以利潤為主要考量之多重最小支持度量化關聯規則

康聖祥　鄭印良　羅貴魁　楊達立
國立虎尾科技大學資訊管理系
n461106@moon.nfu.edu.tw

摘要

一般關聯規則的特性主要是考慮商品在交易中的關聯性，使得與重要商品相關的關聯規則能被挖掘出來；而挖掘量化關聯規則的主要目的，則是從交易資料庫中找出大部分的客戶購買了哪些數量的商品，也會同時購買哪些數量的其他商品。但是，一般關聯規則沒有考慮商品被購買的數量，而量化關聯規則也沒有考慮到商品本身的利潤，因此如何同時考慮商品本身的利潤與其被購買的數量，成為本研究的議題。

本論文主要應用分割演算法(Partition algorithm)將其資料量化，之後建立成 P-tree 的結構，再加上 ABC 方法及多支持度的概念，提出了一個新 PMSFP-tree（Profit and Multiple minimum Supports Frequent Patterns tree）結構與 PMSQFP-growth(Profit and Multiple minimum Supports QFP-growth)演算法。

關鍵字：資料探勘、關聯規則、FP-tree、QFP-growth、ABC 方法、量化關聯規則。

Abstract

Association rules takes an important rule in Data Mining. By applying this technique, the relations between the data in a transaction database can be found, and then managers utilize the explored information to make decisions. The traditional association rules technique mostly focuses on the amount of trades, so that the goods with high profits and low sales volume will be neglected. This paper proposes a method using the concept of P-tree (Patterns tree), CFP-growth and QFP-growth. Subsequently, the common stock-checking method, ABC, is utilized to classify the goods so as to those goods with high profits and high support will be remained. Finally, the explored information will be much expected to conform to managers. By combining the concepts as mentioned above, a novel technique, PMSFP-tree (Profit and Multiple minimum Supports Frequent Patterns tree) and PMSQFP-growth (Profit and Multiple minimum Supports QFP-growth) is proposed in this paper. Experimental results demonstrate that our method can improve the drawbacks of CFP-growth. Besides, by taking account of trade profits, the mined results will be much respond to its actual facts.

Keywords: Data mining、Multiple minimum supports、FP-tree、QFP-growth、ABC Approach、Quantitative Association Rules.

1. 前言

關聯規則主要是被利用來尋找資料庫中資料間彼此的相關性。因此，若能藉由此關聯規則的特性，找出消費者與產品特徵之間的關係，相信應該更能掌握消費者的心態，開發出迎合消費者興趣的產品。這也正是本研究想利用關聯規則，瞭解於產品開發過程中消費者與產品特徵間相互關係的主要動機。以往在探勘交易資料的過程中，多僅考慮各商品在交易中是否被購買，而忽略了其他資料的可利用性，將造成挖掘出來的法則有所偏差。本研究在挖掘關聯規則時除考量交易中哪些商品被購買之外，還希望能利用資料庫中的商品資訊如利潤、購買數量等，將之回饋在關聯規則產生的過程中，使挖掘出的法則更為合理、更具有價值，達成

讓商家感興趣的項目可以被保留或突顯出來之目標。

2. 文獻探討

2.1 關聯規則

關聯規則探勘(Association rule mining)是從大量資料項目集合之間發現有趣的關聯或相關(interesting association or correlati on relationships)。隨著大量資料不停地被收集和儲存，許多業界人士對於他們的資料庫中探勘關聯規則越來越感到興趣。從大量商業交易記錄中發現有趣的關聯規則，可以幫助許多商業決策的制定，如產品目錄設計(catalog design)、交叉行銷(cross-marketing)和賠本銷售分析(loss-leader analysis)。

關聯規則探勘首先由 Agrawal，Imiclinski 和 Swami[9]於1993年提出，主要是用來挖掘資料庫中交易項目間的關聯性。假設$I=\{i_1,i_2,i_3,\ldots,i_n\}$，$I$ 為所有商品項目(items)所成的集合，一筆交易(Transaction)紀錄，紀錄著某一顧客於某次所購買的商品交易明細。交易資料庫(transaction database)則是由許多交易資料所組成。交易資料庫中的交易資料，包括交易編號、客戶編號、所購買的項目、時間 …等。項目集(Itemsets)是由項目所構成，每一次交易即為一個項目集，項目集長度(length)為項目集所包含項目的個數，一個長度為K的項目集，我們稱此項目集為k-項目集。

