在串流資料中探勘多元關聯法則之研究

李建億
台南大學數位學習科技系
leeci@mail.nutn.edu.tw

蘇聖富
台南大學數位學習科技系
su.sf@msa.hinet.net

摘要

近年來，資料串流探勘已成為熱門的研究議題，並常被用來解決日益龐大的交易資料。然而，傳統的探勘方式，大多僅用單一模式(model)尋找交易頻繁的樣式，卻容易忽略了實際存在於交易活動中的交易行為模式。譬如，冰品在白天比熱食更為暢銷，將使熱食的交易行為不易被發現並容易被忽略。本研究使用動態串流樹(Dynamic Data Stream Tree)探勘多元關聯法則。使用動態多個循環模式找尋交易的行為樣式，遠比採用單一模式演算法更具多樣化、更準確，且更符合實際存在的交易模式。

關鍵詞：資料串流探勘、頻繁樣式、關聯法則

ABSTRACT

Data stream mining is a hot research topic in recent years ,which can sovle the increasingly extensive transactions data. However, the traditional mining methods merely use the single model to find out these frequent trading patterns, which may ignore the practial trading activities in trading patterns easily. For instance, if the ices are sold much better than the hot foods in the daytime, the trading patterns of the hot foods may not be find out and it may be ignored esaily. The study uses dynamic data stream tree and mining multiple association rules. Using the multiple dynamic cycle models to look for the numerous transactions patterns would be more various, accurate and correspondent with the practical transaction patterns than the single model algorithm.

1. 前言

隨著時代的進步發展，企業所面臨的資料處理量也與日俱增，熱門的商業網站也紛紛開張，而現代人對於網站交易日漸頻繁，所以對企業處理資料上將是一個很大挑戰。資料探勘技術是企業主一個很大的分析利器，對於偵測網路流量是否異常、和各種交易行為的分析，有效歸納這些分析結果，都對企業經營是個莫大幫助。但隨之而來的資料大量處理，已成為資料探勘技術熱門的討論議題，因此在有效時間之內擷取所需的資訊，引起了國內外學者廣泛討論，也有了資料串流(Data Stream)名詞的產生，串流探勘技術(Data Stream Mining)發展。

電話通信上的記錄，分析通信期間的關係，甚而現今蓬勃發展的網際網路，所帶來的網路流量分析，以及電子商務行為，信用卡和電子轉帳、股票交易以及感測網路的應用等等，這些都是目前最熱門的話題。但所產生大量資料結果，稱之為資料串流，從中去獲得有效資訊，則是資料串流探勘技術討論重點。

所謂資料串流，它與傳統資料庫有何區別，在最近相關的研究報告指出[8,12,14]，大致有以下的特性：

(1) 連續不斷的高速資料流
(2) 探勘必須能夠即時反應
(3) 資料處理速度必須快速
(4) 分析結果必須隨著更新
(5) 有效使用硬體配備資源

上述的這些特性，紛紛有學者在資料串流探勘技術上，對於這些議題提出不同的解決辦法，有些學者在資料上的選擇以抽樣的方式以克服龐大的串流資料，但此種抽樣的方式，會降低所建構模組的準度，而在分析模式上，有些學者認為現今的資料比較有用，對於資料與時間關係的比重，衍伸出串流探勘的幾個模式，至於細節都會在本論文接下來的章節談到。

以往用來處理傳統資料庫，所被人熟知的 Apriori-like 的方法，需多次對資料掃瞄，以及大量候選資料集的產生；甚而之後 FP-Growth 的方法，雖然改善上述方法的缺點，無須候選資料集和僅需兩次對資料庫的掃描，但仍然無法適用在資料串流環境，在串流環境下，找出對我們有用、有趣的資訊，將是本論文討論的重點。

2. 文獻探討

2.1 資料串流探勘

資料串流(Data stream)是指一連串隨著時間快速產生連續不斷的項目序列，有別於以往的靜態資料，基本上掃描資料庫以一次為原則，效能上必須應付不間斷的資料流，能及時回應使用者的要求，至於分析出來的結果，容許在一個固定範圍誤差值間。

目前的研究對於找尋頻繁項目集有[4,5,6,9,10,15,19,20,22,23]，然後對於這些研究分類為三種不同視窗模式，分別是標界模式(landmark model)、傾斜式視窗模式(tilted-time window model)和滑動視窗模式(sliding window model)，至於演算法則所產生的結果，可分為精確演算法(Exact algorithms)和近似演算法(approximate algorithms)兩

種。

2.2 視窗模式

由於資料串流所產生的資料是一連串的，因此對於所要分析的資料會進行切割，以往的研究可分三種視窗模式，在這三種視窗模式底下進行探勘，找出所要的頻繁項。

2.2.1 標界模式

標界模式(landmark model)是指某一使用者想探勘的時間點到最近的時間點間資料，相關的研究[5,7,13,15,19,20,22,23]都是以此種模式。此種模式被詬病的是，由於時間改變迅速，會被舊有的資訊所影響，所得到關係未必適合現在。而人們比較有興趣的是新資料的關係，在電子商務交易行為如此快速，以及股市交易行為變化如此劇烈，新的資訊變化還是人們所關注的焦點。

2.2.2 傾斜式視窗模式

傾斜式視窗模式(tilted-time window model)則加入了權重的概念，其認為舊資料不該和新資料一樣重要，因此加入權重減少舊資料對於新資料的干擾，舊資料會隨著時間的久遠逐漸減少影響，其相關的研究有[6,10]。

2.2.3 滑動視窗模式

滑動視窗模式(sliding window model)，是指離現在一個固定的交易量視窗，也就是說以離現在最近的這個視窗為探勘依據，完全捨棄舊有的資料，而著重在離現在最近的這個交易視窗，探勘我們所需的結果。也最適合像串流資料這樣快速的異動、增加方式，能快速反應其變化。其相關的研究[4, 6]，本論文研究主題亦是採用此種模式。

