有效率地線上探勘關聯規則

陳垂呈
南台科技大學資訊管理系
E-mail: ccchen@mail.stut.edu.tw

陳俊堯
南台科技大學資訊管理研究所
E-mail: m9190223@webmail.stut.edu.tw

摘要

資料探勘(data mining)是從大量交易資料中找出潛在有用的知識與資訊，在資料探勘技術可完成的工作中，以關聯規則(association rules)來表示項目之間的關聯性，是資料探勘最常使用的方法之一。在探勘關聯規則的過程中，若有新增交易資料時，則必須重新計算一次，將造成前次探勘計算的重複及資源的浪費，因此，在考量線上新增交易資料時，如何避免探勘計算的重複，以較少的計算時間來更新關聯規則，即成為探勘關聯規則最重要的問題之一。在本篇論文中，我們提出兩個演算法來線上探勘關聯規則：一是先以 Apriori 演算法來找出所有的關聯規則，在探勘的過程中，我們保留計算過程中的項目組與其出現次數、及高頻項目組與其出現次數於記憶體中，若有新增交易資料時，我們考量新增交易資料對記憶體中項目組及高頻項目組之出現次數的影響，若有產生新項目組時，則必須掃瞄全部的交易資料，以判斷這些項目組是否為高頻項目組；二是我們以布林運算為基礎，提出一個演算法，將可有效地提升前一方法的執行效能。從效能實驗顯示，我們所提出的探勘方法，將可有效地提升線上探勘關聯規則的執行效率。

關鍵詞：資料探勘、關聯規則、高頻項目組、布林運算

一、簡介

近幾年來，資料探勘(data mining)技術已廣泛地運用在資料庫領域中，其目的是從大量交易資料中找出潛在有用的資訊與知識，以支援企業的行銷決策。經由資料探勘可擷取出項目(items)之間的關聯性，並以關聯規則(association rules)的形式表示。在探勘關聯規則的過程中，若有新增交易資料時，往往必須重新探勘，如此將導致前次探勘計算的重複及資源的浪費，因此，在考量線上新增交易資料時，如何避免探勘計算的重複，以較少的計算時間來更新關聯規則，即成為探勘關聯規則最重要的問題之一。

在探勘關聯規則的方法中，由 Agrawal 等人所提出的 Apriori 演算法[5]是最具代表性的方法之一，而[8]提出一個布林演算法，並且證明其演算法的執行效率優於 Apriori 演算法。在本篇論文中，我們將根據 Apriori 演算法的執行步驟及以布林運算為基礎，考量線上新增交易資料時，如何有效率地探勘關聯規則。

在本篇論文中，我們提出兩個方法來線上探勘關聯規則：一是先以 Apriori 演算法來找出所有的關聯規則，在探勘的過程中，我們保留計算過程中的項目組(itemsets)與其出現次數、及高頻項目組(frequent itemsets)與其出現次數於記憶體中，若有新增交易資料時，我們考量新增交易資料對記憶體中項目組及高頻項目組之出現次數的影響，若因此而產生新項目組時，則必須掃瞄全部的交易資料，以判斷這些新項目組是否為高頻項目組，藉由更新後的高頻項目組，來計算出所有更新後的關聯規則；二是我們以布林運算為基礎，提出一個演算法，將可有效地提升前一方法的執行效能。

在探勘的過程中，我們先考量新增交易資料對記憶體中項目組及高頻項目組之出現次數的影響，惟有產生新項目組時，才必須掃瞄全部的交易資料，以判斷這些新項目組是否為高頻項目組。因為新增交易資料的數量往往遠小於原先資料庫中交易資料的數量，因此新增交易資料並不會對記憶體中的項目組及高頻項目組，造成太大的變動，對掃瞄全部交易資料的次數將可大幅減少。

本篇論文的架構如下：下一節中，我們介紹探勘關聯規則的相關研究；第三節中，我們考量若有新增交易資料時，如何線上探勘關聯規則；第四節中，我們以布林運算為基礎，考量若有新增交易資料時，如何線上探勘關聯規則；第五節中，我們實驗評估前面章節所描述之演算法的執行效能；最後，在第六節中我們做一結論。

