots + c_n \leq D$. Notice that hyperplane arrangements are a special case here. Generalizing the result in Example B.15, one can easily show that the Hilbert series of $I(Z_{\mathcal{A}})$ (and $J(Z_{\mathcal{A}})$) is

$$\mathcal{H}(I(Z_{\mathcal{A}}), t) = \mathcal{H}(J(Z_{\mathcal{A}}), t) = f(t) \doteq \frac{\prod_{i=1}^n \left(1 - (1-t)^{c_i}\right)}{(1-t)^D}. \tag{C.13}$$

The values of the Hilbert function $h_I(i)$ can be easily computed from the coefficients of the function $f(t)$ associated with t^i. ■

However, if the dual subspaces S_i^* do have non-trivial intersections, the computation of Hilbert series and function becomes much more complicated. Below we give some special examples and leave the general study to the next section.

Example C.16 (Hilbert Function of Two Subspaces). We here derive a closed-form formula of $h_I(2)$ for an arrangement of $n = 2$ subspaces $Z_{\mathcal{A}} = S_1 \cup S_2$ in general position (see also Example C.8). Suppose their co-dimensions are c_1 and c_2, respectively. In $R_1 \sim \mathbb{R}^D$, the intersection of their dual subspaces S_1^* and S_2^* has the dimension

$$c \doteq \max\{c_1 + c_2 - D,\ 0\}. \tag{C.14}$$

Then we have

$$\begin{aligned} h_I(2) &= c \cdot (c+1)/2 + c \cdot (c_1 - c) + c \cdot (c_2 - c) + (c_1 - c) \cdot (c_2 - c) \\ &= c_1 \cdot c_2 - c \cdot (c-1)/2. \end{aligned} \tag{C.15}$$

■

Example C.17 (Three Subspaces in $\mathbb{R}^5$). Consider an arrangement of three subspaces $Z_{\mathcal{A}} = S_1 \cup S_2 \cup S_3 \subset \mathbb{R}^5$ in general position. After a change of coordinates, we may assume $S_1^* = \text{span}\{x_1, x_2, x_3\}$, $S_2^* = \text{span}\{x_1, x_4, x_5\}$, and $S_3^* = \text{span}\{x_2, x_3, x_4, x_5\}$. The value of $h_I(3)$ in this case is equal to $\dim(S_1^* \cdot S_2^* \cdot S_3^*)$. Firstly, we compute $S_1^* \cdot S_2^*$ and obtain a basis for it:

$$S_1^* \cdot S_2^* = \text{span}\{x_1^2, x_1x_4, x_1x_5, x_2x_1, x_2x_4, x_2x_5, x_3x_1, x_3x_4, x_3x_5\}.$$

From this, it is then easy to compute the basis for $S_1^* \cdot S_2^* \cdot S_3^*$:

$$\begin{aligned} S_1^* \cdot S_2^* \cdot S_3^* = \text{span}\{&x_1^2x_2, x_1x_2x_4, x_1x_2x_5, x_1x_2^2, x_2^2x_4, x_2^2x_5, x_1x_2x_3, x_2x_3x_4, \\ &x_2x_3x_5, x_1^2x_3, x_1x_3x_4, x_1x_3x_5, x_1x_3^2, x_3^2x_4, x_3^2x_5, x_1^2x_4, x_1x_4^2, \\ &x_1x_4x_5, x_2x_4^2, x_2x_4x_5, x_3x_4^2, x_3x_4x_5, x_1^2x_5, x_1x_5^2, x_2x_5^2, x_3x_5^2\}. \end{aligned}$$

Thus, we have $h_I(3) = 26$. ■

Example C.18 (Five Subspaces in $\mathbb{R}^3$). Consider an arrangement of five subspaces $S_1, S_2, \ldots, S_5$ in $\mathbb{R}^3$ of co-dimensions $c_1, c_2, \ldots, c_5$, respectively. We want to compute the value of $h_I(5)$, i.e., the dimension of homogeneous polynomials of degree five that vanish on the five subspaces $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_5$. For all the possible values of $1 \leq c_1 \leq c_2 \leq \cdots \leq c_5 < 3$, we have computed the values of $\mathcal{D}_5^3$ and listed them in Table C.3. Notice that the values of $h_I(3)$ in the earlier Table C.1 is a subset of those of $h_I(5)$ in Table C.3. In fact, many relationships like this one exist among the values of the

c_1	c_2	c_3	c_4	c_5	$h_I(5)$
1	1	1	1	1	1
1	1	1	1	2	2
1	1	1	2	2	4
1	1	2	2	2	7
1	2	2	2	2	11
2	2	2	2	2	16

Table C.3. Values of the Hilbert function $h_I(5)$ for arrangements of five subspaces in $\mathbb{R}^3$.

