# 一個適用於不平衡資料庫的多重關聯分類器

# A Multi-relational Classifier for Imbalanced Database


| 李建億 | 蔡政容 | 吳同欽 |
|---|---|---|
| Chien-I Lee[1] | Cheng-Jung Tsai[2] | Tong-Qin Wu[3] |
| NUTN | NCTU | NUTN |
| leeci@ipx.nutn.edu.tw | tsaicj@cis.nctu.edu.tw | jim@benz.nutn.edu.tw |



## 摘要

現今大多數的結構化資料都是被儲存在關聯式資料庫中，儲存於資料庫中的多個關聯，則經由實體/關聯模型連結在一起。近年來，多重關聯的分類已經被廣泛的應用到許多方面，如財務決策、醫學研究…等。雖然已經有許多的學者提出各種多重關聯資料探勘分類器如 TILDE、FOIL、CrossMine…等，但是在資料集不平衡的情況下，這些多重關聯資料探勘分類器並不能有效的分類出佔少數比例的資料(正例)。因此，本論文提出 Multi-relational G-mean decision Tree (Mr.G-Tree)演算法來解決在多重關聯下各分類器無法正確預測不平衡資料集的問題。從實驗數據中可以發現，Mr.G-Tree 演算法能更正確的探勘多重關聯下之不平衡資料集。



**關鍵詞**：多重關聯資料探勘、分類、不平衡、幾何平均值



## Abstract

Most of today's structured data is stored in relational databases. Multiple relations in a relational database have to be connected via entity/relationship model. Recently multi-relational classification has been widely applied in many aspects, such as financial decision making, medical research etc.. Although a lot of multi-relational data mining classifiers have been proposed, such as TILDE, FOIL, and CrossMine etc., but they are unable to accurately classify minority data (positive data) when the datasets are imbalanced. In this thesis, we propose a Multi-relational G-mean decision Tree algorithm, called Mr.G-Tree, to solve these problems mentioned above. Finally, as shown in the experiment, Mr.G-Tree can accurately classify multi-relational imbalanced dataset.



**Keywords**: Multi-relational Data Mining, Classification, Imbalance, G-mean


## 1. Introduction

傳統資料探勘中的分類技術如類神經網路(neural network)[8]和支持向量機(support vector machine)[1]…等，都只能被應用在單一的資料表格上[27]。然而現今大多數的結構化資料如財務決策、醫學研究…等，都是被儲存在包含有多張資料表格的多重關聯資料庫(multi-relational database)中。因此，如何在龐大的關聯式資料庫中找出有用的資訊愈來愈受到重視。

在多重關聯資料庫中，一個關聯即表示一張資料表格，每張資料表格則含有多筆資料(tuples)。而一個多重關聯資料庫會包含一個目標關聯(target relation)和多個非目標關聯(non-target relation)。其中目標關聯中的每筆資料具有一般屬性值和目標類別值，因此在目標關聯中的資料亦稱為目標資料(target tuples)。而非目標關聯中的資料則只包含一般屬性值，因此必須藉由一些連結的技巧將目標關聯中之目標類別(target class)的資訊複製到各個非目標關聯中。此外，不論是目標關聯或非目標關

聯，都會包含一個主鍵(primary key)及多個外來鍵(foreign keys)，而外來鍵會指向(pointing)其他關聯的主鍵。關聯和關聯之間主要有兩種連接(joins)：

1. 在主鍵 $K$ 和一些指向 $K$ 的外來鍵之間的連接。
2. 在 2 個外來鍵 $K_1$ 及 $K_2$ 之間的連接，且各自指向相同的主鍵 $K$。

在多重關聯資料探勘分類器(multi-relational data mining classifiers)的研究領域中，最常使用的方法是歸納邏輯程序設計(Inductive Logic Programming，ILP) [21,22]，如 FOIL[26]、Golem[24]與 Progol[23]。

目前在分類的領域中，傳統的演算法如 C4.5 等探勘出的規則會被佔有較大比例的資料主導，其所產生的決策樹在整體上雖能達到十分高的預測正確性，但對於佔少數比例的資料卻無法完整且正確地推導出其規則。然而在實際的運用上，有時使用者感興趣的反而會是少量但卻非常重要的資料，如偵測電話詐欺行為、不可靠的電信使用者、工業生產過程中的失誤或延遲、罕見疾病的偵測…等。因此，很多學者提出不同的方法來解決不平衡資料集(imbalance dataset)的問題。

目前解決不平衡之資料集的方法，主要分為兩種類型：平衡資料的方法(balanced imbalances)、不平衡資料的方法(imbalanced imbalances)[20]。

然而，上述解決不平衡資料集的方法皆只適用於單一的資料表格上，並無法直接套用於多重關聯資料庫中，而現存用來解決多重關聯資料庫的分類器亦無法解決不平衡資料集的問題。因此，本論文提出 Multi-relational G-mean decision Tree (Mr.G-Tree)演算法來解決在多重關聯資料庫下不平衡資料集的問題。

在第二章之中，將會針對各種多重關聯資料探勘分類器作詳細的探討，並分析目前處理不平衡之資料集的方法及其優缺點；第三章則提出 Multi-relational G-mean decision Tree (Mr.G-Tree)演算法來解決在多重關聯資料庫下各分類器無法正確預測不平衡資料集的問題；第四章為相關的實驗模擬與效能分析；最後，在第五章則是本論文的結論以及未來的研究方向。

## 2. Related Work

### 2.1 多重關聯資料探勘分類器

目前在多重關聯分類的研究領域中，最常使用的分類法就是歸納邏輯程序設計(Inductive Logic Progamming, ILP)[21,22]，如 FOIL[26]、Golem[24]、Progol[23]。

