Available online at www.sciencedirect.com

ScienceDirect

Expert Systems with Applications 34 (2008) 3021–3032

Expert Systems with Applications

www.elsevier.com/locate/eswa

An approach to mining the multi-relational imbalanced database

Chien-I Lee [a], Cheng-Jung Tsai [b,*], Tong-Qin Wu [a], Wei-Pang Yang [c]

[a] Department of Information and Learning Technology, National University of Tainan, Tainan, Taiwan, ROC
[b] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, ROC
[c] Department of Information Management, National Dong Hwa University, Hualien, Taiwan, ROC

Abstract

The class imbalance problem is an important issue in classification of Data mining. For example, in the applications of fraudulent telephone calls, telecommunications management, and rare diagnoses, users would be more interested in the minority than the majority. Although there are many proposed algorithms to solve the imbalanced problem, they are unsuitable to be directly applied on a multi-relational database. Nevertheless, many data nowadays such as financial transactions and medical anamneses are stored in a multi-relational database rather than a single data sheet. On the other hand, the widely used multi-relational classification approaches, such as TILDE, FOIL and CrossMine, are insensitive to handle the imbalanced databases. In this paper, we propose a multi-relational g-mean decision tree algorithm to solve the imbalanced problem in a multi-relational database. As shown in our experiments, our approach can more accurately mine a multi-relational imbalanced database.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Data mining; Classification; Imbalance; Relational database

1. Introduction

Most structural data nowadays, like financial transactions and medical anamneses, are stored in the multi-relational database. A multi-relational database usually consists of numerous *data sheets* and each of them would contain a number of *tuples*. A data sheet is also called a *relation* and there are usually one *target relation* and several *non-target relations* in a multi-relational database. Each tuple in the target relation is composed of several *attribute values* and a *target class*, but a tuple in the non-target relations contains only the attribute values. Therefore, a tuple in the target relation is also called *target tuple*. No matter the target or non-target relation, they all possess a *primary key* and some *foreign keys*. The foreign key usually connects to the primary or foreign key in other relations to make the database operations such as *join* or *select* feasible. There are mainly two kinds of connection between the relations, which are (a) the connections between a primary key K and some foreign keys pointing to K; (b) the connections between two foreign keys K_1 and K_2 which connect to an identical primary key K.

Unfortunately, the classification techniques of traditional data mining, such as decision trees (Quinlan, 1993), neural networks (Mitchell, 1997) and support vector machines (Burges, 1998), are only applied for a singular data sheet. In order to extract out the practical and valuable information from the relational databases, many multi-relational classification approaches had been proposed in recent years. The most widely used category of approaches to multi-relational classification is *Inductive Logic Programming* (Lavrac & Dzeroski, 1994) such as FOIL (Quinlan et al., 1993), Golem (Muggleton & Feng, 1990), Progol (Muggleton, 1995), TILDE (Blockeel, De Raedt, & Ramon, 1998), and CrossMine (Yin, Han, Yang, & Yu, 2004). However, the rules mined by these methods will be guided by the *negative/major tuples*. That is,

* Corresponding author. Tel.: +886 6 2133111x777; fax: +886 6 3017137.
E-mail addresses: leeci@mail.nutn.edu.tw (C.-I. Lee), tsaicj@cis.nctu.edu.tw (C.-J. Tsai), wpyang@cis.nctu.edu.tw, wpyang@mail.ndhu.edu.tw (W.-P. Yang).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.05.048

although it can generate a classifier with high accuracy, it is unable to provide the rules precisely for the *positive/minor tuples*. For some practical applications such as fraudulent telephone calls and rare diagnoses, users would be more interested in the minor tuples. Although there are also many proposed solutions for the imbalanced problem (Japkowicz & Stephen, 2002; Tsai, Lee, Chen, & Yang, 2007), they are unsuitable to be directly applied in a multi-relational database. Thus, in this paper, we propose a multi-relational g-mean decision tree, called Mr.G-Tree, as the new solutions for an imbalanced multi-relational database. As it is named, Mr.G-Tree is a decision-tree-based algorithm. We focus on the decision-tree-based approaches since decision tree has several advantages (Rastogi & Shim, 2000), which are (a) decision tree is more efficient for large training data than neural networks which would spend a lots time on thousands of iterations; (b) a decision tree algorithm does not require the domain knowledge or prior knowledge; (c) decision tree displays the good classification accuracy compared to other techniques. For clear explanation, in the paper we will use "the positive" or "the minority" to denote the minor but interesting tuples, and on the contrary "the negative" or "the majority" to represent the major but trivial target classes in a database.

The remainder of this paper is organized as follows. Section 2 is the review of the related works. In Section 3, multi-relational g-mean decision tree will be introduced. The performance evaluation of will be shown in Section 4. Finally, the conclusion and future research directions will be demonstrated in Section 5.

2. Related work

In this section, we first introduce the main principle for inducing a decision tree in Section 2.1. We then review some approaches which were proposed to mine a multi-relational database in Section 2.2. The imbalanced problem will be discussed in Section 2.3.

2.1. Decision tree

A decision tree (Han & Kamber, 2001; Quinlan, 1993) is a flow-chart-like tree structure, which is constructed by a recursive divide-and-conquer algorithm. In a decision tree, each *internal node* denotes a *test* on an attributes, each branch represents an outcome of the test, and each *leaf node* has an associated *target class*. The top-most node in a tree is called *root* and each path from the root to a leaf node represent a *rule*. A typical decision *tree* is shown in Fig. 1. To classify an unknown example, beginning with the root node, successive internal nodes are visited until this example reaches a leaf node. The class of this leaf node is then assigned to this example as a prediction. For instance, the decision in Fig. 1 will approve a golden credit card application if the applicant has a salary higher than 85000 and his repayment record is good.

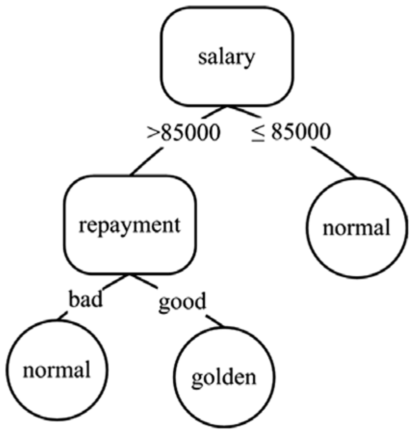

Fig. 1. A typical decision tree.

2.2. Multi-relational classifier

In the current field of the multi-relational classification, the most common one is Inductive Logic Programming (ILP). The formal definition of the ILP problem is:

"Given background knowledge B, positive tuples TP, and negative tuples TN, find a hypothesis H, which is a set of Horn clauses, such that $\forall tp \in TP : H \cup B| = tp$ and $\forall tn \in TN : H \cup B| \neq tn$".