Hilbert function. If properly harnessed, they can significantly reduce the amount of work for computing the values of the Hilbert function. ■

Example C.19 (Five Subspaces in $\mathbb{R}^4$). Similar to the above example, we have computed the values of $h_I(5)$ for arrangements of five linear subspaces in $\mathbb{R}^4$. The results are given in Table C.4. In fact, using the numerical method described earlier, we have computed using computer the values of $h_I(5)$ up to five subspaces in $\mathbb{R}^{12}$. ■

c_1	c_2	c_3	c_4	c_5	$h_I(5)$
1	1	1	1	1	1
1	1	1	1	2	2
1	1	1	1	3	3
1	1	1	2	2	4
1	1	1	2	3	6
1	1	1	3	3	8
1	1	2	2	2	8
1	1	2	2	3	11
1	1	2	3	3	14
1	1	3	3	3	17
1	2	2	2	2	15
1	2	2	2	3	19
1	2	2	3	3	23
1	2	3	3	3	27
1	3	3	3	3	31
2	2	2	2	2	26
2	2	2	2	3	31
2	2	2	3	3	36
2	2	3	3	3	41
2	3	3	3	3	46
3	3	3	3	3	51

Table C.4. Values of the Hilbert function $h_I(5)$ for arrangements of five subspaces in $\mathbb{R}^4$.

C.3.3 Formulae for the Hilbert Function

In this section, we give a general formula for the Hilbert polynomial of the subspace arrangement $Z_{\mathcal{A}} = S_1 \cup S_2 \cup \cdots \cup S_n$. However, due to the limit of space, we will not be able to give a detailed proof for all the results given here. Interested readers may refer to [Derksen, 2005].

Let U be any subset of the set of indexes $\underline{n} \doteq \{1, 2, \ldots, n\}$, we define the following ideals

$$I_U \doteq \bigcap_{u \in U} I(S_u), \quad J_U \doteq \prod_{u \in U} I(S_u). \tag{C.16}$$

If U is empty, we use the convention $I_\emptyset = J_\emptyset = R$. We further define $V_U = \bigcap_{u \in U} S_u$, $d_U = \dim(V_U)$, and $c_U = D - d_U$.

Let us define polynomials $p_U(t)$ recursively as follows. First we define

$$p_\emptyset(t) = 1.$$

For $U \neq \emptyset$ and $p_W(t)$ is already defined for all proper subsets W of U, then $p_U(t)$ is uniquely determined by the following equation

$$\sum_{W \subseteq U} (-t)^{|W|} p_W(t) \equiv 0 \bmod (1-t)^{c_U}, \quad \deg(p_U(t)) < c_U. \tag{C.17}$$

Here $|W|$ is the number of indexes in the set W.

With the above definitions, the Hilbert series of the product ideal J is given by

$$\mathcal{H}(J, t) = \frac{p_{\underline{n}}(t) t^n}{(1-t)^D}. \tag{C.18}$$

That is, the Hilbert series of the product ideal J depends only on the numbers $c_U, U \subseteq \underline{n}$. Thus, the values of the Hilbert function $h_J(i)$ are all combinatorial invariants – invariants that depend only on the values $\{c_U\}$ but not the particular position of the subspaces.

Definition C.20 (Transversal Subspaces)**.** *The subspaces $S_1, S_2, \ldots, S_n$ are called* transversal *if $c_U = \min\left(D, \sum_{u \in U} c_u\right)$ for all $U \subseteq \underline{n}$. In other words, the intersection of any subset of the subspaces has the smallest possible dimension.*

Notice that the notion of "transversality" defined here is less strong than the typical notion of "general position." For instance, according to the above definition, three coplanar lines (through the origin) in $\mathbb{R}^3$ are transversal. However, they are not "in general position."