其中 FOIL(First-Order Inductive Learning)是以 CN2 演算法[29]為基礎，採用由上而下搜尋的方法(top-down general-to-specific search)，建立多條規則，而每條規則都包含許多的正例以及少數的負例。Golem 是採用由下而上搜尋的方法(bottom-up specific-to-general search)，並利用 RLGG(relative least general generalization)的技巧，從多數特定的規則(specific rules)中進行歸納(generalization)。Progol 則是以 AQ 演算法[28]為基礎，並結合 FOIL 以及 Golem 搜尋的方法。然而 ILP 系統共通的缺點，就是當資料庫很大的時候，必須花費很高的系統計算成本(cost)。

TILDE(TOP-down Induction of First order Logical Decision Trees)[18]，是較 ILP 系統後期的演算法，是以 C4.5 演算法[25]為基礎所發展出來的二元邏輯決策樹，而且 TILDE 採用 divide-and-conquer 演算法，所以在準度上較傳統的 ILP 採用 separate-and-conquer 演算法好[11]。

CrossMine 演算法[27]，是以 FOIL 演算法為基礎，並提出 Tuple ID Propagation 演算法來改善記憶體空間浪費的問題，因為其考量到各關聯(relation)之間關聯性的問題，所以準度上較 FOIL 演算法為佳。

### 2.2 CrossMine 演算法

CrossMine 提出了 Tuple ID Propagation 演算法，將目標關聯(target relation)和其他非目標關聯(non-target relations)虛擬連結(virtual join)在一起。其將"目標關聯 $T$ 中的主鍵：T-id"以及"目標類別：class label"增殖(propagation)到其他非目標關聯中，所以在每個非目標關聯中皆多了 IDs、class labels 這兩個欄位。Tuple ID Propagation 增殖的方

法定義如下：

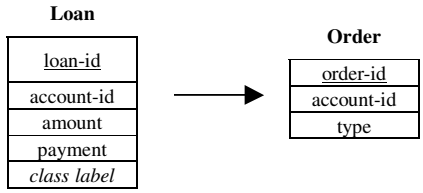
假設一資料集合有二種目標類別 $C_1$、$C_2$，且目標關聯 $R_1$ 和非目標關聯 $R_2$ 可以藉由屬性 $R_1.A$ 和 $R_2.A$ 連結在一起，若 $ID_t$ 表示 $R_1$ 中資料 $t$ 的資料編號，$C_t$ 表示該筆資料的目標類別，則 $R_2$ 中每筆資料 $u$ 可增殖欄位 IDs 和欄位 class labels，其中 IDs = $\{ \bigcup ID_t \mid t \in R_1,\ t.A = u.A \}$, class labels = $\{ \sum (C_t = C_1),\ \sum (C_t = C_2) \}$。

以表 2.1 為例，在非目標關聯 *Order* 中，資料{21}皆可利用 account-id = 124 與目標關聯 *Loan* 中的資料{3, 4, 5, 6, 7, 8, 9}連結在一起，所以資料{21}的 IDs 包括" 3, 4, 5, 6, 7, 8, 9"，class labels 為"0+，7－"；資料{22}的 IDs 為"10"，class labels 為"1+，0－"…等，其增殖的結果如表 2.2 所示。

藉由 Tuple ID Propagation 的方法由目標關聯不斷地外向增殖，所有的非目標關聯都會多出 IDs、class labels 兩個欄位。因此，在不須將所有關聯連結成單一關聯資料表格的情形下，CrossMine 可以計算任意關聯中各個預測的 foil gain。

由於 Tuple ID Propagation 法並不需要很多額外的儲存空間，因此在空間成本的花費上要比實際連結(physical join)來的少。

表 2.1　樣本資料庫(The financial database from PKDD CUP 99.)




| ***Order*** | | |
|---|---|---|
| order-id | account-id | type |
| 21 | 124 | insurance |
| 22 | 79 | insurance |
| 23 | 108 | insurance |
| 24 | 45 | insurance |
| 25 | 60 | loan |
| 26 | 60 | loan |
| 27 | 111 | loan |
| 28 | 111 | loan |
| 29 | 95 | loan |

| ***Loan*** | | | | |
|---|---|---|---|---|
| loan-id | account-id | amount | payment | **class** |
| 1 | 135 | ≦3000 | > 40 | + |
| 2 | 30 | ≦3000 | > 40 | + |
| 3 | 124 | > 3000 | ≦40 | － |
| 4 | 124 | > 3000 | ≦40 | － |
| 5 | 124 | > 3000 | ≦40 | － |
| 6 | 124 | > 3000 | ≦40 | － |
| 7 | 124 | > 3000 | ≦40 | － |
| 8 | 124 | > 3000 | ≦40 | － |
| 9 | 124 | > 3000 | ≦40 | － |
| 10 | 79 | > 3000 | ≦40 | + |
| 11 | 108 | > 3000 | ≦40 | + |
| 12 | 45 | > 3000 | ≦40 | + |
| 13 | 60 | > 3000 | ≦40 | + |
| 14 | 111 | > 3000 | ≦40 | + |
| 15 | 111 | > 3000 | ≦40 | － |
| 16 | 60 | > 3000 | ≦40 | － |
| 17 | 95 | > 3000 | ≦40 | － |
| 18 | 95 | > 3000 | ≦40 | － |
| 19 | 95 | > 3000 | ≦40 | － |
| 20 | 95 | > 3000 | ≦40 | － |

