CMSC422 Proj1 WU

Shuo Wang, Chenhongshu Yu

WU1:

Since the dataset only contains 1 or -1. When we check if the data > 0, we can distinguish the 1 and -1 perfectly and generate True and False for each.

eg. (datasets.TennisData):

predict data: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

training data: [-1. -1. 1. 1. 1. -1. 1. -1. 1. 1. 1. 1. 1. -1.]

predict data (with > 0):[True True True True True True True True True True True True True True]

training data (with > 0):[False False True True True False True False True True True True True False]

Then, we compare predict data and training data and get training accuracy list. When predict data is the same as training data, we have True, otherwise, we have False:

eg. (datasets.TennisData):

training accuracy: [False False True True True False True False True True True True True False]

Then we take mean of that list, since True = 1, False = 0, and total element number is 14. Taking mean is equivalent to check the percentage of True in the list, which is also the percentage of predict data is the same as training data.

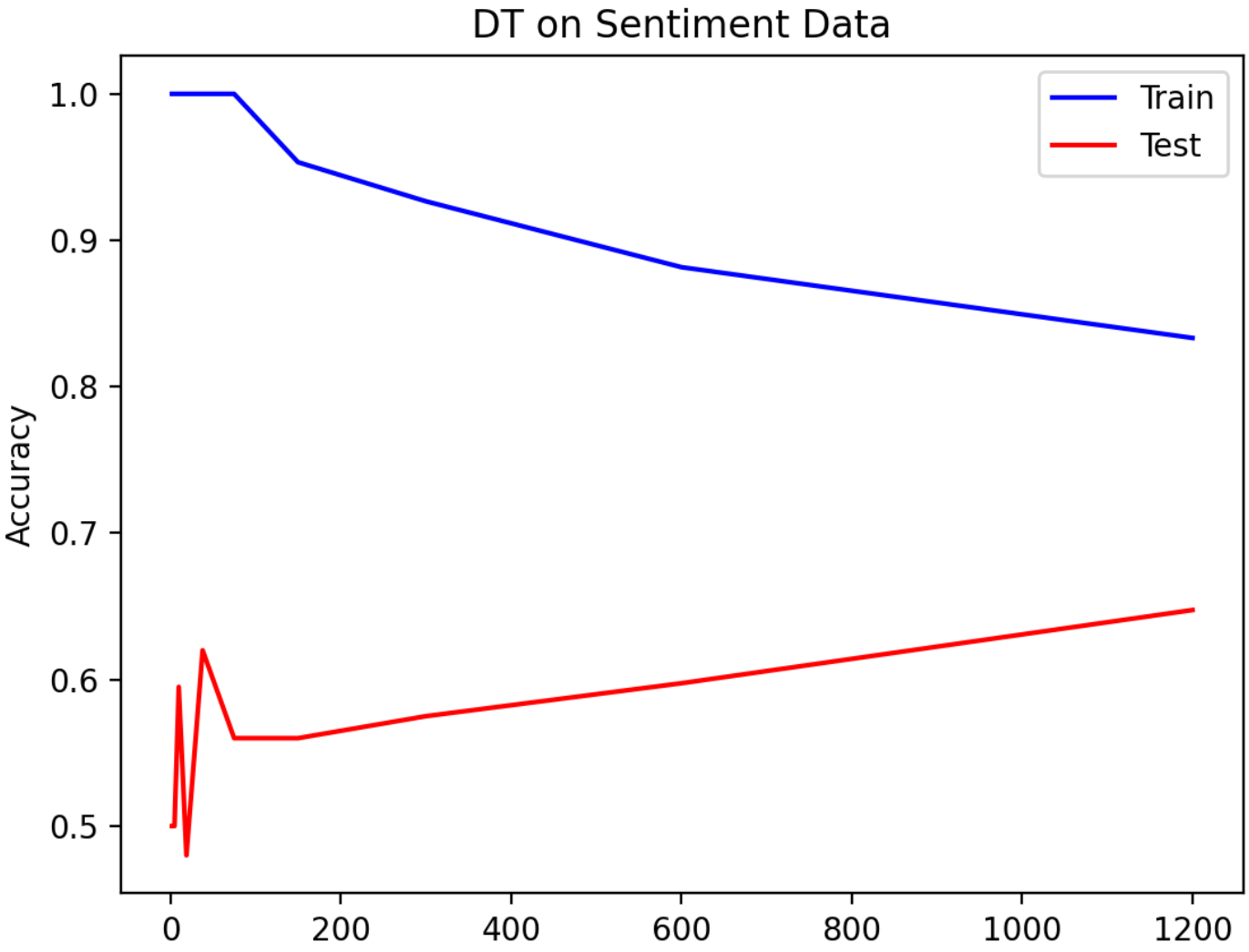
Thus, it is exactly the accuracy of classification.

WU2:

As the number of training samples increase, the accuracy of training accuracy (roughly) going down. It is normal since the accuracy cannot stay at 1 all the time(Notice training accuracy starts at 1). If the training accuracy stays at 1 or increases later,we may have overfitting in the classification because our decision tree focus too much on the training data. Not to mention there can be noisy data, which can also cause training accuracy decreases to avoid overfitting.

The testing accuracy is not increasing as the number of training samples increase. We can see that especially when training size is under 200. It is because when the training sample is small, we have underfitting, which means the decision tree has not learned enough from the training data. Additionally, the total sample is small, when we calculating accuracy, a little difference can cause dramatically changes in the testing accuracy. As we can see that as the training size increases, the increase of testing accuracy becomes monotonically.

This also explains why there is a jaggedness in the test curve toward the left.


