# Decision Trees Over DNA Introns and Exons

Carolyn Atterbury and Keira Haskins
June 17, 2020

## 1 Introduction

In this paper we discuss an ID3 decision tree implementation which we use to classify introns, exons, and the lack of introns and exons in a sequence of 60 nucleotides. We use the nucleotide positions as attributes, which are important for determining the final classification of each length 60 string of DNA. At each position in the string, there are eight possible values that could possibly populate a given position. The attributes consist of the regular DNA nucleotide values A, T, G, C, as well as additional values D, N, S, and R, which represent certain probabilistic combinations of the original four nucleotides. For example N may be one of the four values A, T, G, and C, and S may only be either C, or G. At each level of the algorithm, we find the best possible location to split the tree by determining which location had the highest information gain were we to split at that particular location. After determining the location, we use the Chi Squared statistical test as a pruning metric to prevent over-fitting the data.

After trying various combinations of the parameters, as well as various methods for handling the special values (D, N, S, and R), our decision tree performed with an accuracy of around 73% on the testing data when using misclassification error.

## 2 Methodology

In order to generate a decision tree with maximum accuracy, we run our decision tree implementation with varying parameters. In order to find the best method for information gain, we implement Entropy, Misclassification Error, and Gini Index, and compare the results of each. To determine the best pruning metric, we use three different confidence levels for the Chi Squared test, which are $\alpha = 0$, $\alpha = 0.95$, and $\alpha = 0.99$. Additionally, we try three different methods for using the special nucleotide values (D, N, S, and R), in order to improve the accuracy of our method.

### 2.1 Calculating Information Gain

In order to calculate the information gain, we first need to get information on the impurity of the data. A dataset that is more homogeneous is considered to have lower levels of impurity. Datasets that are less homogeneous, are considered to have higher levels of impurity. The function for information gain (IG) is as follows, where D is the data and v is an value (A, T, G, C):

$$IG(D) = Impurity(D) - \sum_{v=1}^{N} P(v) \cdot Impurity(D_v) \tag{1}$$

In this equation, the information gain is essentially a measure of the impurity before splitting the data, and after the data has already been split by the attributes. High Information gain indicates that there is a big difference in impurity before and after the split. To calculate the Impurity, we choose between three different methods: Entropy, Information Gain, and Misclassification Error. These equations are below, where C is the number of outcomes, and S is the subset of data. In each equation, $P_i$ is the size of outcome $i$ over the total sameple size $|S|$. This represents the probability of each outcome, given the value that it was split over:

$$Entropy(S) = -\sum_{i=1}^{C} P_i \log(P_i) \tag{2}$$

Entropy is a measure of uncertainty in the data, and is more computationally intensive than the other options due to the logarithm.

$$GiniIndex(S) = 1 - \sum_{i=1}^{C} P_i^2 \tag{3}$$

Gini index takes each probability, squares them, and subtracts them from 1. This method has proved to have a similar effectiveness to Entropy.

$$MisclassificationError(S) = 1 - max(P_i) \tag{4}$$

Each option for impurity was tested individually and they all had varying affects on the accuracy of the decision tree. Misclassification Error is the least computationally intensive of the three options.

### 2.2 Using the special values D, N, S, R

In our first implementation of the ID3 algorithm, we ignore the special values (D, N, S, and R), and do not include them in our counts when determining the impurity (Entropy, Gini Index, or Misclassification Error) for a particular node. This means that the information gain that we calculate at each iteration of the recursive ID3 algorithm does not include any information about the special values and what they may represent.

Our second implementation of the ID3 algorithm uses randomization. When encountering a D, N, S or R value while building the tree, we select randomly from the nucleotides that the special value represents. For example, since R represents either A or G, our algorithm will randomly select between A and G if we come across an R value. The randomly selected value will then be added to the pool of nucleotide values that we use to calculate the information gain.

Our final method for incorporating the special values (D, N, S and R) in our decision tree process involves including all of the potential nucleotide values in our impurity and therefore information gain calculations. For example, when encountering an R (which represents A or G), we include both A and G in the counts that then determine the probabilities which are part of the information gain equation. This method ultimately gave us our highest accuracy score.

