Project 1 Decision Trees, CS529, Fall 2015

Matthew Letter

February 7, 2015

1 Introduction To And Overview Of The Code

The goal of this project was to implement the Decision Tree algorithm. This algorithm was used to classify validation DNA sample data, which were provided a machine learning repository. At a high level, the decision tree is built our (trained) using the provided training data and then validated using the validation data. Since we know the what the class was for each validation DNA sample an error rate can be calculated.

All the training samples have to be read as the first step of building this decision tree. A sample data structure was created for this purpose,know as dna.py. A DNA sample object was built for each input DNA sample. the dna.py data structure was as a dictionary with book keeping. That is, it kept a dictionary of each position of the DNA strand it represented as well as having update and delete feature associated with the data structure. Each DNA strand, input from the data file, had 57 base positions. In the DNA data structure's dictionary a base a given a key in the structure of Posistion- plus the position number. Doing this looking up a value at a position became a trivial task. When reading the data it was noted that the last position contained the DNA's classification. This special case value was labeled as Promoter in the dictionary.

Once the input data was parsed and objectified into dna.py pieces a list of all the dna.py was passed to the main recursive decision tree building function. The thought process behind the design of the recursive node function was find a node to split on and re-curse. Each time the decision tree function is called, it will first determine if the list associated with the current node is pure or not. In other words the program is looking for weather the list contains either all positive or all negative DNA sequences. If the program finds a pure list, it will stop the re-cursing at that point and mark the current node as a leaf node. Otherwise, it will continue classifying until the node sample list is pure or all attributes are used for classification.

The determine splitting node function was the powerhouse behind the program. this method is designed to either calculate information gain or miss-classification error for node splitting . In order to decided on what base position to split on, either the information gain or the miss-classification error was calculated. A decision was made base on one of these calculated values. With information gain the highest valued base position was chosen as the splitting node. with miss-classification error the base position with the lowest error value was chosen to split on. In both cases when a node was determined a good point for splitting a chi-square test was run on that node. The purpose of running this test was in order to determine whether the split was by chance or not. I calculated the chi- square value in order to implement split stopping and compared it with the value from the P-table for the percent confidence and dimensionality of 1. If the chi-square value was not significant, the node was skipped for the current split. After a good classification splitting node is determined, the new node is created and set up for splitting. At this point it was a recursive call to the running method for the child node until the decision tree was obtained.

Noe that the decision tree has been created the validation samples need to be classified and an error rate needs to be determined. The validation samples are used to verify the decision trees that are created by the two algorithms. Again the file parser class was used to first read all samples an objectify them into a list of dna.py (the previously used data structure). Once the sample dna was objectified a simple classification was performed on the samples using the decision tree that was built preciously. The DNA objects traverse the tree and are classified base on where, in the leaf nodes, they end up in the tree. After the classification of the validation samples an error rate was calculated for each leaf node using the fact that I know what the DNA samples should have been classified and what the decision tree classified them as. This ended the program with the printing of the accuracy.

$$total\ error = \sum_{first\ dna}^{last\ dna} error$$

2 Accuracy Results

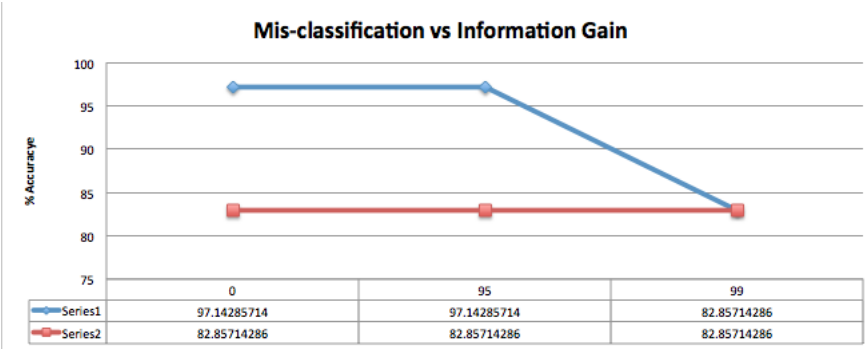

Figure 1: accuracy for 0, 95 and 99 percent confidence run with either Mis-classification or Information gain splitting procedures. Series 2(Blue) represent information gain and Series 1(Red) represents mis-classification

Figure one is the accuracy results produced after of building out the two trees at different confidence intervals. Whats amazing is that although information gain produced nearly identical results for all chi-square values the trees that were produced were radically different from one another. As expected the tree produced at 0% confidence was much larger than that of the tree produced at 99% confidence. This was true for both mis-classification and information gain trees. Base on the fact that the lower the confidence the larger the tree, it is determined that the chi-square splitting methods was proper. Furthermore, after comparing the results with other students in the class this result seemed universal.

After much thought I believe this is because base 16(starting from 0) had such a high amount of information that it would always pass the chi-square test and split at that point. We do see that at 99% confidence the mis-classification's tree is most likely under-fit and has lost a critical information node split which is causing a lower accuracy rate. The other confidence options work well because they are not under-fitting the decision tree based on the training data. We would expect to see over fit trees at lower confidence intervals but as described earlier there are a key set of base position that overcome the 0% confidence stop splitting test causing even the 0% confidence trees not to over-fit the validation data.

3 Conclusion

After tediously reviewing all the trees produced at each confidence I would conclude that there is not much that could be done in order to improve the accuracy of 97%, which is 34/35 correct classification of the validation set, produced by mis-classification error spitting. Base on my results I would choose the mis-classification error of the information gain method as it produced the best results. Based on the result I would choose a 95% confidence interval as that produced the smallest tree with the best results showing both a tree that was not over fit or under-fit. In conclusion, in order to truly determine the affect on over-fitting trees we would need more validation data points in order to determine if at the lower confidence intervals the trees were over fitting