DT on Sentiment Data
Train
Test
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
200
400
600
800
1000
1200

WU3:

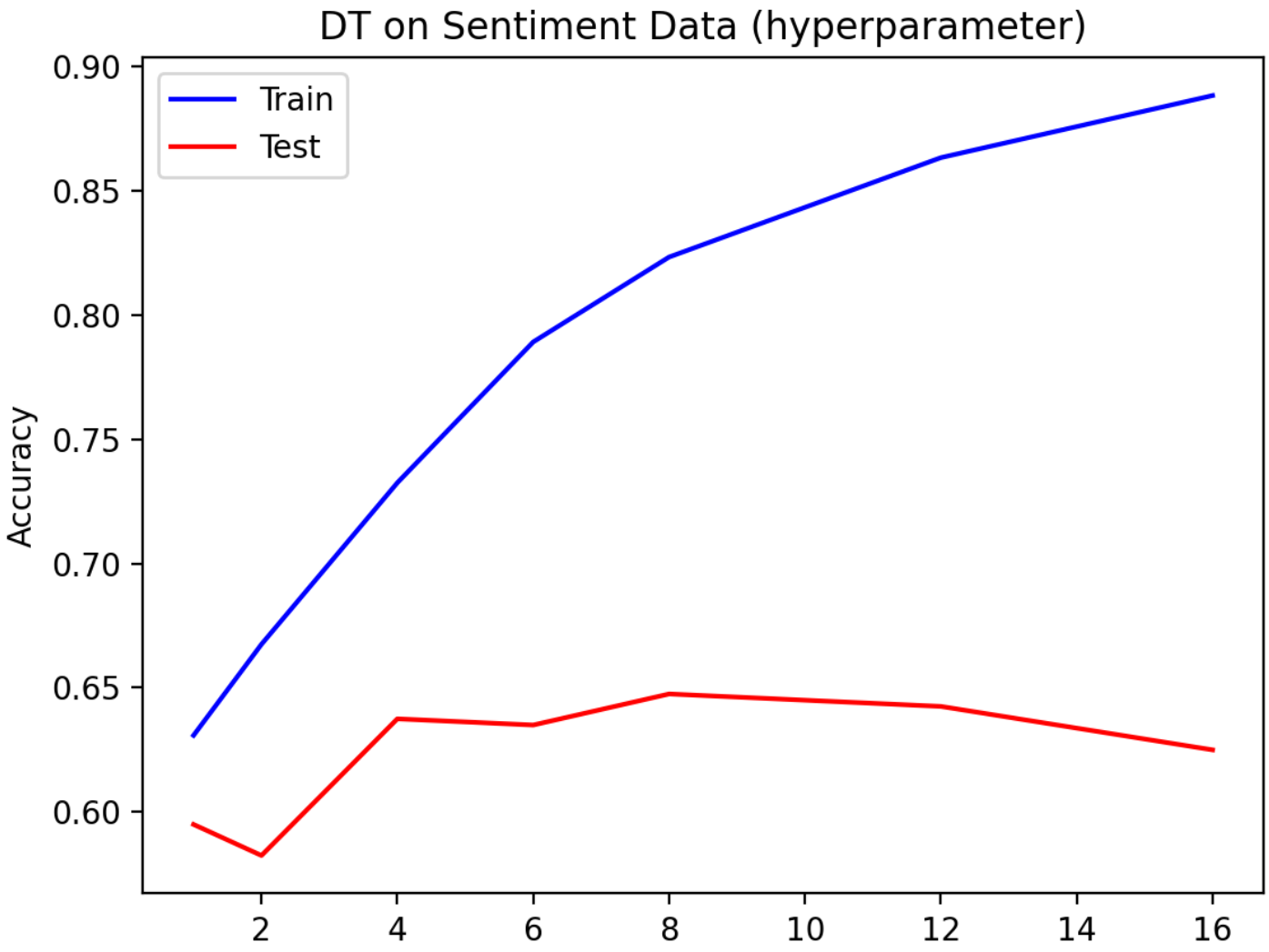
Training accuracy monotonically increasing is guaranteed to happen and the hill in test accuracy is expected to happen.

As the depth increases, the decision tree uses more features to more accurately fit the training data. Thus, the training accuracy keeps increasing.

The testing accuracy increases at the beginning because we have underfitting, our decision does not have enough data to predict. But as the depth increases, our decision tree is more complete, and then testing accuracy increases.

