Decision Tree Classifiers
A Comparative Study

Nick Carey

April 21, 2014

Abstract

According to the "No Free Lunch" theorem [6] there is no best algorithm for solving a classification problem. We investigate properties of several decision tree classifier learning algorithms. Traditional decision tree learning algorithms depend on a heuristic for ordering object attributes by importance while constructing a decision tree classifier. We perform experiments on a couple different information-gain based heuristics and analyze relative performance. On most datasets, an information-gain ratio heuristic is slightly more accurate at a cost of slightly longer execution time.

1 Introduction

It is incredibly valuable for a program, without human intervention, to make a correct decision given a description of an environment or situation. With traditional imperative programming, a programmer might specify all possible situations along with the correct decision for each situation. However, in most applications this approach would be prohibitively labor-intensive and expensive. Rather, if a program could be given examples of situations paired with correct program behavior, perhaps it could learn how to make the right decision in similar future situations. This is the fundamental machine learning problem of classification; a program, or classifier, must correctly classify, or categorize, a previously unseen object based on past experiences and examples.

Classification is a very difficult task for a program to accomplish. There exist many different techniques and algorithms for classification, and entire textbooks are devoted to different classification algorithms[2]. There is no one best classification algorithm for all applications, as gaurenteed by the "No Free Lunch" theorem[6]. In fact, the best all-purpose classification programs are actually ensembles of classification algorithms where the final classification decision is typically the result of a special voting between many different classifiers running on the same dataset[3]. Yet even ensemble classifiers are not always ideal; it may require prohibitively many resources to be able to run several sub-classifiers at the same time on the same dataset. Also consider that an ensemble classifier will behave differently based on the weighted importance of each sub-classifier[5].

Since there is no one best classification algorithm[6], it is useful to compare classifier performance in different situations. With knowledge of the relative performance of classifiers we can make informed decisions on which classifier to apply to a given application. For example, some classifiers take a very long time to come up with an answer, but the classification is very accurate; other classifiers may give an answer very quickly, but can be less accurate. These properties, along with others, are important to consider when choosing a particular classifier for an application, or when choosing which sub-classifiers to consider in an ensemble classifier[5].

In this paper, we investigate properties of several different algorithms that generate decision tree classifiers. Decision trees are a common and natural method for classifying an object based on that object's attributes[4]. A decision tree takes a list of attributes belonging to an object and returns a classification for that object[4]. As indicated by its name, a decision tree follows a tree structure where tests on an object's attributes determine the path taken through the tree and the resulting object classification. Starting with the root node and at each node in the tree, a single attribute is examined. Based on the value of the examined attribute, a path to a child node is taken. By testing attributes and following a path through the tree based on the values of those attributes, eventually a leaf node is reached. Each leaf node contains a classification, and arriving at a leaf node is equivalent to deciding a classification for the object under examination.

Decision tree classifiers are appropriate for classifying a wide variety of objects. More specifically, objects whose attributes can take a small, finite amount of values can easily be represented in a decision tree, whereas objects whose attributes could take any value from a continuous spectrum are not.

This is because for a given meaningful attribute, an accurate decision tree will have a child node for each possible value that the attribute could take. If an object's attribute could take one of an infinite amount of values and each different value has an impact on the classification, a decision tree may not be a good classifier for that object. The decision tree size would quickly become intractable.

This paper compares the relative performance of decision tree classifiers generated by different decision tree learning algorithms. In Section 2 we describe decision tree classifiers and the algorithms that attempt to learn the optimal decision tree for a dataset. In Section 3 we outline our experimental set-up for comparing the different decision tree algorithms. Section 4 shows the results of the experiments run, and Section 5 is a discussion of the experimental results.

2 Learning Decision Trees

Decision trees themselves are efficient classifiers. To classify an object, all one needs to do is access the object's attributes and attribute values while traversing the decision tree. If a decision tree has one test for each object attribute, then classification typically happens after a single access to each of the object's attributes. The real work is done when constructing a decision tree classifier from a set of objects used as training data. The accuracy and inherent value of a decision tree classifier is only as good as the learning algorithm that constructed it.

A decision tree learning algorithm is given as input a set of training objects along with the correct classification for each object. The learning algorithm will then use the training objects to construct a decision tree classifier which will classify future objects similarly to the way the training objects were classified. In this section we present and describe two different algorithms for generating decision tree classifiers; in subsequent sections we compare the algorithms and their learned classifiers with several experiments.