Theorem C.21. *Suppose that $S_1, S_2, \ldots, S_n$ are transversal, then $\mathcal{H}(I, t) - f(t)$ and $\mathcal{H}(J, t) - f(t)$ are polynomials in t, where $f(t) = \frac{\prod_{i=1}^n \left(1-(1-t)^{c_i}\right)}{(1-t)^D}$.*

Thus, the difference between $\mathcal{H}(I, t)$ and $\mathcal{H}(J, t)$ is also a polynomial. As a corollary to the above theorem, we have

Corollary C.22. *If $S_1, S_2, \dots, S_n$ are transversal, then $h_I(i) = H_I(i) = h_J(i) = H_J(i)$ for all $i \geq n$. That is, the Hilbert polynomials of both the vanishing ideal I and the product ideal J are the same, and the values of their Hilbert functions agree with the polynomial with $i \geq n$.*

One of the consequences of this corollary is that for transversal subspace arrangements, we must have $I_i = J_i$ for all $i \geq n$. This is a result that we have mentioned earlier in Section C.2.

Example C.23 (Hilbert Series of Three Lines in $\mathbb{R}^3$). For example, suppose that $Z_{\mathcal{A}}$ is the union of three distinct lines (through the origin) in $\mathbb{R}^3$. Regardless whether the three lines are coplanar or not, they are transversal. We have

$$\mathcal{H}(J(Z_{\mathcal{A}}), t) = \frac{7t^3 - 9t^4 + 3t^5}{(1-t)^3} = 7t^3 + 12t^4 + 18t^5 + \cdots .$$

However, one has

$$\mathcal{H}(I(Z_{\mathcal{A}}), t) = \frac{t + t^3 - t^4}{(1-t)^3} = t + 3t^2 + 7t^3 + 12t^4 + 18t^5 + \cdots$$

if the lines are coplanar, and

$$\mathcal{H}(I(Z_{\mathcal{A}}), t) = \frac{3t^2 - 2t^3}{(1-t)^3} = 3t^2 + 7t^3 + 12t^4 + 18t^5 + \cdots$$

if the three lines are not coplanar. Notice that the coefficients of these Hilbert series become the same starting from the term t^3. ■

Then, using the recursive formula (C.18) of the Hilbert series $\mathcal{H}(J, t)$, we can derive a closed-form formula for the values of the Hilbert function $h_I(i)$ with $i \geq n$:

Corollary C.24 (A Formula for the Hilbert Function)**.** *If $S_1, S_2, \dots, S_n$ are transversal, then*

$$h_I(i) = h_J(i) = \sum_U (-1)^{|U|} \binom{D + i - 1 - c_U}{D - 1 - c_U}, \quad i \geq n, \tag{C.19}$$

where $c_U = \sum_{m \in U} c_m$ and the sum is over all index subsets U of $\underline{n}$ for which $c_U < D$.

Example C.25 (Three Subspaces in $\mathbb{R}^4$). Suppose that $Z_{\mathcal{A}} = S_1 \cup S_2 \cup S_3$ is a transversal arrangement in $\mathbb{R}^4$. Let d_1, d_2, d_3 (respectively c_1, c_2, c_3) be the dimensions (resp.

codimensions) of S_1, S_2, S_3. We make a table of $h_I(n)$ for $n = 3, 4, 5$.

c_1, c_2, c_3	d_1, d_2, d_3	$h_I(3)$	$h_I(4)$	$h_I(5)$
$1, 1, 1$	$3, 3, 3$	1	4	10
$1, 1, 2$	$3, 3, 2$	2	7	16
$1, 1, 3$	$3, 3, 1$	3	9	19
$1, 2, 2$	$3, 3, 2$	4	12	25
$1, 2, 3$	$3, 2, 1$	6	15	29
$1, 3, 3$	$3, 1, 1$	8	18	33
$2, 2, 2$	$2, 2, 2$	8	20	38
$2, 2, 3$	$2, 2, 1$	11	24	43
$2, 3, 3$	$2, 1, 1$	14	28	48
$3, 3, 3$	$1, 1, 1$	17	32	53

Note that the codimensions c_1, c_2, c_3 are almost determined by $h_I(3)$. They are uniquely determined by $h_I(3)$ and $h_I(4)$. ■

Corollary below is a general result that explains why the codimensions of the subspaces c_1, c_2, c_3 can be uniquely determined by $h_I(3), h_I(4), h_I(5)$ in the above example. The corollary also reveals a strong theoretical connection between the Hilbert function and the GPCA problem.