表 2.2　Tuple ID Propagation 增殖的結果

| ***Order*** | | | | |
|---|---|---|---|---|
| order-id | account-id | type | **IDs** | **class labels** |
| 21 | 124 | insurance | 3, 4, 5, 6, 7, 8, 9 | 0+, 7－ |
| 22 | 79 | insurance | 10 | 1+, 0－ |
| 23 | 108 | insurance | 11 | 1+, 0－ |
| 24 | 45 | insurance | 12 | 1+, 0－ |
| 25 | 60 | loan | 13, 16 | 1+, 1－ |
| 26 | 60 | loan | 13, 16 | 1+, 1－ |
| 27 | 111 | loan | 14, 15 | 1+, 1－ |
| 28 | 111 | loan | 14, 15 | 1+, 1－ |
| 29 | 95 | loan | 17, 18, 19, 20 | 0+, 4－ |

## 2.3 不平衡之資料集

而目前解決不平衡之資料集的方法，主要分為兩種類型：

一、平衡資料的方法(balanced imbalances)：

此類方法可再分成兩大類：

1. 敏感學習(cost sensitive learning)：強調正負例要給予不同的錯誤分類成本，以減少整體分類所花費的成本 [4, 5, 7]。此種方法最大的缺點在於須要使用者自行定義不同目標類別的錯誤分類成本，當使用者對於該領域的知識不足時，並無法定義出適當的錯誤成本。

1. 抽樣(sampling)：抽樣的技巧又可分為減少負例(Down-Sizing，reduce negative)、增加正例(Over-Sampling，duplicate positive)、整合減少負例與增加正例三種。減少負例的方法中，Warmuth[10]、Lewis 與 Catlett[15]是藉由抽樣的技術，將大量的負例資料減少。而 Kubat 與 Matwin[14]是將多餘的資料、干擾資料以及邊界資料皆移除的演算法。增加正例的方法中，DeRouin 等人[9]是利用類神經網路的技術來得到人造資料以增加正例。而 Ling 與 Li[6]是利用複製正例的技巧，將正例的個數增加至與負例一樣，其中複製正例的動作，等於是在增加正例的權重。Solberg[17]、Chawla 等人[2]則同時採取減少負例與增加正例的方法。然而，抽樣最大的缺點在於會改變原始資料的分布情形導致預測模組失真。

二、不平衡資料的方法(imbalanced imbalances)：

1. BRUTE[16]：BRUTE 法先藉由一般決策樹演算法產生規則集合後，再依其演算法將規則減少以提高分類正例之規則的正確率。因此，它是一開始即將重心放在正例，也就是說一開始的規則就必須包含全部的正例，之後再以正例的最大正確率作為規則選取的準則。雖然最後產生的規則能將正例之正確率提高，但如果一開始的決策樹演算法無法產生可分類正例之規則時，則 BRUTE 法將無法產生任何作用。

2. SHRINK[12,13]：SHRINK 基於 BRUTE 法的原則，不僅是尋找正例的最大正確率，亦利用幾何平均值(*g-mean*)同時將負例的最大正確率考量進去，以求得其最佳分隔準則。SHRINK 的缺點在於只能處理數值屬性(numeric attribute)，且針對每個屬性只求得一個區間作為最佳區間，如果遇到正例資料分佈於屬性值的兩端時，其分類效果將會大打折扣。

3. His-Tree[19]：其是以 SHRINK 法為基礎，階層式地計算所有可能的最佳區間，並以分類函數(classification function)作為節點分隔的準則。His-Tree 更利用集合(Set)的概念，解決 SHRINK 無法處理類別屬性(categorical attribute)的問題。

## 2.4 His-Tree 演算法

His-Tree 法 [19] 採用混亂矩陣(confusion matrix)，如表 2.3 所示，來定義分類的效能準則。在混亂矩陣中，*a* 是負例分類正確的個數；*b* 是負例分類成正例的個數；*c* 是正例分類成負例的個數；*d* 是正例分類正確的個數。

表 2.3 混亂矩陣(confusion matrix)

| 實際＼預測 | 分類為負例 | 分類為正例 |
|---|---|---|
| 真正的負例 | *a* | *b* |
| 真正的正例 | *c* | *d* |

