CMSC 422: Assignment 1 – Perceptrons – Spring 2020

The main purpose of this assignment is to help you gain some familiarity with perceptron learning. Problems 1 and 2 are "paper and pencil" problems (handwritten answers are fine); problem 3 involves doing some simple computational experiments using a version of the perceptron learning algorithm in Scikit-learn.

1. Measuring Performance

A perceptron with output values of {0, 1} is trained on 1,000 examples of a binary classification problem. Following training, the learning system classifies all 1000 of these training examples correctly. When subsequently applied to 100 different, randomly selected examples drawn independently based on the same probability distribution as the training data, the results are:

		Predictions	
		1	0
Target	1:	3	11
Classes	0:	2	84

In the context of this information, give the following for this trained perceptron:

a) name of the table displayed here
b) apparent error rate
c) best estimate for true error rate
d) 95% confidence interval for the estimated true error rate
e) sensitivity for class 1
f) specificity for class 1

2. Elementary Perceptron Calculation by Hand (paper & pencil problem; a calculator may be used)

Consider an elementary perceptron with three input nodes having activation levels a_1, a_2, and a_3, a bias node with fixed $a_0 = 1$, and a single response/output node r that uses a linear threshold/step activation function with outputs of 0 or 1.

a. Consider training this perceptron, using the perceptron learning algorithm discussed in class, to perform the following classification of input patterns:

Pattern	a_1	a_2	a_3	Target Class
A	0	0	0	0
B	1	1	1	1
C	0	0	1	0
D	1	1	0	1
E	0	1	0	0
F	1	0	1	1
G	1	0	0	0
H	0	1	1	1

Specifically, starting with weights $w_{r1} = -1$, $w_{r2} = -1$, $w_{r3} = -1$, a bias $w_{r0} = 0$, and using a learning rate of $\eta = 1$, suppose this perceptron is first shown patterns B and G in that order. Show the weights and bias value immediately after the perceptron sees each of these two patterns and adjusts its weights and bias according to the perceptron learning algorithm.

b. Sketch and describe the resultant decision surface in $\vec{a}$ state space after just the two patterns B and G have both been processed as in (a), sketching the weight vector $\mathbf{w_r}$, and clearly indicating the regions where 0 and 1 classifications occur.

c. Yes or no: Will this perceptron eventually learn to do the above classification correctly if it repeatedly cycles through all possible input patterns?

d. If you answered yes to (c), give the name of the theorem that guarantees your answer. If you answered no, then give the name of the property that justifies your answer. In either case, give a clear justification for why your answer here is correct, illustrating it with a drawing of the input state space.

e. If you answered yes to (c), use an analysis (i.e., not further perceptron learning) of the algebraic constraints on the weights and bias value implied by the full set of training data to derive by hand a specific set of weights $\vec{w}_r$ and bias value w_{r0} that enable the perceptron to correctly perform the classification. If you answered no to (c), then prove that your answer is correct by analyzing the algebraic constraints on the weights and bias value implied by the training data.

3. Behavior of A Perceptron Classifier as Noise Varies in the Dataset

Scikit-learn is a widely used, python-based, software environment for machine learning. To access the documentation for scikit-learn, go to its home page https://scikit-learn.org/stable/documentation where you can find its User Guide and instructions for downloading it (e.g., *pip* or *conda*). In this problem, you will use scikit-learn's *Perceptron()* to examine how the amount of noise impacts a perceptron's performance on a classic dataset, the "two moons" data.

Use scikit-learn as follows. Create a python script file *moons.py* that first generates a data set with 500 samples by calling *make_moons(),* where you use the default parameter values except for assigning an initial random seed that you use consistently in all runs (e.g., *random_state* = 13). The function *make_moons()* generates data in the form of two interleaved half circles in a 2D input space, and you can vary the amount of noise in the data by assigning the appropriate input parameter (*noise* = …). Once you create this data set, randomly partition it into stratified training (75% of data) and test (25%) data sets using *train_test_split()*. Your script file should then print to a file named *results.txt* your name, UID, the noise level and random seed used in generating the data, the exact number of training and test data examples produced by *train_test_split(),* and the mean and standard deviation values for the total data set. Also generate a plot of the full data set in the 2D input space. Once the data has been generated, partitioned and displayed, your script file should continue on to train scikit-learn's *Perceptron()* on the training data to classify examples correctly. Use a maximum of 100 epochs and a learning rate (*eta0*) of 0.1 as arguments to the perceptron. Output to the same *results.txt* file the perceptron's post-training weights and bias value (intercept), and its post-training accuracies on the training data and the test data.

a. Using your script file, do five independent runs of a perceptron learning the two moons data. For each run, generate 500 samples using the same random seed but based on a different value 0.01, 0.1, 0.2, 0.5, or 2.0 for the *noise* input parameter in each run. Based on the results as described above, create and turn in a five line table where the columns are (left to right): noise level, first learned weight, second learned weight, intercept, accuracy on the training data, and accuracy on the test data.

b. For which of the five noise levels used is the data linearly separable? Explain how you know for certain that your answer to this question is correct for each of the noise levels.

c. How did the perceptron's accuracy change as the noise level increased from near zero to 2.0? Explain why this occurred.

d. If one only had a much smaller amount of data, say ≤ 50 samples rather than the 500 examples used above, would one expect the difference between training data accuracy and test data accuracy to be higher, about the same, or lower than what was observed with the runs done in (a). Why?

- continues on next page -

Important: Save your results from Problem 3 – you will need these for comparisons in the next homework assignment.

What should I turn in?

The hardcopy and electronic submissions are due at different times. The hard copy portion is due at the start of class Thursday, February 13; the electronic submission is due earlier at 11:30 pm Wednesday, February 12.

Hardcopy: Your answers to the questions in problems 1 - 3.

Electronic submission: For problem 3, turn in a single zip file that includes your script file *moons.py*, your output file *results.txt* (both just for the case where no noise is present), and two plots of your data for the cases where there is no noise (*plotNone.pdf*) and where 2.0 is used for the noise parameter value (*plotMax.pdf*); these latter two plots can alternatively be submitted as png/jpg/gif/pdf files. Use the Computer Science Department project submission server at https://submit.cs.umd.edu to submit this zip file.