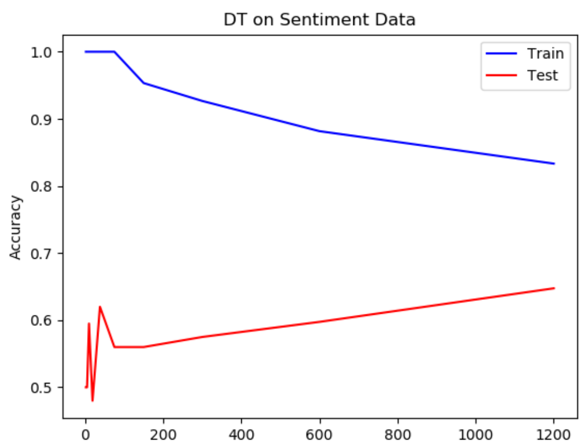
WU1:

There are only two possible categories: -1 and 1, so “datasets.TennisData.Y>0” is equivalent to “datasets.TennisData.Y==1”, and “h.predictAll(datasets.TennisData.X)>0” is equivalent to “h.predictAll(datasets.TennisData.X)==1”. When these two are the same (both true or both false), it shows the consistency of the prediction with the real categorization, and the value would be true (1); otherwise, there is a difference, and the value would be false (0). When calculating the mean of the list of “true”s or “false”s, it would sum all the “true”s as 1s and “false”s as 0s, then divide it by the total samples. As a result, it is equivalent to divide the number of true occurrences by the number of all occurrences, which is the definition of “accuracy”.

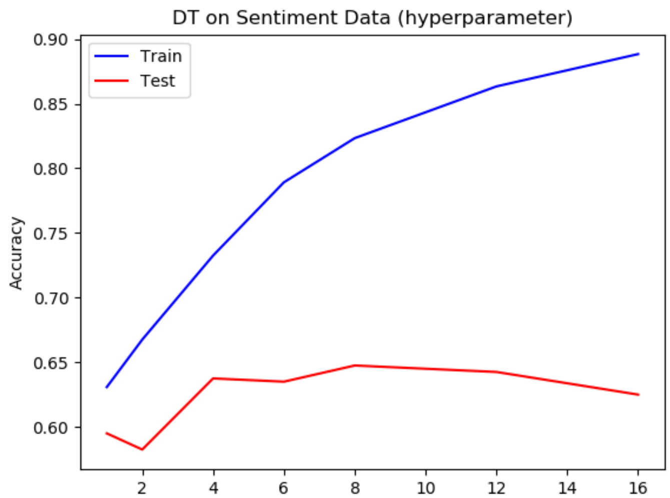
WU2:



Training accuracy tends to go down because when the training process continues, more samples will be used to train this decision tree, and more possible combinations of features can be traversed during the training process, which makes the decision tree more and more general.

Test accuracy is not monotonically increasing because in the beginning, the samples used to train the decision tree are not adequate to show the pattern, as a result, the decision tree in the beginning may not be able to reflect the real pattern of all the samples, and we may not be able to predict how the “partial” pattern would change in a short period, and it totally depends on the samples that have been used in that period.

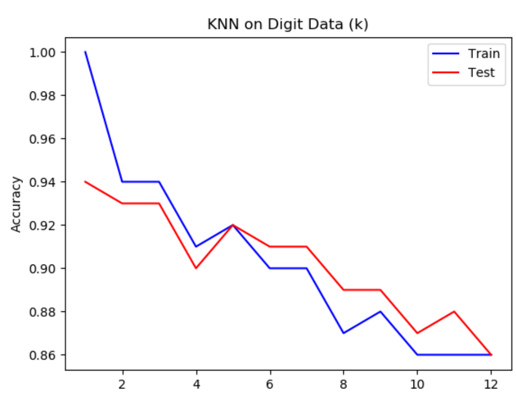
WU3:

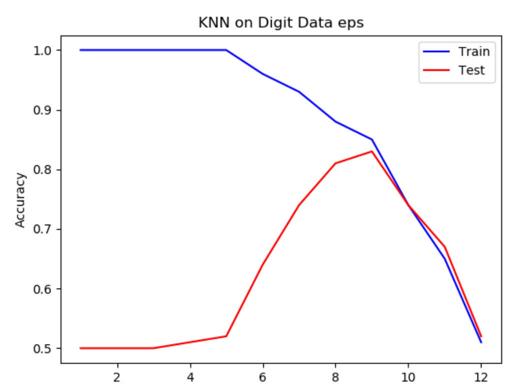


The monotonically-increasing training accuracy is guaranteed to happen because in the training process with an increasing max depth, the temporary best decision tree is either a new one which is better than all the previous one, or still the previous one, which guarantees that training accuracy cannot decrease but only increasing monotonically.

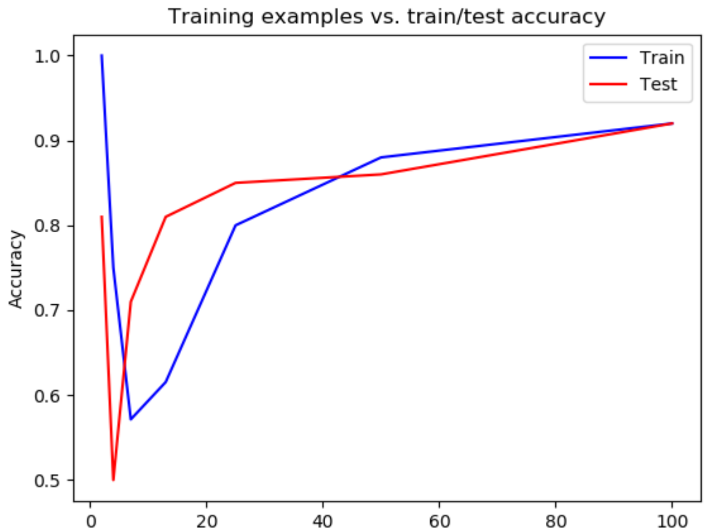
The hill-like test accuracy is just something we might expect to happen because it shows the situation of overfitting. If the training process ends earlier, the decreasing process may not be shown in the plot of test accuracy.

WU4:



