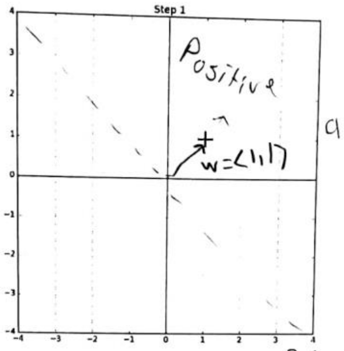
446 A2 - Johan J How 1143297

0 Policies

0.1 *None*
0.2 *None*
0.3 I have read and understood these policies

1 Perceptron Exercise

initial $W = \langle 0, 0 \rangle$



$a = \sum_{d}^{D} W_d \cdot X_d = 0$

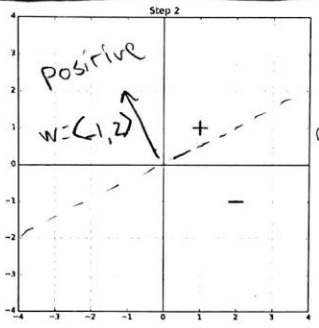
$y \cdot a = 0 \leq 0$

update:

$w \leftarrow w + x$

$w = \langle 1, 1 \rangle$

$B \perp w$



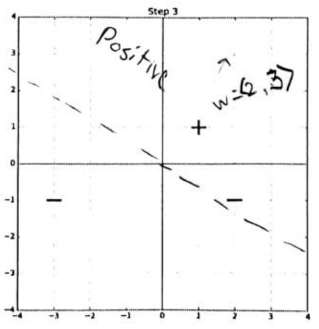
$a = \sum_{d}^{D} W_d \cdot x_d = 1$

$y \cdot a = -1 \leq 0$

update:

$w \leftarrow w - x$

$w = \langle -1, 2 \rangle$



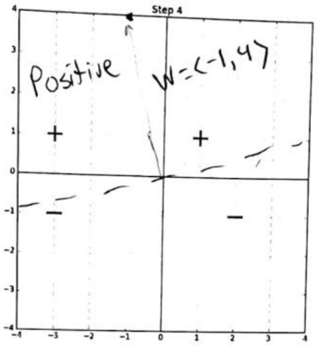
$a = \sum_{d}^{D} W_d \cdot x_d = +$

$y \cdot a = - \leq 0$

update:

$w \leftarrow w - x$

$w = \langle 2, 3 \rangle$



$a = \sum_{d}^{D} W_d \cdot x_d = -$

$y \cdot a = - \leq 0$

update:

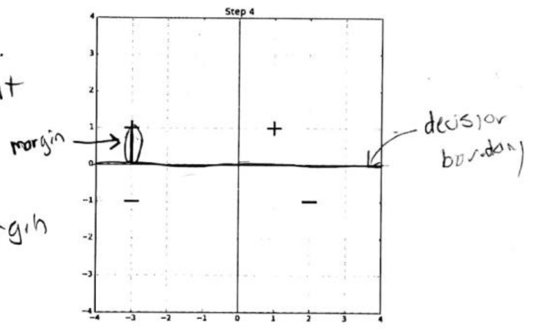
$w \leftarrow w + x$

$w = \langle -1, 4 \rangle$

2. See the slides for a defintion of the geometric margin and for a correction to CIML. Does our learned perceptron maximize the geometric margin between the training data and the decision boundary? If not, draw the maximum geometric margin, decision boundary on the graph below. [5 points]

It doesn't maximize.
The decision boundary that lies on the x axis maximizes.

The maximum geometric margin is 1.



3. Assume that we continue to collect data and train the perceptron. If all data we see (including the points we just trained on) are linearly separable with geometric margin $\gamma = 0.5$ and have maximum norm $\|\mathbf{x}\|_2 \leq 5$, what is the maximum number of mistakes we can make on future data? [5 points]

$\frac{1}{\gamma^2}$ updates $= \frac{1}{0.25} = 4$ updates. We already made 4 updates, so 0 left.

