探討一個更有效的 K-means 組群化演算法

Investigations on An Efficient K-Means-Based Clustering Algorith

洪明傳 mchong@fcu.edu.tw
楊東麟 dlyang@fcu.edu.tw
張金華 jhchang@soft.iecs.fcu.edu.tw
劉振緒 jsliu@fcu.edu.tw

逢甲大學資訊工程研究所
台中市西屯區文華路100號

摘要

當我們要分割資料集(dataset)成爲多個組群(cluster)的時候，K-means 組群化(clustering)演算法可以說是一種很合適而且普遍使用的方式，然而大多數以 K-means 爲基礎的組群化實作卻當要花費大量的距離計算次數才能夠算出收斂時的形心。在本篇文章中，我們提出一個新的以 K-means 爲基礎的組群化演算法，它以一種較有效率的方式算出與傳統方法一樣結果的形心。在本演算法中，我們將原始的資料集分割成一個個大小相同的子空間稱爲單位區塊。我們使用一種簡單的計算方式算出每個單位區塊的形心稱爲CUB(centroid of unit block)，而這些 CUB 可以代表簡化後的資料集，其個數遠小於原始的資料集。利用這簡化後的資料集與處理位於組群邊界上單位區塊的機制，便可以很快的算出收斂的形心。我們的方法不但可以加速形心收斂的時間與計算次數，亦可得到與傳統方式一樣的品質。爲了方便使用者的應用，我們在文中特別討論單位區塊最佳化和效能評估。

關鍵字：組群化(clustering)，K-means，形心(centroid)，資料挖掘(data mining)

1. 前言

對一空間點或樣本(pattern)的集合分割成多個集團(group)或組群(cluster)的程序稱爲組群化(clustering)，這些空間點或樣本的屬性爲非監督型的(unsupervised)資料，將無法明顯來訂定類別。組群化的問題是如何有效且迅速的將集合中相似性質的資料結合成一組群。一個良好群組化的結果，可使在同一組群中的樣本彼此具有相似的性質。反之，屬於不同組群的樣本性質是不同的。至今，它已被廣泛的應用到各種不同領域之中，諸如資料挖掘(data mining)[16]、統計資料分析[5, 17]、壓縮技術[6]及人工智慧[4]等等。

爲了有效且迅速的完成組群化的程序，有許多的演算法(algorithm)被提出[1, 7, 8, 9, 10, 11, 12, 13, 14, 15]，其中以 k-mean 組群化演算法最爲普遍的被使用[7]，它會被採用主要是容易實作，而且能有效獲致良好的結果。然而，直接式 k-means (direct k-means)組群化演算法其每一回合(iteration)的執行時間是與樣本數乘上組群個數的結果成正比[1]，使得在大樣本數的資料集(dataset)及多組群的情況下，它將很耗費距離計算時間。如此，有些應用系統要求反應時間非常迅速時將無法滿足要求，要如何改進它的計算時間就成爲一個重要的課題。因而，有 Alsabti 等作者提出一以個 k-d tree 資料結構的演算法[1]，其在樣本歸屬組群的計算上採行一個刪除的機制，的確可以縮短距離計算時間並能顯著的改善其執行效能。然而，它在建立 k-d tree 結構所需花費的時間也不少，主要是與資料集的大小成比例。本文提出另一演算法，係利用分割成多個大小相同的次空間集合(subspace set)，再經由簡化程序並配合刪除機制，結果大大降低距離計算時間，其效能遠較直接式k-mean 及 Alsabti 的演算法爲佳，本文所提演算法的優點與貢獻，可由文中有關效能的分析上得到印證。

2. 相關文獻

在本節中，我們簡介直接式 K-mean s與 Alsabti 的 K-mean 組群化演算法。

2.1 直接式 K-means 組群化演算法

MacQuee 於 1967 年提出此演算法[7]，也是最早的組群化計算方式。其形心的計算程序簡述如下：

(i) 在原始的資料集中隨機選取 K 個候選形心。
(ii) 對於原始資料集內的每個樣本點，計算其最爲接近的候選組群形心，並將其歸屬此一候選組群。
(iii) 執行完一回合的全部資料集的組群分配後，對於歸屬每個候選組群的所有樣本點再重新計算出其所構成的形心。
(iv) 計算錯誤函數(error function)的值不再改變時，表示已找到最後收斂的形心位置，而完成整個組群化的程序；否則回到(ii)。