二、相關研究

從大量的交易資料中探勘項目之間的關聯規則，由 Agrawal 等人首先提出[4]，之後相關的研究也相繼的被發表出來[3, 5-8]，其中又以 Apriori 演算法[5]最具代表性，而 Wur & Leu 提出一個布林演算法可以有效率地擷取出關聯規則[8]。接下來，我們說明關聯規則的相關定義。

假設 I 是資料庫中所有項目的集合，每一筆交易資料 T 是由一些項目所形成的集合，並且 $T \subseteq I$，在項目集合 X 與 Y 之間有一關聯規則

被表示成 $X \rightarrow Y$，其中 X 稱之為前項目組(antecedent)，Y 稱之為後項目組(consequent)，X、$Y \subseteq I$ 且 $X \cap Y = \varnothing$。有兩個參數 s 與 c 分別為支持度(support)與信賴度(confidence)，用來決定關聯規則 $X \rightarrow Y$ 是否為有效規則(strong rules)；支持度 s 表示為：在所有的交易資料集合中，同時包含有$(X \cup Y)$的比率值，即 s=(同時包含有$(X \cup Y)$的交易資料數量)/(總交易資料數量)；而信賴度 c 表示為：在包含有 X 的交易資料集合中，也同時包括有 Y 的比率值，即 c=(同時包含有$(X \cup Y)$的交易資料數量)/(包含有 X 的交易資料數量)。擷取出來的關聯規則，其支持度與信賴度必須大於或等於所指定的最小支持度與最小信賴度，這樣的關聯規則才有意義。

關聯規則的擷取過程主要分成兩個階段：在第一階段中，先找出滿足最小支持度的項目組，這些滿足最小支持度的項目組，稱之為高頻項目組(frequent itemsets)，若一個項目組包含有 k 個項目，稱之為 k-項目組(k-itemsets)，以 $itemset_k$ 表示之，若某 k-項目組滿足最小支持度，稱之為高頻 k-項目組(frequent k-itemsets)，以 $frequent_k$ 表示之。在第二階段中，以最小信賴度為條件，計算高頻項目組所形成的關聯規則，若滿足最小信賴度，則關聯規則成立，例如 ABC 為高頻 3-項目組，A、B、C$\in I$，若關聯規則 AB$\rightarrow$C 滿足最小信賴度，則此關聯規則成立。

在探勘關聯規則的過程中，若有新增交易資料，依據上述演算法往往必須重新探勘，如此將造成前次探勘計算的重複及資源的浪費，因此，如何有效地處理線上新增交易資料以更新關聯規則，即成為探勘關聯規則重要的研究問題之一。線上探勘關聯規則已有許多的相關研究被發出來[1, 2, 9, 10]，其中[9]提出一FUF(Fast Update)演算法，來對動態資料庫做即時的關聯規則維護，在[10]中，則對於[9]的方法提出修改，以減少在執行過程中產生過多的候選項目組。

在本篇論文中，我們將根據 Apriori 演算法[5]的執行步驟及以布林運算為基礎，考量線上新增交易資料時，如何有效率地探勘關聯規則。

三、線上探勘關聯規則

在此章節中，我們先以 Apriori 演算法來找出關聯規則，但是儲存探勘過程中的項目組與其出現次數、及高頻項目組與其出現次數於記憶體中。然後考量若有新增交易資料時，在避免重複前次探勘計算的情況下來線上探勘關聯規則。此章節共分為兩小節如下：第一小節中，我們說明以 Apriori 演算法來探勘關聯規則，並考量新增交易資料時的線上探勘關聯規則；第二小節中，我們以一實例做說明。

(一) 探勘關聯規則

在傳統探勘關聯規則的方法中，若資料庫有新增交易資料時，就必須重新計算探勘關聯規則的過程，如此大量耗費計算的方式，並不太能符合線上探勘的時效性。我們利用表 1 的表格來儲存探勘關聯規則過程中產生的項目組、出現次數、及是否為高頻項目組。而後，當資料庫有新增交易資料進來時，將以即時的處理方式將其拆解成 1-項目組，並且將次數累加進入其相對的欄位中，高頻項目組也會隨著出現次數的變動而改變，當計算儲存於表格中之項目組的出現次數時，只須掃瞄此新增交易資料即可。

