Assignment 3

EECS 4404/5327, Fall '20

The assignment is due on Monday, November 9, by the end of the day.

In this assignment you will implement the perceptron algorithm. As in the previous assignment, you are asked to include a print out of your code in your submission.

Step 1 - load first dataset and analyze first dataset
Load the first dataset. The inputs are stored in `6pointsinputs.txt` and the binary outputs in `6pointsoutputs.txt`. Draw (or plot) the datapoints (there are only 6 points in this dataset 3 from each class). Prove that, independently of the order in which the perceptron algorithm gets to see these datapoints, the algorithm will terminate after one update (if you run perceptron to train a homogeneous linear classifier in 2 dimensional space).

Step 2 - implement perceptron
Implement the perceptron algorithm. Your method should take in a $n \times D$-matrix X of n D-dimensional input datapoints and an n-dimensional label vector t with entries in $\{-1, +1\}$. The algorithm should shuffle (use a random order) the datapoints, and add a column of ones to the data matrix (so that the algorithm will learn a general linear classifier in the D-dimensional inputs space by learning a homogeneous linear classifier in $D+1$ dimensional space).

Since we will use this algorithm also on data that is not linearly separable, add a stopping criterion that will terminate the algorithm after 100 passes over the dataset (if, by that point the perceptron didn't find a weight vector that correctly classifies all datapoints; of course, the algorithm should also terminate as soon as a weight vector with empirical binary loss 0 over the training data is found). Keep track of the number of updates the perceptron makes and ensure that, in the end, it outputs a normalized weight vector (output $\mathbf{w} \in \mathbb{R}^{D+1}$ should have norm 1).

Run your perceptron algorithms multiple times on the datapoints from step 1. Does it always terminate after one update? Does it always find a predictor that correctly classifies all 6 datapoints? Explain your findings.

Step 3 - implement 3 loss functions
Implement functions that compute the empirical binary loss, hinge-loss and logistic loss for a dataset of D-dimensional ($D+1$-dimensional when constant 1 coordinate is added) vectors and a $D+1$ dimensional weight vector $\mathbf{w}$.

Run your perceptron algorithm 20 times (ie each time on a different order of the datapoints) on the 6 points from step 1. For each run, evaluate the binary, the hinge, and the

logistic loss over the datapoints. Plot these three loss curves (the x-axis of your plot takes values $1, 2, \ldots, 20$ and the y axis corresponds to the losses).

What do you observe? Which losses vary and why? Interpret your findings.

Step 4 - compare loss functions

Now load the second dataset `fg_inputs.txt` and `fg_outputs.txt`. Run the your perceptron algorithm on this dataset 20 times and, as in the previous step, evaluate and plot binary loss, hinge loss and logistic loss of the resulting weight vectors. What do you observe now? How do the loss functions relate to each other? (You may want to do this several times and look at various versions of the resulting plot).

Now, in addition keep track of which run resulted in a weight vector of smallest binary loss, hinge loss and logistic loss. What do you observe here and what does this mean for the use of surrogate loss functions?

Step 5 - multiclass

Load the third dataset `irisnum.txt`. This is the iris-dataset from the UCI repository `https://archive.ics.uci.edu/ml/datasets/iris`. Here the last column actually corresponds to the labels. There are three class-labels 1–Iris Setosa, 2–Iris Versicolour, 3– Iris Virginica, and the dataset contains 50 points from each class (the first 50 points from class 1, points 51-100 from class 2 and points 101-150 from class 3). Remove the last column from the dataset to obtain a matrix of the input vectors.

We now will train three linear "one versus all" classifiers (rather than a multiclass-classifier). That is, you will run the perceptron algorithm on the iris-inputs, but with three different label vectors. To separate class 1 from classes 2 and 3, use a label vector with fifty $+1$ entries followed by a hundred -1 entries. Set up the other two label vectors accordingly (the fifty points from the single class get label 1 and the other one hundred points get label -1).

For each of the three tasks, run the perceptron multiple times, keep track of the three losses and include the corresponding plots in your writeup. What is the best binary loss you can achieve for each of the three tasks? What do these mean about the relationship between the three classes? Are they pairwise linearly separable from each other? Do you have a suggestion of how to turn the three one-versus-all predictors into multi-class predictor?

(4 + 4 + 4 + 4 + 4 marks)