Extension and Evaluation of ID3 – Decision Tree Algorithm

Anand Bahety
Department of Computer Science
University of Maryland, College Park
Email: abahety@cs.umd.edu

Abstract

Decision tree algorithms are a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. These kinds of algorithms are famous in inductive learning and have been successfully applied to a broad range of tasks. In this paper, I examine the decision tree learning algorithm – ID3 against nominal and continuous attributes extend it to handle missing value. Experiments to evaluate the performance of the algorithm with continuous valued attributes and missing attribute values reveal that ID3 does not give acceptable results for continuous valued attributes and works well in certain data sets with missing values.

1. Introduction

1.1 Decision Tree

Decision trees [1] classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some attribute of the instance and each branch descending from that node corresponds to one of the possible values for this attribute. For example figure below explains a decision tree based on attribute name outlook.

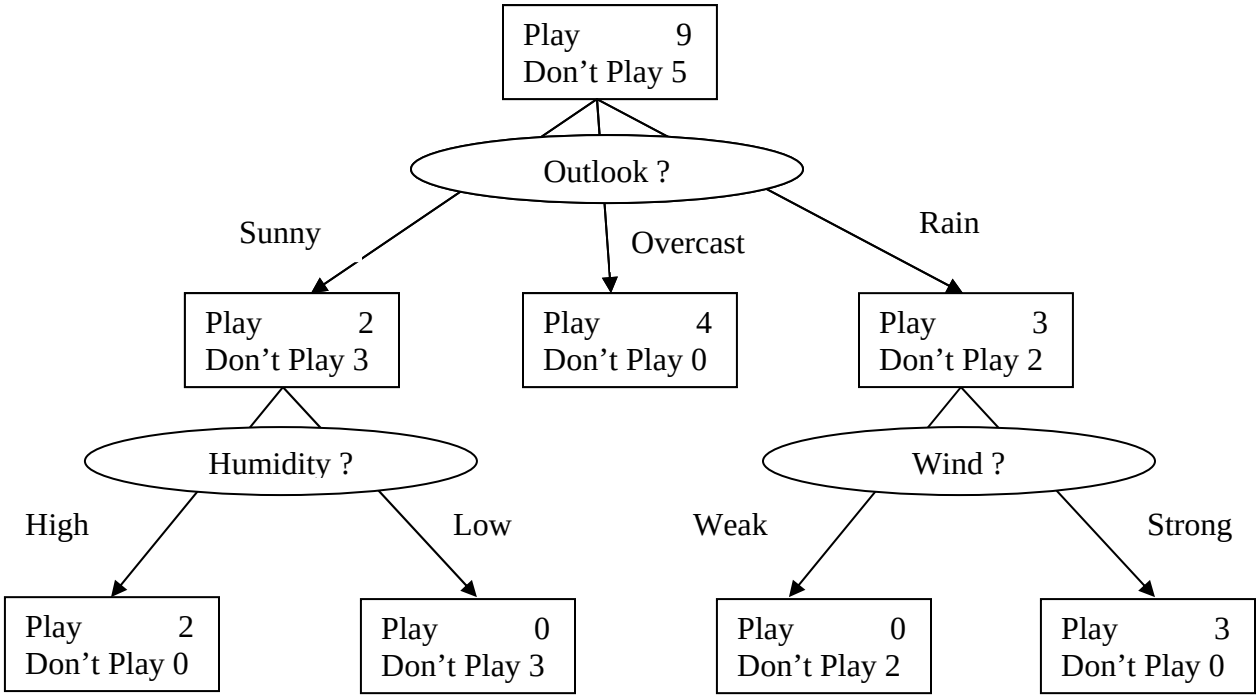

Figure 1 Decision tree

In the above figure, instance with value overcast for the attribute Outlook are positive. With other values like Sunny and Rain, it leads to sub trees. This type of task to classify examples into one of a discrete set of possible categories, are often referred to as *classification problems*.

The reasons for decision learning tree algorithms to be attractive are: -

1. They generalize in a better way for unobserved instances, once examined the attribute value pair in the training data.
2. They are efficient in computation as it is proportional to the number of training instances observed.
3. The tree interpretation gives a good understanding of how to classify instances based on attributes arranged on the basis of information they provide and makes the classification process self-evident.

