Homework 2: Mathematics of Deep Learning (EN 580.745)

Instructor: René Vidal, Biomedical Engineering, Johns Hopkins University

Due Date: 11/21/2019, 11:59PM Eastern Time

Instructions.

- You can discuss the problems with your peers at a high level, but you must write your own solutions.
- You can consult relevant background material, but don't seek out solutions to the problems themselves. Cite the outside material use.
- You are free to use any of the well-known libraries (e.g. Tensorflow, PyTorch, etc.).
- Submit a zip file containing a single scanned PDF writeup along with your code and figures. Typed writeups are nice, but clearly handwritten solutions are also fine.

1. Balanced weight matrices and positive homogeneity. Consider the single hidden layer neural network training problem

$$\operatorname*{minimize}_{\boldsymbol{U},\boldsymbol{V}} \sum_{i=1}^{N} \mathcal{L}(y_i, \boldsymbol{U}\psi(\boldsymbol{V}^\top \boldsymbol{x}_i)), \tag{1}$$

where $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ is a fixed training dataset, $\mathcal{L}$ is an arbitrary convex and non-negative loss function (e.g. logistic loss), and ψ denotes the pointwise ReLU activation $\psi(z) = \max(z, 0)$.

(a) **(5 points)** Prove that if a global minimizer to (1) exists, then the set of minimizers is unbounded.

(b) **(5 points)** Now suppose we add the following regularization

$$\operatorname*{minimize}_{\boldsymbol{U},\boldsymbol{V}} \sum_{i=1}^{N} \mathcal{L}(y_i, \boldsymbol{U}\psi(\boldsymbol{V}^\top \boldsymbol{x}_i)) + \frac{\lambda}{2}(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) \tag{2}$$

Prove that any global minimizer to (2) must have balanced weight matrices. I.e. $\|\boldsymbol{U}\|_F = \|\boldsymbol{V}\|_F$. Show that as a consequence, the set of minimizers to (2) is bounded.

(c) **(5 points)** Finally, suppose we instead use a regularizer with *unbalanced* degrees of homogeneity

$$\operatorname*{minimize}_{\boldsymbol{U},\boldsymbol{V}} \sum_{i=1}^{N} \mathcal{L}(y_i, \boldsymbol{U}\psi(\boldsymbol{V}^\top \boldsymbol{x}_i)) + \frac{\lambda}{2}(\|\boldsymbol{U}\|_F + \|\boldsymbol{V}\|_F^2) \tag{3}$$

Prove that by contrast, any global minimizer to (3) will typically *not* have balanced weight norms. I.e. will not satisfy $\|\boldsymbol{U}\|_F = \|\boldsymbol{V}\|_F$. Specifically, let $(\boldsymbol{U}^*, \boldsymbol{V}^*)$, be a global minimizer and let $\|\boldsymbol{U}^*\|_F \|\boldsymbol{V}^*\|_F = c$. Assume $c > 0$. Show that $\boldsymbol{U}^*$, $\boldsymbol{V}^*$ will be balanced if and only if $c = 1/4$.

(d) **(5 points)** Give some intuition in 1-3 sentences why a bounded set of minimizers, or balanced weight matrices might be beneficial for optimization (take your pick).

2. Benefit of over-parameterization on optimization. In this exercise you will attempt to replicate an experiment from [1] illustrating the benefit of over-parameterization for optimization (Figure 1). To save effort and keep the code short, you should implement the experiment using PyTorch, Tensorflow, or some other deep learning framework. (I.e. it would be better not to implement everything from scratch in MATLAB.)