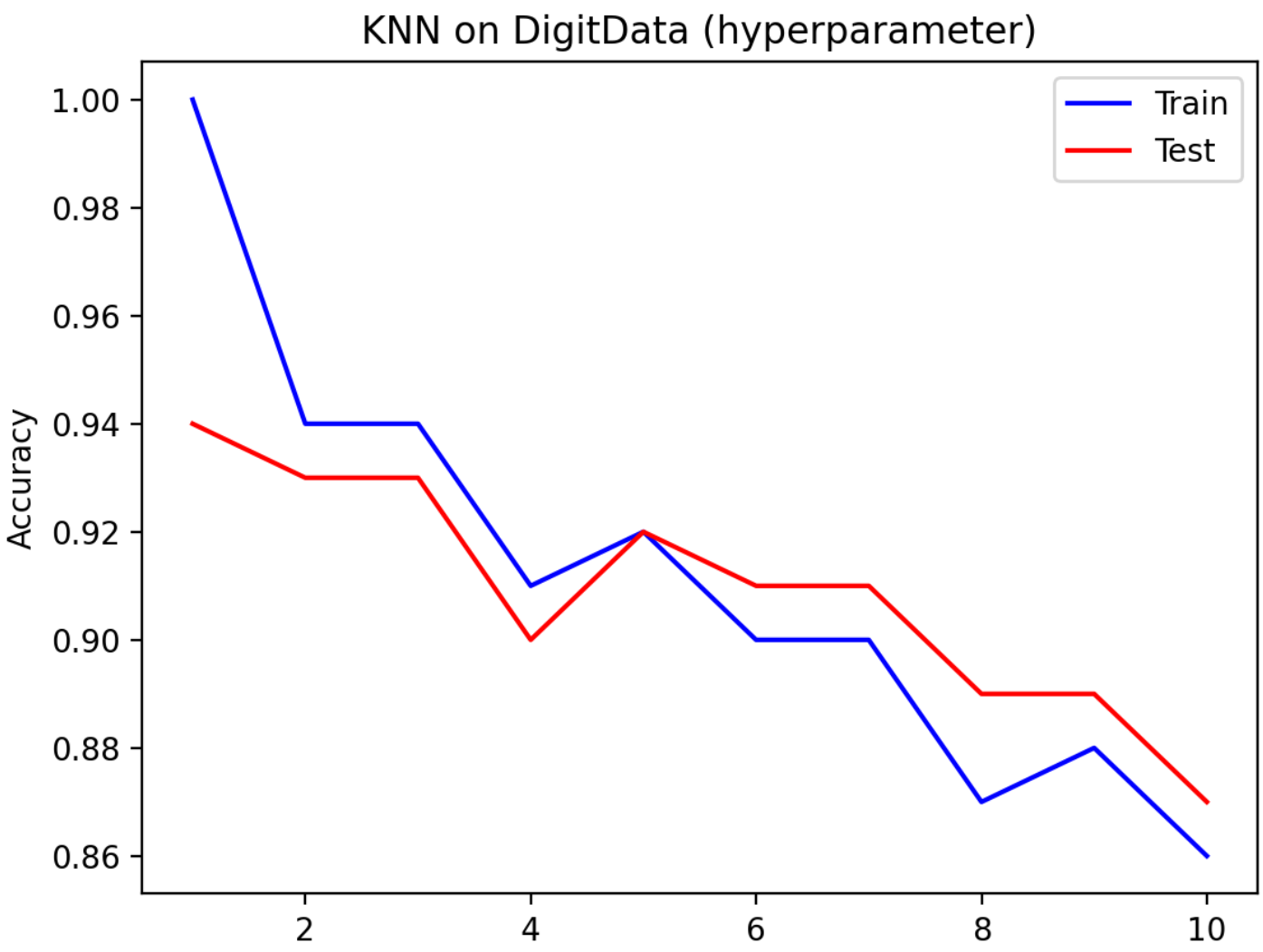
If we continue increasing the depth, the program will have overfitting. When we have overfitting, the decision tree is not generalized enough for testing data. So the testing accuracy decreases.

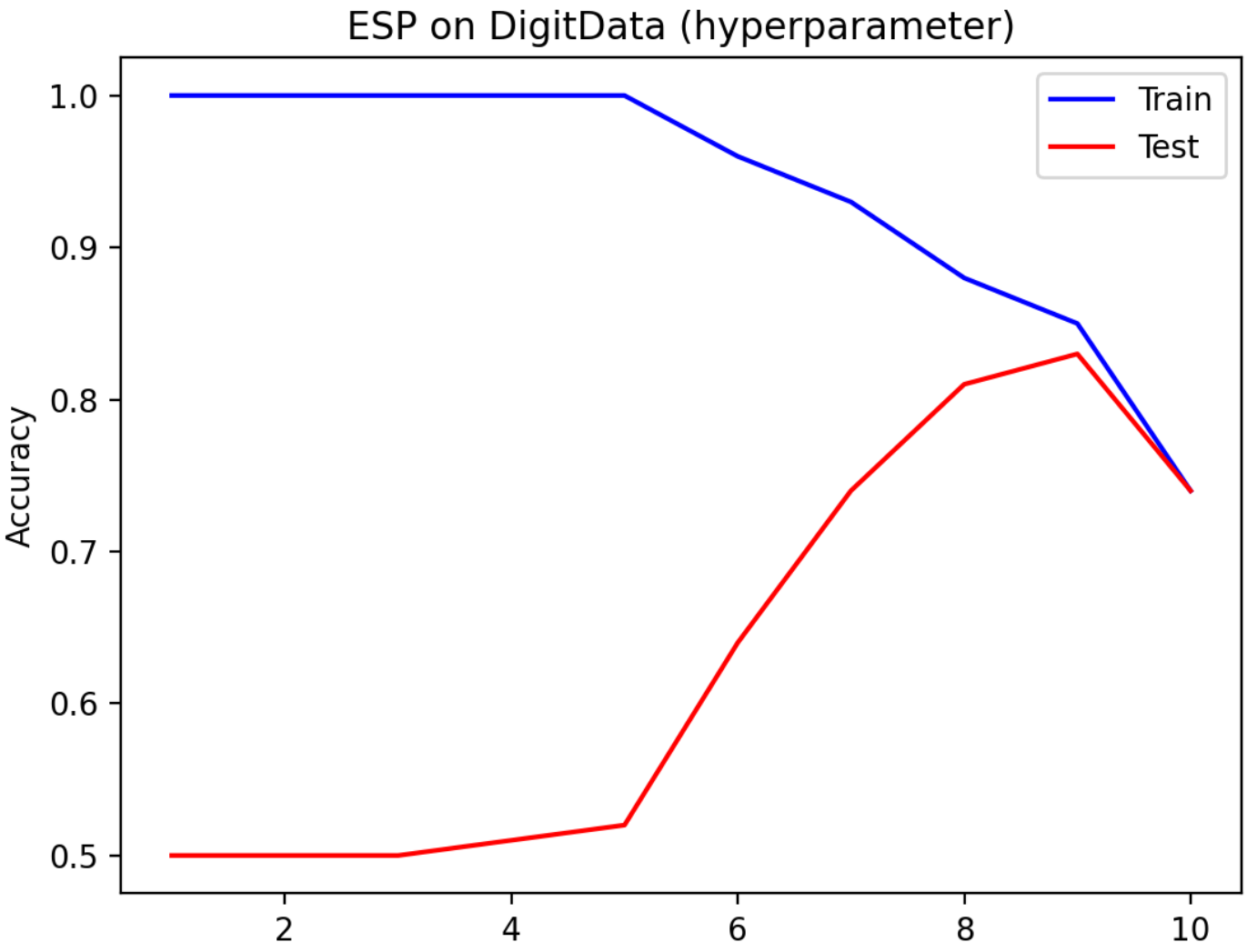
Thus, we have a "hill"-like graph for testing accuracy.

