應用自我建構類神經網路於線上分析處理索引之研究

Apply the SOFM to the On Line Analysis Processing (OLAP)

孫光天
國立台南師範學院資訊教育研究所
ktsun@ipx.ntntc.edu.tw

張鈺玟
國立台南師範學院 資訊教育研究所
ywchang@ipx.ntntc.edu.tw

摘要

線上分析處理 (On Line Analysis Processing, OLAP) 的索引為資料倉儲研究之重點，它比一般索引架構更著重於範圍查詢效率，尤其是對不同範圍資料的整合性查詢，更是大家熱烈討論與研究之主題。目前一般常用技術，例如：R*a tree 和 X-tree 則是適用於 OLAP 高維度資料的索引架構，並已有很好的查詢效率。本研究目的則是利用類神經網路中自我組織映射網路 (SOFM) 的區域密集性 (locality) 的特性，將 OLAP 的大量高維度資料加以有效率的群聚後，再運用 X-tree 技術，使範圍查詢時能減少樹的搜尋層數，並提高資料的儲存空間使用率，以進一步提升 OLAP 的範圍查詢效率，經多種不同維度之模擬實驗後顯示，本研究技術可大量減少查詢次數，驗證本文所提方法有極佳的效能。

關鍵字：線上分析處理、自我組織映射網路、R*a 樹、X-樹。

一、簡介

資料倉儲 (Data Warehouse) 是針對決策支援系統的需求所發展出來的資料庫觀念，其資料常經由線上分析處理 (On Line Analysis Processing, OLAP) 提供資訊，做為管理者決策時的參考[2]。OLAP 查詢主要是針對大數量的資料，提供不同的範圍查詢 (region query)。大部分的 OLAP 查詢可以被視為對一維或多維資料方格 (data cube) 的部分匹配查詢 (partial match queries)，其需要有效率的獲取對應範圍的資料[14]。目前多維空間資料的索引方式，比較常見的有 R tree [6]、R+ tree [15]、Packed R-tree [13]、Grid files [10]、Bitmap index[11]、R*a tree [7]、X tree [5]、K-D-B tree[12] 等。其中 R tree 是一項針對多維空間資料而設計的方法，適用於維度高、區域密度大且更新頻繁的情況，能對稱性的處理所有維度，並可應用在索引 OLAP 資料。由於 R tree 對於區域的分割為非規則性的切割，且每一個節點所涵蓋的區域範圍，必涵蓋所有子節點所包含的區域，而任二個節點所代表的區域可能產生重疊 (overlap)，造成在點的查詢上，R tree 可能產生多條的查詢路徑，使查詢的效率降低，因此在 R tree 的相關索引法中，影響查詢效率的因素有二，一是重疊的區域範圍，二是節點涵蓋的矩形範圍，如何使重疊的區域範圍與節點涵蓋的矩形範圍達到最小，便成為改善查詢效率的主要關鍵[1,3,4]。

由於 R*a tree 與 X tree 是適用於 OLAP 高維度資料的索引架構，X tree 是改進 R tree 中間節點重疊情形的一種樹狀索引架構，能提高 R tree 的範圍查詢效能，然而就均勻分配的資料而言，X tree 資料節點的空間使用率僅 66% [5]，就儲存空間的使用度而言，尚有改進的空間。另一方面，類神經網路技術 [9,16] 中，屬於非監督式 (unsupervised) 類神經學習網路模式的自我建構網路 (Self Organizing Feature Mapping, SOFM)[8] 具有能將大量資料藉由類神經網路訓練模式加以組織，將特徵相近的資料聚類在一起的特性，因此本研究的目的在於利用 SOFM 將 OLAP 的大量高維度資料加以有效率的聚類，運用於 X-tree 葉節點層的建構上，並於中間節點增加一個存放特徵值的欄位，將其所屬的子節點存放的鍵值加以總合，使範圍查詢時能減少樹的搜尋層數，並依據 SOFM 能使資料具有區域密集 (locality) 的特性，以提升 OLAP 範圍查詢的節點存取效率。

二、自我組織映射網路於線上分析處理索引之技術

本研究採用 Xa tree [1] 的資料結構，即採用 X tree 的分裂方法，並於中間節點增加一存放特徵值的欄位，做為高維度資料索引時中間節點的主要架構。本研究首先利用 SOFM 將 OLAP 的大量高維度資料加以有效率的組