Corollary C.26 (Subspace Dimensions from the Hilbert Function)**.** *Consider a transversal arrangements of n subspaces. The co-dimensions $c_1, c_2, \ldots, c_n$ are uniquely determined by the values of the Hilbert function $h_I(i)$ for $i = n, n+1, \ldots, n+D-1$.*

As we have alluded to earlier, in the context of GPCA, these values of the Hilbert function are closely related to the ranks of the embedded data matrix $\boldsymbol{V}_i$ for $i = n, n+1, \ldots, n+D-1$. Thus, knowing these ranks, in principle, we should be able to uniquely determine the (co)dimensions of all the individual subspaces. These results suggest that knowing the values of the Hilbert function, one can potentially develop better algorithms for determining the correct subspace arrangement from a given set of data.

C.4 Bibliographic Notes

Subspace arrangements constitute of a very special but important class of algebraic sets that have been studied in mathematics for centuries [?, **?**, Orlik, 1989]. The importance as well as the difficulty of studying subspace arrangements can hardly be exaggerated. Different aspects of their properties have been and are still being investigated and exploited in many mathematical fields, including algebraic geometry & topology, combinatorics and complexity theory, and graph and lattice theory, etc. See [?] for a general review. Although the results about subspace arrangements are extremely rich and deep, only a few special classes of subspace arrangements have been fully characterized. Nevertheless, thanks to the work of [Derksen, 2005], the Hilbert function, Hilbert polynomial, and

Hilbert series of the vanishing ideal (and the product ideal) of transversal subspace arrangements have been well understood recently. This appendix gives a brief summary of these theoretical developments. These results have provided a sound theoretical foundation for many of the methods developed in this book for GPCA.

References

[Akaike, 1977] Akaike, H. (1977). A new look at the statistical model selection. *IEEE Transactions on Automatic Control*, 16(6):716–723.

[Barnett and Lewis, 1983] Barnett, V. and Lewis, T. (1983). *Outliers in Statistical Data*. John Wiley & Sons, second edition.

[Belhumeur et al., 1997] Belhumeur, P., Hespanda, J., and Kriegeman, D. (1997). Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):711–720.

[Beltrami, 1873] Beltrami, E. (1873). Sulle funzioni bilineari. *Giornale di Mathematiche di Battaglini*, 11:98–106.

[Bickel, 1976] Bickel, P. J. (1976). Another look at robustness: A review of reviews and some new developments. *Scand. J. Statist.*, 3(28):145–168.

[Bickel and Doksum, 2000] Bickel, P. J. and Doksum, K. A. (2000). *Mathematical Statistics: Basic Ideas and Selected Topics*. Prentice Hall, second edition.

[Bochnak et al., 1998] Bochnak, J., Coste, M., and Roy, M. F. (1998). *Real Algebraic Geometry*. Springer.

[Boult and Brown, 1991] Boult, T. and Brown, L. (1991). Factorization-based segmentation of motions. In *IEEE Workshop on Motion Understanding*, pages 179–186.

[Broomhead and Kirby, 2000] Broomhead, D. S. and Kirby, M. (2000). A new approach to dimensionality reduction theory and algorithms. *SIAM Journal of Applied Mathematics*, 60(6):2114–2142.

[Campbell, 1978] Campbell, N. (1978). The influence function as an aid in outlier detection in discriminant analysis. *Applied Statistics*, 27(3):251–258.

[Campbell, 1980] Campbell, R. J. (1980). Robust procedures in multivariate analysis i: Robust covariance analysis. *Applied Statistics*, 29:231–237.

[Chen et al., 2003] Chen, J.-Q., Pappas, T. N., Mojsilovic, A., and Rogowitz, B. E. (2003). Image segmentation by spatially adaptive color and texture features. In *IEEE Int. Conf. on Image Processing*.

[Chen et al., 1998] Chen, S., Donoho, D., and Saunders, M. (1998). Atomic decomposition by basis pursuit. *SIAM Journal of Scientific Computing*, 20(1):33–61.

