Ruoxi Li
Austin Wei
Fan Yang

Write Up

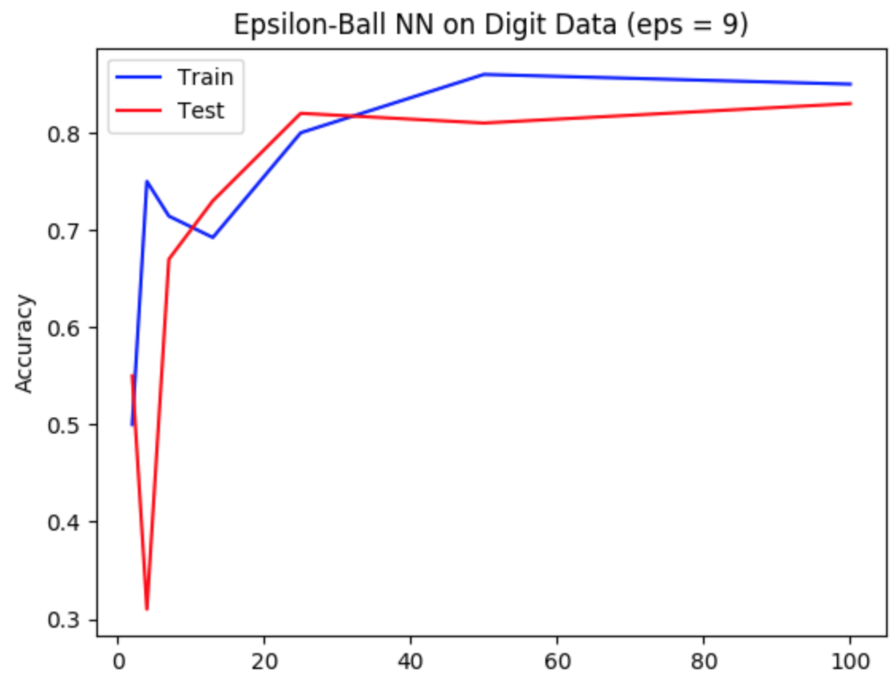
WU1:
"datasets.TennisData.Y > 0" returns a boolean array corresponding to the values of TennisData (true if the data label is greater than 0; false if the data label is less than 0). Using "==" between two boolean arrays compares them element-wise (true if the corresponding values are the same; vice versa). Then, the "mean" of the comparison indicates the mean times when the prediction is correct, therefore, equivalent to computing classification accuracy.

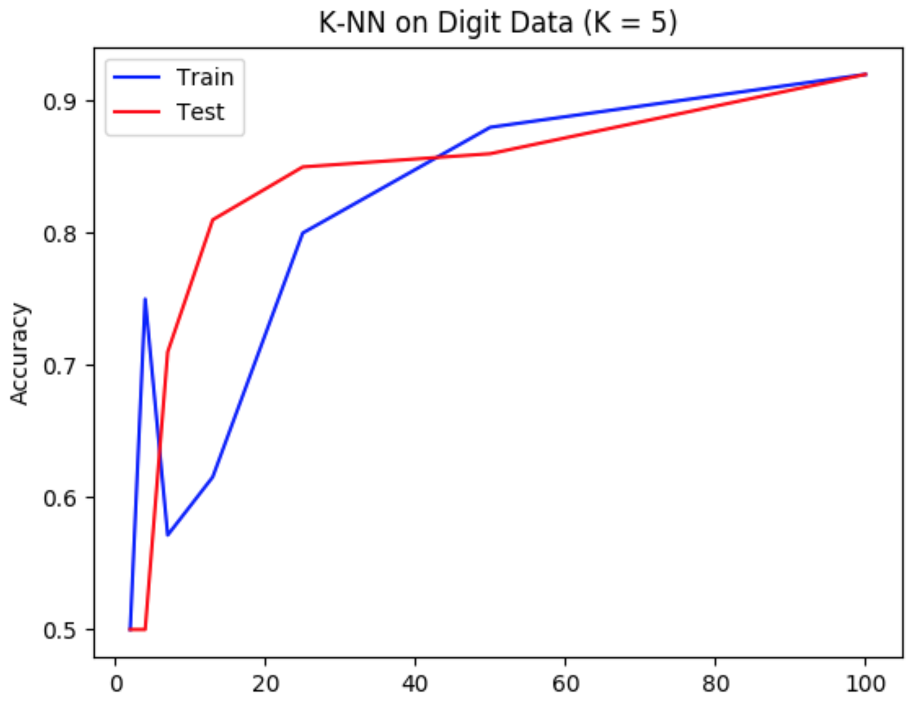
WU2:
When the training set is small, the decision tree has enough branches and leaves to simply memorize most (or all) of the information in the training data, therefore the training accuracy is high at the beginning. As the size of the training set increases, memorization becomes less feasible, and the decision tree has to choose only the features that give the most information gain to split upon. Inevitably, some information in the training data is thrown away in this selection process, and the training accuracy decreases consequently. The test accuracy roughly goes up all the way because the decision tree is underfitting at the beginning, and more knowledge are learned in the training process as the size of the training set increases. The jaggedness at the left of the curve is due to the simple memorization of the training data, which contains random noise and results in random accuracy of the learned tree.