His-Tree 法使用幾何平均值(*g-mean*)代表正負例的正確率之平均比例，其公式為：

$$g-mean=\sqrt{\frac{a}{a+b}*\frac{d}{c+d}}$$

以表 2.4 為例,His-Tree 法的執行步驟如下：

1、根據數值與類別屬性的分類規則，求得所有屬性的最佳區間或最佳集合，並計算其 *g-mean* 值。

2、假如該屬性的最大 *g-mean* 值小於 0.5，則放棄該屬性。

3、然後，再計算每一個剩下來的屬性之權重

$$w_i = \log_{10}\left(\frac{g_i}{1-g_i}\right)。$$

4、定義 $h_i$ 為第 $i$ 個屬性的輸出函數。$h_i$=1，表示該屬性值落在該屬性的最佳區間內，反之則 $h_i$=-1。

5、計算每筆資料的分類函數值(classification function)：

$$SF = \sum_i h_i * w_i$$

若某筆資料 $p$ 的分類函數值 $SF$ 大於等於 0，則分到左子節點，並分類為正例，反之則分到右子節點，分類為負例。

6、重複步驟 1～5，直到所有節點的資料完全純化(purity)，即分類屬性值完全一樣，或無法繼續分類為止。

表 2.4　Training data tuples from the AllElectronics customer database

| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 1 | 20 | 100000 | yes | high | yes |
| 2 | 45 | 150000 | yes | middle | yes |
| 3 | 56 | 200000 | no | middle | no |
| 4 | 22 | 30000 | yes | high | no |
| 5 | 26 | 45000 | no | high | no |
| 6 | 30 | 28000 | yes | high | no |
| 7 | 31 | 176000 | no | high | no |
| 8 | 39 | 220000 | yes | low | yes |
| 9 | 47 | 143000 | no | low | yes |
| 10 | 58 | 35000 | yes | middle | no |
| 11 | 43 | 80000 | yes | high | no |
| 12 | 16 | 26000 | yes | high | no |
| 13 | 28 | 63200 | no | low | no |
| 14 | 24 | 33000 | yes | middle | no |
| 15 | 41 | 43000 | yes | low | no |
| 16 | 62 | 26400 | no | middle | no |
| 17 | 64 | 20000 | yes | low | no |
| 18 | 60 | 218000 | no | high | no |
| 19 | 55 | 34000 | no | middle | no |
| 20 | 18 | 51200 | yes | middle | no |

圖 2.1 為 His-Tree 法根據表 2.4 的客戶資料庫所建構的決策樹，其所產生的分類函數 $SF(i)$如下所示，其中 $i$ 表示決策樹中的第 $i$ 個節點。

$SF(1) = 0.63\times h_{age}[35, 51] + 0.96\times h_{income}[90000, 220000] + 0.24\times h_{credit\_rating}[\text{low}] + 0.13\times h_{student}[\text{yes}]$；

$SF(2) = 0.38\times h_{age}[20, 40] + 1\times h_{income}[71500, 220000] + 0.38\times h_{credit\_rating}[\text{high, middle}]$。

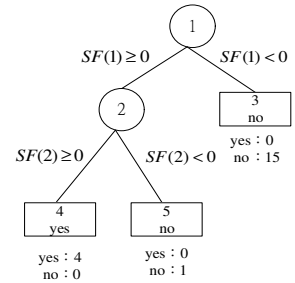



圖 2.1　AllElectronics customer database 之 His-Tree

## 3. 多重關聯 G-mean 決策樹演算法

目前已提出的多重關聯資料探勘分類器(multi-relational data mining classifiers)如 TILDE、FOIL、CrossMine…等，並無法解決資料集不平衡的問題；而傳統解決不平衡資料集的演算法如 His-Tree 亦無法直接套用於多重關聯資料探勘中。因此，本論文以 His-Tree 為基礎，提出一個適用於不平衡資料庫的多重關聯分類器 Multi-relational G-mean decision Tree，簡稱為 Mr.G-Tree。為了方便說明，本論文只探討兩個目標類別(two target class)的情形，若遇到多目標類別(multiple target class)時，則可將資料庫中所佔比例最小的目標類別設為正例，其餘皆設為負例。此外，傳統皆以整體正確率 *accuracy* 做為來評估分類器優劣的準則，以表 2.3 的混亂矩陣為例，其公式為

$$accuracy = \frac{a+d}{a+b+c+d},$$

然而，此一評估準則會被較多的負例個數所主導，因此並不適用於不平衡資料集[3]；所以本論文除

了使用正例正確率

$$accuracy^{+} = \frac{d}{c+d}$$

來評估分類器對於少量資料的預測能力外，亦採用權重正確率(weighted accuracy)[3]

$$w\text{-}accuracy = \frac{1}{2}\left(\frac{a}{a+b}+\frac{d}{c+d}\right)$$

作為評估整體正確率的準則。

## 3.1 G-mean 資料增殖演算法 (G-mean Tuple ID Propagation algorithm)

為了能在多重關聯的資料庫中進行資料探勘，並避免將所有資料表格連結成單一資料表格會浪費記憶體空間的缺點，Mr.G-Tree 延用 CrossMine 增殖的觀念，將目標關聯和其他非目標關聯虛擬連結在一起。然而，CrossMine 資料增殖的方法會在每個關聯中產生目標類別個數增多而改變其原始資料分布的情形，進而造成每個關聯中計算出來之幾何平均值(*g-mean*)失真，所以並無法直接套用於 Mr.G-Tree。舉例來說，假設目標關聯 $R_1$ 中有兩筆資料 $t_1$、$t_2$，可以和非目標關聯 $R_2$ 中的一筆資料 $a_1$ 連結，若採用 CrossMine 的增殖方式，則資料 $a_1$ 會同時具有兩個目標類別，導致非目標關聯 $R_2$ 中目標類別的原始分布會被改變。此時若根據此增殖結果來計算 *g-mean*，則因為資料 $a_1$ 的兩個目標類皆會被列入計算，會進一步造成 $R_2$ 中所有屬性計算得到的 *g-mean* 失真。以表 2.2 中的 type 屬性為例，非目標關聯 Order 中原本只有 9 筆資料，若依 CrossMine 的增殖方式，則資料會增殖變成 22 筆，關聯 Order 中資料之分布也因此隨之改變。此時若將每一筆資料依目標類別分類，則可計算得到 type 屬性的子集合及相對應的 *g-mean*，因此 type 屬性的最佳區間為 S(loan)。

但對任一關聯而言，任意改變其原始資料的分布並不合理，因為此一舉動會使得訓練資料集失真進而降低最後建構出之多重關聯分類器的準確度。為了解決這個問題，Mr.G-Tree 提出 G-meam 資料增殖演算法(G-mean Tuple ID Propagation algorithm, 簡稱 GTIP algorithm)，藉由將每筆資料的目標類別個數還原成一個的方式來維持每個關聯中的原始資料分布。對於非目標關聯 $\overline{R}$ 中的任一屬性，GTIP 在讀取資料值之前，會先查詢該筆資料值對應的 class labels 欄位，並根據增殖後目標類別的分布情形加以處理如下列定義一所示，藉以消除增殖後非目標關聯 $\overline{R}$ 中目標類別的原始分布被改變及 *g-mean* 值失真的問題。圖 3.1 為 GTIP 演算法的虛擬碼。

