**EE 516: Mathematical Foundations of Machine Learning** **Winter 2023**

# Homework 7
# Due: March 5, 2023, 11:59PM PT

*Student Name:* *Instructor Name: John Lipor*

**Problem 1** (4 pts, 2 pts, 2 pts)

Define the unit $\ell_p$-norm ball as

$$B_p = \left\{ x \in \mathbb{R}^D : \|x\|_p \leq 1 \right\}.$$

It is often useful to visualize the unit ball for different norms, as we saw in the lecture on sparse regression. To do this, we instead visualize the unit sphere for each norm, which is

$$\mathbb{S}_p = \left\{ x \in \mathbb{R}^D : \|x\|_p = 1 \right\}.$$

One way to visualize this sphere is to generate a large number of random points and normalize these points so that they have $\|x\|_p = 1$.

(a) Complete the script `normalize` that takes in a matrix $X = [x_1\ x_2\ \dots\ x_N] \in \mathbb{R}^{D \times N}$ and normalizes it so that each *column* of $X$ has unit $\ell_p$-norm. **Do not** use any loops or built in `norm` or `norms` functions. **Turn in** your code and a brief explanation of how it works.

(b) Test your implementation by running `prob1`. **Turn in** the resulting plot.

(c) Intuitively, a set is convex if the line connecting any two points in a set is contained within that set. Based on this definition and your plot, for which norms (i.e., which values of $p$) is the unit ball convex?

**Problem 2** (5 pts, 4 pts, 4 pts)

In this problem, you will learn a linear classifier using empirical risk minimization (ERM) with the *logistic loss*. In this case, for a linear function parameterized by the weight vector $w$, ERM seeks to minimize

$$J(w) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, x_i^T w) + \frac{\lambda}{2}\|w\|_2^2, \tag{1}$$

where

$$L(y,t) = \log(1 + e^{-yt}). \tag{2}$$

The above loss is convex and differentiable. Using the above loss and assuming there are two class labels $y_i \in \{-1, 1\}$, solving (1) yields the optimal *logistic regression* classifier. In this problem, you will find this classifier using gradient descent and stochastic gradient descent.

(a) First, rewrite (1) as a summation

$$J(w) = \sum_{i=1}^{N} J_i(w) \tag{3}$$

for an appropriately defined $J_i(w)$. Using the loss function defined in (2), differentiate (3) for a single $i$ with respect to the vector $w \in \mathbb{R}^D$ to obtain the gradient.

(b) Solve (1) with logistic loss using gradient descent by computing the full gradient vector, which is the sum of the above "stochastic" gradients. Complete the `lrgd` function using the learning rate $\mu = \frac{100}{kk}$, where $kk$ is the iteration number. Test your code on digits 1 and 2 from the MNIST dataset *after reducing the dimensionality to 2 using PCA* by running `prob2`. **Turn in** your code. Plots will be formed in part (c).

(c) Solve (1) using *stochastic* gradient descent by taking 20 full passes through the data in random order. Complete `lrsgd` using the learning rate $\mu = \frac{100}{kk}$, where $kk$ is the iteration number. **Turn in** the following:

- A plot of the resulting cost as a function of the number of iterations
- The resulting plot of the two separators and the reduced-dimension data (code provided in `prob2`)
- A sentence or two describing how the two methods compare.

**Note:** For fair comparison, you should compute the cost for SGD after each of the 20 cycles, not after each individual update.

**Problem 3** (5 pts, 4 pts, 4 pts, 4 pts)

In this problem, you will learn a linear classifier using empirical risk minimization (ERM) with the *hinge loss*. You will again minimize (1), but this time take the loss defined by

$$L(y,t) = \max(0, 1 - yt). \tag{4}$$

The above loss is convex but not differentiable at 0, so we will make use of subgradients (see notes on sparse regression). Using the above loss and assuming there are two class labels $y_i \in \{-1, 1\}$, solving (1) yields the optimal *soft margin hyperplane*, which is the subspace of dimension $D-1$ that best separates data from two classes (where $D$ is the dimension of the features/examples/data points). In this problem, you will find this classifier using gradient descent and stochastic gradient descent.

(a) First, rewrite (1) as a summation

$$J(w) = \sum_{i=1}^{N} J_i(w) \tag{5}$$

for an appropriately defined $J_i(w)$. Using the loss function defined in (4), differentiate (5) for a single $i$ with respect to the vector $w \in \mathbb{R}^D$ to obtain a subgradient. Be careful at the point $yt = 1$.

(b) Solve (1) with hinge loss using gradient descent by computing the full gradient vector, which is the sum of the above "stochastic" gradients. Complete the `osmgd` function using the learning rate $\mu = \frac{100}{kk}$, where $kk$ is the iteration number. Test your code on digits 1 and 2 from the MNIST dataset *after reducing the dimensionality to 2 using PCA* by running `prob3`. **Turn in** your code. Plots will be formed in part (c). **Hint:** It may be easier to create a separate `subg` function to compute the subgradient, which you can use in both parts (b) and (c).

(c) Solve (1) using *stochastic* gradient descent by taking 20 full passes through the data in random order. Complete `osmsgd` using the learning rate $\mu = \frac{100}{kk}$, where $kk$ is the iteration number. **Turn in** the following:

- A plot of the resulting cost as a function of the number of iterations
- The resulting plot of the two separators (code provided in `prob3`)
- A sentence or two describing how the two methods compare.

**Note:** For fair comparison, you should compute the cost for SGD after each of the 20 cycles, not after each individual update.

(d) Finally, plot the reduced-dimension data with the classifiers obtained via (1) logistic loss + SGD, (2) hinge loss + SGD, and (3) LS classifer from Homework 6. **Turn in** the resulting plot and a table displaying the training and test errors in the form below.

| | Logistic Loss | Hinge Loss | Least Squares |
|---|---|---|---|
| Training Error (%) | | | |
| Test Error (%) | | | |

Table 1: Comparison of classification performance on MNIST digits 1 and 2 after reducing dimension to two using PCA.