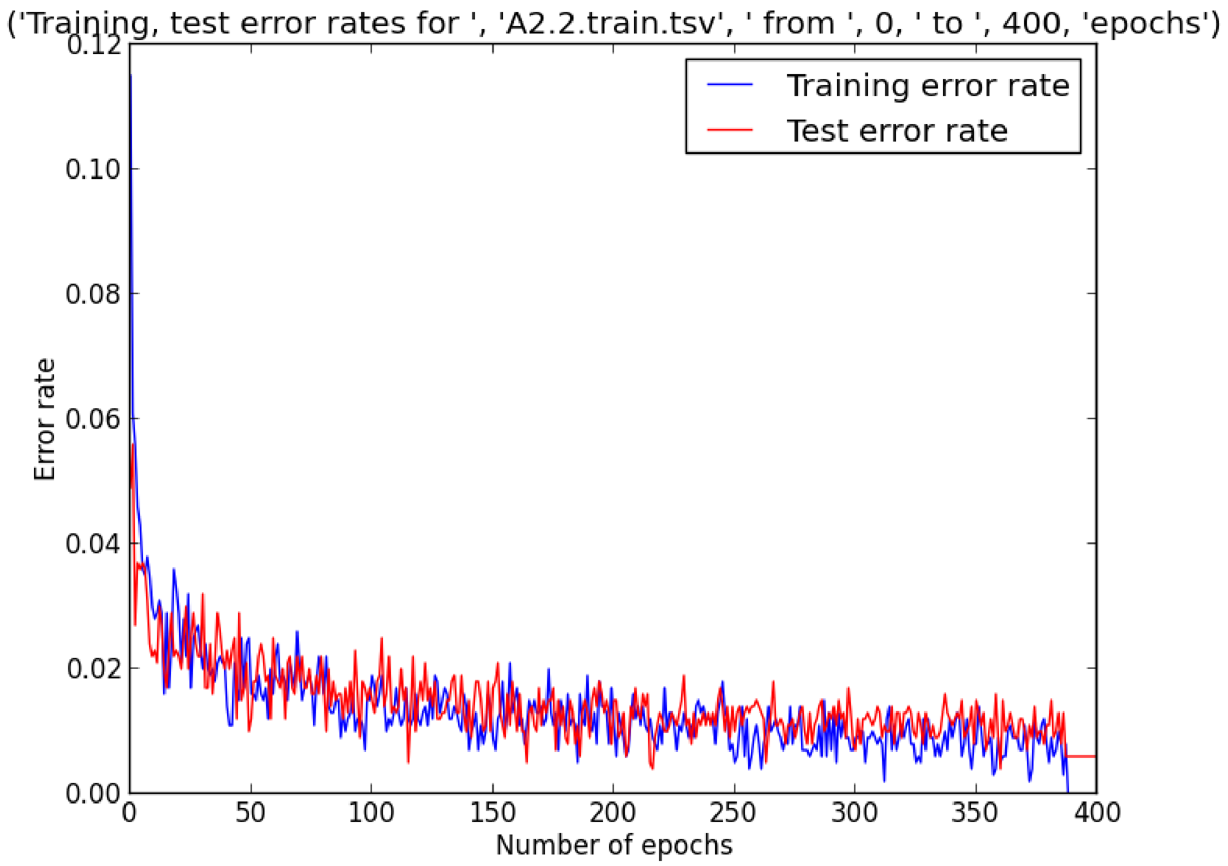
2 Implementation: Perceptron [70 points]

Implement the perceptron learning algorithm for binary classification, as described in class, from scratch, in Python. All of the files mentioned here are available in the tarball `A2.tgz` available on Canvas.

2 Implementation: Perceptron

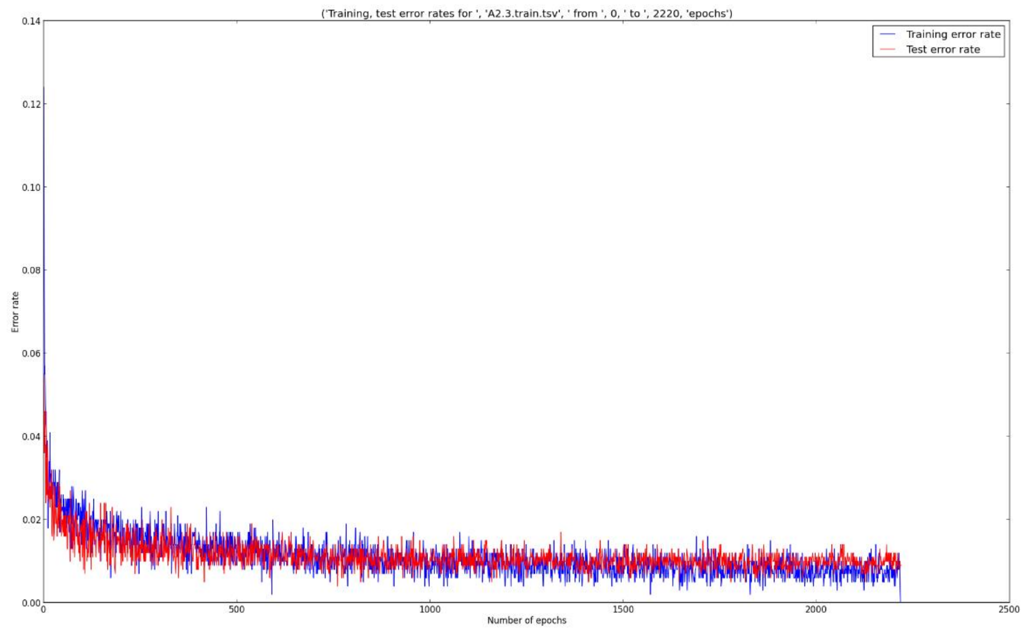
Plots for Dataset 2:

0 to 400 epochs

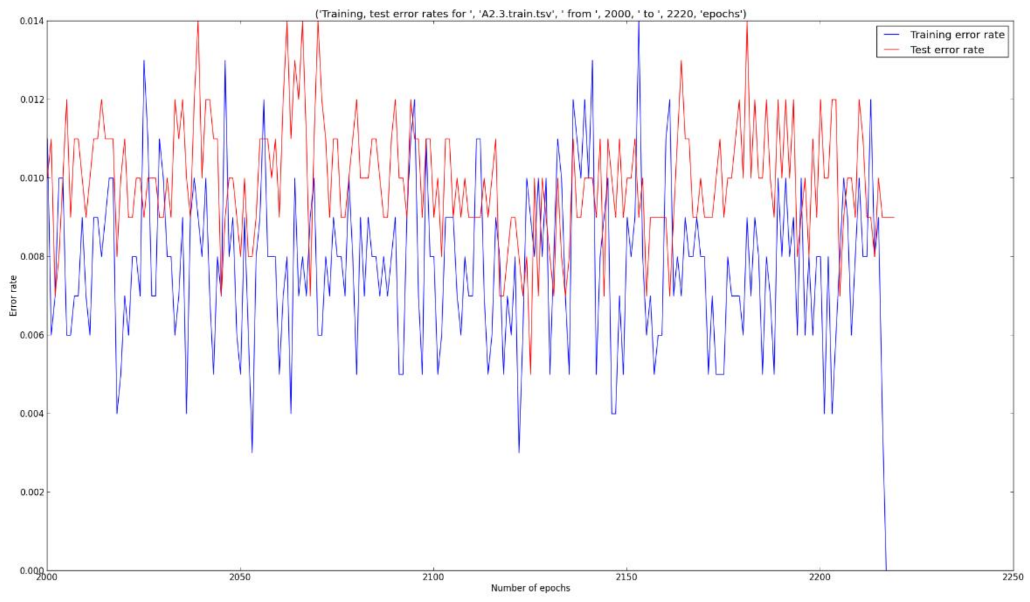


Plots for Dataset 3:

