對動態新增資料集挖掘技術之設計

邱宏彬
清雲技術學院助理教授
hpchiu@cyit.edu.tw

毛立仁
南華大學資訊管理所碩士班研究生
lewismlr@ms17.url.com.tw

摘要

本篇論文提出了一個名為 MUM(Multidimensional Update Mining)的演算法，用以抽取各種交易資料庫中的關聯法則(Association Rules)[1,2,3,4,5,6,7,8]。此一演算法是以 Apriori 演算法爲基礎所推導出的一個資料挖掘技術，應用於目前市面上動態新增的資料庫，其資料挖掘的方式，是利用多個支持度(multi-supports)及備用資料表(temp table)的觀念，將主要的相關項目組(Large-Itemsets)及支持度不足的項目組分別處理，而支持度不足的項目儲存在備用資料庫中。日後新增資料時僅需對新增的部分做處理，再與預先處理過的資料合併計算其支持度，無須重新掃描資料庫。如此即可對新增的資料做有效率的處理，並能利用多維資料庫理念節省 I/O 存取時間，降低資料挖掘過程的成本，提高使用者的整體效益。

關鍵詞：Data Mining 、Multidimensional Update Mining、Association Rules。

一、 研究動機與目的

目前以 Apriori 演算法爲基礎所推導出的各種資料挖掘技術，都是針對靜態的資料做挖掘的動作，而從中擷取出使用者有興趣的相關法則。但在現實生活中，資料是日漸遞增的，如便利商店的銷售商品、醫院每日的門診資料等，都是在動態的增加資料庫中的資料項目；若是依照 Apriori 演算法，則每當有新增資料的時候，都必須重新掃描資料庫，如此的方式實在太浪費計算的時間及硬體的 I/O。如何在動態新增資料項目的時候，能夠兼顧時間成本及搜尋效率，唯一的方式就僅有將已經挖掘過的資料妥善的處理，以便新增資料的時候能再重複利用[10][11][12][13]。

而在目前的研究中，以 DHP[7][14]及 FUP[2]演算法作爲加速擷取相關法則及對動態新增的資料，提出了相對應的演算方式，但是在舊有的資料項目組中，有些可能爲日後大項目組的資料集，會因爲前面提及的演算法在最初的擷取過程中，遭到刪除，對於新增的資料項目組中，也無法達到使用者預設的支持度門檻值(minimum supports)，而被誤認爲不重要的

相關法則，因此無法有效的挖掘出其中的關聯性。

本篇論文試著將每次動態新增過的項目組，加以挖掘過之後，妥善的處理，以利每次新增的擷取過程，可直接與計算過的資料之接合併，如此便可以順利的改善必須每次重新掃描資料庫的缺點，也不會因爲刪除部分的相關法則，而導致誤將具有潛力的項目組事先刪除，無法達到資料挖掘的目的。

二、 相關研究

1. Apriori 演算法

Agrawal et al. 在 1974 年首先提出了一種名爲 Apriori 的演算法，而此一演算法已成爲現今在研究關聯法則中最具代表性的演算法之一。Apriori 演算法中包含了以下兩個重要的步驟，整體的架構如圖所示：

(1)反覆的產生候選項目組合搜尋整個資料庫，直到找出所有的大項目組。

(2)利用(1)所找出的大項目組，推導出所有的相關法則。

在步驟(1)中，候選項目組的支持度必須大於或是等於使用者所設定的最小支持度門檻值，才能成爲大項目組。同樣的，在步驟(2)中所產生的關聯法則期可靠度也必須達到使用者最初設定的最小可靠度門檻值。

