Expert Systems with Applications 36 (2009) 12113–12119

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.

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

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

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

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

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 $N(A_j = \text{“}X\text{”}|C_i)$ = 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.

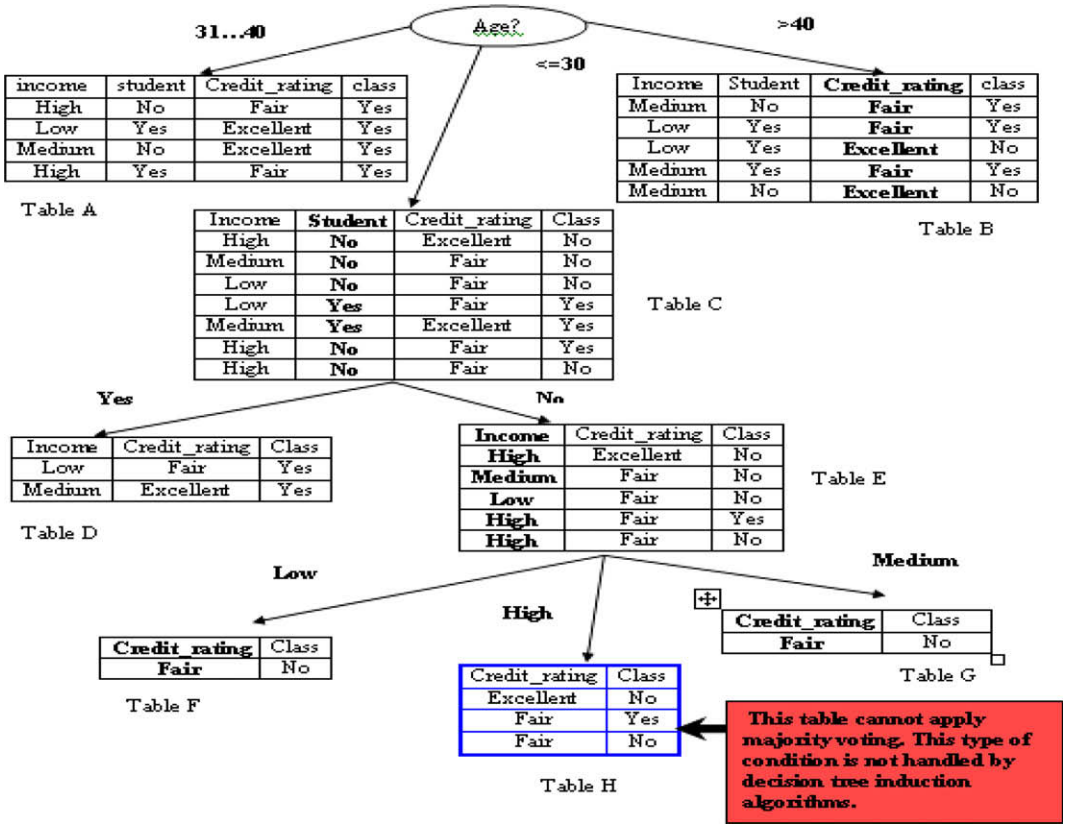

Fig. 1. The attribute age has the highest information gain and therefore becomes a test attribute at the root node of the decision tree. Branches are grown for each value of age. The samples are shown partitioned according to each branch.

Now the influence factor of attribute say A_j in class C_i is found by the formula

$$A_jC_i = \text{Max } (I(A_j = \text{“}X_l\text{”}|C_i)) \text{ where l varies from } 1 \ldots k$$

where

A_j = attribute that is currently considered for calculation, j varies from $1 \ldots n$. Here n refers to maximum number of predictive attributes and k is maximum number of attribute values for the attribute A_j.

5.3. Step 2a

Find the influence factor for all the attribute values of the Table 5.