0 to 2220 epochs

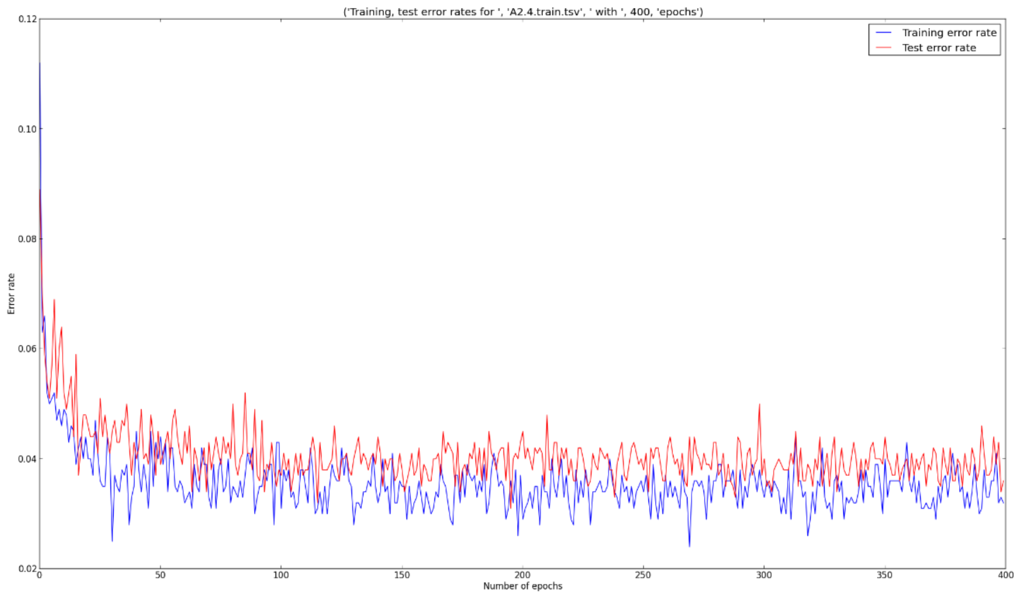


2000 to 2220 epochs

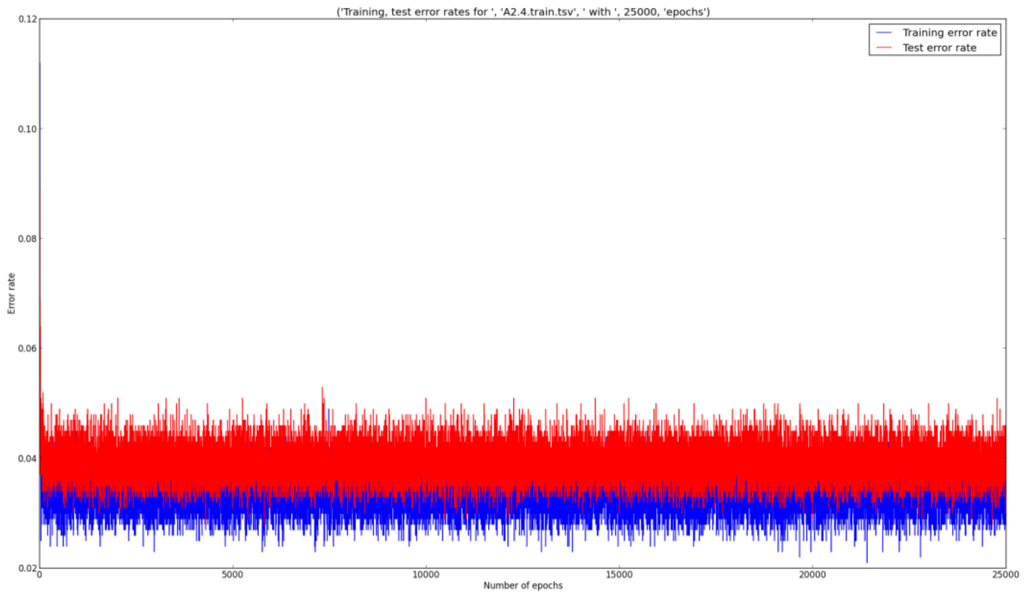


Plots for Dataset 4:

0 to 400 epochs

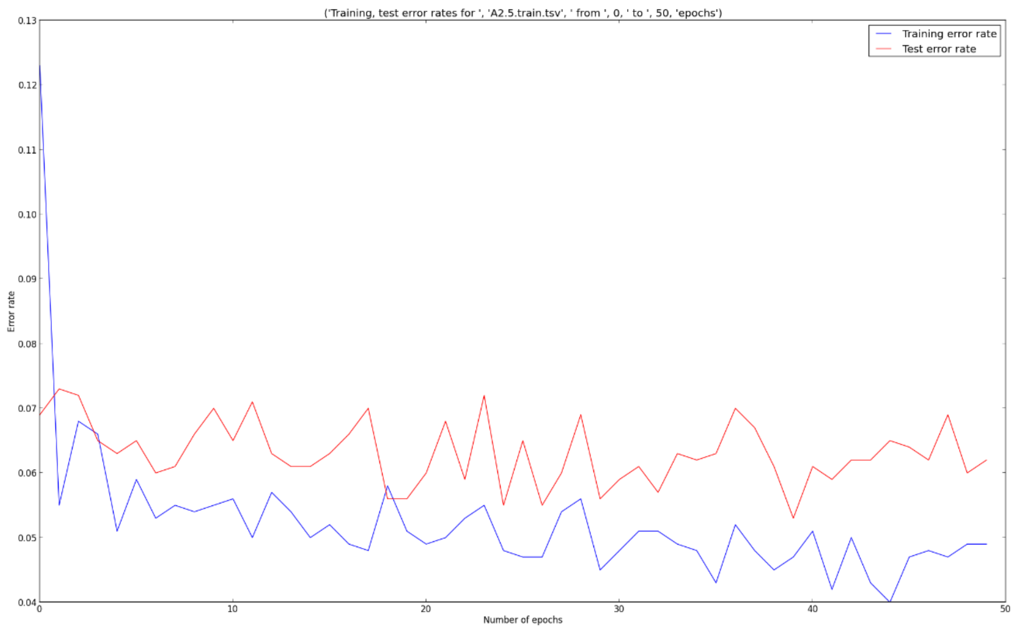


0 to 25000 epochs

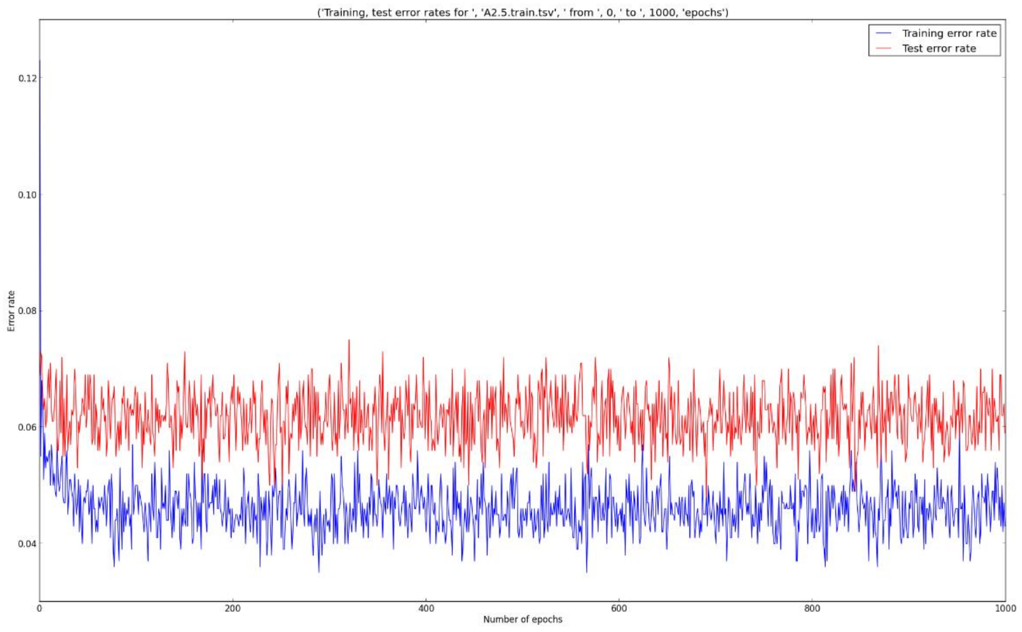


Plots for Dataset 5

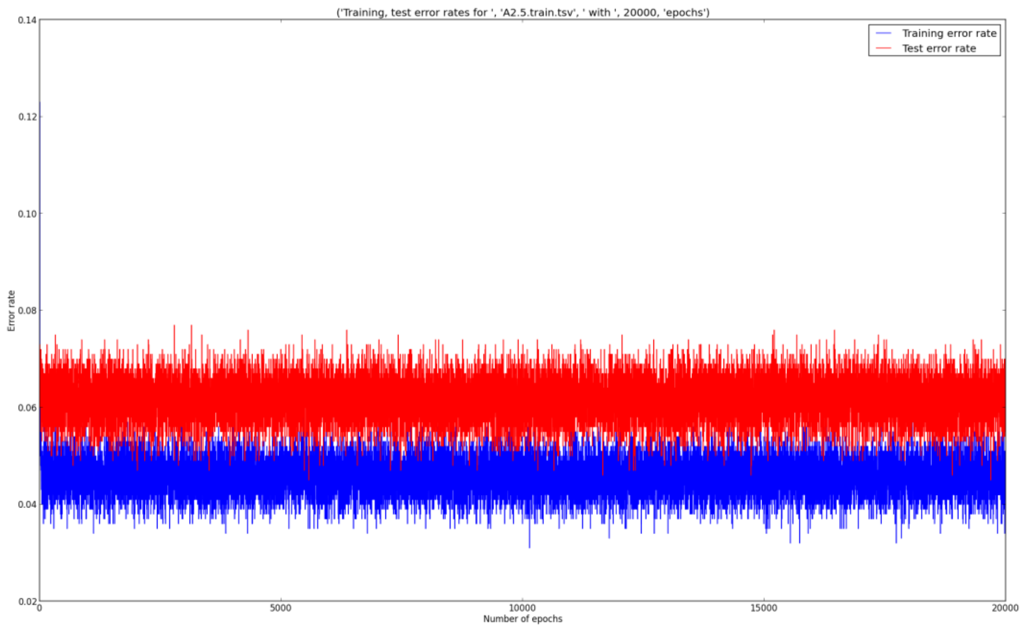
0 to 50 epochs



0 to 1000 epochs

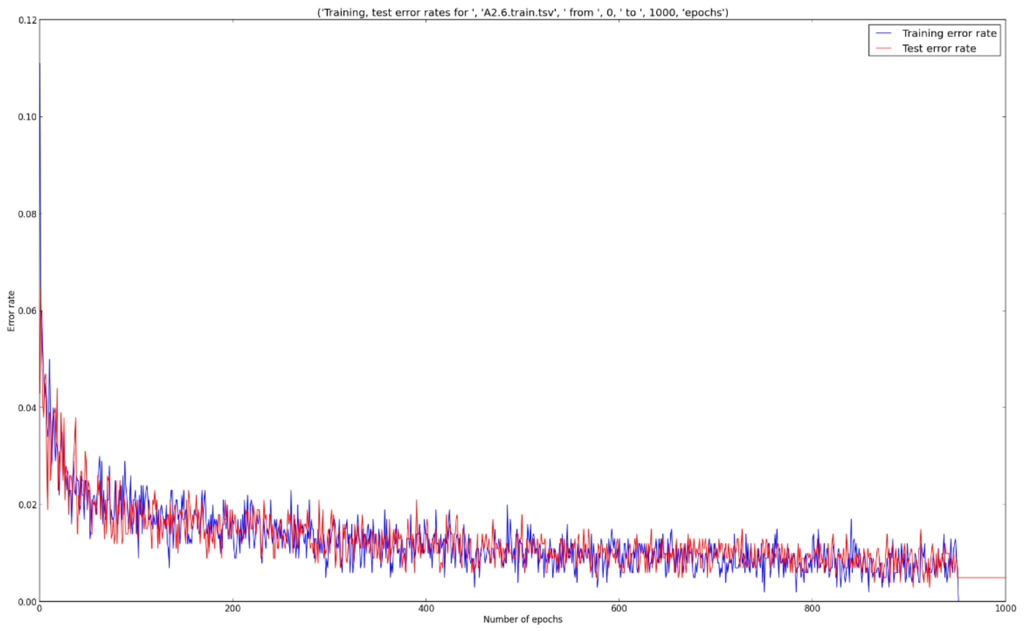


0 to 20000 epochs

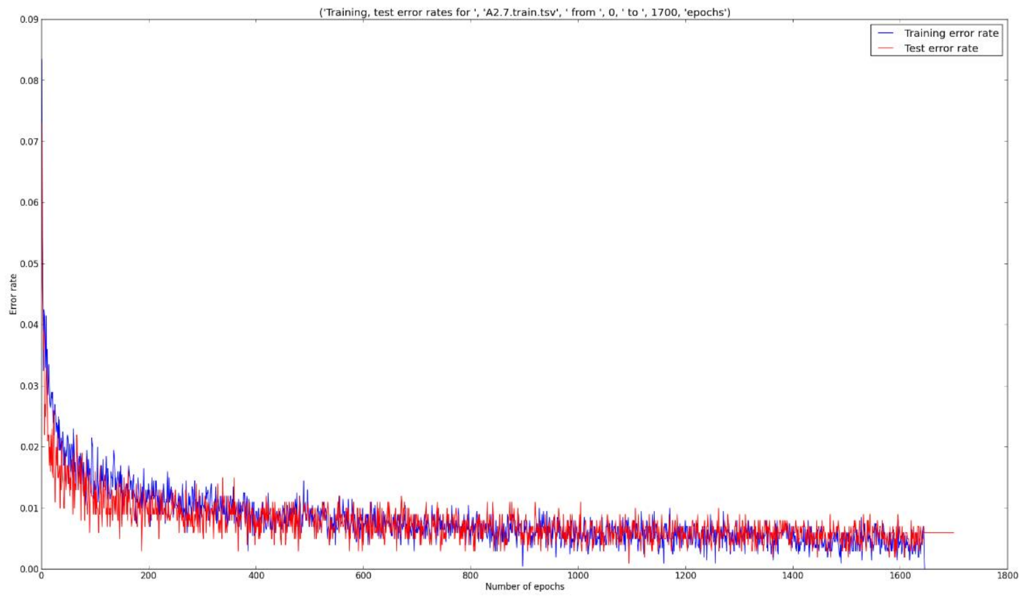


Plots for Dataset 6

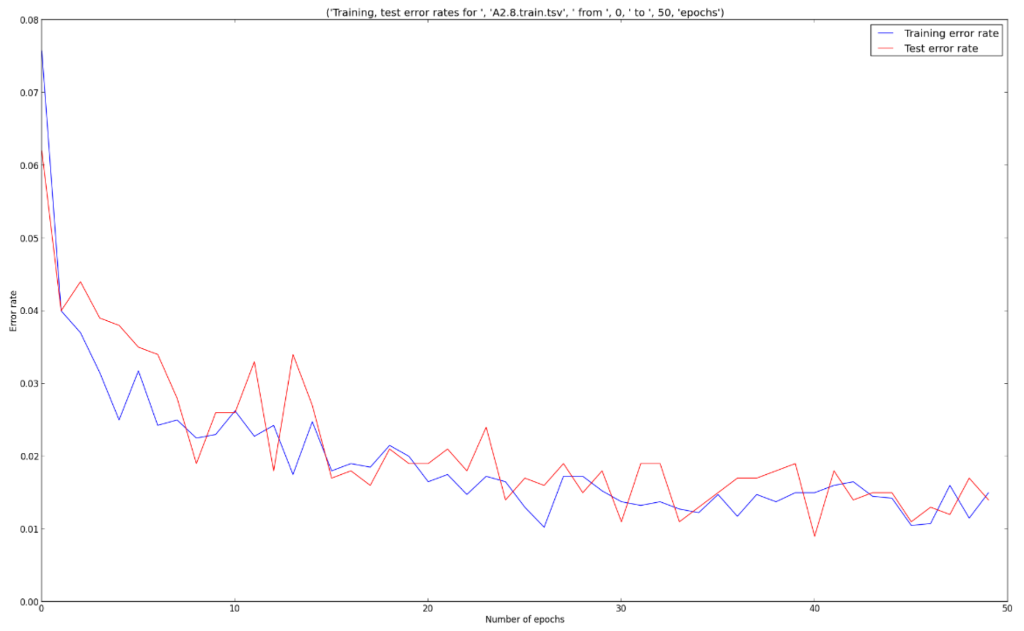
0 to 1000 epochs



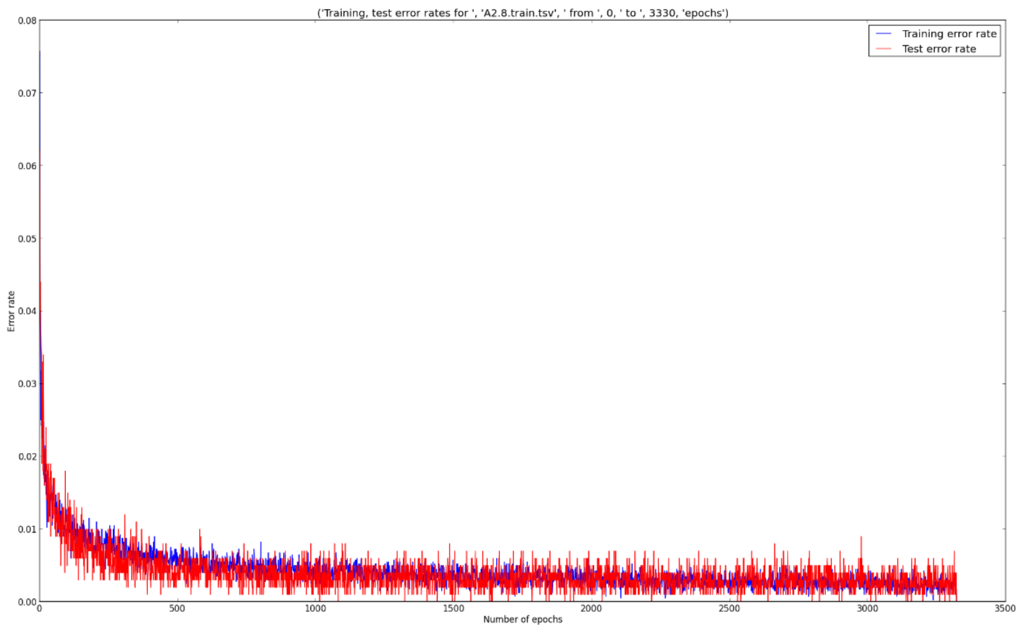
Plots for Dataset 7
0 to 1700 epochs



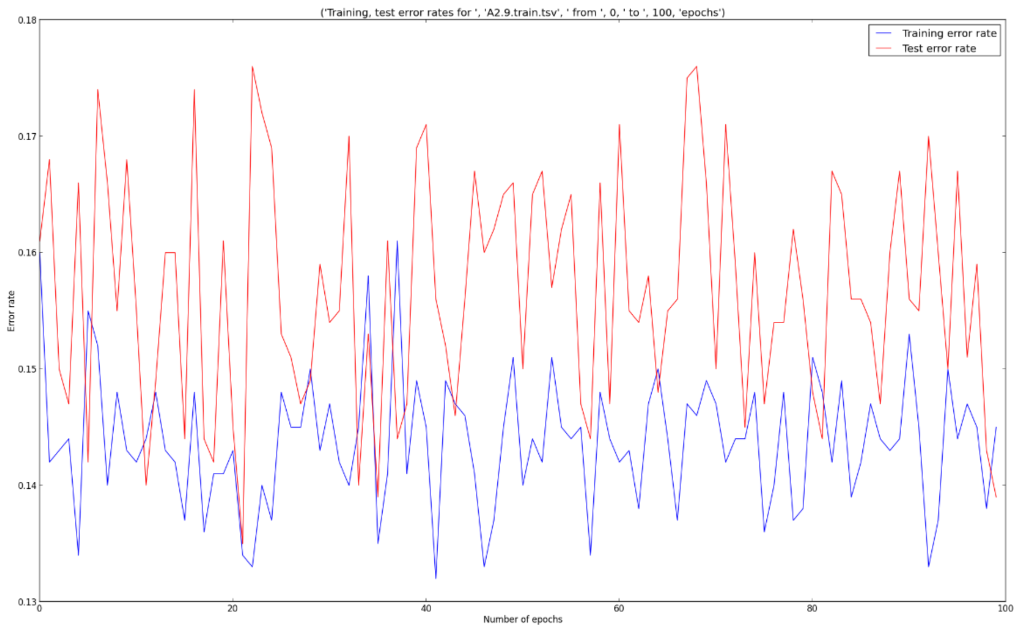
Plots for Dataset 8
0 to 50 epochs



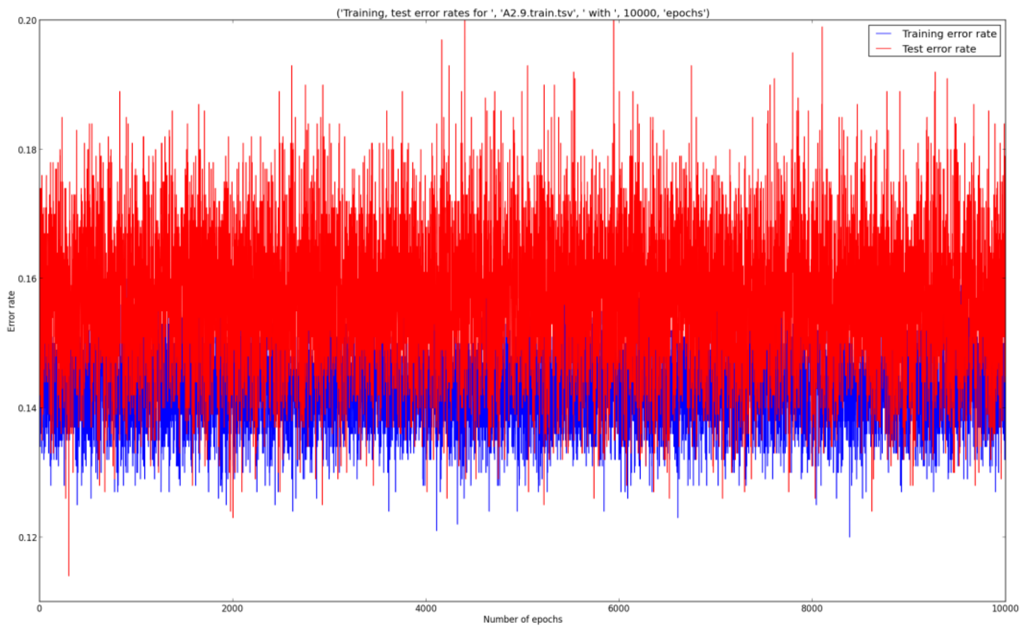
0 to 3300 epochs



Plots for Dataset 9
0 to 100 epochs



0 to 10000 epochs



Dataset	Question 2	Question 3	Question 4	Question 5
2	Yes, it is linearly separable because at around the 400th epoch the training error goes to 0. This shows that the perceptron found a decision boundary that could completely separate data points with positive labels from those with negative labels.	