A Comprehensive Study of Decision Trees to Classify DNA Sequences

MOHAMMAD ASHRAF SIDDIQUEE, University of New Mexico
HUMAYRA TASNIM, University of New Mexico

In this paper, we describe and analyze our implementation of a well- known machine learning decision tree algorithm: ID3. Here we classify DNA sequences using ID3 algorithm. The implementation of the decision tree is done by training with a given data set and using multiple evaluation criteria. Chi-square testing is used to implement the split stopping method of the tree. The accuracy of the method is tested by validating on a test data set. Variation on the evaluation criteria and confidence level in determining split stopping, has allowed us to compare the accuracies and obtain a well performed decision tree with highest accuracy. The project is a part of CS-529 Machine learning course.

ACM Reference format:
Mohammad Ashraf Siddiquee and Humayra Tasnim. 2018. A Comprehensive Study of Decision Trees to Classify DNA Sequences. 1, 1, Article 1 (September 2018), 4 pages.
https://doi.org/unassigned

AUTHOR CONTRIBUTIONS

Mohammad Ashraf Siddiquee and Humayra Tasnim both participated and worked equally for the project.

1 INTRODUCTION

Machine learning is a branch of computer science where large amount of data is analyzed using computer models. These models can identify data patterns using statistical techniques to train or learn and implement the training to make predictions about data classification. Decision tree learning is a well known machine learning method that classify instances by sorting them down the tree from the root to some leaf node that provides the classification of the instance [4]. In this method data is classified based on attributes where the learning function is represented by a decision tree. The training tree is formed by applying the ID3 algorithm [5] and is used to classify the test dataset.

The given dataset for training consists of instances of DNA sequences of length 60. Given any position in that sequence we need to identify boundaries such as 'donors' **[IE]** or 'acceptors' **[EI]**. If the sequence does not fall in either of the boundaries then it is classified as 'neither' **[N]**. There is a training data set and a testing data set. We have to create a decision tree using the training dataset that can be used for the classification of the testing dataset. The goal for the project is to implement ID3 decision tree learner using two types of evaluation criteria: information gain by gini index and information gain by entropy of the attributes. We also determined tree split stopping criteria using the chi-square test. Based on the variation of the evaluation criteria and different confidence levels, we have calculated the accuracies of the test data, compared the result and chose best accuracy as per our implementation.

There are notable advantages of decision tree. Decision trees indicate important attributes of classification clearly and is robust to noisy data. Also, it does not require much data pre-processing and learning of the disjunctive attribute values of instances are well understood. Therefore, the data set we are given is perfect for applying decision trees as classification method.

Thanks goes to Professor Trilce Estrada for teaching the Machine Learning course and Janie for assisting.

2 DESIGN AND IMPLEMENTATION OF CODE

2.1 Theory and Algorithm

The main concept of ID3 algorithm is selecting which attribute to test in each node of the tree [4]. The mathematical measure that is used to select the attribute and proceed with building the tree is information gain. To calculate information gain in every node two different statistical measures are used: Entropy and Gini Index. Based on the maximization of the information gain at each node we decide to split the tree further. Next in the section, we will describe how entropy and gini index are used to calculate information gain, how tree split stopping criteria is applied by chi-square testing and how the noise in the data is handled.

2.1.1 Information Gain using Entropy. Entropy is a well known measure in information theory. The definition of entropy calculation of a node S with c target classes is [4]:

$$Entropy(S) \equiv -\sum_{i=1}^{c} p_i \log_2 p_i \tag{1}$$

Here p_i is the proportion of S belonging to class i. Computation of the information gain on attribute A is defined as:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2}$$

Here $Values(A)$ is the set of all possible values for attribute A, and S_v, is the subset of S for which attribute A has value v.

2.1.2 Information Gain using Gini Index. For a node S in the dataset with c target classes, gini index is defined as [1]:

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

HHere p_i is the proportion of S belonging to class i. Calculation of gini index of a split with attribute A is defined as:

$$Gini(A_{\text{split}}) = \sum_{i=1}^{c} p_i Gini(F_i) \tag{4}$$

Here where p_i is the proportion of class i after split. F stand for the features. Computation of the information gain on attribute A after split is defined as:

$$Gain(S, A) = Gini(S) - Gini(A_{\text{split}}) \quad (5)$$

In our dataset, we have 60 attributes, 4 Features **(i.e. 'A', 'C', 'G', 'T')** and 3 classes. For evaluation criteria, in case of both measures, we have chosen the maximum information gain to split the tree further. Moreover, there are some other features which are considered as noise. Handling of noise is described later in this section.