假設 X 與 Y 都是 I 的子集合，D 是資料庫交易的集合，$X \subset I$，$Y \subset I$ 並且 $X \cap Y = \phi$ 。Support(X→Y)表示，D 交易總數中包含 $X \cup Y$ 的百分比，也就是說若購買 Itemset{X}後會購買 Itemset{Y}的可能性。Confidence(X→Y)表示，D 交易總數中包含 X 的交易同時也包含 Y 中的百分比，也就是說在 Itemset{X}發生的條件下 Itemset{Y}發生的比例。

2.2 單支持度關聯規則演算法

在挖掘關聯規則的領域中，主要可以分成使用 Apiori-like[6]的方法跟使用 None-Apriori-like 的方法，第一類會產生候選頻繁項目集，之後在找出符合最小支持度的頻繁項目集，第二類會使用 Non Apriori-like[4,5,7]的方法找出頻繁項目集。而第一類中可能需要產生大量的候選頻繁項目集合及重複掃資料庫，所以明顯第二類的方法優於第一類。關聯規則最早是由學者 Agrawal 等所提出[9]，其主要是挖掘出交易資料庫中交易項目間之關聯性。關聯規則的形式可表示為 X→Y，其中 X 和 Y 分別代表項目的集合，在購物籃分析的應用上即表示若購買 itemset{X}後會購買 itemset{Y}的可能性。如(圖 1)所示，假設交易資料庫中的交易紀錄共有四筆交易資料，所有交易項目為{A,B,C,D}四項，若我們設定最小支持度門檻值為 50%及最小信心水準門檻值 50%。

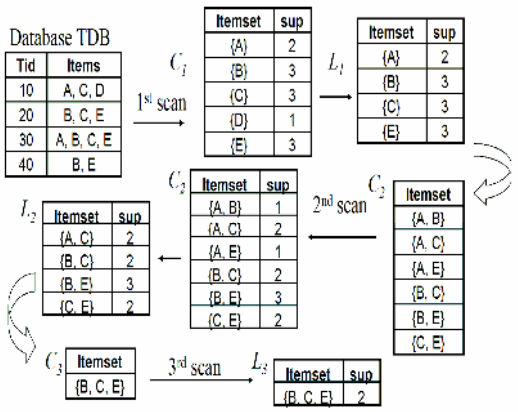

圖 1 產生 large itemset 的過程

最後挖掘出來的結果如下：

Frequent-Patterns-List：{A, B, C, E, AC, BC, BE, CE, BCE} Associational Rules：

A→C (Confidence=2/2) C→A (Confidence=2/3)

B→C (Confidence=2/3) C→B (Confidence=2/3)

B→E (Confidence=3/3) E→B (Confidence=3/3)

C→E (Confidence=2/3) E→C (Confidence=2/3)

B→CE (Confidence=2/3) C→BE (Confidence=2/3)

E→BC (Confidence=2/3) BC→E (Confidence=2/2)

CE→B (Confidence=2/2) BE→C (Confidence=2/3)

在 none-apriori like 演算法中，以 FP-tree(Frequent Pattern Tree)的結構結合 FP-growth 演算法的效率較為突出，並且解決 Apriori-like 方法中，產生大

量的候選項目集合及重複掃瞄資料庫的缺點，而在整個資料關聯規則挖掘中，如果能減少重覆掃描資料庫的次數將可大大減少 I/O 所花費的時間，可見 Frequent Patrrern 有舉足輕重的地位。

由於 FP-tree 通常都比原本的交易資料庫小很多而且可以直接儲存於主記憶體中來執行，所以只要 FP-tree 建構完成就可以快速的以 FP-growth 的演算法[3]來產生 frequent patterns，FP-Growth 演算法由 Header table 中的項目欄依由下而上(Bottom_Up)的順序探勘。