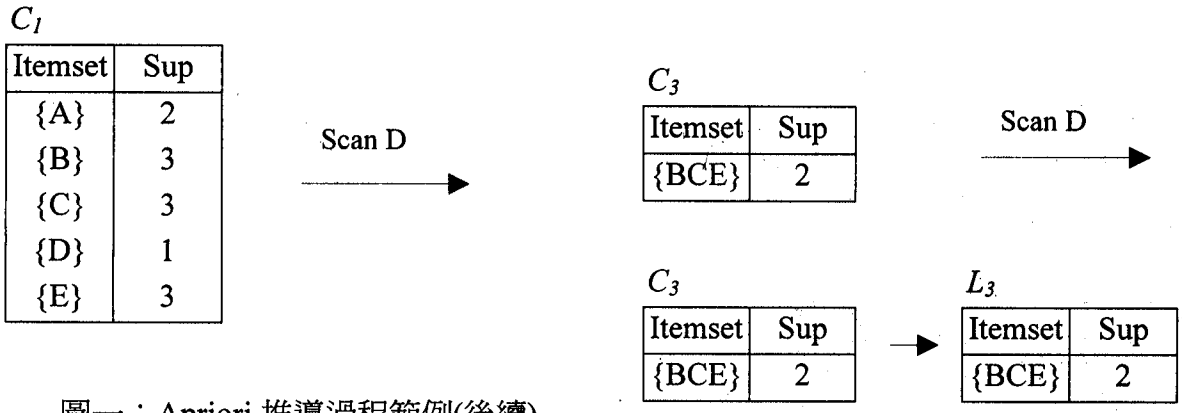
在步驟(1)產生大項目組的過程中，Apriori 演算法由單一項目組(1-itemset)開始，逐層產生相關項目組。此過程分爲兩個階段，第一個階段爲產生新的項目組，若相關項目的長度爲 k，則稱爲候選 k-項目組(candidate k-itemset)，記爲 C_k；第二階段爲搜尋資料庫中 C_k 的支持度是否大於使用者最初設定的最小支持度門檻值的限制，符合條件的項目組 C_k 便稱爲大項目組(或稱爲高頻項目組)，稱爲大 k-項目組(large k-itemset)，記爲 L_k 而不符合最小支持度限制的 C_k 項目組則刪除。

根據以上的步驟，而後再由 L_k 與 L_k 的聯集產生下一層的新候選項目組 C_{k+1}，並再搜尋資料庫以產生 L_{k+1}。如此反覆遞迴產生下一層級的 C_{k+1} 與 L_{k+1} 直到資料庫中所有的大項目組均被搜尋出來爲止。