不過此種滑動視窗模式尚可細分兩種方式：時間感知滑動視窗(time-sensitive sliding window)，是利用時間來當一個視窗的交易量，例如以每小時的交易量當一個滑動視窗，下一個小時的交易量進來就刪除前一個小時的交易；另一則為交易感知視窗(transaction- sensitive sliding window)，乃是固定交易量，例如每五筆交易進來當一個視窗，等後五筆交易進來，我們才刪除前五筆交易。

2.3 相關演算法

由於資料的變化迅速，資料串流探勘的相關演算法，依探勘結果分為精確探勘和近似探勘兩種演算法。

2.3.1 精確演算法

MOMENT 演算法[23]，採用 CET(closed enumeration tree)樹狀結構，主要探勘緊密頻繁項集(closed frequent itemsets)，採字母順序所建構而成的一棵樹，是以交易感知視窗(transaction- sensitive sliding window)方式，來當它的滑動視窗，樹的節點表達方式，以虛圓形線代表此項目(item)不頻繁，虛長方形線代表此 item 頻繁但並非緊密的(closed)，以實長方形線代表此 item 頻繁且緊密的，然後使用一個 Hash table 紀錄計數值(support)和該 item 所出現在各交易中的交易編號總和，以確認是否緊密，利用這個檢查機制，隨時調整樹結構節點的屬性，即可探勘出所有的緊密頻繁項目集。

INSTANT 演算法[10]，主要探勘最大頻繁項目序列，其利用的概念為交集和聯集，來計算每個項目集是否有過門檻值，以準確的探勘出最大頻繁項目序列。CFI-stream 演算法[11]，採用 Direct Update(DIU) tree 結構，每一層只儲存該 item 長度的緊密項目集，整棵樹都維持緊密的頻繁項，提供一個漸增式方法確認是否緊密。NewMOMENT 演算法[18]，主要來改進 MOMENT 演算法，其加入了位元的概念，每個 item 以位元表示，當該交易含有此 item 以 1 表示，該交易未含有此 item 則以 0 表示，當視窗滑動時，把前面的位元刪除，再增加下一筆時，再增加位元進來。

2.3.1 近似演算法

Lossy counting 演算法[20]文中定義了兩個門檻值，一個是最小支持度 s，另一個是最大支持度誤差 ε，N 為交易數量，其中 ε 值必須小於 s 值，儲存了大於 ε 的項目集，並以估算的方式，得到未儲存的項目集。該論文得出了幾個結果，該方法可以找出所有的常見項目集，也就是大於 sN 的所有項目集都會被找出，不會有小於 $(s-\varepsilon)N$ 被找出來，估計的誤差值不會超過 $\varepsilon \times N$。Fp-stream 結構[9]，是以 Lossy counting 方式砍除非常見頻繁項，採用傾斜式視窗模式(tilted-time window model)，探勘的區間以人們慣用的時間概念為區間，一個小時一個區間，之後一天當一個區間，再來一年當一個區間，所以我們也可以知道越久的資料，在此模式之下會越不重要。

使用滑動視窗模式(sliding window model)的演算法，以最近一個視窗的交易量，文中稱之為 CTL[6]，採用 Lossy counting 方式估算計數值，節點組成包含，項目名稱，累計計數值，以及第一次進入該結構的交易編號。滑動視窗五個步驟，第一當增加一筆交易進來，包含其子集計數值都要加一，第二更新原有已存在或未存在節點新增，第三砍掉 CTL 裡面的舊資料，第四刪除計數值低於門檻值的項目集，第五找出所有頻繁項。DSMFI 的演算法[19]，延伸前置樹(Prefix tree)的方式，所製造出來森林架構，組成有動態標頭表(Dynamic Header Table)以及候選的頻繁項目樹，每個節點會有 item 名稱、記數值、區段名稱、標頭連結，是以標界模式(landmark model)作為探勘。[16]提出兩種摘要結構採用類 Fp-tree，儲存資料項集出現時間的摘要資訊，其方法稱為平均時戳法 ATS 演算法和頻率改換點 FCP 演算法，可不必儲存所有以前的交易資料，可準確砍除不頻繁項。

2.4 關聯法則

Agrawal 等人所提的關聯法則探勘[1,2]，也就是購物籃分析，即哪些產品常常會被一起購買，對問題的基本定義如下：假設資料庫 D 是顧客的交易集合，而 TID 是每個交易 T 是交易編號，以及所有購買的項目集合(Itemset)。假設項目集合為 $I=\{i1, i2,\ldots\ldots, im\}$，交易資料庫內每一筆交易的項目集合 A 皆包含於 I（$A\subseteq I$），則關聯法則可以 $A\subseteq B$ 表示，其中集合 A、B 皆包含於 I（$A\subseteq I$、$B\subseteq I$），且 $A\cap B=\varnothing$。為了知道交易行為的關聯性，又會設最小支持度，即交易 D 中 A 與 B 的聯集所佔的百分比；以及設最小信賴度，即交易 D 中包含 A 卻也同時含有 B 的百分比，當關聯法則大於等於其使用者所設的門檻值，即 $A\Rightarrow B$ [support, confidence]，support $\geqq min\_supp$ 和 confidence$\geqq min_conf$，人們便認為此交易行為是有趣的。

2.4.1 Apriori 演算法

Apriori 演算法首先於 1993 年由 Agrawal et al.提出[1,2]，也是目前資料探勘的代表技術之一，其用的一個技術稱做逐層搜尋的疊代方法，使用 k-項目集合探勘 $k+1$-項目集合，先所找出的長度為 1，該集合我們稱為 $L1$，即是頻繁 1-項目集合，再用 $L1$ 所組合出來的候選項目集合，該集合稱為長度為 2 的候選項目，我們稱作 $C2$，再經過掃描，確定頻繁 2-項目集合，稱之為 $L2$，如此類推，找出所有的頻繁項目。