AGE Attribute:

I (AGE=“ >40” | Buys_Computer=“Yes”) = 3/9
I (AGE=“31–40” | Buys_Computer=“Yes”) = 4/9
I (AGE=“ <30” | Buys_Computer=“Yes”) = 2/9

Age_{yes}=Max (I (Age=“X_l” | Buys_Computer=“Yes”)) where l varies from $1 \ldots k$
Age_{yes}=4/9=0.4.
The influence factor is more for the attribute value ‘31–40’.

INCOME Attribute:

I (INCOME=“low” | Buys_Computer=“Yes”) = 3/9
I (INCOME=“medium” | Buys_Computer=“Yes”) = 4/9
I (INCOME=“high” | Buys_Computer=“Yes”) = 2/9

$Income_{yes}$=Max (I (Age=“X_l”| Buys_Computer=“Yes”)) where l varies from $1 \ldots k$
$Income_{yes}$=4/9=0.4.
The influence factor is more for the attribute value ‘medium’.

STUDENT Attribute:

I (STUDENT=“yes” | Buys_Computer=“Yes”)=6/9
I (STUDENT=“no” | Buys_Computer=“Yes”)=3/9

$Student_{yes}$=Max (I (Age=“X_l”| Buys_Computer=“Yes”)) where l varies from $1 \ldots k$
$Student_{yes}$=6/9=0.6.
The influence factor is more for the attribute value’yes’.

CREDIT_RATING Attribute:

I (CREDIT_RATING=“fair” | Buys_Computer=“Yes”)=6/9
I (CREDIT_RATING=“excellent” | Buys_Computer=“Yes”)=3/9

$CREDIT\_RATING_{yes}$=Max (I (Age=“X_l” | Buys_Computer=“Yes”)) where l varies from $1 \ldots k$
$CREDIT\_RATING_{yes}$=6/9=0.6.
The influence factor is more for the attribute value’fair’.

5.4. Step 2b

Find the influence factor for all the attribute values of the Table 6.

AGE Attribute:

I (AGE=“ >40” | Buys_Computer=“No”)=2/5
I (AGE=“31–40” | Buys_Computer=“No”)=0/5
I (AGE=“ <30” | Buys_Computer=“No”)=3/5

Age_{yes}=Max (I (Age=“X_l”| Buys_Computer=“Yes”)) where l varies from $1 \ldots k$

1. Let $A_1 \ldots A_n$ be the total number of Predictive Attributes in both the test and training data where n refers to maximum number of Predictive Attributes. An attribute say A_l has $X_1 \ldots X_k$ values where k is maximum number of attribute values for the attribute A_l. Similarly the attributes $A_1 \ldots A_n$ has its corresponding attribute values.

2. Let $C_1 \ldots C_m$ be the different class labels assigned to the training data where m refers to the number of classes. The records in the training data having the class label Ci is placed in Table C_i* where i vary from 1...m. For example the records in the training data having the class label C_1 are placed in Table C_1*.

3. The Influence factor for an attribute with class label C_i is found from its respective Table C_i*. 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.

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

Where $N (A_j = \text{“}X\text{”} | C_i)$ = 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

4. Similarly the Influence factor of all attributes $A_1 \ldots A_n$ in the Table C_i* is found where n refers to maximum number of Predictive Attributes. Now the Influence factor of attribute say A_j in class C_i is found by the formula

$$A_j C_i = \text{Max } (I (A_j = \text{“}X_l\text{”} | C_i)) \quad \text{where l varies from 1…k}$$

Where
A_j = Attribute that is currently considered for calculation, j varies from 1…n here n refers to maximum number of Predictive Attributes and k is maximum number of attribute values for the attribute A_j.

Now the Influence factor of an attribute A_1 in all the available classes $C_1 \ldots C_m$ is referred to as A_1C_1, A_1C_2,… A_1C_m. In general the Influence factor of an attribute A_j in all the available classes $C_1 \ldots C_m$ is referred to as A_jC_j, A_jC_2,… A_jC_m.

