CS 434 Implementation Assignment 3 Report

Wenbo Hou Zhi Jiang

May 01, 2017

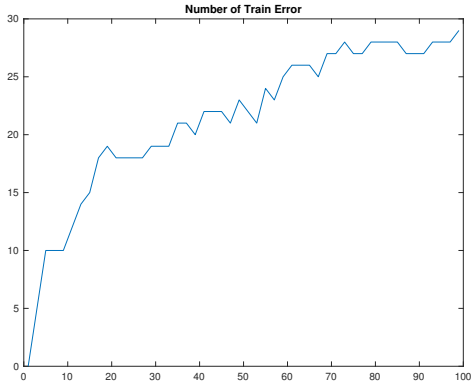
Introduction

In the first part, all programs are implemented by Matlab script. There are three Matlab scripts: `Assignment3_Part1.m`, `KNN.m`, and `k_function.m`. `Assignment3_Part1.m` is used to read file and call `k_function.m`. `KNN.m` is used to implement KNN algorithm. `k_function.m` is used to calculate each kind of error by using `KNN.m`. Please run `Assignment3_Part1.m` on Matlab.

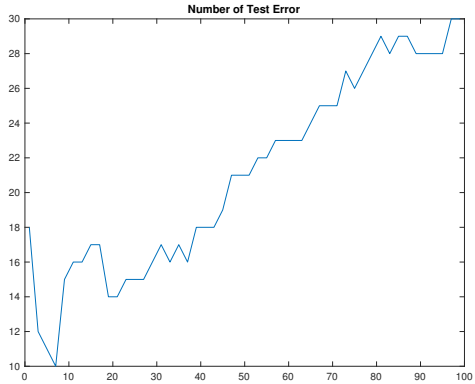
The second part is accomplished in Python in script `part2.py`. You can run this script as running other normal python scripts. It will print the decision stump and the complete decision tree in your console. Information, like error rates, information gain, and θ will be printed out, either. We collected those data and display it in this document in a pretty way.

Part I

According to these plots we gained, we found training error and testing error both have the common trend, which is the number of errors is increased as the value of k increased. As we know, while we are increasing the value of k for KNN algorithm, we are decreasing model complexity, so there exists over-fitting when the value of k is small. Look at the training error plot, the minimum number of errors occurs at $k = 1$ because its nearest neighbor in training set is always itself, and this value of k must be ignored. So we cannot use training error to select k. On the other hand, the plot of test error provides a reasonable value of k and it is 7.



(a) Number of Train Error



(b) Number of Test Error

According to formula $\varepsilon = \frac{1}{S}\sum_{i=1}^{S}\varepsilon_i$, we calculated the leave-one-out cross-validation error on training data. Then, we combined these three kinds of errors: train error, test error, and cross validation error. The following figure 2 can obviously show results of these three kinds of error. Look at cross-validation error, we can find the number of errors is minimum when $k = 1$, but in fact, this value of k must be ignored. The reason is that the value of k is too small thus the model is over-fitting. So now $k = 5$ and $k = 7$ are both reliable. We tried to compare these three different errors, we found when $k = 7$, the test error still had the minimum number of errors. Overall, we think $k = 7$ should be the best choice.