There are various algorithms in this area like ID3, C4.5, ASSISTANT etc. I selected ID3 algorithm to evaluate because it builds tree based on the information (information gain) obtained from the training instances and then uses the same to classify the test data. ID3 algorithm generally uses nominal attributes for classification with no missing values. My hypothesis is that ID3 can even work well on datasets with missing attribute values to certain extent. I try to evaluate the same with first evaluating the algorithm in normal circumstances and then proceed to test my hypothesis. The basics of the algorithm are explained in brief and then implementation and evaluation part is elaborated.

1.2 Basics of ID3 Algorithm

ID3 is a simple decision learning algorithm developed by J. Ross Quinlan (1986). ID3 constructs decision tree by employing a top-down, greedy search through the given sets of training data to test each attribute at every node. It uses statistical property call *information gain* to select which attribute to test at each node in the tree. Information gain measures how well a given attribute separates the training examples according to their target classification.

1.2.1 Entropy

It is a measure in the information theory, which characterizes the impurity of an arbitrary collection of examples. If the target attribute takes on *c* different values, then the entropy S relative to this c-wise classification is defined as

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

where p_i is the proportion/probability of S belonging to class *i*. Logarithm is base 2 because entropy is a measure of the expected encoding length measured in bits.

For e.g. if training data has 14 instances with 6 positive and 8 negative instances, the entropy is calculated as

$$\text{Entropy ([6+, 8-])} = -(6/14)\log_2(6/14) - (8/14)\log_2(8/14) = 0.985$$

A key point to note here is that the more uniform is the probability distribution, the greater is its entropy.

1.2.2 Information gain

It measures the expected reduction in entropy by partitioning the examples according to this attribute. The information gain, Gain(S, A) of an attribute A, relative to the collection of examples S, is defined as

$$\text{Gain(S, A)} = \textit{Entropy}(S) - \sum_{v \in \textit{Values}(A)} \frac{|S_v|}{|S|} \textit{Entropy}(S_v)$$

where $\textit{Values}(A)$ is the set of all possible values for attribute A, and S_v is the subset of S for which the attribute A has value *v*. We can use this measure to rank attributes and build the decision tree where at each node is located the attribute with the highest information gain among the attributes not yet considered in the path from the root.

1.2.3 ID3 Algorithm

The ID3 algorithm implemented by me is as follows: -

ID3 (Examples, Target_Attribute, Attributes)
Examples are the training examples. Target_Attribute is the attribute whose value is to be predicted by the tree. Attributes is the list of attributes which may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a root node for the tree
- IF all examples are positive, Return the single-node tree Root, with label = +
- If all examples are negative, Return the single-node tree Root, with label = -
- If number of predicting attributes is empty, then Return the single node tree Root, with label = most common value of the target attribute in the examples
- Otherwise Begin
    - A ← The Attribute that best classifies examples
    - Decision Tree attribute for Root ← A
    - For each positive value, vi, of A,
        - Add a new tree branch below Root, corresponding to the test A = vi
        - Let Examples(vi), be the subset of examples that have the value vi for A
        - If Examples(vi) is empty
            - Then below this new branch add a leaf node with label = most common target value in the examples
            - Else below this new branch add the subtree ID3 (Examples(vi), Target_Attribute, Attributes – {A})
- End
- Return Root

1.2.3 Example

ID3 algorithm is explained here using the classic 'Play Tennis' example.
The attributes are Outlook, Temp, Humidity, Wind, Play Tennis. The Play Tennis is the target attribute.

Outlook	Temp	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Mild	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Calculating entropy based on the above formulas gives: -
Entropy ([9+,5-]) = $-(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$

Gain(S,Humidity) =0.151
Gain(S,Temp)=0.029
Gain(S, Outlook) = 0.246

Based on the above calculations attribute *outlook* is selected and algorithm is repeated recursively. The decision tree for the algorithm is shown in Figure 1.

2. Implementation and Evaluation

I have implemented the ID3 algorithm explained in the above section using Java programming language. The program takes two files, first the file containing the training data and next the file containing the test data. The format of the file includes the names of the attributes in the first line and then the list of instances. The last attribute is always taken as the target attribute. The program was then executed with different datasets taken from the UCI Machine Learning Repository [2]. The output of the program includes the decision tree and also the accuracy results of the classification.

ID3 algorithm is best suited for: -

1. Instance is represented as attribute-value pairs.
2. Target function has discrete output values.
3. Attribute values should be nominal.