織，藉由 SOFM 將資料值相近的節點聚類，使資料具有區域密集 (locality) 的特性，並考慮資料的儲存空間使用率，運用於葉節點層的建立上。之後，由經過 SOFM 組織後的葉節點層開始，採用 X tree 中間節點的分裂方法，往上建構整個樹狀索引結構，以期提升葉節點的資料佔滿率，減少葉節點的數目，使資料集中於較少數的節點，藉以改進現行 X tree 在葉節點僅 66% 的儲存空間使用率之問題。

Kohonen 所提出的 SOFM 演算法，可將任意維度的輸入資料分佈狀況映射到一個二維的輸出層上。由於 SOFM 訓練資料後的結果，輸出單元的分佈將會反應輸入資料的分佈，因此當多維空間的資料映射到一個一維的輸出層時，在多維空間中距離很近的資料，將對應到同一輸出節點；且歸類到鄰近輸出節點的資料，應較歸到其它距離較遠的輸出節點的資料，在多維樣本空間中的距離來的近，為達到這個目的，我們在輸出單元 (output node) 呈現上，採用一維環狀陣列 (ring) 的方式。在 SOFM 演算法中，更新輸入單元與各輸出單元間 weight 值的計算，則加上一修正量，此 weight 修正量的計算與輸出單元和 winner 間的距離有關，即輸出層使用環狀陣列的方式，將影響到 weight 值在 SOFM 訓練 (training) 過程中的修正。由於環狀陣列中任二個節點，彼此間具有二種距離算法的特性，因此我們取其中較小者做為二個節點間的距離。

針對 SOFM 演算法須預先限制輸出節點個數的限制，我們採用下列公式計算輸出節點的個數，亦即未來建樹時的葉節點數目：

$$M = (data\#/data\_capacity * full\_rate) + 1 \quad (1)$$

其中，*data#* 為資料筆數，*datanode_capacity* 為葉節點的分支容量，*full_rate* 指葉節點的平均佔滿率。此公式的意義為將資料總數除以每個葉節點預計佔滿的分支數再加一，用以衡量 SOFM 歸類的輸出節點數目，做為未來建樹時的葉節點數目。歸類到同一輸出節點的資料，日後將分配到同一葉節點的分支。在經過 SOFM 對建樹資料進行 locality 特性的組織後，方才以資料所歸屬的輸出節點在陣列中的索引值，由小而大排序，而歸屬到同一輸出節點的資料，則按照資料與其歸屬的輸出節點 weight 間誤差平方值的差距，由小而大排序，以此順序做為插入資料到葉節點的依據。

由於在 SOFM 的演算法中，僅限制了輸出節點的數目，對分配到輸出節點的資料數目並無限制，因此可能造成歸類到同一輸出節點的資料數目大於葉節點的最大分支容量，亦即發生節點滿溢 (overflow) 的問題。有關插入資料到葉節點與 overflow 處理的方法如圖 1 所示，圖 2 為整個 SOFM 於 OLAP 之處理程序。

Step 1. 計算葉節點的資料數目。

$$M = (data\#/data\_capacity * full\_rate) + 1$$

其中，*M* 為葉節點數目，*data#* 為資料筆數，*datanode_capacity* 為葉節點的分支容量，*full_rate* 指葉節點的平均佔滿率。

Step 2. 呼叫 SOFM 函數，取得一按照輸出節點在陣列中的 index、與 winner 的 weight 誤差平方值由小到大排序的 data 陣列。

Step 3. 按 Step2 傳回的資料順序，在不造成 overflow 的情況下，依序插入葉節點。

Step 4. 將經過 Step3 後仍未分配的資料，依序分配到與 winner 距離為 1、2、3、…..等葉節點中，直到發現一個節點有空位，即插入資料。

圖 1、將 SOFM 訓練資料插入葉節點演算法

為為驗證本方法的可行性，本研究以 C++ 語言來實作，作業系統為 Windows98，記憶體大小為 128MB RAM。我們將以二維、三維、四維、五維、六維、八維、十維、十二維等不同維度，以電腦亂數產生均勻分佈 (uniform distribution) 的亂數資料十萬筆，其中每個維度的資料值於正規化後皆在 [0, 1]，以進行樹狀索引結構的建構和效能評估。本研究將藉由實作樹狀結構的程式，分析本研究的方法(以下簡稱 SOFM X-tree)、R* tree、X tree、R*a tree 和 Xa tree 等四種不同的樹狀索引結構，以便研究其在大量資料範圍查詢上的效果。我們以範圍查詢的效率作為效能分析的項目。在範圍查詢的效能表現上，我們將在上述八種不同維度的情形下，進行範圍大至整個資料庫，小至資料庫空間的十分之一，共十種不同資料庫範圍大小的範圍查詢各十次。實驗結果將記錄節點存取次數的數據結果，節點存取次數的數據係在不同資料庫範圍大小的範圍查詢下，進行十次隨機查詢後的平均值。