範例1：如表1交易資料庫D 內有十筆交易資料，交易項目集$I=\{A, B, C, D, E, F\}$。假設最小支持度為40%，最小信賴度為50%。如圖1首先找出1-itemset集合成為1-candidate(候選項目集)，由於1-candidate中的itemset 皆有可能成為frequent itemset，但又包含了不為frequent itemset者，故稱為候選項目集。之後產生Large1-itemsets，指此項目集支持大於等於最小支持度。如圖**1**過程一$\{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}\}$經Scan DB 之後取得其Count 保留其支持度大於等於40%者，則產生$\{\{B\}, \{C\}, \{D\}, \{E\}\}$，成為頻繁1-項目集。再由頻繁1-項目集組成2-candidate中的itemset$\{\{BC\}, \{BD\}, \{BE\}, \{CD\}, \{CE\}, \{DE\}\}$。如圖1過程二經Scan DB 之後取得其Count 保留其支持度大於等於40%者，則產$\{\{BC\}, \{CD\}\}$，成為頻繁2-項目集。再經由頻繁2-項目集組成3-candidate中的itemset$\{BCD\}$，經Scan DB 之後無支持度大於等於40%者，所以所有探勘工作到此結束。由圖1的頻繁項集合，可以計算出所有關聯法則和信賴度。例如：法則$B=>C$ 在資料庫的交易記錄中出現次數佔40%，同時只要交易紀錄中含有項目集 $\{B\}$ 意謂著項目集$\{C\}$的出現機率為60%。

表 1 交易資料庫 D

TID	Itemset
T1	ABC
T2	CDE
T3	BCF
T4	ACD
T5	BE
T6	ABCE
T7	CDF
T8	CF
T9	BE
T10	BCD

過程一：掃描資料庫

C1

Itemset	Count
A	3
B	6
C	8
D	4
E	4
F	3

L1

Itemset	Count
B	6
C	8
D	4
E	4

過程二：掃描資料庫

C2

Itemset	Count
BC	4
BD	1
BE	3
CD	4
CE	2
DE	1

L2

Itemset	Count
BC	4
CD	4

過程三：掃描資料庫

C3

Itemset	Count
BCD	1

產生關聯法則

rule	support 值	confidence 值
B⇒C	40%	66%
C⇒D	40%	50%

圖 1Apriori 執行過程

2.4.2 Fp-tree 演算法

J. Han 等提出了不須產生候選項目即可探勘頻繁項集的方法[9]，稱之 FP-tree 演算法，他所採取的方式分而治之。將所有頻繁項目集合，壓縮成一棵頻繁樣式樹來表示，建立一個項目標頭表，每個項目都有一個鏈結。因此要找到每個項目，就會比較容易。探勘頻繁項時，由標頭表最後一項，往上探勘，會產生各項的條件 Fp-tree，最後即可產生所有的頻繁項目集。

範例 2：原始交易資料庫如表 2，共有五筆交易，以最小支持度為 3，依照出現次數排序，根據出現的次數排列後其結果如表 2。依照 L1 砍除不頻繁項目後，每筆交易所排序的結果如表 **3**。假設我們找 m 的條件式，會有兩條路徑，分別是 *fcam* 和 *fcabm*，如圖 **3**，經過合併之後，可以得出 *f*:3，*c*：3，*a*：3，*m*：3。因此可以找出 *am*：3，*cam*：3，*fcam*：3，*fam*：3；*cm*：3，*fcm*：3；*fm*：3，有關於 m 的所有頻繁項，如圖 4。因此其他條件式的頻繁項也可以這樣的找出來，完成所有的探勘工作。

表 2 交易資料庫 D 和 L1 根據次數排序過程

TID	Items
100	f, a, c, d, g, i, m, p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

L1		L1
a:3		f:4
b:3		c:4
c:4	sort →	a:3
f:4		b:3
m:3		m:3
p:3		p:3

表 3 砍除不頻繁項排序後

TID	Ordered
100	f, c, a, m, p
200	f, c, a, b, m
300	f, b
400	c, b, p
500	f, c, a, m, p

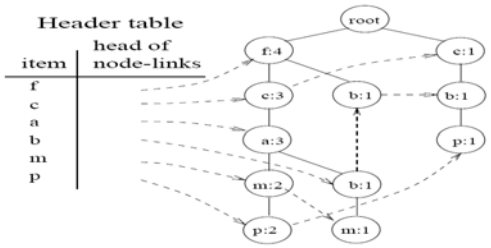


圖 2 所建的 FP-tree

資料來源：Jiawei Han, Jian Pei, and Yiwen Yin(2000:3)

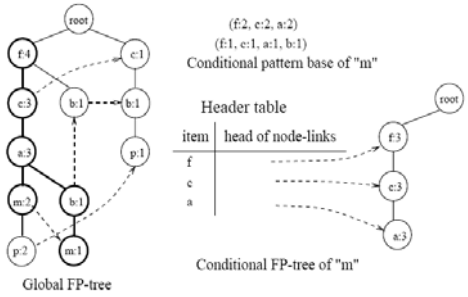


圖 3 整體 FP-tree 和 m 的條件樹

資料來源：Jiawei Han, Jian Pei, and Yiwen Yin(2000:6)

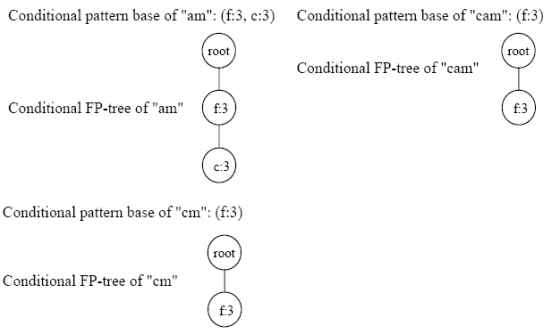


圖 4 找尋有 m 的頻繁項

資料來源：Jiawei Han, Jian Pei, and Yiwen Yin(2000:6)

3. 動態串流樹探勘演算法

3.1 介紹

本章延伸(Data stream Tree)提出一個類 Fp-tree 的架構，可以動態分割合併的樹狀結構，可適用於串流資料中找尋關聯法則。讓進入的資料形成不同的預測模型(model)，不會因為串流資料流動快速，讓部分有用資訊流失在快速的交易行為中。