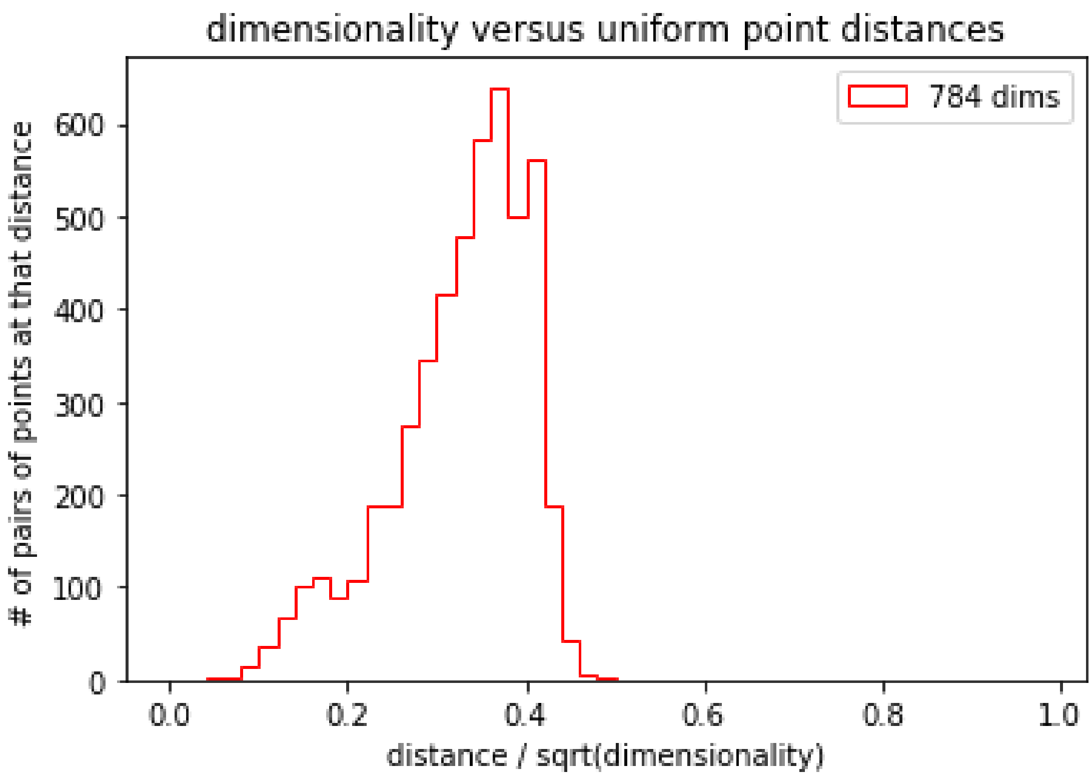


Yes, I do. There is underfitting with small K (less than 5) and epsilon (less than 10) values that train accuracy is higher than test accuracy. And there is overfitting with large K (greater than 5) and epsilon (greater than 10) values that train accuracy is lower than test accuracy.


Training examples vs. train/test accuracy
Train
Test
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100

WU5:

a.

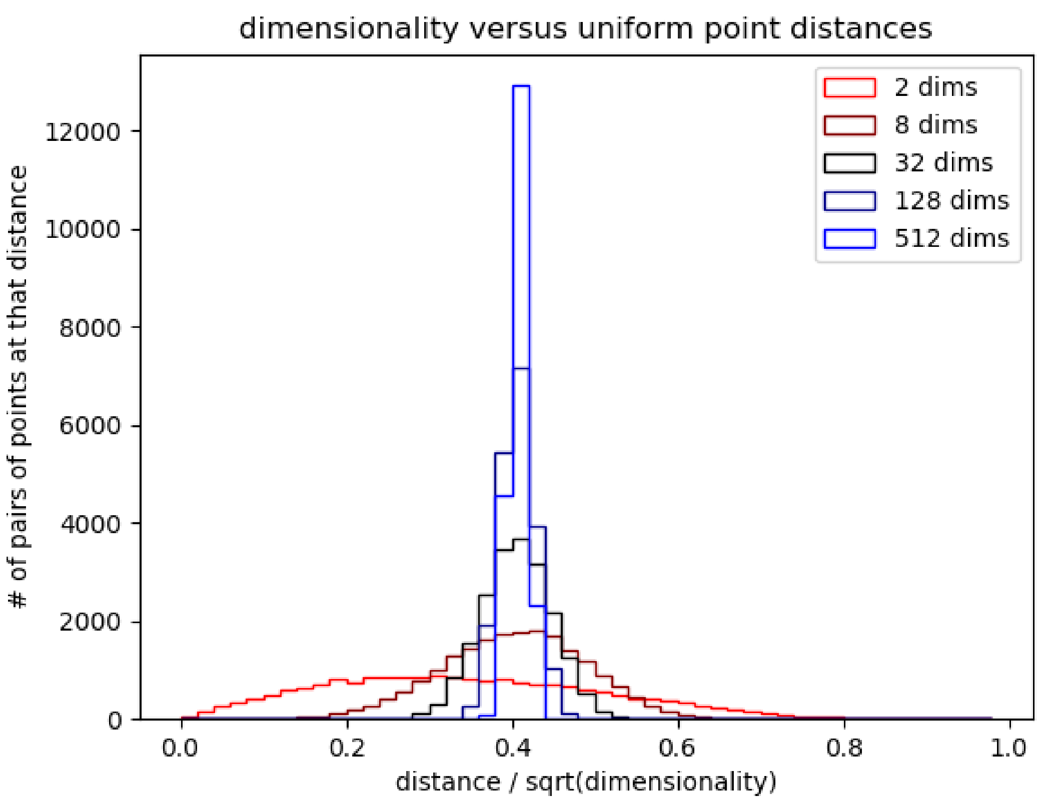

dimensionality versus uniform point distances
784 dims
# of pairs of points at that distance
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
distance / sqrt(dimensionality)