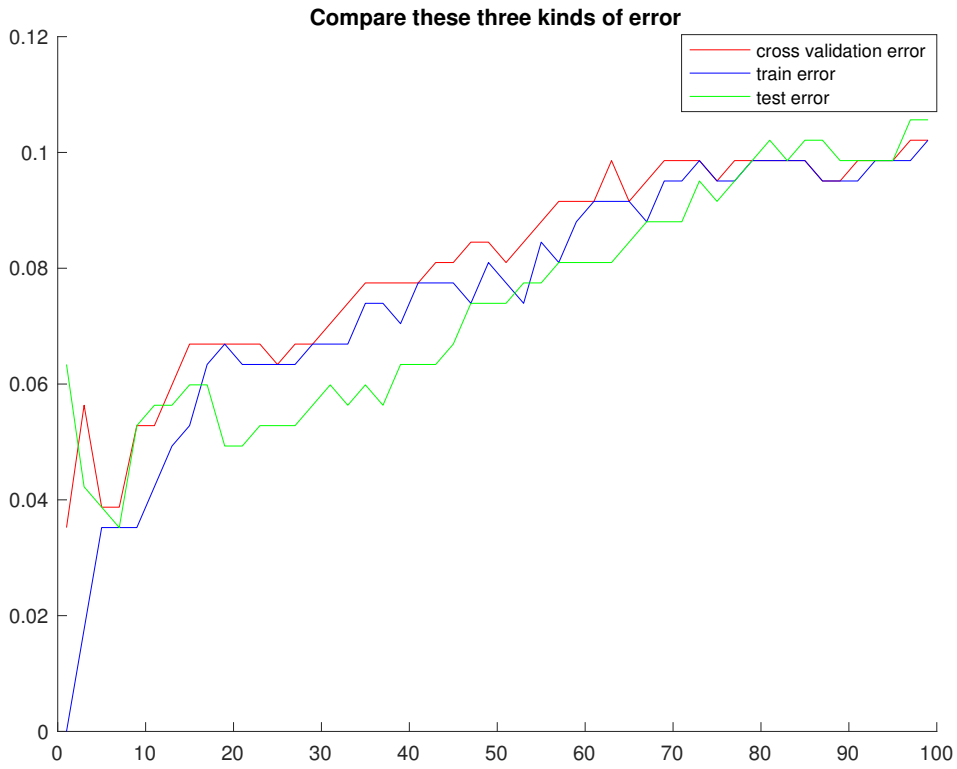


Figure 2: Leave-one-out Cross Validation Error

Part II

The second part of this assignment is to build a decision tree with the dataset from Part One. As the assignment requirement states, we first built the decision stump and classified our data with it. We got the decision stump like this. **The left side represents samples with feature value greater than θ and right side is that feature value less than θ.**

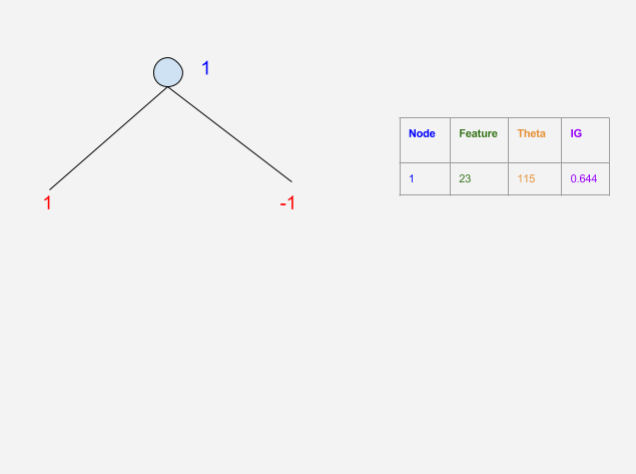
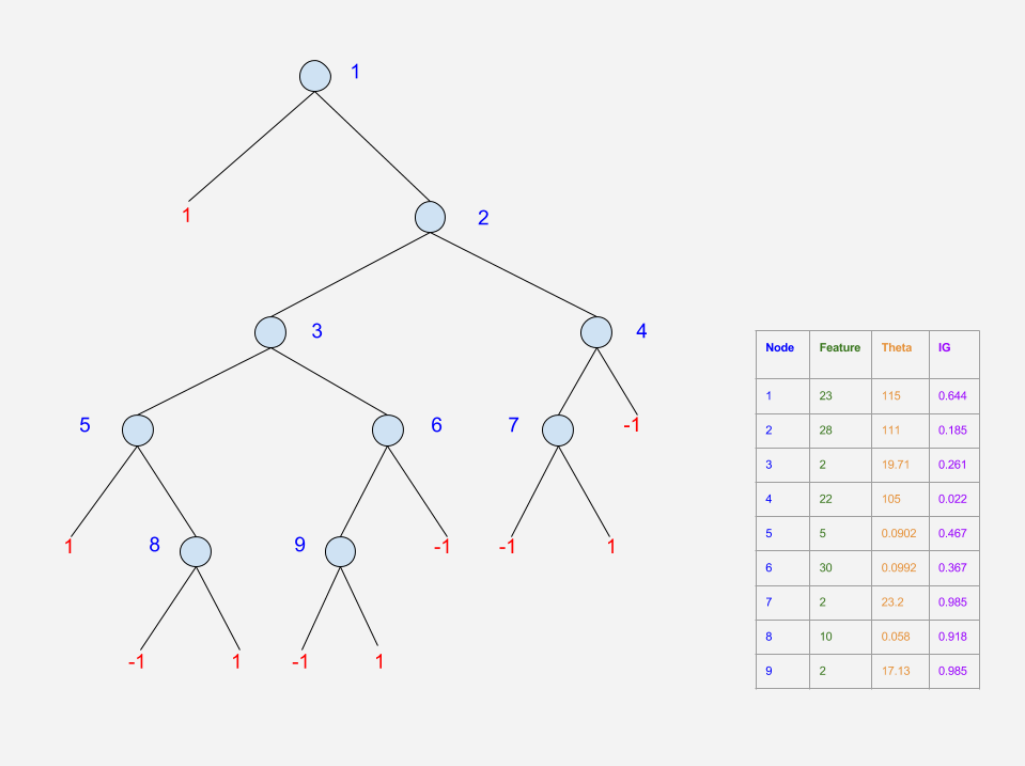


Figure 3: The Decision Stump

We used this stump to classify both the training data and the testing data. We had 6% error classification on the training data and 10.6% error classification on the testing data.