The well known ILP systems are FOIL (Quinlan et al., 1993), Golem (Muggleton & Feng, 1990), Progol (Muggleton, 1995), TILDE (Blockeel et al., 1998), and CrossMine (Yin et al., 2004). First-order inductive learning (FOIL) takes CN2 algorithm (Clark & Niblett, 1989) as the basis and applies top-down and general-to-specific search to establish numerous rules. Each rule will include as many the positive as possible. On the contrary, Golem adopts bottom-up and specific-to-general search, it uses the techniques of RLGG (Relative Least General Generalization) to undertake generalization among a number of specific rules. Regarding with Progol, it is based on the AQ algorithm (Michalski, 1969) and integrates the searching methods in FOIL and Golem. However, higher calculating cost will be caused when the database is enlarged as the common disadvantage for traditional ILP approaches.

In order to reduce the time cost, TOP-down induction of first order logical decision trees (TILDE) is proposed. TILDE is a binary logical decision tree and based on the C4.5 algorithm. Experiments also show that its accuracy is definitely better than the traditional ILP approaches (Boström, 1995). However, TILDE is inapplicable the large-scale databases. To solve the scalability problem, based on FOIL, CrossMine algorithm (Yin et al., 2004) is proposed. In order to reduce the requirement of memory, CrossMine virtually joins target relation and non-target relation together. It propagates the primary key and the target class in target relation to all non-target relations. Accordingly, there are two additional columns "IDs" and "class labels" in each non-target relation. By this propagation approach, CrossMine can calculate the *foil gain* in each relation without joining all the relations into an individual table and therefore the memory cost can be well reduced. Owing to the well consideration of connections among all relations, its accuracy performs much better than FOIL.

2.3. Class imbalanced problem

The proposed techniques aiming at the class imbalanced problem so far could be classified into three categories as follows (Barandela, Sanchez, Garcia, & Rangel, 2003):

2.3.1. Sampling-based

Over-sampling and *under-sampling* are two main techniques in this category. Over-sampling could be further classified into *random over-sampling* and *focused over-sampling* (Aha, Kibler, & Albert, 1991; Han, Wang, & Mao, 2005). Random over-sampling approach over-samples the minority class at random until it matches the size of the majority class. Focused over-sampling approach over-samples the minority class only with data close to the boundaries between the minority class and the majority class. Similarly, under-sampling could be also classified into *random under-sampling* and *focused under-sampling* (Dehmeshki, Karakoy, & Casique, 2003; Derouin, Brown, Beck, Fausett, & Schneider, 1991; Lewis & and Catlett, 1997). The former approach removes the majority class at random until it contains as many examples as the minority class, and the latter one removes the majority examples lying further away. The main idea of focused under-sampling is to remove the noise or outlier data and to reduce the size of majority class by sampling. The combination of over-sampling and under-sampling has also been proposed (Cohen, Hilario, Sax, Hugonnet, & Geissbühler, 2006; Zhou & Liu, 2006). However, over-sampling will increase the training set size and therefore enlarge the computational burden and the impact of noise data; under-sampling has been proven to be ineffective since it results in excluding some useful information (Barandela et al., 2003; Japkowicz & Stephen, 2002).

2.3.2. Cost-based

Cost-Modifying approach (Pazzani et al., 1994; Zadrozny & Elkan, 2001) reduces the relative misclassification cost of the majority class (or increasing that of the minority class) to make it correspond to the size of the minority class. However, it is hard for a user to assign a proper cost when he/she is unfamiliar with the domain knowledge (Japkowicz & Stephen, 2002).

2.3.3. Imbalance-insensitive

This approach is more attractive and has been proven to be more effective than the above two approaches (Japkowicz & Stephen, 2002). The main idea of this technique is to develop an approach that is insensitive to the imbalance problem. Proposed techniques including example weighting, rule removing, attribute correlation analysis, etc. (Anto, Susumu, & Akira, 2000; Ezawa, Singh, & Norton, 1996; Fawcett & Provost, 1996; Kubat, Holte, & Matwin, 1998; Lawrence, Burns, Back, Tsoi, & Giles, 1998). Among the proposed imbalance-insensitive approaches, some of them are limited to specific dataset and some take a lot of training time due to the natural property of neural network. Comparatively, SHRINK (Kubat et al., 1998) is applicable to most applications with numeric attribute data and eliminates the disadvantage described above. SHRINK was developed by the principle of BRUTE (Riddle, Segal, & Etzioni, 1994), it searches for the most accuracy for not only the minority, but also the majority simultaneously by the use of *g-mean* to reach the best partition. However, its shortcoming is to only take care of numeric attribute and search for the best interval in a single one to each attribute. Once if the positive are distributed in two extremities of attribute values, SHRINK will have a poor accuracy.

3. Multi-relational g-mean decision tree algorithm

Although there are many multi-relational database mining classifiers as described in Section 2.1, they can not solve the imbalanced problem. Furthermore, the proposed algorithms to solve the imbalanced dataset are unsuitable to be directly applied on a multi-relational database either. Thus, in this paper, we propose the Multi-relational g-mean decision tree, called Mr.G-Tree, to solve this problem. Without loss of generality, only the case that there are two target classes in a database will be discussed in our paper. If more than two target classes are involved, the target class with the smallest ratio or the interesting one can be identified as the minority, and the rest will be regarded as the majority.

3.1. G-mean based classification model

The *confusion matrix* in Table 1 was a widely used matrix to illustrate different classification measurements. In this matrix, a means the number of the negative that is accurately classified, b is the number that the negative is classified into the positive, c denotes the number that the positive is classified into the negative, and d signifies the number of the positive that is accurately classified. The traditional accuracy is defined as $(a+d)/(a+b+c+d)$. However, to handle the imbalanced problem, g-mean in Eq. (1) was introduced to represent the

Table 1
The confusion matrix

	Predicted negative	Predicted positive
True negative	*a*	*b*
True positive	*c*	*c*

average proportion for the accuracy of the minority and the majority (Rastogi & Shim, 2000).

$$g = \{(a \times d)/[(a+b) \times (c+d)]\}^{1/2}. \tag{1}$$

As stated in Section 2.3, the shortcomings of SHRINK are (a) it establishes only a best interval for each attribute; (b) it is inapplicable to the categorical attribute. These two properties would reduce the predicted accuracy when there are categorical attributes in the training data and when the minority distribute over several intervals of an attribute (Tsai et al., 2007). In order to build a classifier which can handle the imbalanced data more accurately, based on *g-mean* measurement, we establish a *best interval* for each numeric attribute and a *best subset* for each categorical attribute in every internal node o. Each best interval/best subset of the attribute i denotes a test for a tuple. To give higher importance to a test with smaller error, each test is associated with a weight as in Eq. (2) according to its *g-mean* value g_i

$$w_i = \log[g_i/(1-g_i)]. \tag{2}$$

Then all of them are combined as a splitting function as shown in Function 3 for an internal node of Mr.G-Tree.

$$SF(o) = \sum_i h_i \times w_i, \tag{3}$$

In Eq. (3), $h_i = 1$ if the value of attribute i falls in this interval and $h_i = -1$ otherwise. However, to ensure that all weight is larger than 0, the test with $g_i < 0.5$ is discarded.