由演算法可以看出，每個樣本點必須在 K 次的距離計算後才可以知道其最爲接近的形心，如果資料集共 n 個樣本點，則必須經過 nK 次的距離計算後才可以得到一組新的候選形心，是故，當資料集的維度爲 d 時，每一次距離計算的時間複雜度爲$O(d)$，此演算法在每一回合的候選形心計算其時間複雜度爲$O(nKd)$。我們可以看出此時間複雜度與樣本點的大小和欲分割的組群數K皆成正比，在這樣子的情況之下進行形心的計算是相當耗時

的。

2.2 Alsabti 的 K-means 組群化演算法

在 1998 年 Alsabti 提出一個新的 K-means 組群化演算法[1]。其目的是在計算新候選形心時，儘量以減少距離計算的次數，以便有效率的算出收斂後的形心。其主要作法是以 k-d tree 的方式分割資料集成多個次空間，稱爲 box。再利用其所提出的刪除功能(pruning function)以去除那些與 box 距離較遠的候選形心，如此便可以省掉一些距離計算次數。其演算法在刪除機制雖能夠爲每個 box 去除部份候選形心的距離不需要計算所節省的時間，卻增加了分割資料集所花的時間，其效果雖較直接式 K-mean 演算法佳，但仍不是最好。

3. 我們的新演算法

我們所提出新的 k-mean s組群化演算法主要目標在於減少樣本點對 k 組群候選形心距離的計算次數，並能加速收斂時間，使總執行時間降低，以獲得良好的效能。以下謹就此演算法的程序做一陳述。

3.1 分割資料集

首先我們將需分組群的資料集分割成數個大小相等的次空間集合，我們稱它爲單位區塊(unit block) 。到底要分成幾個區塊較適合，也是本文要探討的主題之一，其關係著執行效能的良窳，此問題在第 4 節單位區塊的最佳化中會有詳細的說明。

分割單位區塊的做法是以資料集依序選取某個維度(dimension)，例如二度空間的 x 軸維度，以其最大值與最小值的差值所決定的範圍進行等份分割，其分割的個數即爲單位區塊的個數，我們採行二元分割(binary partition)的方式，其分割位置是以每次分割後長度的中點再行分割。因此，各區塊在該維度的範圍可迅速的得知。我們只需掃瞄資料集二次即可確定各樣本點是在那一區塊之內，很容易而且很快的完成分割資料集的程序，做爲下一簡化步驟的先行作業。Alsabti 的演算法也須要先分割資料集，但是我們所提的演算法有著更佳的效能。此因 Alsabti 的演算法每次在 tree 的節點要再分割爲次一層的子結點時，必須再次掃瞄資料集以計算出此節點的次空間集合範圍，依此遞回處理直到所設定的階層爲止。因爲每節點分割時需先掃瞄資料集，因而會增加其處理的時間所致。

將資料集分割成我們所設定的區塊數後，每一區塊中可能存在某些樣本點，假如區塊中未出現任一樣本點時，則此區塊將被捨棄在後序的作業中不予考慮。分割完後的狀況，如 Figure 1 初始的資料集，以二維圖例表示。

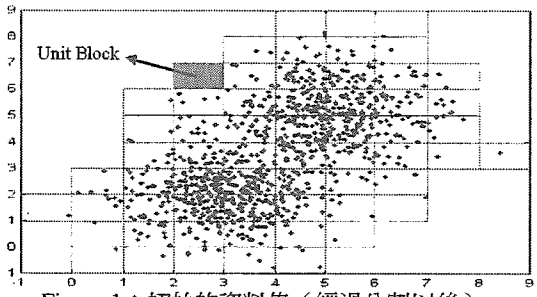

Figure 1：初始的資料集（經過分割以後）。

3.2 簡化資料集

在完成資料集的分割之後，我們分別對每一個存在有樣本點的單位區塊，計算出在它的範圍內所有樣本點的形心值(centroid)，我們就以此形心資訊代表區塊中所有的樣本點。因此，可將龐大樣本的資料集只以大約爲區塊數的形心點數來表示，如Figure 2 所示。而後在分割組群的計算上都以此形心資訊來處理，可大大降低距離計算成本，相對的加快組群分割收斂的速度。

