Abbas Paryavi

Project 1

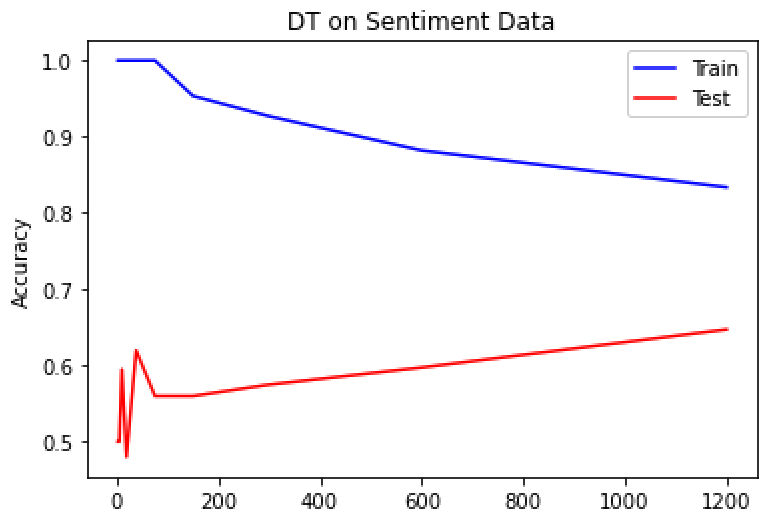
WU #1

That is because in truth we are computing the classification accuracy. Basically what that piece of code does is compare the actual value which is stored in Y, with the predicted value which is always going to be one due to our classifier’s prediction. We are just getting the average of the data that match. If they match the result is 1 and otherwise, it will be 0. Taking the average/mean will give the classification accuracy.

WU #2

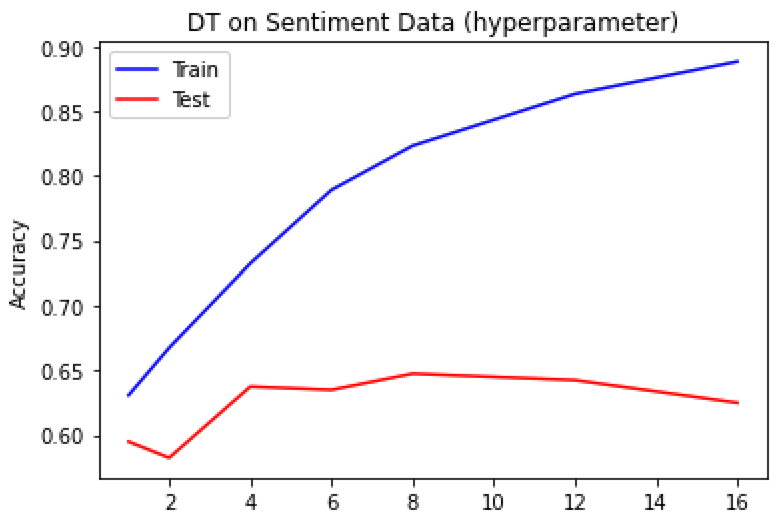
The training accuracy keeps going down as the number of classifier points increases. This is because on a small sample such as 2, 3, 5, etc. it is very easy to find the best decision tree with 100% accuracy, especially with a max depth of 9. However, this causes underfitting with insufficient classification points to create a good decision tree. But as the number of points increases, we will see a reduction in the training accuracy, because we will face different cases that can’t be generalized with a simple decision tree.

For the test accuracy, we see a general increase as the number of points increases but at the very beginning, we see a flux between 0-100 points in the test accuracy. This is because at these points we don’t have enough training data thus our decision trees are very general, plus we have overfitting with high training accuracy and low test accuracy and thus we get this flux. However, as the number of points increases, we see the flux fading away.