**定義一：**給定一非目標關聯 $\overline{R}$，並假設此關聯經過增殖後具有 IDs 以及 class labels 欄位，若某筆資料 $tuple_i$ 其 class labels 欄位中所有的目標關聯皆為正例或負例，則將資料 $tuple_i$ 的目標類別分類為正例或負例；反之則資料 $tuplei$ 包含 $\frac{CP}{CP + CN}$ 個正例以及 $\frac{CN}{CP + CN}$ 個負例。其中 $CP$ 代表該筆資料 class labels 欄位中正例的個數，$CN$ 代表該筆資料 class labels 欄位中負例的個數。

```
Procedure GTIP(n)
Begin
  for each relation R̄
  {
    for each primary key/foreign-key k of R̄
    {
      if R̄ can join to some relation R with R̄.k
        then
          propagate IDs and class labels from R to R̄
    }
  }
  for each tuple of relation R̄   /* 修正每筆資料之目標類別 */
  {
    if target classes of the tuple_i are all Positive or Negative
      then
        classify tuple_i as Positive or Negative
      else
        classify tuplei as CP/(CP + CN) Positive and
        CN/(CP + CN) Negative
  }
   return new target class to each tuple
end
```

圖 3.1 GTIP 演算法之虛擬碼

同樣再以表 2.1 中的 type 屬性為例，若依 GTIP 的增殖方式，則增殖結果如表 3.1 所示，而且將每一筆資料依目標類別分類，則可計算得到 type 屬

性的子集合及相對應的 *g-mean*。CrossMine 的增殖方式因改變了關聯 Order 中目標類別的原始分布，所以其 type 屬性的最佳區間為 S(loan)。但從 GTIP 的結果可以得知，type 屬性的合理最佳區間應是 S(insurance)。換而言之，GTIP 演算法避免了非目標關聯增殖時，目標類別增多造成 *g-mean* 值失真的問題。

表 3.1　GTIP 增殖的結果

| Order | | | | |
|---|---|---|---|---|
| order-id | account-id | type | **IDs** | **class labels** |
| 21 | 124 | insurance | 3, 4, 5, 6, 7, 8, 9 | 0+, 1− |
| 22 | 79 | insurance | 10 | 1+, 0− |
| 23 | 108 | insurance | 11 | 1+, 0− |
| 24 | 45 | insurance | 12 | 1+, 0− |
| 25 | 60 | loan | 13, 16 | 1/2+, 1/2− |
| 26 | 60 | loan | 13, 16 | 1/2+, 1/2− |
| 27 | 111 | loan | 14, 15 | 1/2+, 1/2− |
| 28 | 111 | loan | 14, 15 | 1/2+, 1/2− |
| 29 | 95 | loan | 17, 18, 19, 20 | 0+, 1− |

## 3.2 Mr.G-Tree 法樹葉節點之處理

傳統的決策樹演算法如 C4.5 等，探勘出的規則會被佔有較大比例的資料主導，其所產生的決策樹在整體上雖能達到十分高的預測正確性，但對於佔少數比例的資料卻無法完整且正確地推導出其規則。His-Tree 雖然利用 *g-mean* 作為挑選最佳分裂屬性的準則來解決不平衡資料集的問題，但 His-Tree 並無法直接套用在多重關聯資料庫中。此外，His-Tree 在建樹時以分類函數(classification function)作為判別樹葉節點目標類別的準則：其將所有分到左子節點的資料分類為正例，分到右子節點的資料分類為負例；當樹葉節點同時含有正、負例(即樹葉節點無法純化)並為右子節點時，在此樹葉節點中的正例將被分類為負例，進而降低正例的分類正確性。

因此，為了能更精確的探勘出使用者感興趣但卻非常少量的重要資料，在建樹的過程中，Mr.G-Tree 在樹葉節點目標類別之判別上加入了不平衡資料集的考量如下列定義所示，圖 3.2 則為相對應的虛擬碼。

**定義二：**給定一個只具有正、負二種目標類別的訓練資料集 *T*。對於 Mr.G-Tree 中任一樹葉節點 $N_i$，假設 *NP* 代表 $N_i$ 中正例的個數，*NN* 表示 $N_i$ 中負例的個數，*TP* 為該 *T* 中所有正例的總個數，*TN* 則為 *T* 中所有負例總個數。若 $NP \le \frac{NN \times TP}{TN}$，則樹葉節點 $N_i$ 之目標類別為負例；反之為正例。

```
Procedure Leaf_Node(n)
begin
    for each leaf node     /* 決定樹葉節點的分類結果 */
    {
      if   NP ≤ (NN × TP) / TN     then
        the target class of this node is
          Negative
    else
          the target class of this node is Positive
}
end
```

圖 3.2　Mr.G-Tree 判別樹葉節點目標類別之虛擬碼

以表 2.4 的客戶資料庫為例，該資料庫中共有 20 筆資料，其中包含 4 筆正例(buys_computer ="yes")和 16 筆負例(buys_computer = "no")。若假設此時資料庫只含有 student、credit_rating、buys_computer 三種屬性，則依 His-Tree 所其計算出之分類函數 *SF*(*i*)分別為

