结合限制的分隔模型及 *K*-Means 算法*

何振峰[1,2+], 熊范纶[2]

[1](中国科学技术大学 自动化系,安徽 合肥 230027)
[2](中国科学院 合肥智能机械研究所,安徽 合肥 230031)

A Constrained Partition Model and *K*-Means Algorithm

HE Zhen-Feng[1,2+], XIONG Fan-Lun[2]

[1](Department of Automation, University of Science and Technology of China, Hefei 230027, China)
[2](Hefei Institute of Intelligent Machines, The Chinese Academy of Sciences, Hefei 230031, China)
+ Corresponding author: Phn: +86-551-3430270, E-mail: hezhenfeng@yahoo.com.cn, http://www.ustc.edu.cn
Received 2004-01-09; Accepted 2004-03-17

He ZF, Xiong FL. A constrained partition model and *K*-means algorithm. *Journal of Software*, 2005,16(5): 799−809. DOI: 10.1360/jos160799

Abstract: Incorporating instance-level constraints into *K*-means algorithm can improve the accuracy of clustering. As the partition generated is represented by *K* centers and a cluster is represented by only one center, the representation model prevents further improvement of the accuracy. Based upon the instance-level constraints, two types of constraints between instance and class are presented, three types of constraints between classes are presented too, and the constrained partition model is presented and analyzed. In this model, based upon the constraints between sub-clusters, more centers are utilized to represent one cluster, which makes the representation of partition flexible and precise. An algorithm CKS (constrained *K*-means with subsets) is presented to generate the constrained partition. The experiments on three UCI datasets: Glass, Iris and Sonar, suggest that CKS is remarkably superior to COP-*K*-means in accuracy and robustness, and is better than CCL too. The time for running CKS is neither significantly influenced by the number of constraints compared with COP-*K*-means, nor remarkably increased when the number of instances is increased compared with CCL.
Key words: clustering analysis; constrained clustering; semi-supervised learning; background knowledge; machine learning

摘 要: 将数据对象间的关联限制与 *K*-means 算法结合可以取得较好的效果,但由于划分是由 *K* 个中心决定的,每一类仅由一个中心决定,分隔的表示方法限制了算法效果的进一步提高.基于数据对象间的两类限制,定义了数据对象和集合间的两类关联,以及集合间的 3 类关联,在此基础上给出了结合限制的分隔模型.在模型中,基于集合间的正关联,多个子集中心可以用来表示同一类,使划分的表示可以更为灵活、精细.基于此模型,给出了

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2002AA243031 (国家高技术研究发展计划(863))

作者简介: **何振峰**(1971−),男,安徽石台人,讲师,主要研究领域为机器学习,知识发现;**熊范纶**(1940−),男,教授,博士生导师,主要研究领域为人工智能,模式识别,机器学习,农业信息处理.

相应的算法 CKS(constrained *K*-means with subsets)来生成结合限制的分隔.对 3 个 UCI 数据集的实验结果显示:在准确率及健壮性上,CKS 显著优于另一个结合关联限制的 *K*-means 类算法 COP-*K*-means,与另一个代表性的算法 CCL 相比,也有相当优势;在时间代价上,CKS 也有一定优势.

关键词: 聚类分析;限制聚类;半监督学习;背景知识;机器学习

中图法分类号: TP311 **文献标识码**: A

聚类在模式识别、决策支持、机器学习、图像分隔等领域有广泛应用,是最重要的数据分析方式之一.它将数据对象分入不同的集合,使得同一集合中的数据对象相似,而不同集合的数据对象相对来说有较大差别.在聚类过程中通常没有教师指导,是一种无监督的学习.随着研究的深入,主观因素的重要性逐渐为人们所认识."对于不同的应用,其相应的聚类结果应该不同,如对金枪鱼、鲸和大象进行聚类,根据它们的相似性,鲸和大象也许会因为都是哺乳动物而分入一类,可是若用户的兴趣是基于'是否生活在水中'这一特征,则鲸和金枪鱼应分入一类"[1].如何把用户倾向结合入聚类过程成为一个具有挑战性的问题.