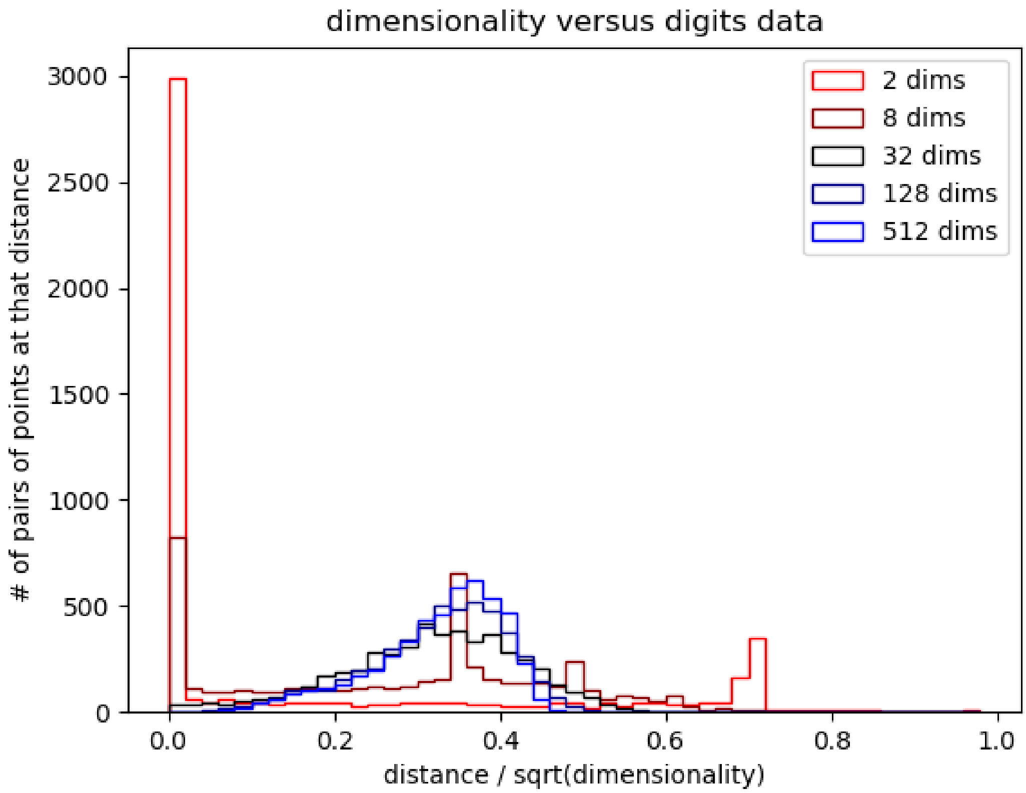
b.

```python
def cEDL(x1,x2,d):
    dist = 0.0
    for d in range(d):
        dist += (x1[d] - x2[d]) * (x1[d] - x2[d])
    return sqrt(dist)

def computeDistancesSUbdims(data, d):
    N = len(data)
    dist = []
    for n in range(N):
        for m in range(n):
            dist.append(cEDL(data[n], data[m],d) / sqrt(d))
    return dist
```