三、效能評估

本研究採用建樹狀況及進行範圍查詢時的節點存取次數，作為效能評估的項目。在建樹狀況的呈現上，我們記錄了由葉節點層到樹根為止的樹高層數、內部節點數、葉節點數、內部節點的最大分支容量及葉節點的最大分支容量。

Step 1. 初始化鍵值。

初始化從 N 個輸入單元到 M 個輸出單元的在樣本空間中的位置，亦即 weight 值，並定義鄰近區域 (neighborhood) 的起始半徑值 R，N 為建樹資料的維度，M 為葉節點的數目。

Step 2. 輸入資料。

輸入輸入單元 $X_i(t)$，亦即輸入在單位時間 t 時的第 i 筆資料。

Step 3. 計算輸入的資料與所有輸出節點 weight 的誤差平方值。

計算介於輸入資料與每個輸出單元 j 的 weight 誤差平方值 d_j。

$$d_j = \sum_{i=0}^{N-1}\left(X_i(t) - W_{ij}(t)\right)^2$$

其中 $W_{ij}(t)$ 是從輸入資料 i 到輸出單元 j 於單位時間 t 的 weight 值。

Step 4. 從 Step3 中，找出與輸入資料 $X_i(t)$ 具最小 weight 誤差平方值的輸出單元，又稱為優勝單元 (winner)，亦即選擇輸出單元 j^*，此節點具有與 $X_i(t)$ 最小的 weight 誤差平方值 d_{j^*}。

$$d_{j^*} = Min\{d_j, \forall j \in Output\_Layer\}$$

Step 5. 更新輸入資料 $X_i(t)$ 到輸出單元 j^*及其鄰近區域的鍵值。

從輸入單元到輸出單元 j^*及鄰近區域的 weight 值都被更新。

$$W_{ij}(t+1) = W_{ij}(t) + eta * \left(X_i(t) - W_{ij}(t)\right) * e^{-(r\_from\_win / R)}$$
$$\forall j \in NBj^*(t), 0 \le i \le N-1$$

其中 eta 是單位時間 t 內的學習速率，r_from_win 為輸出單元 j^*與其它輸出單元間的距離，R 為鄰近半徑 $(0 < eta < 1)$ 。

Step 6. 更新鄰近半徑並檢查是否收斂。

$$R(t+1) = R(t) * R\_rate$$

其中 R_rate 為鄰近半徑縮小因子 $(R\_rate < 1)$ 。檢查半徑是否已達最小值，否則將鄰近區域縮小，並回到 Step2，進一步修改 weight 值，直到鄰近區域的半徑縮到最小值($R\_min \cong 0$)。

圖 2、利用 SOFM 訓練建 Xa 樹資料之演算法

（一）輸入資料建樹狀況

表 1~表 8 為不同維度且資料為均勻（任意）分佈時，建樹的相關資料。

表 1、二維十萬筆資料建樹狀況

樹別	樹高（層）	內部節點數	葉節點數	內部節點量	葉節點量
SOFM X-tree（葉節點平均佔滿率 0.85）	3	17	695	72	170
Xa tree	3	18	887	72	170
Ra tree	3	18	887	72	170
X tree	3	18	887	72	170
R tree	3	18	887	72	170

表 2、三維十萬筆資料建樹狀況

樹別	樹高（層）	內部節點數	葉節點數	內部節點量	葉節點量
SOFM X-tree（葉節點平均佔滿率 0.8）	3	29	991	56	127
Xa tree	3	33	1193	56	127
Ra tree	3	33	1193	56	127
X tree	3	33	1193	56	127
R tree	3	33	1193	56	127

表 3、四維十萬筆資料建樹狀況

樹別	樹高（層）	內部節點數	葉節點數	內部節點量	葉節點量
$d_{j^*} = Min\{d_j, \forall j \in Output\_Layer\}$ SOFM X-tree（葉節點平均佔滿率 0.9）	3	28	1099	46	102
Xa tree	3	5	1499	46	102
Ra tree	4	51	1475	46	102
X tree	3	5	1499	46	102
R tree	4	51	1475	46	102