The best interval of a numeric attribute is defined as $[1/2(a_{ip} + mina_i), 1/2(a_{iq} + maxa_i)]$, where $mina_i$ and $maxa_i$, respectively denotes the minimal and maximal attribute value of the interval with the maximal *g-mean*, a_{ip} is the attribute value of tuple p with the order prior to the tuple with attribute value $mina_i$, and a_{ip} is the attribute value of tuple q with the order posterior to the tuple with attribute value $maxa_i$. The reason is that when predicting the unseen data, the output h_i of an example with attribute value lies in the $[a_{ip}, mina_i]$ and $[a_{iq}, maxa_i]$ is unclear. To deal with categorical attribute, we adopt *Set Theory* to establish the best subset. First, for each categorical attribute, we classify all tuples by their target classes and establish a corresponding *power set*. Each power set would contain 2^k-2 subsets, where k is the number of values in this categorical attribute. For each subset, the tuples belonging to it are regard as the positive, and are regard as the negative otherwise. Finally, the *g-mean* of each subset is computed and the subset with the maximal *g-mean* is selected as the best subset. However, when a categorical attribute contains many values, the computational cost for a categorical attribute by our approach would be nontrivial. To solve this problem, when the number of values in a categorical attribute is larger than 10, we use a simpler approach. That is, for each value, the tuples belonging to it are regard the positive, and are regard as the negative otherwise. As a result, there are only k subsets when k is larger than 10.

In addition, when a leaf node is not pure, most decision tree algorithms will assign this leaf node a negative class if there are more negative instances in this node. However, such a method may reduce the accuracy of the positive that the users are interested in. Thus, in order to mine the positive more accurately, when assigning the target class to a leaf node, Mr.G-Tree adds in the consideration of the imbalanced problem as shown in Definition 2. Fig. 2 represents the corresponding pseudo code.

Definition 1. Given a training dataset T that includes both the minority and majority. For any leaf node N_i on Mr. G-Tree, assume that NP represents the number of the minority in N_i, NN represents the number of the majority in N_i, TP represents the number of all the majority in T, and TN represents the number of all the majority in T; if $NP \leqslant (NN \times TP)/TN$, then the target class of leaf node N_i will be negative, and vice versa.

Example 1. To make readers understand our approach mentioned in this section more clearly, here we take the single-relational database in Table 2 as an example. This database contains 20 tuples in which 2 tuples are the positive and 18 ones are the negative and the column BMI denotes "body mass index". The Mr.G-Tree trained from Table 2 is illustrated in Fig. 3. In the Node 1 of Fig. 3, the best interval of Attribute "age" is [48, 66] and its g-mean is $[(13/18) \times (2/2)]^{1/2} = 0.85$, consequently the weight of this best interval is $\log[0.85/(1-0.85)] = 0.75$. Similarly, the best interval, g-mean, and weight of Attribute "BMI" is [13, 18], 0.85, and 0.75, respectively. As for the subsets and the corresponding g-mean of Attribute "sport" and "smoking" in Node 1, we detail them in Table 3 in which S(i) denote a subset that contains the attribute-value i. Obviously, the best subset Attribute "sport" is [occasional,

```
Procedure Leaf_Node(n)
Begin
For each leaf node
  If $NP \leq (NN \times TP) / TN$ then
    The target class of this node is negative;
  Else
    The target class of this node is positive;
  End if
End
```

Fig. 2. The pseudo code of the handling of leaf nodes in Mr.G-Tree.

Table 2
A training database

Patient	Age	BMI	Sport	Smoking	Diagnosis
1	41	12	Daily	Yes	Health
2	64	14	Occasional	Yes	Cancer
3	57	17	Occasional	No	Health
4	27	30	Daily	Yes	Health
5	31	25	Daily	Yes	Health
6	35	28	Daily	Yes	Health
7	37	15	Daily	Yes	Health
8	49	17	Never	Yes	Cancer
9	52	15	Never	No	Health
10	60	15	Occasional	No	Health
11	47	28	Daily	Yes	Health
12	23	26	Daily	Yes	Health
13	33	32	Never	Yes	Health
14	28	33	Occasional	Yes	Health
15	45	19	Never	Yes	Health
16	68	26	Occasional	Yes	Health
17	72	19	Never	Yes	Health
18	62	21	Daily	Yes	Health
19	56	16	Occasional	No	Health
20	24	20	Occasional	Yes	Health

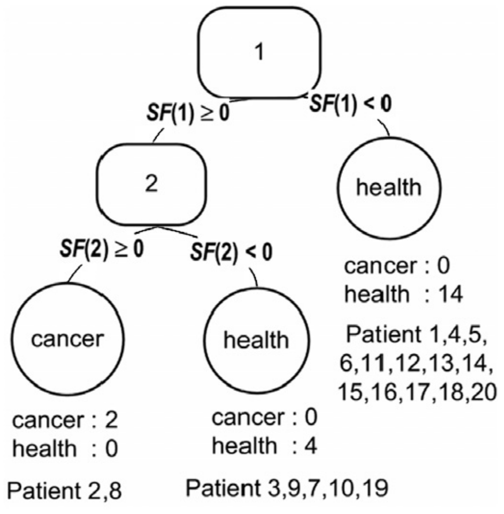

Fig. 3. The classification model constructed by using Table 2.

never] and the g-mean and weight of this subset is $[(8/18) \times (2/2)]^{1/2} = 0.67$ and $\log[0.67/(1-0.67)] = 0.31$, respectively. Note that since the g-mean of the best subset S(yes) in Attribute "smoking" is less than 0.5, it is not used in Node 1. Finally, we can get the splitting function SF(o) in each node o in Fig. 3 is

Table 3
The power set and corresponding *g-mean* of Attribute (a) "sport" and (b) "smoking" in Node 1 of Fig. 3

(a) Subset	S(d, o)	S(o, n)	S(d, n)	S(d)	S(o)	S(n)
G-mean	$\sqrt{\frac{4}{18}\times\frac{1}{2}}$	$\sqrt{\frac{8}{18}\times\frac{2}{2}}$	$\sqrt{\frac{6}{18}\times\frac{1}{2}}$	$\sqrt{\frac{9}{18}\times\frac{0}{2}}$	$\sqrt{\frac{12}{18}\times\frac{1}{2}}$	$\sqrt{\frac{14}{18}\times\frac{1}{2}}$
(b) Subset			S(yes)			S(no)
G-mean			$\sqrt{\frac{4}{18}\times\frac{2}{2}}$			$\sqrt{\frac{10}{18}\times\frac{0}{2}}$

$$SF(1):\ 0.75 \times h_{\text{age}}[48, 66] + 0.75 \times h_{\text{BMI}}\ [13, 18] + 0.31 \times h_{\text{sport}}[\text{occasional, never}];$$
$$SF(2):\ 0.66 \times h_{\text{smoking}}[\text{yes}].$$

3.2. Handling an imbalanced multi-relational database

In order to mine the multi-relational database and avoid the shortcomings of waste memory by joining all datasheets to a single one, Mr.G-Tree extended the concepts of propagation introduced in CrossMine to virtually link the target relation and non-target relation together. Yet, CrossMine will generate redundant target classes in each non-target relation, which will result in the incorrect g-mean calculated. For example, assuming there are two tuples t_1 and t_2 in target relation R_1 and both of them link to the tuple a_1 in non-target relation R_2. If applying CrossMine, tuple a_1 will possess two target classes simultaneously, affecting the original distribution of target classes in non-target relation R_2. If g-mean is calculated by such propagation outcomes, the unreal values for all attributes in R_2 will be obtained owing to two target classes of a_1 are considered. Take the attribute "smoking" in Table 4 as the example, there are only 9 tuples in non-target relation "Physical", but if taking the propagation method proposed in CrossMine, the number of tuples will be increased to 18 and then the original class distribution in non-target relation "Physical" will be changed.