[Coifman and Wickerhauser, 1992] Coifman, R. and Wickerhauser, M. (1992). Entropy-based algorithms for best bases selection. *IEEE Transactions on Information Theory*, 38(2):713–718.

[Collins et al., 2001] Collins, M., Dasgupta, S., and Schapire, R. (2001). A generalization of principal component analysis to the exponential family. In *Neural Information Processing Systems*, volume 14.

[Costeira and Kanade, 1998] Costeira, J. and Kanade, T. (1998). A multibody factorization method for independently moving objects. *Int. Journal of Computer Vision*, 29(3).

[Cover and Thomas, 1991] Cover, T. and Thomas, J. (1991). *Information Theory*. John Wiley & Sons, Inc.

[Critchley, 1985] Critchley, F. (1985). Influence in principal components analysis. *Biometrika*, 72(3):627–636.

[Delsarte et al., 1992] Delsarte, P., Macq, B., and Slock, D. (1992). Signal-adapted multiresolution transform for image coding. *IEEE Transactions on Information Theory*, 38:897–903.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

[Derksen, 2005] Derksen, H. (2005). Hilbert series of subspace arrangements (preprint).

[DeVore, 1998] DeVore, R. (1998). Nonlinear approximation. *Acta Numer.*, 7:51–150.

[DeVore et al., 1992] DeVore, R., Jawerth, B., and Lucier, B. (1992). Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2):719–746.

[Ding et al., 2004] Ding, C., Zha, H., He, X., Husbands, P., and Simon, H. D. (2004). Link analysis: Hubs and authoraties on the world wide web. *SIAM Review*, 46(2):256–268.

[Do and Vetterli, 2002] Do, M. N. and Vetterli, M. (2002). Contourlets: A directional multiresolution image representation. In *IEEE Int. Conf. on Image Processing*.

[Donoho, 1995] Donoho, D. (1995). Cart and best-ortho-basis: A connection. *Manuscript*.

[Donoho, 1998] Donoho, D. (1998). Sparse components analysis and optimal atomic decomposition. *Technical Report, Department of Statistics, Stanford University*.

[Donoho and Elad, 2002] Donoho, D. and Elad, M. (2002). Optimally sparse representation in general (non-orthogonal) dictionaries via L^1 minimization. *Manuscript*.

[Donoho and Elad, 2003] Donoho, D. and Elad, M. (2003). Optimally sparse representation in general (nonorthogonal dictionaries via L^1 minimization. *Proceedings of National Academy of Sciences*, 100(5):2197–2202.

[Donoho, 1999] Donoho, D. L. (1999). Wedgelets: nearly-minimax estimation of edges. *Ann. Statist.*, 27:859–897.

[Donoho et al., 1998] Donoho, D. L., Vetterli, M., DeVore, R., and Daubechies, I. (1998). Data compression and harmonic analysis. *IEEE Transactions on Information Theory*, 44(6):2435–2476.

[Eckart and Young, 1936] Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218.

[Effros and Chou, 1995] Effros, M. and Chou, P. (1995). Weighted universal transform coding: Universal image compression with the Karhunen-Loéve transform. In *IEEE Int. Conf. on Image Processing*, volume 2, pages 61–64.

[Eisenbud, 1996] Eisenbud, D. (1996). *Commutative Algebra: with a view towards algebraic geometry*. GTM. Springer.

[Elad and Bruckstein, 2001] Elad, M. and Bruckstein, A. (2001). On sparse signal representations. In *IEEE Int. Conf. on Image Processing*.

[Elad and Bruckstein, 2002] Elad, M. and Bruckstein, A. (2002). A generalized uncertainty principle and sparse representation in pairs of bases. *IEEE Transactions on Information Theory*, 48(9):2558–2567.

[Ferguson, 1961] Ferguson, T. (1961). On the rejection of outliers. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*.

[Feuer and Nemirovski, 2003] Feuer, A. and Nemirovski, A. (2003). On sparse representation in pairs of bases. *IEEE Transactions on Information Theory*, 49(6):1579–1581.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). RANSAC random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 26:381–395.

[Fisher, 1995] Fisher, Y. (1995). *Fractal Image Compression: Theory and Application*. Springer-Verlag Telos.