用户倾向实际上就是背景知识,Wagstaff 研究了一类特殊的背景知识:数据对象间的关联限制,并提出了 3 种新的算法,其中基于软关联的 *K*-means 算法 SCOP-*K*-means 被成功用于分析火星遥感数据[2].由于数据对象间的关联限制在实际问题中经常会遇到,如何有效地利用它们引起了极大的关注.本文第 1 节回顾关联限制的定义及当前研究进展.第 2 节基于两种限制提出一个新的分隔模型.第 3 节给出一个相应的算法 CKS(constrained *K*-means with subsets).第 4 节给出 CKS 与相关算法的实验结果比较.第 5 节在分析基于两类关联限制的算法在执行时可能遇到的问题之后,分析了 CKS 的收敛性及时间复杂性.第 6 节作简单总结.

1 相关工作

1.1 数据对象间的两种关联限制

在聚类分析时,可能需要对聚类结果做一些限制:某两个数据对象应该分入一个集合,或某两个数据对象不应分入一个集合.为描述数据对象间的这类关系,Wagstaff 引入了 Must-link(正关联)和 Cannot-link(负关联)两种限制[3],并分析了两类限制的一些性质:

Must-link 和 Cannot-link 都是布尔函数.给定数据对象集 S,已知 $P,Q\in S$,若 P 和 Q 应当在同一类中,则 Must-link(P,Q)=True;若 P 和 Q 不应在同一类中,则 Cannot-link(P,Q)=True.显然:

(1) Must-link 和 Cannot-link 具有对称性,对 $P,Q\in S$:

Must-link(P,Q)$\Leftrightarrow$Must-link(Q,P)

Cannot-link(P,Q)$\Leftrightarrow$Cannot-link(Q,P)

(2) Must-link 和 Cannot-link 具有有限的传递性,对 $P,Q,R\in S$,

Must-link(P,Q) && Must-link(Q,R)$\Rightarrow$Must-link(P,R)

Must-link(P,Q) && Cannot-link(Q,R)$\Rightarrow$Cannot-link(P,R)

现有结合两类关联限制的算法在利用限制之前,均依据以上性质来扩充限制集.这样,算法在执行时所加的限制数常会多于最初提供的限制数.

1.2 基于数据对象间限制的聚类算法

结合两类限制,引入了很多算法,有 SCOP-*K*-means[2],COP-COBWEB[3],PC-*K*-means[4],COP-*K*-means[5],CCL[7]等,在聚类分析过程中,都取得了一定的效果.其中有代表性的有 COP-*K*-means(以下简称为 CKM)和 CCL.

CKM 是对 *K*-means 算法的改进,通过强制性地将一些数据对象归入(或不归入)同一组,来直接提高准确率,同时,一些数据对象分类的改变会影响类中心的位置,进而影响其他数据对象的分类,以提升整体的准确率.

CCL 是对 Complete-link 算法的改进,在算出数据对象间的距离后,再依据限制信息来调整距离矩阵.先将正关联的数据对象间距离置 0;然后进一步修改距离矩阵:若 Distance(A,B)+Distance(A,C)<Distance(B,C),则置

Distance(B,C)的值为 Distance(A,B)+Distance(A,C),其中 A,B,C 为数据对象,Distance 函数为两个数据对象间的距离,这样可扩大正关联的影响.最后将负关联的数据对象间的距离设为无穷大.距离矩阵经调整后,正关联的数据对象间距离为 0,而负关联的数据对象间距离为无穷大,在其后的聚类过程可基本满足关联限制.由于正关联的影响在调整距离矩阵时得到了扩展,准确率能够进一步提高.

尽管 CKM 和 CCL 较其所改进的算法,可得到更高的准确率,但还是被原先算法的主要限制所困扰.CCL 改进的是相似度的计算方法,但它与 Complete-Link 算法相似,孤立点的影响在限制不多时,仍然明显;由于 CCL 在修改距离矩阵后又未能有效利用关联限制,孤立点的影响有时在限制数较多时仍很显著.CKM 实际上是对 K-means 的搜索策略进行了改进,它相对不太受孤立点的影响,但问题依然是划分的表示模型(SCOP-K-means 和 PC-K-means 也存在同样问题):聚类结果仍由 K 个中心来决定(若不考虑给定关联限制的数据对象),每个类由一个中心代表,限制对整体效果的影响需要通过 K 个中心的改变来完成,所给限制的放大作用有限.如果能用更多的中心来表示一个类,则结果可能更为精确.而对于基于两类限制的聚类来说,正关联也为使用更多的中心来表示一个类提供了可能.