In order to solve this problem, Mr.G-Tree introduces the g-mean Tuple ID propagation algorithm, also known as GTIP algorithm. GTIP maintains the original data distribution in each non-target relation by restoring the number of target classes of each tuple to a single one as described in Definition 2.

Definition 2. Given a non-target relation R' and assume that each tuple t_i in R' contain two additional columns "IDs" and "class labels" after the propagation. If the class labels of a tuple are all the positive, this tuple will be regarded as a positive tuple; if the class labels of a tuple are all the negative, it will be regarded as a negative tuple; if tuple t_i contains both positive and negative class labels, this tuple would contains $CP/(CP+CN)$ positive class and $CN/(CP+CN)$ negative class, where CP and CN, respectively represents the numbers of the positive and that of the negative in the class labels column.

By Definition 2, GTIP can prevent the original distribution of target class in non-target relation from the distortion and the inaccurate g-mean. Fig. 4 is the pseudo code of GTIP algorithm. Here we again take Table 4c as an example. The result of GTIP is shown in Table 5.

In addition, *semantic links* is always a major issue in the research field of Multi-relational database. The research shows that the relations with longer semantic links will become less important in the Multi-relational database (Liu, Yin, & Han, 2005). Thus, in order to design a classifier suitable for the multi-relational database, Mr.G-Tree

Table 4
A relational database in which (a) the connection between two relations "Patient" and "Physical"; (b) the details of relation "Patient"; (c) the details of relation "Physical"

(a) Patient		Physical
Patient-id		Physical-id
Condition-id	→	Condition-id
Age		BMP
Sport		Smoking
Class label		

(b) Patient

Patient-id	Condition-id	Age	Sport	Class
1	12	41	Daily	Health
2	14	64	Occasional	Cancer
3	17	57	Occasional	Health
4	26	27	Daily	Health
5	26	31	Daily	Health
6	28	35	Daily	Health
7	15	37	Daily	Health
8	17	49	Never	Cancer
9	15	52	Never	Health
10	15	60	Occasional	Health
11	24	47	Daily	Health
12	24	23	Daily	Health
13	24	33	Never	Health
14	33	28	Occasional	Health
15	19	45	Never	Health
16	26	68	Occasional	Health
17	19	72	Never	Health
18	14	62	Daily	Health
19	16	56	Occasional	Health
20	20	24	Occasional	Health

(c) Physical

Physical-id	Condition-id	BMP	Smoking
21	24	22	Yes
22	14	16	Yes
23	17	16	Yes
24	14	18	Yes
25	15	24	No
26	15	21	No
27	28	20	No
28	28	25	No
29	26	14	No

takes the semantic link into account and defines a non-target relation as the *effective relation* or *non-effective relation* as shown in Definition 3. While constructing each node, Mr.G-Tree will only propagate the IDs and class labels to an effective relation and then calculate the best intervals/subsets of all attributes in the effective relation. In other words, the tuples used to build the node in an upper layer will be closer to the target relation, which can solve the problem of weaker semantic links.

Definition 3. Assume that the length of semantic links between a non-target relation and the target relation is l, and the depth of Mr.G-Tree is d (the depth of root node is 0). While constructing a node in k-layer of Mr.G-Tree, a non-target relation is defined as an effective relation if $l \leqslant d$, and is defined as a non-effective relation otherwise.

3.3. The detailed step of Mr.G-Tree

Integrating the methods mentioned in Sections 3.1 and 3.2, we detail the steps of multi-relational g-mean decision tree as follows. Fig. 5 is the pseudo code of Mr.G-Tree.

1. Performing GTIP algorithm to all effective relation.
2. Calculating the best interval or best subset of each attribute in every effective relation and compute the g-mean.

```
Procedure GTIP(R')
Begin
  For each non-effective relation R' /* a non-effective will be defined in Definition 3
        For each primary key and foreign-key k in R'
              If R' can connect to a relation R that contain the column "class labels" via
                R'.k then
                    Propagate IDs and class labels from R to R';
              End if
  For each tuple $t_i$ in relation R'
        Assign tuple $t_i$ with CP / (CP+CN) positive class and CN / (CP+CN) negative
class;
        Return the new target class to each tuple;
End
```

Fig. 4. The pseudo code of g-mean tuple ID propagation algorithm.

Table 5
The propagation results of Table 4c by using GTIP

Physical					
Physical-id	Condition-id	BMP	Smoking	IDs	Class labels
21	24	22	Yes	11, 12, 13	c:0, h:1
22	14	16	Yes	2, 18	c:1/2, h:1/2
23	17	16	Yes	3, 8	c:1/2, h:1/2
24	14	18	Yes	2, 18	c:1/2, h:1/2
25	15	24	No	7, 9, 10	c:0, h:1
26	15	21	No	7, 9, 10	c:0, h:1
27	28	20	No	6	c:0, h:1
28	28	25	No	6	c:0, h:1
29	26	14	No	4, 5, 16	c:0, h:1

