物件—屬性空間的分割技術

高學東　　武森
北京科技大學經濟管理學院　　北京科技大學經濟管理學院
gaoxuedong@manage.ustb.edu.cn

摘要

高維稀疏資料是比較常見的一種資料形式，對高維稀疏資料的處理能力是資料挖掘演算法研究的一個重要內容。本文在系統總結資料挖掘相關理論和現有各類演算法的基礎上，針對高維大資料集的資料，提出用聚類方法的思想在正式的資料挖掘前對等待挖掘的資料進行物件—屬性空間的劃分方法。劃分後資料的規模和資料的維數都會有一定規模的縮減，正式的資料挖掘將在已劃分的維數相對較低的類中進行，從而提高各種挖掘演算法的效率和挖掘結果的品質。文中最後給出了一個數值例子用以示例該演算法的計算過程。

關鍵詞：資料挖掘，物件集合內聚度，預聚類，空間分割。

Abstract

High-dimensional sparse data is a kind of relatively common data format, and the processing capability of data mining algorithm to high-dimension sparse data is an important part of data mining algorithm research. Based on the summary of the theories of data mining and many available algorithms, this paper puts forth the idea of Clustering for the high-dimensional sparse data by adopting the method of partitioning Object-Attribute Space prior to the data mining process. After the partition, the scale and the dimension of the original data will reduce to a certain extent. Therefore, the data mining can be processed in the partitioned and relatively lower dimensional clusters, so that the efficiency of the algorithms and the quality of the mining result will be improved. In the last part of the paper, a numerical value example is provided to show the whole process of the method.

Keywords: Data Mining , Object Aggregate Cohesion , Pre-Clustering , Space Partition.

1. 物件—屬性空間分割問題的提法

在資料挖掘的應用中，經常會遇到這類問題：物件的數目很多，用來描述物件的屬性也很多，但是對於每一個物件來說具有屬性值的屬性個數占總屬性個數的比例很小。例如鋼鐵企業中，有很多的客戶（物件）和很多的產品（屬性），各個客戶購買的產品一般很少，而且各客戶購買的產品種類也有很大不同。

在文獻[3]中，上述問題被稱為具有“高維稀疏資料”的問題。對於這類問題，一般只關心各物件間非零屬性的相似性[2]，因此引入稀疏特徵來描述物件屬性的稀疏性。

定義 1（稀疏特徵）：假設有 n 個物件，描述第 i 個物件的 m 個屬性值分別對應於區間變數值 xi1，xi2，…，xi m，將其轉換為二態變數並表示為 yi1，yi2，…，yi m，轉換方法為：

$$y_{ij}=\begin{cases}1, & \text{如果} x_{ij}>0\\ 0, & \text{如果} x_{ij}=0\end{cases} \qquad (1)$$

其中 {1，2，…，n}； {1，2，…，m} yij，{1，2，…，n}；

{1，2，…，m}表明了各個物件在個屬性上的稀疏情況，稱為第 i 個物件在第 j 個屬性上的稀疏特徵。如果 yij=1，表明第 i 個物件在第 j 個屬性上是非稀疏的；如果 yij=0，則表明第 i 個物件在第 j 個屬性上是稀疏的。實際上從客戶購買產品的角度來看，如果 yij=1，表明第 i 個客戶購買了第 j 種產品；如果 yij=0，表明第 i 個客戶沒有購買第 j 種產品。

稀疏特徵表是物件與屬性之間的二維表，它描述了每個物件屬性稀疏特徵。若相應屬性值非零，則表中該物件與該屬性相對應的位置取 1，否則取 0，如表 1 所示。

這種情況造成在資料存儲中很多物件的屬性取值為零，而各個物件的非零屬性分佈很分散，在此種情況下進行資料挖掘是非常耗費時間和空間資源的，很可能影響資料挖掘的效果。很多聚類演算法在屬性為維數比較低的情況下能夠生成品質比較高的聚類結果，卻難以應用于高屬性維資料的情況，有時甚至可能會產生錯誤的聚類結果[1]。因此，考慮在對類似資料進行挖掘前先對其進行預聚類，使得到的預類（預聚類得到的結果）中物件的非零屬性盡可能集中。預聚類結束後再在各個子類中進行相應的資料挖掘，降低參與資料挖掘資料的規模。