2 结合限制的分隔模型

设有数据对象 $D_0,D_1,\ldots,D_{N-1}$,要将其分入 K 类 $C_0,C_1,\ldots,C_{K-1}$,对于 K-means 算法来说,就是寻找 K 个中心 $E_0,E_1,\ldots,E_{K-1}$,每个中心代表一类,再将每个数据对象分入与其最近中心所代表的类中,为了使各数据对象与其所在类中心距离的平方和较小,这个过程可能会执行多次.图 1 为对一组数据对象当 K=3 时,进行 K-means 聚类的结果示意图.显然,使用经典的 K-means,无论 3 个中心取在何处,左下角的点难以与右上角的点分入一类,尽管主观愿意将其分入一类.

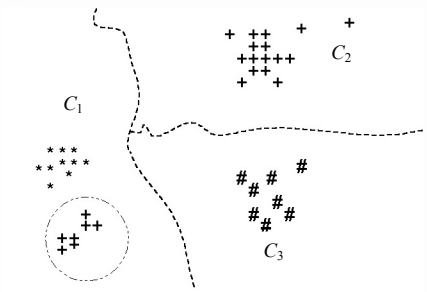

Fig.1 The partition generated by K-means
图 1 用 K-means 聚类的结果

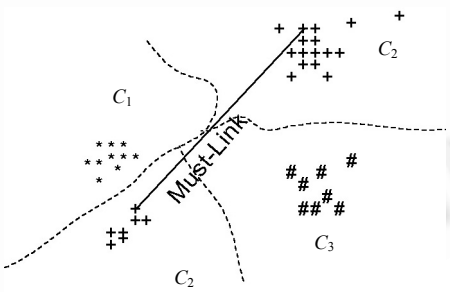

Fig.2 Constrained partition model
图 2 结合限制的分隔模型

图 2 给出了结合限制的分隔模型,看起来二维图像被分成了 4 类,但正关联将右上角和左下角的两类合为一类,总体上仍是 3 类,其中类 C_2 由两个子集构成.

为了描述图 2 的分隔模型,这里首先给出数据对象与集合间正关联及负关联的定义,再给出集合间的直接正关联、正关联及负关联的定义,最后给出结合限制的分隔模型.

定义 1. 给定数据集 S,对于 S 的某一个子集 C_1 和 S 中的一个数据对象 E,可定义数据对象 E 与集合 C_1 间的正关联 Must-link 及负关联 Cannot-link 如下:

(1) 若$\exists D_1\in C_1$,使 Must-link(D_1,E)=true,则称集合 C_1 与数据对象 E 正关联,记为 Must-link(E,C_1)=true.

(2) 若$\exists D_1\in C_1$,使 Cannot-link(D_1,E)=true,则称集合 C_1 与数据对象 E 负关联,记为 Cannot-link(E,C_1)=true.

定义 2. 给定数据集 S,对于 S 的子集 C_1,C_2,C_3,可定义集合间的直接正关联 Must-link$D$,负关联 Cannot-link 和正关联 Must-link 如下:

(1) 若$\exists D_1\in C_1,\exists D_2\in C_2$,Must-link($D_1$,$D_2$)=true,则称集合 C_1 与 C_2 直接正关联,记为 Must-link$D$(C_1,C_2)=true;

(2) 若 Must-link$D$(C_1,C_2)=true,则称集合 C_1 和 C_2 正关联,记为 Must-link(C_1,C_2)=true;

(3) 若 Must-link(C_1,C_2)=true,Must-link(C_2,C_3)=true,则称集合 C_1 与 C_3 正关联,记为 Must-link(C_1,C_3)=true;

(4) 若$\exists D_1\in C_1$,$\exists D_2\in C_2$,Cannot-link(D_1,D_2)=true,则称集合 C_1 与 C_2 负关联,记为 Cannot-link(C_1,C_2)=true.

显然,3 种集合间的关联限制均是对称的.另外,Must-link 是传递的,而 Must-link$D$ 不是.