2.1 Handling of attributes with nominal values

Experiments conducted on various datasets gave the following results: -

Name of Dataset	# of training instances	# of test instances	Apparent Error Rate %	Estimated True Error Rate %
TicTacToe	669	289	0.0	17.65
Chess	1548	1648	0.0	0.73
Hayes-roth	77	55	7.8	36.37
Balance-Scale	522	335	0.0	31.95

The above datasets contained nominal attribute values. From the results, apparent error rate, i.e., the error rate on the training examples is very less or zero. But the estimated true error rate on the test data varies. This is because the results depend on how well the attributes generalize the dataset and how much training data is available and how relevant are the attributes in context with the target attribute. For instance in the Chess dataset there are 36 attributes with boolean target attribute, which helps to generalize in a better way compared to other data sets.

2.2 Handling attributes with continuous values

Handling of continuous attributes may lead to inappropriate classification. For e.g., consider a data set where one of the attribute is date. Well each value of date will have a different value and thus this attribute will have the highest information gain. But the problem with such attribute is that they do not generalize in a better way. I have tried to evaluate the performance of the algorithm on datasets with continuous attributes which is as follows: -

Name of Dataset	# of training instances	# of test instances	Apparent Error Rate %	Estimated True Error Rate %
CPU performance	422	231	0.0	68.84
Yeast	838	646	0.0	88.70
Breast Cancer	451	118	0.0	96.60

From the above observation we notice that the performance of ID3 algorithm degrades to a large extent on datasets with attributes having continuous values. It has zero error rates on training examples but the same increases drastically on test datasets. Some of the reasons for the same are: -

1. The attribute which has the highest number of distinct values tends to have the highest information gain but that attribute may not generalize the dataset in a better way.
2. Not all the values are seen during training (because of attributes with continuous values), certain unseen values in the test dataset result in improper classification of the instance. In order to overcome this problem, the algorithm divided the continuous range of values into discrete ranges.

In order to improve the performance, the values for an attribute should be mapped to some discrete values and try to keep them normal. One way is to selects just one value in a selected interval as a representative of the entire range of values in that interval. Usama et al.[3] form a cut point over the continuous range to divide the values into two categories. Equal width interval and adaptive discretization is used [4]. Ventura et al. [5] present a comparison of several preprocessing steps for converting continuous attributes to discrete values.

2.3 Handling attributes with missing values

In real world, datasets contain noise. Well not all the values of each attribute are available at the time of learning. Missing attribute values can be considered as noise. The way ID3 algorithm is formulated, it is difficult to deal with missing values.

I tried to evaluate several datasets with missing attribute values in different way. First of all I just ran the algorithm without any modifications to the algorithm. This means that any missing value ("?") will be taken as a new attribute value. The experimental results are: -

Name of Dataset	# of training instances	# of test instances	Apparent Error Rate %	Estimated True Error Rate %
Soybean	512	171	0.0	20.0
Voting	330	105	0.0	4.80
Mushroom	3955	4169	0.0	4.20

The results show that ID3 seems to give acceptable results for datasets even with missing attributes. The results depend on the type of dataset. The Soybean dataset has a large number of missing values but still during learning it develops a classification even for instances with "?" values for attributes. The Voting data set has few missing values but all the attributes are boolean which makes the classification process easy. The decision tree tries to over generalize the data with considering "?" as a new value because any unseen/missing value for an attribute will come under this category.

I tried another way to deal with missing values for attributes, i.e., to completely neglect the instance with attributes having missing values, while learning. Doing so on the same datasets gave the following results: -

Name of Dataset	# of training instances	# of test instances	Apparent Error Rate %	Estimated True Error Rate %
Soyabean	427	171	0.0	31.60
Voting	176	105	0.0	50.50
Mushroom	2300	4169	0.0	24.00

The above results show that performance of ID3 degrades because in this case all the instances with missing values are classified incorrectly, as no missing value information is embedded in the decision tree while learning from training data.

There are some other techniques as well to deal with noise like replacing missing values with most common value, pruning of the decision tree, having some background information to preprocess the data into more informative feature space and others. Karmaker et al. [6] use EM approach to handle missing attribute values.

The average performance of the ID3 over all that cases considered above is shown in the graph. The average of estimated true error rate in each case is shown and results are compared.