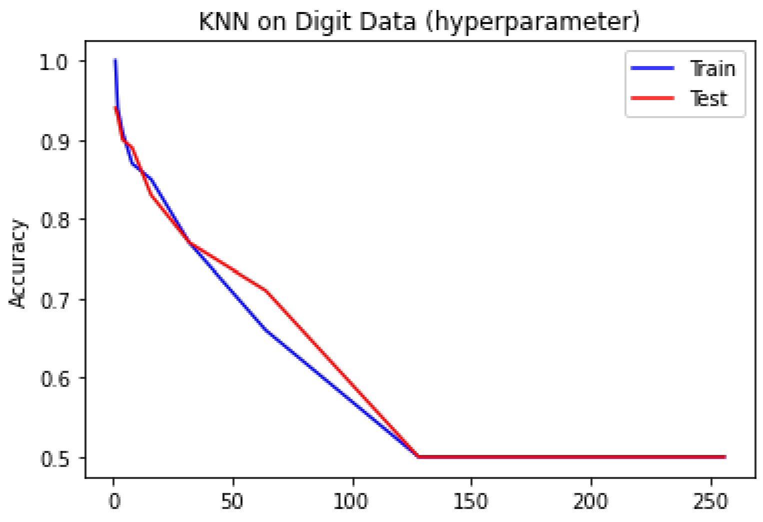
WU #3

The training accuracy is pretty much guaranteed to increase as the depth increases, that is because as the depth increases we can create a better decision tree that is more precise towards our training data but we aren't guaranteed to have an increase in test accuracy because we will be creating an overfitting on the training data. However, as we have lower depths such as 1, 2, etc. we will be creating a lot of generalizations and thus prone to underfitting which will result in lower training and test accuracy.

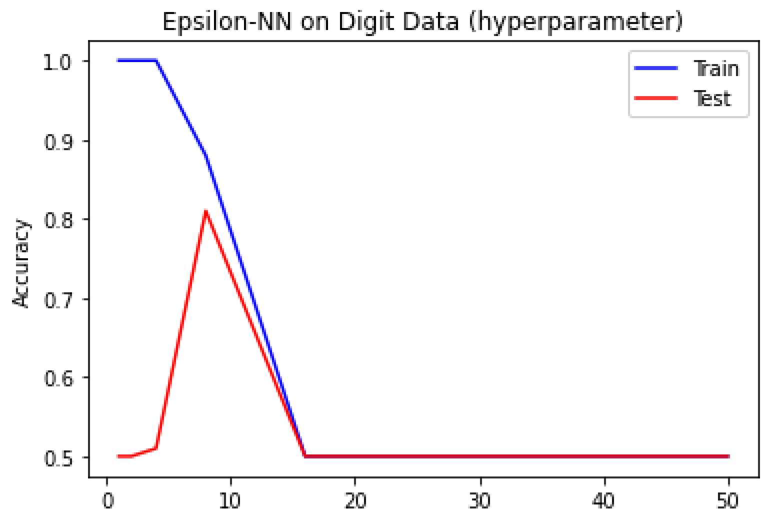


WU #4

Looking at the training/test curves for k and epsilons with k running from 1-256, we don’t see much of an overfitting using k as our parameter. We have very high training accuracy initially but the test accuracy is also very high. However, as the k increases, we can see serious underfitting, with very low numbers on both training and test accuracies. This is especially noticeable after k = 100.



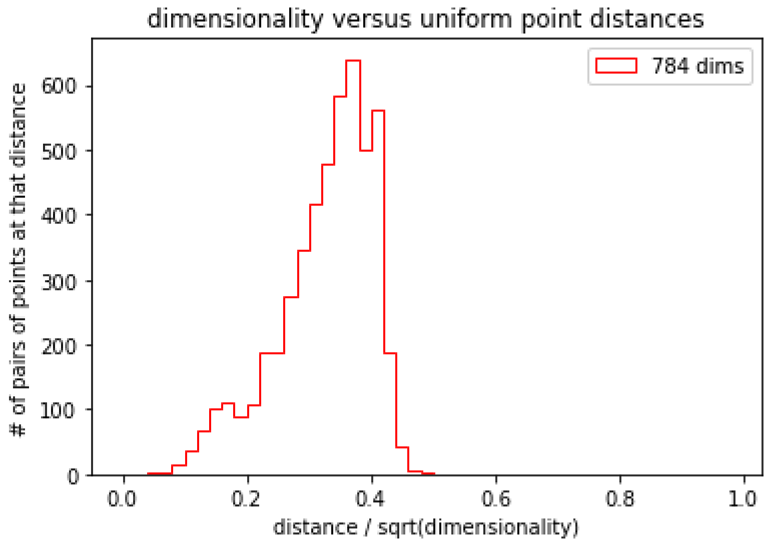
Looking at the epsilon data ranging from 1-50, we see huge overfitting initially when epsilon is roughly less than 6 with high training accuracy and low test accuracy. From roughly eps = 6 to eps = 12 we have the peak precision and anything after that has huge underfitting with low test and training accuracies.