我們以形心來描述區塊所包含的次資料集，是使用相對的統計量來表示，這些統計量是在分割資料集時，於第二次掃描資料集的同時，計算得到。這些統計量可作爲劃分組群歸屬的計算時被使用，可省略再次掃描整體資料集的次數與時間，能夠減少 I/O 次數，是提昇整體效能的另一有用的技巧。

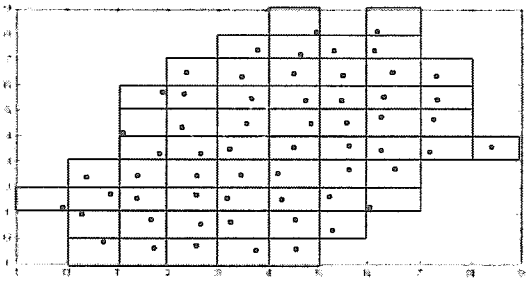
Figure 2：簡化後的資料集。

3.3 計算組群的形心

K-mean 組群化的做法是以反覆方式計算K個組群的形心，直到前、後兩次的K個形心誤差量的和沒有改變爲止，此即完成所有資料集組群化的程序。在整個處理的過程中，組群完全是以形心來表示，對於不是最終的組群形心我們稱爲候選形心(candidate centroid)。以下就是我們所提出的演算法對於組群形心計算的步驟，其中使用到的符號定義在Table 1：

(i) 首先從原來的資料集中隨機選取K個樣本點，做爲K個組群的初始候選形心，這與直接式k-means及Alsabti的演算法的做法是相同。並且我們在實驗時是採用相同的初始候選形心，因爲初始候選形心位置會影響最後的執行效能，如果幸運的隨機選到的初始形心是接近真正最後的組群形心，則會很快的收斂，相對的執行時間會更短。因此，我們選用相同的初始候選形心，這可使我們的演算法與另兩個

方法在比較時有共同的基準。有關初始候選形心的純化問題，是另一探討的課題，請參考[3]。

Table 1：本演算法所用到的符號

a	單位區塊的個數
α	在組群邊界上單位區塊的個數
k	組群的個數
k'	共用邊界的組群平均個數
n	所有樣本點的個數
i_l	樣本點
d	樣本點的維度
W_j	形心點
C_j	組群
BUNB	不在邊界上的單位區塊
BUB	在邊界上的單位區塊
CUB	單位區塊的形心
LSUB	單位區塊內所有樣本點的線性總合
WUB	單位區塊內所有樣本點的個數

(ii) 有了初始候選形心，接著利用前小節所提過簡化的資料集來計算新的形心，以歐基里德(Euclidean)的方法計算兩空間點間的距離L。

$$L=\sqrt{\sum_{i=1}^{d}|a_i-b_i|^2}$$

我們計算代表每一單位區塊的形心CUB_a到K個組群候選形心的距離，找到距離最接近的候選形心，我們就將此區塊中的所有樣本點都歸入這一最近的組群中，以此方式我們可以非常快速的達成組群的分割。

