HW 1: Advanced Topics in Machine Learning

Instructor: René Vidal, E-mail: rvidal@cis.jhu.edu

Problems 1-3 are to be done individually.
Problems 4-5 are to be done in groups of 2 students.

Due 3/3/10 in class

1. **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?

2. **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 $t, \tau \in \mathbb{R}$.

   (a) Show that $K(t,\tau)$ has a family of orthonormal eigen-functions $\{\phi_i(t)\}_{i=1}^{\infty}$ that are defined as

   $$\int K(t,\tau)\phi_i(\tau)\, d\tau = \lambda_i \phi_i(t), \quad i = 1, 2, \ldots. \tag{1}$$

   (Hint: First show that $K(t,\tau)$ is a positive definite function and then use Mercer's Theorem.)

   (b) Show that with respect to the eigen-functions, the original random process can be decomposed as

   $$x(t) = \sum_{i=1}^{\infty} x_i \phi_i(t), \tag{2}$$

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

3. Let $\{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^N$ be a set of points you believe live in a manifold of dimension d. Imagine you have applied PCA, KPCA with kernel k and LLE with K-NN to the data. Assume now you are given a new point $\boldsymbol{x} \in \mathbb{R}^D$ and you wish to find its corresponding point $\boldsymbol{y} \in \mathbb{R}^d$ according to each one of the three methods. How would you compute $\boldsymbol{y} \in \mathbb{R}^d$ without applying PCA, KPCA or LLE from scratch to the $N+1$ points? Under what conditions the solution you propose is equivalent to applying PCA, KPCA or LLE to the $N+1$ points?

4. **Face recognition using PCA, PowerFactorization and Robust PCA.** In this exercise you will use a small subset of the Yale B dataset [1], which contains photos of 10 individuals under various illumination conditions. For this exercise we will use only images from the first three individual under ten different illumination conditions. Divide all the images in two sets: *Training Set* (images from individuals 1 to 3 and images 1-5) and *Test Set* (images from individuals 1-3 and images 6-10). Notice also that there are 5 non-face images (accessible as Individual #4, poses from 1 to 5). We will refer to these as *Outlier Set*. Download the file *HW2-10-Dataset.zip*. This file contains the image database along with the MATLAB function `loadimage.m`. Decompress the file and type `help loadimage` at the MATLAB prompt to see how to use this function. The function operates as follows.

[1] `http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html`.

Function `img=loadimage(individual,condition)`

Parameters

`individual` Number of the individual.

`condition` Number of the image for that individual.

Returned values

`img` The pixel image loaded from the database.

Description

Read and resize an image from the dataset. The database (directory `images`) must be in the same directory as this file.

(a) Apply PCA to the *Training Set*. Plot the mean face and a few eigenfaces. Plot also the singular values of the data matrix. Use the BIC, AIC and G-AIC to determine the number of eigenfaces. Project the *Test Set* onto the face subspace given by PCA. Plot the projected faces. Classify these faces by using 1-nearest-neighbors, that is, label an image $\boldsymbol{x}$ as corresponding to individual i if its projected image $\boldsymbol{y}$ is closest to a projected image $\boldsymbol{y}_j$ associated with individual i. Report the percentage of correctly classified face images.

(b) Repeat the experiment in part (a), but this time apply PCA to the images in *Training Set* $\cup$ *Outlier Set*. Comment on the differences in the results relative to part (a). Implement the distance-based outlier rejection scheme (with distance to the subspace as the criterion) and the consensus-based outlier rejection scheme to deal with sample outliers. Report the percentage of correctly detected outliers and the percentage of correctly classified face images after the outliers have been removed.

(c) Repeat the experiment in part (a), but this time assume that images 1 and 6 of individual 1 are missing 1% of the pixels. You can use your PowerFactorization code from HW1 instead of PCA to compute the mean face and the eigenfaces. You can do the following to set the matrix of missing data $W \in \mathbb{R}^{\#pixels \times \#images}$

```
W1 = ones(size(image11));
W1(1:10:end,1:10:end) = 0;
W = ones(npixels,nimages);
W(:,1) = W1();
```

Notice that you will need to develop new code to project an image with missing entries $\boldsymbol{x}$ onto the face subspace. This will involve solving a linear system of the form $\boldsymbol{x} = U\boldsymbol{y}$, where U is the basis for the subspace (already computed by PowerFactorization) and $\boldsymbol{y}$ is the projected image.

(d) Repeat the experiment in part (c), but this time replace the missing entries by arbitrary values. Assuming that the location of these intrasample outliers is unknown, apply PCA as in part (a) and report the results. Now apply the Augmented Lagrange Multiplier (ALM) approach to Robust PCA. The code is available at `http://watt.csl.illinois.edu/~perceive/matrix-rank/sample_code.html`. How well can you recover the location of the intrasample outliers as well as the correct image intensities?

5. Implement the KPCA algorithm for an arbitrary kernel function `kernel.m`. The format of your function should be as follows.

Function `[y]=kpca(x,d,kernel,params)`

Parameters

`x` $D \times N$ matrix whose columns are the data points

`d` dimension of the projected dataset

`kernel` name of the MATLAB function that computes the kernel `k = kernel(x1,x2,params)`

`params` parameters needed by the kernel function, such as the degree in the polynomial kernel or the standard deviation in the Gaussian kernel

Returned values

`y` $d \times N$ matrix containing the projected coordinates

Description

Computes the kernel principal components of a set of points.

Also implement the functions `k = poly_kernel(x1,x2,n)` for the polynomial kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1^\top \boldsymbol{x}_2)^n$ and `k = gauss_kernel(x1,x2,sigma)` for the Gaussian kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2/\sigma^2)$, where $k \in \mathbb{R}^{N \times N}$ and $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^{D \times N}$. Try your code on the example given in class. The code generating the data can be found at `http://www.kernel-machines.org/code/kpca_toy.m`