$SF(1) = 0.24 \times h_{credit\_rating}[\text{low}] + 0.13 \times h_{student}[\text{yes}]$；

$SF(2) = 0.14 \times h_{student}[\text{no}]$；

$SF(3) = 0.03 \times h_{credit\_rating}[\text{middle}] + 0.33 \times h_{student}[\text{yes}]$；

$SF(4) = 0.06 \times h_{credit\_rating}[\text{middle}]$。

其中 *i* 表示 His-Tree 中的第 *i* 個節點，圖 3.3 則為 His-Tree 建構出之決策樹。

如第二章所述，His-Tree 以分類函數(classification function)作為判別樹葉節點目標類別的準則，所以 His-Tree 會將圖 3.3 中的節點 4、8 之目標類別設為正例，節點 5、7、9 的目標類別則設為負例；節點 5 和節點 9 中共有 2 筆正例被分類

為負例，節點 4 和節點 8 中共有 4 筆負例被分類為正例；其正例的正確率為 2 / 4 = 50%，整體正確率為 $\frac{1}{2}(50\%+75\%)$ = 62.5%。若以傳統決策樹如 C4.5 所使用的大數法則作為判別樹葉節點目標類別的準則，則正例的正確率為 1 / 4 = 25%，整體正確率為 $\frac{1}{2}(25\%+81.25\%)$ = 53.125%。最後，若以定義一作為判別樹葉節點之目標類別的準則，其結果如圖 3.4 所示，則節點 9 中有 1 筆正例被分類為負例，節點 4、5、8 中共有 6 筆負例被分類為正例；因此 Mr.G-Tree 法正例的正確率為 3 / 4 = 75%，整體正確率為 $\frac{1}{2}(75\%+62.5\%)$ = 68.75%。而且由此例子中得知，Mr.G-Tree 利用正例與負例之總個數的比例來作為目標類別判別的標準，在正例正確率以及整體正確率上皆能夠達到較佳的分類準度。

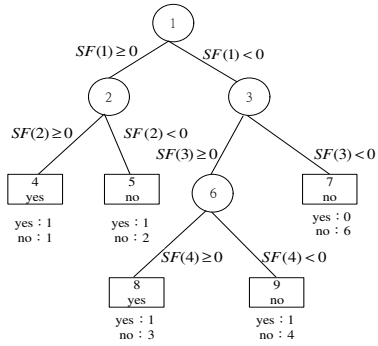



圖 3.3　依表 2.4 的 student、credit_rating、buys_computer 三種屬性建構出之 His-Tree

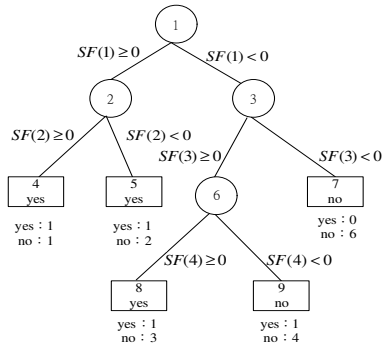



圖 3.4　依表 2.4 的 student、credit_rating、buys_computer 三種屬性建構出之 Mr.G-Tree

## 3.3 Multi-relational G-mean decision Tree (Mr.G-Tree)

結合前兩節之所述，本論文提出多重關聯 G-mean 決策樹演算法(Multi-relational G-mean decision Tree，簡稱為 Mr.G-Tree)，以解決在多重關聯資料庫下不平衡資料集的問題。為了清楚了解 Mr.G-Tree 法建樹的過程，在此將其步驟簡述如下：

一、首先，對所有有效的關聯執行 GTIP 演算法，修正每筆資料之目標類別，並計算每一個有效關聯中每一屬性最佳區間或最佳集合，並計算其 *g-mean* 值，數值和類別屬性的演算法分別如圖 3.5 及 3.6 所示。

二、假如該屬性的最大 *g-mean* 值小於 0.5，則放棄該屬性。

三、計算每一個剩下來的屬性之權重

$$w_i = \log_{10}\left(\frac{g_i}{1-g_i}\right)。$$

四、定義 $h_i$ 為第 $i$ 個屬性的輸出函數。$h_i = 1$，表示該屬性值落在該屬性的最佳區間內，反之則 $h_i = -1$。

五、結合所有屬性的輸出函數和權重成為資料的分類函數值：

$$SF = \sum_i h_i * w_i$$

若某筆資料 $p$ 的分類函數值 $SF$ 大於 0，則分到左子節點，反之則分到右子節點。

六、此時當其他的關聯可經由主鍵或外來鍵連接到有效的關聯時，則將 IDs 以及 class labels 欄位中的資料增殖到該關聯中，並將該關聯設定為有效的。

七、重複步驟一～六，直到所有節點的資料完全純化(purity)，或無法繼續分類為止。

八、利用定義二所描述之規則來判別所有樹葉節點之目標類別。

```
procedure NUMERIC(n)
begin
  for each numeric attribute i
    {
      call GTIP(n)
      sort all examples according the value of this attribute
      for each minority example j
        {
          calculate the interval
          [1/2(a_ib + mina_i),1/2(a_in + maxa_i)]
          calculate the g_i of the interval
        }
          select the best interval whose g_i is the maximal.
          if the value of g_i of this attribute i < 0.5  then
            discard this attribute
          else if g_i =1  then
                w_i =1
          else
              w_i = log(g_i /(1 − g_i))
           end if
              return the best interval and its corresponding
                weight w_i
      }
end
```

