DNA Classification using Decision Trees

Sandeep Reddy Komatireddy (101950696)

Smita Sanjay Deore (101925234)

1 Introduction

ID3 (Iterative Dichotomizer 3) is an algorithm that uses a top-down greedy approach to build a decision tree from a dataset. This algorithm uses the top-down approach i.e., it starts from the root node and at each iteration chooses the best feature to create a node. Decision trees are said to be one of the most preferable methods for representing classifiers. This algorithm learns from the training data and classifies the unknown data.

2 Problem Definition

Splice junctions are points on a DNA sequence at which 'superfluous' DNA is removed during the process of protein creation in higher organisms. The problem posed in this dataset is to recognize, given a sequence of DNA, the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns (the parts of the DNA sequence that are spliced out). In the biological community, IE borders are referred to as "acceptors" while EI borders are referred to as "donors".[1]

This problem consists of two subtasks: recognizing exon/intron boundaries (EI sites) and recognizing intron/exon boundaries (IE sites). We aim to predict if the element from given a position in the middle of a window 60 DNA sequence is:

1. "intron -> exon" boundary (IE).
2. "exon -> intron" boundary (EI).
3. neither (N)

[1] https://www.kaggle.com/c/cs529project1-2022/overview/description

3 Data Set

Dataset consists of training and validation data sets. Training dataset consists of 61 attributes which consist one of 'A',' G',' C' and 'T' and class to which the DNS sequence belongs i.e. Neither(N), exon/intron boundaries (EI) referred to as donors and intron/exon boundaries (IE) referred as acceptors. The training dataset has 2000 samples including 494 donors, 468 acceptors, and 1038 neither.

The test data set consists of 60 attributes which consist of 'A',' G',' C', and 'T'. Other characters indicate ambiguity among the standard characters according to the following table:

D	A or G or T
N	A or G or C or T
S	C or G
R	A or G

4 Feature Selection

The main choice in ID3 is feature selection i.e., selecting which attribute is best to test at each node of the tree, For the selection of the best attribute we define a statistical property, called **Information Gain** that measures how well a given attribute separates the training examples according to their target classification.

4.1 Information Gain

on gain as follows:

$$Gain(D, A) = Impurity(D) - \sum_{v \in values(A)} \frac{Dv}{D} | Impurity(Dv) |$$

$Impurity(Dv)|$

$values(A)$

where

D: *dataset*,

A: *attribute*,

v: *values of attribute A*,

Dv: *Dataset of value v*.

Figure 1: Information Gain

The impurity calculation can be done using 3 methods. Each of them is explained below.

4.2 Miss-classification Error

The Miss-classification error for each attribute can be calculated by:

$$Error\ (t) = 1 - max(P(j \mid t))$$

where

$P(j \mid t)$: *is probability of each class*

Figure 2: Miss-classification Error

4.3 Entropy

Entropy is one of the other impurity measures used for calculating information gain. Entropy is calculated by:

$$Entropy = -\sum_{v \in C} P(x=v)\ \log P(x=v)$$

where

v: *value of class*

C: *class*

Figure 3: Entropy

4.4 Gini Index

Gini Index is also one of the methods we have used to calculate impurity. It is given by:

$$Gini\ Index = 1 - \sum_{v \in C} P(x=v)^2$$

where

v: *value of the class*

C: *class*

Figure 4: Gini Index

The impurity is calculated for each attribute using any of these 3 methods and the attribute with minimum impurity is selected. If two or more attributes have the same impurity, then we will choose the attribute that comes first.

4.5 Chi-square Test

We use the chi-square test to estimate whether further expanding a node is likely to improve performance over the entire instance distribution, or only on the current sample of training data. It can be calculated by:

$$\chi^2 = \sum_i \frac{(Pi - Pi')^2}{Pi'}$$

where

Pi: real count

Pi' : observed count

Figure 5: Chi-square Test

Once we have calculated the chi-square value for that attribute, a test is performed to find if we can use this attribute to expand further or not. It is done by checking where the value falls under the critical value in the chi-square distribution table. To find the critical value we need a confidence level. Here we have used the confidence level in determining split stopping as 99%, 95%, and 0%. The degree of freedom is given as:

$$DF = (no.\ of\ classes - 1)\ (no.\ of\ labels - 1)$$

Figure 6: Degree of freedom

If the chi-square value of the attribute is less than the critical value, the split for that attribute is considered random and the node is made as a leaf node. Otherwise, the split is considered not random, and the attribute is used to split the data at that node.