根據以上所描述的步驟，推導 Apriori 演算法的範例如下：

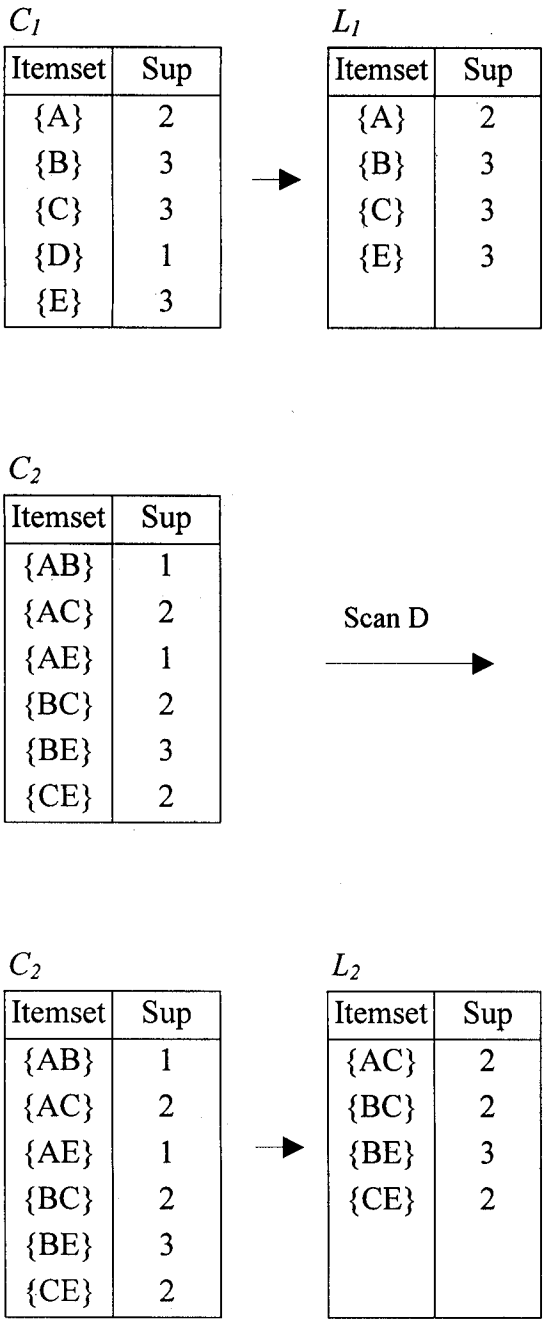
Database D

TID	Items
100	ACD
200	BCE
300	ABCE
400	BE



圖一：Apriori 推導過程範例(後續)

圖一：Apriori 推導過程範例(續)



2. Multidimensional Update Mining (MUM) 技術之探討

Apriori 演算法在最早提出的時候，是假設資料是靜態的，需要反複掃描整體的資料庫直到找出使用者有興趣的相關法則。但是在現實生活中，交易資料常爲動態新增，若是每次新增資料的時候，就必須重新掃描一次原始資料庫，則會浪費太多的時間。隨者日漸增加的資料儲存量，資料挖掘的工作會變得負擔太重而沒有效率。

本篇論文欲利用多維陣列的資料表格方式，將 1-Itemset、2-Itemsets、....、n-Itemsets 建立一個多維陣列，以提供每一筆資料新增時，動態產生大項目組，紀錄相對應的資料項目，並將利用備用資料表(temp table)的觀念，將不符合支持度的項目組移至 temp table，以減少主資料表格(base table)的資料量，加快資料庫搜尋速度。

在資料挖掘最初階段，以多個表格來

記錄大項目組的出現次數(count)值及可信度，若此一資料爲目前主資料表格中沒有之資料，則動態新增此一項目，在主資料表格中搜尋不到所要尋找的項目組時，則可以退到備用表格搜尋，並記錄該項目組的資訊，以便下次新增資料時可以即時更改所需資料而不需要掃描整體資料庫，而在備用資料表中的項目組，相對應的資料新增的時候，其 count 的值還是會動態增加，當其支持度達到 base table 的支持度門檻值時，即可移入 base table，成爲大項目組。如此既可以節省 I/O 及 CPU 運算時間，提高擷取速度，又不至於刪除日後具有潛力的項目組，對於目前動態新增的資料項目，提供一種十分良好的演算方法。

以下爲原始資料表格及 MUM 演算法的推導過程：

step1：讀入目前資料項目。

step2：動態將資料分解成 1-Itemset、2-Itemsets、...、n-Itemsets，並掃描對應的 cube 中是否已經有資料存在。

step3：有資料：將對應資料 count 數加 1
無資料：新增項目組。

step4：計算此項目組的 support 是否達到目前的 minimum support。

a. 是，則繼續留在此 base table 中。

b. 否，則移到 temp table 中。

step5：將目前的資料指標移到下一筆，重新回到 Step1，直到資料全部掃描結束。

此一演算法雖然會記錄過多的項目組，但是在 CPU 的計算及硬體 I/O 的考量下，多維陣列的儲存方式，是可以有效提供較佳的搜尋速度，以便使用者快速尋找相關法則。而以目前硬體配備的價格日趨下降的趨勢來看，資料存放空間已經不再是一個嚴重的問題了。以下是利用 MUM 演算方法的推導過程。

假設第一次新增的資料爲原始的資料項目，其資料如下：

表一：原始資料表

Incremental database	
TID	Items
100	ACD
200	BCE
300	ABCE
400	ABE
500	ABE
600	ACD
700	BCDE
800	BCE