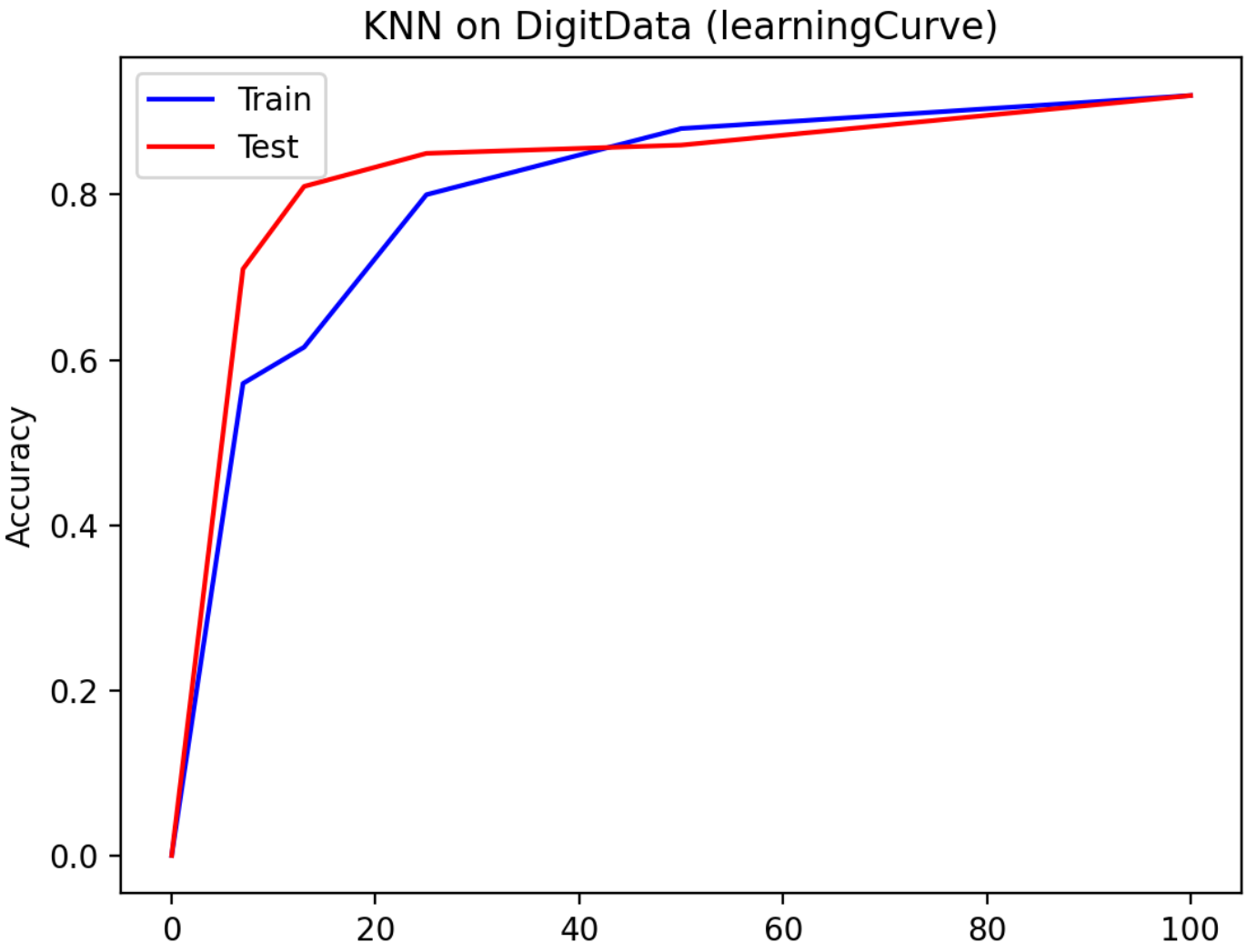

DT on Sentiment Data (hyperparameter)
Train
Test
Accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
2
4
6
8
10
12
14
16

WU4:

For KNN at different K, there is no clear evidence that shows overfitting or underfitting. However, clearly there is overfitting when $\epsilon < 5$ since our testing accuracy is 1 and testing accuracy is very low. There's a sweet spot at k=5.


KNN on DigitData (hyperparameter)
Train
Test
Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
2
4
6
8
10


ESP on DigitData (hyperparameter)
Train
Test
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
2
4
6
8
10


KNN on DigitData (learningCurve)
Train
Test
Accuracy
0.0
0.2
0.4
0.6
0.8
0
20
40
60
80
100

WU5:

A:

D=784, average distance=9.10764

B:

