A New Method to Construct Decision Trees from Training Instances

Yih-Jen Horng*, Chia-Hoang Lee*, and Shyi-Ming Chen**

***Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, R. O. C.**

****Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, R. O. C.**

Abstract

In this paper, we present a new learning method to construct decision trees from training instances for handling classification problems. The decision trees constructed by the proposed method can be much smaller than the ones constructed by the ID3 algorithm. The proposed method is better than the ID3 algorithm when noise or coincidental regularities exist in the training instances.

Keywords: ID3 Algorithm, Classification Problems, Decision Trees, Instance-Based Learning, Overfitting Problems.

1. Introduction

Machine learning is one of the most important research topics in the field of Artificial Intelligence. The goal of machine learning is to build a knowledge base by observing human behaviors or operation models with respect to the encountered environments or objects, where the human behaviors or operation models are represented in the form of training examples. Then, the learned knowledge base is transferred to some corresponding operational rules that let machines to act like human beings.

Many machine learning algorithms have been proposed in the past decades, such as the version space learning methods [17], the decision tree learning methods [19], [20], the artificial neural networks [22], the genetic algorithms [10], …, etc. Among these machine learning methods, the decision tree learning methods provide an easy and efficient way to derive knowledge from the training data.

In 1986, Quinlan [19] proposed a decision-tree construction method called the ID3 algorithm. It has been widely used to construct decision trees. But as stated in [18], when the noise is contained in the training instances or when the number of training instances is too small to produce a representative sample of the instance space, the decision trees constructed by the ID3 learning algorithm may "overfit" the training instances. That is, although the decision tree constructed by the ID3 algorithm can correctly classify the training instances, it may be not an optimal decision tree due to the fact that there may exist other decision trees which have a higher classification accuracy rate than

the one constructed by the ID3 algorithm with respect to the instances distributed in the whole instance space.

There are two approaches [18] to solve the "overfitting" problem of the ID3 algorithm. The first approach adopts the methods of stopping growing unnecessary branches before the construction of the decision tree is completed. The other approach takes a two-stage process by allowing the decision tree to grow completely at first, and then a pruning operation is performed to prune the unnecessary branches. Since it is difficult to estimate which branches are unnecessary when the decision trees are growing, the methods which prune the complete decision trees are more practical. However, since many computational efforts are needed to decide which branches are unnecessary and to prune these unnecessary branches, the two-stage process may not be efficient enough in the applications that need to construct decision trees quickly.

In fact, one common characteristic of the methods to solve the "overfitting" problem of the ID3 algorithm is to construct a decision tree that doesn't classify the training instances perfectly. That is, instances labeled as different classes can be allowed to affiliate to the same leaf node. When these imperfect decision trees are used to classify an unseen instance, the unseen instance is labeled as the most common classification of the existing instances affiliating to the leaf node that corresponds to the attribute values of the unseen instance. This means that the classification of an unseen instance is based on the existing instances from different classes. If the classification accuracy rate is good enough, the underlying branches which perfectly classify the existing instances are not necessary. The advantage of this approach is some branches misled by noise or by "coincidental regularities" may be avoided.

However, the classification strategy adopted so far is simply voted by the existing instances. That is, the unseen instance is labeled as the most common classification of the existing instances. If we can use a more powerful classification strategy, then the classification accuracy rate should be increased. Therefore, the impact of noisy data and "coincidental regularities" should be decreased.

In this paper, we propose a new method to construct decision trees from training instances which allows instances from more than one class existing in the same leaf node. Moreover, we apply the instance-based learning techniques [18] to deal with the classification of the unseen instances. The proposed method performs better than the ID3 algorithm when noise or coincidental regularities exist in the training data from the viewpoint of the size and the classification accuracy rate of the constructed decision trees.

The rest of this paper is organized as follows. In Section 2, we briefly review the ID3 algorithm [19] and discuss the "overfitting" problem. In Section 3, we present a new method to construct decision trees. In Section 4, we

present a classification method based on the constructed decision tree. The conclusions are discussed in Section 5.

2. A Review of the ID3 Algorithm

The decision tree learning method starting from taking training instances as an input and ending in constructing a decision tree as the output is one of the widely used inductive learning methods. The decision tree learning methods search the hypothesis space for hypotheses approximating the target function consistent with the training instances. In a decision tree learning method, the resulting hypotheses are represented in the form of decision trees. However, unlike other induction learning methods (e.g., the version space method [17]), the decision tree learning method maintains only a single hypothesis for each training data set instead of all the possible hypotheses.