而在此一演算法中所提出的多維陣列，則是將每一層級的項目組(Itemsets)加以區分，分別存入不同的陣列中，而此處所提出由的每一個層級的項目組集合，都是一個資料表，如表二所示，相對於 1-Itemset 的資料項目組，就存放於 1-Itemset Table 表格中；2-Itemset 的資料項目組，就存放於 2-Itemset Table 表格

中，以此類推。而由於存放的表格不同，資料量相對的減少許多，加上備用資料表的觀念，實際儲存於主資料表，成爲大項目組的資料量可以大大的減少，使得每一次新增交易時，可以快速的將新增的項目逐一加入對應的表格中，以下是 MUM 的資料表：

表二：MUM 資料表

		3-Itemsets	Supports
	2-Itemsets	Supports	
1-Itemset	Support		

3-Itemsets Table

2-Itemsets Table

1-Itemset Table

而未通過最小支持度門檻值的項目組也必須加以紀錄，以備下次新增資料時彙總資料之用；而備用資料表的格式如同主資料表，其型式如表三所示 ：

表三：MUM 備用資料表

		3-Itemsets	Supports
	2-Itemsets	Supports	
1-Itemset	Support		

3-Itemsets Table

2-Itemsets Table

1-Itemset Table

資料挖掘方式如以下步驟，共分爲兩個階段：

(1) 將每一層的 Itemstes 定義不同的 minimum support，假定 1-Itemset 到 4-Itemsets 的 minimum support 分別爲 50,20,10,5 等四個不同的門檻值。

(2) 逐一將資料表格欄位中資料放入對應的資料表中，若是表格內無此項目組，則動態新增此一項目組；若已經存在時，則將其對應項目組的 Support 的值加一。並同步計算出其目前支持度的值。

以此種方式推導，直到有某一個項目的支持度經過計算不足其 minimum support 時，則將此一項目組將會移到 temp table 中。

表四是將 MUM 資料表依照其層級分解，紀錄此一交易資料表的每一個項目集合，並用行列轉換的方式，將各筆交易資料紀錄至表格中，左方的部份爲資料庫中

交易的資料項目組，上方爲 1-itemset 的項目，而表格中的資訊則是其出現次數及支持度：

表四：1-ItemSet 對應表

Pass		A	B	C	D	E
1	ACD	1/100	/	1/100	1/100	/
2	BCE	1/50	1/50	2/100	1/50	1/50
3	ABCE	2/66.6	2/66.6	3/100	1/33.3	2/66.6
4	ABE	3/75	3/75	3/75		3/75
5	ABE	4/80	4/80	3/60		4/80
6	ACD	5/83.3	4/66.6	4/66.6		4/66.6
7	BCDE	5/71.4	5/71.4	5/71.4		5/71.4
8	BCE	5/62.5	6/75	6/75		6/75

minimum support = 50%

請注意掃描過程中，其中項目{D}在 pass3 的過程中，因爲不足 50%的 minimum support，所以被移入 temp table 中，但是在往後的 pass 過程中，其 count 數值還是會累加；若是在某次交易時，在 temp table 中項目{D}的 minimum support 值大於 50%的時候，依然會移入 base table 中。以此種方式，可以迅速的新增資料，不須重新掃描原始資料庫。

表五表示在處理此交易資料時，不足支持度的資料被移入 temp table 的情形。

表五：備用資料表

pass	Items	Support
1		
2		
3	D	33.3
4	D	25
5	D	20
6	D	33.3
7	D	42.8
8	D	37.5

在表四的 pass3 的過程中，項目組{D}被移出 base table，此時備用資料表就建立一個{D}的項目，並將其 support 的值紀錄起來，往後的資料新增時，其 support 的數值依舊會即時的做異動，若是其支持度滿足預設的 50%時，再將其項目組{D}移入 base table 中。

對於 2-Itemset 、 3-Itemset 、 .... 、 n-Itemsets 的資料表，都是依照上述的處理流程，將各項目逐一置放到對應的表格中，以下是各表格中資料放置的情形：

