Part B Report

Name: Anthony Zhang

Assignment 6: Perceptron Classification and Training

CSE 415 Introduction to Artificial Intelligence, Autumn 2022, University of Washington

Please answer each question using text in Blue, so your answers stand out from the questions.
Note: If not otherwise specified, use the default parameters present in the code to answer the questions.

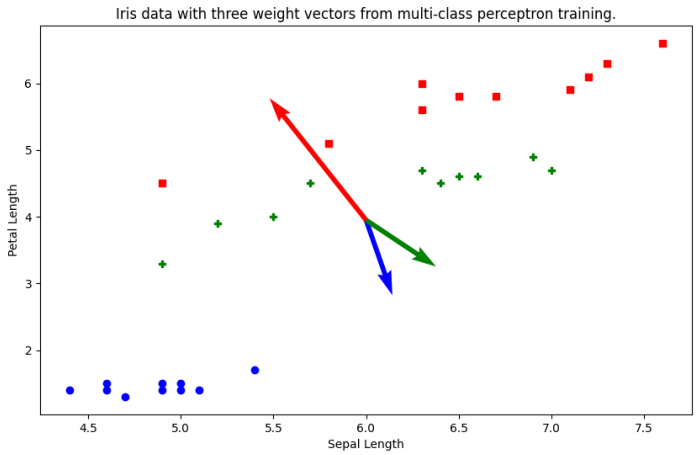
B1. How many epochs were required to train your perceptron on the 3-class Iris data having 4 features (the given training file, with 30 examples)? How many of the test data examples (out of 120) were mis-classified? Determine the percentage error rate and write that here.

It took 85 epochs to train my perceptron on the 3-class Iris data having 4 features

There are 14 errors/misclassification out of the 120 data examples.

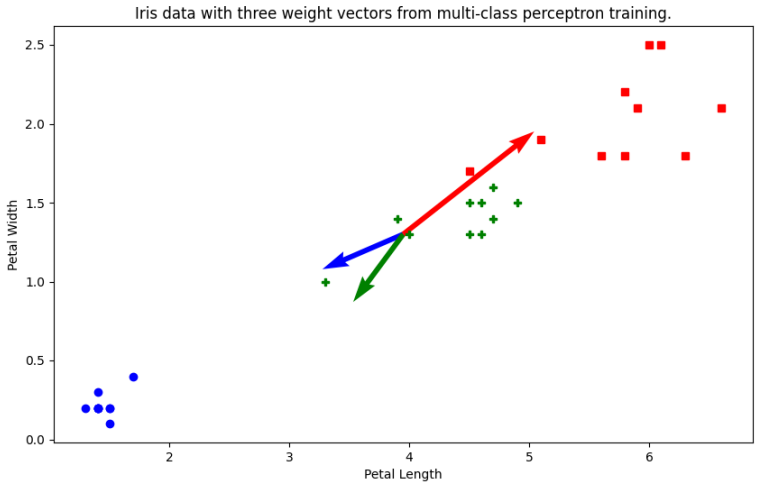
The percentage of error rate is 12/120 * 100 = 11.67%

B2. Capture the plot that is produced by the program showing the training data and the weight vectors when projected onto the 2-D subspace spanned by sepal length and petal length (which is the starter-code default in run_3_class_4_feature_iris_data.py. Paste it here, reduced to fit in the remaining space on this page.



B3. In the file run_3_class_4_feature_iris_data.py, now modify the code so you can see the data projected onto the subspace spanned by features 2 and 3 (petal length and petal width). Describe the

how the data seems to be distributed in this view. Describe how the weight vectors seem to be pointing. Finally, describe the relationship between the weight vectors and the distribution of the data.



The data is roughly distributed on a diagonal line between the top right corner and the bottom left corner of the graph. The red vector seems to be pointing on the top right corner while the green and blue vector are pointing at the bottom left corner. The weight vectors point toward the general area that the target group of data is distributed on the graph.

B4. In the file run_3_class_4_feature_iris_data.py, instead of using all zero weight, let

W = [[1, 1, 1, 1, 1], [-1, -1, -1, -1, -1], [0, 0, 0, 0, 0]]. Now, for learning rates starting from 1e-3 to 1e+3 (all powers of 10), investigate how many epochs it takes for the model to converge. (You may similarly investigate the model for other initializations of W if you wish). Also, find the number of errors on the test set for each ternary perceptron. What kinds of trends do you observe?