Each non-leaf node of the decision tree represents a selected attribute A that is used to test instances; each edge represents a branch which take one possible value of attribute A; each leaf node represents one classification. Fig. 1 shows a decision tree, where A_i is the selected attribute, V_{ij} is the branch value of attribute A_i, and L_i is the classification name. The instances are tested by parts of the non-leaf nodes of the decision tree, which form a path that directs the instance from the root node to a leaf node which provides a classification to the instance.

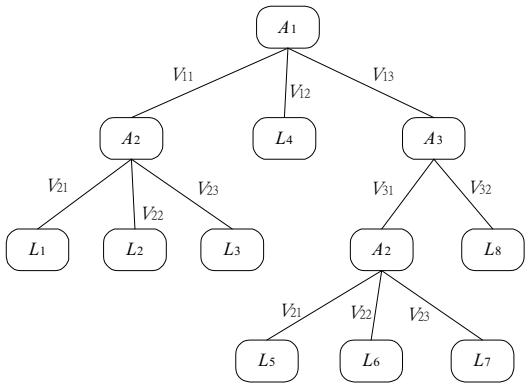

Fig. 1. A decision tree.

Many decision tree construction methods have been proposed. In 1986, Quinlan proposed a decision tree construction method named as "ID3" algorithm [19], which has been used in many practical applications since its ease of use. The ID3 algorithm grows the decision tree from the root node downward and greedily selects the best attributes as interior nodes when branches are needed to perfectly classify the training instances. The goodness of an attribute A is the degree that attribute A discriminates the training instances with respect to their classifications. Let A be an attribute and let S be a collection of training instances. The information gain Gain(S, A) of the attribute A with respect to the set S of the training instances can be calculated as follows [18]:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v), \quad (1)$$

where $Values(A)$ denotes a set of all possible values of the attribute A, S_v is a subset of S in which the attribute A has the value v, $|S_v|$ denotes the number of elements in S_v, $|S|$ denotes the number of elements in S, and $Entropy(S)$ is defined as follows:

$$Entropy(S) \equiv \sum_{i=1}^{c} - p_i \log_2 p_i, \qquad (2)$$

where p_i is the proportion of S belonging to class I, and c is the number of classifications of the training instances.

The ID3 algorithm chooses the most suited decision tree among all consistent decision trees based on its inductive bias. That is, the smaller decision tree is preferred than the larger one, and the tree selects the attribute which has the highest information gain as the root node.

Although the ID3 algorithm intends to construct the best decision tree for classifying the overall instances distributed in the instance space, the performance of the constructed decision trees depends on the training instances used to construct the decision trees. When there are other decision trees perform better than the ones constructed by the ID3 algorithm over the entire instances distributed in the instance space, the decision tree "overfit" the training instances [18].

There are two reasons of the overfitting problem of the ID3 algorithm which both cause the ID3 algorithm to generate some unreliable branches. That is, the constructed tree tends to be very large. One reason is that the data contained in the training examples are not deterministic but uncertain. The uncertain data come from the noise in the measurements and represented both in attribute values and in the classifications of the training instances. When this happens, the unseen instances that go into the leaf nodes containing noise may make a wrong decision. In [14], Minger pointed out that when noisy and uncertain data is contained in the training instances, the decision tree constructed by the ID3 algorithm tends to be very large and will decrease ten to twenty five percent of the classification accuracy rate.

The other one is that the number of the training instances is too small to represent the whole instances in the instance space. When this happens, it is quite possible for coincidental regularities to occur, in which some attribute happens to partition the examples very well, despite being unrelated to the actual target function.

There are two kind of methods proposed to solve the overfitting problem of the ID3 algorithm [19]. The first one is the approach that stops growing the tree earlier before it reaches the point where it perfectly classifies the training data. The other one is the approach that allows the tree to overfit the data, and then post-prune the tree.

Although the first approach might seem more straightforward, the second approach of post-pruning overfit trees has been found to be more successful in practice [18]. This is due to the difficulty of the first approach to estimate precisely when to stop growing the tree.