2.1 Information-Theoretic Methods

A good decision tree is as short in depth as possible. By keeping the tree depth short, classification time is shortened and the memory requirements of the tree are kept down. Traditional methods for constructing a short

decision tree involve using a heuristic to greedily select the most polarizing attribute as a root node and recursing on subsets of the training data from there[4]. By fixing attribute values and following branches of a decision tree, we narrow down the space of possible objects. Therefore, it is important to select an attribute that divides the dataset the most as the root attribute. Russell and Norvig give the traditional decision tree learning algorithm below in Algorithm 1. Note that the attributeImportance function is the heuristic for selecting the most important attribute, the details of which are explained in the next subsection.

Algorithm 1 Traditional Decision Tree Learning Algorithm

```
function DECISIONTREELEARNING(objects, attributes, parentObjects)
returns a tree
    if objects is empty then
        return mostCommonClassification(parentObjects)
    end if
    if all objects have same classification then
        return mostCommonClassification(objects)
    end if
    if attributes is empty then
        return mostCommonClassification(objects)
    end if
    A ← argmax_{a∈attributes} attributeImportance(a, objects)
    tree ← new decision tree with A as root node attribute
    for each value v_k of A do
        exs ← {e : e ∈ examples and e.A = v_k}
        subtree ←decisionTreeLearning(exs, attributes − A, examples)
        add a branch to tree with label (A = v_k) and subtree subtree
    end for
    return tree
end function
```

Entropy and Information Gain

Information Gain is commonly used as a heuristic for selecting the most important attribute. Recall that the decision tree is constructed from the root first; attributes are greedily selected for nodes in order to narrow down

all possible objects as quickly as possible. Information Gain of an attribute is defined as the reduction in entropy resulting from fixing the attribute to a value[2]. An attribute that polarizes the classifications of objects will have a high Information Gain value, while attributes that do not seem to affect object class will have low Information Gain.

Information Gain does have a weakness. An attribute that could take one of a range of many possible values will be given a disproportionately high Information Gain value. Information Gain Ratio addresses this problem. Information Gain Ratio is the Information Gain divided by the entropy of the values the attribute could take.

3 Algorithms and Experimental Methods

The traditional decision tree learning algorithm was implemented according to the pseudocode listed in the previous Information Theoretic Methods section. This algorithm was tested with two different attribute importance heuristic functions; one function ordered attributes according to information gain while the other function ordered attributes according to information gain ratio. In the following experiments we compare the performance of the two different heuristics.

For each experiment, each dataset was divided into a training set and a testing set. The algorithms used the training set to construct thier decision trees and the testing set to report the average classification accuracy, precision, and recall of the generated decision tree. For each experimental run, the testing and training datasets were created by first randomly shuffling the data objects and placing the first fifth of the objects into the testing dataset and placing the remainder in the training dataset. By averaging each result data point over twenty experimental runs where each training and testing set are randomized selections from the entire dataset, we ensure that the results are stable and not impacted by strange training sets. The random division of objects into training and testing sets ensures minimal outlier impact.

Information Gain vs Information Gain Ratio

For this experiment we compare the classification accuracy, precision, recall, and training times of the two variants of the traditional decision tree learning algorithm on each of the datasets described in the next section. The

Information Gain variant uses the Information Gain heuristic as its attribute importance function, while the Information Gain Ratio variant uses the Information Gain Ratio heuristic as its attribute importance function. This experiment is meant to highlight the differences in performance resulting from using Information Gain and Information Gain Ratio as a measure of attribute importance when construction the decision tree.

Data Sets

Each experiment was executed on each of the datasets described below, and seperate results are reported for each dataset.

The datasets used to test the decision tree learning algorithms were taken from the UCI machine learning database[1].

Congressional Voting Data

The Congressional Voting Dataset, abbreviated to Congress in the results tables, represents the classification of congressional representatives as democrat or republican based on the voting data of each representative. This is a easy starting dataset. Its small size, low number of attributes, and low number of possible attribute values make this dataset a good starting point for testing a classifier.