5. The Influence factor for all attributes of a Class C_i is found. Now the Influence factors for all other classes are found by referring its corresponding Tables.

6. Now the Maximum Influence factor is found by the formula given below:

Maximum Influence factor i.e. MIF (A_j) = ChkMax (A_jC_i)

Where A_j =Attribute that is currently considered for calculation.

Now the MIF of all attributes are found. For example, consider a training data having attribute A_1 and class labels C_1 and C_2, now the Maximum Influence factor i.e.

MIF $(A_1) = A_1C_1$ when $A_1C_1 > A_1C_2$. The function Max (A_jC_i) returns this maximum value.

7. The Maximum Influence factor of an attribute A_j can't be found when that attribute A_j has Influence Factor Values such that $A_jC_1 = A_jC_2 = \ldots = A_jC_m$. Maximum Influence factor i.e. MIF (A_j) is Zero in this case. The attributes whose Maximum Influence factor is null are not considered when determining the class label.

8. Now consider the Decision tree. Now prune the decision tree by removing the attributes whose Maximum Influence factor is Zero. While removing these attributes, remove the entire child attributes that have been branched out of these attribute and classify the test sample with this pruned tree.

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

Age_{yes}=3/5=0.6.
The influence factor is more for the attribute value' <30'.
INCOME Attribute:

I (INCOME="low" | Buys_Computer="No")=2/5
I (INCOME="medium" | Buys_Computer="No")=2/5
I (INCOME="high" | Buys_Computer="No")=1/5

$Income_{yes}$=Max (I (Age="X_l"| Buys_Computer="Yes")) where l varies from 1…*k*

$Income_{yes}$=2/5=0.4.
The influence factor is more for the attribute values 'low' and 'medium'.
STUDENT Attribute:

I (STUDENT="yes" | Buys_Computer="No")=1/5
I (STUDENT="no" | Buys_Computer="No")=4/5

$Student_{yes}$=Max (I (Age="X_l"| Buys_Computer="Yes")) where l varies from 1…*k*

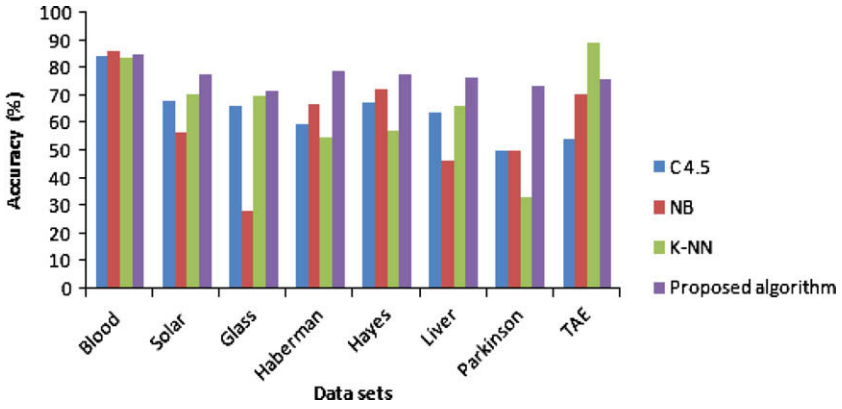

Fig. 3. Performance of classifiers.

Table 1
General structure of the training data set.

A_1	A_2	A_3	A_4	A_5	...	A_m	C_1
V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	...	V_{1m}	C_2
V_{21}	V_{22}	V_{23}	V_{24}	V_{25}	...	V_{2m}	C_3
V_{31}	V_{32}	V_{33}	V_{34}	V_{35}	...	V_{3m}	C_4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
V_{n1}	V_{n2}	V_{n3}	V_{n4}	V_{n5}	...	V_{nm}	C_n

Table 2
Input format for majority voting condition.