However, these decision tree pruning methods all need some post processing which need many computational efforts. Furthermore, the post pruning approach needs separating the available examples into two different sets, i.e., the training set and the validation set. The

training set is used to induct a decision tree, while the validation set is used as the evaluation utility of post-pruning nodes from the tree. A major drawback of this approach is that when the number of data is limited, withholding part of it for the validation set reduces even further the number of examples available for training.

Thus, it is necessary to develop an effective and efficient decision tree induction method that can reduce the impact of the overfitting problem of the ID3 algorithm.

3. A New Method to Construct Decision Trees from Training Instances

A new decision tree induction method is proposed in this section, which intends to overcome the overfitting problem of the ID3 algorithm. Moreover, in order to enhance the ability of the proposed method, we apply some strategies to make the proposed method can deal with the data whose attribute values are continuous type, where the ID3 algorithm can't handle this situation.

In order to overcome the overfitting problem, we adopt the approach of stopping growing the decision tree completely. That is, instead of perfectly classifying instances associating with each leaf node, we take a voting strategy, i.e., when the proportion of some training instances belonging to a specific class to all classes exceed a predefined proportion threshold value α, where $\alpha \in [0, 1]$, the nodes stop branching.

However, since there may be more than one class exists in the leaf node, when a unseen instance gets into the leaf node, there should have some methods to decide the class of the new instance based on the training instances affiliated with the leaf node. We adopt the instance-based learning techniques [18] to deal with the classification. The details of the classification process are described in the next section.

Another issue of the ID3 algorithm [19] is that it can't deal with continuous-valued attributes. This drawback can be overcomed by preprocessing the continuous-valued attributes. That is, by defining new discrete-valued attributes and mapping the continuous-valued attributes into the new discrete-valued attributes. A common approach to this goal is by setting a set of "cut points" to each continuous-valued attributes, i.e., the possible values are broken into several intervals and the continuous-valued attributes can then be mapped into distinct intervals. An alternative way is mapping the continuous values into some predefined fuzzy sets [25], and a fuzzy decision tree can be obtained [4], [6], [11], [12], [23], [24].

In the proposed decision tree learning method, we employ a strategy that is similar to the "cut points" approach to handle continuous-valued attributes, but the number of cut point is only one. That is, we simply divide the possible values of the continuous-valued attribute into two parts. Furthermore, the continuous-valued attributes are normalized

before using them to induct a decision tree. Assume that A is a continuous-valued attribute and assume that $A(max)$ denotes the maximum value of A, and assume that $A(min)$ denotes the minimum value of A, then a possible value $A(x)$ of the attribute A is normalized as follows:

$$A_{normalization}(x) = \frac{A(x) - A(\min)}{A(\max) - A(\min)}, \qquad (3)$$

where $A_{normalization}(x)$ is the normalized value of $A(x)$. The possible values of the continuous-valued attribute are between zero and one after the normalization process. The advantage of the normalization is that only one common cut point should be set for all continuous-valued attributes since different scales of different attributes are unified. In this paper, we set the cut point to 0.5. The proposed decision-tree learning algorithm is now presented as follows:

Step 1: **for** each continuous-valued attribute C **do**
begin
find the maximum value and the minimum value of attribute C;
normalize the attribute value of each training instance with respect to attribute C according to formula (3);
divide the normalized attribute values of each training instance with respect to attribute C into two parts depending on whether they are less than the cut point λ or not, where $\lambda \in [0, 1]$
end.

Step 2: Create a root node R.

Step 3: **if** the proportion of some involved class L exceeds the proportion threshold value α, where $\alpha \in [0, 1]$, **then** return the root node R.

Step 4: **if** no further attribute can be used to test instances **then** return the root node R
else choose an attribute A which has the largest information gain among available attributes according to formula (1) and let A be the test attribute in root node R.

Step 5: **for** each possible value V_i of attribute A **do**
begin
add a new branch below R corresponding to the test "$A = V_i$";
let E_{Vi} be the subset of the training instances whose value for attribute A is V_i;
if E_{Vi} is empty **then** below this new branch add a leaf node
else below this new branch add a subtree generated by the proposed decision-tree learning algorithm but using E_{Vi} as the training instances and take attribute A away from the available attributes
end.

Step 6: Return root node R.

It should be noted that the leaf nodes are not labeled with a classification name. That is, when this kind of decision tree is used to classify unseen instances, furthermore classification process is required. We will introduce a classification method based on the constructed decision tree by the proposed learning algorithm in the next section.