表 1 6 個物件 10 個屬性的稀疏特徵表

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
O_1	1	0	1	1	0	1	1	0	0	1
O_2	0	0	1	1	0	0	0	0	1	1
O_3	0	0	1	1	0	0	0	1	1	1
O_4	1	1	1	1	0	0	0	1	0	1
O_5	1	0	0	1	1	0	0	1	0	0
O_6	1	1	0	0	0	0	1	1	0	0

2.物件內聚度及物件集合內聚度

為了解決具有“高維稀疏資料”的資料預聚類問題，我們首先提出物件內聚度和物件集合內聚度兩個關鍵概念。

2.1 對象內聚度

對象內聚度（Object Cohesion）：假設有 n 個物件，記為 O={O1，O2，…，On}，描述每個物件的屬性（Attribute）有 m 個，記為 A={A1，A2，…，Am}。物件 Oi（i=1，2，…，n）的非零屬性集合（即物件屬性特徵值取 1 的屬性集合）為 OAi（i=1，2，…，n），其中的屬性個數記為|OAi|（i=1，2，…，n），則兩個物件 Oi 和 Oj 的內聚度定義為：

$$OC(O_i,O_j)=\frac{|OA_i\cap OA_j|}{|OA_i\cup OA_j|} \qquad (2)$$

（i，j=1，2，…，n　且 i≠j）

當且僅當 i=j 時，$OC(O_i,O_j)=1$，即物件和其本身的內聚度為 1。

物件內聚度反映的是兩個物件間的屬性取值的相似程度。例如表一中，先將所有屬性編號，分別記為 1，2，…，10，其中 O1={1，3，4，6，7，10}，O2={3，4，9，10}，則對象 O1 和 O2 的內聚度為：

$$OC(O_1,O_2)=\frac{|OA_1\cap OA_2|}{|OA_1\cup OA_2|}=3/7=42.85\%$$

定理 1：$dis(O_x,O_y)=1-OC(O_x,O_y)$，其滿足：

（1）$dis(O_x,O_x)=0$；

（2）$dis(O_x,O_y)=dis(O_y,O_x)$；

（3）$dis(O_x,O_y)+dis(O_y,O_z)-dis(O_x,O_z)\geq 0$；

即上述定義滿足距離的三個基本性質。

2.2 物件集合內聚度

物件集合內聚度（Object Aggregate Cohesion）：設集合 X 和集合 Y 為 O 中不相交的物件子集，則集合 X 和集合 Y 的物件集合內聚度定義為集合 X 中物件和集合 Y 中物件的物件內聚度的最大值，即：

$$OAC(X,Y)=\max_{O_i\in X,O_j\in Y}\{OC(O_i,O_j)\}\quad (i\neq j) \qquad (3)$$

例如表 1 中，假設 X={O1，O2 }，Y={O4，O5 }，其中 O1={1，3，4，6，7，10}，O2={3，4，9，10}，O4={1，2，3，4，8，10}，O5={1，4，5，8}，則物件集合 X 和物件集合 X 的物件集合內聚度為：

$$OAC(X,Y)=\max\{OC(O_1,O_4),OC(O_2,O_4),OC(O_1,O_5),OC(O_2,O_5)\}$$
$$=\max\{4/8A3/7A2/8A1/7\}=50\%$$

物件集合的內聚度度量了集合中所有物件由於屬性取值的內聚性。內聚度越高說明集合內或集合間的物件越相似內聚性越強，越有可能被聚為一類，反之物件越不相似，越不可能被聚為一類。

3.演算法描述

3.1 對象內聚度

物件—屬性空間分割的目的是在正式的資料挖掘前對規模龐大的資料從資料規模和屬性維度兩個方面進行精簡，使經典的資料挖掘演算法能夠得到較好的挖掘結果。

物件—空間分割技術是利用了層次聚類演算法中聚結型層次聚類法（Agglomerative Hierarchical Clustering）的思想，其基本思路是：以物件的非零屬性是否相似作為判斷物件間是否相似的依據，將非零屬性近似的物件歸為一類，作為後續資料挖掘的物件，判斷標準就是前一節提到的物件集合內聚度。