表 4、五維十萬筆資料建樹狀況

樹別	樹高（層）	內部節點數	葉節點數	內部節點量	葉節點量
SOFM X-tree（葉節點平均佔滿率 0.85）	3	7	1316	39	85
Xa tree	3	14	1789	39	85
Ra tree	4	74	1817	39	85
X tree	3	14	1789	39	85
R tree	4	74	1817	39	85

表 5、六維十萬筆資料建樹狀況

樹別	樹高（層）	內部節點數	葉節點數	內部節點量	葉節點量
SOFM X-tree（葉節點平均佔滿率 1）	2	1	1389	33	72
Xa tree	4	87	2121	33	72
Ra tree	4	107	2135	33	72
X tree	4	87	2121	33	72
R tree	4	107	2135	33	72

表 6、八維十萬筆資料建樹狀況

樹別	樹高（層）	內部節點數	葉節點數	內部節點量	葉節點量
SOFM X-tree（葉節點平均佔滿率 0.85）	2	1	1786	26	56
Xa tree	4	107	2737	26	56
Ra tree	4	168	2734	26	56
X tree	4	107	2737	26	56
R tree	4	168	2734	26	56

表 7、十維十萬筆資料建樹狀況

樹別	樹高（層）	內部節點數	葉節點數	內部節點量	葉節點量
SOFM X-tree（葉節點平均佔滿率 1）	3	4	2174	22	46
Xa tree	4	185	3332	22	46
Ra tree	4	258	3354	22	46
X tree	4	185	3332	22	46
R tree	4	258	3354	22	46

表 8、十二維十萬筆資料建樹狀況

樹別	樹高（層）	內部節點數	葉節點數	內部節點量	葉節點量
SOFM X-tree（葉節點平均佔滿率 1）	3	8	2565	18	39
Xa tree	5	314	3993	18	39
Ra tree	5	397	3975	18	39
X tree	5	314	3993	18	39
R tree	5	397	3975	18	39

（二）範圍查詢效能評估

本節記錄了二維到十二維中，八種不同維度下十萬筆均勻（任意）分佈的亂數資料，在範圍查詢時的節點存取次數。此外，我們並計算以另四種樹狀結構中，整體表現較佳的 Xa tree 的節點存取次數為分母，以（Xa tree 減去 SOFM X-tree）的節點存取次數為分子的效能改進因子（improve ratio），以瞭解 SOFM X-tree 改進 Xa tree 節點存取次數的百分比。效能改進因子的計算公式如下：

$$improve\_ratio = \frac{Xa\_tree\_access\# - SOFMx\_tree\_access\#}{Xa\_tree\_access\#} \quad (2)$$

其中，*SOFMx_tree_access#* 係指 SOFM X-tree 範圍查詢的節點存取次數，*Xa_tree_access#* 為 Xa tree 範圍查詢的節點存取次數。圖 3 為二維十萬筆時，資料訓練次數與能量函數之關係。

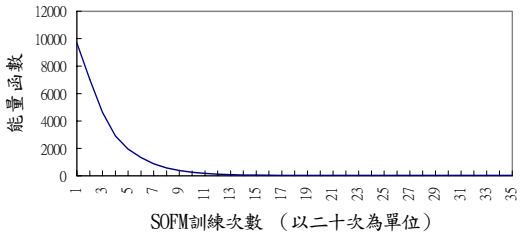

圖 3、SOFM X-tree 葉節點平均佔滿率為 0.85，在二維十萬筆均勻分佈亂數資料時，訓練次數及能量函數之關係

表 9～ 表 16 與圖 4～圖 11 為不同維度時，不同搜尋範圍與存取次數之關係。

表 9、五種樹狀結構在二維十萬筆均勻分佈亂數資料，不同範圍查詢與節點存取次數比較

data search range	R*	X	R*a	Xa	SOFM X-tree 葉節點平均佔滿率 0.85	improve ratio (%)
1	904	904	1	1	1.0	0
0.9	867	867	126	126	120.7	+ 4.21
0.8	774	774	120	120	113.0	+ 5.83
0.7	684	684	114	114	107.2	+ 5.96
0.6	593	593	107	107	100.6	+ 5.98
0.5	497	497	98	98	92.7	+ 5.41
0.4	402	402	89	89	83.9	+ 5.73
0.3	306	306	76	76	72.0	+ 5.26
0.2	211	211	62	62	57.4	+ 7.42
0.1	114	114	45	45	40.5	+ 10.00