[Forgy, 1965] Forgy, E. (1965). Cluster analysis of multivariate data: efficiency vs. interpretability of classifications (abstract). *Biometrics*, 21:768–769.

[Gabriel, 1978] Gabriel, K. R. (1978). Least squares approximation of matrices by additive and multiplicative models. *J. R. Statist. Soc. B*, 40:186–196.

[Geman and McClure, 1987] Geman, S. and McClure, D. (1987). Statistical methods for tomographic image reconstruction. In *Proceedings of the 46th Session of the ISI, Bulletin of the ISI*, volume 52, pages 5–21.

[Gersho and Gray, 1992] Gersho, A. and Gray, R. M. (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

[Gnanadesikan and Kettenring, 1972] Gnanadesikan, R. and Kettenring, J. (1972). Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics*, 28(1):81–124.

[Guo et al., 2003] Guo, C., Zhu, S., and Wu, Y. (2003). A mathematical theory of primal sketch and sketchability. In *IEEE Int. Conf. on Computer Vision*.

[Hampel et al., 1986] Hampel, F., Ronchetti, E., Rousseeuw, P., and Stahel, W. (1986). *Robust statistics: the approach based on influence functions*. John Wiley & Sons.

[Hampel, 1974] Hampel, F. R. (1974). The influence curve and its role in robust estiamtion. *J. Amer. Statist. Assn.*, 69:383–393.

[Hansen and Yu, 2001] Hansen, M. and Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of American Statistical Association*, 96:746–774.

[Harris, 1992] Harris, J. (1992). *Algebraic Geometry: A First Course*. Springer-Verlag.

[Hastie, 1984] Hastie, T. (1984). Principal curves and surfaces. *Technical Report, Stanford University*.

[Hastie and Stuetzle, 1989] Hastie, T. and Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, 84(406):502–516.

[Hirsch, 1976] Hirsch, M. (1976). *Differential Topology*. Springer.

[Ho et al., 2003] Ho, J., Yang, M.-H., Lim, J., Lee, K.-C., and Kriegman, D. (2003). Clustering apperances of objects under varying illumination conditions. In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 11–18.

[Hotelling, 1933] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441.

[Householder and Young, 1938] Householder, A. S. and Young, G. (1938). Matrix approximation and latent roots. *America Math. Mon.*, 45:165–171.

[Huang et al., 2004] Huang, K., Ma, Y., and Vidal, R. (2004). Minimum effective dimension for mixtures of subspaces: A robust GPCA algorithm and its applications. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 631–638.

[Huber, 1981] Huber, P. (1981). *Robust Statistics*. John Wiley & Sons, New York.

[Hubert et al., 2000] Hubert, L., Meulman, J., and Heiser, W. (2000). Two purposes for matrix factorization: A historical appraisal. *SIAM Review*, 42(1):68–82.

[Jancey, 1966] Jancey, R. (1966). Multidimensional group analysis. *Austral. J. Botany*, 14:127–130.

[Jolliffe, 1986] Jolliffe, I. (1986). *Principal Component Analysis*. Springer-Verlag, New York.

[Jolliffe, 2002] Jolliffe, I. (2002). *Principal Component Analysis*. Springer-Verlag, 2nd edition.

[Jordan, 1874] Jordan, M. (1874). Mémoire sur les formes bilinéaires. *Journal de Mathématiques Pures et Appliqués*, 19:35–54.

[Kanatani, 2001] Kanatani, K. (2001). Motion segmentation by subspace separation and model selection. In *IEEE Int. Conf. on Computer Vision*, volume 2, pages 586–591.

[Kanatani, 2002] Kanatani, K. (2002). Evaluation and selection of models for motion segmentation. In *Asian Conf. on Computer Vision*, pages 7–12.

[Kanatani, 2003] Kanatani, K. (2003). How are statistical methods for geometric inference justified? In *Workshop on Statistical and Computational Theories of Vision, IEEE International Conference on Computer Vision*.

[Kleinberg, 1999] Kleinberg, J. M. (1999). Authorative sources in a hyberlinked environment. *J. ACM*, 48:604–632.

[Lang, 1993] Lang, S. (1993). *Algebra*. Addison-Wesley Publishing Company, 3rd edition.

[Leonardis et al., 2002] Leonardis, A., Bischof, H., and Maver, J. (2002). Multiple eigenspaces. *Pattern Recognition*, 35(11):2613–2627.