上述物件——空間分割方法的具體的演算法描述為：

聚類的每一個物件建立一個集合，形成 n 個集合。首先創建類 1，將物件 1 歸入類 1，然後計算物件 2 與物件 1 的內聚度，如果內聚度達到預先設定的值α，則將物件 2 亦歸入類 1；如果物件 2 與物件 1 的內聚度未達到預先設定的值α，則創建新類 2，將對象 2 歸入類 2 中。然後計算物件 3 與類 1 中的各個物件的內聚度，根據預先設定的值α判斷是否能歸入類 1，如果能歸入類 1 則繼續根據以上規則判斷其他物件，如果不能歸入類 1 則計算物件 3 與類 2 中的物件的內聚度，也根據預先設定的值α判斷是否能歸入類 2，如果能歸入類 2 則繼續判斷其他物件，如果不能歸入類 2 則創建新類 3，將對象 3 歸入類 3。依此類推，對於掃描到的每一個物件，或將其歸入已存在的類中，或由其建立新的類，直到所有物件掃描結束。再針對第一遍掃描後形成的類按照以上規則進行第二遍掃描，依此類推，直到形成的類不再變化為止。

對形成的各個類進行屬性的約減，首先針對類 1 中的第一個屬性 Ai1，計算類 1 中 Ai1 取值不為零的對象個數 N1，將 N1 與類 1 中物件總數 N 進行比較，如果 N1：N 小於預先設定的值β，則在類 1 中刪除屬性 Ai1，否則保留屬性 Ai1。依此類推，對形成的所有類中的所用屬性進行上述操作。

3.2 演算法步驟

Step1.由每一個物件建立一個集合，分別記為 ，i∈{1，2，…，n}

Step2.計算 X1(0)和 X2(0)的內聚度 OAC（X1(0)，X2(0)），如果 OAC（X1(0)，X2(0)）≥α，則合併 X1(0)和 X2(0)，合併後的類記作 X1(1)，如果 OAC（X1(0)，X2(0)）＜α，則分別記 X1(0)和 X2(0)為類 X1(1)和類 X2(1)。將類的個數記作 c。

Step3.計算 X3(0)和 Xc(1)的內聚度，計算方法採用公式（3-2）為：

$$OAC(X_3^{(0)}, X_c^{(1)}) = \max_{X_k^{(0)} \in X_c^{(1)}} \{OAC(X_3^{(0)}, X_k^{(0)})\}$$

如果 OAC（X1(0)，Xc(1)）≥α，則合併 X3(0)和類 Xc(1)，合併後的類仍記作 Xc(1)；如果 OAC（X3(0)，Xc(1)）＜α，則將 X3(0)作為一個新的類，記為 Xc+1(1)，類的個數 c=c+1。

Step4.對 $X_i^{(0)}$，i∈{4，5，…，n}，依次進行步驟 3 的操作。

Step5.將 $X_c^{(1)}$ 看作新的待聚類物件，進行步驟 2 至步驟 5 的操作。

Step6.直到形成的類不再有變化為止。

Step7.對形成的各類，在類內部進行屬性約減。約減的方法為：在該類中如果某屬性有取值的次數占該類物件總數的比例小於β，則該屬性被認為在以後的資料挖掘中影響較小，在類中將該屬性剔除。

3.3 演算法示例

表 2 是待聚類的 30 個物件 45 個屬性取值的情況，下面就以這 30 個資料為例，手工解析上述演算法。30 個物件的稀疏特徵表見表 2。

表 2　30 個物件 45 個屬性的取值情況

對象序號	取值為 1 的屬性序號集
1	2，3，4，6，12，23，25，26，30，32，45
2	5，16，19，24，27，33，38，44
3	4，6，7，13，15，25，26，28，35，45
4	1，3，9，10，15，22，29，37
5	3，8，9，18，22，34，35，37，42
6	11，16，21，27，31，33，38，41
7	5，11，19，21，24，31，38，44
8	1，3，8，9，10，15，18，22，29，34，35，37，42
9	1，9，10，15，22，29，34，35，42
10	2，3，6，7，13，23，28，30，32，35，45
11	11，19，21，24，27，33，38，41，44

12	3，4，6，12，15，23，25，26，32，35，45
13	2，4，7，12，13，25，26，28，30，32，45
14	5，19，24，31，33，38，41
15	8，10，15，18，29，35，37
16	2，4，6，7，12，15，23，28，30，32
17	10，13，15，17，36，40，43
18	4，7，12，13，23，25，26，30，32，35
19	11，16，21，24，27，31，41，44
20	1，3，8，10，22，29，34，35，37
21	5，11，19，24，31，33，41，44
22	2，3，6，12，13，15，25，26，28，35
23	3，8，9，15，18，22，34，35，42
24	4，8，16，18，23，38，39，42，
25	4，7，12，13，15，23，26，30，32，35，45
26	1，3，9，18，29，34，35，42
27	2，3，6，12，23，25，28，30，32，35，45
28	1，8，10，15，18，29，37，42
29	9，15，18，22，34，35，42
30	4，7，12，15，23，26，30，32，35