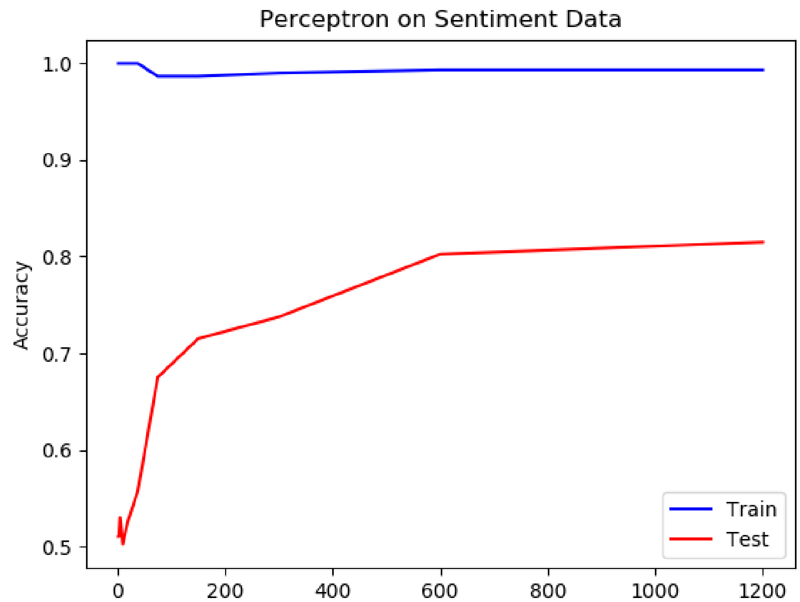
c.



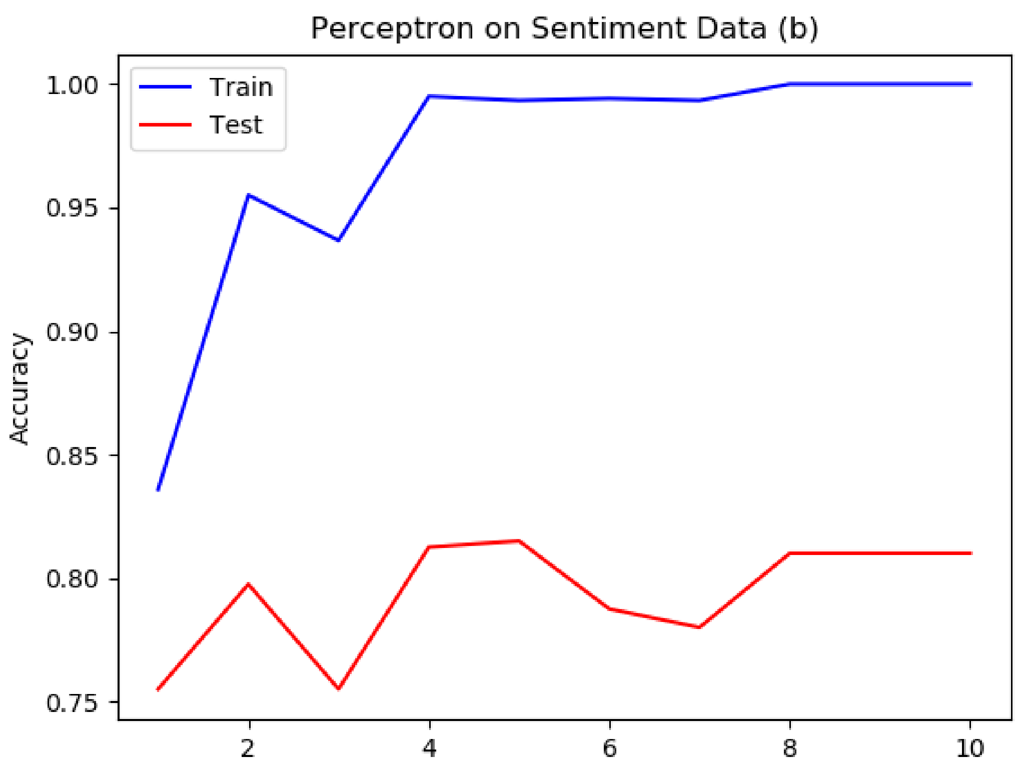


Random data is generated from a normal-distributed function between 0 and 1, the variance of distances decreases when the dimensional number increases. The other one, the digit data shows either skewed or multimodal distribution, for different number of dimensions. When the number of dimension increases, there is a change from multimodal distribution to skewed distribution, and infinitely, it may change to normal distribution I think.

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

(b)


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