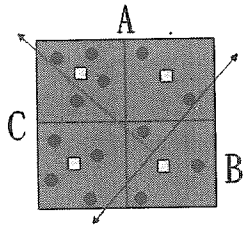

Figure 3：組群間的邊界區塊。

然而，有時會發生某個區塊的CUB_a同時距某幾個組群候選形心的距離都很接近。這時候，我們不能將此區塊的所有點都歸到最接近的一個候選組群中，因爲這區塊中的樣本點可能有一部份是歸屬於某一較接近的候選組群，而其他的樣本點則應歸入另外接近的組群，才是正確。我們將這種距離兩個或兩個以上候選組群都很接近的區塊稱爲邊界區塊(boundary block)，如Figure 3所示，圖中A，B與C分別爲三個組群，箭頭爲虛擬的邊界且跨過四個單位區塊，圓點爲樣本點，小區塊點爲CUB。要如何判斷單位區塊是否爲邊界區塊，我們的做法是當區塊的CUB_a距其附近幾個候選形心距離的差值，假如小於此區塊對角線的長度時，就認定此區塊爲多個組群所共用的邊界。因此，對於邊界區塊中的所有樣本點必須逐一計算它距那一候選形心最近，再將此點歸入這最近的組群中。在判斷邊界區塊上樣本點是歸屬那一組群，我們只要計算該樣本點與共用此邊界區塊的少數幾個候選形心間的距離就可以，共用邊界的組群平均個數爲K'。而不像直接式K-means演算法是每一點樣本都必須計算它與K個候選形心間的距離。K'值遠比K值爲小，加上我們使用的刪除機制(pruning mechanism)，確可減少距離計算的次數，使得執行的效能可以顯著的提昇。

(iii) 在執行完一個回合的組群的重分配之後，我們以這些新分配到K組群的樣本點重新算出新的組群候選形心。

(iv) 再使用誤差函數E，計算出資料集所有樣本點距這新K組群候選形心差值的平方和。

$$E=\sum_{j=1}^{k}\left(\sum_{(CUB_a\subset BUNB)\in C_j}|CUB_a-W_j|^2+\sum_{(i_l\subset BUB)\in C_j}|i_l-W_j|^2\right)$$

假如新的E值與前一回合的E值相等，表示組群的候選形心不再改變，即完成整個組群分割的工作。否則前、後回合的E值不相等時，則以新的候選形心再一回合重新分割資料集，直到我們設定的最大執行回合次數爲止。本演算法在計算誤差函數E的時間亦較直接式及Alsabti的演算法所花的時間爲短。新演算法的虛擬碼如Figure 4、5與6所示，因爲篇幅關係不做詳述，請參考[2]。

Proc main ()
Assume there are n data elements $(i_1,\ldots,i_n)$ and g unit blocks $(UB_1,\ldots,UB_g)$ in the dataset, and the number of clusters is k $(C_1,\ldots,C_k)$, such that $l\in\{1\ldots n\}$, $j\in\{1\ldots k\}$, $a\in\{1\ldots g\}$

/* Select initial k centroids $(W_1,\ldots,W_k)$ from the original dataset */
$W_j=i_l$
/* Partition the dataset into several unit blocks */
partition_dataset (original dataset)

/* Calculate centroids for simplified dataset */
W_j =Calcu_Cen_for_Simp_Dataset()

/* Calculate final centroids */
repeat
for each CUB_a
if UB_a is on the boundary *then*
for each $i_l\in UB_a$
find the nearest centroid P_j from k' centroids to which UB_a belong.
assign i_l to P_j
update the centroid's statistics
else
find the nearest centroid P_j

assign $LSUB_a$ and WUB_a to P_j
update the centroid's statistics
end if

Compute the error function

$$E = \sum_{j=1}^{k}\left(\sum_{(CUB_a \succ BUNB) \in C_j} |CUB_a - W_j|^2 + \sum_{(it \succ BUB) \in C_j} |it - W_j|^2\right)$$

until E does not change significantly.
End main

Figure 4：我們演算法的主程式

Proc partition_dataset (original dataset)
for each dimension D_m of the dataset
/* find the range for dimension D_m */
Range_Max[m] = the maximum value of dimension D_m
Range_Min[m] = the minimum value of dimension D_m
/* calculate the number of segments it would partition for dimension D_m */
/* *Num_of_Split*[m] is the number of splits for dimension D_m */

Num_of_Seg[m] = (*Range_Max*[m] - *Range_Min*[m]) / *Num_of_Split*[m]

for each data element it
/* calculate the *UB* that data element it belongs to */
for each dimension D_m of it, named $it[m]$
Point_in_Dim[m] = ($it[m]$ – *Range_Min*[m]) / *Num_of_Seg*[m]
/* Use the value of *Point_in_Dim*[m] to calculate the UB_a that it belongs to, named *UB_Location* of it */
/* calculate *LSUB* and *WUB* for each UB_a */
*UB*_process (it, *UB_Location*)
/* compute *CUB*: Centroid of Unit Block */