The special values are also included in the classification process on the test data after the decision tree is already built. When encountering a special value, we randomly select from one of the nucleotide values that it represents. After randomly selecting a nucleotide value, we would then take the branch in the decision tree that corresponds with that value. For example, if encountering an R in the test data, we will randomly select from A and G, and then take the branch that corresponds with that selection.

## 3 Code Design

- Our algorithm begins by processing the training.csv or other training data files and parses them into a list of tuples wherein each tuple contains: the index for the data, the 60 character DNA data, and lastly the classification value, IE, EI, N.
- Next our algorithm calls a recursive function **dt$_c$onstruct**. This function both determines what the next new root node is to be and builds a decision tree with fields: children, label, attr, and classification. Children is a list of nodes which are the children of the current node, label is an integer value which is the index marking the next position (0-59) in our data set which we care about, attr holds the attribute value; A, T, G, C, and classification is a value assigned to a given leaf node of IE, EI, or N (representing intron, exon, or neither). After determining the next position to split, and adding the node to the tree, **dt$_c$ontruct** splits the data according to the attributes at that position. The function is called again on each of the subsets of split data.

- When trying to determine what the new root node is, we calculate the information gain for each position, which can be computed using either the entropy, the miss classification error, or the gini index, as described above. In order to do this, we first parse the csv data and then count the number of each occurrence of each EI, IE, etc for each value of A, T, G, C and store these values for each position P in a new data structure. From there we can calculate the probability of each outcome, each value, and each outcome given a particular value. These probabilities are then used to calculate the impurity of the data at that position, which then can be used to calculate information gain. After finding the position with the highest information gain, we use the function **chi-square** to determine when to stop splitting.

# 4 Results

We test our decision tree against the testing set of data, and ultimately we have been able to achieve an accuracy of around 73 percent (0.73739). After trying three methods of handling the special values (D, N, S, and R), the third method of incrementing all the corresponding values, provides the highest overall accuracy rating. The other two methods have a similar accuracy to each other at around 61% (0.61), which is lower than the third method. The results for the third method are described in this section.

We modify our input parameters and gather data while running $\alpha = 0$, $\alpha = 0.95$, and $\alpha = 0.99$ confidence intervals for our Chi Squared test. For each of these we also submit data for the three impurity measures, entropy, misclassification error, and Gini index, when computing the information gain. The results of each combination of parameters is below:

| * | $\alpha = 0.95$ | $\alpha = 0.99$ | $\alpha = 0$ |
|---|---|---|---|
| Entropy | 0.62184 | 0.62815 | 0.67016 |
| Gini Index | 0.62394 | 0.63025 | 0.67226 |
| Misclassification Error | 0.72689 | 0.73739 | 0.61974 |

Overall misclassification error provides the highest accuracy out of the three options, but only with high values of $\alpha$. When $\alpha = 0$, misclassification error has the lowest option of the three different impurity measures, which means that is suffered from over-fitting.

# 5 Conclusion and Discussion

In this paper we present an implementation of the ID3 algorithm for generating decision trees wherein we were ultimately able to achieve a high accuracy of 0.73739 when using misclassification error for $\alpha = 0.99$. At a high-level, our algorithm implements entropy, Gini index, and misclassification error in order to compute the information gained when running our algorithm.

Overall we tried to improve the accuracy of our model by implementing three different techniques when handling the values of N, D, S, and R in both our training set as well when we were classifying the new set of data: First we simply omit them entirely, second, we implement a technique for random selection wherein we simply choose a random possible value of A, T, G, or C based upon the rules for the given nucleotide. Lastly, we simply increment all the possible values for a given placeholder nucleotide and go from there. When we traverse a new data set along with the tree that we produced beforehand we simply choose one of the possible branches at random and continue traversing the tree. As mentioned before, we were able to increase our accuracy from about 0.61 to 0.73 by ultimately choosing to use the last of these three techniques.

We overall found that misclassification error gives us a higher accuracy for both 95 and 99 percent confidence intervals over both entropy and Gini index, we suspect that that we may be overfitting our data with both entropy and Gini index, because they have higher accuracy at lower confidence intervals. We think that misclassification error is the least specific method for calculating impurity, because it is only taking a max of the probabilities. This may overall result in less overfitting.