(a) **(5 points)** Implement a synthetic dataset where samples $(\boldsymbol{x}, \boldsymbol{y})$ are drawn from a single hidden layer ReLU network with planted weights. Specifically, generate random Gaussian weight matrices $\boldsymbol{U}_0 \in \mathbb{R}^{n \times r_0}$, $\boldsymbol{V}_0 \in \mathbb{R}^{D \times r_0}$ with $(\boldsymbol{U}_0)_{ij} \sim \mathcal{N}(0, 1/r_0)$, $(\boldsymbol{V}_0)_{ij} \sim \mathcal{N}(\boldsymbol{0}, 1/D)$. Generate samples using these fixed "planted" weights as follows

$$\mathbb{R}^D \ni \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \qquad \mathbb{R}^n \ni \boldsymbol{y} = \boldsymbol{U}_0 \psi(\boldsymbol{V}_0^\top \boldsymbol{x}), \tag{4}$$

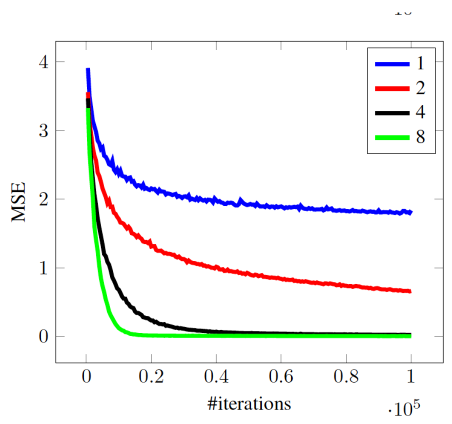

Figure 1: Effect of over-parameterization on single hidden layer network training from [1]. The data are synthetically generated from a network with fixed "planted" weights, and hidden layer size r_0. Mean squared error (MSE) is shown as a function of iteration for varying "over-parameterization levels". (I.e. a value of 2 means the trained network had a hidden layer of size $2r_0$.)

where ψ again denotes the ReLU activation. The specifics of the implementation will vary depending on which framework you use. But the key requirement is that the dataset has no fixed size. Every mini-batch drawn from the dataset should contain fresh samples. To show that your code works, implement a short test with $D = 150$, $r_0 = 60$, and $n = 10$ that generates 10 mini-batches of size 50 and prints the average $\|\boldsymbol{y}_i\|_2^2$ value for each.

(b) **(5 points)** Implement SGD to optimize the following least-squares objective for single hidden layer networks using the planted network dataset from part (a)

$$\underset{\boldsymbol{U},\boldsymbol{V}}{\text{minimize}} \quad f(\boldsymbol{U},\boldsymbol{V}) \triangleq \underset{(\boldsymbol{x},\boldsymbol{y})}{\mathbb{E}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{U}\psi(\boldsymbol{V}^\top \boldsymbol{x})\|_2^2. \tag{5}$$

Your algorithm should have the following parameters: an initial learning rate $\eta > 0$, and a fixed batch size $m > 0$. You should implement a learning rate schedule that decreases η by 0.5 every 50 epochs, where each "epoch" contains 10K samples. Log the average mini-batch objective value on each epoch.

(c) **(10 points)** Set $D = 150$, $n = 10$, $r_0 = 60$. Set the batch size $m = 10$ and hidden layer size $r = 2r_0 = 120$. Consider 4 choices for the initial learning rate η on a log scale, $\eta \in \{10^{-4}, 10^{-3}, \dots, 0.1\}$. Run the optimization for 20 epochs. Plot the average objective value as a function of epoch for each choice of η. Which value yields convergence to the lowest objective value?

(d) **(10 points)** Using the same problem setting and the optimal η from part (c), train networks of varying sizes $r/r_0 \in \{1, 2, 4, 8\}$ for 200 epochs. Again, plot the average objective value as a function of epoch, but now for each hidden layer size. How does your result compare to that of Livni & co-authors (Figure 1)?

(e) **(bonus)** If your result is different from Livni et al., experiment with the various problem settings (e.g. scaling of weights, η, batch size, choice of optimization algorithm) to see if you can replicate their observed pattern. Conversely, if you did replicate their result, see if you can get fast convergence to near zero error for all choices of r under a different problem setting.

References

[1] R. Livni, S. Shalev-Shwartz, and O. Shamir. On the computational efficiency of training neural networks. In *Advances in neural information processing systems*, pages 855–863, 2014. 1, 2