Epsilon-NN on Digit Data (hyperparameter)
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
10
20
30
40
50
Train
Test

WU #5

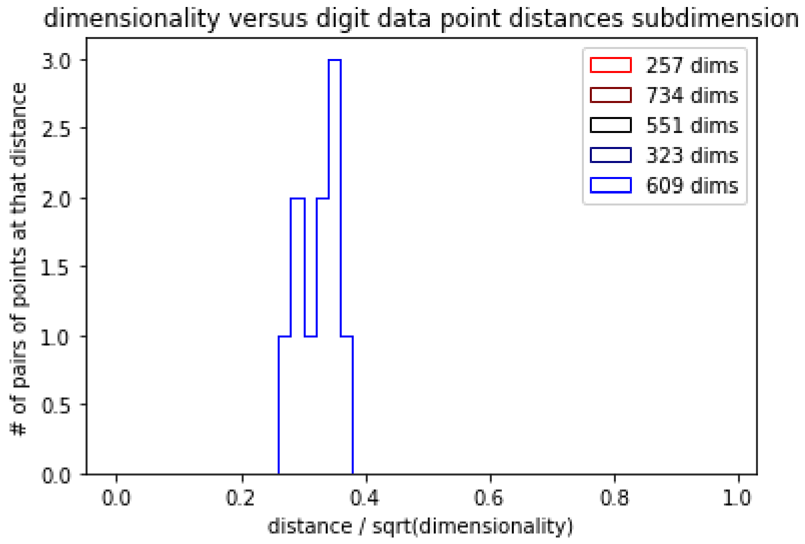
a)

```
D=784, average distance=9.10764
```



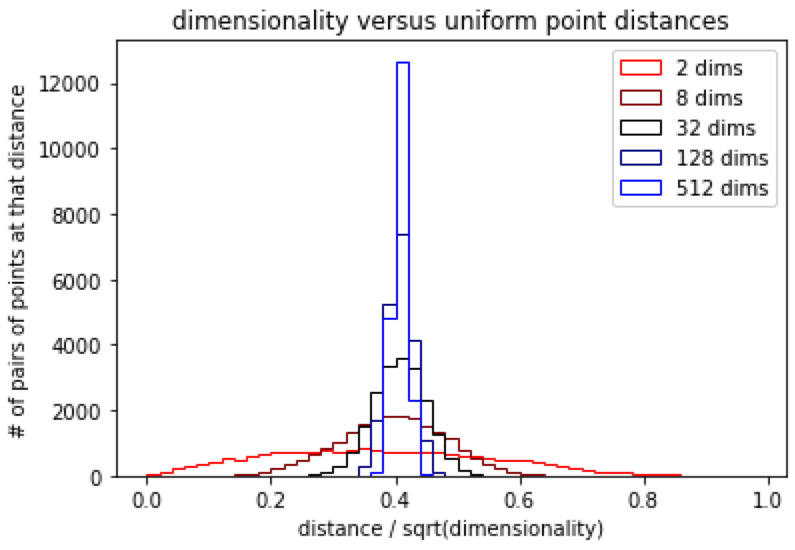
b)

```
D=257, average distance=5.11964
D=734, average distance=8.65209
D=551, average distance=7.49633
D=323, average distance=5.7395
D=609, average distance=7.88101
```