表 1：項目組儲存表格

項目組	出現次數	是否為高頻項目組

以下我們說明 Apriori 演算法[5]探勘關聯規則的執行步驟：

(1) 找出 $frequent_{k-1}$，k>1，並儲存於記憶體的表格中，若為$\varnothing$，則停止執行。
(2) 由(1)中找出任兩個有 k-2 項目相同的 $frequent_{k-1}$，組合成 $itemset_k$。
(3) 判斷由(2)所找出的 $itemset_k$，其所有包括的 $itemset_{k-1}$ 之子集合是否都出現在(1)中，假如成立就保留此 $itemset_k$，否則就刪除。
(4) 儲存由(3)所擷取的 $itemset_k$，於記憶體的表格中，並檢查否滿足最小支持度，假如符合就成為 $frequent_k$，否則就刪除。
(5) 計算 $frequent_k$所有可能形成的關聯規則，若滿足最小信賴度，則關聯規則成立。
(6) 跳至(1)找 $frequent_{k+1}$，直到無法產生高頻項目組為止。

經由上述演算法，可以找出所有的關聯規則，並且儲存探勘過程中之項目組與其出現次數。現在考量當有新增一筆交易資料T_{new}進來時，其各項目組的支持度必須更新為：項目組更新之後的出現次數/(原先資料庫中交易資料的數目+1)，探勘關聯規則的過程說明如下：

(1) 讀入T_{new}。
(2) 計算之前儲存的$itemset_1$，若$itemset_1$有出現在T_{new}中，則其對應的項目組次數加1，若滿足最小支持度，則成為 $frequent_1$。

(3) 組合任兩個$frequent_1$，形成$itemset_2$，若已儲存於表格中，則掃瞄T_{new}，若$itemset_2$為新形成的項目組，則儲存於記憶體的表格中，並掃瞄原先資料庫與T_{new}，計算$itemset_2$的出現次數，若滿足最小支持度，則成為$frequent_2$。

(4) 找出$frequent_{k-1}$，k>2。

(5) 由(4)中，組合任兩個有k-2項目相同的$frequent_{k-1}$，形成$itemset_k$。

(6) 判斷由(5)所找出的$itemset_k$，其所包含的子集合$itemset_{k-1}$是否都有出現在(4)中，假如成立就保留此$itemset_k$，否則就刪除。

(7) 檢查由(6)所找出的$itesmet_k$，若已儲存於表格中，則掃瞄T_{new}，若為新形成的項目組，則儲存於記憶體的表格中，並掃瞄原先資料庫與T_{new}，計算$itemset_k$的出現次數，若滿足最小支持度，則成為$frequent_k$。

(8) 計算$frequent_k$可能形成的關聯規則，若滿足最小信賴度，則關聯規則成立。

(9) 跳至(4)繼續找出$frequent_{k+1}$，直到無法產生高頻項目組為止。

在以上演算法的探勘過程中，對新增交易資料而言，由於此新增交易資料而變動的項目組，若之前已儲存於表格中，則只須掃瞄此新增交易資料，即可更新各項目組的出現次數。若為新產生的項目組，則必須掃瞄原先資料庫與此新增交易資料，才能計算出新項目組是否為高頻項目組。對線上新增交易資料而言，其數量往往遠小於原先資料庫中交易資料的數量，因此在更新關聯規則的過程中，應不至於變動太多已儲存於表格中的項目組。雖然對新產生的項目組必須掃瞄原先資料庫與新增交易資料，但相較於必須重新探勘關聯規則而言，仍然減少很多重複性的計算過程。

若有 s 筆新增交易資料，s≥1，更新關聯規則之過程如上述演算法的執行步驟，其計算各項目組的支持度必須更新為：項目組更新之後的出現次數/(原先資料庫中交易資料的數目+s)。

(二) 實例說明

表 2 為資料庫 D 中有 4 筆的交易資料，{A、B、C、D、E}為所有項目所形成的集合，{T_1, T_2, T_3, T_4}為 4 筆交易資料所形成的集合。設定最小支持度為 40％(即最小支持數量為 1.6)，最小信賴度為 60%。