3.2 DDST 演算法

Carson 等提出一個 Data stream tree 資料結構

[20]，主要兩個特徵，按照字母順序建立樹的分支，以及節點不止儲存一個計數值。而是儲存維持幾個視窗於串流資料中的支持度，例如維持三個視窗滑動，則每個節點分別儲存三個計數值欄位。

本研究維持這兩個特性，延伸提出 Dynamic Data Stream Tree，定義如下：

定義一：一個視窗交易量大小稱為 w，假設維持一個視窗三筆交易，則記為 w=3。

定義二：一棵樹狀結構維持幾個視窗，假設維持兩個，則記為 wincount=2。

定義三：定義兩個 support 值，一個為 s，一個為 ε，當大於交易量乘以 s，稱之為頻繁項。亦即該項目的出現次數大於交易量乘以 s；反之若小於交易量乘以 ε，稱之為不頻繁項。

定義四：建立一個標頭表(Header table)，以便尋訪時，找到相關項目連結。

定義五：每個節點需記錄項目名稱、視窗的計數值。

先行讀取一段時間交易視窗資料，以建立該段時間的模型樹(model tree)，然後產生該段時間的頻繁項目集。再以後來進入視窗的滑動視窗當測試集，與目前產生的頻繁項比較。若預測準，則加入原先的模型樹；若預測不準，則建立新產生的模型樹，當模型樹產生的頻繁項相似時，則進行合併。

3.2.1 建立樹狀結構

假設固定每個視窗為五筆交易，維持兩個滑動視窗，目前讀取表 4 原始交易檔，交易一到交易五，即為第一個滑動視窗。建立此樹狀結構，依照字母順序建立其分支，每個節點記錄兩個視窗的計數值，並建立其項目標頭表。

我們設定最小支持度 s 為 0.5，第二個支持度為 0.25，檢查是否有不頻繁的節點，即計數值小於交易量乘以 ε。以例子而言，即＜5×0.25=1.25 的節點需刪除，如圖 6，若有節點小於，則需移除此節點。刪除節點時，須依底下原則：

(1) 若為第一層節點，還有子節點，不移除第一層節點，將該計數值設為 0；若無任何子節點，則該節點移除，並移除標頭表的該項目。
(2) 若不為第一層節點，則直接刪除。
(3) 刪除節點後，檢查路徑是否一樣，一樣則合併，如圖 7。

探勘頻繁項目集，以標頭表項目大於交易量乘以 s，以例子而言，即大於 5×0.5=2.5 的項目，按字母順序向下搜尋。項目 b、d 符合，先產生 b 的條件式，bd=>3,2，所產生的項目集，b(3)、bd(2)；d=>2，所產生的項目集 d(2)；d=>2，所產生的項目集 d(2)。

儲存準頻繁項，即大於交易量乘以 ε 以上的項目集，因為未來可能成為日後的頻繁項集。以例子而言，大於 5×0.25=1.25，需儲存下來，找出頻繁項目集，即大於交易量乘以 s 以上的項目集。以例子而言，大於 5×0.5=2.5，即為頻繁項，如表 5。

表 4 原始交易檔資料

Tid	1	2	3	4	5
items	bd	bcd	b	ad	ad
Tid	6	7	8	9	10
items	bd	ade	abd	bd	bc
Tid	11	12	13	14	15
items	bd	ac	bc	bc	bc
Tid	16	17	18	19	20
items	bc	be	ce	ad	bc

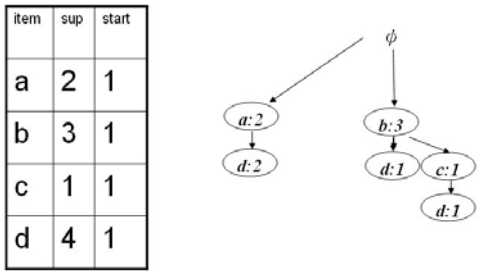


圖 5 讀入第一個視窗所建的樹

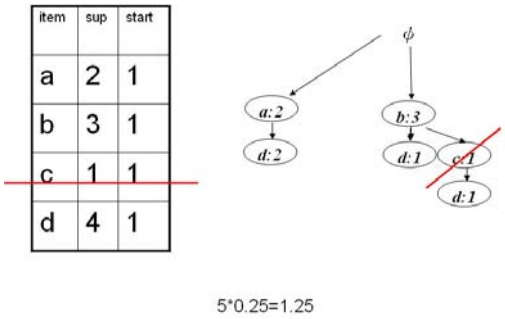


圖 6 刪除不頻繁節點

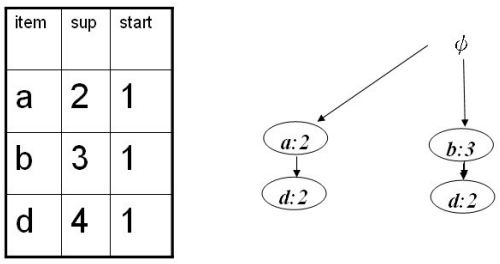


圖 7 刪除不頻繁節點後

表 5 第一個視窗儲存和頻繁項結果

item	W1	frequent
b	b(3),bd(2)	b(3)
d	d(4)	d(4)

讀取交易六到交易十，也就是第二個滑動視窗，所建立的樹狀結構如圖 8。檢查是否有不頻繁的節點，即計數值小於交易量乘以 ε。以例子而言，第二個視窗進來的節點，小於 5×0.25=1.25 的節點需刪除。但是第一個視窗進來節點，小於 10×0.25=2.5，才需刪除，若有節點小於上述的值，則需移除此節點。刪除節點時，如圖 9，須依上述所說的原則：

(1) 若為第一層節點，還有子節點，不移除第一層節點，將該計數值設為 0；若無任何子節點，則該節點移除，並移除標頭表的該項目。
(2) 若不為第一層節點，則直接刪除。
(3) 刪除節點後，檢查路徑是否一樣，一樣則合併。