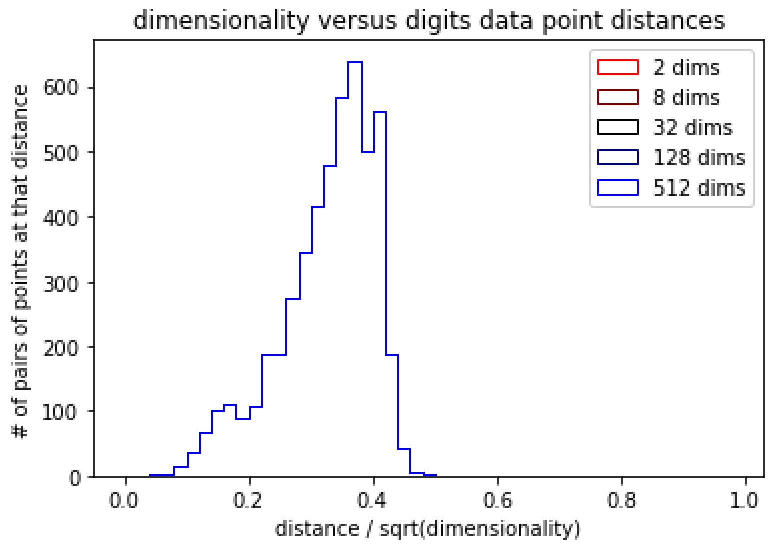


c)

```
D=2, average distance=0.531501
D=8, average distance=1.11712
D=32, average distance=2.30048
D=128, average distance=4.60629
D=512, average distance=9.21618
```



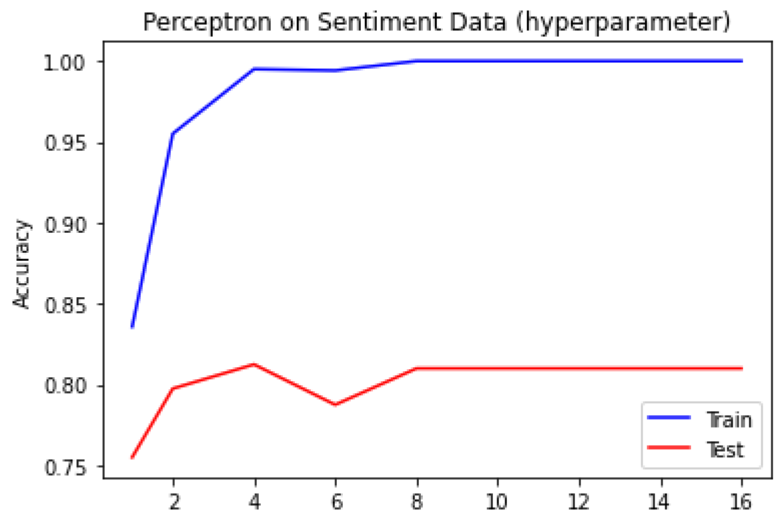
```
D=2, average distance=0.460005
D=8, average distance=0.92001
D=32, average distance=1.84002
D=128, average distance=3.68004
D=512, average distance=7.36008
```



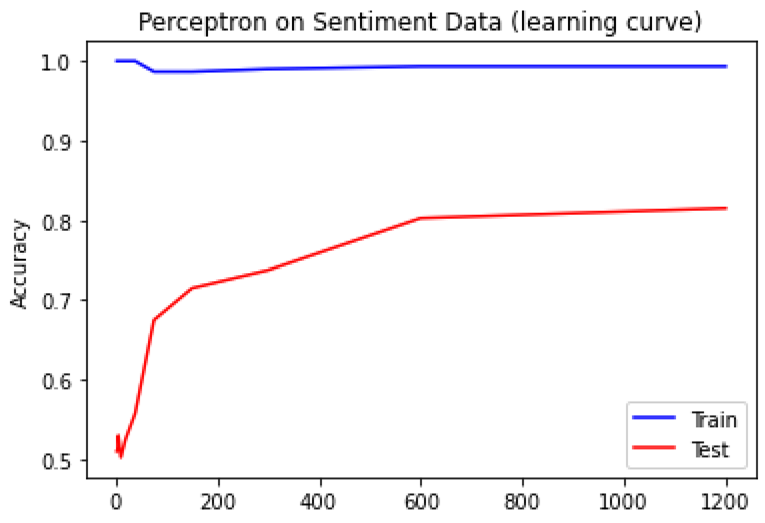
We can clearly see that first of all the uniform points are way more uniformly distributed, however, that is expected. But also at low dimensions, we don’t see anything for the digits data, and it only has pairs at high dimensions such as 512.

WU #6

a)


Perceptron on Sentiment Data (hyperparameter)
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
2
4
6
8
10
12
14
16
Train
Test

b)


Perceptron on Sentiment Data (learning curve)
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
200
400
600
800
1000
1200
Train
Test