定义 3(结合限制的分隔模型). 给定数据集 $S=\{D_1,D_2,\ldots,D_N\}$,限制集 $S_C=S_{\text{Must}}\cup S_{\text{Not}}$,其中 $S_{\text{Must}}=\{(D_i,D_j)|\text{Must-link}(D_i,D_j)=\text{true}\}$;$S_{\text{Not}}=\{(D_i,D_j)|\text{Cannot-link}(D_i,D_j)=\text{true}\}$,现将其划分成 K 组 $C_1,C_2,\ldots,C_K$,每组 C_i 进一步划分为 M_i 个子集($M_i\in N$);子集 $C_{i,j}$ 是组 i 的第 j 个子集,它的中心记为 $E_{i,j}$,所有子集中心的集合记为 E,即 $E=\{E_{1,0},E_{1,1},\ldots,E_{1,M1-1},E_{2,0},\ldots,E_{K,Mk-1}\}$.

若所得分隔满足以下条件:

(1) $\bigcup_{i=1}^{k}C_i=S$.

(2) 对任意 $1<=i<j<=K$,有 $C_i\cap C_j=\varnothing$.

(3) 以$|C|$表示集合 C 中的数据对象个数,若集合 C_i 由 t 个子集 $C_{i,0},C_{i,1},\ldots,C_{i,t-1}$ 构成,且 $t>=2$,则对于 $1<=m<t$,有$|C_{i,0}|>=|C_{i,m}|$成立.每个集合的第 0 个子集称为主子集.

(4) 若集合 C_i 由 t 个子集 $C_{i,0},C_{i,1},\ldots,C_{i,t-1}$ 构成,且 $t>=2$,则对于 $0<=m<n<t$,有 Must-link($C_{i,m}$,$C_{i,n}$)=true.

(5) 若集合 C_i 由 t 个子集 $C_{i,0},C_{i,1},\ldots,C_{i,t-1}$ 构成,且 $t>=2$,则不存在 $0<=m<n<t$,使 Cannot-link($C_{i,m}$,$C_{i,n}$)=true.

(6) 若集合 C_i 由 t 个子集 $C_{i,0},C_{i,1},\ldots,C_{i,t-1}$ 构成,且 $t>=2$,则对于 $0<m<t$,有 Must-link$D$($C_{i,0}$,$C_{i,m}$)=true.

(7) 已知 $X\in C_{i,j}$,则对$\forall E_{m,n}\in E$:有 Distance(X,$E_{i,j}$)<=Distance(X,$E_{m,n}$)成立,(Distance(X,$E_{i,j}$)为 X 与 $E_{i,j}$ 之间的距离).

完全满足以上 7 个条件的分隔方案,称为结合限制的分隔模型.满足条件 1~条件 4 的称为部分结合限制的分隔模型.结合限制的分隔模型与部分结合限制的分隔模型不同之处在于:是否必须满足集合间的负关联限制;是否要求主子集与其他子集直接正关联;是否要求每个数据对象与其所属子集合的中心距离不小于与其他子集合中心的距离.主子集与其他子集直接正关联(条件 6)只有在限制数相对很多时,才可能做到,但尽可能满足对改善算法的执行效果有很大帮助(说明如图 3 所示).集合间的负关联限制(条件 5)应该满足,但在算法执行过程中,常会遇到在第 5 节将要讨论的插入矛盾问题,强制满足这一条件可能会极大地影响算法的效率(这一点在实验中得到证明),故此要求只有放宽.

以下给出寻找较好的结合子集的分隔方案的算法 CKS.CKS 也追求各数据对象与其对应子集中心距离平方和的最小化,但同时也能保证每一次迭代过程结束时的分隔均是部分结合限制的分隔,并且第 1 个集合之外的其他集合,都符合条件 5 和条件 7.CKS 也考虑到使尽量多的数据对象在满足条件 6 的子集中.

3 结合子集的 *K*-means 算法 CKS

CKS 算法的输入为数据集 S,总记录数 *RowCount*,限制集 S_C,最大循环次数 *Loops*,返回的是部分结合限制的分隔(以上 5 个参数均为全程变量,后面的过程参数中不再给出).

(1) CKS 函数

```
CKS()
{
    取前 K 个数据对象作为 K 个集合第 0 个子集的中心,每个集合的子集数设为 1
    LoopCount=0;
    do{
        清空各子集;
        for(i=0;i<RowCount;i++) AddID(i);
        PostProcess();
        ReProcess();
        PostProcess();
```

```
            LoopCount++;
        }while(分隔有变化 && LoopCount<Loops);
    }
```

与 *K*-means 类似,CKS 的执行过程也分为置初始点和循环两部分.不同之处在于:*K*-means 为 *K* 个集合置中心点,而 CKS 置的是 *K* 个集合的第 0 个子集的中心,同时还置各个集合的子集个数为 1.