Algorithm 1 CreateDecisionTree (EntryList, BreakingFeature)

Require: $EntryList \leftarrow$ a set of training data entry
Require: $BreakingFeature \leftarrow$ the feature based on which the this tree node is created
Ensure: Creates the decision tree using the $EntryList$ and statistical analysis

if All entry in $EntryList$ have same class c **then**
  Create a leaf-node lf with class c
  $breakingFeature(lf) \leftarrow breakingFeature$
  return lf
else
  $maxIG \leftarrow$ maximum $InformationGain$ based on remaining attributes
  $maxIGpos \leftarrow$ attribute position for $maxIG$
  $\chi^2 \leftarrow$ chi square value based on $MaxIGpos$
  if $\chi^2 < criticalValue$ **then**
    $c \leftarrow$ max occured class in $EntryList$
    Create a leaf-node lf with class c
    $breakingFeature(lf) \leftarrow breakingFeature$
    return lf
  else
    $node \leftarrow$ new $TreeNode$
    Split $EntryList$ into four subsets S based on the feature in $maxIGpos$
    for every subset s in S **do**
      $breakingFeature \leftarrow$ feature value in $maxIGpos$
      $child \leftarrow CreateDecisionTree(s, breakingFeature)$
    $breakingPos(node) \leftarrow maxIGpos$
    return $node$

2.1.3 Overfitting and Split Stopping. A tendency for decision tree models is to overfit the training examples [4]. When a decision tree model performs well with training data but, performs poorly with testing data as it contains new examples, then it is termed as overfitting. If we determine if a split in the tree is advantageous or not, we can stop further splitting and overcome the overfitting of the model. This is also known as pruning and we have used chi-square test for this. Chi-square test figures out if there is association between categorical variables. In our case it is telling us the statistical significance of information gain between sub nodes and parent node. We calculate sum of squares of differences between observed and expected frequencies, normalized by expected frequencies of target variables to determine chi-square value. Here is the formula:

$$\chi^2 = \sum_{c,f} \frac{(Observed_{\text{c,f}} - expected_{\text{c,f}})^2}{expected_{\text{c,f}}} \quad (6)$$

Here c and f are classes and features that are used in the dataset.

We compare the χ^2 value with the critical value from the χ^2 distribution table. Critical value depends on degrees of freedom and the confidence interval. In our case the degrees of freedom (DF) is 6 as we have 3 classes and 4 features $[DF = (c-1)(f-1)]$. We have used confidence interval 0%, 95% and 99%.

If the χ^2 value > critical value we continue to split the tree, other wise, the node is classified as a leaf node. Detailed discussion is presented in the results section.

Algorithm 2 GetTestReults (EntryList, TreeNode)

Require: $EntryList \leftarrow$ a set of test data entry
Require: $Root \leftarrow$ root node of the decision tree
Ensure: Classifies each $Entry$ in $EntryList$ based in decision tree $TreeNode$

$returnList \leftarrow$ new List of $Entry$
for every entry E in $EntryList$ **do**
  $e \leftarrow GetEntryCategory(e, TreeNode)$
  push e to $returnList$
return $returnList$

2.1.4 Noise Handling. In the DNA sequence, 4 characters **(i.e. 'A', 'C', 'G', 'T')** are considered as the 4 features for creating the decision tree. In our dataset there is very low frequency of some other characters **(i.e. 'D', 'N, 'S', 'R')** and we regard these characters as noise in the data. We filtered out these ambiguous noise by ignoring them while training the tree. While testing the dataset, if we encounter an ambiguous character, we classified it as neither class **(N)** as they could not be classified as donors or acceptors.

Our methods are summarized as algorithms. In Algorithm 1, we describe the steps how we trained the decision tree. Algorithm 2 receives the test data and based on the trained decision tree from Algorithm 1, returns the classified dataset. Algorithm 3 is used in Algorithm 2 to compute the classification procedure for each data entry in the testing dataset.

