Classification of DNA Sequence using ID3 Decision Tree

Mohammad R. Yousefi
Department of Computer Science
University of New Mexico
Albuquerque, New Mexico

Dheeman Saha
Department of Computer Science
University of New Mexico
Albuquerque, New Mexico

***Abstract*—DNA Sequencing is a process to determine the sequence of *nucleotides* which is composed of the elements *adenine (A)*, *guanine (G)*, *cytosine (C)*, and *thymine (T)*. This elements of the sequence are annotated as *intron*, *exon* or *neither*. The *intron* is the parts of the DNA sequence that are spliced out. Where else *exon* is the parts of the DNA sequence retained after splicing Thus, by using the ID3 Decision Tree algorithm will recognize *exon/intron* boundaries (referred to as EI sites), and recognizing intron/exon boundaries (IE sites). Initially, the Decision Tree will be trained using the training dataset using 3 different evaluation criteria. Then, the χ^2 algorithm will be used to determine the statistical difference between sub-nodes and parent node which will determine when to stop the tree from splitting. The accuracy of our approach is validated using the test dataset.**

I. Introduction

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that helps to automate the learning process by evaluating the patterns in the given dataset and aids for better decision-making processes on the unlabeled dataset. These decision-making processes make use of statistical methodologies, where the ML algorithms are trained using the labeled dataset and then the classification of the samples is done on an unlabelled dataset. One of the common ML algorithms that we used for our purpose is ID3 Decision Tree [1]. The algorithm makes use of the greedy search approach with top-down parsing and backtracking is not available. The tree is constructed by making use of *Information Gain* and *Entropy* to construct the branches.

The Decision Tree consists of some advantages that are helpful for classification problems. These sorts of trees are easy to implement and follow a logical approach to decision making. Moreover, since Decision Tree is robust to noise, it does not require any pre-processed data. However, the disadvantage of a Decision Tree is a high possibility of over-fitting. To reduce over-fitting we used pruning. The χ^2-test is used to determine whether pruning or extending a particular branch will improve the accuracy beyond the training dataset.

The dataset that we worked on composed of 60 DNA sequence elements (called "nucleotides" or "base-pairs"). The task is to determine at any given position of the sequence we need to identify boundaries. The boundaries are identified as such as 'donors' [intron to exon (IE)] or 'acceptors' [exon to intron (EI)]. If the sequence does not fall in either of the boundaries then it is classified as 'neither' [N].

The training dataset composed of 2000 DNA samples and the test dataset 1190 samples. The test samples are used to create the Decision Tree and the test samples are used to determine the classification of the Decision Tree. The Decision Tree is evaluated using the criteria Information Gain with Gini-index, Entropy, and Miss-Classification Error.

II. Design and Implementation

The Decision Tree is implemented based on the algorithm provided in the paper [1] using different evaluation criteria. In the later subsection, we will discuss the different evaluation criteria and then how those criteria are used for building up the tree. All the executable functions related to different evaluation criteria is located under the stat sub folder purity.

A. Miss-Classification Error

The Miss-Classification Error (MCE) measures the amount of samples that we have incorrectly classified with the wrong label. The formula we have used for the miss-classification purpose is:

$$MCE(DS, A) = 1 - max(p_1, p_2, ..., p_n) \tag{1}$$

where *DS* is the dataset available on the current node, *A* represents the attribute for which the *MCE* is been calculated, and p_x is the proportion of samples in *DS* such that $DS[A] = x$.

After calculating the *MCE* for each attributes (*A*) then the attribute with minimum *MCE* is selected at each node.

B. Entropy

The Entropy is the measurement of the unpredictability which is stated as:

$$Entropy(DS) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{2}$$

where p_i is the proportion of DS in class i.

C. Gini

The Gini is one of the evaluation criteria which is represented as:

$$Gini(DS) = 1 - \sum_{i=1}^{n} {p_i}^2 \tag{3}$$

where p_i is the proportion of DS in class i.

D. Information Gain

The Information Gain measures the importance of a given attribute among the other. The Information Gain is defined as:

$$Gain(DS, A) = CF(S) - \sum_{v \in value(A)} \frac{|S_v|}{|S|} CF(S_v) \tag{4}$$

where DS is the dataset having attribute A and $values(A)$ is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has $valuev$. The term CF is the different cost functions that we have stated previously.

E. Decision Tree

The Decision Tree is been built based on the algorithm shown below where we have made use of the different evaluation criteria to calculate the Information Gain. This information is used as a splitting factor for the Decision Tree.

Algorithm 1 CreateDecisionTree

List of Inputs:
Dataset
Attributes A
Class c
if *Dataset has the same Class c* **then**
  Create Leaf Node lf with Class c
  A(lf) $\leftarrow A$
  return lf
else
  Calculate MaxIG
  calcuate χ^2
  if $\chi^2 < CriticalValue$ **then**
    Max occurrence c in the Dataset
    create lf with c A(lf) $\leftarrow Areturn$(lf)
  else
    node $\leftarrow newTreeNode$
    Split Dataset on the last known position
    for every subset s in S **do**
      Select A based on MaxIG
      Child $\leftarrow CreateDecisionTree$
    return node
  end
end