$$CUB_a = \frac{LSUB_a}{WUB_a}$$

End partition_dataset

Figure 5：分割資料集

Proc Calcu_Cen_for_Simp_Dataset ()
/* Calculate centroids */
repeat
for each CUB_a
find the nearest centroid P_j
assign $LSUB_a$ and WUB_a to P_j
update the centroid's statistics
Compute the error function

$$E = \sum_{j=1}^{k}\left(\sum_{(CUB_a \succ BUNB) \in C_j} |CUB_a - W_j|^2\right)$$

until E does not change significantly.
End Calcu_Cen_for_Simp_Dataset

Figure 6：由簡化後的資料集計算形心

4. 單位區塊最佳化

我們的演算法主要是將資料集的分佈先分割成許多等份的單位區塊，再加以簡化，利用區塊的形心值代表區塊中所有樣本點的資料，這樣可使後續的距離計算上大大降低執行的時間。區塊切割的方式及切割的個數就成爲新演算法執行效能好壞的重要參數。尤其是單位區塊到底應該分成幾個才是最佳，這是我們要進一步探討的課題。

在資料集的樣本點數n及組群個數K已知的條件下，以新演算法經由實作所得的數據顯示，當單位區塊個數爲遞增或遞減兩種狀況時，我們發現以下的現象：

(i) 當單位區塊個數增加時，相對產生組群邊界的區塊也會隨著增加，但在每一區塊中平均存在的樣本點數卻相對的減少。我們發現，雖然邊界區塊個數爲增加，但需對各別樣本點判定其組群歸屬的總點數卻相對的減少，也使得需要計算距離的總次數跟著減少。因此，要判定所有邊界區塊中樣本點歸屬那一組群的計算時間會隨著分割的單位區塊個數的遞增而遞減。相反的，單位區塊個數減少時，所產生的組群邊界區塊個數也會減少，而每一區塊平均點數卻相對的增加，使得需判別歸屬的樣本點個數隨著遞增，要計算距離的次數也就相對增加，這可由Figure 7看出，是由實驗數據(DS3,k=32,iteration=10)導出的式子 $y=57912x^2-1581766x+10746945$ 來表示，式中y表距離計算的次數，x爲$\log_2 a$，a爲單位區塊的個數。我們是使用[15]中的DS3資料集，設定K=32以及最大執行回合數10爲例。

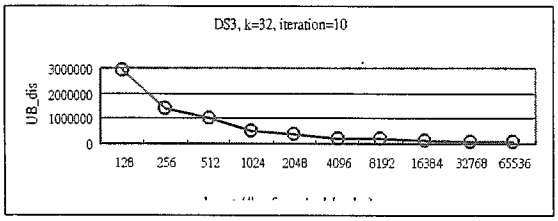

Figure 7：在組群邊界上所有樣本點的距離計算次數與單位區塊個數關係圖(UB_dis表距離計算次數)

(ii) 同樣的當單位區塊個數增加時，不在邊界上的區塊個數亦隨著遞增。我們要判定歸屬那一個組群，就如前章節所述，以區塊形心對所有K組群候選形心間距離最近者，將此區塊範圍內所有點劃歸此一組

群。其距離計算次數可由不在邊界上的區塊個數與K的乘積表示。因此，計算距離的次數就呈現與單位區塊的個數增加而遞增的現象，相反的則會遞減，此現象如Figure 8所示。同樣可由實驗數據導出式子$y=110055x^2-2067684x+9456495$。

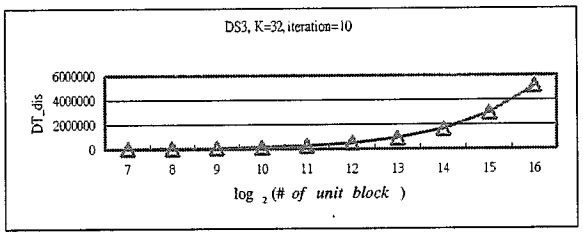

Figure 8：不在組群邊界上所有CUB的距離計算次數與單位區塊個數關係圖(DT_dis表示距離計算次數)

新演算法對資料集分組群的做法，主要是分爲邊界區塊與非邊界區塊兩種處理方法，距離計算的總次數應爲兩者的總和(Total_dis = UB_dis + DT_dis)，這結果如Figure 9所示，其表示式爲$y=167967x^2-3649450x+20203440$ 。另外，計算距離的總次數與分割的組群數K亦有關係，其結果如Figure 10所示，我們選用K值爲8到64個組群數來分析。