Algorithm 3 GetEntryCategory (Entry, TreeNode)

Require: $EntryList \leftarrow$ a set of training data entry
Require: $BreakingFeature \leftarrow$ the feature based on which the this tree node is created
Ensure: Creates the decision tree using the $EntryList$ and statistical analysis

if $TreeNode$ is a leaf node **then**
  $category(Entry) \leftarrow$ category of $TreeNode$
  return $Entry$
else
  $nBase \leftarrow$ feature value of $Entry$ in $TreeNode(maxIGpos)$
  for every child ch in $TreeNode$ **do**
    if $nBase$ equals to $BreakingFeature(ch)$ **then**
      return $GetEntryCategory(Entry, ch)$

2.2 Programming Language

To implement this ID3 decision tree, we used Java as our platform. Java is one of the most popular programming languages used to create both Web and desktop. It was designed for flexibility, allowing developers to write code that would run on any machine, regardless of architecture or platform. Moreover, Java has all the object oriented programming features we need to develop our projects. Also, we both are comfortable in Java as we have done many previous projects in Java before.

2.3 Overview of the code

In this project, we used Java 8 as our development platform and we used Netbeans 8.2 as our IDE. The entire code base consists of two major modules: training module and testing module. A wrapper class was developed to integrate both of the modules. There are three packages:

- Modules: The modules package includes the classes training and testing
- Data Structures: The Data Structure package includes the classes for different data structures we need to create the decision tree and to store our training and testing data.
- Utilities: This module includes all utility classes. We have utility classes for file IO, statistical analysis and mathematical analysis

In this section, we describe our code-base using two subsections. First, we describe all the dependencies we used to implement the project. Second, we describe each class, and their tasks.

2.3.1 Dependencies. To build this project, we used JDK 1.8. We also used the following libraries in the project:

- Apache-log4j-1.2.17
- Commons-lang3-3.8
- Opencsv-4.2

First two dependencies were used to create different logs (i.e. INFO, ERROR, DEBUG) in the project. The third dependency library was used to read, parse and write from csv files. All the dependency jar files are included with the project source code.

2.3.2 Classes. In this subsection, we describe each of nine classes we implemented for the project:

- ID3DecisionTree: This class is the entry point of the project. The purpose this this class is to initialize the project and integrate training and testing modules.
- Defs: This is a definition class. This class contains different running mode of the project, constants, parameters and other hard-coded values.
- CsvUtils: This is a utility class and is used to read from and write to the csv files. This class filters our any invalid inputs found in the csv files. The functions in this class also parse the csv files and data structures and acts as in interface between modules and the files.
- StatsUtil: StatsUtil is also a utility class used for statistical analysis. Different statistical calculations such a Information Gain, Gini index, Entropy are performed using functions in this class. The algorithm to calculate information gain, gini index and entropy is described in equations 2, 4, 3, and 1 respectively.
- MathUtils: This class was used to calculate mathematical function (like log base 2 calculation).
- TreeNode: TreeNode class was used to create the decision tree. Each TreeNode type object represents a node in the decision tree. TreeNode can keep track of the decision tree structure, decision, feature value, and class in the decision tree.
- Entry: Entry is another data structure class which is used to store both training and testing data entries in the project. Entry class stores ID, DNA sequence and class of each data entry.
- Id3Training: Id3Training class represents the training module. The task of this class is to create the decision tree based on the training data. To build the decision tree, this class uses a recursive function named createNode. Statistical analysis needed to build the decision tree are in the StatsUtil class. This class also uses Defs utility class to determine which of the statistical analysis and also which constant values should be used to make the decision tree. After creation of the decision tree, the root node is returned.
- Id3Testing: Id3Testing class represents the testing module. The task of this class is to test and classify testing data based on previously created decision tree. This class reads testing data from csv given file using the utility class CsvUtils and run test for each data entry on the decision tree. Each of the entry in test data gets a class assigned among EI, IE and N.

In the project, we have a file named *log4.properties* which contains patterns and format for the project logs. A log file named *ID3DecisionTree.log* will be generated after the first run. For the successive runs, project logs will be appended to the log file.

Standard JavaDoc commenting format was used throughout the code. Also, Java standard naming convention was followed. All of our experiments are reproducible, the source code and sample training and test data can be found in the GitHub repository [3].

