EE 559 **Homework 4 (Week 6)** Posted: Fri., 2/18/2022
Jenkins Due: Fri., 2/25/2022, 11:59 PM PST

Please note: Coding for computer problems on homework assignments must be done in python, from now on.

1. In Lecture 8, we defined the criterion function for a 2-class Perceptron Learning problem (in augmented space) as:

$$J(\underline{w}) = -\sum_{n=1}^{N} [\![\underline{w}^T z_n \, \underline{x}_n \leq 0]\!] \, \underline{w}^T z_n \, \underline{x}_n$$

Prove that $J(\underline{w})$ is convex.

Hints:

(i) Write $J(\underline{w})$ replacing the indicator function with 2 possibilities, for example:

$$a[[a > 0]] = a \text{ if } a > 0, \;\; 0 \text{ if } a \leq 0,$$

in which $[[..]]$ denotes the indicator function.

(ii) Then write $J(\underline{w})$ using the $max\{\cdot\}$ function.

(b) Show that the your expression from (ii) is convex; you may find Discussion 3 (Week 4) notes helpful.

2. Code up a 2-class perceptron learning algorithm and classifier. For this problem, you may use only python built-in functions, numpy, matplotlib; you may use the PlotDecBoundaries.py function provided with Homework 1; and you may use pandas only for reading and/or writing csv files.

Tip: For randomly shuffling the order of data points, you may import 'random' library package. Below is an example to give you an idea of how the function works.

list1 = [1,2,3,4,5,6,7,8,9,10]

idx = random.sample(list1,4) → generate a list of 4 samples chosen at random from list1 without replacement.

idx = [5,1,9,3]

The elements in 'idx' can further be used to shuffle your data by indexing. **Be careful while shuffling. Make sure that you shuffle the samples along with the labels.**

Please observe the following:

(i) For the optimization, use basic sequential GD.

(ii) For the initial weight vector, use $\underline{w}(0) = a\underline{1}$, in which $a = 0.1$.

(iii) For the learning rate parameter, use $\eta(i) = 1$ $\forall_i$.

(iv) For the halting condition, use 2 conditions, such that it halts when either one is met:

i.1 When all the training data points are correctly classified. In this case, it also outputs the statement "data is linearly separable".

i.2 When 10,000 iterations have been performed. In this case, it should compare the criterion function value $J(\underline{w})$ for each of the previous 500 iterations, and choose for the final weight vector $\underline{\hat{w}}$ as the weight vector corresponding to the lowest $J(\underline{w})$ over those previous 500 iterations.

Hint1: You might find it useful to store the weight vectors inside a list on axis=0 (row wise).

Hint2: Save on space complexity by only storing the weight vectors and $J(\underline{w})$ values starting with iteration 9500.

(v) You will also need a function that classifies any given data point, using the optimal $\underline{\hat{w}}$ from the learning algorithm.

Tip: Although the datasets in (a) and (b) below have 2 features, it may be better to code for *D* features; the dataset in (c) has 13 features.

Please answer the questions, or proceed as instructed, below:

(a) For the synthetic1 datasets given with Homework 1 (Week 3), implement the following:

(i) Run the perceptron learning algorithm to find $\underline{\hat{w}}$.

Give the resulting $\underline{\hat{w}}$ vector; state whether the algorithm converged (i.1 reached) or halted without convergence (i.2 reached); and give the final criterion function value $J(\underline{\hat{w}})$.

(ii) Run the perceptron classifier on the training set and the test set. Give the classification error of each.

(iii) Plot in feature space the training data points, decision boundaries, and decision regions.

(b) Repeat part (a) except use the synthetic2 datasets given with Homework 1 (Week 3).

(c) Repeat part (a) except use the wine dataset, and use all 13 features. Use data only from classes 1 and 2 (both training data and test data), to make it a 2-class problem. No need to plot. Also answer the following:

(iii) Is the 2-class data linearly separable? Answer yes, no, or don't know. Briefly justify your answer.

3. Derive the gradient descent (GD) formula(s) for multiclass perceptron, starting from the criterion function as given in Lecture 9, Page 8.

Tip: there are *C* weight vectors to optimize. So the gradient of *J* would be the gradient with respect to all weight vectors $\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_C$. You can equivalently calculate the gradient for each weight vector $\underline{w}_j$ separately, and give a weight-update formula for each weight vector separately.

(a) Derive and state the batch GD weight-update formula(s).

(b) Derive the sequential GD weight-update formula(s), and state the entire sequential weight-update algorithm including the weight-update. Note that because there are different versions of the multiclass weight update(s), the correct answer may or may not be the same as the weight updates given in lecture.

(c) Is your weight-update of (b) the same as the multiclass weight update given in lecture? If the answer isn't obvious from an initial look, then justify your answer.

4. For 2-class perceptron *with margin* algorithm, using basic sequential GD, fixed increment, prove convergence for linearly separable training data by modifying the perceptron convergence proof covered in class. You may write out the proof, or you may take the proof from lecture and mark it up to show all changes as needed. If you mark up the existing proof, be sure to mark everything that needs changing (*e.g.*, if a change propagates through the proof, be sure to make all changes for a complete answer), and please make your mark-ups stand out for readability.