Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Effective solution for unhandled exception in decision tree induction algorithms

S. Appavu alias Balamurugan [*], Ramasamy Rajaram

Department of Computer Science and Information Technology, Thiagarajar College of Engineering, Thiruparamkundram, Madurai, India

ARTICLE INFO

Keywords:
Data mining
Classification
Decision tree
Majority voting
Influence factor
Pruning

ABSTRACT

This paper deals with some improvements to rule induction algorithms in order to resolve the tie that appear in special cases during the rule generation procedure for specific training data sets. These improvements are demonstrated by experimental results on various data sets. The tie occurs in decision tree induction algorithm when the class prediction at a leaf node cannot be determined by majority voting. When there is a conflict in the leaf node, we need to find the source and the solution to the problem. In this paper, we propose to calculate the Influence factor for each attribute and an update procedure to the decision tree has been suggested to deal with the problem and provide subsequent rectification steps.

© 2009 Published by Elsevier Ltd.

1. Introduction

Decision tree is an important classification tool and various improvements such as ID3 (Quinlan, 1986), ID4 (Utgoff, 1989), ID5 (Utgoff, 1988), ITI (Utgoff, 1994), C4.5 (Quinlan, 1993) and CART (Breiman, Friedman, Olsen, & Stone, 1984), over the original decision tree algorithm have been proposed. All of them deal with the concept of incrementally building a decision tree in real time. In decision tree learning, a decision tree is induced from a set of labelled training instances represented by a tuple of attribute values and a class label. Because of the vast search space, decision tree learning is typically a greedy, top-down recursive process starting with the entire training data and an empty tree. An attribute that best partitions the training data is chosen as the splitting attribute for the root, and the training data are then partitioned into disjoint subsets satisfying the values of the splitting attribute. For each subset, the algorithm proceeds recursively until all instances in a subset belong to the same class. However, prior decision tree algorithms do not handle the exception such as, "when two or more classes have equal probabilities in a tree leaf". This paper investigates exception handling in decision tree construction. We examine one type of exception such as "how to produce classifications from a leaf node which contains ties"; in such a leaf, each class is represented equally, preventing the tree from using "majority voting" to output a classification prediction. We propose new techniques for handling this exception and used real-world data sets to show that this technique improve classification accuracy. One of the problem identified in decision tree learning is that there is a 20% of chance of occurrence of a conflict in a leaf node when applied to real-world data sets. On analysis it has been found that when the Influence factor values are calculated, they are equal in the problem causing node and other nodes in the tree. In the process of finding the source of problem, we propagate backwards in the decision tree until we reach a level wherein the Influence factor values are different and unique. An update procedure to the decision tree has been suggested in this paper to deal with the problem. The paper is organized as follows: Section 2 defines related works in this area. Section 3 portrays the problem handled in this paper. Section 4 explains our proposed algorithm. Section 5 illustrates our proposed update procedure with an example. Section 6 gives a note of comparison between the traditional classification algorithms and the proposed method, highlighting its advantages. Finally, Section 7 summarizes the proposed algorithm and concludes the paper.

2. Related work

Decision tree learning is one of the most widely used and practical methods for inductive learning. The ID3 algorithm (Quinlan, 1986) is a useful concept-learning algorithm because it can efficiently construct a decision tree that is well generalized. For non-incremental learning tasks, this algorithm is often an ideal choice for building a classification rule. However, for incremental learning tasks, it would be far preferable to accept instances incrementally, without the necessity to build a new decision tree each time. There exist several techniques to construct incremental decision tree based models. Some of the earlier efforts include ID4 (Utgoff, 1989), ID5 (Utgoff, 1988), ID5R (Utgoff, 1989), and ITI (Utgoff, 1994). All these systems work using the ID3 style "information gain" measure to select the attributes. They are all designed to incrementally build a decision tree using one training instance at a time by keeping the necessary statistics (measure for information gain) at each decision node.

* Corresponding author.
E-mail addresses: sbit@tce.edu (S. Appavu alias Balamurugan), rrajaram@tce.edu (R. Rajaram).

0957-4174/$ - see front matter © 2009 Published by Elsevier Ltd.
doi:10.1016/j.eswa.2009.03.072

The ID4 algorithm (Utgoff, 1989) builds decision trees incrementally. Many learning tasks are incremental as new instances or details become available over time. The ID4 algorithm (Utgoff, 1989) works by building a tree and updating it as new instances become available. The ID3 algorithm can be used to learn incrementally by adding each new instance to the training set as it becomes available and by re-running ID3 against the enlarged training set. This is however computationally inefficient. The ID5 (Utgoff, 1988) and ID5R (Utgoff, 1989) are both incremental decision tree builders that overcome the deficiencies of ID4. The essential difference is that when tree restructuring is required, instead of discarding a sub tree due to its high entropy, the attribute that is to be placed at the node is pulled up to the node and the tree structure below the node is retained. In the case of ID5 (Utgoff, 1988) the sub trees are not recursively updated while in ID5R (Utgoff, 1989) they are updated recursively. Leaving the sub trees unrestructured is computationally more efficient. However the resulting sub tree is not guaranteed to be the same as the one that would be produced by ID3 on the same training instances. The Incremental Tree Inducer (ITI) (Utgoff, 1994) is a programme that constructs decision tree automatically from labelled examples. The most useful aspect of the ITI algorithm is that it provides a mechanism for incremental tree induction. If one has already constructed a tree, and then obtains a new labelled example, it is possible to present it to the algorithm, and have the algorithm revise the tree as necessary. The alternative would be to build a new tree from the scratch, based on the augmented set of labelled examples, which is typically much more expensive. ITI handles symbolic variables, numeric variables, and missing data values. It includes a virtual pruning mechanism too.

