CS 5350/6350: Machine Learining Fall 2015

Homework 2

Handed out: Sep 17, 2015
Due date: Oct 1, 2015

1 Warmup: Boolean Functions

1.

Solution:

y = x_4
y = $x_2 \wedge x_3 \wedge x_4$
y = $x_1 \vee x_2 \vee x_3 \wedge x_4$

2.

Solution:

y = x_4 will make 0 errors
y = $x_2 \wedge x_3 \wedge x_4$ will make 2 errors
y = $x_1 \vee x_2 \vee x_3 \wedge x_4$ will make 0 errors

3.

Solution:

The data is linearly separable.

The linear threshold function that classifies the data is $y = 1$ if $x_4 > 0$

2 Mistake-bound learning

1.

Solution:

a. C_n contains 2^n functions.

b. As per the function defined there is only one input which will match z and hence before that happens all outputs will be zero. So when that input occurs the output will vary from the majority label and so all functions will be discarded except for one. Hence the halving algorithm will make only 1 mistake.

c. Yes. Halving Algorithm is a mistake bound algorithm for this Concept class.

2.

Solution: We have a concept class C with N elements. We have M perfect experts out of the N. As the pool is halved everytime a mistake is made. We will stop when $\mid C_T \mid \geq M$. We also know that $C_1 = N$.

Consider that Z is the total number of mistakes we make.

$$\mid C_T \mid \leq \frac{\mid C_1 \mid}{2^Z}$$

$$M \leq \frac{N}{2^Z}$$

$$2^Z \leq \frac{N}{M}$$

$$Z \leq \log \frac{N}{M}$$

Therefore the bound of the Halving algorithm is $O(\log \frac{N}{M})$

3 The Perceptron Algorithm and its Variants

Experiment 1

Solution:

I have taken a randomized vector of weights. I run once over an array of rates and find the one which makes the least mistake along with the randomized weight vector. As the vector is randomized the mistakes made may vary depending on the vector generated.

Output:
Weights = 0.067 0.069 0.067 0.067 0.07
Mistakes: 3

Experiment 2

Solution:

I took arrays of 10 rates and 10 margins. I ran a 10-fold cross validation to find the best rate for Simple Percetron and best rate-margin combo for Margin Perceptron. The accuracy may vary depending on the weight vector generated as it is randomized.

Output:
Updates made by Simple Perceptron: 53
Simple Perceptron Accuracy(On training data): 100
Simple Perceptron Accuracy(On test data): 100

Updates made by Margin Perceptron: 61
Margin Perceptron Accuracy (On training data): 99.9
Margin Perceptron Accuracy (On test data): 100

Experiment 3

Solution:

Using the same arrays as above, I ran a 10-fold cross validation for each train & test file to get the best rate and rate-margin combo for Simple Perceptron and Margin Perceptron.The accuracy on data0 gradually decreased as the number of dimensions increased. I have attached graphs below explaining comparison between Simple and Margin Perceptron as well as dimension vs updates or dimensions vs accuracy for each of the algorithm for both data sets separately.The randomized weight vector has the ability to change the accuracies by a margin.