表六爲 2-Itemset 的演算過程：

表六：2-Itemset 對應表

pass		AB	AC	AD	AE	BC	BD	BE	CD	CE	DE
1	ACD		1	1					1		
2	BCE					1		1		1	
3	ABCE	1	2		1						
4	ABE	2			2			2			
5	ABE	3			3			3			
6	ACD		2	2					2		
7	BCDE					2	1	4	3	2	1
8	BCE					3				3	

minimum support = 20%

表七爲 3-Itemset 的演算過程：

表七：3-Itemset 對應表

Pass		ABC	ACD	ABD	ABE	BCD	BCE	CDE
1	ACD		1					
2	BCE						1	
3	ABCE	1			1		2	
4	ABE				2			
5	ABE				3			
6	ACD		2					
7	BCDE					1	3	1
8	BCE						4	

minimum support = 10%

表八爲 4-Itemset 的演算過程：

表八：4-Itemset 對應表

		ABCD	ABCE	ABDE	ACDE	BCDE
pass 1	ACD					
pass 2	BCE					
pass 3	ABCE		1			
pass 4	ABE					
pass 5	ABE					
pass 6	ACD					
pass 7	BCDE					1
pass 8	BCE					

minimum support = 5%

由此一方式可以快速的處理新增的交易，而不須將舊有的原始資料表重新掃描，大大的增進了資料挖掘的速度。

2. MUM 演算法與其他方法之比較

對於 MUM 演算法來說，其所著重的部分有兩以下兩點：

(1) 對於資料的處理，是直接由資料的第一筆向下逐一處理到最後一筆，將支

持度及可信度直接加總，無須反覆的掃描資料庫以求取各階層項目組的相關資訊。

(2) 對於新增的資料而言，無須重新掃描整個資料庫，僅需要對新增的資料加以處理，再彙總之前處理過的資料，即可對所有的關聯項目進行計算相關資訊。

假設資料庫中的交易有 N 筆資料，而所有交易的資料中全部有 l 個項目，新增的交易有 m 筆資料，N 遠大於 m；一般的演算法必須在每次新增資料的時候，重新掃描含有新增資料的整體資料庫，而 MUM 演算法則僅需對新增資料做處理，再彙總原有的資料即可。

