k-Nearest Neighbors Algorithm Report

Ethan Beaird

February 3, 2023

1 Introduction

The k-Nearest Neighbors algorithm is an effective technique commonly used for classification problems in Machine Learning. In this report, I detail the technical aspects of two different variants of KNN and analyze the resulting data to help describe and understand the model. I wrote two different variations of kNN training:

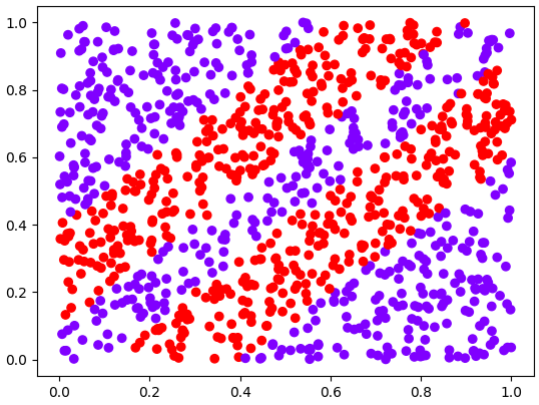
- Store All: this variant stores every training instance when training the model. Training is quite easy, but classification and storage costs are higher.
- Store Errors: this variant stores the first k instances of each class of the training data and only stores other instances if they are misclassified. Classification and storage costs are lower.

My implementation of KNN works as follows:

1. Data is imported and cleaned and fit according to the classifier's selected variant.
2. We use N-fold Cross-Validation (N=5) to partition our data into five equally-sized subsets and rotate our test data, letting the remaining four subsets be our training data in each instance.
3. We predict a label for each data instance by using Euclidean distance to find the closest k neighbor data points and taking the majority label. We then compare this label to its actual label and compute an accuracy for both our training and testing data.
4. We average our training and testing accuracy over all 5 test subsets and return those values. We repeat this process for a variety of reasonable selections of k.