表 2：資料庫 *D*

交易資料編號	項目
T_1	ACD
T_2	BCE
T_3	ABCE
T_4	BE

首先以 Apriori 演算法擷取高頻項目組的過程如表 3。得到的高頻項目組有 BCE、AC、BC、BE 及 CE，分別計算其可能形成的關聯規則，若滿足最小信賴度，則關聯規則成立。

表 3

$itemset_1$
A
B
C
D
E

掃瞄 D →

$itemset_1$	出現次數	是否高頻
A	2	
B	3	
C	3	
D	1	
E	3	

≥1.6 →

$itemset_1$	出現次數	是否高頻
A	2	*
B	3	*
C	3	*
D	1	
E	3	*

$itemset_2$
AB
AC
AE
BC
BE
CE

掃瞄 D →

$itemset_2$	出現次數	是否高頻
AB	1	
AC	2	
AE	1	
BC	2	
BE	3	
CE	2	

≥1.6 →

$itemset_2$	出現次數	是否高頻
AB	1	
AC	2	*
AE	1	
BC	2	*
BE	3	*
CE	2	*

表 3 (續)

$itemset_3$
ABC
ACE
BCE

(3)步驟及掃瞄 D →

$itemset_3$	出現次數	是否高頻
BCE	2	

≥1.6 →

$itemset_3$	出現次數	是否高頻
BCE	2	*

如果目前有一筆新增交易資料T_5進來，其交易資料為ACE，最小支持數量變更為40%×5=2，則探勘高頻項目組的過程如表4。在計算因新增交易資料T_5而更新關聯規則的過程中，只有ACE是新產生的項目組，故必須掃瞄原先資料庫D與T_5，以判斷ACE是否為高頻項目組。經更新計算後，得到的高頻項目組有ACE、BCE、AC、AE、BC、BE及CE，分別計算其可能形成的關聯規則，若滿足最小信賴度，則關聯規則成立。

四、以布林運算為基礎線上探勘關聯規則

[8]已經描述使用布林運算的方式，可使提升原先 Apriori 演算法的執行效率，而[3]根據 Apriori 演算法的執行步驟，提出一個以布林運算為基礎的方法，將可提升[8]所描述之演算法的執行效能。在此章節中，我們首先根據[3]

表4

$Itemset_1$
A
B
C
D
E

掃瞄 T_5 →

$itemset_1$	出現次數	是否高頻
A	3	
B	3	
C	4	
D	1	
E	4	

≥2 →

$itemset_1$	出現次數	是否高頻
A	3	*
B	3	*
C	4	*
D	1	
E	4	*

$Itemset_2$
AB
AC
AE
BC
BE
CE

掃瞄 T_5 →

$itemset_2$	出現次數	是否高頻
AB	1	
AC	3	
AE	2	
BC	2	
BE	3	
CE	3	

≥2 →

$itemset_2$	出現次數	是否高頻
AB	1	
AC	3	*
AE	2	*
BC	2	*
BE	3	*
CE	3	*

$itemset_3$
ABC
ABE
ACE
BCE

(3)步驟 →

$itemset_3$
ACE
BCE

掃瞄 D 與 T_5 →

掃瞄 T_5 →

$itemset_3$	出現次數	是否高頻
ACE	2	
BCE	2	

≥2 →

$itemset_3$	出現次數	是否高頻
ACE	2	*
BCE	2	*

所提出之方法，來探勘關聯規則，但是儲存探勘過程中的項目組與其出現次數、及高頻項目組與其出現次數於記憶體中。然後考量若有新增交易資料時，在避免重複前次探勘計算的情況下來線上探勘關聯規則。此章節共分為兩小節如下：第一小節中，我們說明以[3]所提出之演算法來探勘關聯規則，並考量新增交易資料時的線上探勘關聯規則；第二小節中，我們以一實例做說明。