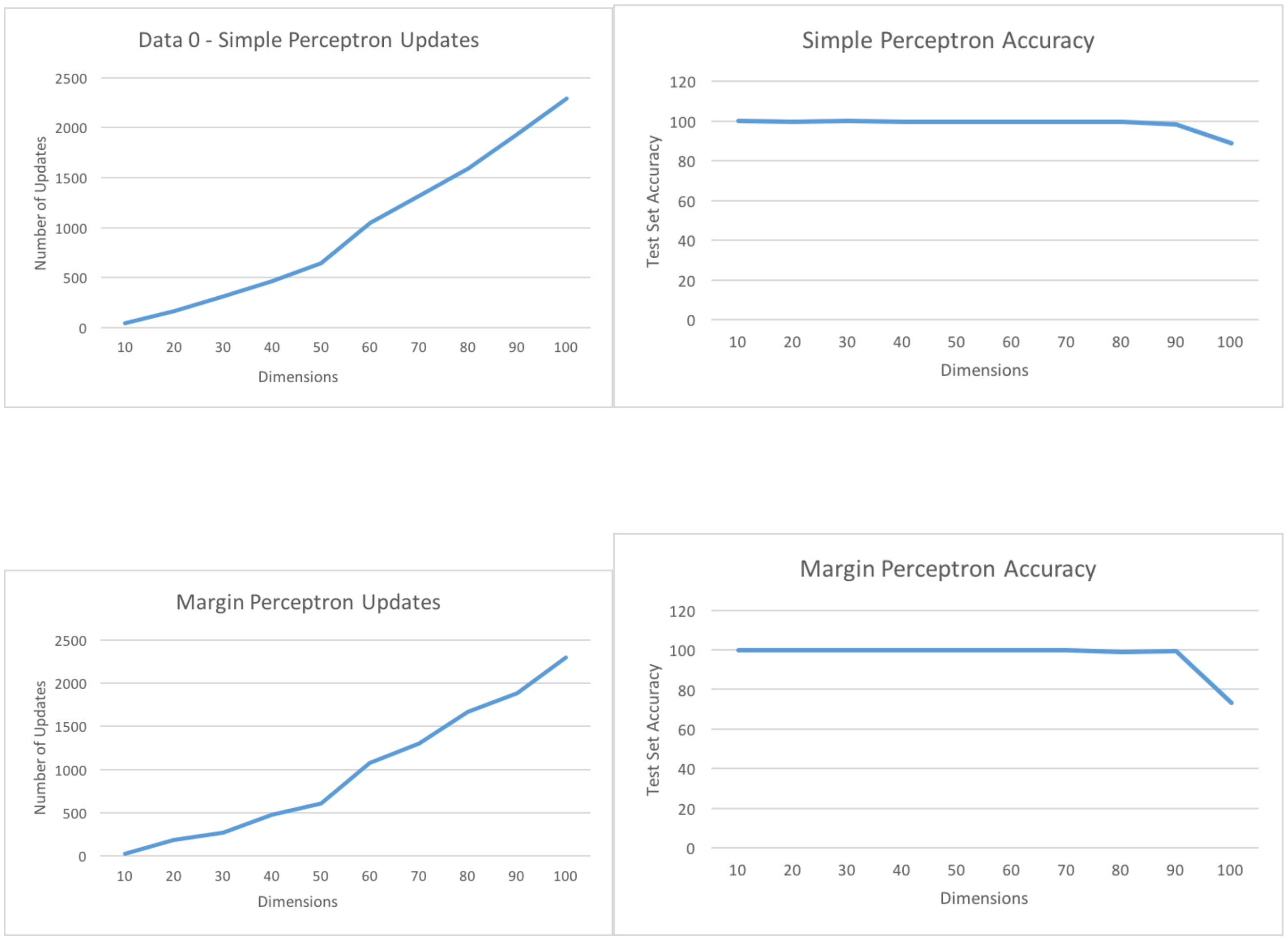

Figure 1: Data0- For 10 epochs

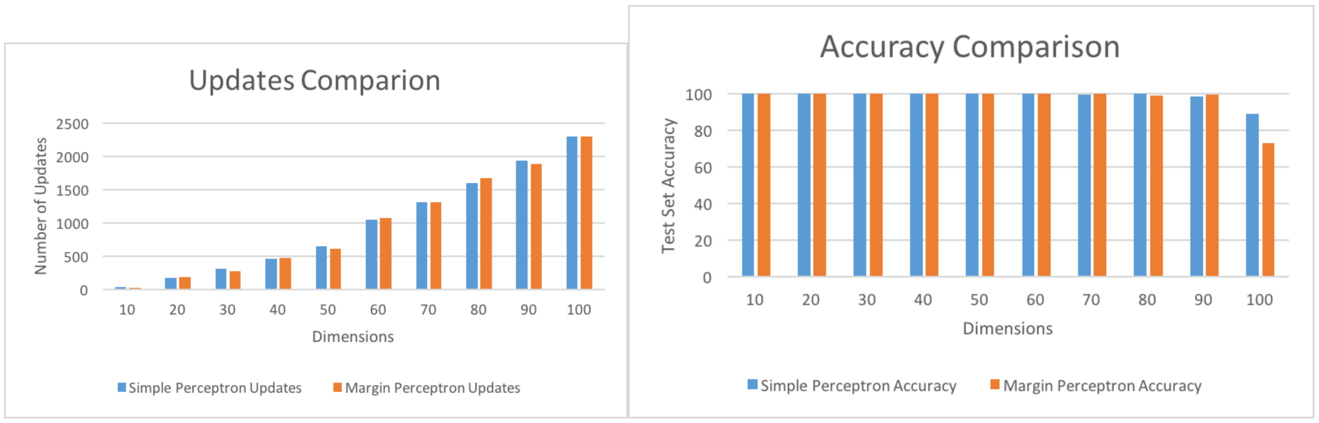

Figure 2: Data0 - Comparisons

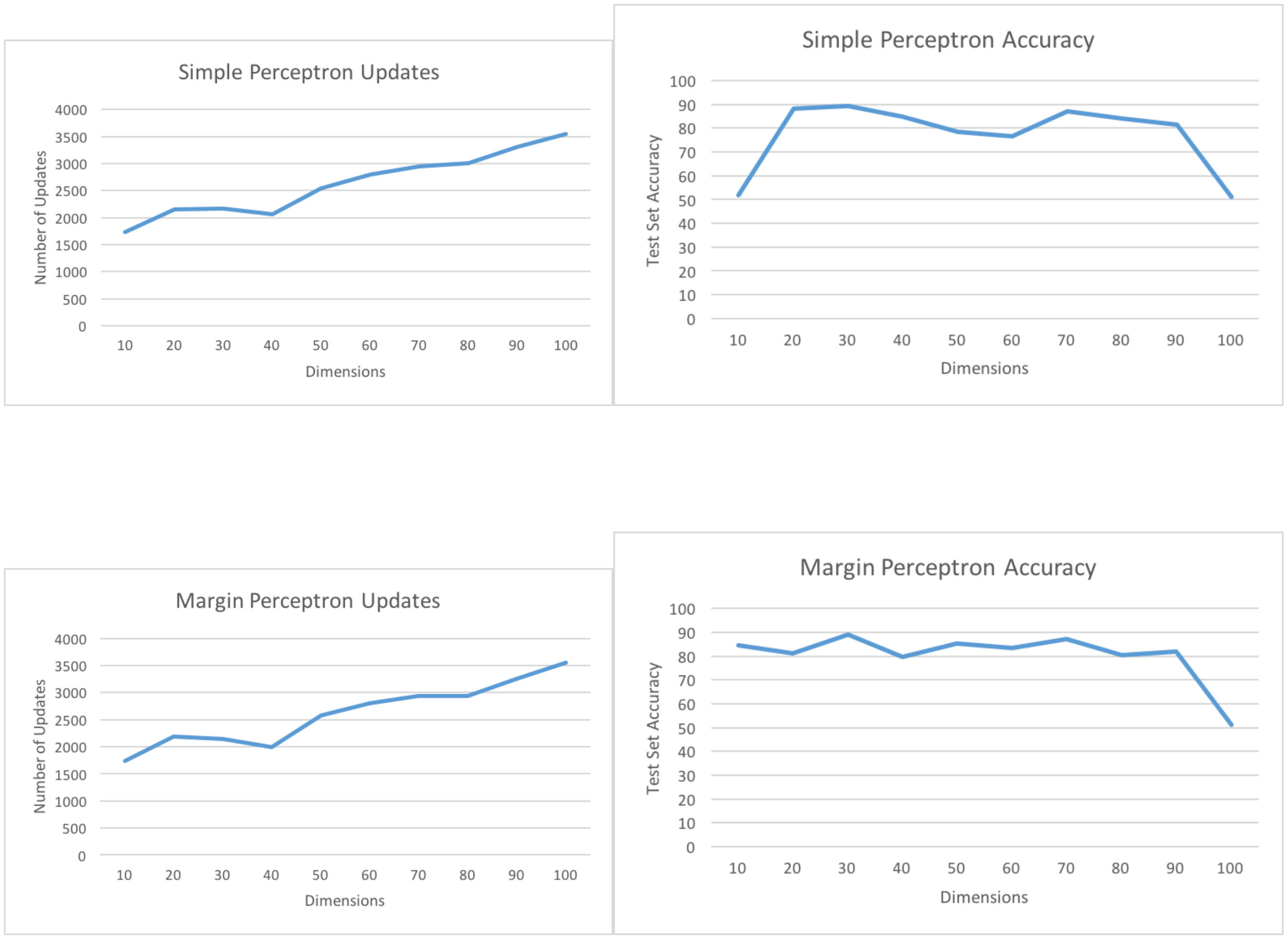

Figure 3: Data1- For 10 epochs