4. A Classification Method Based on the Constructed Decision Tree

After the decision tree has been constructed by means of the proposed learning method, the decision tree can then be used to classify the unseen instances. By means of the decision trees constructed by the ID3 algorithm, the unseen instances are tested by parts of the non-leaf nodes of the constructed decision tree, which form a path that directs the instance from the root node to a leaf node that provide a classification to the instance. However, if the classification process based on the decision trees constructed by the proposed method, the leaf nodes won't provide any classification to the instance. Thus, an additional judgment mechanism is required to give the classification to an unseen instance based on its relationship to the training instances affiliated with the leaf node that the unseen instance gets into. However, the additional judgment mechanism must satisfy the following conditions. Firstly, the impact of noisy data should be effectively reduced when noisy data is involved. Secondly, the estimation of the classification accuracy rate must based on part of the total training instances, that is, the estimation must based on the training instances affiliated with one particular leaf node of the constructed decision tree. In this paper, we adopt the instance-based learning techniques [18] to assign the classification to a new instance due to the fact that the instance-based learning techniques are robust to noisy training data and can estimate the target function locally and differently for each new instance to be classified.

Instance-based learning techniques such as the "nearest neighbor" and the "locally weighted regression" assumed instances can be represented as points in a Euclidean space. Unlike other learning methods, the instance-based learning techniques simply store the training instances. When a new instance is encountered, a set of similar related instances is retrieved to estimate the classification of the new instance. Thus, the instance-based learning methods are also referred to as "lazy" learning methods [18] because they delay processing until a new instance to be classified.

The instance-based learning method could overcome the impact of isolated noisy training examples. For example, Fig. 2 presents a two-dimensional instance space made by the training instances affiliated with a leaf node, where "+" stands for a positive example, "-" stands for a negative example, and X is the unseen instance to be classified. Assume that the only negative example is a noise, which is supposed to be labeled as positive. If we use the

distance-weighted nearest neighbor algorithm [18] that takes all these four instances into account to estimate the classification of *X*, then *X* is classified as "positive" if it is positioned in the left side of the curve, whereas classified as "negative" if it is on the right side of the curve.

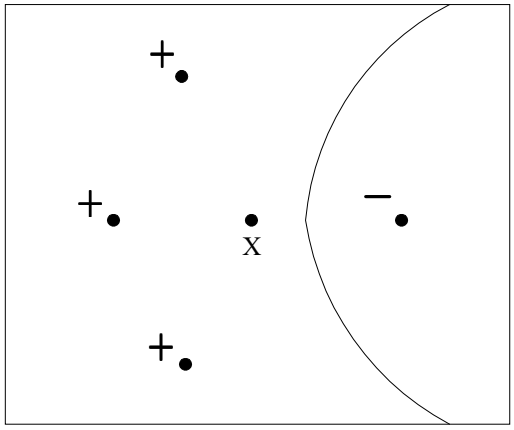

Fig. 2. A set of positive and negative training examples along with an unseen instance X to be classified.

We adopt an approach that is similar to the distance-weighted nearest neighbor algorithm [18] except that the number of nearest neighbors to take into account is not given in advance. In stead, we take all of the training instances in a specific leaf node as the basis to predict the classification of the new instance.

However, since the set of attributes of the instance consist of discrete-valued attributes and continuous-valued attributes, a strategy is required to calculate the distance between discrete-valued attributes and continuous-valued attributes, respectively, and then merge these distances. The distance calculation method between two instances that we use in this paper is presented as follows:

***Distance Calculation Algorithm*:**

Input: Two instances I_i and I_j;

Output: Distance between I_i and I_j;

Variable: d_distance, c_distance;

/* d_distance denotes the summation of the differences between the values of the instances I_i and I_j of all discrete-valued attributes; c_distance denotes the summation of the differences between the values of the instances I_i and I_j of all continuous-valued attributes */

Step 1: **for** every attribute A contained in the instances,

if attribute A is a continuous-valued attribute **then** go to Step 2

else go to Step 3.

Step 2: normalize the values of attribute A in instances I_i and I_j according to formula (3), respectively;

calculate the difference of these two normalized values and add it to c_distance;

go to Step 4.

Step 3: **if** the value of attribute A in instances I_i and I_j is different from the value of attribute A in instances I_i and I_j **then** add one to d_distance.

Step 4: Return the summation of c_distance and d_distance.