圖 3.5　Mr.G-Tree 的數值屬性之演算法

```
procedure CATEGORY(n)
begin
   for the categorical attribute i, building the corresponding
        power set.
      {
        call GTIP(n)
        for each subset
          {
            classify all examples to the corresponding set by
              their class
            calculate the g_i of this subset
          }
            select the best subset whose g_i is the maximal
            if the value of g_i of this attribute i < 0.5  then
                discard this attribute
              else if g_i =1  then
                      w_i =1
              else
                      w_i = log(g_i /(1 − g_i))
            end if
              return the best subset and its corresponding
                weight w_i
      }
end
```

圖 3.6　Mr.G-Tree 的類別屬性之演算法

# 4. 實驗分析與效能評估

為了分析 Mr.G-Tree 的準確性和效能，本論文採用 Microsoft Visual C++ 6.0 實做 TILDE、CrossMine、His-Tree、Mr.G-Tree 並進行各種實驗模擬。如第三章所述，傳統用來評估分類器優劣的整體正確率會被較多的負例個數所主導，因此並不適用於不平衡資料集[3]；所以本論文除了使用正例正確率

$$accuracy^{+} = \frac{d}{c+d}$$

來評估分類器對於少量資料的預測能力外，亦採用權重正確率(weighted accuracy) [3]

$$w\text{-}accuracy = \frac{1}{2}\left(\frac{a}{a+b}+\frac{d}{c+d}\right)$$

作為評估整體正確率的準則。

實驗模擬時的軟硬體環境以及實驗之資料庫將詳述於第一節中。在第二節中，為了解 TILDE、CrossMine、 Mr.G-Tree 處理不平衡關聯式資料庫的能力，本論文利用實際關聯式資料庫進行準確性分析。

## 4.1 實驗模擬環境

本論文採用的實驗模擬環境為：

一、實驗使用的軟硬體：

CPU：Intel Pentium IV 3.0G

RAM：512MB DDR RAM

OS：Windows XP Professional

二、實驗資料庫：

實際關聯式資料庫：

本論文沿用 CrossMine 中的二個實際關聯式資料庫 Financial database 及 Mutagenesis database [27]。其中 financial database 包含 1 個目標關聯、7 個非目標關聯以及 75982 筆資料；Mutagenesis database 則包含 1 個目標關聯、3 個非目標關聯以及 15218 筆資料。

### 4.2 TILDE、CrossMine、Mr.G-Tree 之實驗分析

為了探討當關聯式資料庫中的資料集呈現不平衡的情況時，TILDE、CrossMine、Mr.G-Tree 在正例正確率以及整體正確率上之優劣，本論文利用實際關聯式資料庫進行實驗模擬。

圖 4.1 為 TILDE、CrossMine、Mr.G-Tree 在二個實際關聯式資料庫的正例正確率之比較；圖 4.2 則為整體正確率之比較。由圖 4.1 可以得知，Mr.G-Tree 在正例正確率上皆比 TILDE 以及 CrossMine 來的高；由圖 4.2 可以得知， TILDE、CrossMine、Mr.G-Tree 在整體正確率上並無顯著差異。

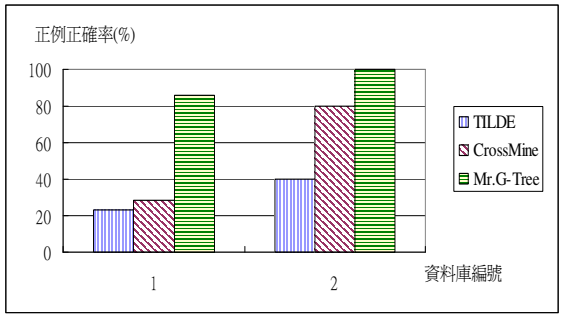



圖 4.1　實際關聯式資料庫正例正確率之比較

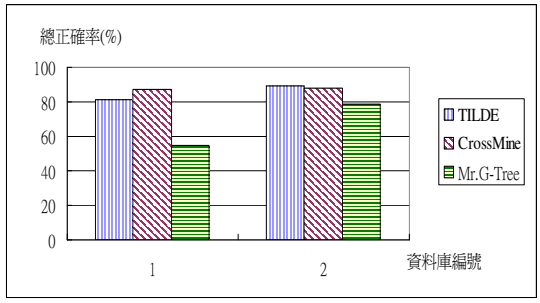



圖 4.2　實際關聯式資料庫整體正確率之比較

## 5. 結論與未來研究方向

由於現今大多數的結構化資料都是被儲存在關聯式資料庫中，傳統資料探勘的技術已不再適用。雖然已經有許多多重關聯資料探勘分類器提出，例如：TILDE、FOIL、CrossMine…等，但是在資料集不平衡的情況下，這些多重關聯資料探勘分類器並不能有效的分類出佔少數比例的資料(正例)。因此，本論文提出 Mr.G-Tree 演算法來解決這個問題。Mr.G-Tree 除了利用 GTIP 演算法避免了非目標關聯增殖時，目標類別增多造成 *g-mean* 值失真的問題，亦在樹葉節點目標類別之判別上加入了不平衡資料集的考量以提升其對正例正確率的預測準度。

在實驗分析方面，因為傳統用分類器優劣的整體正確率並不適用於不平衡資料集，所以本論文除了使用正例正確率來評估分類器對於少量資料的預測能力外，亦採用權重正確率作為評估整體正確率的準則。實驗結果顯示，對於多重關聯不平衡資料集，Mr.G-Tree 在正例正確率上能較 TILDE、CrossMine 達到更好的分類正確率，在整體的正確率上亦能得到不錯的結果。

然而，Mr.G-Tree 目前只適用於兩個目標類別(two target class)的資料集，當遇到多目標類別(multiple target class)時，雖然可將資料庫中所佔比例最小的目標類別值設為正例其餘皆設為負例，但此一方式會改變原始資料的分布情形。因此，本論文未來將擴展 Mr.G-Tree 使其能處理多目標類別(multiple target class)的資料庫。此外，當多重關聯資料庫中的資料隨著時間改變時(如新增、刪除、更新)，如何能快速且有效地修改 Mr.G-Tree 而不需重建亦是一個值得研究的議題。

## 參考文獻

[1] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” *Data Mining and Knowledge Discovery*, Vancouver, Vol. 2, pp.121-168, 1998.