如何去避免 FP-growth 演算法在整個挖掘中不斷被重覆地產生 conditional bases 及 conditional FP-trees，有學者提出一個演算法稱之為 QFP-growth[6]演算法來改善大量產生 conditional bases 及 conditional FP-trees 的問題。其運用暫存根目錄(temporary root)的技術，使得 QFP-growth 演算法在挖掘頻繁項目集過程當中減少了執行時間及相關儲存空間，且這個方法在挖掘大型資料庫或資料倉儲上是很有效率且可靠的。其演算法是以排序方式產生頻繁項目集，所以挖掘的結果是有排序的，相較於運用 FP-growth 方式產生的結果在使用方面上將更為方便。

2.3 多支持度關聯規則演算法

在傳統的方法中所有的交易項目使用單一最小支持度門檻值，然而在現實的生活中不同的交易項目有其不同的重要性，例如在企業中 20%商品通常創造 80%的利潤，這些項目大都是一些奢侈品或高單價的物品。在這種情況下，如果將單一支持度門檻值設的太高則無法挖掘出有用的關聯規則來，相反的設的太低將會挖掘出一堆沒有意義的規則。多重最小支持度（Multiple minimum supports）關聯規則最早在 1999 年由 Liu[1]等學者提出，主要是以 MSapriori 的方式來挖掘出多重最小支持度的所有交易項目之頻繁項目集；利用多重支持度門檻值來進行關聯規則挖掘是一項相當重要且符合現實生活的資料探勘技術，與傳統單一門檻值的關聯規則挖掘比較，它允許使用者可以針對每個不同交易項目設定不同的最小支持度門檻值，以反映真實世界中購買各種商品頻率不一的問題。

2.4 ABC 方法(ABC approach)

此方法是在 1950 年代由通用電機 (General Electric)公司H.F. Dickie所提出。其利用重點管理觀念，用來確認與控制存貨項目，將存貨項目加以分類，然後據此施以管理。通常存貨項目分為三等：A(很重要)、B(次重要)、C(不重要)。而A類存貨項目佔總存貨項目5%到10%，但金額卻佔60%到70%，C類存貨項目佔總存貨項目60%，而金額卻佔15%。此百分比每家廠商會有不同，但重點在於通常相當少的存貨項目佔有存貨成本的大部份，而這類存貨會得到較多的關照。

2.5 量化關聯規則

量化關聯規則主要的方法與傳統的二元布林關聯規則相似，僅是在整個挖掘的過程當中把交易資料中的購買數量加了上去，因為不同的交易資料中可能有不同的購買數量，若這些購買數量分佈的範圍很廣，為了降低演算法的複雜度，會將各物品的購買數量分割成區間或經由 Fuzzy sets 的觀念將數量轉換成相對應的區間值或歸屬度。從交易資料庫中找出交易次數與數量頻繁的交易項目，並從中挖掘出頻繁交易項目間之關聯規則就是挖掘量化關聯規則的主要目的。在這些量化關聯規則當中可以得知那些項目會被依何種數量一起被購買。在不考慮數量的關聯規則中可以明確的給定最小支持度門檻值（minimum support threshold）進而找出所有的關聯規則來，但是在挖掘量化關聯規則當中，交易項目會因購買數量的不同而被視為不同的交易項目，由於同時考慮“交易項目”及“購買數量”時，會造成滿足最小支持度門檻的交易項目變得很少，甚至可能無法產生任何滿足最小支持度門檻值的量化關聯規則。為了解決這個問題，可以將“數量”分割成許多區間（intervals），提高每一個交易項目在其所屬區間的支持度，進而挖掘出更多潛在有用的關聯規則。