(一) 探勘關聯規則

首先，我們說明一些名詞定義如下：

- $I=\{i_1, i_2, \ldots, i_n\}$，是全部項目(items)的集合，共有 n 項。
- $T=\{T_1, T_2, \ldots, T_j, \ldots, T_m\}$，是全部交易資料的集合，共有 m 筆，其中 T_j 為第 j 筆交易資料，$1\le j\le m$；。
- T_j 由 n 位元(bits)所組成，其格式表示成 $T_j=[b_1, b_2, b_3, \ldots, b_f, \ldots, b_n]$，$b_f\in\{0, 1\}$，$1\le f\le n$，若交易資料 T_j 中有出現第 f 項的項目，則 $b_f=1$，否則 $b_f=0$。
- $itemset_k$ 表示包含有 k 個項目的項目組，其資料格式如同 T_j。
- $frequent_k$ 表示包含有 k 個項目的高頻項目組，其資料格式如同 T_j。

在探勘關聯規則的過程中，我們儲存探勘過程中產生的項目組、出現次數、及是否為高頻項目組。而後，當資料庫有新增交易資料進來時，將以即時的處理方式，當計算儲存於記憶體中之項目組的出現次數時，只須掃瞄此新增交易資料即可。我們分別使用 *or*(如圖 1)及 *xor* 布林運算(如圖 2)，可以很有效率地分別計算出兩項目組之間聯集及相異的位元。

or	0	1
0	0	1
1	1	1

圖 1

xor	0	1
0	0	1
1	1	0

圖 2

我們將每一交易資料轉換成位元的資料型態，若項目有出現在交易資料中，則相對位元設定為“1”，否則設定為“0”，每筆交易資料都是以 n 位元的格式表示之。以下為[3]所提出以布林運算為基礎來探勘關聯規則之執行步驟：

(1) 找出 $frequent_{k-1}$，並儲存於記憶體中，$k>1$，若為∅，則停止執行。
(2) 任意兩個 $frequent_{k-1}$ 做 *or* 布林運算，假如結果為 $itemset_t$，即有 k 個項目其值為“1”，且非重複者，就保留此 $itemset_k$，否則就刪除[8]。
(3) 判斷由(2)所找出的 $itemset_k$，其包含的 $itemset_{k-1}$ 之子集合，是否都出現在(1)中，可將 $itemset_k$ 與(1)中所有各 $frequent_{k-1}$ 做 *xor* 布林運算，計數結果為 $itemset_1$ 的數目是否等於 k，假如成立就保留此 $itemset_k$，否則就刪除。
(4) 儲存由(3)所擷取出的 $itemset_k$，並檢查是否滿足最小支持度，可將 $itemset_k$ 與交易資料 T_j 執行以下的運算：

$$itemset_k \text{ or } T_j \text{ xor } T_j,\ (1\le j\le m)$$

若結果為 $itemset_0$，則表示 $itemset_k\subseteq T_j$，掃瞄所有交易資料之後，判斷出現的次數是否滿足最小支持度，假如符合就成為 $frequent_k$，否則就刪除。
(5) 對 $frequent_k$ 計算可能形成的關聯規則，若前項目組設定為 X，則後項目組 Y 可由以下布林運算找出[8]：

$$Y=frequent_k \text{ xor } X$$

若關聯規則 $X\rightarrow Y$ 滿足最小信賴度，則關聯規則成立。
(6) 跳至(1)找 $frequent_{k+1}$，直到無法產生高頻項目組為止。

現在考量當有一筆新增交易資料為 T_{new} 進來時，依據前一章節的演算法及支持度的更新計算，其探勘關聯規則的過程說明如下：

(1) 讀入 T_{new}，並將之轉換成位元的資料格式。
(2) 計算之前儲存的 $itemset_1$，判斷 $itemset_1$ 是否有出現在 T_{new} 中，可執行以下的運算：

$$itemset_1 \text{ or } T_{new} \text{ xor } T_{new} \quad \ldots\ldots\ldots\text{(a)}$$