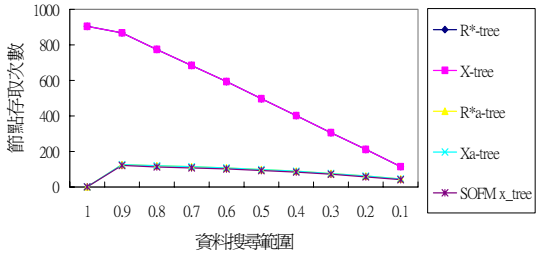

圖 4、五種樹狀結構二維十萬筆均勻分佈亂數資料範圍查詢節點存取次數比較

表 10、五種樹狀結構在三維十萬筆均勻分佈亂數資料，不同範圍查詢與節點存取次數比較

data search range	R*	X	R*a	Xa	SOFM X-tree 葉節點平均佔滿率 0.8	improve ratio (%)
1	1225	1225	1	1	1.0	0
0.9	1224	1224	610	610	468.6	+ 23.18
0.8	1202	1202	614	614	498.2	+ 18.86
0.7	1131	1131	592	592	515.8	+ 12.87
0.6	1010	1010	536	536	466.6	+ 12.95
0.5	827	827	470	470	423.4	+ 9.91
0.4	708	708	414	414	373.4	+ 9.81
0.3	560	560	349	349	303.9	+ 12.92
0.2	397	397	272	272	235.8	+ 13.31
0.1	230	230	176	176	153.3	+ 12.90

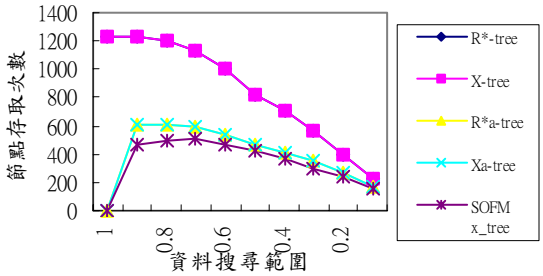

圖 5、五種樹狀結構三維十萬筆均勻分佈亂數資料範圍查詢節點存取次數比較

表 11、五種樹狀結構在四維十萬筆均勻分佈亂數資料，不同範圍查詢與節點存取次數比較

data search range	R*	X	R*a	Xa	SOFM X-tree 葉節點平均佔滿率 0.9	improve ratio (%)
1	1525	1503	1	1	1.0	0
0.9	1525	1503	1193	1140	841.9	+ 26.15
0.8	1524	1503	1234	1180	866.5	+ 26.57
0.7	1491	1502	1248	1192	881.8	+ 26.02
0.6	1382	1491	1246	1190	901.3	+ 24.26
0.5	1417	1382	1185	1119	924.0	+ 17.43
0.4	1225	1168	1049	970	871.2	+ 10.19
0.3	995	923	874	786	710.1	+ 9.65
0.2	745	678	594	594	548.9	+ 7.59
0.1	455	407	433	361	346.6	+ 3.99

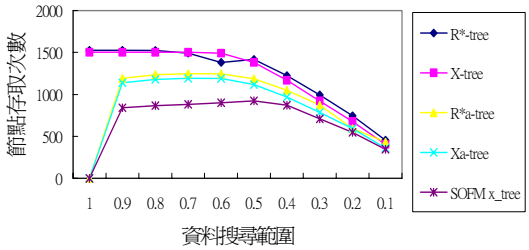

圖 6、五種樹狀結構四維十萬筆均勻分佈亂數資料範圍查詢節點存取次數比較

表 12、五種樹狀結構在五維十萬筆均勻分佈亂數資料，不同範圍查詢與節點存取次數比較

data search range	R*	X	R*a	Xa	SOFM X-tree 葉節點平均佔滿率 0.85	improve ratio (%)
1	1890	1802	1	1	1.0	0
0.9	1890	1802	1717	1602	1169.9	+ 26.97
0.8	1890	1802	1797	1679	1210.7	+ 27.89
0.7	1890	1802	1818	1702	1227.8	+ 27.86
0.6	1889	1801	1828	1711	1240.3	+ 27.51
0.5	1881	1783	1825	1702	1249.4	+ 26.59
0.4	1803	1680	1775	1641	1262.1	+ 23.09
0.3	1617	1448	1583	1442	1261.5	+ 12.48
0.2	1340	1202	1319	1173	1107.2	+ 5.61
0.1	903	757	895	744	737.9	+ 0.82