A_k	C
V_{1k}	C_2
V_{2k}	C_2
V_{1k}	C_2
V_{1k}	C_1

If $A_{K=}V_{1k}$ then $C = C_2$.
If $A_{K=}V_{2k}$ then $C = C_2$.

Table 3
Example training data set where majority voting cannot be applied.

A_k	C
V_{1k}	C_1
V_{2k}	C_2
V_{1k}	C_1
V_{1k}	C_2
V_{1k}	C_2

Table 4
Training data tuples from the all electronics customer database.

RID	Age	Income	Student	Credit_rating	Class: Buys_computer
1.	>40	Medium	No	Fair	Yes
2.	>40	Low	Yes	Fair	Yes
3.	>40	Low	Yes	Excellent	No
4.	>40	Medium	Yes	Fair	Yes
5.	>40	Medium	No	Excellent	No
6.	31…40	High	No	Fair	Yes
7.	31…40	Low	Yes	Excellent	Yes
8.	31…40	Medium	No	Excellent	Yes
9.	31…40	High	Yes	Fair	Yes
10.	<=30	High	No	Excellent	No
11.	<=30	Medium	No	Fair	No
12.	<=30	Low	No	Fair	No
13.	<=30	Low	Yes	Fair	Yes
14.	<=30	Medium	Yes	Excellent	Yes
15.	<=30	High	No	Fair	Yes
16.	<=30	High	No	Fair	No

Table 5
The records having the class label 'Class: Buys_computer=Yes'.

AGE	INCOME	STUDENT	CREDIT_RATING	BUYS_COMPUTER
>40	Medium	No	Fair	Yes
>40	Low	Yes	Fair	Yes
>40	Medium	Yes	Fair	Yes
31–40	High	No	Fair	Yes
31–40	Low	Yes	Excellent	Yes
31–40	Medium	No	Excellent	Yes
31–40	High	Yes	Fair	Yes
<30	Low	Yes	Fair	Yes
<30	Medium	Yes	Excellent	Yes

Table 6
The records having the class label 'Class:Buys_computer=No'.

AGE	INCOME	STUDENT	CREDIT_RATING	BUYS_COMPUTER
>40	Low	Yes	Excellent	No
>40	Medium	No	Excellent	No
<30	High	No	Excellent	No
<30	Medium	No	Fair	No
<30	Low	No	Fair	No

$Student_{yes}$=4/5=0.8
The influence factor is more for the attribute value 'No'.
CREDIT_RATING Attribute:

I (CREDIT_ RATING="fair" | Buys_Computer="No")=2/5
I (CREDIT_RATING="excellent" | Buys_Computer="No")=3/5

$CREDIT\_RATING_{yes}$=Max (I (Age="X_l"| Buys_Computer="Yes")) where l varies from 1…*k*
$CREDIT\_RATING_{yes}$=3/5=0.6.
The influence factor is more for the attribute value 'excellent'.

5.5. Step 3

Now the Maximum Influence factor is found by the formula given below:

Maximum influence factor i.e. $MIF\ (A_j) = \text{ChkMax}\ (A_jC_i)$

where A_j = attribute that is currently considered for calculation.
AGE Attribute:

MIF (Age)=ChkMax ($AgeC_i$)
MIF (Age)=3/5=0.6

The numerical value of Age= < 30 for buys_computer='no' is greater. i.e. (0.6 > 0.4) so, age= < 30 is considered and given preference.

Table 7
The resulting maximum influence factor values.

	AGE	INCOME	STUDENT	CREDIT RATING
YES	31–40	Medium	yes	Fair
NO	<30	Low/medium	no	Excellent

Table 8
The final table after executing proposed method.

	AGE	STUDENT
YES	31–40	Yes
NO	<30	No

Table 9
The datasets that possess majority voting problem in the UCI Repository.