After the distance between the training instance and the new instance has been calculated, the weight $w(I)$ of the training

instances to predict the classification can then be obtained according to the following formula:

$$w(I) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(D(I))^2} \quad (4)$$

where *D(I)* denotes the distance between the training instance *I* and the new instance.

The weights of all training instances are then summarized to get the classification weight of each classification. The classification with the highest weight is then chosen as the classification result of the unseen instance.

5. Conclusions

In this paper, we have presented a new method for quickly constructing decision trees from training instances. By allowing impure classifications of the instances associating to the same leaf node, the proposed learning method could generate a much smaller decision tree than the one constructed by the ID3 algorithm. The proposed method is better than the ID3 algorithm when noise or coincidental regularities exist in the training instances.

References

[1] L. Breiman, J. H. Friedman, J. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. CA: Wadsworth International, 1984.

[2] D. W. Aha, *UCI Machine Learning Group*, http://www.ics.uci.edu/~mlearn/MLRepository.html.

[3] C. Apté and S. Weiss, "Data mining with decision trees and decision rules," *Future Generation Computer Systems,* vol. 13, no. 2-3, pp. 197-210, 1997.

[4] R. L. P. Chang and T. Pavlidis, "Fuzzy decision tree algorithm," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 7, no. 1, pp. 28-35, 1977.

[5] C J. Chen and Q. Y. Shi, "Shape features for cancer cell recognition," in *Proceedings of the 5th International Conference on Pattern Recognition*, Miami Beach, Florida, U. S. A., pp. 579-581, 1980.

[6] K. J. Cios and L. M. Sztandera, "Continuous ID3 algorithm with fuzzy entropy measures," in *Proceedings of the First IEEE International Conference on Fuzzy Systems*, San Diego, U. S. A., pp. 469-476, 1992.

[7] G. Cornell and T. Strain, *Delphi Nuts & Bolts for Experienced Programmers*. CA: McGraw-Hill, 1995.

[8] G. R. Dattatreya and V. S. Sarma, "Decision tree design for pattern recognition including feature measurement cost," in *Proceedings of the 5th International Conference on Pattern Recognition*, Miami Beach, Florida, U. S. A., pp. 1212-1214, 1980.

[9] M. Dong and R. Kothari, "Look-ahead based fuzzy decision tree induction," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 3, pp. 461-468, 2001.

[10] D. Goldberg, *Genetic Algorithms in*

Search, Optimization, and Machine Learning. MA: Addison-Wesley, 1989.

[11] C. Z. Janikow, "Exemplar learning in fuzzy decision trees," in *Proceedings of the 5th IEEE International Conference on Fuzzy Systems*, New Orleans, U. S. A., pp. 1500-1505, 1996.

[12] C. Z. Janikow, "Fuzzy decision trees: Issues and methods," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 28, no. 1, pp. 1-14, 1998.

[13] Y. K. Lin and K. S. Fu, "Automatic classification of cervical cells using a binary tree classifier," in *Proceedings of the 5th International Conference on Pattern Recognition*, Miami Beach, Florida, U. S. A., pp. 570-574, 1980.

[14] J. Mingers, "Expert systems — rule induction with statistical data," *Journal of the Operational Research Society*, vol. 38, no. 1, pp. 39-47, 1987.

[15] J. Mingers, "Rule induction with statistical data — a comparison with multiple regression," *Journal of the Operational Research Society*, vol. 38, no. 4, pp. 347-352, 1987.

[16] J. Mingers, "An empirical comparison of pruning methods for decision tree induction," *Machine Learning*, vol. 4, no. 2, pp. 319-342, 1989.

[17] T. M. Mitchell, "Version spaces: a candidate elimination approach to rule learning," in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Massachusetts, U. S. A., pp. 305-310, 1977.

[18] T. M. Mitchell, *Machine Learning*. Singapore: McGraw-Hill, 1997.

[19] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81-106, 1986.

[20] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-Machine Studies*, vol. 27, no. 3, pp.221-234, 1987.

[21] J. R. Quinlan, C4.*5: Programs for Machine Learning*. CA: Morgan Kaufmann Publishers, 1992.

[22] E. Rich and K. Knight, *Artificial Intelligence*. Singapore: McGraw-Hill, 1991.

[23] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414-1427, 1992.

[24] Y. Yuan and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets and Systems*, vol. 69, no. 2, pp. 125-139, 1995.

[25] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338-353, 1965.