3. Discarding the attribute if its best g-mean is less than 0.5.
4. Calculating the weight of every attribute as $w_i = \log[g_i/(1-g_i)]$.
5. Let $h_i$denote the output function of attribute i. If an attribute value lies in the best interval then $h_i = 1$; $h_i = -1$ otherwise.
6. Combining the output function and weight of all the attributes to generate the splitting function in each node o as $\mathrm{SF}(o) = \sum_i h_i \times w_i$.
7. A tuple is allotted to the left child node if the SF is greater than 0; otherwise, to the right one.
8. When other relations are able to link to an effective relation through primary key and foreign key, the data in the column of IDs and class labels will be

```
Procedure Traintree($D$)
Begin
    Initial Mr. G-Tree's current depth $d = 0$;
    If $D$ is pure or each weight $w_i = 0$ in $D$ then
        Return;
    Else
        If the Mr.G-Tree get deeper then $d = d + 1$;
        Initial $Left\_D = \phi$ , $Right\_D = \phi$;
    End if
    For each non-effective relation $R'$
        Call GTIP($R'$);
    For each effective relation $R$
        For every attribute $i$
            If the attribute is numeric then
                Sort all examples according the value of this attribute;
                For each positive example $j$
                    Calculate the interval;
                    Calculate the g-mean $g_i$ of the interval [1/2($a_{ip}$ + $mina_i$), 1/2($a_{iq}$ + $maxa_i$));
                    Select the best interval whose $g_i$ is the maximal;
                        If $g_i < 0.5$ then discard this attribute;
                        Elseif $g_i = 1$ then $w_i = 1$;
                        Else $w_i = \log[g_i / (1-g_i)]$
                        End if
            Else
                Building the corresponding power set;
                For each subset
                    Classify all examples to the corresponding subset by their class;
                    Calculate the g-mean $g_i$ of this subset;
                    Select the best subset whose $g_i$ is the maximal;
                        If $g_i < 0.5$ then discard this attribute;
                        Elseif $g_i = 1$ then $w_i = 1$;
                        Else $w_i = \log[g_i / (1-g_i)]$
                        End if
            End if
            Classification function $SF = \ h_i \times w_i$;
    For each training example $t \in D$
        If $SF(t) \geq 0$ then
            $Left\_D = Left\_D \cup t$;
        Else
            $Right\_D = Right\_D \cup t$;
        End if
    Traintree ($Left\_D$);
    Traintree ($Right\_D$);
    Call Leaf_Node($D$);
End
```

Fig. 5. The pseudo code of Mr.G-Tree.

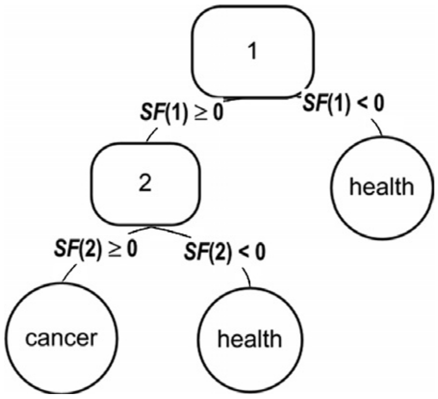

Fig. 6. The Mr.G-Tree constructed by using the relational database in Table 4.

propagated to this relation. After that, setting the relation as the effective one.

9. Repeating the steps from 1 to 8 until all the nodes are pure or can not be further spilt.
10. Assigning the target class to each leaf node according to Definition 2.

3.4. An example of Mr.G-Tree

Here, we clearly introduce the process of Mr.G-Tree by using Table 4 again. In the beginning, Mr.G-Tree calculates the best interval/subset and the corresponding g-mean of Attribute "age" and "sport" in the effective relation "Patient". As described in Example 1, the best interval of the numeric attribute age sport is $[48,66]$ and the corresponding *g-mean* is 0.85; as for the categorical attribute "sport", its best subset is [occasional, never] and 0.67 for *g-mean*. The weight for those two attributes is respectively $w_{\text{age}} = 0.75$ and $w_{\text{sport}} = 0.31$. Hence, the splitting function in the root node of Mr.G-Tree is

$$\mathrm{SF}(1) = 0.75 \times h_{\text{age}}[48,66] + 0.31 \times h_{\text{sport}}[\text{occasional}, \text{never}].$$

Since the non-target relation "Physical" can link to effective relation "Patient" through foreign key "condition-id", Mr.G-Tree will propagate the data in the column "IDs" and "class labels" to the relation "Physical" in the next loop. The propagation results have been shown in Table 5. In the meanwhile, the relation "Physical" will be set as effective one. Next, Mr.G-Tree calculates the *g-mean* of Attribute "age", "sport", "BMP", and "smoking" from all effective relation. The result comes that the *g-mean* of Attribute "age" and "sport" are both less than 0.5, so the two attributes are discarded. However, the best interval of the numeric attribute "BMP" is $[15,19]$ and the corresponding *g-mean* is 0.84; as for the categorical attribute "smoking", its best subset is [yes] and 0.77 for *g-mean*. The weight for those two attributes is $w_{\text{smoking}} = 0.52$ and $w_{\text{BMP}} = 0.72$, respectively. Hence, the splitting function in Node 2 of Mr.G-Tree is

$$\mathrm{SF}(2) = 0.52 \times h_{\text{smoking}}[\text{yes}] + 0.72 \times h_{\text{BMP}}[15,19].$$

Table 6
The details of two real multi-relational databases

	Id	
Database	Financial (1)	Mutagenesis (2)
# Non-target relation	7	3
# Attribute in target relation	4	4
# Positive tuples in target relation	76	64
# Total tuples in target relation	400	2
# Tuples in all relations	75,982	15,218

Finally, taking use of the rules described in Definition 2 to assign the target class to all leaf nodes, we can get the Mr.G-Tree as illustrated in Fig. 6.

4. Experimental analyses

In this section, the software and hardware environment and the experimental databases for our experiments will be shown in Section 4.1. The measurement for evaluation of an imbalanced database will be introduced in Section 4.2. Finally, the comparisons among TILDE, CrossMine and Mr.G-Tree are illustrated in Sections 4.3 and Section 4.4.

4.1. The experimental environment and databases

We implemented our Mr.G-Tree and two state-of-the-art multi-relational classifiers TILDE and CrossMine in Microsoft Visual C++ 6.0 for performance analysis. All experiments in this paper are implemented under the Windows 2000 professional on a PC equipped with Intel Pentium IV 3.0 GHz CPU and 512 MB DDR memory. Our experimental databases can be divided into 2 parts as follows.

4.1.1. Real multi-relational databases

Two real multi-relational databases: Financial database and Mutagenesis database which had been used in CrossMine (Yin et al., 2004) are again applied. Financial database includes 1 target relation, 7 non-target relations and 75982 tuples; Mutagenesis database contains 1 target rela-

Table 7
The parameters of synthetic multi-relational database generator

Name	Description	Parameter
$\|R\|$	Number of relation	x
T	Number of tuples in each relation	y
Imbalance	The percentage of positive tuples in target relation	$z\%$
$T_{\min}$	Minimal number of tuples in each relation	50
$A_{\min}$	Minimal number of attributes in each relation	2
$A_{\max}$	Maximal number of attributes in each relation	5
$V_{\min}$	Minimal value in each attribute	2
$V_{\max}$	Maximal value in each relation	1000
$F_{\min}$	Minimal number of foreign keys in each relation	2
$F_{\max}$	Maximal number of foreign keys in each relation	5

tion, 3 non-target relations and 15,218 tuples. The details of the two databases are listed in Table 6.

4.1.2. Synthetic multi-relational databases

To have further experimental analysis, we code a data generator to generate proper synthetic multi-relational databases (Quinlan et al., 1993). Each synthetic multi-relational database contains one target relation and several non-target relations. Table 7 is the related parameter of our data generator. Five parameters: $T_{\min}$, $A_{\min}$, $A_{\max}$, $V_{\min}$ and $F_{\min}$, will be fixed by referencing the settings applied in CrossMine.