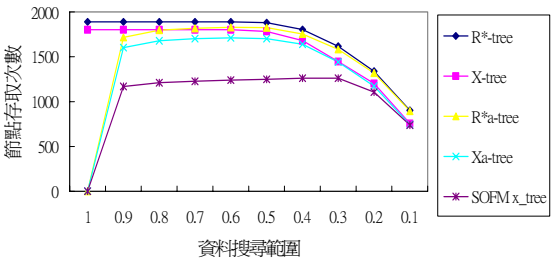

圖 7、五種樹狀結構五維十萬筆均勻分佈亂數資料範圍查詢節點存取次數比較

表 13、五種樹狀結構在六維十萬筆均勻分佈亂數資料，不同範圍查詢與節點存取次數比較

data search range	R*	X	R*a	Xa	SOFM X-tree 葉節點平均佔滿率 0.85	improve ratio (%)
1	2241	2207	1	1	1.0	0
0.9	2241	2207	2112	2074	1318.1	+ 36.45
0.8	2241	2207	2208	2171	1359.4	+ 37.38
0.7	2241	2207	2227	2191	1369.6	+ 37.49
0.6	2241	2207	2234	2198	1375.6	+ 37.42
0.5	2239	2205	2235	2199	1379.5	+ 37.27
0.4	2216	2184	2213	2213	1382.6	+37.52
0.3	2112	2080	2110	2077	1384.8	+ 33.33
0.2	1911	1890	1910	1800	1385.5	+ 23.03
0.1	1570	1547	1570	1545	1326.0	+ 14.17

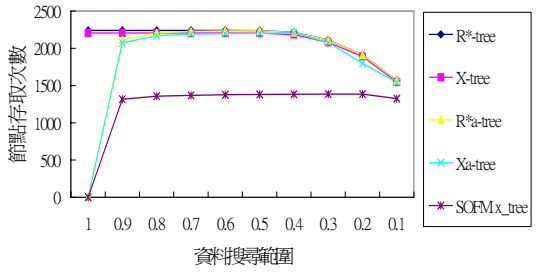

圖 8、五種樹狀結構六維十萬筆均勻分佈亂數資料範圍查詢節點存取次數比較

表 14、五種樹狀結構在八維十萬筆均勻分佈亂數資料，不同範圍查詢與節點存取次數比較

data search range	R*	X	R*a	Xa	SOFM X-tree 葉節點平均佔滿率 0.85	improve ratio (%)
1	2901	2869	1	1	1.0	0
0.9	2901	2869	2732	2864	1735.9	+ 35.32
0.8	2901	2869	2876	2853	1777.2	+ 37.71
0.7	2901	2869	2877	2866	1782.4	+ 37.81
0.6	2901	2869	2900	2868	1784.7	+ 37.77
0.5	2901	2869	2900	2868	1785.9	+ 37.73
0.4	2900	2869	2900	2868	1785.9	+ 37.73
0.3	2898	2861	2868	2868	1785.9	+ 37.73
0.2	2882	2861	2860	2831	1786.0	+ 36.91
0.1	2812	2817	2812	2817	1786.0	+ 36.60

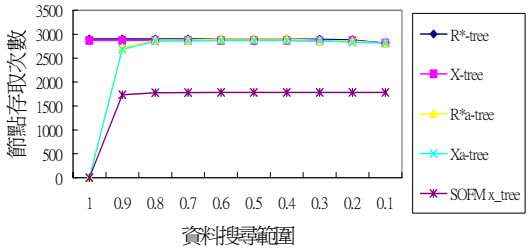

圖 9、五種樹狀結構八維十萬筆均勻分佈亂數資料範圍查詢節點存取次數比較

表 15、五種樹狀結構在十維十萬筆均勻分佈亂數資料，不同範圍查詢與節點存取次數比較

data search range	R*	X	R*a	Xa	SOFM X-tree 葉節點平均佔滿率 0.85	improve ratio (%)
1	3611	3516	1	1	1.0	0
0.9	3611	3516	3212	3254	2115.2	+ 35.00
0.8	3611	3516	3562	3487	2168.2	+ 37.82
0.7	3611	3516	3604	3511	2172.9	+ 38.11
0.6	3611	3516	3610	3515	2173.4	+ 38.17
0.5	3611	3516	3610	3515	2173.9	+ 38.15
0.4	3611	3516	3610	3516	2174.0	+ 38.17
0.3	3610	3515	3610	3515	2174.0	+ 38.17
0.2	3607	3514	3607	3514	2174.0	+ 38.13
0.1	3590	3504	3590	3504	2174.0	+ 37.96

