Shuang Liang

Hui Sun

CS529

1. Introduction

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute. The paths from root to leaf represent classification rules. The decision tree is one of the most commonly used classification techniques in machine learning.

In order to build a decision tree, first we need to decide which feature is used to split the data. To determine this, you will try every feature and calculate which split will give you the best outcome. Then, the dataset will be split into subsets. The subsets will then traverse down the branches of the first decision node. If the data on the branches is the same class, then you've properly classified it and don't need to continue splitting it. If the data isn't the same, then you need to repeat the splitting process on this subset. The decision on how to split this subset is done the same way as the original dataset, and you repeat this process until you've classified all the data.

This project is to determine the boundaries between exons and introns by decision tree, given by a window 60 DNA sequence elements. Each DNA sequence consist of base A,G,C,T and will be recognized as EI(intron -> exon), IE(exon -> intron) or N(neither).

2. Theoretical Background

The Information gain is implemented with Entropy, Gini-index, and Misclassification error.

1) $\text{Entropy(S)} = \sum_{i=1}^{c} -P_i \log_2 P_i$

where S represent Sample dataset, c is the number of classes and pi is the probability of choosing class i.

2) $\text{Gini(S)} = 1 - \sum_{i=1}^{k} p_i^{\,2}$

where S represent Sample dataset, c is the number of classes and pi is the probability of choosing class i.

3) $\text{Error(S)} = 1 - \max(\text{p}_\text{i})$

where S represent Sample dataset, pi is the probability of choosing class I, max(pi) is to choose the maximum Probability among all Pi.

And information gain can be defined as follow:

$$\text{Gain(S, A)} = \text{Impurity(S)} - \sum_{\text{v}\in\text{Values(A)}} \left(\frac{|\text{Sv}|}{|\text{S}|} * \text{Impurity(Sv)}\right)$$

where S is the dataset and A is a particular attribute and Sv represents the subset of dataset S that has value v for attribute A. The impurity could be entropy, Gini-index or misclassification error.

3. Program Design

1) Data structure

We need a appropriate data structure to represent the data in the .csv file so

that we can use the data in our ID3 algorithm. In our program, dictionary is used. For the training data, each entry is represented as the following format:

{‘attrs’: ’ANGTANGT’}

Because training set is only used for creating the decision tree, so there is no need to store the index for each entry.

For the testing data, each entry is represented as:

{‘index’:0, ‘attrs’:’ANGTANGT’}

Index for each entry is stored because we need to store the index to the final result file.

Also, we need an efficient data structure to represent the decision tree so that we can use the tree the classify the testing data. Dictionary is also used here. An example decision tree is as follows:

{43: {'T': 'EI', 'A': 'N', 'C': 'IE', 'G': 'N'}}

It contains the nested dictionaries. Reading left to right, the first key, ‘43’ is the index of the first attribute that was split by the create tree. The value of this tree is another dictionary. The values of the keys are the children of the ‘43’ node. The values is either a class label or another dictionary. If the value is a class label, then that child is a leaf node. If the value is another dictionary, then that child node is a decision node and the format repeat itself. In our example, the decision tree has one root node and four leaf nodes.

2) Gini index, entropy, misclassification error computation

We can use the formula described in the last chapter to calculate the gini index, entropy and classification error of a dataset.

3) Calculate Information gain

$$Gain(D,\ F)\ =\ Impurity(D)\ -\ \sum_{v\ \in\ Values}^{totalValues} \frac{D_v}{D} * Impurity(S_v)$$

We use the above function to calculate the information gain for a dataset of a particular attribute. The method is specified by the 'method' variable. Different method value means different method used.

4) ID3 algorithm

We design our ID3 algorithm according to the following pseudocode:

```
id3(data, features):
If classifications in data are all the same:
        Return classification
If features is empty:
         Return most_popular_classification
Most_important_feature = calculate best feature using Information Gain
If this Most_important_feature is not accepted by the chi-squared test:
          Then we should return the most_popular_classification
Else:
          Mark most_important_feature from features as None
          Node = most_important_feature
```

```
        For each value of Node:
            subFeatures=(features)
            Subset_data = get subset data that have matching values for this value
            If subset_data is empty:
              Return most_popular_classification(parent)
            Else:
                Add subtree to value from ID3(subset_data, subFeatures)
    Return Node
```

5) Using the decision tree for classification

After we learned the tree from our training data, we need to classify the testing data.

With that in mind, you can recursively travel the tree, comparing the values in test data to the values in the tree. If you reach a leaf node, you've made your classification and it's time to exit. The following is the pseudocode:

```
def classify(inputTree,featLabels,testVec):
    firstStr = get the first key of the decision tree
    secondDict = inputTree[firstStr]     //this is the value of the first key(also a dictionary)
```

```
    for key in secondDict.keys():    //for every value of the    root node
        if testVec[firstStr] == key:
            if  type(secondDict[key]).__name__=='dict': //if  the  type  of  the value is dictionary, need recursion here
                classLabel = classify(secondDict[key],featLabels,testVec)
            else:     classLabel = secondDict[key]      //get a final result.
    return classLabel
```

Then we can save every classification for every entry in the testing data and save them into a file.

4. Result

1) Data

The testing dataset contains 1190 instances. Each of the instances contains 60 sequential DNA nucleotide positions.

2) Performance

Method	Confidence Level	Testing Accuracy
Entropy	0.00	86.764%
	0.95	89.705%
	0.99	89.915%

Gini-index	0.00	88.235%
	0.95	90.546%
	0.99	91.176%
Misclassification error	0.00	88.865%
	0.95	91.596%
	0.99	91.806%

As can be seen from the table, all three methods can accurately predict the boundary type of the DNA sequence. In general, when the method is Entropy and the confidence level is 0 (no pruning at all), the accuracy is the lowest, which is 86.764%. When the method is Misclassification error and the confidence level is 0.99, the accuracy rate is the highest at 91.806%. It shows that the Misclassification error was able to provide a rigorous generalization to DNA classification. And for each method, with the rise of the confidence level, the prediction accuracy rate has also greatly improved. This shows that post-pruning can effectively solve the overfitting problem, and the prediction of decision tree after pruning is more accurate.

5. Discussion

1) The ID3 algorithm is an efficient algorithm in classifying the testing data, giving us the accuracy of 87%. We were successfully able to classify whether the DNA sequence was IE, EI, or N.

2) Although 87% is a only a decent accuracy, we saw significant improvement once Chi-squared early-stopping was implemented, regardless of the method

used(entropy, Gini-index, classification error) to calculate Information Gain. Also, accuracy improves as we widen our confidence level(from 95 to 99).

3) IG with classification error gave us the highest accuracy when compared with IG with entropy and IG with Gini value. However, we cannot say that IG with classification error is best among all three. Because this only applies to this training data and this testing data. Whether IG with classification error is best needs more experiments.