Dataset	No. of instances	No. of attributes	Associate tasks
Blood transfusion	748	5	Classification
Teaching assistant evaluation	151	5	Classification
SPECT heart	267	22	Classification
Haberman's survival	306	3	Classification
Contraceptive method choice	1473	9	Classification
Hayes Roth	160	5	Classification
Concrete	1030	9	Classification
Forest-fires	517	13	Classification
Solarflare 1	323	13	Classification
Solarflare 2	1066	13	Classification

Table 10
The datasets that possess majority voting problem in the UCI repository (after discretization).

Dataset	No. of instances	No. of attributes	Associate tasks
Abalone	4177	9	Classification
PIMA-diabetes	768	9	Classification
Flag	194	30	Classification
Glass	214	11	Classification
Housing	506	14	Classification
Image segmentation	210	20	Classification
Ionosphere	351	35	Classification
Iris	150	5	Classification
Liver disorder	345	7	Classification
Parkinson	195	24	Classification
Yeast	1484	10	Classification

INCOME Attribute:

MIF (Income)=ChkMax (IncomeC_i)
MIF (Income)=0

The numerical value of Income="medium" for buys_computer='yes' is equal to Income="medium/low" for buys_computer='yes' (0.4 = 0.4) so, Income is not considered in the decision tree as they cause ambiguity.

STUDENT Attribute:

MIF (Student)=ChkMax (StudentC_i)
MIF (Student)=0.8

The numerical value of Student="no" for buys_computer='no' is greater. i.e. (0.8 > 0.6) so, Student="no" is considered and given preference.

CREDIT_RATING Attribute:

MIF (Credit_rating)=ChkMax (Credit_ratingC_i)
MIF (Credit_rating)=0

The numerical value of Credit_rating="fair" for buys_computer='yes' is equal to Credit_rating="excellent" for buys_computer='no' (0.6 = 0.6) so, Credit_rating is not considered in the decision tree as they cause ambiguity (see Table 7).

The final table is given as Table 8.

Now Test Data=(Age=" <30", Income="high", Student="no", Credit_Rating="fair"). Here we compare the test data with the final table (see Table 8). Only the attribute values Age=" <30" and Student="no" corresponds correctly. The remaining attributes i.e. income and credit rating are not taken into consideration and are thus pruned from the tree. So the final class label value is "No" for the given test data.

6. Experimental results and performance evaluation

The performance of the proposed algorithm was evaluated on artificial and real-world domains. Artificial domains are useful because they allow varying parameters, understanding the specific problems that algorithms exhibit, and testing conjectures. Real-world domain are useful because they come from real-world problems that we do not always understand and are therefore actual problems on which we would like to improve performance. All real-world data sets used are from the UC Irvine repository (Blake & Merz, 2006), which contains more over 177 data sets mostly contributed by researchers in the field of machine learning. The data sets that possess majority voting problem are purportedly chosen

Table 11
Performance of classifiers.

Dataset	C4.5			NB			K-NN			Proposed algorithm		
	CC	NCC	Accuracy	CC	NCC	Accuracy	CC	NCC	Accuracy	CC	NCC	Accuracy
Blood transfusion	154	28	84.61	157	25	86.26	152	30	83.51	155	27	85.16
Solar Flare1	55	26	67.90	46	35	56.79	57	24	70.37	63	18	77.78
Glass	35	18	66.04	15	38	28.30	37	16	69.81	38	15	71.70
Haberman	19	14	59.57	22	11	66.67	18	15	54.54	26	7	78.79
Hayes Roth	27	13	67.50	29	11	72.50	23	17	57.50	31	9	77.50
Liver disorder	55	31	63.95	40	46	46.51	47	29	66.28	66	20	76.74
Parkinson	15	15	50.00	15	15	50.00	10	20	33.33	22	8	73.33
Teaching assistant evaluation	20	17	54.00	26	11	70.27	33	4	89.19	28	9	75.68
Mean			64.20			59.66			65.57			77.09

CC – total number of correctly classified instances.
NCC – total number of incorrectly classified instances.

for experimentation to prove the efficacy of our algorithm i.e. two records having the same attribute values but different class value (see Tables 9 and 10). The experiments measuring, the performance of the proposed algorithm was conducted. The proposed algorithms performance is compared with various existing classification algorithms.