F. Splitting of the Decision Tree

We have previously stated that the Decision Tree tends to overfit that means tree model performs well with training data but, performs poorly with testing data. To tackle this issue we have used the statistical idea of χ^2 which determines the splitting factor by comparing the value with the $CriticialValue$. This statistical idea helps us to determine if a split in the tree is advantageous or not. Thus in this way, no more splitting will take place and will resolve the issue of overfitting. Thus the χ^2 test figures out the statistical significance of Information Gain between the sub-nodes and parent node. The chi-square equation is calculated sum of squares of differences between observed and expected values, divided by expected frequencies of the target variables.

$$\chi^2 = \sum_{c,a} \frac{(observed_{c,f} - expected_{c,f})^2}{expected_{c,f}} \tag{5}$$

where c and f are the class and attributes of the dataset. The outcome is compared with the chi-square distribution table The Critical value depends on Degrees of Freedom (DoF), and the Confidence Interval (CI). The DoF in our case is 6 using the below formula.

$$DoF = (class - 1) * (feature - 1) \tag{6}$$

where in our case we have 3 different classes and 4 different features.

The CI we have considered are 0%, 95% and 99%. In this case if the chi^2 value is greater than the Critical Value the split will continue else the node will be considered as the leaf node.

III. Dataset Overview

The DNA Sequence data mainly consist of 4 main character (**'A', 'C', 'G', 'T'**) which stands for adenine, guanine, cytosine, and thymine. Other than that there are low-frequency characters (**'D', 'N, 'S', 'R'**). These sets of characters are regarded as noise in the sequence.

Initially, we have converted the training data to numerical representation using Pandas and Numpy. Each position i in the DNA sequence is considered an attribute with values A,C,G,T,D,N,S corresponding to numerical values 0,1,2,3,4,5,6 respectively. The processed data is represented by a 2D Numpy array where each row is a training example. The first column is the sample ID which shall be ignored in training and prediction. The last column in the training data is the label of the sample.

The data is been processed in two different way:

The handling of the data for values which are not a single 'A', 'C', 'G', 'T' will be done using the following strategies:

- Uniform: The attribute value is selected uniformly at random from the applicable 'A', 'C', 'G', 'T' value for the given symbol.
- Max: The attribute value is the maximum applicable 'A', 'C', 'G', 'T' value of the feature in the entire training set.

In both of the procedures, we did not see much change in results and that why we have decided to reflect the outcome of the 'MAX' outcome in the Result section.

IV. Implementation

The implementation of this code is done using Python and the following functions and classes are used for the implementation purpose:

- **read_file:** is used for reading the input file
- **_convert_samples:** converted the common features into numeric form
- **_convert_labels:** converting the labels into the numeric
- **convert_to_labels:** convert the numeric labels back into character form
- **mce:** This function is used to calculate the MCE as mentioned in Equation 1
- **entropy:** This function is used to calculate the entropy as mentioned in Equation 2
- **gini:** This function is used to calculate the gini as mentioned in Equation 3
- **gain:** This function is the function that is used for Gini Index as mention in Equation 4
- **chi_square:** This is the function that is used to calculate Chi-Square
- **Class Node:** This class is used to generate the Nodes. The Data Node represents the internal nodes of the tree. Each node must specify the decision Attribute and all children nodes associated with it. The Data Node cannot be a leaf node in the tree.
- **Class LeafNode:** This class is sued to generate the Leaf Node. A Class Node represents a classification by the tree. Each node must specify a class label associated with the node. A Class Node can only be a leaf in the tree.
- **build_tree:** This function is used to built the tree
- **write_output:** This function will be used to generate the final outcome and this output generated file is used for the platform Kaggle.

Other than that the folder "Predictions" stored all the output predictions which are upload into the Kaggle platform. Moreover, the "Attmpts" folder stores the outcome using different evaluation criteria and confidence interval.

Other folder that are compromised of "stats", "tree" and "data" which consists of the above implemented functions and classes. The whole can be executed using the "main.py" file by using the command "Python 3.x main.py".

V. Results

In this section, we will explore the outcome of the different evaluation criteria for different Confidence Interval. The accuracy was calculated by uploading the predicted results on Kaggle.

From the table below the best result, we have generated is with MCE with 95% Confidence Interval. In this case for each level, the MCE performed much better than other evaluation approaches. The reason for having higher accuracy with MCE is due to the classification of a feature that takes place against all other features. However, we can also see that Gini has out-performed Entropy by a big margin. Moreover, we can also see that in most cases we got much better results at 95% Confidence Interval and the results of 99% Confidence Interval did not change much. Moreover, we know that both Entropy and Gini are sensitive to changes in node probabilities, which is not in the case of the MCE. So, from the outcome of the result, we can state that using Information Gain with Gini or MCE at 95% Confidence Interval will provide the most accurate results.

	Confidence Interval		
Information Gain	0%	95%	99%
Entropy	87.42%	86.55%	85.50%
Gini	90.16%	90.34%	90.13%
MCE	89.18%	90.97%	90.55

TABLE I
ACCURACY OF THE TEST DATASET

VI. Conclusion

In conclusion, we can state that we have implemented the ID3 Decision Tree using the DNA Sequencing dataset. The overall results we have achieved are round to 91% and we are expecting this result is higher than the benchmark.

Although the results do not vary excessively by changing the Evaluation Criteria or Confidence Interval. But, there is a scope to improve the results and we would like to achieve better accuracy.

References

[1] J. R. Quinlan, "Induction of decision trees," *MACH. LEARN*, vol. 1, pp. 81–106, 1986.