3. 研究方法

在前面文獻探討過程當中，我們得知 FP-growth 演算法中在建構 FP-tree 結構於整個 frequent patterns mining 過程僅需要掃描原資料庫 2 次，大大的改善 apriori-like approach 中重覆產生的 candidate generation-and-test 及 I/O 的花費，而 QFP-growth 演算法避免 FP-growth 演算法挖掘過程中需要重覆不斷產生 conditional frequent-bases 及 conditional FP-trees 的瓶頸，其運用暫存根目錄 (temporary root)的技術，使得 QFP-growth 演算法在挖掘頻繁項目集過程當中減少了執行時間及相關儲存空間，且這個方法在挖掘大型資料庫或資料倉儲上是有效率且可靠的。其演算法是以排序方式產生頻繁項目集，所以其挖掘的結果是有排序的，比用 FP-growth 方式挖掘出的結果在應用方面上更為方便。接下來文獻中也提出以多支持度的演算法來探勘資料庫，並且說明應用多支持度來探勘資料比較附合現實情況之需求，也可解決單支持度向下封閉之特性。之後也由 ABC 方法來得知一般較少的項目佔大部份的存貨的觀念，最後提及量化關聯規則，讓每一個交易加上數量的想法，讓資料更附合真實的買賣交易。

將整個交易資料庫中的所有交易資料給予量化，在此我們應用分割演算法來達到量化關聯規則，其中應用到多支持度的觀念，因此最小支持度門檻值(minimum support threshold) 會依每個項目的支持度來當其門檻值，進而將將“數量”分割成許多區間（intervals），進而挖掘出更多潛在有用的關聯規則。假設全部的交易總數為 T，支持度門檻值（lower and upper support threshold）為 S_l，其分割的演算法如下：

Algorithm 1 (Partition algorithm)

Let $\{\langle i,q_1\rangle,\langle i,q_2\rangle,\cdots,\langle i,q_n\rangle\}$ be the set of q_items in DB which have the same item i, and $q_1 < q_2 < \cdots < q_n$. Assume that $c_1, c_2, \cdots,$ and c_n are the numbers of transactions containing $\langle i,q_1\rangle,\langle i,q_2\rangle,\cdots,$ and $\langle i,q_n\rangle$, respectively.

Let T be the total number of transactions.

First, we find a minimum integer k_1 such that $\frac{\sum_{m=1}^{k_1} c_m}{T}$ is greater than or equal to the lower support threshold, say S_l, and generate the interval $\lfloor q_1..q_{k_1} \rfloor$

Then, we find a minimum integer $k_2 (k_2 > k_1)$ such that $\frac{\sum_{m=k_1+1}^{k_2} c_m}{T} \geq S_l$, and generate the interval $\lfloor q_{(k_1+1)}..q_{k_2} \rfloor$.

The remaining intervals are similarly computed.

If $\frac{\sum_{m=k_j+1}^{n} c_m}{T} < S_l$, the last generated interval $\lfloor q_{(k_{(j-1)}+1)}..q_{k_j} \rfloor$ is combined with $\lfloor q_{(k_j+1)}..q_n \rfloor$ to form the interval $\lfloor q_{(k_{(j-1)}+1)}..q_n \rfloor$

利用分割演算法對購買交易項目的數量做分割，將數量轉換成所屬區間對應的整數。將整個交易資料庫中的所有交易資料，以 prefix tree 相似的結構建構成包含所有交易項目的 P-tree，其中 P-tree 與 FP-tree 不同的地方在於 FP-tree 內僅包含頻繁項目集，但是 P-tree 包含所有頻繁及非頻繁項目集，也就是所有交易資料庫的所有項目集都對應於 P-tree 結構中的相關節點，其如(Algorithm 2)所示：

Algorithm 2 (P-tree construction)
Input: A transaction database DB’ .
Output: P-tree, the pattern tree of DB’.
Method: The P-tree is constructed as follows.

1 . Scan the transaction database DB once. Collect F, the set of items, and the support of each item. Sort F in support-descending order as FList, the list of items.
2. Create the root of a P-tree, T , and label it as “null”. For each transaction Trans in DB do the following.
3. Select the all items in Trans and sort them according to the order of FList. Let the sorted frequent-item list in Trans be [p | P], where p is the first element and P is the remaining list. Call insert tree([p | P], T).

The function insert tree([p | P], T) is performed as follows. If T has a child N such that N.item -name =

p.item-name, then increment N's count by 1; else create a new node N, with its count initialized to 1, its parent link linked to T , and its node-link linked to the nodes with the same item-name via the node-link structure. If P is nonempty,call insert tree(P,N) recursively.