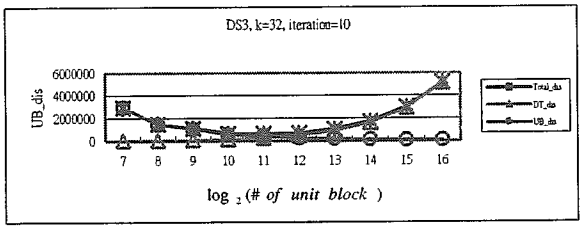

Figure 9：本演算法的總距離計算次數(Total_dis）與單位區塊個數關係圖

接著希望能找到最佳的單位區塊個數，以使新的演算法獲得最好的效能。最佳區塊個數可由Figure 10看出，它是出現在曲線最低處所對應的區塊個數即是。我們在資料集的樣本點數n、組群數K及共用邊界區塊的平均組群數K'已知的條件下，以Total_dis方程式對區塊數的變數微分一次等於零，即可求解出最佳的區塊個數。這樣的推導結果證實是與我們實作的實驗數據相當吻合，印證了我們提出的區塊最佳化計算模式是正確的。

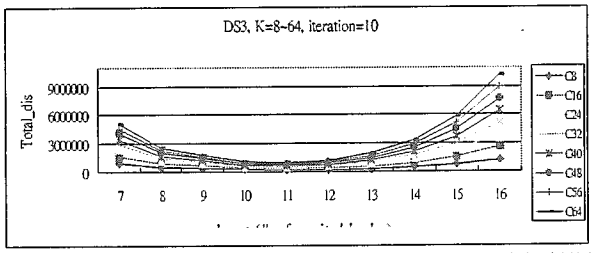

Figure 10：不同K值時，距離計算次數與單位區塊個數關係圖。

我們以三種不同的資料集如DS1、DS2及DS3[15]，分別在不同組群個數的情況下，所對照的最佳單位區塊個數如Table 2所示，U' 表示推導出的最佳區塊個數，$U_{optimal}$ 表示由實驗得出的最佳區塊個數，由此表可以看出U' 都在$U_{optimal}$ 附近振盪，可以證明我們所提出的距離計算模式是正確的。

Table 2：最佳單位區塊個數對照表

	k=8			*k*=16		
	DS1	DS2	DS3	DS1	DS2	DS3
U'	1710	5970	2560	1330	5950	2530
$U_{optimal}$	2048	4096	2048	1024	4096	2048
	k=24			*k*=32		
	DS1	DS2	DS3	DS1	DS2	DS3
U'	1310	5910	2490	1290	5890	2460
$U_{optimal}$	1024	4096	2048	1024	4096	2048
	k=40			*k*=48		
	DS1	DS2	DS3	DS1	DS2	DS3
U'	1270	5870	2430	1250	5840	2390
$U_{optimal}$	1024	4096	2048	1024	4096	2048
	k=56			*k*=64		
	DS1	DS2	DS3	DS1	DS2	DS3
U'	1220	5810	2360	1200	5780	2320
$U_{optimal}$	1024	4096	2048	1024	4096	2048

由Table 2可以看出最佳區塊個數與資料集的分佈有關，而與組群個數沒有顯著的關係，就以DS3資料集來說，不論組群個數是多少，以二元分割方式其最佳區塊個數都是2048。而組群數K爲32時，資料集DS1、DS2及DS3的最佳區塊數分別爲1024、4096及2048有所不同，它是與資料的分佈狀況有關。

5. 效能分析與比較

以上我們對新演算法的程序、做法及優點做有系統的分析與說明，強調它的效能比其他 K-mean 演算法好。我們進一步再針對其時間的複雜度來探討，依其程序分成幾個部份：

(i) 將資料集等分成幾個單位區塊，並且以區塊形心進行簡化，此階段所需時間爲 O(2n)。其細部包括：
   a. 讀取資料集找到最大及最小的樣本點的時間是 O(n)；
   b. 以二元分割方式，劃分單位區塊至最佳個數所需時間爲 O(log a)；
   c. 決定單位區塊空間中的樣本點及其形心，並以統計量描述，其所需時間 O(n)。