D=95, average distance=3.17037

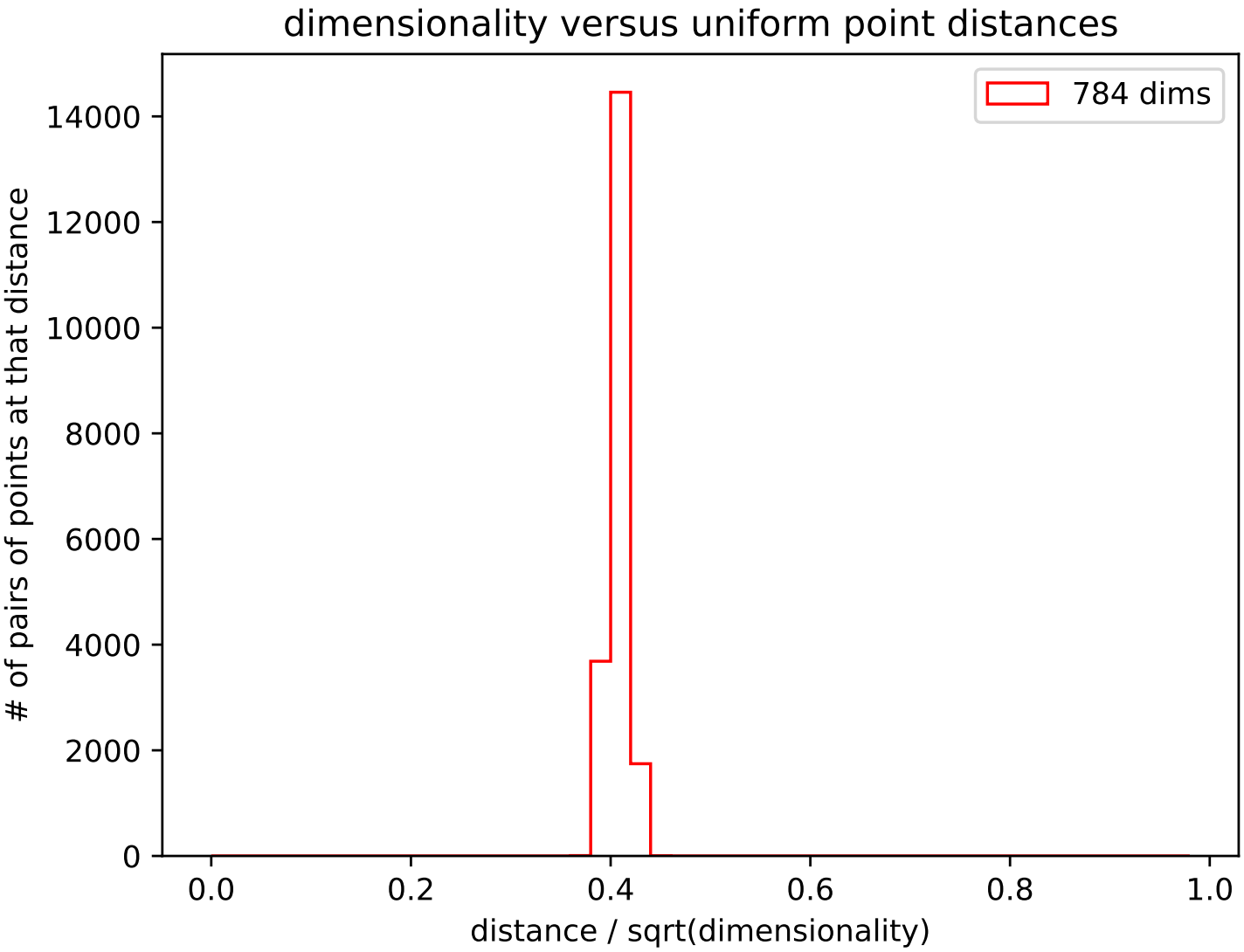
C:

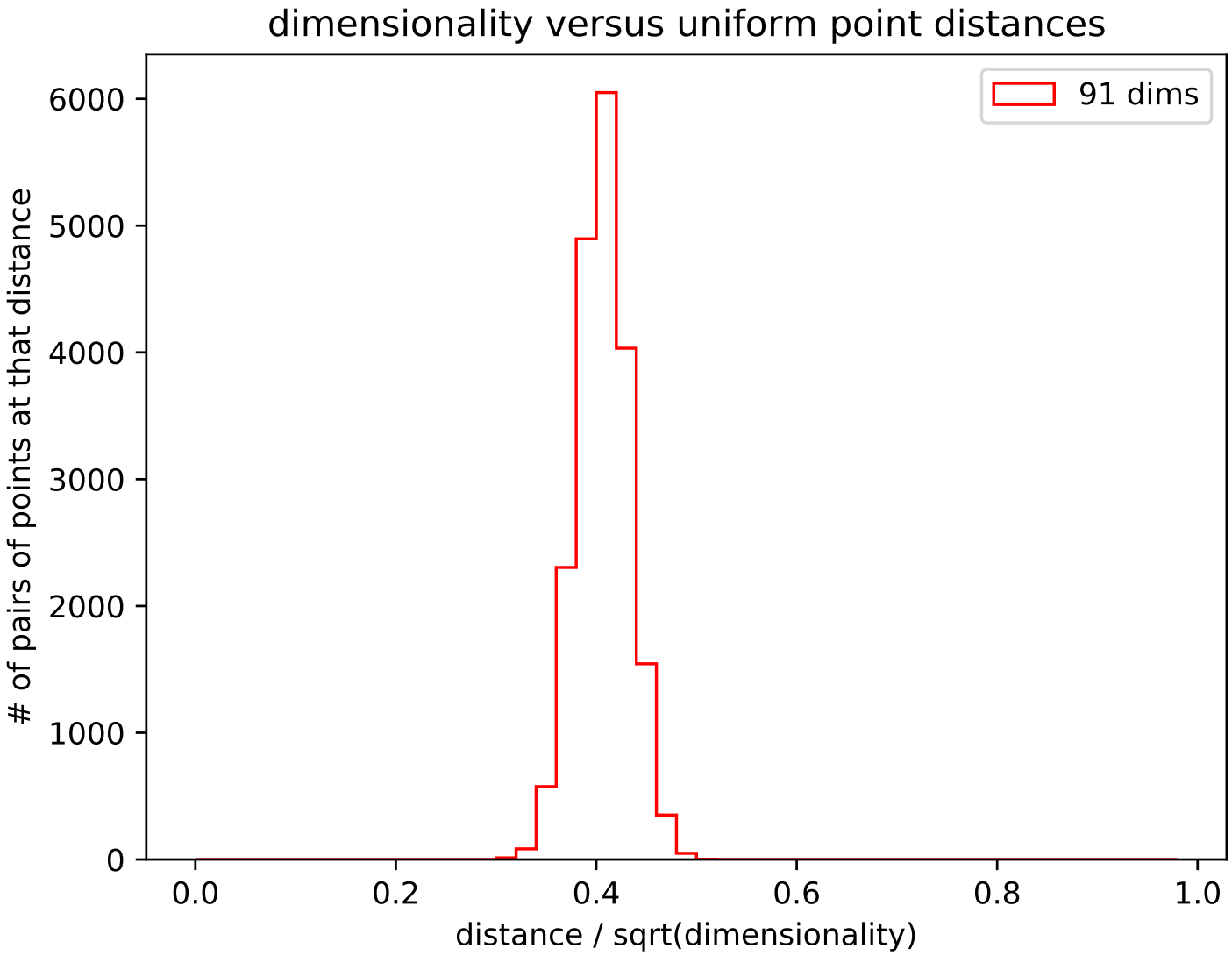
Random data shows a normal distribution.

