Isai Mercado Oliveros
misaie
January 8, 2016

Perceptron Report

1. Try a couple different learning rates and discuss the effect of the learning rate, including how many epochs are completed before stopping.

Learning Rate	Epochs	Accuracy
0.00001	4	0.3970
0.0001	4	0.3970
0.001	15	0.9000
0.01	25	0.9544
0.1	30	0.9456

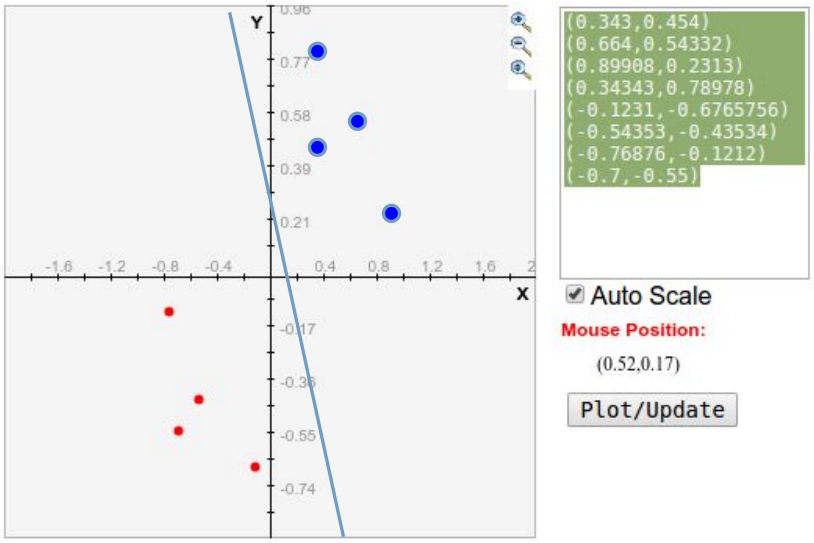
It seems like with an small learningRate, the learning is so small that my algorithm perceives it as if it did not have any improvements, so it stops. That explains the small amount of epochs and accuracy. On the other hand, once the learning rate is above 0.001 the algorithm starts to have significant changes in its predictions, so we see more epochs, and the accuracy improves from .90 to .95.

2. Describe your specific stopping criteria.

I have a no-improvement-counter that watches for no improvement from the current epoch and the last epoch's errors even if they are not consecutive, so when it counts 3 times, the algorithm stops.

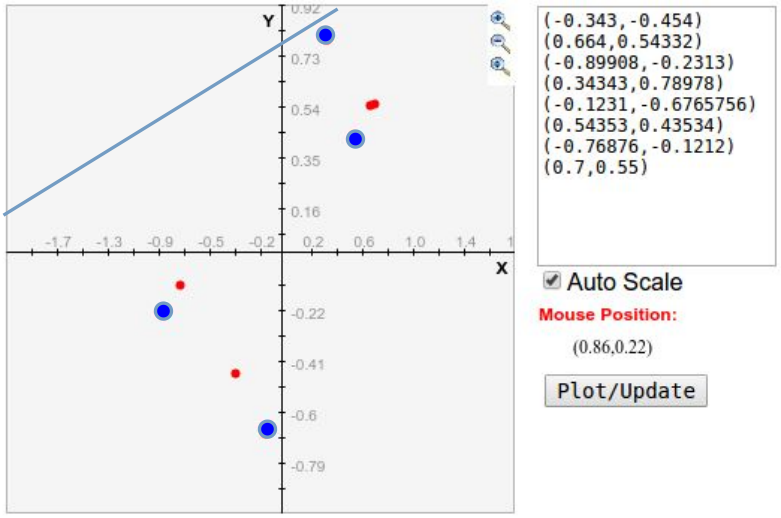
3. Graph the instances and decision line for the two arff files you created

Linear Separable
Equation: -0.1758X -0.0636Y +0.0342 = 0
Accuracy: 1.0



In this set, the points are linear separable. The equation of the line given by the perceptron separates them with no problem and the accuracy is 1.0.

Non Linear separable
Equation: 0.0220X -0.1212Y +0.1626 = 0
Accuracy 0.5



When the data set is not linear separable the perceptron's hyperplane keeps rotating, but it will not find a solution. Finally, my algorithm stopped when it detected that there was no change in outcome three times. The algorithm has an accuracy of 0.5 because the points under the line have a 50/50 chance of being red or blue.