[2] N. V. Chawla, A. Lazarevic, L. O. Hall and K. W. Bowyer, “SMOTEboost: Improving prediction of the minority class in boosting,” In *7th European Conference on Principles and Practice of Knowledge Discovery in Databases Cavtat-Dubrovnik,* Croatia, pp. 107–119, 2003.

[3] C. Chen, A. Liaw and L. Breiman, “ Using random forest to learn imbalanced data,” Technical Report, No. 666, Department of

Statistics, University of California at Berkeley, 2004.

[4] Domingos and Pedro, "Metacost: A general method for making classifiers cost sensitive," In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, San Diego, pp. 155-164, 1999.

[5] B. Zadrozny and C. Elkan, "Learning and Making Decisions When Costs and Probabilities are Both Unknown," In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, San Francisco, pp. 204-213, 2001.

[6] C. Ling and C. Li, "Data mining for direct marketing problems and solutions," In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, New York, 1998.

[7] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume and C. Brunk, "Reducing misclassification costs," In *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick, New Jersey, pp. 217-225, 1994.

[8] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[9] E. DeRouin, J. Brown, H. Beck, L. Fausett and M. Schneider, "Neural Network Training on Unequally Represented Classes," In *Intelligent Engineering Systems Through Artificial Neural Networks, ASME Press*, New York, pp. 135-145, 1991.

[10] S. Floyd and M. Warmuth, "Sample Compression, Learnability, and the Vapnik-Chervonenkis Dimension," *Machine Learning*, Vol. 21, pp. 269-304, 1995.

[11] H. Boström, "Covering vs. Divide-and-Conquer for Top-Down Induction of Logic Programs," In *Proc. of the Fourteenth International Joint Conference on Artificial Intelligence* , Morgan Kaufmann, pp. 1194-1200, 1995.

[12] M. Kubat, R. Holte and S. Matwin, "Learning when Negative Examples Abound," In *Proceedings of the 9th European Conference on Machine Learning*, Prague, Czech Republic, pp. 146-153, 1997.

[13] M. Kubat, R. Holte and S. Matwin, "Machine Learning for The Detection of Oil Spills in Satellite Radar Images," *Machine Learning*, Vol. 30, pp. 195-215, 1998.

[14] M. Kubat and S. Matwin, "Addressing the Curse of Imbalanced Training Sets: One-Sided Selection," In *Proceedings of the 14th International Conference on Machine Learning*, Nashville, US, pp. 179-186, 1997.

[15] D. D. Lewis and J. Catlett, "Heterogeneous Uncertainty Sampling for Supervised Learning," In *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, pp. 148-156, 1997.

[16] P. Riddle, R. Segal and O. Etzioni, "Representation Design and Brute Force Induction in a Boeing Manufacturing Domain," *Applied Artificial Intelligence*, Vol. 8, pp. 125-147, 1994.

[17] A. H. S. Solberg and R. Solberg, " A Large-Scale Evaluation of Features for Automatic Detection of Oil Spills in ERS SAR Images," *International Geoscience and Remote Sensing Symposium*, Lincoln, NE, pp. 1484-1486, 1996.

[18] H. Blockeel, L. De Raedt and J. Ramon, "Top-down induction of logical decision trees," Artificial Intelligence, Madison, WI, Vol.101, pp. 285-297, 1998.

[19] C. Lee, C. Tsai and C. Chen, "A Multivariate Decision Tree for Imbalanced Datasets," Master Thesis, Taiwan, ROC, 2002.

[20] N. Japkowicz and S. Stephen, "The Class Imbalance Problem: A Systematic Study,"

Intelligent Data Analysis Journal, Vol. 6, No. 5, pp429-450, 2002.

[21] N. Lavrac and S. Dzeroski, "Inductive Logic Programming: Techniques and Applications," Ellis Horwood, New York, 1994.

[22] S. Muggleton, "Inductive Logic Programming," Academic Press, New York, NY, 1992.

[23] S. Muggleton, "Inverse entailment and progol," In *New Generation Computing, Special issue on Inductive Logic Programming*, Vol. 13, pp.245-286, 1995.

[24] S. Muggleton and C. Feng, "Efficient induction of logic programs," In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, Tokyo, Japan, pp. 368-381, 1990.

[25] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann, 1993.

[26] J. R. Quinlan and R. M. Cameron-Jones, "FOIL: A midterm report," In *Proc. 1993 European Conf. Machine Learning*, Vienna, Austria, pp. 3-20, 1993.

[27] X. Yin, J. Han, J. Yang and P. S. Yu, "Crossmine: Efficient classification across multiple database relations," In *Proc. of the 20th International Conference on Data Engineering*, Boston, MA, pp. 399-411, 2004.

[28] R. S. Michalski, "On the Quasi-minimal Solution of the General Covering Problem," In *Proceedings of the 5th International Symposium on Information Processing* (FCIP69), , Bled, Yugoslavia, pp.125-128, 1969.

[29] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, Vol. 3, No. 4, pp.261-283, 1989.