由上述演算法中 FList 就是 P-tree 定義中的 frequent-list，可知要建構 P-tree 正好需要兩次掃描資料庫的動作，第一次是找出所有長度為 1 的 itemsets，第二次則是建構 P-tree，並將所有交易資料項目放入 P-tree 的 node 節點中。

PMSFP-tree 是由 P-tree 為主要結構，經過以下面二個方式來篩選 ：

1. 以 ABC 方法，讓支持度(Counts)乘上利潤來作為分類的考慮因素，之後再刪除 B、C 類中的所有項目。
2. 所有 MIS(a_i)大於等於 minMIS 的所有相關節所建構而成，也就是對 P-tree 執行刪除的動作（Pruning），將那些 MIS(a_i)小於 minMIS 的相關節點刪除。

其相關的演算法如(Algorithm 3)所示：

Algorithm 3 (PMSFP-tree construction)
Construction PMSFP-tree from pruning the original Pattern tree
Input: P-tree
Output：PMSFP-tree, the profit and multiple minimum supports frequent pattern tree of DB.
Method：The PMSFP-tree is constructed as follows.

1. Find out for item of pattern-header table that its MIS(a_i)≦minMIS in the P-tree structure
2. Delete for all corresponding relevant nodes that chain of head of node-link of items which MIS(a_i)≦minMIS
3. Select the all items in Trans and sort them according to the order of Counts(a_i) * Profit(a_i). After that use ABC approach to sort out the B、C class
4. Delete for all corresponding relevant nodes that chain of head of node-link of items which B、C class

用 PMSQFP-growth 探勘 PMSFP-tree 、QFP-growth 演算法來改善 FP-growth 演算法產生大量 conditional bases 及 conditional FP-trees 的問題，並且運用暫存根目錄(temporary root)的技術，使得 QFP-growth 演算法在挖掘頻繁項目集過程當中減少了執行時間及相關儲存空間，且這個方法在挖掘大型資料庫或資料倉儲上是很有效率且可靠的。而且 QFP-growth 演算法是以排序方式產生頻繁項目集，所以挖掘的結果是有排序的，相較於運用 FP-growth 方式產生的結果在使用方面上將更為方便。然而，FP-growth 及 QFP-growth 都是單一最小支持度門檻值方法，造成無法表示所有不同特性交易項目的支持度，因此，Chen[2]等學者於 2006 年提出了一種與 FP-tree（稱為 MIS-tree）和 FP-growth(稱為 CFP-growth)相似的結構與方法來進行多重支持度門檻值下的頻繁項目集挖掘，實驗結果顯示其效率較傳統 MSapriori 演算法好上許多。

因此，本研究提出 PMSQFP-growth 演算法，它保留了 QFP-growth 演算法的所有優點，並且使用了多支持度門檻值方法，讓此演算法更符合現實情況之需求。如(Algorithm 4)所示：

Algorithm 4 (PMSQFP-growth)
Mining frequent patterns without conditional FP-trees construction using multiple minimum supports
Input: A database DB, represented by PMSFP-tree constructed according to Algorithm 3, and multiple minimum supports MIS(a_i) .
Output: The complete set of frequent patterns.
Method: call PMSQFP-growth (root, null).
Procedure PMSQFP-growth (ROOT Q, FP_prefix α)
{ for each item a_i in Q , in top down order{
  // Mining multipath FP-tree
  Sum all count of a_i under Q into a_i.support
  If a_i.support≧MIS(a_i) and minMIS(α)≧MIS (a_i) then {
    generate pattern β =α∪a_i with support = a_i .support;
    construct temp_root with a_i's child node;
    If temp_root has child
    then call PMSQFP-growth (temp_root, β)}
  If sum of count of temp_root's child ≧MIS(a_i)
    then call PMSQFP-growth (temp_root, β)}}