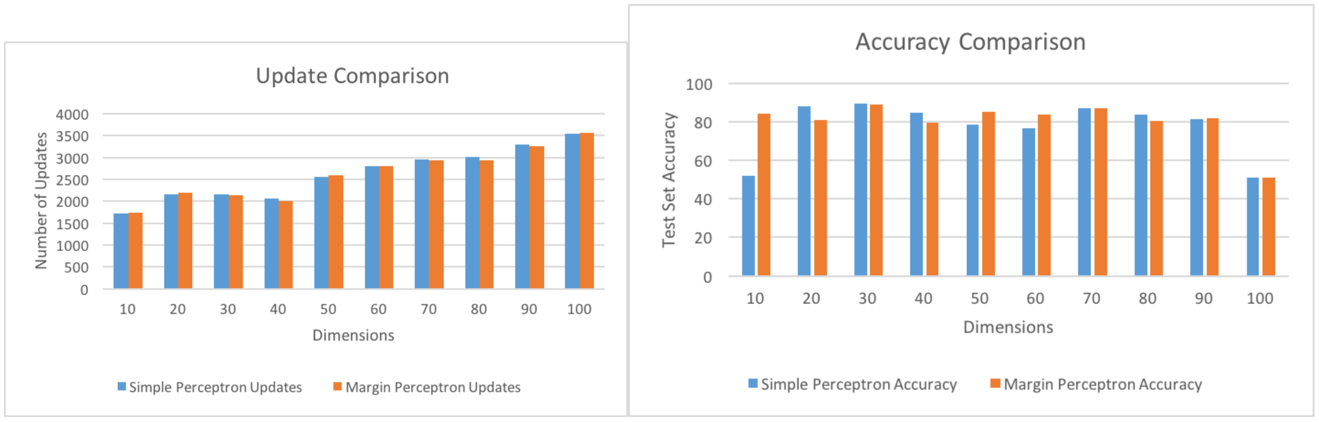

Figure 4: Data1 - Comparisons

Experiment 4

Solution:

Using the margin array as above, I ran a 10-fold cross validation for each train & test file to get the best margin for Aggressive Perceptron with Margin. On running on only 10 epochs my accuracy was dropping drastically and so to validate i ran for 100,500 and 1000 epochs. Here i noticed that in Data0 for 100 and above epochs the drop of accuracy against dimensions was gradual.