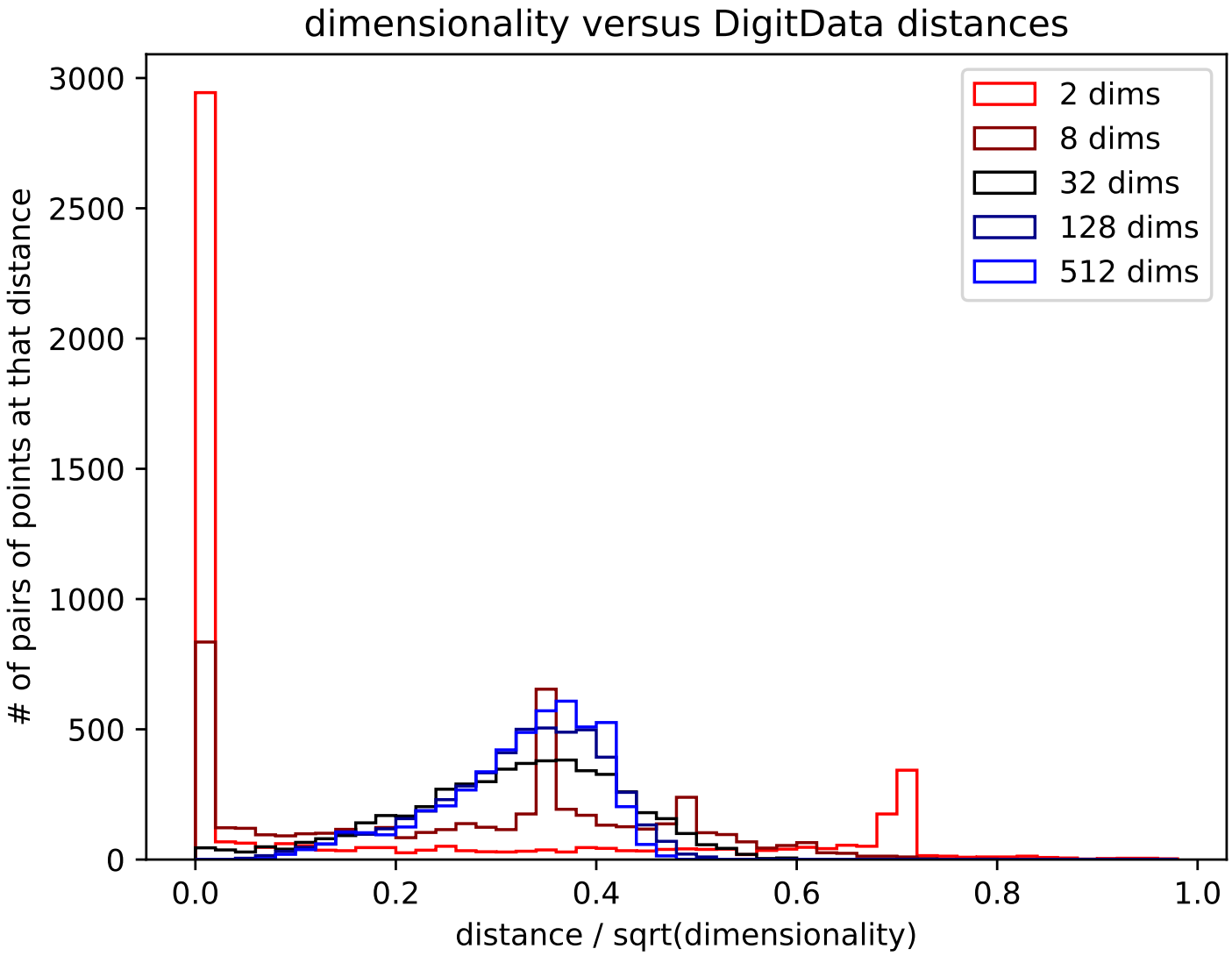
The digit data appears to have skewed and multimodal distributions. As the number of dimensions incrases, the distribution is more likely a single model dimension.