若為 $itemset_0$，則表示 $itemset_1\subseteq T_{new}$，其對應項目組的次數加1，若滿足最小支持度，則成為 $frequent_1$。
(3) 將任兩個 $frequent_1$ 做 *or* 布林運算，形成 $itemset_2$，若已儲存於記憶體中，則執行公式 (a) 的運算，若為 $itemset_0$，則表示 $itemset_2\subseteq T_{new}$，其對應項目組的次數加1，若 $itemset_2$ 為新產生的項目組，則掃瞄原先資料庫與 T_{new}，並儲存之，計算 $itemset_2$ 的出現次數，若滿足最小支持度，則成為 $frequent_2$。
(4) 找出 $frequent_{k-1}$，$k>2$。
(5) 由(4)中，將任兩個 $frequent_{k-1}$ 做 *or* 布林運算，若形成 $itemset_k$，且非重複，就加以保留。
(6) 判斷由(5)所找出的 $itemset_t$，其所包含的子集合 $itemset_{k-1}$ 是否都有出現在(4)中，其判斷方式可執行以下的運算：將 $itemset_k$ 與(4)中所有各 $frequent_{k-1}$ 做 *xor* 布林運算，計數結果為 $itemset_1$ 的數目是否等於 k，假如成

立就保留此$itemset_k$，否則就刪除。

(7) 檢查由(6)所找出的$itesmet_k$，若已儲存於記憶體中，則只須掃瞄T_{new}，執行公式(a)的運算，若為$itemset_0$，則表示$itemset_k \subseteq T_{new}$，其對應項目組的次數加1，若$itemset_k$為新產生的項目組，則掃瞄原先資料庫與$T_{new}$，並儲存之，計算$itemset_k$的出現次數，若滿足最小支持度，則成為$frequent_k$。

(8) 對$frequent_k$計算可能形成的關聯規則，若前項目組設定為X，則後項目組Y可由以下布林運算找出[8]：

$$Y = frequent_k \ xor \ X$$

若關聯規則 $X \rightarrow Y$ 滿足最小信賴度，則關聯規則成立。

(9) 跳至(4)繼續找出$frequent_{k+1}$，直到無法產生高頻項目組為止。

(二) 實例說明

我們仍以表 2 之資料庫 D 為例，各交易資料轉換成位元格式為：T_1=[10110]、T_2=[01101]、T_3=[11101]、T_4=[01001]。最小支持度為 40%(即最小支持數量=1.6)，最小信賴度=60%。

首先以[3]所描述之演算法擷取高頻項目組的過程如表 5。得到的高頻項目組有 BCE、AC、BC、BE 及 CE，分別計算其可能形成的關聯規則，若滿足最小信賴度，則關聯規則成立。

表 5

	T_1	T_2	T_3	T_4
A	1	0	1	0
B	0	1	1	1
C	1	1	1	0
D	1	0	0	0
E	0	1	1	1

(4)步驟 →

$itemset_1$

	A	B	C	D	E
	1	0	0	0	0
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	1	0
	0	0	0	0	1
出現次數	2	3	3	1	3

≥1.6 →

$frequent_1$

	A	B	C	D	E
	1	0	0	0	0
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	1	0
	0	0	0	0	1
出現次數	2	3	3	1	3
是否高頻	*	*	*		*

(2)、(4)步驟 →

$itemset_2$

	AB	AC	AE	BC	BE	CE
	1	1	1	0	0	0
	1	0	0	1	1	0
	0	1	0	1	0	1
	0	0	0	0	0	0
	0	0	1	0	1	1
出現次數	1	2	1	2	3	2

≥1.6 →

$frequent_2$

	AB	AC	AE	BC	BE	CE
	1	1	1	0	0	0
	1	0	0	1	1	0
	0	1	0	1	0	1
	0	0	0	0	0	0
	0	0	1	0	1	1
出現次數	1	2	1	2	3	2
是否高頻		*		*	*	*