步驟 1：由每個物件建立一個集合，分別記為 $\boldsymbol{X}_i^{(0)}$，i∈{1，2，…，30}；

步驟 2：X1(0)和 X2(0)的內聚度為 OAC（X1(0)，X2(0)）=0＜50%，不能合併，則 X1(0)記為 X1(1)，則 X2(0)記為 X2(1)，c=2；

步驟 3：X3(0)和 X1(1)的內聚度為 OAC（X1(0)，X3(0)）=5/16＜50%，不能合併，X3(0)和 X2(1)的內聚度為 OAC（X2(0)，X3(0)）=0＜50%，不能合併，則 X3(0)記為 X3(1)，c=3；

步驟 4：同步驟三的方法計算下去，X4(0)記為 X4(1)，X5(0)記為 X5(1)，X6(0)記為 X6(1)，X7(0)記為 X7(1)，c=7，X8(0)和 X1(1)的內聚度為 OAC（X1(0)，X8(0)）=1/23＜50%，不能合併，X8(0)和 X2(1) 的內聚度為 OAC（X2(0)，X8(0)）=0＜50%，不能合併，X8(0)和 X3(1) 的內聚度為 OAC（X3(0)，X8(0)）=2/21＜50%，不能合併，X8(0)和 X4(1) 的內聚度為 OAC（X4(0)，X8(0)）=8/13＞50%，合併後仍記為 X4(1) ，則 X4(1)={X4(0)，X8(0)}，c=7

同理，計算下去，第一次聚類結果為 10 個類，分別為：

X1(1)={X1(0) ， X12(0) ， X13(0) ， X16(0) ， X18(0) ， X22(0) ， X25(0) ，X27(0) ，X30(0)}；

X2(1)={X2(0) ， X11(0) ， X14(0) ， X21(0) }；

X3(1)={X3(0) }；

X4(1)={X4(0)，X8(0) ，X9(0) ，X15(0) ， X20(0) ， X23(0) ， X26(0) ， X28(0) ，X29(0)}；

X5(1)={X5(0) }；

X6(1)={X6(0)，X19(0) }；

X7(1)={X7(0) }；

X8(1)={X10(0) }；

X9(1)={X17(0) }；

X10(1)={X24(0) }；

步驟 5：針對 X1(1)，X2(1)，X3(1)，X4(1)，X5(1)，X6(1)，X7(1)，X8(1)，X9(1)，X10(1)進行二次聚類，重複上述步驟 2 到步驟 5 的操作。

X1(1)和 X2(1)的內聚度為 max{OAC（X1(0)，X2(0)），OAC（X1(0)，X11(0)），OAC（X1(0)，X14(0)），……，OAC（X30(0)，X3(0)）}= OAC（X8(0)，X22(0)）=3/20＜60%，不能合併，則 X1(1)記為 X1(2)，X2(1)記為 X2(2)，X1(2)={ X1(0)，X12(0) ，X13(0) ，X16(0) ，X18(0) ，X22(0) ，X25(0) ，X27(0) ，X30(0))}，X2(2)={ X2(0)，X11(0) ，X14(0)，X21(0) }，c=2，X3(1)和 X1(2)的內聚度為 max{OAC（X1(0)，X3(0)），OAC（X12(0)，X3(0)），……，OAC（X30(0)，X21(0)）}= OAC（X3 (0)，X22(0)）=7/13＞50%，合併後仍記為 X1(2) ，則 X1(2)={ X1(0)，X3(0)，X12(0) ，X13(0) ，X16(0) ，X18(0) ，X22(0) ，X25(0) ，X27(0) ，X30(0)}，c=2，