No, I don’t think it overfits the data. The test data error rate when the training error goes to 0 is not significantly higher than the data training error rate.		
3	Same as Dataset2, except around the 2220th epoch.	Yes, I think it overfits the data because if we take a look at the 2000-2220 epochs plot, one can see a period where the test error is strictly greater than the training error.		
4	No, After 25000 epochs, the training set error had not gone to 0, which shows the perceptron was unable to correctly classify all data points with the correct labels.	No. At 0-50 epochs, both test and training errors show a downwards trend, but after then there does not seem to be a downwards trend for either.		
5	No, After 20000 epochs, the training set error had not gone to 0, which shows the perceptron was unable to correctly classify all data points with the correct labels.	No. Neither the training error or the test error show a downwards trend as number of epochs increases.		
6	Same as Dataset2, except around the 950th epoch.	No, for the same reason as 2.	I suspect 10 might be noise.	

7	Same as Dataset2, except around the 1600th epoch.	Yes. When the training error rate goes to 0, the test error rate is significantly higher than at lower epochs.	I suspect 10 might be noise, due to the fact it varies greatly between Datasets 6, 7, 8 despite them being part of the same distribution.	
8	Same as Dataset2, except around the 3330th epoch.	No, for the same reason as 2.	I suspect 10 might be noise.	
9	No. After 10000 epochs, the training set error rate does not look much lower than at epoch 0.	No. Neither the training error or the test error show a downwards trend as number of epochs increases.		

Surprise!

If we take the resulting W from the result of running perceptron on Dataset 6, and use that as the initial W for 7, we can see that 7's training set error goes to 0 in fewer epochs (810 as compared to 1600). Similarly, if we take the resulting Ws from each fo 6, 7, 8 and apply it to the others, we will see an reduced number of epochs needed for the training set error to go to 0. This is because instead of starting with a W of <0, 0, …, 0>, we can start with a W that is closer to the one that will be the decision boundary.

3 Beat the Perceptron

I chose AveragedPerceptronTrain as described in CIML chapter 4, and implemented it in avgperceptron.py.

I chose to use Dataset 4, because it training and test error appeared to go down in the beginning but did not decrease afterwards, even after 25000 epochs.