4.2. The measurement

Since the accuracy which is widely used to evaluate a classifier is not a proper metric (Joshi, 2002); in this paper, we use the accuracy of the minority as shown in Eq. (4), where c and d have been described in Table 1, to evaluate the accuracy of a classifier on the minority.

$$\text{accuracy}^{+} = d/(c+d), \tag{4}$$

Furthermore, Z_R and Z_P (Joshi, 2002) in Eqs. (5) and (6) are utilized to evaluate the overall accuracy between Classifier A and B. In Eqs. (5) and (6), R_A, R_B, P_A, and P_B, respectively represents the recall of Classifier A, the recall of Classifier B, the precision of A and that of B; n^c denotes the total number of the positive, n_0^A is the number of the positive predicted by A, n_0^B is the number of the positive predicted of B, $R = (R_A + R_B)/2$, and $P = [(n_0^A \times P_A) + (n_0^B \times P_B)]/(n_0^A + n_0^B)$.

$$Z_R = \frac{R_A - R_B}{\sqrt{2R(1-R)/n^c}} \tag{5}$$

$$Z_P = \frac{P_A - P_B}{\sqrt{P(1-P)(1/n_0^A + 1/n_0^B)}} \tag{6}$$

If $Z_R \geqslant 1.96$, the overall accuracy of R_A is better than R_B(denoted by $R_A \gg R_B$). If $|Z_R| < -1.96$, the overall accuracy of R_B is better than R_A(denoted by $R_A \ll R_B$). If $|Z_R|$<1.96, there is no significant difference between Classifier A and B and will be denoted by $R_A \sim R_B$. The way to use the Z_P to evaluate two classifiers is similar. By using Z_R and Z_P, we can compare two classifiers in the aspect of overall accuracy as follows.

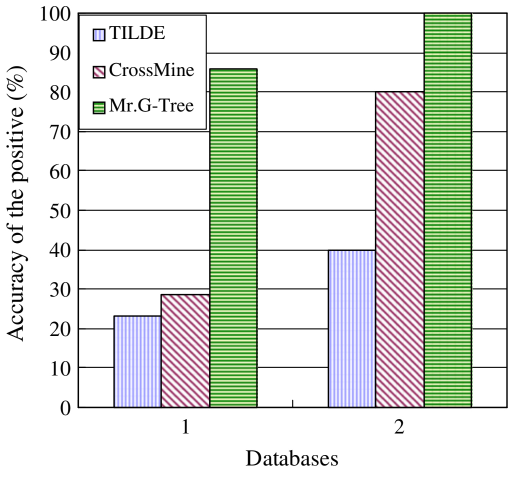

Fig. 7. The comparison of the accuracy of the positive in two real relational databases.

(a) If $(R_A \gg R_B$ and $P_A \sim P_B)$ or $(R_A \gg R_B$ and $P_A \gg P_B)$ or $(R_A \sim R_B$ and $P_A \gg P_B)$, then the overall accuracy of Classifier A is better than the Classifier B $(A > B)$.
(b) The method of judgment for $(B > A)$ is identical to the one mentioned above.
(c) If $(R_A \sim R_B$ and $P_A \sim P_B)$, that will be $A \sim B$.
(d) It may happen that one metric is significantly better but the other is significantly worse. If $(R_A \gg R_B$ and $P_A \ll P_B)$ or $(R_A \ll R_B$ and $P_A \gg P_B)$, *F-measure* as shown in Eq. (7) would be utilized as the way for the accuracy judgment. However, since F-measure does not have any probabilistic interpretation, we cannot apply any significance test to its value. So we use a heuristic such that the improvement in F-measure by at least 1% is required to call a classifier is better.

$$\textit{F-measure} = (2 \times P_A \times R_A)/(P_A \times R_A) \tag{7}$$

4.3. The comparison among TILDE, CrossMine and Mr.G-Tree on real multi-relational databases

In this section, we apply two real multi-relational databases to compare the accuracy of the positive and the overall accuracy among TILDE, CrossMine and Mr. G-Tree. Fig. 7 is the comparison of the accuracy of the positive in two real relational databases and Table 8 is the comparison of overall accuracy. In Table 8, the comparison is marked in bold if the Condition d mentioned

Table 8
The comparison of overall accuracy in two real relational databases

Database-id	Classifier	Z_R	Z_P	F-measure	Overall accuracy
1	Mr.G vs. TILDE	2.348	−0.722	40 vs. 31.69	Mr.G > TILDE
	Mr.G vs. CrossMine	2.159	−2.149	40 vs. 41.32	Mr.G ~ CrossMine
2	Mr.G vs. TILDE	2.07	−1.181	71.43 vs. 57.14	Mr.G > TILDE
	Mr.G vs. CrossMine	1.054	−0.731	71.43 vs. 69.72	Mr.G ~ CrossMine

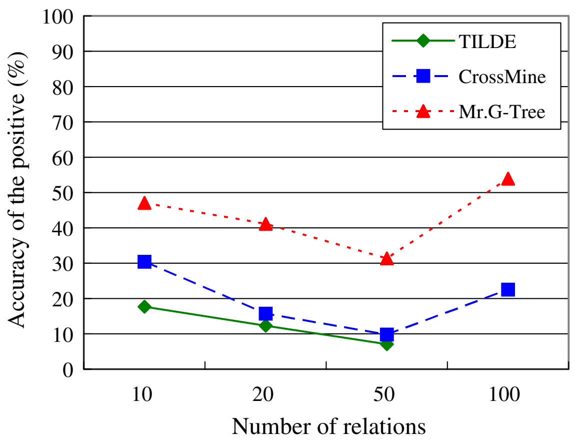

Fig. 8. The comparison of the accuracy of the positive on different number of relations.

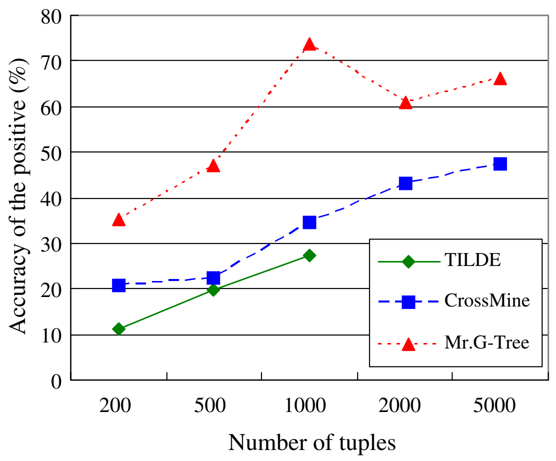

Fig. 9. The comparison of the accuracy of the positive on different number of tuples.

in Section 4.2 is met and consequently the F-measure is used to compare the classifiers. All experiments related to the overall accuracy in the following will be presented in the same way. From Fig. 7, it is clear that Mr.G-Tree performs much better than TILDE and CrossMine in predicting the positive. From Table 8, it can be noted that the overall accuracy of Mr.G-Tree is greater than TILDE and is comparable to CrossMine.