(2)步驟 →

$itemset_3$

ABC	ACE	BCE
1	1	0
1	0	1
1	1	1
0	0	0
0	1	1

(3)步驟 →

$itemset_3$

ABC	ACE	BCE
1	1	0
1	0	1
1	1	1
0	0	0
0	1	1
出現次數	2	2	3

=3 →

$itemset_3$

BCE
0
1
1
0
1
3

4)步驟 →

$itemset_3$

BCE
0
1
1
0
1
2

≥1.6 →

$frequent_3$

BCE
0
1
1
0
1
2
是否高頻 *

如果目前有一筆新增交易資料 T_5 進來，其交易資料為 ACE，轉成位元格式為[10101]，最小支持數量變更為 40%×5=2，則擷取高頻項目組的過程如表 6。在計算因新增交易資料 T_5 而更新關聯規則的過程中，只有 ACE 是新產生的項目組，故必須掃瞄原先資料庫 D 與 T_5，以判斷 ACE 是否為高頻項目組。經更新計算後，得到的高頻項目組有 ACE、BCE、AC、AE、BC、BE 及 CE，分別計算其可能形成的關聯規則，若滿足最小信賴度，則關聯規則成立。

表 6

	T_1	T_2	T_3	T_4	T_5
A	1	0	1	0	1
B	0	1	1	1	0
C	1	1	1	0	1
D	1	0	0	0	0
E	0	1	1	1	1

掃瞄 T_5 →

$itemset_1$

	A	B	C	D	E
	1	0	0	0	0
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	1	0
	0	0	0	0	1
出現次數	3	3	4	1	4

≥2 →

$frequent_1$

	A	B	C	D	E
	1	0	0	0	0
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	1	0
	0	0	0	0	1
出現次數	3	3	4	1	4
是否高頻	*	*	*		*

(3)步驟 →

$itemset_2$

	AB	AC	AE	BC	BE	CE
	1	1	1	0	0	0
	1	0	0	1	1	0
	0	1	0	1	0	1
	0	0	0	0	0	0
	0	0	1	0	1	1
出現次數	1	3	2	2	3	3

≥2 →

$frequent_2$

	AB	AC	AE	BC	BE	CE
	1	1	1	0	0	0
	1	0	0	1	1	0
	0	1	0	1	0	1
	0	0	0	0	0	0
	0	0	1	0	1	1
出現次數	1	3	2	2	3	3
是否高頻		*	*	*	*	*

(5)步驟 →

$itemset_3$

	ABC	ACE	BCE
	1	1	0
	1	0	1
	1	1	1
	0	0	0
	0	1	1

(6)步驟 →

	ABC	ACE	BCE
	1	1	0
	1	0	1
	1	1	1
	0	0	0
	0	1	1
出現次數	2	3	3

=3 →

	ACE	BCE
	0	0
	1	1
	1	1
	0	0
	1	1
	3	3

對 BCE 掃瞄 T_5，對 ACE 掃瞄原先資料庫與 T_5 ≥2 →

	ACE	BCE
	0	0
	1	1
	1	1
	0	0
	1	1
出現次數	2	2
是否高頻	*	*

在計算因新增交易資料 T_5 而更新關聯規則的過程中，只有 ACE 是新產生的項目組，故必須掃瞄原先資料庫 D 與 T_5，以判斷 ACE 是否為高頻項目組。經更新計算後，得到的高頻項目組有 ACE、BCE、AC、AE、BC、BE 及 CE，分別計算其可能形成的關聯規則，若滿足最小信賴度，則關聯規則成立。

五、效能評估

在不失一般情況下，我們使用隨機亂數來產生每筆交易資料的項目，做為評估前面章節演算法之執行效能的資料來源。實驗平台為 P4-1.7 G、RAM 為 512M、作業系統為 Windows 2000 Sever，使用 C#來撰寫程式。我們分別說明前面所描述之演算法的執行效能如下：

假設項目共有 26 項，每筆交易資料所包含的項目以亂數產生，我們分別使用 Apriori 演算法(每次新增交易資料就須重新計算)、第三節及第四節所描述之演算法，來找出關聯規則，並評估三者的執行效能。在圖 3 中，我們初始以 10 萬筆交易資料為探勘的資料來源，在設定最小支持度為 0.4 及最小信賴度為 0.7 的條件下，然後以每次新增 1000 筆交易資料，來評估三個演算法的執行時間。