1e-3: Converge in 261 epochs, 11 errors

1e-2: Converge in 76 epochs, 12 errors

1e-1: Converge in 57 epochs, 17 errors

1e0: Converge in 50 epochs, 10 errors

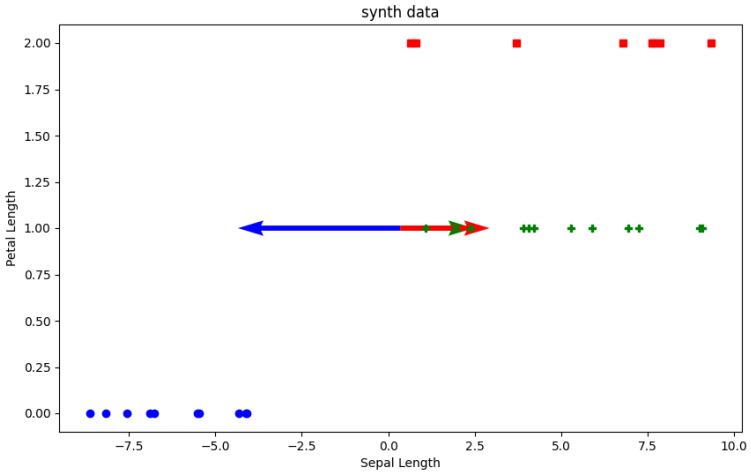
1e+1: Converge in 73 epochs, 5 errors

1e+2: Converge in 85 epochs, 19 errors

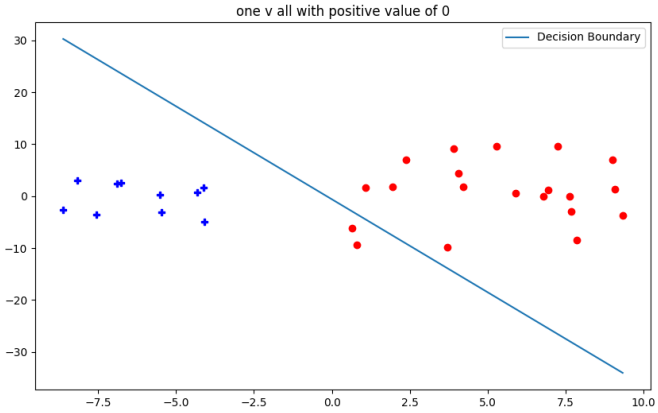
1e+3: Converge in 85 epochs, 14 errors

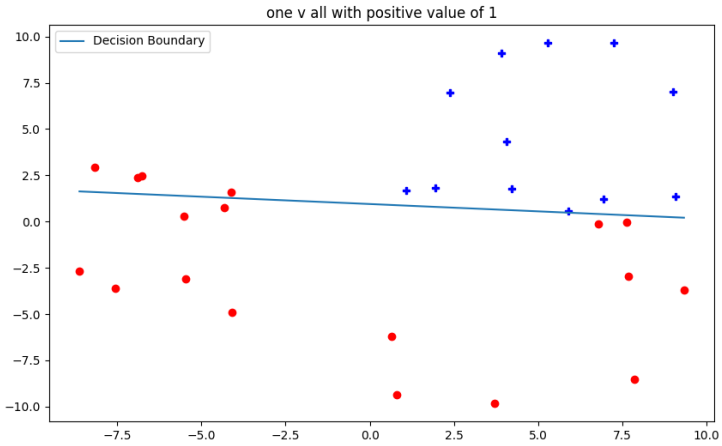
The closer the learning rate is to 1, the less epochs need to converge.

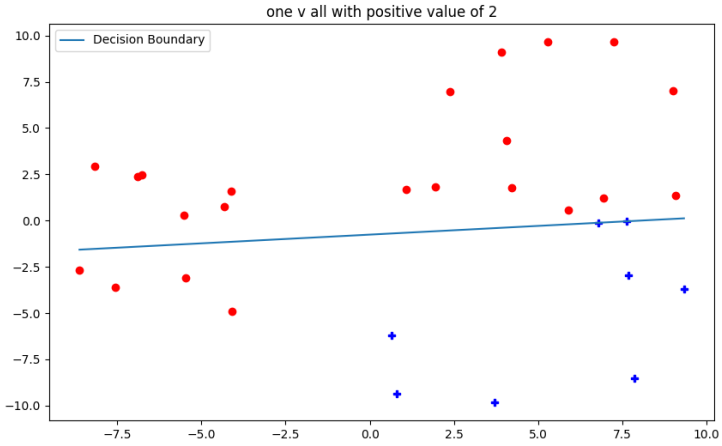
B5. Using the file run_synth_data_ternary.py, capture the plot of the ternary perceptron for the synthetic dataset and paste it here. (Let the maximum number of epochs be 50 and learning rate 0.5, and the weights be all zeros).