(ii) 需經過多個回合的重新分割組群，才能得到最後我們所要的結果，每一回合執行所需時間是$O((n/a)\alpha k')+O((a-\alpha)k)$。

(iii) 計算新的候選形心時間爲 O(1)。

(iv) 計算誤差函數所需時間是 O(((n/a) α +(a- α))d)。
至於其他演算法所需時間複雜度可參考[1, 7]。
爲了驗證新演算法是優於其他的演算法，我們選擇幾種不同的資料集來實驗，以比較新演算法與另兩種 K-mean 組群化演算法的執行效能，我們以總距離計算次數、總執行時間及到達收斂的回合次數做爲比較的基準。我們是在 Sun Ultra Enterprise 1000 機器上使用 Solaris OS，並在 250MHz 的處理機、4GB 記憶體容量的環境下進行實作。
所選的資料集如下：

- DS1 爲 grid 狀的均勻分佈(uniform distribution)、樣本大小爲 100,000、原始分佈 100 個組群及維度爲 2。
- DS2 爲 Sine 狀的特別分佈(special distribution)，其樣本大小、原始組群數及維度同 DS1。
- DS3 爲隨機分佈(random distribution) ，其樣本大小、原始組群數及維度亦同 DS1。
- R1 至 R12 也都爲隨機分佈，但其樣本大小、原始組群數及維度互有變化[1]。

在實驗中使用新演算法時，我們是以最佳的單位區塊數來分割資料集。對於三個不同 K-mean 演算法都使用相同的起始組群候選形心，分別進行組群化處理，所得到的效能指標如 Figure 11、12 與 13。

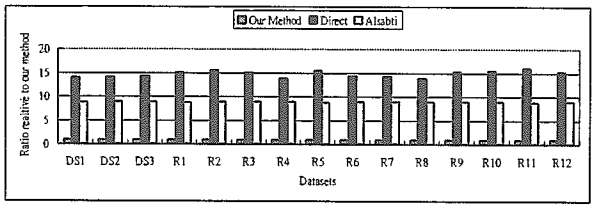

Figure 11：當 k=32 時的時間比

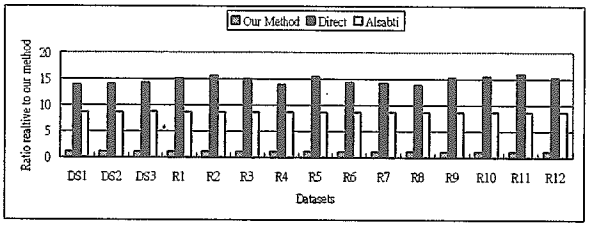

Figure 12：當 k=32 時的總距離計算次數比

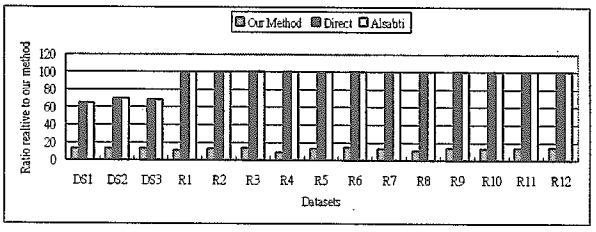

Figure 13：當 k=32 時的收斂次數

Figure 11 顯示 3 種演算法分別將各種資料集分成32 個組群所需的時間比，其中以新演算法爲時間 評比的基準，可看出直接式演算法是它的13 倍至 15 倍，而 Alsabti 的演算法亦由 8 倍至 9 倍。而 Figure 12 是表示以距離的計算次數來衡量效能，其結果與時間的比較相似。另外，達到收斂的回合次數與效能亦有密切關係，同時也是影響準確度的參數。我們以誤差函數來判斷組群化是否已完成，爲避免過多回合的執行，往往會另設定一個最多回合數做爲結束，假如最多回合數設定過少則會影響結果的準確性。Figure 1 就是表示各種資料集分成 32 個組群要到達收斂的回合次數，我們可明顯的看出新演算法的收斂次數遠較其他兩種演算法爲少。直接式 K-mean 及 Alsabti 兩種演算法在最多回合限度設爲 100 時，由圖中可看出還有部份資料集尚未到最後的收斂就被停止執行，其準確度可能受影響；但對新演算法而言，早在 20 回合前就已達到收斂，其準確度不受影響，顯然較其它兩演算法爲佳。