在每次新增交易資料時，第一種以 Aprioi 演算法來探勘關聯規則，且每次都須重新計算，其執行時間大致上呈現一條緩慢上升的直線。而第三節所描述的演算法，一開始探勘的執行時間，與原先 Apriori 演算法之執行時間相同，由於第一次探勘時就將其過程中的項目組與其出現次數儲存於記憶體中，每次新增交易資料時，其執行時間會遠少於第一次探勘所花的執行時間。由第四節所描述之以布林運算為基礎的演算法，其執行效能更優於第三節所描述的演算法。

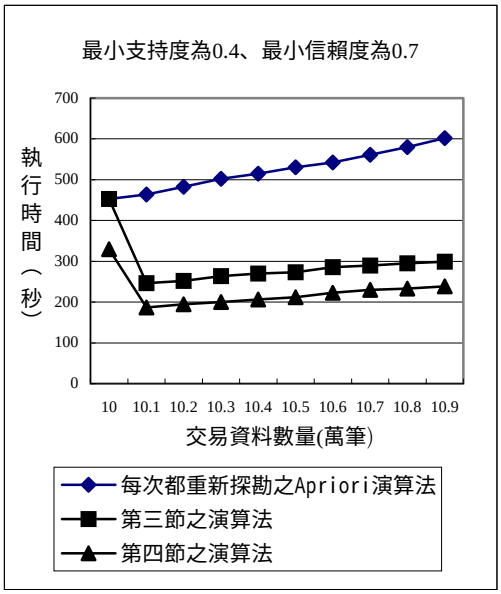

圖 3

六、結論

探勘關聯規則是資料探勘技術中最重要的研究主題之一，但在面臨有新增交易資料時，就必須重新計算的探勘方式，將造成前次探勘計算的重複及資源的浪費，因此，如何避免探勘計算的重複，以較少的計算時間來線上探勘關聯規則，即成為探勘關聯規則最重要的問題之一。在本篇論文中，我們提出兩個演算法來線上探勘關聯規則：一是先以 Apriori 演算法來找出所有的關聯規則，在探勘的過程中，我們保留計算過程中的項目組與其出現次數於記憶體中，當有新增交易資料時，則考量新增交易資料對記憶體中項目組之出現次數的影響，若有產生新項目組時，則必須掃瞄全部的交易資料，以判斷這些新產生的項目組是否為高頻項目組；二是我們以布林運算為基礎，提出一個演算法，將可有效地提升前一方法的執行效能。雖然對新產生的項目組必須掃瞄原先資料庫與新增交易資料，但相較於必須重新探勘關聯規則而言，仍然減少很多重複性的計算過程。從效能評估顯示，我們所提出的探勘方法，將可有效地提升線上探勘關聯規則的執行效率。

七、參考文獻

[1] 陳可欣，*在動態交易資料庫中探勘線上關聯法則之設計與分析*，臺南師範學院資訊教育研究所，碩士論文，2001。

[2] 蘇家輝，*線上多維度關聯規則採掘系統之架構*，義守大學資訊工程所，碩士論文，2002。

[3] 陳垂呈，“以有效率的布林演算法來擷取關聯規則”，*2002 數位生活與網際網路科技研討會*，台南，成功大學，六月，2002。

[4] R. Agrawal, T. Imielinski, and A. Swami, “Mining Association Rules between Sets of Items in Very Large Database,” *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 207-216, 1993.

[5] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules in Large Database,” *Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487-499, 1994.

[6] J. S. Park, M. S. Chen, and P. S. Yu, “Using a Hash-Based Method with Transaction Trimming for Mining Association Rules,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, No. 5, pp. 813-825, 1997.

[7] R. Srikant and R. Agrawal, “Mining Generalized Association Rules,” *Proceedings of the 21th International Conference on Very Large Data Bases*, pp. 407-419, 1995.

[8] S. Y. Wur and Y. Leu, “An Effective Boolean Algorithm for Mining Association Rules in Large Databases,” *DASFAA*, 1999.

[9] D.W. Cheung, J. Han, V. Ng, and C.Y. Wang, “Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique,” *Proc. Int'l Conf. Data Eng.*, 1996.

[10] N. F. Ayan, A. U. Tansel and E. Arkun, “An Efficient Algorithm to Update Large Itemsets with Early Pruning,” *Proc. of 1999 Int. Conf. on Knowledge Discovery and Data Mining*, 1999.