在循环时,两者主要都是执行加入数据对象及后续处理两步操作.每次循环中,*K*-means 会把各个数据对象依次加入 *K* 个集合中的某一个,这种加入操作只执行一次;后续处理是算出各集合的中心,本工作在加入所有数据对象后执行一次.而对于 CKS 来说,部分数据对象可能进行两次加入操作(第 2 次加入操作在函数 ReProcess 中执行);CKS 的后续处理(PostProcess 函数)也需执行两次,其功能除计算各子集中心以外,还将含有数据对象最多的子集与第 0 个子集对调,再去除那些空的且不是主子集的子集.函数 ReProcess 承上启下,它选出需要重新处理的子集,将其中的数据对象加入临时集后,去除这些子集,再将临时集中的数据对象重新加入.函数 AddID 具体完成单个数据对象的加入.

(2) AddID 函数

```
    AddID(int RowID)
    {
        GetSubsets(RowID,C_M,D_M,C_C,D_C,C_N,D_N,Sign);
        Op=Judge(C_M,D_M,C_C,D_C,C_N,D_N,Sign)
        Switch(Op)
        {
            Case 1:当前数据对象加入子集 C_M;break;
            Case 2:当前数据对象加入子集 C_N;break;
            Case 3:在 C_M 所属集合中加入一个子集 C_new,将当前数据对象加入 C_new,设 C_new 的中心为当
                   前数据对象;break;
            Case 4:当前数据对象加入第 1 个集合的第 0 个子集;
        }
    }
```

AddID(*RowID*)完成加入序号为 *RowID* 的数据对象的操作.对 *K*-means 来说,加入一个数据对象分为两步,先算出它与各中心的距离,然后加入与其最近的中心所代表的集合.而对 CKS 来说,要执行较多的工作,它可细分成 3 步:先调用 GetSubsets 计算 3 类距离,然后调用 Judge 确定下一步的操作 Op,最后依据 Op 的值进行具体操作.

其中,GetSubsets 函数的输入为 *RowID*,输出为 3 类距离(D_M,D_C,D_N)、相应的子集(C_M,C_C,C_N)和相应的标记 *Sign*.它分析当前数据对象与各子集的关系(Must-link,Cannot-link 或无限制),计算它与各子集中心的距离,找到与其最近的满足 Must-link 限制的子集 C_M 及相应距离 D_M,最近的满足 Cannot-link 限制的子集 C_C 及距离 D_C,以及最近的无关联限制的子集 C_N 及距离 D_N(D_M,D_C 或 D_N 为无穷大时,表示找不到符合相应要求的子集,此时相应的子集为空),C_M 不为空时,*Sign*=1;否则若 C_C 不为空,则 *Sign*=2;若 C_M 和 C_C 均为空,则 *Sign*=3.

Judge 函数的输入为 3 类距离(D_M,D_C,D_N)、相应的子集(C_M,C_C,C_N)和相应的标记 *Sign*,返回值为 1、2、3 或 4,指示应该进行的操作:*Sign*=1 时,若 D_M 的值小于 D_C 和 D_N,则返回 1,表示可以直接加入 C_M;否则说明 C_M 不是较好的选择,此时返回 3,表示需要在 C_M 所在集合中新建一个集合并将其加入;*Sign*=2 时,若 D_N 小于 D_C,则 C_N 是较好选择,此时返回 2,表示可以直接加入 C_N;否则无法找到较好的方案,返回 4,直接加入第 1 个集合的第 0 个子集;*Sign*=3 时,不需要考虑关联限制,可直接加入 C_N,此时返回 2.

(3) PostProcess 函数

```
    PostProcess()
    {
```

```
        for(i=0;i<k;i++)
        {
            计算各子集中元素的个数,
            若子集 C_{i,SubClass} 中元素个数最多,且 SubClass≠0
                交换子集 C_{i,0} 与子集 C_{i,SubClass} 中的元素
            若子集 C_{i,SubClass} 中元素个数为 0,且 SubClass>0
                去除子集 C_{i,SubClass}
            重新计算各子集的中心,
        }
    }
```

PostProcess 函数完成后续处理,它扫描各集合,依次完成 4 项工作:计算各子集中数据对象个数;交换第 0 个子集与含有数据对象最多的子集,以确保主子集中数据对象最多;清除空集(主子集保留);计算各子集的中心.