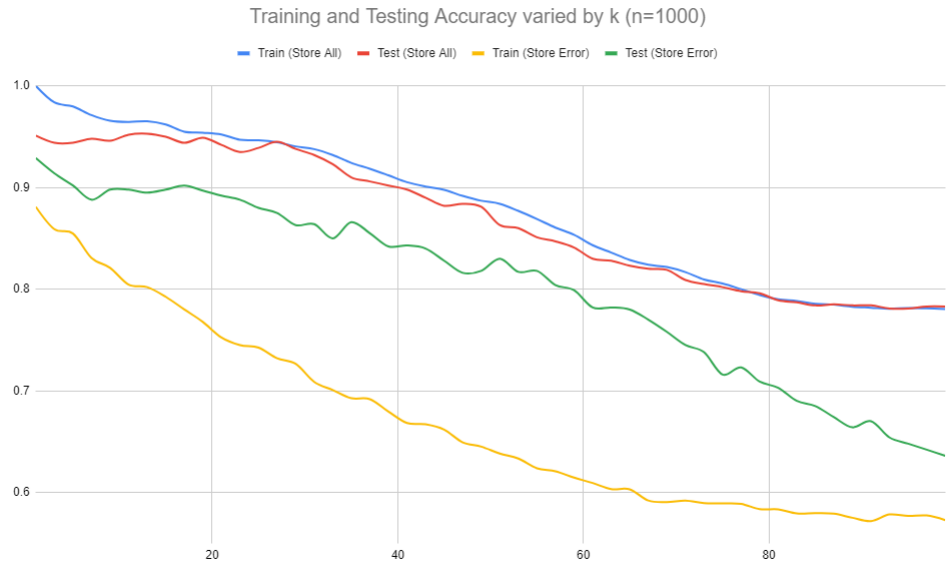
2 Data

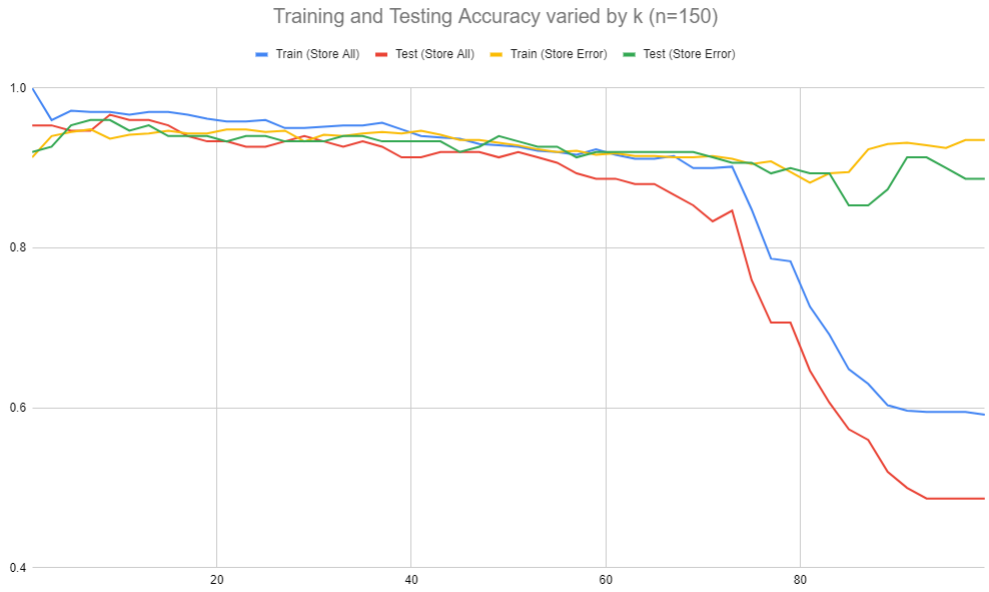
The data used in this report are a dummy set of data (n = 1000, features = 2, labels = 2) and the classic "Iris" data set (n = 150, features = 4, labels = 3) from UCI. The dummy set of data can be easily visualized and plotted since it only has 2 features, but the Iris data set is quite difficult to visualize. Below is a visualization of the dummy set of data.



3 Results

Below are the results for the dummy data set and Iris data set (respectively).





4 Analysis

I ran two sessions with each data set- one with Store All training, and one with Store Errors training. In each session, I ran the model with varying values of k to see how varying this parameter would affect the accuracy of the model.

The dummy data set results are fairly expected; the data points do not have a sporadic distribution, so it makes sense that accuracy would decrease as k increases for both the Store All and Store Errors training (since points on the edge of data would grow to compare more incorrectly labeled neighbors).

Initially, the model was extremely overfit and produced much worse results (not pictured); however, by shuffling the data before training the model, this issue was largely resolved. Still, the Store Error training accuracy is consistently lower than the Store Error testing accuracy, which

is unexpected. This is likely due to overfitting on the test set; this issue could be resolved by averaging or varying the first k stored points (which I did not do for this report).

The Store All tests of the Iris data set follow a similar story as the dummy data set. However, the Store Error data is a bit peculiar. Accuracy follows a downward trend as k increases until it spikes with k = around 60 percent of the total data set size. I can only assume this is due to how small the data set is, and this trend would likely disappear were more Irises to be gathered.

5 Conclusion

It was very interesting learning about k-Nearest Neighbors, cross-validation, and other machine learning topics during this project. It was fairly frustrating getting the hang of Python, but I ultimately found it fairly intuitive.

Considering the 800 trials of data that I have gathered, I think it is reasonable to generalize that large choices of k in the k-Nearest Neighbors produce worse accuracy than smaller choices of k. I know from research that very small choices of k produce similarly poor results, but I believe that my data sets are too small/uniform to necessarily be penalized by that.

Also, accuracy for Store All training is higher than accuracy for Store Error training, and I assume the difference between the two grows as data sets become larger. This difference is not necessarily noticeable in a relatively small data set like the Iris data set, but it is fairly evident in the dummy data set. However, training in Store All takes substantially longer (since a much smaller portion of the data is being considered).