[LePennec and Mallat, 2005] LePennec, E. and Mallat, S. (2005). Sparse geometric image representation with bandelets. *IEEE Trans. on Image Processing*, 14(4):423–438.

[Lloyd, 1957] Lloyd, S. (1957). Least squares quantization in PCM. *Technical Report, Bell Laboratories, Published in 1982 in IEEE Trans. Inf. Theory 28: 128-137*.

[Ma and Vidal, 2005] Ma, Y. and Vidal, R. (2005). Identification of deterministic switched ARX systems via identification of algebraic varieties. In *Hybrid Systems: Computation and Control*, pages 449–465. Springer Verlag.

[Ma et al., 2008] Ma, Y., Yang, A., Derksen, H., and Fossum, R. (2008). Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM Review*.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

[Mallat, 1999] Mallat, S. (1999). *A Wavelet Tour of Signal Processing*. Academic Press, 2nd edition.

[Mallows, 1973] Mallows, C. (1973). Some comments on C_p. *Technometrics*, 15:661–675.

[Maronna, 1976] Maronna, R. A. (1976). Robust M-estimators of multivariate location and scatter. *Ann. Statist.*, 4:51–67.

[McLanchlan and Krishnan, 1997] McLanchlan, G. J. and Krishnan, T. (1997). *The EM Algorithms and Extentions*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc.

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative types and their connection with the theory of integral equations. *Philosophical Transactions, Royal Society London*, A(209):415–446.

[Meyer, 2000] Meyer, F. (2000). Fast adaptive wavelet packet image compression. *IEEE Trans. on Image Processing*, 9(5):792–800.

[Meyer, 2002] Meyer, F. (2002). Image compression with adaptive local cosines. *IEEE Trans. on Image Processing*, 11(6):616–629.

[Muresan and Parks, 2003] Muresan, D. and Parks, T. (2003). Adaptive principal components and image denoising. In *IEEE Int. Conf. on Image Processing*.

[Neal and Hinton, 1998] Neal, R. and Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models, M. Jordan (ed.), Kluwer Academic Publishers, Boston*, pages 355–368.

[Olshausen and D.J.Field, 1996] Olshausen, B. and D.J.Field (1996). Wavelet-like receptive fields emerge from a network that learns sparse codes for natural images. *Nature*.

[Orlik, 1989] Orlik, P. (1989). *Introduction to Arrangements*, volume 72 of *conference board of the mathematical sciences regional conference series in math.* American Mathematics Society.

[Overschee and Moor, 1993] Overschee, P. V. and Moor, B. D. (1993). Subspace algorithms for the stochastic identification problem. *Automatica*, 29(3):649–660.

[Pavlovic et al., 1998] Pavlovic, V., Moulin, P., and Ramchandran, K. (1998). An integrated framework for adaptive subband image coding. *IEEE Transactions on Signal Processing*.

[Pearson, 1901] Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosphical Magazine and Journal of Science*, 2:559–572.

[Rabiee et al., 1996] Rabiee, H., Kashyap, R., and Safavian, S. (1996). Adaptive multiresolution image coding with matching and basis pursuits. In *IEEE Int. Conf. on Image Processing*.

[Ramchandran and Vetterli, 1993] Ramchandran, K. and Vetterli, M. (1993). Best wavelet packets bases in a rate-distortion sense. *IEEE Trans. on Image Processing*, 2:160–175.

[Ramchandran et al., 1996] Ramchandran, K., Vetterli, M., and Herley, C. (1996). Wavelets, subband coding, and best basis. *Proceedings of the IEEE*, 84(4):541–560.

[Rao et al., 2005] Rao, S., Yang, A. Y., Wagner, A., and Ma, Y. (2005). Segmentation of hybrid motions via hybrid quadratic surface analysis. In *IEEE Int. Conf. on Computer Vision*, pages 2–9.

[Rissanen, 1978] Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

[Rousseeuw, 1984] Rousseeuw, P. (1984). Least median of squares regression. *Journal of American Statistics Association*, 79:871–880.

[Schindler and Suter, 2005] Schindler, K. and Suter, D. (2005). Two-view multibody structure-and-motion with outliers. In *IEEE Conf. on Computer Vision and Pattern Recognition*.