5 Building the decision tree

The initial data is imported and preprocessed by separating the DNA sequence into separate attributes to make it easily iterable through the sequences. Each character of the sequence is considered as a separate attribute which generates 60 attributes of the sequence and 1 for the class. The functions necessary for calculating and generating impurity, information gain, decision tree, and prediction are defined in the code. The function **decision_tree** is the main function from where the execution of the algorithm starts. We pass the training dataset as arguments to the function. The root node of the decision tree will be the attribute with more information gain. The information gain is calculated with the help of the **information_gain** function which takes the data set and the attribute as arguments. It involves calculating the impurity of the attribute based on the value of the attribute. This impurity calculation is taken care of by another function called **impurity** which takes the method and the counts of class labels for each value of the attribute. The method name takes the value of "**gini**" for Gini Index, "**entropy**" for Entropy, or "**mis**" for miss-classification error and returns the impurity using the specified method. Then we use a dictionary to store information gain of all attributes. The attribute with more information gain is added to the tree and then selected as the root node. Now to check if this is a good attribute to expand further, a chi-square test is performed in which current data along with the attribute is passed to the **chi_square** function. Once we calculate the chi-square value for the attribute, it is checked if the calculated chi-square value is more than the critical value using the chi-square distribution table. The confidence level which we are using for stopping the split is at 99%, 95%, and 0%. If the attribute has more chi-square value than the critical value, then it is further expanded i.e., the split is continued on the attribute and a sub-data set is formed based on the values of this attribute. These sub-datasets are further passed to the decision_tree function recursively. If a data_set has all rows belonging to the same class, then the function returns that class as the node label. The resultant subtrees are added to the main tree to keep track of the path of the original decision tree. If the chi-square test fails, then the attribute is no longer used to split at that node and that node is considered as a leaf node which holds the label of the major class present in the data at that node. Once the tree is built, we predict the classes for testing data using the **prediction** function which takes the sequence of attribute values for a sample and the final tree as input arguments. Here we check if the value at the index in the given sequence exists in the values of that attribute in the tree. If it exists, then subtree for that value of the attribute is stored in **sub_tree**. Next, we check if **sub_tree** is a tree or a class label. If it is a tree, the prediction is performed recursively on it. If not, it will return the class label. If all the conditions fail, it returns the parent node class.

6 Results

The Chi-Square test is performed to train the decision tree classifier and report the accuracy on validation. Chi-Square is used during training, to stop the growth of the tree. The results obtained using different impurity calculation methods and different confidence intervals are shown below:

	Confidence Level	0.99	0.95	0.0
Impurity Method	Gini Index	0.89411	0.88235	0.88403
	Entropy	0.83865	0.85042	0.77647
	Miss-classification error	0.75126	0.71428	0.52100

Based on the above results, using Gini-Index as the impurity calculation method gave the best accuracy of 0.894 than Entropy and Mis-classification error. This is because Gini-Index is faster and is less computationally expensive than the other two impurity methods. It is observed that the accuracy of the prediction increased as the confidence level increased from 0.0 to 0.99 while using Gini Index and Miss-classification error. Generally, Entropy and Gini Index provide an almost similar level of results, but Gini Index is preferred as it is easy to compute and faster than entropy. As expected, the Miss-classification error gave the least accuracy of all because of its inefficiency in calculating the actual impurity of the data.

Improving the accuracy of prediction

From the above results, the accuracy is considered good when we use the Gini index and Entropy. It is very low when a Miss-classification error is used as the impurity method. For improving the accuracy of the classification, we can make use of an algorithm called Random Forest which is a classification algorithm containing many decision trees. Using Random Forest, we can introduce randomness in the construction of the decision trees by using bagging and boosting for feature selection. Thus, this randomized feature selection makes the Random Forest algorithm more accurate than decision trees.