刪除節點後的結果，其他節點結果，如圖 10。

探勘頻繁項目集，以標頭表項目大於交易量乘以 s，但須注意項目進來的視窗次序。以例子而言，第一個視窗進來的項目，計數值須大於 5×0.5=2.5 的項目，第二個視窗進來的項目，計數值須大於 10×0.5=5 的項目。若加入新進視窗後頻繁者，前面的視窗亦需走訪。按字母順序向下搜尋，項目 b、d 符合，先產生 b 的條件式，bd=>1,1、bd=>2,2、bd=>1 所產生的項目集，b(4)、bd(3)；d=>1;d=>1;d=>2 所產生的項目集 d(4)。

儲存準頻繁項，即大於交易量乘以 ε 以上的項目集，未來可能成為日後的頻繁項集。以例子而言，第二個視窗進來的項目集大於 5×0.25=1.25，需儲存下來，第一個視窗進來的項目集，則須大於 10×0.25=2.5。找出頻繁項目集，即大於交易量乘以 s 以上的項目集。以例子而言，第二個視窗進來的項目集大於 5×0.5=2.5，第一個視窗進來的項目集大於 10×0.5=5，即為頻繁項，即結果為表 6。

item	sup	start
a	4	1
b	7	1
c	1	2
d	8	1
e	1	2

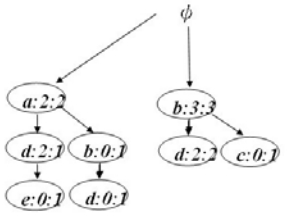


圖 8 讀取第二個視窗所建的樹

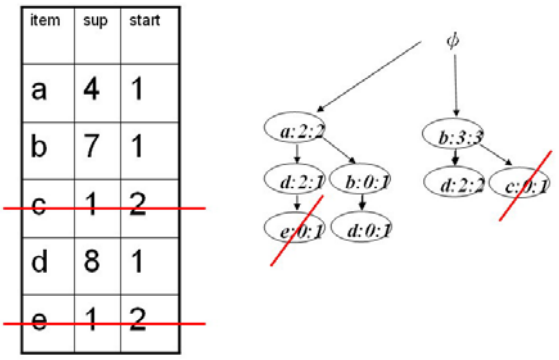


圖 9 砍除讀取第二個視窗後的不頻繁項

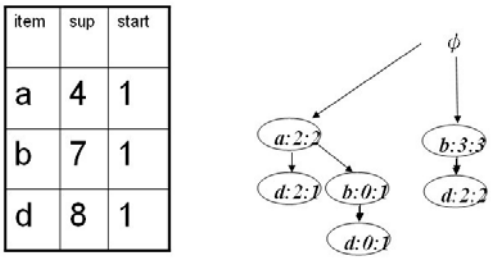


圖 10 固定兩個視窗砍除不頻繁項結果

表 6 兩個視窗的儲存和頻繁項結果

item	W1	W2	frequent
b	b(3),bd(2)	b(4),bd(3)	b(7),bd(5)
d	d(4)	d(4)	d(8)

3.2.2 新生模型樹

進入的第三個交易視窗，交易資料如 {bd,ac,bc,bc,bc}，以目前的滑動視窗裡的交易資料當測試集，把與原先樹的頻繁項和接下來要進入樹的滑動視窗裡的交易做比較，依據下列幾個步驟：

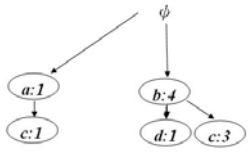
(1) 預測門檻值 δ= 與樹頻繁項一樣項目/視窗交易量，假設定義 0.8，該例子為 1/5=0.2。
(2) 與預測門檻值比較，若大於直接加入該樹中，若小於則新建一棵樹，很明顯例子比預測門檻值小，所以新建一棵樹。

建樹的方式和上述章節所說相同，所產生的第二棵模型樹，如圖 11，砍除不頻繁項之後結果如圖 12。找尋大於 5×0.5=2.5 的項目，發現 b、c 項目符合，bc=>4,3 所找出的項目集，b(4)、bc(3)；c=>1、c=>3 所找出的項目集，c(4)，但是須儲存大於 5×0.25=1.25 的準頻繁項，如表 7。

讀取第四個視窗{bc,be,ce,ad,bc}，與舊樹頻繁項比對後，0/5=0 依舊小於預測的門檻值 0.8，所以讀入新樹建成新模型(model)，如圖 13 和圖 14，而

探勘之後的結果，如表 8。

item	sup	start
a	1	3
b	4	3
c	4	3
d	1	3



5*0.25=1.25
刪除不頻繁節點

圖 11 第三視窗建成新生樹

item	sup	start
a	1	3
b	4	3
c	4	3
d	1	3

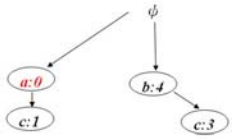
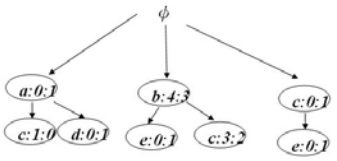


圖 12 砍除新生樹的不頻繁項

item	sup	start
a	1	4
b	7	3
c	7	3
d	1	4
e	2	4



5*0.25=1.25
刪除不頻繁節點

圖 13 第四個視窗進來新生樹的結果

item	sup	start
a	0	0
b	7	3
c	7	3
e	2	4

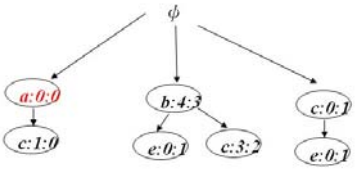


圖 14 砍除第四個視窗進來新生樹的不頻繁項

表 7 第三視窗建成新生樹的儲存以及頻繁項結果

item	W3	frequent
b	b(4),bc(3)	b(4),bc(3)
d	c(4)	c(4)

表 8 第四個視窗進入新生樹的儲存和頻繁項結果

item	W3	W4	frequent
b	b(4),bc(3)	b(3),bc(2)	b(7),bc(5)
c	c(4)	c(3)	c(7)