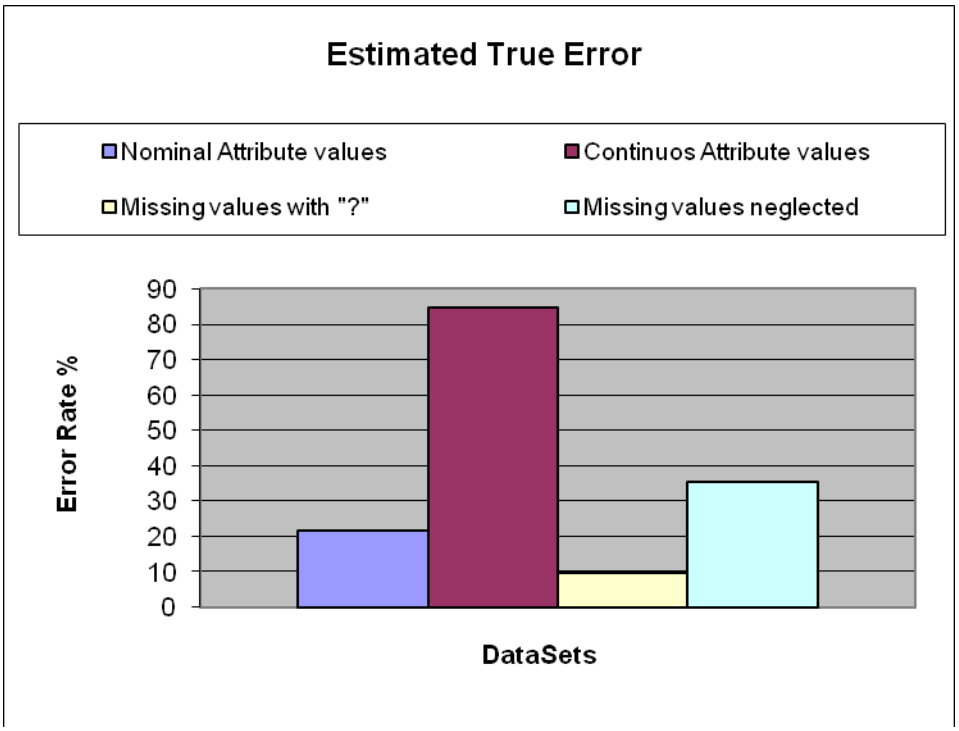

Figure 2. Performance of ID3

The above graph shows that the performance of ID3 in case of continuous attributes deteriorates to a high extent. But in case of missing attribute values the performance of the algorithm actually depends on how the missing attribute is handled. Here the performance is better if missing values are considered compared to nominal attribute value case. Though if the missing values are neglected, the performance slightly deteriorates.

3. Conclusion

The experiments conducted conclude that ID3 works fairly well on classification problems having datasets with nominal attribute values. It also works well in case of missing attribute values but the way missing attributes are handled actually governs the performance of the algorithm. In case of neglecting instances with missing values for the attribute leads to high error rate compared to selecting the missing value as a separate value.

4. Acknowledgement

I thank Dr. James Reggia for his thoughtful discussion and insight in this paper.

5. References

[1] *Tom M. Mitchell*, (1997). Machine Learning, Singapore, McGraw- Hill.

[2] UCI Machine Learning Repository - http://mlearn.ics.uci.edu/databases

[3] *Usama et al*. "On the Handling of Continuous-Values Attributes in Decision Tree Generation". University of Michigan, Ann Arbor.

[4] *R. Chmielewski et al*. "Global Discretization of Continuous Attributes as Preprocessing for Machine Learning". Int. Journal of Approximate Reasoning 1996.

[5] *Dan Ventura et al.* "An Empirical Comparison of Discretization Methods". Proceedings of the Tenth International Symposium on Computer and Information Sciences, pp. 443-450, 1995.

[6] *Karmaker et al.* "Incorporating an EM-Approach for Handling Missing Attribute-Values in Decision Tree Induction"

[7] *Quinlan, J.R. 1986.* Induction of Decision trees. Machine Learning

[8] http://www.cs.cornell.edu/Courses/cs578/2003fa/missing_featsel_lecture.ppt

[9] WEKA Software, The University of Waikato. http://www.cs.waikato.ac.nz/ml/weka/

[10] *Stuart Russell, Peter Norvig*, 1995. Artificial Intelligence: A Modern Approach. New Jersey: Prantice Hall