4.4. The comparison among TILDE, CrossMine and Mr.G-Tree on synthetic multi-relational database

In Section 4.3, the number of relations in two real databases is less than 10. However, in a real case, there might be a lot of relations in a multi-relational database. In order to evaluate if the accuracy of each classifier will be influenced when the number of relations is increased, we fix the variable y and z in Table 7, respectively as 500 and 10% to generate 4 synthetic multi-relational databases. Fig. 8 shows the comparison for the accuracy of the positive; Table 9 is the comparison of overall accuracy. Note that, since TILDE combines all relations into a single one, when the number of relations is large, it will cause the shortage of memory space and then be infeasible. Such a condition is occurred in our experiments when the number of relation is over 50 and we mark this condition by "–" in Table 9 since Z_R and Z_P are both incomputable. All experiments related to the comparison of overall accuracy between Mr.G-Tree and TILDE in the following will be presented in the same way. From Fig. 8, it is found that the accuracy of the positive in Mr.G-Tree is always higher than TILDE and CrossMine. In Table 9, it is worth to note that as the number of relations is larger than 10, the overall accuracy of Mr.G-Tree is always better than TILDE and CrossMine.

Next, we turn to fix the variable x and z, respectively as 20 and 10% to generate 5 databases for the analysis of the accuracy in each classifier when the tuples are getting increased. The results are illustrated in Fig. 9 and Table 10. As from Fig. 9, it is found that the accuracy of the positive in Mr.G-Tree is always higher than TILDE and CrossMine no matter how many tuples there are. From Table 10, it is shown that among the overall accuracy of TILDE, CrossMine and Mr.G-Tree, none is always better than the other two. In other words, Mr.G-Tree displays a comparable overall accuracy to TILDE, CrossMine. Similarly, when the tuples in every relation are over 1000, we are unable to show the accuracy of TILDE in Fig. 9 and Table 10.

Table 9
The comparison of overall accuracy on different number of relations

Number of relations	Classifier	Z_R	Z_P	F-measure	Overall accuracy
10	Mr.G vs. TILDE	3.141	−7.487	23.48 vs. 25.34	Mr.G ~ TILDE
	Mr.G vs. CrossMine	1.714	−2.853	23.48 vs.26.11	Mr.G < CrossMine
20	Mr.G vs. TILDE	3.262	0.328	24.13 vs. 13.47	Mr.G > TILDE
	Mr.G vs. CrossMine	2.824	0.754	24.13 vs. 14.12	Mr.G > CrossMine
50	Mr.G vs. TILDE	3.079	−0.708	18.6 vs. 16.49	Mr.G > TILDE
	Mr.G vs. CrossMine	2.67	1.285	18.6 vs. 8.43	Mr.G > CrossMine
100	Mr.G vs. TILDE	–	–	36.78 vs. –	Mr.G > TILDE
	Mr.G vs. CrossMine	3.231	1.22	36.78 vs. 20.74	Mr.G > CrossMine

Table 10
The comparison of overall accuracy on different number of tuples

Number of tuples	Classifier	Z_R	Z_P	F-measure	Overall accuracy
200	Mr.G vs. TILDE	1.803	−0.663	20.3 vs. 14.99	Mr.G ~ TILDE
	Mr.G vs. CrossMine	1.035	−0.152	20.3 vs.17.74	Mr.G ~ CrossMine
500	Mr.G vs. TILDE	2.881	−0.167	28.78 vs. 16.82	Mr.G > TILDE
	Mr.G vs. CrossMine	2.582	0.091	28.78 vs. 21.23	Mr.G > CrossMine
1000	Mr.G vs. TILDE	6.548	−2.105	20.83 vs. 21.53	Mr.G ~ TILDE
	Mr.G vs. CrossMine	5.521	−2.861	20.83 vs. 22.66	Mr.G ~ CrossMine
2000	Mr.G vs. TILDE	–	–	24.52 vs. –	Mr.G > TILDE
	Mr.G vs. CrossMine	3.523	−4.246	24.52 vs. 26.2	Mr.G ~ CrossMine
5000	Mr.G vs. TILDE	–	–	21.19 vs. –	Mr.G > TILDE
	Mr.G vs. CrossMine	6.065	−11.77	21.49 vs. 32.75	Mr.G > CrossMine

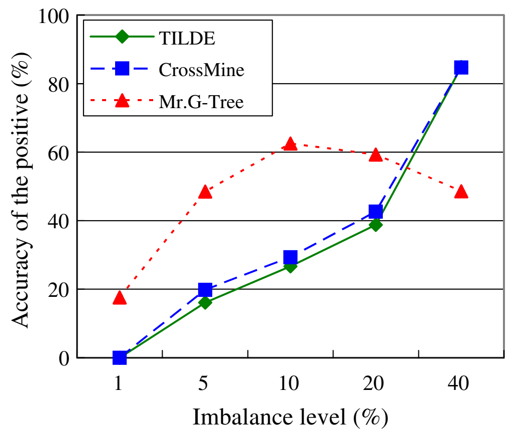

Fig. 10. The comparison of the accuracy of the positive on different imbalanced levels.

Ultimately, for the evaluation of the imbalanced problem, we fix variable x and y in Table 8, respectively as 20 and 500 to produce 5 databases. The corresponding imbalanced level of 5 databases is 1%, 5%, 10%, 20% and 40%. As illustrated in Fig. 10, the accuracy of the positive in Mr.G-Tree is better than TILDE and CrossMine apart from 40% as its imbalanced level. This result is not surprised and demonstrates the ability of Mr.G-Tree to mine an imbalanced multi-relational database. Table 11 shows that among the overall accuracy of TILDE, CrossMine and Mr.G-Tree, none is always superior to the other two.

5. Conclusion and future research directions

Currently, the structural data are all stored in the relational database, so the techniques of traditional data mining are not workable any more. Although there have been a number of multi-relational data mining classifiers proposed, such as TILDE, FOIL, CrossMine and so on, those approaches are unable to well handle the imbalanced problem. Hence, Mr.G-Tree algorithm is proposed in this paper as the solutions. In order to build a classifier which can handle the imbalanced problem more accurately, Mr. G-Tree hierarchically pick up the positive by using g-mean mentioned in Section 3.1. Then, Mr.G-Tree makes the mining in a multi-relational database feasible by using GTIP algorithm proposed in Section 3.2 to propagate necessary information to all non-target relations. More importantly, GTIP algorithm keeps the g-mean reasonable after the