3.2.3 合併模型樹

隨著時間的演進，部分的模型樹(model tree)也會越來越一致，因此這些的 model，必須進行合併，至於合併的原則如下。

(1) 比對各樹的頻繁項，合併相似門檻值 μ=相同項目/總數，即任兩樹的頻繁項的交集/聯集。假設我們要將上述兩棵模型樹合併，所算出的 μ=1/5。

(2) 若合乎我們的要求，則進行合併，合併方式將下一次將進入滑動視窗作預測。假設有兩個模型樹須合併，則對這兩棵樹作預測，將預測低的 buffer 資料讀入預測高的。

(3) 原預測的原則，跟上述章節所談一致，即滑動視窗對樹的頻繁項做比對，相同項目/視窗滑動交易量。

進入的第五個視窗{bd,bd,bd,bc,bc}，原樹預測 3/5 大於新樹 2/5。將新樹合併於原樹，也就是將新樹暫存的資料讀入原樹如圖 15，即 w3{bd,ac,bc,bc,bc}、w4{bc,be,ce,ad,bc}，如圖 16，讀入 w3 即新生樹的第一個流入視窗、如圖 17，讀入 w4 即新生樹的第二個流入視窗、如圖 18，為讀入所有新生樹的 buffer 資料的樹狀結構，其探勘結果如表 9。

item	sup	start
a	4	1
b	7	1
d	8	1

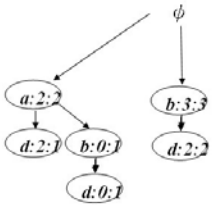


圖 15 原樹

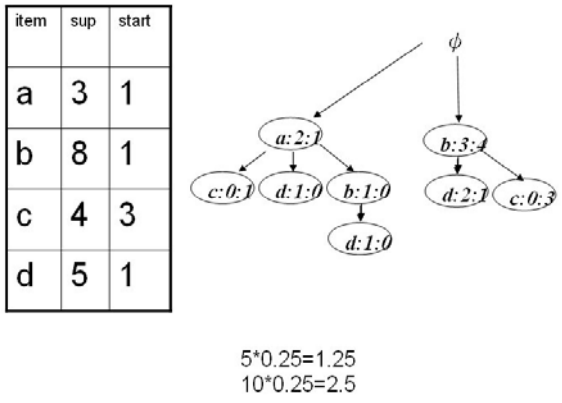


圖 16 原樹讀入新生樹的 buffer 第一筆視窗資料

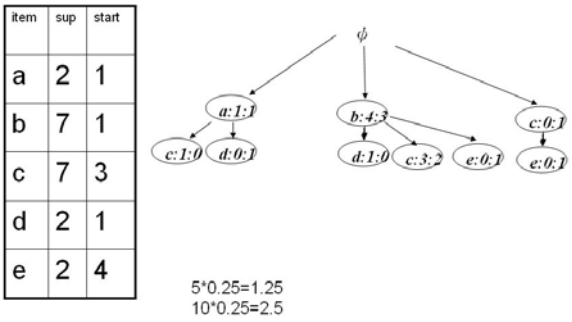


圖 17 原樹讀入新生樹的 buffer 第二筆視窗資料

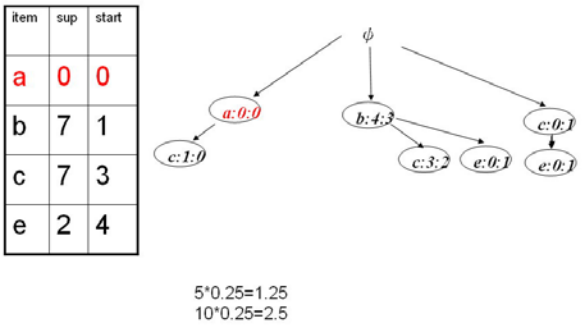


圖 18 原樹讀完新生樹 buffer 資料最後結果

表 9 合併後的儲存和頻繁項結果

item	W3	W4	frequent
b	b(4),bc(3)	b(3),bc(2)	b(7),bc(5)
c	c(4)	c(3)	c(7)

4. 實驗結果與效能分析

本章第一節描述本實驗的環境與評估方式，第二節描述本實驗所使用的資料集，第三節單一串流樹與動態串流樹在資料集上的效能表現。

4.1 實驗環境與評估方式

本實驗所使用到的電腦其中央處理器為 Intel Pentium 4 CPU 2.4GHz，記憶體容量為 1GB，所使用的作業系統是 Microsoft Windows XP Service Pack 2，而實驗中所有程式之執行檔都是以 Borland C++ 6.0 Compiler 編譯而成。為了產生實驗用的資料，採用了本實驗室所開發的人工資料產生器。關於效能評比，主要的準則為：評比單一串流樹和動態串流樹準度上，探勘出來的頻繁項數目和規則總數多寡。執行時間評估則是計算各演算法的關聯法則所花的時間。

關於比較的演算法，本論文將採用動態串流樹的單一模型與動態串流樹的循環模型比較。動態串流樹是改單一樹狀結構而來，主要以單一模型探勘串流環境下的頻繁項目集，並且與本論文所提出的動態串流樹(Dinamic Data Stream Tree)的循環模型比較。

4.2 實驗資料集

本論文所使用的實驗資料集為本實驗室所開發的 SSTGen 模擬資料集產生器，其模擬資料集產生器有數個參數分別列於表 10。

表 10 SSTGen 模擬資料集產生器一些參數意義

參數名稱	意義
$\|D\|$	交易資料庫內交易筆數
$\|L\|$	最大可能交易長度
N	交易項目的數量

在產生模擬資料集時，模擬每季可能擁有四季的商品，模擬出一年的交易資料，本論文第一個實驗設定$|L|$=10、N=26，$|D|$則選用五種數值：10k、20k、30k、40k、50k，此實驗目的是不同交易筆數各演算法所找出的頻繁項數量在相同滑動視窗下是否有差距。第二個實驗設定$|L|$=20、N=26，$|D|$=30k，最小支持度 s 分別以 0.4、0.6、0.8，第二個支持度值 ε 各以 0.2、0.3、0.4，此實驗的目的是以不同支持度下各演算法的執行時間比較是否有差距。