The development of decision tree learning leads to and it encouraged by a growing number of commercial systems such as C5.0/See5 (RuleQuest Research), MineSet (SGI), and Intelligent Miner (IBM). Numerous techniques have been developed to speed up decision tree learning, such as designing a fast tree-growing algorithm, parallelization, and data partitioning.

A number of strategies for decision tree improvements have been proposed in the literature (Buntine, 1992; Hartmann, Varshney, Mehrotra, & Gerberich, 1982; Kohavi & Kunz, 1997; Mickens, Szummer, Narayanan, & Snitch, 2007; Quinlan, 1987; Utgoff, 2004). They aim at "tweaking" an already robust model despite its main obvious limitation. A number of ensemble classifiers have been proposed in the literature (Chipman, George, and Mcculloch, 1998; Kohavi, 1996; Wang et al., 2004; Zhou and Chen, 2002) which appear to have little improvement on accuracy especially when the added complexity of the method is considered.

3. Problem statements

This paper addresses research question in the context of decision tree induction: which class to choose when, classified with respect to an attribute, the number of records having the different class values are equal, i.e. when majority voting fails (see Figs. 1–3).

The set of attributes used to describe an instance is denoted by A, and the individual attributes are indicated as A_i, where i between 1 and the number of attributes, m. For each attribute A_i, the set of possible values is denoted as V_i. The individual values are indicated by v_{ij}, where j between 1 and the number of values for attributes A_i. The notations used to represent the features of training data are, the attributes as A_j, where $j = 1$ to m, the class attribute as C the values to each of the attributes will be V_{ij}, where $i = 1 \ldots n$ and j refers to the attribute to which it belongs. The general structure of the training data set is shown in Table 1.

After analyzing the decision tree induction algorithms, we found that the concept of majority voting has to handle different types of inputs. Consider the attribute A_k between A_1 and A_m and C be the class attribute, the value of A_k are V_{1k} and V_{2k}, and the value of the class attribute be C_1 and C_2. Classification can be done if:

1. Both A_k and C has only one distinct values.
2. When most of the attribute value V_{1k}, $V_{2k} \ldots V_{nk}$ of particular attribute A_k belong to same class that is called majority voting.

From Table 2, the maximum occurrence of the distinct value and its corresponding maximum occurrence of the distinct class label value is obtained. Hence majority voting is successful.

Consider the training data set is shown in Tables 3 where majority voting fails.

Under this condition only one rule can be generated as

If $A_k = V_{2k}$ then $C = C_2$

In Table 3, the count of the distinct value is 2. Hence two rules should be generated from the table but only one rule is generated with the help of majority voting and another rule cannot be generated

If $A_k = V_{1k}$ then $C = ?$

In Table 3, the value for A_k can be found by majority voting as $A_k = V_{1k}$ but its corresponding class value cannot be determined because among the four records there is an equal partition of class attribute $C = C_1$ or C_2, hence the majority voting cannot be applied in this case. The traditional decision tree induction algorithms does not give any specific solution to handle this problem.

4. The proposed learning algorithm

The Decision tree induction algorithms update procedure to handle the cases when the concept of majority voting fails in the leaf node are given in Fig. 2.

5. Implementation of the proposed algorithm

To prove the efficiency of the proposed algorithm we consider Table 4 used in the problem definition. When the concept of majority voting fails in the leaf node, an exception occurs in the decision tree induction algorithm. At this point the proposed algorithm is used.

5.1. Step 1

Divide the training data based on the class label. In this example the records having the class label 'Class: Buys_computer = Yes' are placed in the Table 5 and the records having the class label value 'Class: Buys_computer = No' are placed in the Table 6. The records 1–14 form the training data and the remaining records form the test data.

5.2. Step 2

Find the influence factor for all the attribute values. The influence factor gives the dependability of the attribute value on the class label. The formula for Influence factor for a particular Class C_i is given below

$$\text{Influence factor } I(A_j = \text{“}X_l\text{”} | C_i) = \frac{N(A_j = \text{“}X_l\text{”} | C_i)}{N(C_i)}$$

where N (A_j = "X"|C_i) = number of records in which attribute A_j having the value X_l has the class label C_i.

N (C_i) = total number of records in which the class label is C_i.