[Scholkopf et al., 1998] Scholkopf, B., Smola, A., and Muller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.

[Shapiro, 1993] Shapiro, J. M. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3463.

[Shi and Malik, 1998] Shi, J. and Malik, J. (1998). Motion segmentation and tracking using normalized cuts. In *IEEE Int. Conf. on Computer Vision*, pages 1154–1160.

[Shizawa and Mase, 1991] Shizawa, M. and Mase, K. (1991). A unified computational theory for motion transparency and motion boundaries based on eigenenergy analysis. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 289–295.

[Sikora and Makai, 1995] Sikora, T. and Makai, B. (1995). Shape-adaptive DCT for generic coding of video. *IEEE Transactions on Circuits and Systems For Video Technology*, 5:59–62.

[Starck et al., 2003] Starck, J.-L., Elad, M., and Donoho, D. (2003). Image decomposition: Separation of texture from piecewise smooth content. In *SPIE*.

[Steward, 1999] Steward, C. V. (1999). Robust parameter estimation in computer vision. *SIAM Review*, 41(3):513–537.

[Stewart, 1999] Stewart, C. (1999). Robust parameter estimation in computer vision. *SIAM Review*, 41(3):513–537.

[Taubin, 1991] Taubin, G. (1991). Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138.

[Tipping and Bishop, 1999a] Tipping, M. and Bishop, C. (1999a). Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482.

[Tipping and Bishop, 1999b] Tipping, M. and Bishop, C. (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, 61(3):611–622.

[Torr and Davidson, 2003] Torr, P. and Davidson, C. (2003). IMPSAC: synthesis of importance sampling and random sample consensus. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(3):354–364.

[Torr et al., 2001] Torr, P., Szeliski, R., and Anandan, P. (2001). An integrated Bayesian approach to layer extraction from image sequences. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(3):297–303.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer, N.Y.

[Vasilescu and Terzopoulos, 2002] Vasilescu, M. and Terzopoulos, D. (2002). Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision*, pages 447–460.

[Vetterli and Kovacevic, 1995] Vetterli, M. and Kovacevic, J. (1995). *Wavelets and Subband Coding*. Prentice-Hall.

[Vidal and Hartley, 2004] Vidal, R. and Hartley, R. (2004). Motion segmentation with missing data by PowerFactorization and Generalized PCA. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 310–316.

[Vidal and Ma, 2004] Vidal, R. and Ma, Y. (2004). A unified algebraic approach to 2-D and 3-D motion segmentation. In *European Conference on Computer Vision*, pages 1–15.

[Vidal et al., 2004] Vidal, R., Ma, Y., and Piazzi, J. (2004). A new GPCA algorithm for clustering subspaces by fitting, differentiating and dividing polynomials. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 510–517.

[Vidal et al., 2003] Vidal, R., Ma, Y., and Sastry, S. (2003). Generalized Principal Component Analysis (GPCA). In *IEEE Conference on Computer Vision and Pattern Recognition*, volume I, pages 621–628.

[Wallace and Boulton, 1968] Wallace, C. and Boulton, D. (1968). An information measure for classification. *The Computer Journal*, 11:185–194.

[Wallace and Dowe, 1999] Wallace, C. and Dowe, D. (1999). Minimum message length and Kolmogrov complexity. *The Computer Journal*, 42(4):270–283.

[Wallace, 1991] Wallace, G. K. (1991). The JPEG still picture compression standard. *Communications of the ACM. Special issue on digital multimedia systems*, 34(4):30–44.

[Wilks, 1962] Wilks, S. S. (1962). *Mathematical Staistics*. John Wiley & Sons.

[Wu et al., 2001] Wu, Y., Zhang, Z., Huang, T., and Lin, J. (2001). Multibody grouping via orthogonal subspace decomposition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 252–257.

[Wu et al., 2000] Wu, Y. N., Zhu, S. C., and Liu, X. W. (2000). Equivalence of Julesz ensemble and FRAME models. *Int. Journal of Computer Vision*, 38(3):247–265.

[Zhu et al., 1998] Zhu, S. C., Wu, Y. N., and Mumford, D. (1998). FRAME: Filters, random field and maximum entropy: — towards a unified theory for texture modeling. *Int. Journal of Computer Vision*, 27(2):1–20.