若是計算一般演算法的複雜度，其複雜度爲：

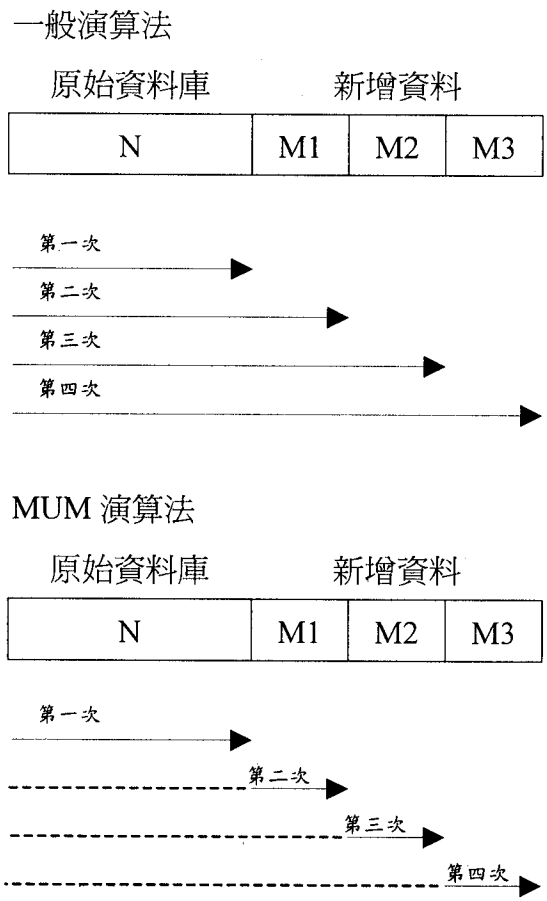
$$\Omega(l^2(N+m)) \qquad (1)$$

而對於 MUM 演算法來說，其複雜度爲：

$$\theta(N+m) \qquad (2)$$

當這兩個公式可以看出，資料庫中新增交易的數量越多的時候，這兩種演算法複雜度的差距也就越來越大，MUM 演算法的優點也就顯明易見了。

以下爲一般演算法及 MUM 演算法對於新增資料的處理所耗費時間之比較示意圖：



圖二：MUM 演算法與一般演算法之比較

圖二中虛線部分即爲節省掃描資料庫的時間成本，我們可明顯的看出其 MUM 演算法在新增資料時較 Apriori 演算法節省許多重複搜尋資料的時間，但是 MUM 在儲存的空間上，卻會花費比 Apriori 演算法大上數倍的存放空間，算是一種以儲存空間換取資料挖掘時間的方式。

三、結論與未來研究之方向

本文中所提出的 MUM 演算法，可以有效的解決目前現實生活中動態新增的

交易資料，在挖掘相關法則時的效率，而以下列出的幾個相關的研究方向，依然值得我們繼續的探討及研究。

1. Temp table 資料表格之設計，以及多個支持度的門檻值的設定。
2. 在動態掃描交易資料時，以何種方式(動態或是批次)將項目組移出 base table，或是將 temp table 中滿足支持度的項目從 temp table 中移入 base table。
3. 動態或是批次移動項目組的最佳策略方式探討。
4. 如何設計能有效節省儲存空間的資料結構，或是利用資料挖掘的軟體來實現此一技術。

四、參考文獻

1. R. Agrawal, T. Imielinssi, and A. Swami, "Mining association rules between sets of items in large database, " the ACM SIGMOD Conference, Washington DC, USA, 1993
2. R. Agrawal and R. Srikant, "Fast algorithm for mining association rules, " the International Conference on Very Large Database, pp. 487-499,1994
3. R. Agrawal, R. Srikant, and Q. Vu, "Mining association rules with item constraints, " in 3th International Conference on Knowledge Discovery in Database and Data Mining, Newport Beach, California, 1997.
4. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining optimized association rules for numeric attributes, " the ACM SIGACT–SIGMOD -SIGART Symposium on Principles of Database Systems, pp. 182-191,1996.
5. J. Han and Y. Fu, "Discovery of multiple-level association rules from large database, " in 21th International Conference on Very Large Databases, Zurich, Swizerland, pp. 420-431, 1995.
6. H. Mannila, H. tovionen, and A. Inkeri Verkamo, "Efficient algorithm for discovering association rules," Proc AAAI Workshop Knowledge Discovery in Databases, pp. 181-192, 1994.
7. 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.812-825, 1997.
8. R. Srikant and R. Agrawal, "Mining generalized association rules," in 21th International Conference on Very Large Databases, Zurich, Swizerland, pp. 407-419, 1995.
9. R. Srikant and R. Agrawal, "Mining

quantitative association rules in large relational tables," the 1996 ACM SIGMOD International Conference on Management of Data, Monreal, Canada, June, pp. 1-12, 1996.

10. D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large database; an incremental updating approach," in 12th IEEE International Conference on Data Engineering 1996.
11. D. W. Cheung, S. D. Lee, and B. Kao, "A general incremental technique for maintaining discovered association rules," in Proceedings of Database Systems for Advanced Applications, DASFAA'97, Melbourne, Australia, pp. 185-194, 1997.
12. M. Y. Lin and S. Y. Lee, "Incremental update on sequential patterns in large databases," in 10th IEEE International Conference on Tools with Artificial Intelligence, 1998.
13. N. L. Sarda and N. V. Srinivas, "An adaptive algorithm for incremental mining of association rules," in 9th International Workshop on Database and Expert Systems, 1998.
14. J. S. Park, M. S. Chen, and P. S. Yu, "An Effective Hash Based Algorithms for Mining Association Rules," Proc. ACM SIGMOD, pp.175-186, 1995.