(4) ReProcess 函数

```
    ReProcess()
    {
    清空临时集合 Temp;
    对每一个集合,考察各主子集 C_S 与主子集 C_0 的关联限制:
        若 Must-linkD(C_S,C_0)=true
            则跳过;
        否则
            将 C_s 中的全部数据对象(即其对应的 RowID)加入临时集 Temp;
            删除子集 C_s;
    对临时集合 Temp 中的所有数据对象 D,依据其 RowID,执行 AddID(RowID);
    }
```

ReProcess 函数分成两步:子集筛选和临时集中数据对象的重新加入.在子集筛选时,考察各非主子集 C_S 与对应主子集 C_0 的关联限制,删除那些不与主子集直接正关联的子集,并将其中数据对象加入临时集.临时集中数据对象的加入就是对临时集中的数据对象再次调用 AddID 函数加入,这样的结果会更加稳定可靠.

图 3 给出了子集的两种关联策略.A,B,C,D 是同一个集合中的子集,A 是主子集.虽然在两图中 A 与其他子集均正关联,但左图中只有 B 与 A 直接正关联,而右图中所有子集与 A 均直接正关联.在左图中,一些小的错误可能对整体效果产生极大影响:如 B 中的一个数据对象是被错误地划入的,可是该数据对象与 C 有正关联,而 C 与 D 又正关联,这样 C 和 D 两集合中的所有数据对象就可能被错误地划入;而右图中类似错误的影响会小得多.因此,尽可能增加主子集以及与其直接正关联的子集中数据对象的数目(满足条件 6),对提高准确率有积极意义.Reprocess 函数的目的就在于为那些既不属于主子集,也不属于与主子集直接正关联的子集中的数据对象(如左图中集合 C 和 D 内的数据对象)提供一次重新选择的机会.

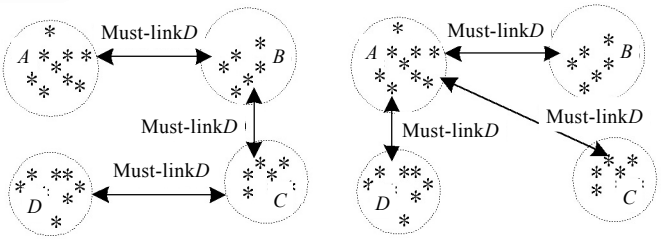

Fig.3 Two approaches to link the subsets
图 3 子集的两种关联策略

4 实验分析

4.1 数据集

实验选择了 3 个 UCI 数据集[6],它们是 Iris,Glass 和 Sonar.3 个数据集中所有属性均为连续数值型.进行聚类前,先将数据零-均值规范化.聚类分析时选用两个 K 值,一个是实际类数,一个则要大一些.对于 Iris 数据集,k 值分别取 3 和 5;对于 Glass 数据集,k 值分别取 6 和 10;对于 Sonar 数据集,k 值分别取 2 和 3.

4.2 限制的发生

由于选用不同的限制,准确率会有差异,因此在限制的选择上必须谨慎、客观.

在设定限制时,采用随机数发生器,由其每次发生一对数字,每一个数字代表数据对象在数据集中的编号(即 *RowID*),这样,一对数字实际上代表着一对数据对象.若此数据对象对未出现在限制集中,则将它加入;反复以上过程,直到限制集中元素数目达到设定值.对于一对数据对象,具体采用 Must-link 或 Cannot-link 关联,是由它们的实际关系确定的,就 Iris 数据集来说,如果两个数据对象的 class 属性取相同的值,则设定它们为 Must-link 关联,否则设为 Cannot-link 关联.在设定限制前,将初始化随机数发生器,每次采用不同的种子来初始化,从而得到不同的限制集.不人为地选择种子,种子由 1 开始递增;如果实验 100 次,则种子为 1~100;如果实验 1000 次,则种子为 1~1000.一般实验 100 次,对于 CKM 算法,在限制数增加后,往往难以找到符合条件的分隔,这时种子数会增加,直至 1000.