同理，計算下去，第二次聚類結果為 5 個類，分別為：

X1(2)={X1(0)，X3(0)，X10(0)，X12(0)，X13(0)，X16(0，X18(0)，X22(0)，X25(0)，X27(0)，X30(0)}；

X2(2)={X2(0)， X6(0)， X7(0， X11(0)， X14(0)，X19(0)，X21(0) }；

X3(2)={ X4(0)，X5(0)，X8(0)，X9(0)，X15(0) ， X20(0) ， X23(0) ，

X26(0)，X28(0)，X29(0) }；

X4(2)={ X17(0) }；

X5(2)={ X24(0) }；

此後聚類結果不再變化。則最終的類如下所示，表 4 展示了經過上述演算法預聚類後的結果。

類 1：{X1(0)，X3(0)，X10(0)，X12(0)，X13(0)，X16(0，X18(0)，X22(0)，X25(0)，X27(0)，X30(0)}

類 2：{X2(0)，X6(0)，X7(0，X11(0)，X14(0)，X19(0)，X21(0) }；

類 3：{ X4(0)，X5(0)，X8(0)，X9(0)，X15(0) ， X20(0) ， X23(0) ，X26(0)，X28(0)，X29(0) }；

類 4：{ X17(0) }；

類 5：{ X24(0) }；

經過預聚類後，資料的規模明顯減小，維度明顯降低，此後的資料挖掘演算法只需在類 1，類 2，類 3 中進行即可，類 4 和類 5 可以作為孤立點單獨研究。

上述預聚類演算法的優點是：（1）與資料的輸入先後沒有關係，即資料登錄順序不影響演算法的結果；（2）可以在預聚類時排除孤立點；（3）對“高維稀疏資料”的降維作用明顯。上述預聚類演算法的不足是：計算複雜度高，計算時間久，在預聚類時需要相當的時間作為聚類品質的代價。

參考文獻

[1] Love Ekenberg, Mats Danielson and Magnus Boman, Imposing Security Constraints on agent_based decision support. Decision Support Systems,No.20, pp.3-15, 1997.

[2] Hosking J, Pednault E, Sudan M, A Statistical Perspective on Data Mining. Future Generation Computer Systems: Special issue on Data Mining, pp. 117-134, 1997.

[3] 武森，高學東，M. Bastian . ***高維稀疏聚類知識發現***，冶金工業出版社，2003。

[定理 1 證明]：

（1）$dis(O_x,O_x)=1-OC(O_x,O_x)=1-\frac{|OA_x\cap OA_x|}{|OA_x\cup OA_x|}=1-\frac{|OA_x|}{|OA_x|}=1-1=0$。

（2）$dis(O_x,O_y)=1-OC(O_x,O_y)$，$dis(O_y,O_x)=1-OC(O_y,O_x)$

因為 $OC(O_x,O_y)=\frac{|OA_x\cap OA_y|}{|OA_x\cup OA_y|}=\frac{|OA_y\cap OA_x|}{|OA_y\cup OA_x|}=OC(O_y,O_x)$

所以有：$dis(O_x,O_y)=dis(O_y,O_x)$。

（3）$dis(O_x,O_y)+dis(O_y,O_z)-dis(O_x,O_z)$

$$=1-OAC(O_x,O_y)+1-OAC(O_y,O_z)-[1-OAC(O_x,O_z)]$$
$$=1-[OC(O_x,O_y)+OC(O_y,O_z)-OC(O_x,O_z)]$$
$$=1-\left[\frac{|OA_x\cap OA_y|}{|OA_x\cup OA_y|}+\frac{|OA_y\cap OA_z|}{|OA_y\cup OA_z|}-\frac{|OA_x\cap OA_z|}{|OA_x\cup OA_z|}\right]$$
$$\geq 1-\left[\frac{|OA_x|}{|OA_x|}+\frac{|OA_z|}{|OA_z|}-\frac{|OA_x\cap OA_z|}{|OA_x\cup OA_z|}\right]$$
$$=1-\left[1+1-\frac{|OA_x\cap OA_z|}{|OA_x\cup OA_z|}\right]$$

當 $\frac{|OA_x\cap OA_y|}{|OA_x\cup OA_y|}=\frac{|OA_x|}{|OA_x|}$ 時，$OA_x=OA_y$，

當 $\frac{|OA_y\cap OA_z|}{|OA_y\cup OA_z|}=\frac{|OA_z|}{|OA_z|}$ 時，$OA_y=OA_z$

所以，當 $\frac{|OA_x\cap OA_y|}{|OA_x\cup OA_y|}=\frac{|OA_x|}{|OA_x|}$，$\frac{|OA_y\cap OA_z|}{|OA_y\cup OA_z|}=\frac{|OA_z|}{|OA_z|}$ 時，$OA_x=OA_z$，

$$\frac{|OA_x\cap OA_z|}{|OA_x\cup OA_z|}=\frac{|OA_x|}{|OA_x|}=\frac{|OA_z|}{|OA_z|}=1$$

總結上述，有 $1-\left[1+1-\frac{|OA_x\cap OA_z|}{|OA_x\cup OA_z|}\right]=0$

所以 $dis(O_x,O_y)+dis(O_y,O_z)-dis(O_x,O_z)\geq 0$。定理證畢。