3 RESULTS AND INTERPRETATIONS

In this section, we are going to discuss the results we have got from our procedure. There are two evaluation criteria that we used : **information gain from entropy** and **information gain from gini index**. We have also used 3 different confidence intervals for chi-square test. These variations enabled us to obtain different accuracies on the test data. The accuracy was calculated by testing these variations and getting the results from uploading in Kaggle [2]. The accuracies we obtained from different variations are summarized in Table 1 and Figure 1.

Table 1. Accuracy of test data with evaluation criteria and confidence level variation

	Confidence Interval		
Information Gain By	0%	95%	99%
Entropy	87.379%	87.815%	87.815%
Gini-Index	87.815%	88.025%	88.025%

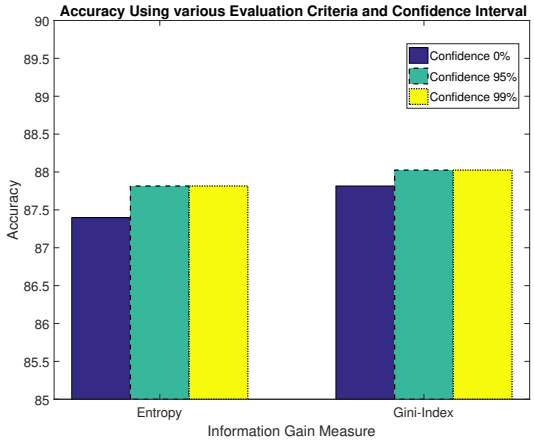

Fig. 1. The bar chart shows accuracy comparison of the test dataset using 2 evaluation criteria: information gain using entropy and gini-index. The confidence intervals 0%, 95% and 99% are shown in blue, green and yellow bars accordingly. X-axis is the evaluation criteria and y-axis represents accuracy level.

The best accuracy that we got from our testing is 88.025%. The evaluation criteria that is used here is gini-Index and we got the same accuracy for 95% and 99% confidence interval. As we have already mentioned that for calculating the critical value we have taken the degrees of freedom (DF) value 6.

$$DF = (Class - 1) \times (Feature - 1) \tag{7}$$

Here number of class is 3 **(i.e. 'EI', 'IE', 'N)** and number of feature is 4 **(i.e. 'A', 'C', 'G', 'T')**. The critical value is obtained from the χ^2 distribution table.

The accuracy that we gained using Information gain from entropy ($\approx$ 87%) is not much lower than the one we obtained from gini-index. As we have discussed in the class both gini index and entropy are similar measures in regards of this kind of calculation. Both are sensitive to changes in node probabilities. Our intuition is that the little difference that we have got is because of our method of implementation.

Also, our intuition is, there is scope of improvement in handling the noise in data. We anticipate that the accuracy would have been different, somewhat better, if we used a new class for the noisy data ans included the ambiguous characters are a feature. That would change the degree of freedom and the calculation of the chi-square test. Therefore, further development can be done in this project from this perspective.

On average, our training module took 170ms to build the decision tree from 2000 training data and our testing module took 5ms to classify 1190 test data.

4 CONCLUSIONS

In conclusion, we implemented an ID3 Decision Tree algorithm [5] in this project and experimented with different statistical modes and parameters. In the end, we are delighted to get accuracy above 88% which is over that benchmark value (80%) for this project. Our results do not vary excessively with the change in statistical analysis method or confidence interval. Therefore, this results advocate what we have learned in the class. Although there are some scopes for further pruning the algorithm, our results indicate that, our implementation of ID3 decision tree algorithm is correct and accurate. In future, we will keep working on pruning and enhancing this to achieve better accuracy and run-time.

REFERENCES

[1] 2018. Gini Calculation Example. (2018). http://dni-institute.in/blogs/gini-index-work-out-example
[2] 2018. Kaggle Leader-board. (2018). https://www.kaggle.com/c/project-1-decision-trees-cs529-2018/leaderboard
[3] 2018. Public GitHub Repository to download code, spreadsheet, datasets. (2018). https://github.com/mashrafsiddiq/decisionTree
[4] Thomas M. Mitchell. 1997. *Machine Learning* (1 ed.). McGraw-Hill, Inc., New York, NY, USA.
[5] J. R. Quinlan. 1986. Induction of Decision Trees. *Mach. Learn.* 1, 1 (March 1986), 81–106. DOI: http://dx.doi.org/10.1023/A:1022643204877