4. 實驗分析與效能評估

傳統 Apriori-like approach 在產生 frequent patterns 時，從長度為 k(k≧1)的 frequent pattern sets 反覆的去產生長度為(k+1)的 candidate patterns sets，再去判斷它在資料庫中出現的相關頻率次數，進而挖掘出之間的相關聯規則出來。然而，在大數量的 frequent patterns、長字元（long pattern）及低最小門檻值（lower minimum support

thresholds）時，這類的方法將遭受到重要的花費管理數以百計的 candidate sets。

PMSQFP-growth 演算法中 P-tree 其樹的高度為 $\max_{T\in DB}\{|\text{item}(T)|\}$，大小等於 $\sum_{T\in DB}|\text{item}(T)|$ 其中的 $|\text{item}(T)|$ 為資料庫(DB)中所有交易資料(T)內長度為 1 的 itemsets 的個數。PMSQFP-growth 演算法使用 QFP-growth 取代 FP-growth 主要是避免產生 conditional pattern based 及建立 conditional FP-tree 的額外執行時間及儲存空間。PMSQFP-growth 演算法中在建構 P-tree 時將屬於交易資料庫的所有頻繁及非頻繁的 1-itemsets 皆存放入 P-tree 的 node 節點中，因此如果 frequent patterns mining 調整改變其相關 MIS 值時，不像 CFP-growth 演算法中 MIS-tree 結構是有所限制地，相反的 PMSQFP-growth 演算法中只需重新篩選 P-tree 結構中符合 ABC 方式的 AB 類的項目並且所有 MIS(ai)大於等於 minMIS 的所有相關節再去產生 PMSFP-tree 的結構而不受 minMIS 值的限制，如此可以大大提昇執行效能及避免重新掃描資料庫的動作，使得能更有彈性的使用 PMSQFP-growth 演算法。

5. 結論

為了維護探勘中多重支持度關聯規則的正確，我們提供了一個很有效率的維護技術 PMSQFP -growth 演算法。這個演算法可以在新增交易資料於目前交易資料庫(動態交易資料庫)以及從目前交易資料庫中刪除舊有或不適用的交易資料的情況下，即時且有效率地調整 PMSQFP-growth 演算法中的 P-tree 結構，避免重建 P-tree 所花費的成本，並加速多支持度關聯規則的產生。它不僅允許使用者可以針對每個不同交易項目設定不同的最小支持度門檻值，以反映真實世界中購買各種商品頻率不一的問題，而且提供調整各個交易項目的最小支持度值以符合實務上的應用，本篇主要是以利潤乘上支持度，並且應用 ABC 方法來過濾不要的項目，讓留在 P-tree 中的項目都是重要利潤來源的 items，因此最後 mining 出來的 Frequent patterns 會更符合管理者所要的資訊，也更附合一般企業中，20%的產品，佔 80%的利潤原則。

6. 參考文獻

[1] B. Liu, W. Hsu, Y. Ma, “Mining association rules with multiple minimum supports ”, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99), San Diego,pp.337-341, CA, USA, 1999.

[2] Hu Y. H., Chen Y. L., “Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism ”, forthcoming in Decision Support Systems, 42(1), October, pp. 1-24, 2006

[3] Han J., Pei, Pei J.,Yin Y.,Mao R., Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, Data Mining and Knowledge Discovery, vol.1,no. 8, pp. 53-87, 2004

[4] J. Roberto and Jr. Bayardo., “Efficiently Mining Long Patterns from Databases”, In Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data, pp.85-93, 1998.

[5] Nicolas Pasquier, et al., “Efficient mining of association rules using closed itemset lattices”, Information Systems, Vol. 24, no.1, pp.25-46, 1999

[6] Qiu Y., Lan Y. J., Xie Q. S., “An Improved Algorithm of Mining from FP-tree”, Machine Learning and Cybernetics, Proceedings of 2004 International Conference, 3(26-29), Aug, pp. 1665 – 1670,2004.

[7] R. Agarwal, C. Aggarwal, and V. V. V. Prasad , ” A tree projection algorithm for generation of frequent itemsets”, In J. Parallel and Distributed Computing, 2000.

[8] R. Agrawal , R. Srikant, “Fast Algorithm for Mining Association Rule ”, In Proc. of The 20th International Conference on Very Large DataBases, 1994.

[9] R. Agrawal, T. Imielinksi, A. Swami, “Mining association rules between sets of items in large database”, In Proc. ACM-SIGMOD Conference Management of Data (SIGMOD’93), pp. 207–216,1993.