6. 結論

本篇文章提出一個較有效的 K-mean 組群化演算法，我們先將資料集分成幾個等分次空間集合的區塊，再以簡化計算量的技巧進行 K 組群的分割，確實可快速且準確的達到收斂的效果。它較直接式 K-mean 及 Alsabti 兩演算法的效能，不論是在執行總時間或單回合形心距離計算的次數上都來得優異，這可從時間複雜度的分析看出，並經由實作的數據可得到印證。
文中我們對於區塊個數的最佳化有完整的探討，它與分割的組群數 K 沒有顯著的關係，但與資料集的分佈有關。目前我們是以二元方式進行單位區塊的分割，其它分割方式是未來我們要再探討的工作。
新演算法完成組群化的時間較直接 K-mean s演算法或 Alsabti 演算法所需時間顯著的減少很多。這對於反應要求快速的應用系統而言，的確是有很大的助益。

Reference

1. K. Alsabti, S. Ranka, V. Singh, An Efficient K-Means Clustering Algorithm, *PPS/SPDP Workshop on High performance Data Mining*, 1997.
2. D.L. Yang, J.H. Chang, M.C. Hong and J.S. Liu, An Efficient K -Means-Based Clustering Algorithm, accepted by IAT'99.
3. P. S. Bradely and U. Fayyad, Refining Initial Points for K-Means Clustering, *Proc. 1 th International Conf. Machine Learing*. Morgan Kaufmann 1998.
4. M.S. Chen, J. Han, P. S. Yu, Data Mining: An Overview from Database Perspective, PKDD98.
5. J. Banfield and A. Raftery, Model-based gaussian and non-Gaussian Clustering, *Biometrics*, Vol. 49: 803-821, pp. 15-34, 1993.
6. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications, *Data Mining and Knowledge Discovery*, Vol. 1, no. 2, 1997.

7. R. C. Dubes and A. K. Jain. *Algorithms for Clustering Data*. Prentice Hall, 1988.
8. M. Ester, H. Kriegel, J. Sander, and X. Xu. A Densit -Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proc. Of the 2nd Int'l Conf. On Knowledge Discovery and Data Mining*, August 1996.
9. M. Ester, H. Kriegel, and X. Xu, Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification. *Proc. Of the Fourth Int'l. Symposium on Large Spatial Databases*, 1995.
10. J. Gracia, J. Fdez-Valdivia, F. Cortijo, and R. Molina. Dynamic Approach for Clustering Data. *Signal Processing*, 44(2), 1994.
11. D. Judd, P. McKinley, and A. Jain. Large -Scale parallel Data Clustering, *Proc. Int'l Conference on Pattern Recognition*, August 1996.
12. L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley & Sons, 1990.
13. R. T. Ng and J. Han, Efficient and Effective Clustering Methods for Spatial Data Mining, *Proc. of the 20th Int'l Conf. on Very Large Databases, Santiago*, Chile, pages 144-155, 1994.
14. E. Schikuta. Grid clustering: An efficien Hierarchical Clustering Method for Very Large Data Sets, *Proc. 13th Int'l Conference on Pattern Recognition*, 2, 1996.
15. T. Zhang, R.Ramakrishnan, and M. Livny, BIRCH: An Efficient Data Clustering Method for very Larg Databases, *Proc. Of the 1996 ACM SIGMOD int 'l Conf. On Management of Data*, Montreal, Canada, Pages 103-114, June 1996.
16. Fayyad, U., G. Piatetsk -Shapior, P. Smyth, and R. Uthurusamy (Eds.) *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996.
17. L. Kaufman and P. Rousseuw, 1989. Finding Group in Data, New York: 4John Wiley and Sons.
18. U. Fayyad, C. Reina, and P. Bradley, Initialization of Iterative Refinement Clustering Algorithms, The Fourth International Conference on Knowledge Discovery and Data Mining, New York City, August 27-31, 1998.