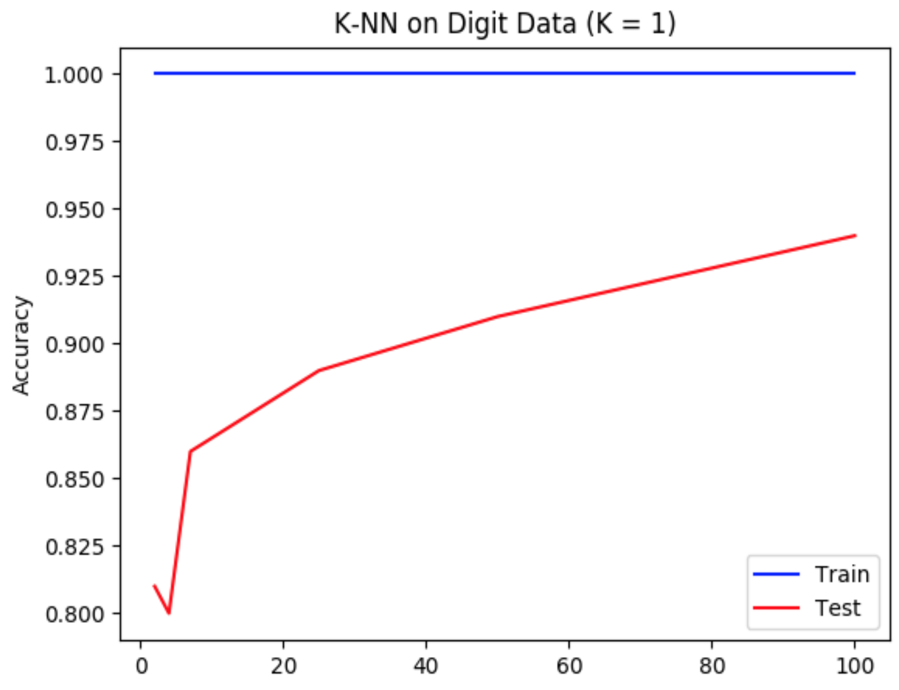
WU3:
The training accuracy monotonically increasing is guaranteed to happen, as a tree with higher maximum depth will always have a higher accuracy on the training data compared to trees with a lower maximum depth. As a tree increases in depth, the more likely it is to fit the noise in the data. As a result, the training accuracy should always monotonically increase as maximum depth increases. Thus, something that we might expect to happen is the testing accuracy making something like a hill. With testing accuracy, higher depth does not necessarily mean be higher accuracy. Deep trees can possibly let the model overfit the data, which can result in test accuracy decreasing at higher maximum depths.

WU4:
For the digits data, generate train/test curves for varying values of K and epsilon (you figure out what are good ranges, this time). Include those curves: do you see evidence of overfitting and underfitting? Next, using K=5, generate learning curves for this data.