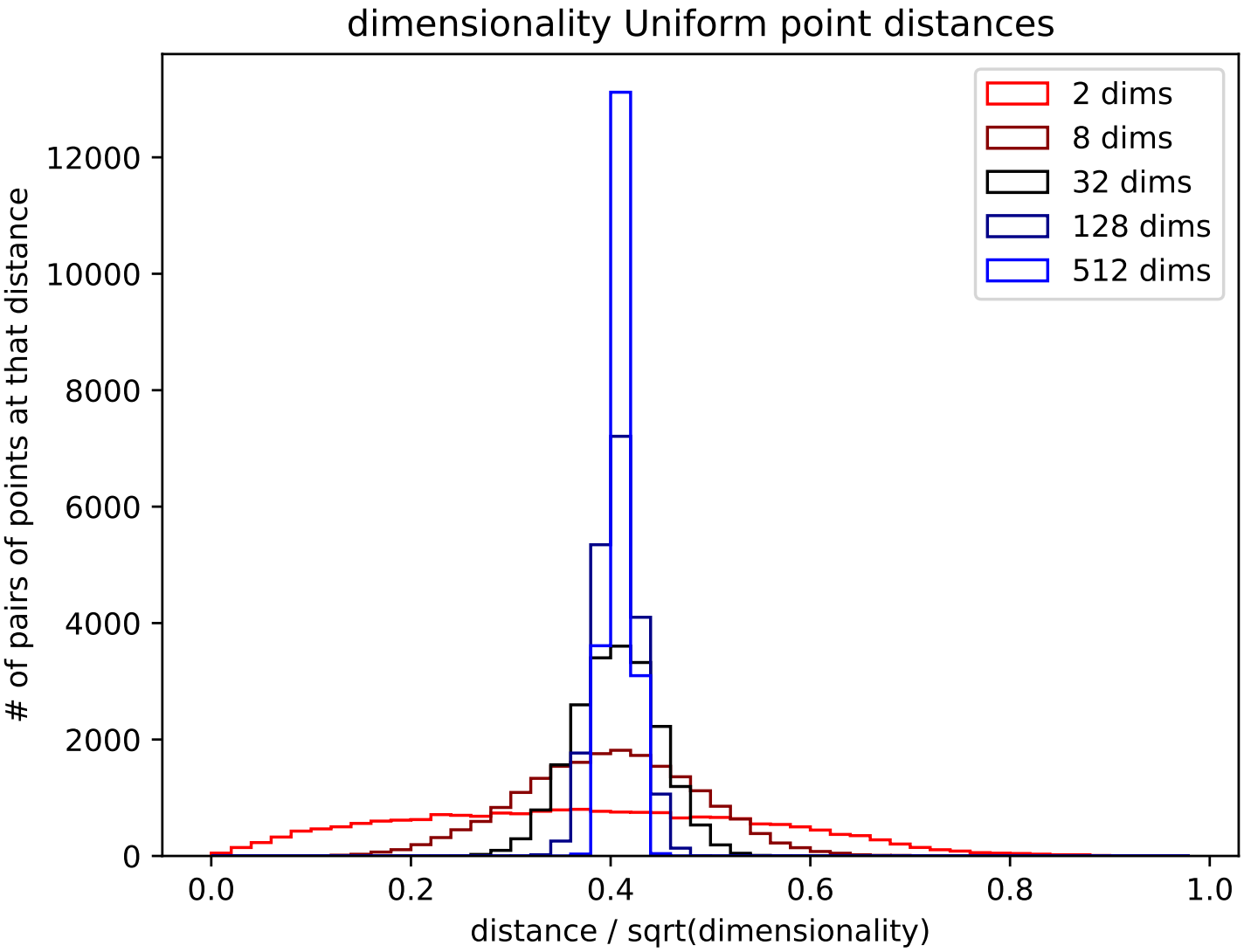
(eg.:

2-dim is bimodal at approximatly 0 and 0.7 but 512-dim is like normal distribution with skewed to left.)