5. Use the perceptron rule to learn this version of the voting task.
- Randomly split the data into 70% training and 30% test set.
- Try it five times with different random splits.
- For each split report the final training and test set accuracy and the # of epochs required.

Accuracy	Epochs
0.9130	17
0.9565	23
0.9478	16
0.9568	29
0.9217	7

In this table the accuracy varies on a small range from .90 to .96, but the epochs are all over the place from 7 to 29. That happens because we start with random weights, which might be close to the real solution, or very far and they need more epochs to be fixed

- Report the average of these values from the 5 trials.
Accuracy Average: 0.9392
Epoch Average: 18

- Explain what the model has learned and how the individual input features affect the result.

The model learned how to separate republicans from democrats. The model initially calculated an hyper-plane with random weights. When the model tries to predict the next row of features and gets the prediction wrong, it updates the weight of the features by the following formula: weight = weight + (targetLabel - predictedLabel) * learningRate * feature. If the prediction is correct, the part (targetLabel - predictedLabel) will cancel the change so the weight will continue to be the same. On the other hand, if the prediction is wrong, the expression (targetLabel - predictedLabel) will return a 1 or -1. This will give direction by pulling or pushing the hyper-plane's dimension. The learningRate is a constant that we set to control the amount of change of the weight. For instance, if the learningRate is small like 0.001 the weight will push or pull the hyper-plane's direction very little; on the contrary, if the learningRate is 3 it will change a lot. Finally, the equation includes the feature because we want to push or pull the dimension in relation to the size of the feature. Since one feature might have an small range such as from -1 to 1, and another feature might have a big range like 0 to 10,000,000, features need to be normalize. Otherwise, the big feature would pull the dimension a lot.

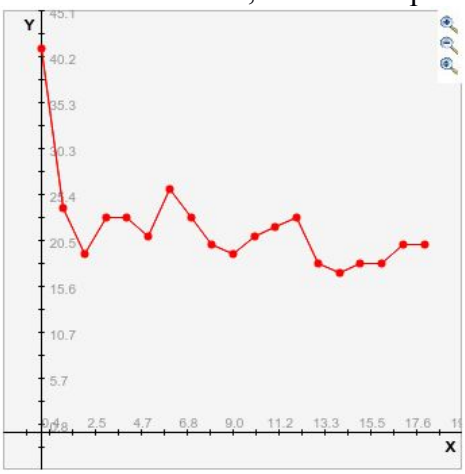
- Which specific features are most critical for the voting task, and which are least critical?

Feature	Weight	Is the feature critical?
handicapped-infants	-0.1216	NO
water-project-cost-sharing	-0.2924	NO
adoption-of-the-budget-resolution	-0.5965	NO
physician-fee-freeze	2.2107	YES
el-salvador-aid	0.4841	NO
religious-groups-in-schools	-0.0253	NO
anti-satellite-test-ban	-0.0622	NO
aid-to-nicaraguan-contras	1.5068	YES
mx-missile	-1.0892	YES
immigration	0.5191	NO
ynfuels-corporation-cutback	-1.2727	YES
education-spending	0.3378	NO
superfund-right-to-sue	0.2079	NO
crime	-0.1837	NO
duty-free-exports	-0.6333	NO
export-administration-act-south-africa	0.5917	NO

The most important features have a large distance from zero. It does not matter if it is on the negative or positive direction. The only important aspect is how much the weight pulls or pushes the hyperplane. If the weight is zero, it means that this feature does not push or pull the hyperplane into its dimension, so that dimension is not important, and therefore, the feature is not important.

- Do one graph of the average misclassification rate vs epochs for the training set.

Y Axis is Error, X Axis is Epochs



In this graph, we can see that the error rate jumps from high to low, but also it jumps from low to high. This happens because in the current epoch the perceptron predicted some labels correctly, but since the weights change and the hyperplane updated, in next iteration the perceptron might have some point misclassified even when they were classified correctly in the last iteration. However, the range of jumping gets smaller and smaller until the error converges to some number or the change is so insignificant, that the algorithm will stop.

6. Use the perceptron rule to learn the iris task. Create 1 perceptron for each pair of output classes, where the training set only contains examples from the 2 classes.

Data set	Accuracy
iris_setosa_versicolor	1.0
iris_versicolor_virginica	0.93
iris_virginica_setosa	1.0

I divided the iris data set in three subsets. The first subset compares setosa vs versicolor. The second subset compares versicolor vs virginica, and the third subset compares virginica vs setosa. I run my perceptron one data set at a time using a 2 cross validation because the data sets where small.