CS 4964: Math for Data **Due: Tuesday 12.09 at noon**

Homework 5: Clustering and Classification

Instructions: Your answers are due at noon on the due date. You must turn in a pdf through canvas. I recommend using latex (`http://www.cs.utah.edu/~jeffp/teaching/latex/`) for producing the assignment answers. If the answers are too hard to read you will loose points, entire questions may be given a 0 (e.g. **sloppy pictures with your phone's camera are not ok, but very careful ones are**)

Please make sure your name appears at the top of the page.

You may discuss the concepts with your classmates, but write up the answers entirely on your own. **Be sure to show all the work involved in deriving your answers! If you just give a final answer without explanation, you may not receive credit for that question.**

We will use two datasets, here: `http://www.cs.utah.edu/~jeffp/teaching/cs4964/P.csv` and here: `http://www.cs.utah.edu/~jeffp/teaching/cs4964/Q.csv`
There are many ways to import data in python (see Canvas for a discussion). The `pandas` package seems to be the most general one.

1. [**40 points**] Download data sets P and Q. Both have 100 data points, each in 6 dimensions, can can be thought of as data matrices in $\mathbb{R}^{100\times 6}$. For each, run some algorithm to construct the k-means clustering of them. Diagnose how many clusters you think each data set should have by finding the solution for k equal to 1, 2, 3, ..., 10.

2. [**20 points**] Construct a data set X with 5 points in $\mathbb{R}^2$ and a set S of $k = 3$ sites so that Lloyds algorithm will have converged, but there is another set of $k = 3$, S' that $\textsf{cost}(X, S') < \textsf{cost}(X, S)$. Explain why S' is better than S, but that Lloyds algorithm will not move from S.

3. [**10 points**] Consider a "loss" function, called an *double-hinged loss function*

$$\ell_i(z) = \begin{cases} 0 & \text{if } z > 1 \\ 1 - z & \text{if } 0 \leq z \leq 1 \\ 1 & \text{if } z \leq 0. \end{cases}$$

   where the overall cost for a dataset (X, y), given a linear function $g(x) = \langle (1, x), \alpha\rangle$ is defined $\mathcal{L}(g, (X, y)) = \sum_{i=1}^{n} \ell_i(y_i \cdot g(x_i))$.

   (a) What problems might this have within a gradient descent algorithm to solve for the best α to minimize $\mathcal{L}$?

   (b) Explain if the problem would be better or worse using stochastic gradient descent?

4. [**30 points**] Consider the following Algorithm 1, called the *Double-Perceptron*. We will run this on an input set X consisting of points $X \in \mathbb{R}^{n\times d}$ and corresponding labels $y \in \{-1, +1\}$.

   For each of the following questions, the answer can be **faster**, **slower**, **the same**, or **not at all**. And should be accompanied with an explanation.

Algorithm 1 Double-Perceptron(X)

Initialize $w = y_i x_i$ for any $(x_i, y_i) \in (X, y)$
repeat
  For any (x_i, y_i) such that $y_i \langle x_i, w \rangle < 0$ (is mis-classified) : update $w \leftarrow w + 2 \cdot y_i x_i$
until (no mis-classified points **or** T steps)
return $w \leftarrow w / \|w\|$

(a) Compared with Algorithm 9.2.1 (Perceptron) in the notes, explain how this algorithm with converge.

(b) Next consider, transforming the input data set X (not the y labels) so that all coordinates are divided by 2. Now if we run *Double-Perceptron* how will the results compare to regular Perceptron (Algorithm 9.2.1) on the original data set X.

(c) Finally, consider taking the original data set (X, y) and multiplying all entries in y by -1, then running the original Perceptron algorithm. How will the convergence compare to running the same Perceptron algorithm, on the original data set.