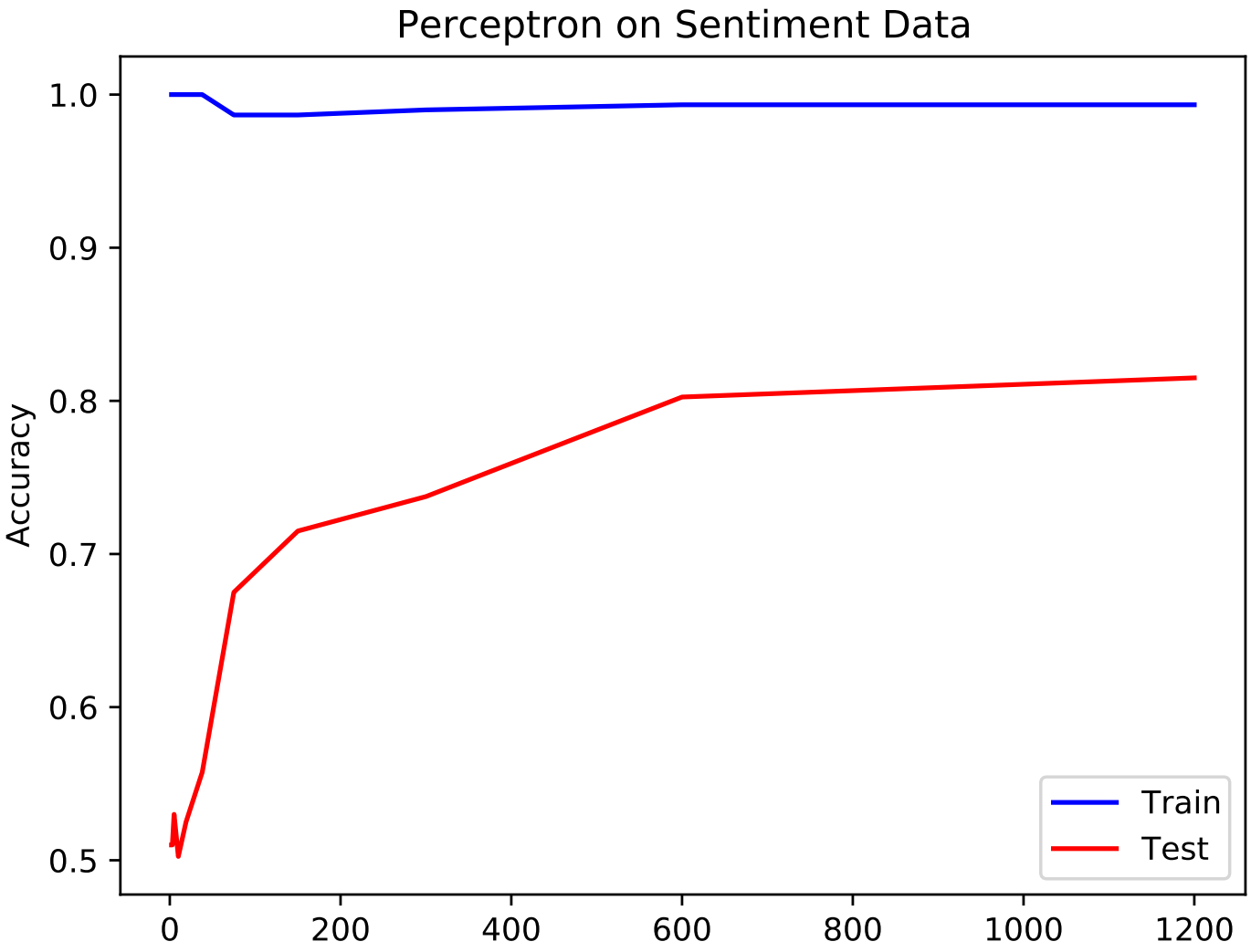

dimensionality versus uniform point distances
784 dims
# of pairs of points at that distance
0
2000
4000
6000
8000
10000
12000
14000
0.0
0.2
0.4
0.6
0.8
1.0
distance / sqrt(dimensionality)

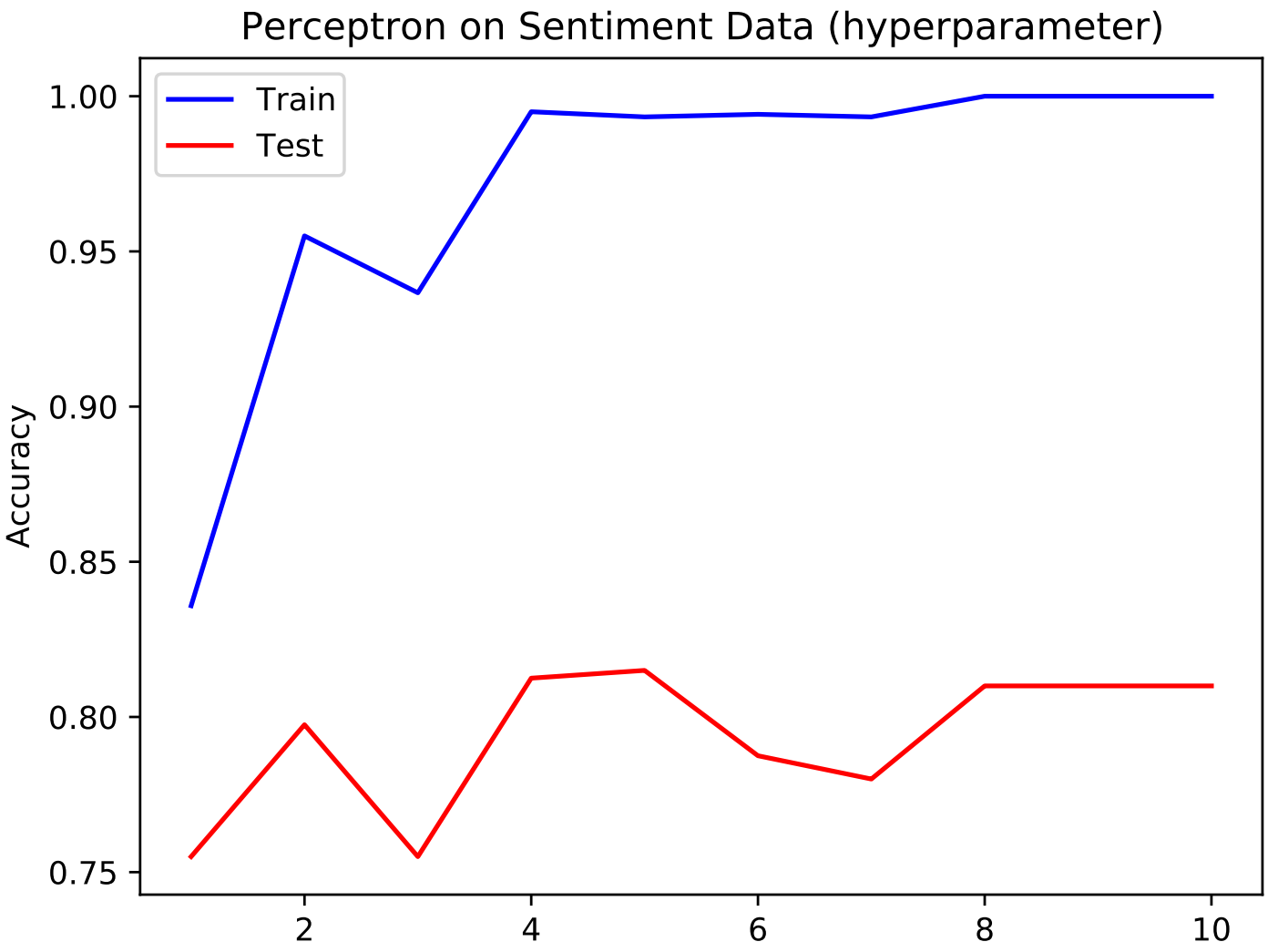

dimensionality versus uniform point distances
91 dims
# of pairs of points at that distance
0
1000
2000
3000
4000
5000
6000
0.0
0.2
0.4
0.6
0.8
1.0
distance / sqrt(dimensionality)


dimensionality versus DigitData distances
2 dims
8 dims
32 dims
128 dims
512 dims
# of pairs of points at that distance
3000
2500
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
distance / sqrt(dimensionality)


dimensionality Uniform point distances
2 dims
8 dims
32 dims
128 dims
512 dims
# of pairs of points at that distance
0
2000
4000
6000
8000
10000
12000
0.0
0.2
0.4
0.6
0.8
1.0
distance / sqrt(dimensionality)

WU6:


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


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