4.3 效能評估

第一個實驗使用了五個人造資料集，以 DDST 單一模型和動態模型方法進行實驗比較。每個滑動視窗 30 筆交易資料，固定 3 個視窗於 DDST 中，s 設為 0.5、ε 設為 0.25，單一模型和動態模型在不同交易筆數的資料庫所探勘的頻繁項數量多寡以及規則數量和執行時間之比較。動態模型參數設定：δ=0.8、μ=0.005、每間隔 50 個視窗檢查是否合併。

在圖 19 以不同資料庫筆數去比較兩個模型所找出的頻繁項的總量，由圖可以看出動態模型所找

出的頻繁項總量明顯比單一模型多。也就是動態模型可以找出更多的交易行為，一些少量容易被忽略交易行為，會因不同模型的建立，將這些樣式探勘出來。在圖 20 以不同資料庫筆數去比較兩個模型在信賴度 50%，所產生的規則總數的比較。可以看出動態模型所找出的規則總量明顯優於單一模型，即對交易行為建立不同模型時，可以更準確的對使用者的行為作預測，對企業經營可以有更好的參考依據。在圖 21 可以看出單一模型在不同資料庫筆數執行時間上優於動態模型，因為動態模型建立不同的交易行為模型，會不停的建樹和合併樹。單一模型不會有這樣的問題產生，所以會優於動態模型。

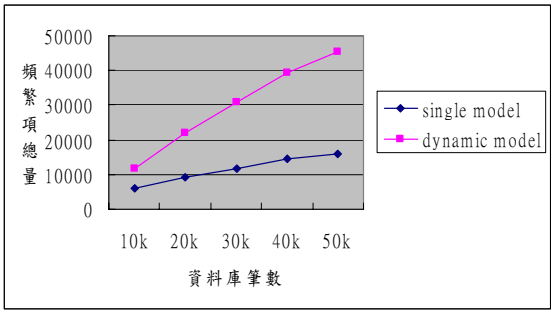


圖 19 單一模型和動態模型在不同資料庫筆數頻繁項總量比較

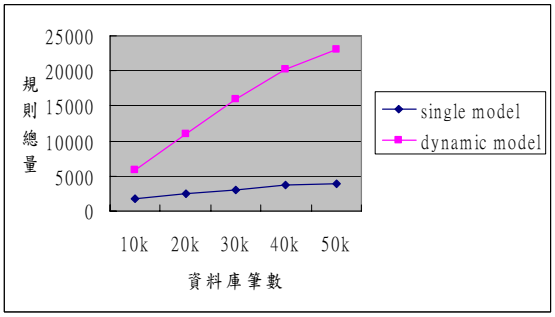


圖 20 單一模型和動態模型在不同資料庫筆數規則總量比較

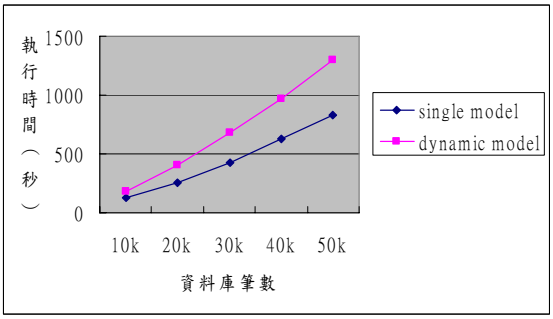


圖 21 單一模型和動態模型在不同資料庫筆數執行時間比較

第二個實驗使用了資料筆數 30k 人造資料集，以 DDST 單一模型和動態模型方法進行實驗比較。每個滑動視窗 30 筆交易資料，固定 3 個視窗於 DDST 中，分別 s 設為 0.8、ε 設為 0.4；s 設為 0.6、ε 設為 0.3；s 設為 0.4、ε 設為 0.2，單一模型和動態模型在相同交易筆數不同支持度下對資料庫所探勘的頻繁項數量多寡以及規則數量和執行時間之比較。動態模型參數設定：δ=0.8、μ=0.005、每間隔 50 個視窗檢查是否合併。

在圖 22 以不同支持度下去比較兩個模型所找出的頻繁項的總量，由圖可以看出動態模型所找出的頻繁項總量明顯比單一模型多。跟本研究第一個實驗討論一樣，也就是動態模型可以找出更多的交易行為，一些少量容易被忽略交易行為，會因不同模型的建立，將這些樣式探勘出來。當支持度越大時，所探勘的頻繁項目會跟著減少，但動態模型依舊優於單一模型。在圖 23 以不同大小的支持度去比較兩個模型在信賴度 50%，所產生的規則總數的比較。可以看出動態模型所找出的規則總量明顯優於單一模型，亦即對交易行為建立不同模型時，可以更準確的對使用者的行為作預測。當支持度越大時，過門檻的行為樣式規則會越來越少，如圖 23 所示，但動態模型依然優於單一模型。在圖 21 可以看出單一模型在不同大小支持度下執行時間上優於動態模型，因為動態模型建立不同的交易行為模型，會不停的建樹和合併樹。單一模型不會有這樣的問題產生，尤其在支持越高時，所過的門檻樣式越少，單一模式執行時間會明顯比動態模型少，雖然動態模型執行時間高，但尚在接受範圍內。