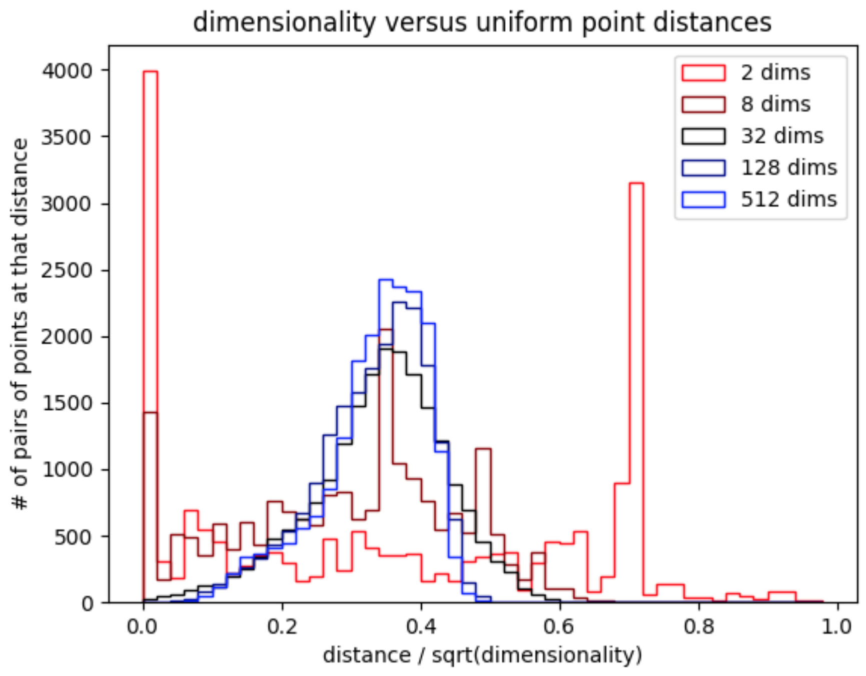
The best hyperparameters found for the digit data is K = 1 (for KNN classification) and epsilon = 9 (for epsilon-ball classification). The two corresponding curves, as well as the curve for K = 5, are shown in next page.
Overfitting is not observed for any of the three classifiers, because their test curves (roughly) never go down as the size of the training set increases. Actually, for the two KNN classifiers their test curves are still going up at the point when all training data are used. This implies that the two classifiers can perform better if we provide more training data to them. In other words, the two KNN classifiers are underfitting (although they already perform quite well).