Table 11
The comparison of overall accuracy on different imbalanced levels

Imbalance (%)	Classifier	Z_R	Z_P	F-measure	Overall accuracy
1	Mr.G vs. TILDE	0.982	−0.843	10.44 vs. 8.69	Mr.G > TILDE
	Mr.G vs. CrossMine	0.982	−0.713	10.44 vs.8.13	Mr.G > CrossMine
5	Mr.G vs. TILDE	2.45	−4.358	23.01 vs. 27.14	Mr.G < TILDE
	Mr.G vs. CrossMine	2.139	−2.874	23.01 vs. 24.42	Mr.G ~ CrossMine
10	Mr.G vs. TILDE	3.601	−6.073	37.66 vs. 42.15	Mr.G < TILDE
	Mr.G vs. CrossMine	3.331	−4.135	37.66 vs. 39.97	Mr.G ~ CrossMine
20	Mr.G vs. TILDE	2.9	−6.288	46.15.vs. 53.35	Mr.G < TILDE
	Mr.G vs. CrossMine	2.362	−3.835	46.15 vs. 49.34	Mr.G ~ CrossMine
40	Mr.G vs. TILDE	−7.681	0.122	62.14 vs. 85.29	Mr.G < TILDE
	Mr.G vs. CrossMine	−7.66	1.636	62.14 vs. 81.68	Mr.G < CrossMine

propagation. Finally, by taking the semantic link into account, Mr.G-Tree can accurately and efficiently mine the imbalanced multi-relational databases. In the aspect of experimental analysis, since the accuracy which was widely used to evaluate a classifier is not a proper metric for the imbalanced datasets; in this paper, not only the accuracy of the positive is applied to evaluate the prediction ability of a classifier on the minority, but also Z_R and Z_P and *F-measure* is utilized to evaluate the overall accuracy. The results shown in Section 4 demonstrate that Mr.G-Tree can reach better classification accuracy of the positive than TILDE and CrossMine. Also, it shows comparable consequence on the overall accuracy.

However, Mr.G-Tree currently is only suitable for the database containing two target classes. Although Mr.G-Tree can set the most minor target class as the positive and the rest as the negative under the condition of multiple target classes, this way will change the original data distribution and consequently influences the mining results. Therefore, we will expand Mr.G-Tree in future to make it can accurately manage the databases contain multiple target classes. Another line of future work is incremental Mr.G-Tree that considers the additions, deletions, and updates of tuples.

References

Aha, D., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning, 6*(1), 37–66.

Anto, S. N., Susumu, K., & Akira, I. (2000). Fog forecasting using self growing neural network CombNET-II – A solution for imbalanced training sets problem. In *Proceedings of the international joint conference on neural networks* (pp. 429–434). Italy.

Barandela, R., Sanchez, J. S., Garcia, V., & Rangel, E. (2003). Strategies for learning in class imbalance problems. *Pattern Recognition, 36*(3), 849–851.

Blockeel, H., De Raedt, L., & Ramon, J. (1998). Top-down induction of logical decision trees. *Artificial Intelligence, 101*, 285–297.

Boström, H. (1995). Covering vs. divide-and-conquer for top-down induction of logic programs. In *Proceedings of the fourteenth international joint conference on artificial intelligence* (pp. 1194–1200). San Mateo.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*, 121–168.

Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3*(4), 261–283.

Cohen, G., Hilario, M., Sax, H., Hugonnet, S., & Geissbühler, A. (2006). Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine, 37*(1), 7–18.

Dehmeshki, J., Karakoy, M., & Casique, M. V. (2003). A rule-based scheme for filtering examples from majority class in an imbalanced training set. In *Proceedings of the 13th international conference on machine learning and data mining in pattern recognition* (pp. 215–223). Germany.

Derouin, E., Brown, J., Beck, H., Fausett, L., & Schneider, M. (1991). Neural network training on unequally represented classes. *Intelligent Engineering Systems Through Artificial Neural Networks*, 135–145.

Ezawa, K. J., Singh, M., & Norton, S. W. (1996). Learning goal oriented bayesian networks for telecommunications management. In *Proceedings of the 13th international conference on machine learning* (pp. 139–147). San Francisco.

Fawcett, T., & Provost, F. (1996). Combining data mining and machine learning for effective user profiling. In *Proceedings of 12th international conference on knowledge discovery and data mining* (pp. 8–13).

Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. Morgan Kaufman.

Han, H., Wang, W., & Mao, B. H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *Proceedings of international conference on intelligent computing* (pp. 878–887). Hefei, China.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal, 6*(5), 429–450.

Joshi, M. V. (2002). On evaluating performance of classifiers for rare classes. In *Proceedings of 12th IEEE international conference on data mining*. Japan.

Kubat, M., Holte, R., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning, 30*(2/3), 195–215.

Lavrac, N., & Dzeroski, S. (1994). *Inductive logic programming: techniques and applications*. New York: Ellis Horwood.

Lawrence, S., Burns, I., Back, A. D, Tsoi, A. C., & Giles, C. L. (1998). Neural network classification and prior class probabilities. Neural Networks: Tricks of the Trade 1998, Lecture Notes in Computer Science (pp. 299–314).

Lewis, D. D., & Catlett, J. (1997). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th international conference on machine learning* (pp. 148–156). San Francisco.

Liu, H., Yin, X., & Han, J. (2005). An efficient multi-relational naïve bayesian classifier based on semantic relationship graphs. In *Proceedings of ACM-SIGKDD workshop on multi-relational data mining*. Chicago.

Michalski, R. S. (1969). On the quasi-minimal solution of the general covering problem. In *Proceedings of the 5th international symposium on information processing* (pp. 125–128). Yugoslavia.

Mitchell, T. M. (1997). *Machine learning*. McGraw Hill.

Muggleton, S., & Feng, C. (1990). Efficient induction of logic programs. In *Proceedings of the 1st conference on algorithmic learning theory* (pp. 368–381). Tokyo.

Muggleton, S. (1995). Inverse entailment and progol. *In New Generation Computing. Special issue on Inductive Logic Programming, 13*, 245–286.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., & Brunk, C. (1994). Reducing misclassification costs. Proceedings of the 11th international conference on machine learning (pp. 217–225).

Quinlan, J. R., & Cameron-Jones, R. M. (1993). FOIL: A midterm report. In *Proceedings of the European conference on machine learning* (pp. 3–20). Vienna, Austria.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgann Kaufmann.

Rastogi, R., & Shim, K. (2000). PUBLIC: A decision tree classifier that integrates building and pruning. *Data Mining and Knowledge Discovery, 4*(4), 315–344.

Riddle, P., Segal, R., & Etzioni, O. (1994). Representation design and brute force induction in a boeing manufacturing domain. *Applied Artificial Intelligence, 8*, 125–147.

Tsai, C. J., Lee, C. I., Chen, C. T., & Yang, W. P. (2007). A multivariate decision tree algorithm to mine imbalanced data. *WSEAS Transactions on Information Science and applications, 4*(1), 50–58.

Yin, X., Han, J., Yang, J., & Yu, P. S. (2004). Crossmine: Efficient classification across multiple database relations. In *Proceedings of the 20th international conference on data engineering* (pp. 399–411). Boston, MA.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the 7th international conference on knowledge discovery and data mining* (pp. 204–213). San Francisco.

Zhou, Z. H., & Liu, X. Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transaction on Knowledge and Data Engineering, 18*(1), 63–77.