MONK Problems Data

The MONK problem set is actually three seperate datasets, monk1, monk2, and monk3. This dataset comes pre-processed into training and test sets, and the randomized preprocessing applied to other datasets is not applied to the MONK problem set. This dataset is a little more challenging to classify, as it has a slightly larger size and attribute range.

Mushrooom Data

This dataset represents the classification of mushrooms as edible or poisonous based on physical attributes of the mushrooms. This dataset has a large number of attributes per object along with a wide variety of values an attribute could take. This dataset will test an efficient decision tree representation, since if an attribute could take many values then the resulting decision tree could become quite large.

Table 1: Information Gain vs Information Gain Ratio

Heuristic and Dataset	Accuracy	Precision	Recall	Training Time(sec)
InfoGain Congress	.95	.943	.953	.046
GainRatio Congress	.955	.948	.96	.048
InfoGain Monk1	.731	.76	.731	.015
GainRatio Monk1	.731	.76	.731	.015
InfoGain Monk2	.595	.53	.53	.013
GainRatio Monk2	.62	.555	.55	.014
InfoGain Monk3	.824	.875	.814	.016
GainRatio Monk3	.87	.886	.864	.02
InfoGain Mushroom	1.0	1.0	1.0	.37
GainRatio Mushroom	1.0	1.0	1.0	.45
InfoGain Splice	.831	.71	.71	.42
GainRatio Splice	.826	.69	.701	.47

Table 2: Performance statistics on two versions of the traditional decision tree learning algorithm with Information Gain and Information Gain Ratio as attribute importance heuristics. Tested on several datasets.

Splice Junction

The splice dataset contains a molecular biology problem. This dataset is unique from the others in that its objects could take one of three classifications. Furthermore, each object has sixty attributes, more than any other dataset. This dataset is included to test the limits of the decision tree learning algorithms.

4 Results

Table 1 shows the accuracy, precision, recall, and training time of the Information Gain and Information Gain Ratio variants of the traditional decision tree learning algorithm described by Algorithm 1. Both variants were executed on each dataset. Each row in the table represents the average output of twenty executions. For each execution, the training and testing set were

randomly selected from the entire dataset as described in the experimental setup section.

5 Discussion

Information Gain vs Information Gain Ratio

As shown in Table 1, the traditional decision tree learning algorithm with the Information Gain Ratio heuristic takes a slightly longer training time relative to the decision tree learning algorithm with the Information Gain attribute ordering heuristic. This is due to the higher complexity of calculating the Information Gain Ratio.

Also shown in Table 1 is a slightly better accuracy, recall, and precision given by the Information Gain Ratio heuristic, with the exception of the splice dataset. The slightly better accuracy is likely due to the fact that the Information Gain Ratio statistic provides more information about the tested attribute than Information Gain. Recall that Information Gain assigns false importance to attributes that can take one of many different possible values; Information Gain Ratio accounts for the entropy in values taken by an attribute and assigns a more appropriate importance to such attributes. The splice dataset attributes can take only one of four values[1] and do not suffer the weaknesses of the Information Gain heuristic, which is likely why we do not see a benefit to using Information Gain Ratio there.

6 Conclusions

As shown by the results, the "No Free Lunch" theorem holds. While using Information Gain Ratio as an attribute ordering heuristic we see better performance at a cost of training time compared to the Information Gain heuristic.

Of course, there are many more experiments to run that will show the relative benefits of different decision tree learning algorithms. Given more time, we would like to run and show results of experiments involving the evolutionary decision tree learning algorithm that we implemented. It would be especially interesting to see the properities and possible uses of a genetic-based algorithm. From the brief tests we ran on our genetic algorithm, we saw a vastly longer training time but the potential for more accurate

decision trees. Unfortunately, due to the unpredictability and long execution times associated with our evolutionary algorithm, we were unable to include experiments involving it in this paper.

References

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[3] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[4] Stuart J. Russell, Peter Norvig, John F. Candy, Jitendra M. Malik, and Douglas D. Edwards. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 3 edition, 1996.

[5] Alixandre Santana, Rodrigo G. F. Soares, Anne M. P. Canuto, and Marcílio Carlos Pereira de Souto. A dynamic classifier selection method to build ensembles using accuracy and diversity. In Anne M. P. Canuto, Marcílio Carlos Pereira de Souto, and Antônio C. Roque da Silva, editors, *SBRN*, pages 36–41. IEEE Computer Society, 2006.

[6] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.