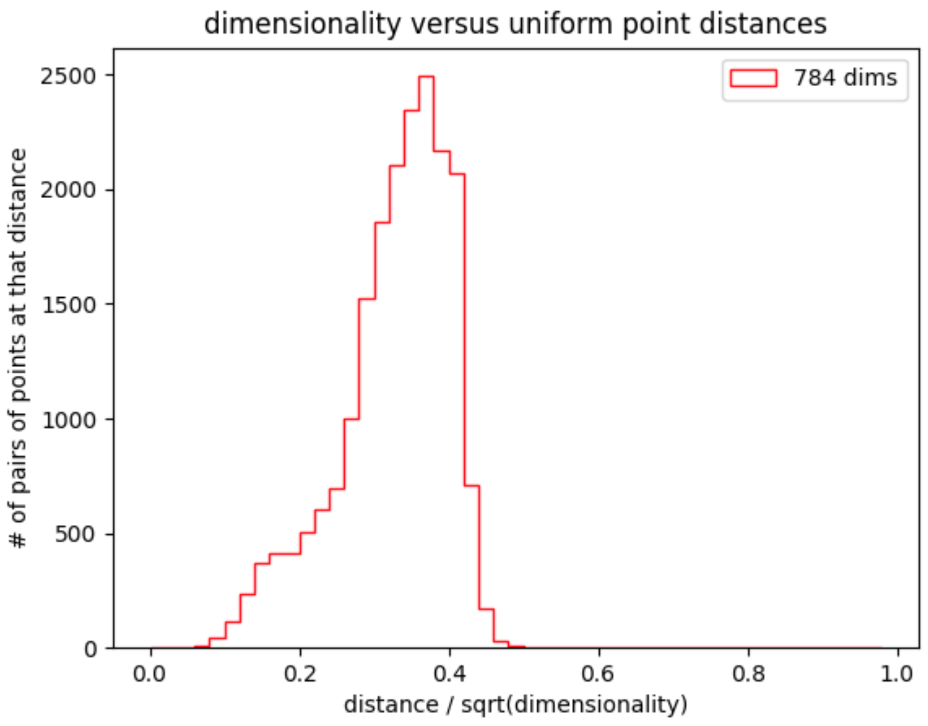




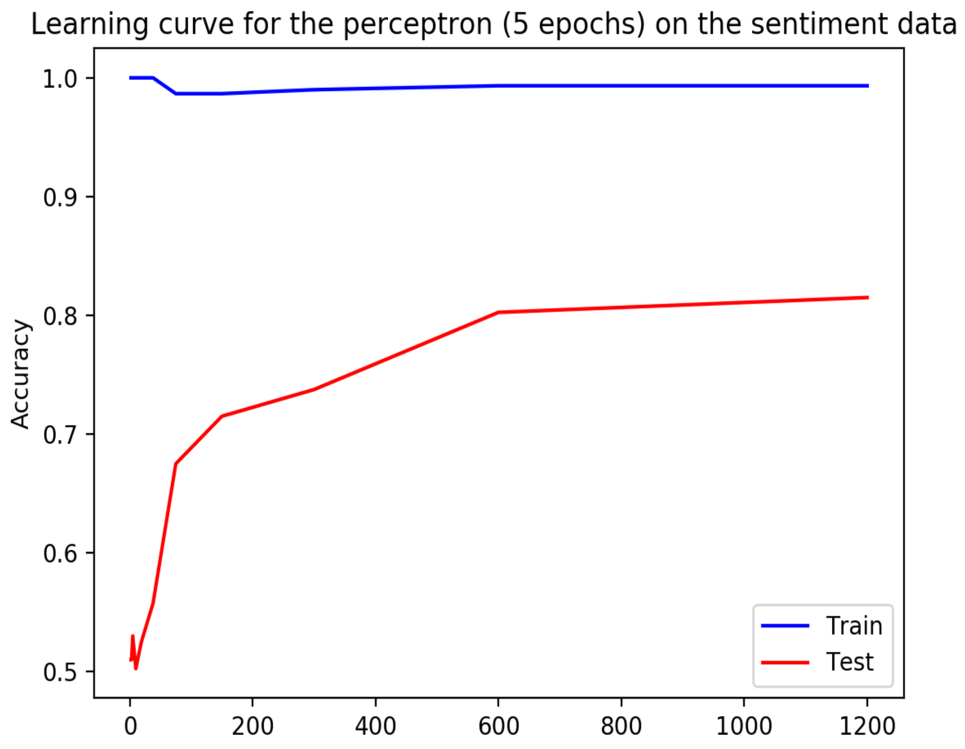


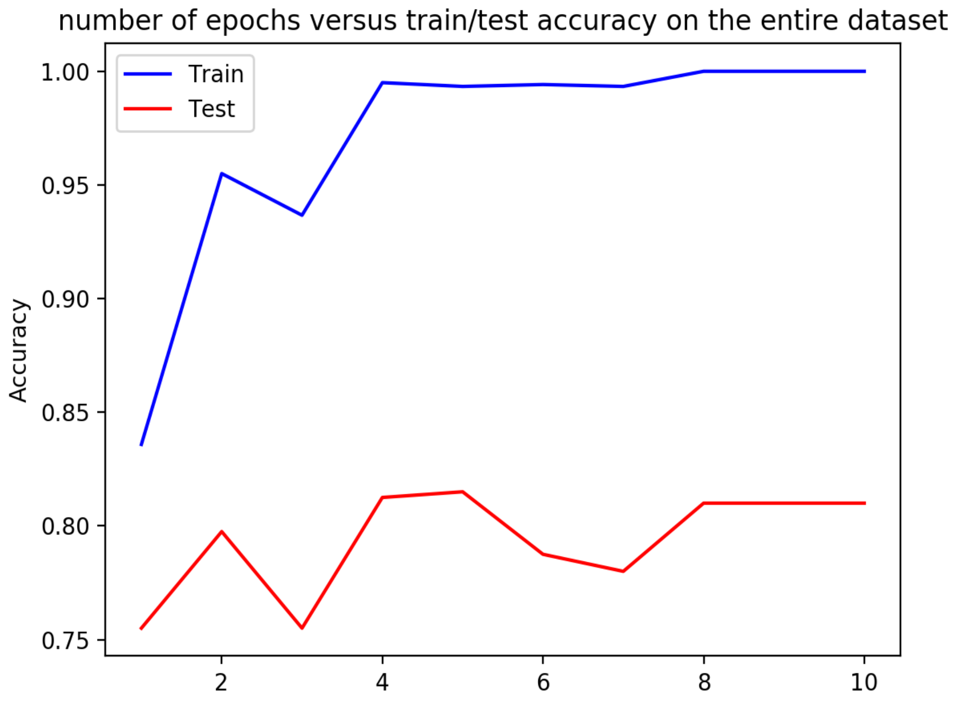
WU5:
The two plots are shown below.
It's observed in the first plot that when the dimensionality is small (e.g., when D = 2), the distances between pairs of points (normalized by square root of D) can be quite different from each other. This is an important feature needed by KNN classifiers. Unfortunately, we also observe that as the dimensionality increases, the distances between pairs of points gradually become more concentrated around the mean, same as what we found for uniformly random data points in HW03. For example, when d = 784, more than 80% of the distances fall into the region [0.2, 0.45]. This observation sadly indicates that KNN classifiers will not work well for data sets with large dimensionalities.




dimensionality versus uniform point distances
784 dims
# of pairs of points at that distance
distance / sqrt(dimensionality)
0
500
1000
1500
2000
2500
0.0
0.2
0.4
0.6
0.8
1.0

WU6:


Learning curve for the perceptron (5 epochs) on the sentiment data
Accuracy
Train
Test
0.5
0.6
0.7
0.8
0.9
1.0
0
200
400
600
800
1000
1200


number of epochs versus train/test accuracy on the entire dataset
Train
Test
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
2
4
6
8
10