Then, we completed the decision tree and classified our data with the whole tree, again. The following figure shows the structure of the decision tree. **The node's left side stands for samples with the feature value greater than θ and right side is that feature value less than θ.**



Node	Feature	Theta	IG
1	23	115	0.644
2	28	111	0.185
3	2	19.71	0.261
4	22	105	0.022
5	5	0.0902	0.467
6	30	0.0992	0.367
7	2	23.2	0.985
8	10	0.058	0.918
9	2	17.13	0.985

Figure 4: The Decision Tree and Table containing feature, θ and Information Gain

We implemented the decision tree to classify the training data and the testing data. For the training data, we got 100% correct classification. For the testing data, we got 8.55% error classification.

After implementing the decision stump and the complete decision tree, we noticed that the root feature is the most dominating feature. This fact proves our tree-building algorithm that finds the best feature and corresponding θ. In other words, the root feature is most representative feature for classifying the whole dataset or building the classifier. As the decision tree goes deeper, more and more feature participating in classification to increase the classification accuracy. The root feature could decide the complexness of the decision tree. If the decision stump could not provide a good classification, the decision tree could be complex. Considering the situation when we do not use binary split, the decision tree could have more classify group because we have more options to test. Thus, the over-fitting occurs, and the decision tree we build might not be conclusive.

Part III

In this part, we will explore how to improve KNN with the decision tree built in part II. The decision tree tells us that samples can be classified based on different feature values. Additionally, different features have different performance. For example, we can use the 23rd feature with $\theta = 115$ to classify samples with approximately 90% correct rate. However, features not existing in the decision contribute little in classifying samples. Go back to KNN, we could increase the effect of certain feature, like 23rd feature, to group our samples better. Thus, we could have a more conclusive prediction. In this assignment, we choose to increase the weight of the 23rd feature.

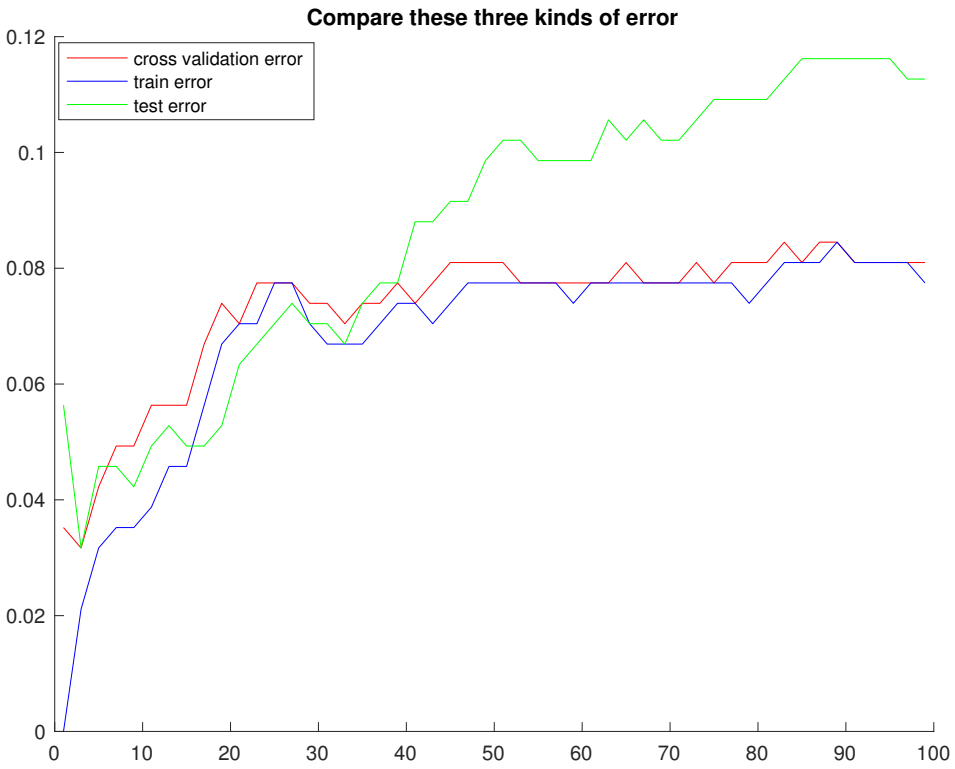


Figure 5: New Error Rates

The coefficient we set is 50, then we gained the above figure 5. In this figure, it is clear that these error rates have been decreased. Meanwhile, the value of k is also changed. The best choice of k should be 3 because the cross-validation error and test error both have the minimum error rate at $k = 3$.