Nialls Chavez
CS529
Project 1
ID3 Algorithm Write Up
niallsc@gmail.com

Introduction

This document provides information concerning my implementation of the ID3 Decision Tree algorithm complete with information on functions used to calculate misclassification impurity, and entropy. This document is broken up into four different sections: Code Description, Accuracy Description, Explanation of Accuracies, and Steps taken to improve those accuracies. A separate document has been provided to describe the use of the program, wherein is described details on inputting training and validation data as well as miscellaneous information on the program's execution.

Code Description: The code is broken up into 5 primary classes: ChiSquaredtest, DNA, ID3Node, Impurity, and Main.

- ChiSquaredTest:
    - This object outlines the the Chi Squared test as outlined in the information given on UNM Learn. The function on creation takes a threshold which is given by console input to the program. The threshold must be given by an integer ranging from 1 to 3. One is a threshold of 0.0, equivaltent to a ChiSquared test of 0. Two is a threshold of 0.99 which is interpreted as a threshold of 11.34. Finally, three is a threshold of 7.82. This threshold is used in the chiSqured test to determine whether or not we should keep traversing through an indivdual node or stop the traversal process. The actual ChiSquared test itself runs by the algorithm, roughly given by the following function: (expected - observed)^2/ expected.
- DNA
    - The DNA class is really just a wrapper class that takes input given from our test and validation files to organize the input in a fashion that allows us to quickly and easily access DNA strings and evaluate whether or not a given string is a promoter or not. This file also has multiple getters and setter files which allow us to set attributes about the dna sequence such as: maxGain, itsLabel, and whether or not it is a promoter. This data is used by the clas ID3Node extensively.
- ID3Node

    - This class is one of the more complicated classes created. ID3Node holds information on itself, a reference to its parent, as well as how to classify itself, and accuracy computation. This file also contains a function in which it writes the structure of the tree to XML for easy viewing. As an aside, it also contains an unfinished toString function which I began to create a way to write the tree in a different format, but ran out of time. The most important parts of this file are its ability to compute the accuracy of its predictions and classify the node as a promoter or not. Computing accuracy is done by checking the node against the actual values given in our training and validation files to see if it our predicted labeling is correct. The classification is an overloaded function with one implementation taking a single DNA object, and the second taking an arraylist of DNA objects . The first is used by the second in a for-loop which evaluates whether or not the dna strand is a promoter or not.

- Impurity
    - The impurity class contains an abstract class called calc which is implemented by two different classes: Entropy and MissFun. The first, Entropy, works by executing the function we were given in class which is, in basic terms: p log (p). The only difference here is that entropy in this calculation must be evaluated using the percentage of promoters and non-promoters distributed over the entire dataset. The misclassification algorithm is significantly more simple in this case all we have to do is determine the percentage of promoters and non promoters and return that number.

- Main
    - This class, as is self evident by its name is the main file for this program in which all control spans out from. Aside from the normal stuff like reading in the training and validation files, and the console input, this class in charge of possibly the biggest part of the program, which is building the tree. When building the tree, we pass it: our root node, which impurity function to use, its chisquared test, and a depth indicator. From there, we recursively call buildtree until the tree is created.

Accuracy Description

The accuracy seen in the Figure 1 shows the differences given by the different impurity functions and accuracy thresholds that we impose on the data. What is interesting about this data is that we can see as we allow the threshold of inaccuracy to get larger we begin to see a decrease in the accuracy of the training data, but an increase in the accuracy of validation data. This shows a crucial part of how our tests work and what the purpose of each test and its threshold is.

Figure 1

Threshold	Entropy	Misclassification
0.0	Training 1.0 Test 0.8	Training 1.0 Test 0.8285714285714286
0.99	Training 0.8591549295774648 Test 0.8571428571428571	Training 0.8873239436619719 Test 0.8
0.95	Training 0.9295774647887324 Test 0.8571428571428571	Training 0.9154929577464789 Test 0.8

Explanation of accuracies

Our entropy is calculated using Shannon Entropy and misclassification impurity which allows us to quantify the randomness in our system. A lower value implies less uncertainty and a higher value implies more uncertainty. What this means for our system is that we can use this information to make a decision on how to grow our tree. If we have a lower level of uncertainty in a given node then we can use that gain of information to know how to build our tree. What this means in our context is that, the values seen in figure 1 are representative of the how we built the decision tree to attempt to figure out whether or not a given sequence will be a promoter or not. As you can see, when we increase our threshold, we become less accurate when testing the data against itself, but overall, we see an increase in the accuracy of our test data. Interestingly enough, we can see that our best accuracies are actually flipped from each other. Our best accuracy using Entropy Impurity is when our threshold is set to .95, whereas our best accuracy for misclassification impurity is given with a threshold of 0. What this tells us is that entropy impurity needs a little bit of room to be able to make the best decisions possible, whereas misclassification impurity needs to be more rigid on its requirements.

Some work was done to attempt to increase the accuracy further. There are some code fragments that are unused in the program that attempt to implement the splitting function described in class. This functionality would have taken our training data, and split it into segments and determine the fragment with the highest information gain and use that subset as a metric against our validation data. Sadly though, I did not have enough time to finish this implementation. This method would have been powerful in making sure that we didn’t over learn from our testing data.

In conclusion, the ID3 algorithm is a very powerful tool that allows us to build fairly accurate decision trees for different types of data. This algorithm however, has been replaced by newer decision tree algorithms such as C4.5 algorithm, which allow us to gain even more accuracy than with the ID3 algorithm as well as move away from purely discrete attributes and be able to handle continuous attributes. For our case though, the ID3 algorithm performs its function well and provides us with a good introductory example of how to implement a machine learning algorithm.