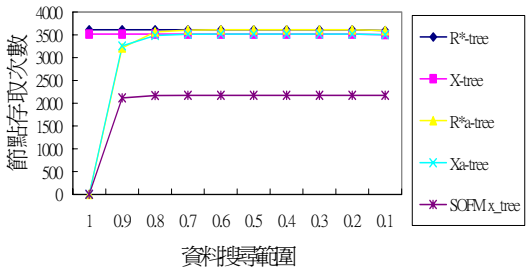

圖 10、五種樹狀結構十維十萬筆均勻分佈亂數資料範圍查詢節點存取次數比較

表 16、五種樹狀結構在十二維十萬筆均勻分佈亂數資料，不同範圍查詢與節點存取次數比較

data search range	R*	X	R*a	Xa	SOFM X-tree 葉節點平均佔滿率 0.85	improve ratio (%)
1	4371	4306	1	1	1.0	0
0.9	4371	4306	3860	3922	2493.8	+ 36.42
0.8	4371	4306	4219	4245	2562.6	+ 39.63
0.7	4371	4306	4358	4296	2564.7	+ 40.30
0.6	4371	4306	4369	4304	2565.0	+ 40.40
0.5	4371	4306	4370	4305	2565.0	+ 40.42
0.4	4371	4306	4370	4305	2565.0	+ 40.42
0.3	4371	4305	4370	4305	2565.0	+ 40.42
0.2	4370	4305	4370	4305	2565.0	+ 40.42
0.1	4366	4303	4366	4303	2565.0	+ 40.39

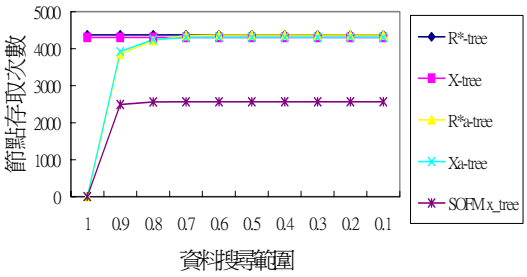

圖 11、五種樹狀結構十二維十萬筆均勻分佈亂數資料範圍查詢節點存取次數比較

四、討論與結論

歸納本研究實驗結果，我們發現運用 SOFM 改良 Xa tree 架構的方法，在均勻分佈的情況下，於高維度（六維以上）的情況下效果明顯較 R* tree、R*a tree、X tree 及 Xa tree 等四種樹狀結構佳；而在五維以下的中低維度情況下，於大範圍搜尋時效果仍較其它四種方法為佳，小範圍搜尋時，效果仍優於另四種方法。

就低維度資料而言，由於在低維度的情況下，X tree 較複雜的分裂方法及 supernode 未能發揮效用，使得 X tree 與 R* tree 的樹狀結構相近。而 R*a tree 及 Xa tree 在中間節點加入一記錄子節點特徵值的欄位，能使範圍查詢上更有效率，尤其是查詢的範圍愈大，記錄子節點的特徵值欄位愈能支援範圍查詢時的需求。而本研究利用 SOFM 改良 Xa tree 架構的樹狀結構，因此同樣具有利於大範圍查詢的優勢。然而隨著葉節點平均佔滿率的提高，亦即平均每個葉節點的儲存空間使用率愈高的情況下，中間節點與葉節點的數目顯著下降，雖然有利於提升範圍查詢的節點存取效率，但由於節點 overflow 的情況亦因葉節點平均佔滿率的提高而增加，造成 overlap 的情形遞增，而低維度下的分支容量較高維度大，且 overlap 的情形尚不如高維度情形嚴重，因此 X tree 的 supernode 機制不易發生，使得本研究之 SOFM X-tree 在低維度、範圍小時，查詢上的節點存取次數仍優於 R*a 及 Xa tree 方法。