There are commonly four approaches for estimating the accuracy such as using training data, using test data, cross-validation, and percentage splitting (Kohavi, 1995; Witten & Frank, 2005). The evaluation function we use is percentage splitting. Table 11 shows, for each data set, the estimated predictive accuracy of the proposed algorithm verses other classification methods. As one can see from Table 11, the predictive accuracy of the proposed algorithm tends to be better than the accuracy of other traditional classification algorithms.

7. Conclusion

This paper proposes a method to resolve one of the exceptions in basic decision tree induction algorithm. The decision tree is constructed based upon the information gain of the attributes in the training data and the classification is done by applying majority voting to the leaf node. But when this majority voting in the leaf node fails an exception occurs, and then the class label is chose randomly in the traditional decision tree induction algorithm. When the proposed procedure is updated to the traditional decision tree induction algorithm, the exception due to majority voting can be resolved. In the proposed procedure the Influence factor of attributes are found and the tree is pruned based on this factor. When the test data is classified based on this pruned tree the class label can be assigned more accurately than the random assignment by traditional decision tree induction algorithms.

References

Blake, C. L., & Merz, C. J. (2006). UCI repository of machine learning databases. Department of Information and Computer Sciences, Irvine: University of California, <http://www.ics.uci.edu/.mlearn/MLRepository.html>.

Breiman, L., Friedman, J., Olsen, R., & Stone, C. (1984). *Classification and regression trees*. Monterey, CA: Wadsworth and Brooks.

Buntine, W. (1992). Learning classication trees. *Statistics and Computing, 2*, 63–73.

Chipman, Hugh, George, Edward I., & Mcculloch, Robert E. (1998). Bayesian CART model search. *Journal of the American Statistical Association, 93*(443), 935–948.

Hartmann, C. R. P., Varshney, P. K., Mehrotra, K. G., & Gerberich, C. L. (1982). Application of information theory to the construction of efficient decision trees. *IEEE Transactions on Information Theory, 28*, 565–577.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1137–1143.

Kohavi, R. (1996). Scaling up the accuracy of Naive Bayes classifiers: A decision tree hybrid. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 202–207.

Kohavi, R., & Kunz, C. (1997). Option decision trees with majority votes. In *Proceedings of the 14th international conference on machine learning*, Morgan Kaufmann.

Mickens, J., Szummer, M., Narayanan, D., Snitch (2007). Interactive decision trees for troubleshooting misconfigurations. In *Proceedings of second international workshop on tackling computer systems problems with machine learning techniques*.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man–Machine Studies, 27*, 221–234.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufman Publishers.

Utgoff, P. E. (1988). ID5: An Incremental ID3. *Proceedings of the fifth international conference on machine learning*. San Mateo, CA: Morgan Kaufmann Publishers. pp. 107–120.

Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning, 4*, 161–186.

Utgoff, P. E. (1989). Improved training via incremental learning. In *Proceedings of the sixth international workshop on machine learning*. Ithaca, NY, United States.

Utgoff, P. E. (1994). An improved algorithm for incremental induction of decision trees. In *Proceedings of the 11th international conference on machine learning*, pp. 318–325.

Utgoff, P. E. (2004). Decision tree induction based on efficient tree restructuring. International Journal of Machine Learning, Springer, pp. 5–44.

Wang et al. (2004). Improving the performance of decision tree: A hybrid approach. *LNCS, 3288*, 327–335.

Witten, Ian H., & Frank, Eibe (2005). *Data mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufman Publishers.

Zhou & Chen (2002). Hybrid decision tree. *Journal of Knowledge-Based Systems, 15*(8), 515–528.