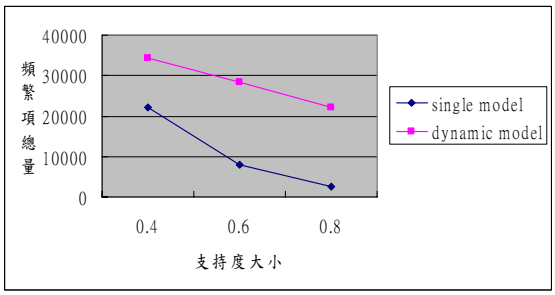


圖 22 單一模型和動態模型在資料庫筆數 30k 不同支持度下頻繁項總量比較

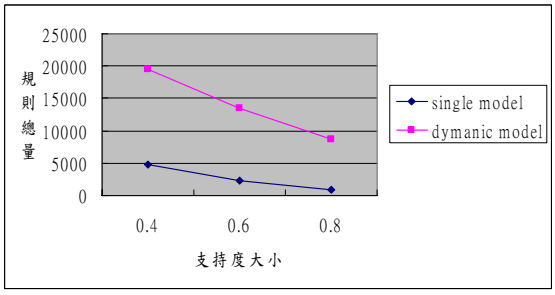


圖 23 單一模型和動態模型在資料庫筆數 30k 不同支持度下規則總量比較

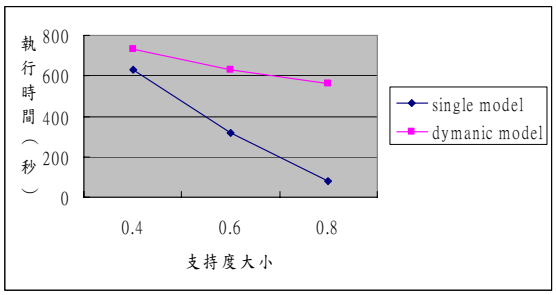


圖 24 單一模型和動態模型在資料庫筆數 30k 不同支持度下執行時間比較

5. 結論與未來研究方向

在串流資料環境下，探勘的方式，都是以單一模型在尋找頻繁項的結果。此種方式雖可找出頻繁項結果，但在消費行為模式上，一些少量存在的消費行為則會在串流環境被忽略。所以本研究提出 DDST 演算法，找出所有的消費模式，變成一個動態模型(model)，所找出的頻繁消費行為，會比單一模型所找的消費行為多，而且對消費行為也更為準確。雖然動態串流樹所花的執行時間會比單一串流樹多，但是所找的結果會更貼近實際在生活上發生的行為。實驗結果顯示，DDST 方法能找出較多的頻繁項，以及更準確預測消費行為模式，因此本方法確能提供串流環境下探勘頻繁循環消費模式預測，一個有效方式。

在探勘循環消費行為模式，在未來的研究方向上，大致可由底下幾點討論：

由於本論文所模擬的資料集，有一定的循環方式，但在實際的資料集上，可能會產生過多的動態樹，記憶體可能不足，未來可以考慮解決過多動態樹的方式。本論文所提的方式，花費執行時間過高，未來可以使用更佳的方式來減少過多的執行時間，產生更好的效能。本論文所使用方式為滑動視窗模式，可考慮將滑動視窗與傾斜式滑動視窗結合，提出另一個新的探勘模式。

參考文獻

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," ACM SIGMOD Record, vol. 22, pp. 207-216, 1993.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," Proc. 20th Int. Conf. Very Large Data Bases, VLDB, vol. 1215, pp. 487-499, 1994.

[3] J. H. Chang and W. S. Lee, "Finding recent frequent itemsets adaptively over online data streams," Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 487-492, 2003.

[4] J. H. Chang and W. S. Lee, "A Sliding Window Method for Finding Recently Frequent Itemsets over Online Data Streams," Journal of Information Science and Engineering, vol. 20, pp. 753-762, 2004.

[5] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," Theoretical Computer Science, vol. 312, pp. 3-15, 2004.

[6] Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz, "Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window," Proc. of, vol. 1401, 2004.

[7] G. Cormode and S. Muthukrishnan, "What's hot and what's not: tracking most frequent items dynamically," ACM Transactions on Database Systems (TODS), vol. 30, pp. 249-278, 2005.

[8] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," ACM SIGMOD Record, vol. 34, pp. 18-26, 2005.

[9] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu, "Mining Frequent Patterns in Data Streams at Multiple Time Granularities," Next Generation Data Mining, vol. 212, 2003.

[10] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach," Data Mining and Knowledge Discovery, vol. 8, pp. 53-87, 2004.

[11] N. Jiang, "CFI-Stream: mining closed frequent itemsets in data streams," Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 592-597, 2006.

[12] N. Jiang, "Research issues in data stream association rule mining," ACM SIGMOD Record, vol. 35, pp. 14-19, 2006.

[13] C. Jin, W. Qian, C. Sha, J. X. Yu, and A. Zhou, "Dynamically maintaining frequent items over a data stream," Proceedings of the twelfth international conference on Information and knowledge management, pp. 287-294, 2003.

[14] W. Jinlong, X. Congfu, C. Weidong, and P. Yunhe, "Survey of the study on frequent pattern mining in data streams," Systems, Man and Cybernetics, 2004 IEEE International Conference on, vol. 6, 2004.

[15] R. M. Karp, S. Shenker, and C. H. Papadimitriou, "A simple algorithm for finding frequent elements in streams and bags," ACM Transactions on Database Systems (TODS), vol. 28, pp. 51-55, 2003.

[16] J. L. Koh and S. N. Shin, "An Approximate Approach for Mining Recently Frequent Itemsets from Data Streams," presented at DaWaK, 2006.

[17] C. K. S. Leung and Q. I. Khan, "DSTree: A Tree Structure for the Mining of Frequent Sets from Data Streams," Proceedings of the Sixth International Conference on Data Mining, pp. 928-932, 2006.

[18] H. F. Li, C. C. Ho, F. F. Kuo, and S. Y. Lee, "A New Algorithm for Maintaining Closed Frequent Itemsets in Data Streams by Incremental Updates,"

Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on, pp. 672-676, 2006.

[19] H. F. Li, S. Y. Lee, and M. K. Shan, "An efficient algorithm for mining frequent itemsets over the entire history of data streams," Proc. of the 1st Intl. Workshop on Knowledge Discovery in Data Streams, 2004.

[20] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," Proc. VLDB, vol. 2, pp. 346–357, 2002.

[21] G. Mao, X. Wu, C. Liu, X. Zhu, G. Chen, Y. Sun, and X. Liu, "Online Mining of Maximal Frequent Itemsequences from Data Streams," 2005.

[22] L. Yang and M. Sanver, "Mining Short Association Rules with One Database Scan," Int'l, 2004.

[23] J. X. Yu, Z. Chong, H. Lu, and A. Zhou, "False Positive or False Negative: Mining Frequent Itemsets from High Speed Transactional Data Streams," Proceedings of the 30th ACM VLDB International Conference on Very Large Data Bases, pp. 204–215, 2004.