在高維度情況下，由於隨著維度愈高、overlap 的情形愈嚴重，雖然 X tree 具有將嚴重 overlap 的中間節點合併成 supernode 的機制，但整體而言，X tree 儲存空間的使用率仍維持在 66%，且節點的分裂亦會使節點的空間使用率下降。而本研究運用 SOFM 改良 Xa tree 的樹狀結構，隨著葉節點平均佔滿率的提高，亦即平均每個葉節點的儲存空間使用率愈高的情況下，中間節點與葉節點的數目顯著下降，雖然葉節點的平均佔滿率愈高造成節點 overflow 的情況增加，使 overlap 的情形遞增，但由於 X tree 的 supernode 機制將高度重疊的中間節點合併，甚至形成一大型中間節點而降低樹高層數，例如六維以上、葉節點平均佔滿率 0.8 以上的資料，平均而言能較另四種樹狀結構壓低一至二層的樹高，因此在愈高維度的情況下，本研究之 SOFM x_tree 在範圍查詢上的效能愈佳。

整體而言，本研究將 SOFM 修改為 ring 架構輸出，再將其對應至 OLAP 索引搜尋問題，可有效減少節點數目、壓低樹高層數的策略，達到減少範圍查詢時節點存取次數的目的，對資料倉儲之索引，將有很大的貢獻。

致謝

本研究經費由國科會專案補助，計劃編號：NSC 90-2520-S-024-006-，特此致謝。

參考文獻

[1] 陳科學，“適用於關聯式線上分析處理資料的索引架構之設計與研究”，國立台南師範學院資訊教育研究所碩士論文，台南：國立台南師範學院，2000。

[2] 郭義中，“遺傳演算法於資料倉儲中構建資料方體之應用”，私立義守大學資訊工程系碩士論文，高雄：私立義守大學，1999。

[3] 蔡宗達，“交錯式二元樹之多維資料索引架構”，私立淡江大學資訊管理系碩士論文，台北：私立淡江大學，1997。

[4] Beckmann, N., Kriegel, H. P., Schneider, R., and Seeger, B., "The R*-tree: An efficient and robust assess method for points and rectangles", Proceedings of ACM - SIGMOD International Conference on Management of Data, pp. 322-331, Atlantic City, NJ, 1990.

[5] Berchtold, S., Keim, D. A., & Kriegel, H. P., "The X-tree: An index structure for high dimensional data", Proceedings of 22th Internal Conference on Very Large Data Base, pp.28-39, Mumbai (Bombay), India, 1996.

[6] Guttman, A. (1984), "R-trees: A dynamic index structure for spatial searching", Proceedings of ACM - SIGMOD International Conference on Management of Data, pp. 47-57, Boston, MA, 1984.

[7] Jurgens, M. and Lenz, H. J., "The R* a tree : An improved R* tree with materialized data for supporting range queries on OLAP data", DEXA Workshop, 186-191, 1998.

[8] Kohonen, T., "Self-Organization and Associative Memory", Springer Verlaog, Berlin, 1980.

[9] Lippmann, R. P., "An introduction to computing with neural nets", IEEE ASSP Magazine , 4, 4-22, 1987.

[10] Nievergelt, J., Hinterberger, H., and Sevcik, K. C., "The Grid File: An adaptable, symmetric multikey file structure", ACM Transaction on Database systems, 9(1), 38-71, 1984.

[11] Neil, P. O. and Graefe, G., "Multi - table joins through bitmapped join indices", SIGMOD Record, 24(3), 8-11, 1995.

[12] Robinson, J. T., "The K-D-B tree: A search structure for large multidimensional dynamic indexes", Proceedings of ACM - SIGMOD Conference, pp. 10-18, Ann Arbor, Michigan, 1981.

[13] Roussopoulos, N. and Leifker, D., "Direct spatial search on pictorial databases using Packed R-trees", Proceedings of ACM - SIGMOD International Conference on Management of Data, pp.17-31, 1985.

[14] Sarawagi, S., "Indexing OLAP Data", IEEE Bulletin on Data Engineering, 20(1), pp. 36-43, 1997.

[15] Sellis, T., Roussopoulos, N., and Falousos, C., "The R+ tree: A dynamic index for multi-dimensional objects", Proceedings of 13th International Conference on Very Large Data Base, pp. 507-518, Brighton, England, 1987.

[16] Sun, K. T., & Fu, H. C., "A hybrid neural network model for solving optimization problem", IEEE Transactions on Computers, 42(2), pp. 218-227, 1993.