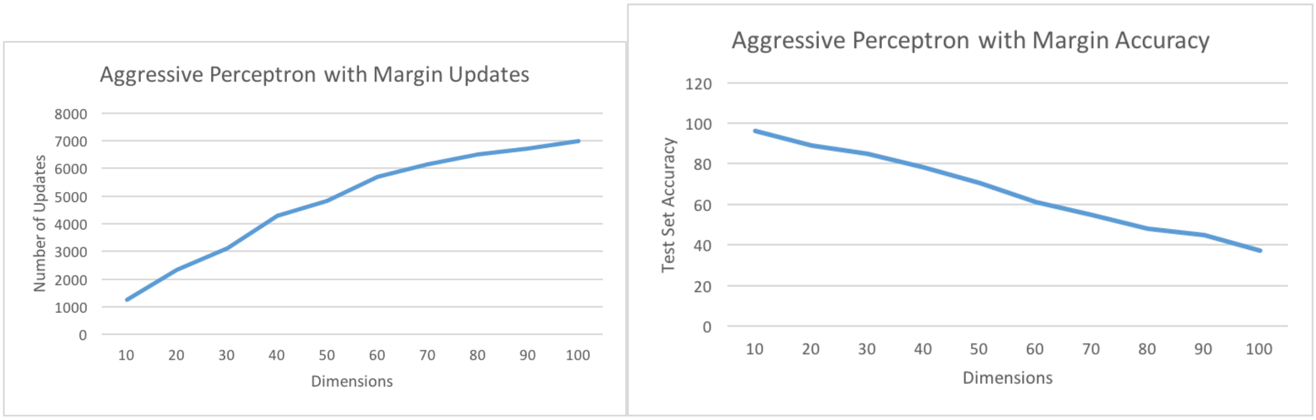

Figure 5: Data0 - For 10 epochs

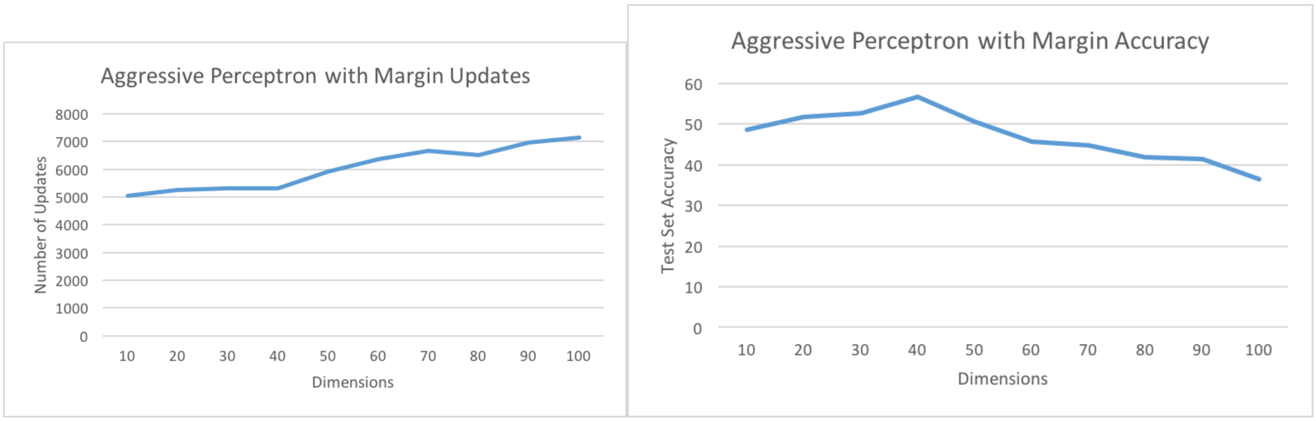

Figure 6: Data1 - For 10 epochs

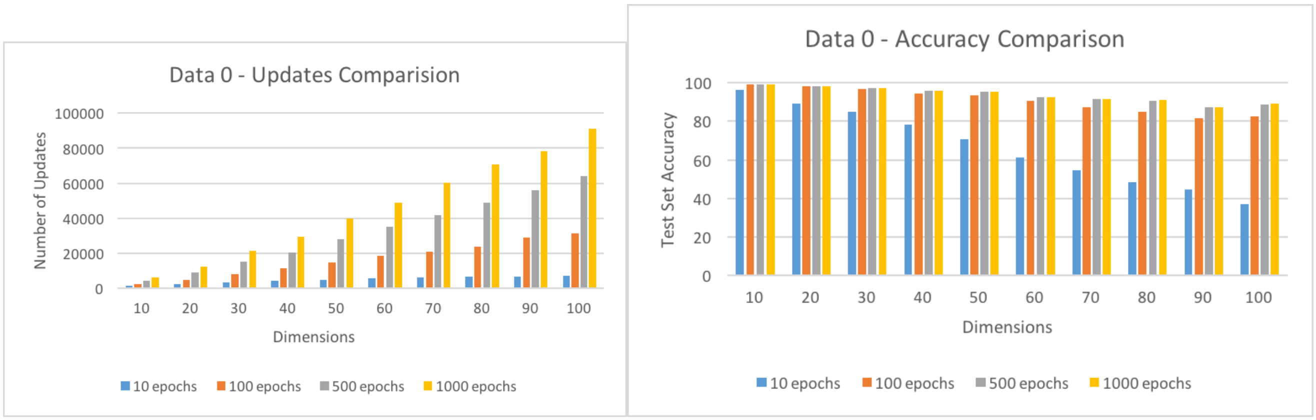

Figure 7: Comparison of epochs - for Data0

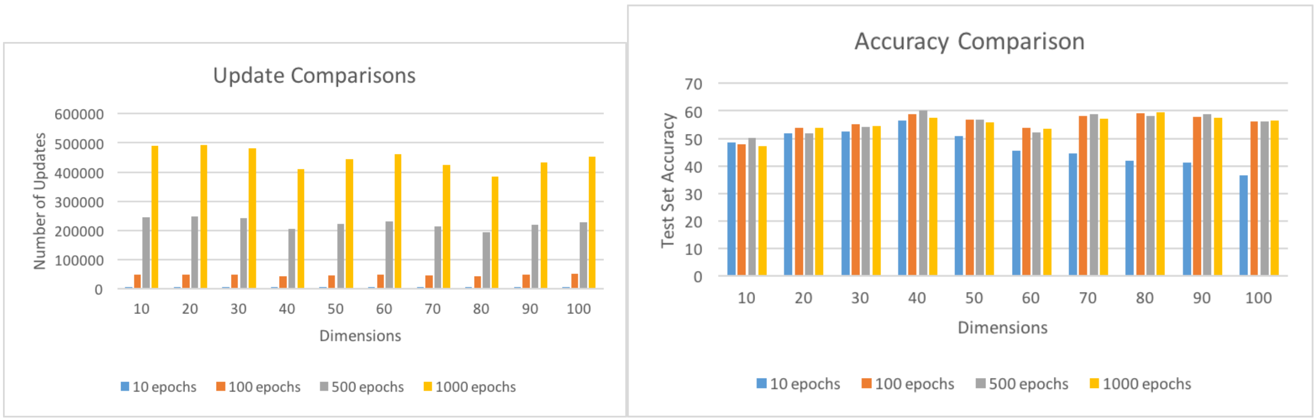

Figure 8: Comparison of epochs - for Data1

4 Important Note

I have added a excel file with all the data along with the other files and folders.