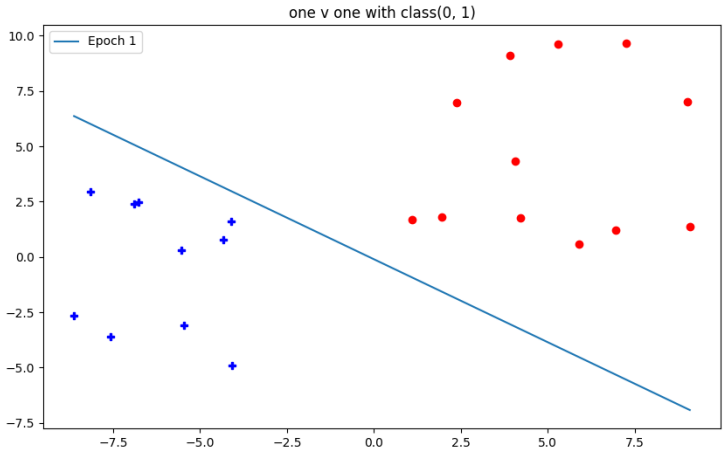
B6. Using the file run_synth_data_1_vs_all.py, capture the plots of all the One-Vs-All classifiers for the synthetic data and paste them here. (Let the maximum number of epochs be 50 and learning rate 0.5).

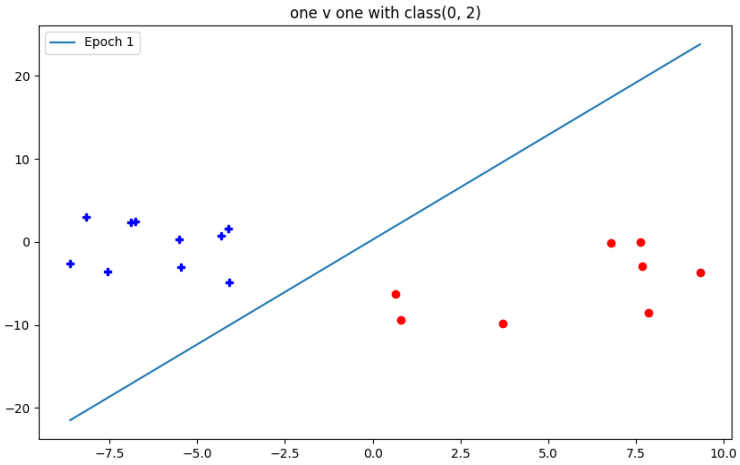


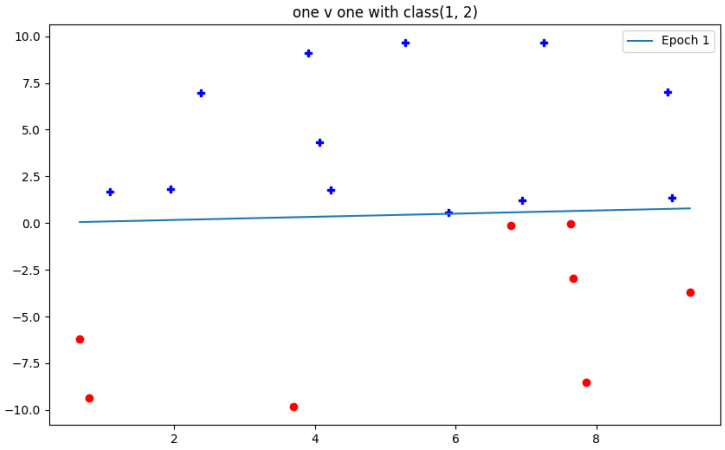




B7. Using the file run_synth_data_1_vs_1.py, capture the plots of all the One-Vs-One classifiers for the synthetic dataset and paste them here. (Let the max number of epochs be 50, and learning rate 0.5)







B8. Using the One-Vs-All classifier, classify the point [6.78, -0.12] as either in class 0, 1, or 2. Briefly explain how you got that class using the individual classifiers. Repeat the same process for One-Vs-One.

For 1-vs-all, the point [6.78, -0.12] in class 0 is classified as -1 because it Is in the red area

For 1-vs-1, the point [6.78, -0.12] in class (0,1) is classified as -1 because it is in the red are