4.3 结果评价方式

将聚类结果与真实划分情况比较,对每一个样本对,存在 4 种可能:

(a) 它们应归入一类,在结果中确实将其归入一类.

(b) 它们应归入一类,在结果中却将它们分入不同类.

(c) 它们应属于不同类,在结果中却将它们归入了一类.

(d) 它们应属于不同类,在结果中确实将它们归入不同类.

设满足以上条件的样本对数分别为:a,b,c 和 d,总样本对数为 n.评价标准常采用正确的样本对数与总样本对数之比(Rand 系数): $\frac{a+d}{n\times(n-1)/2}$ [9].本实验中采用修正的 Rand 系数[7]:假设依据已有的关联限制可以推出 C 对数据对象对的关联情况,则准确率为 $\frac{a+d-C}{n\times(n-1)/2-C}$.

4.4 实验结果

从对 3 个 UCI 数据集的处理结果可以看出,无论从绝对的准确率还是从准确率的提高上看,CKS 均明显优于同类算法 CKM.与另一种结合关联限制的算法 CCL 相比,CKS 在准确率及准确率的提高上也有相当优势:对于 Glass 数据集,在限制数不太多时,CCL 的效果随限制数的增加迅速增加,表现优于 CKS,但当限制数达到一定程度时,CKS 会再次超过 CCL;对于 Iris 数据集,在 K=3 时,CCL 呈现为一种波动,在 K=5 时,CCL 从准确率上优于 CKS,但从准确率的提高效果上,CKS 还是优于 CCL 的,总体上说,对于 Iris 数据集,CKS 的表现优于 CCL;对于 Sonar 数据集,在限制数较少时,准确率均接近于 50%,增加的限制提高了各种算法的准确率,但相对来说,CKS 的表现明显优于 CCL,在 K=2 时,CKM 算法在限制数超过 100 时,即使选择 1000 个种子,每次可以重新选择 500 次开始点,仍无法找到符合条件的分隔.从算法的稳定性上看,CKS 的表现也明显优于其他两种算法,对于 3 个数据集,随着限制数的增加,准确率均呈现出稳定上升趋势.实验结果如图 4~图 9 所示.

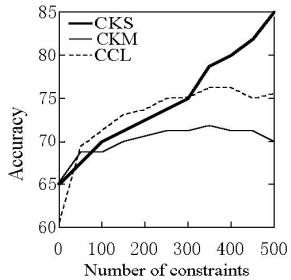

Fig.4 Accuracy on Glass data (K=6)

图 4 处理 Glass 数据集的结果(K=6)

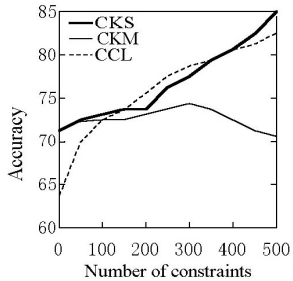

Fig.5 Accuracy on Glass data (K=10)

图 5 处理 Glass 数据集的结果(K=10)

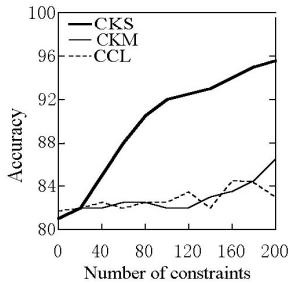

Fig.6 Accuracy on Iris data (K=3)

图 6 处理 Iris 数据集的结果(K=3)

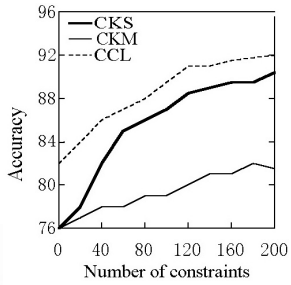

Fig.7 Accuracy on Iris data (K=5)

图 7 处理 Iris 数据集的结果(K=5)

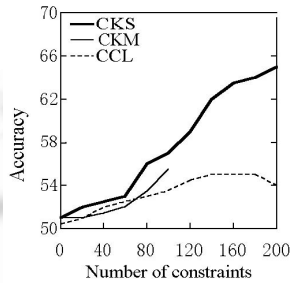

Fig.8 Accuracy on Sonar data (K=2)

图 8 处理 Sonar 数据集的结果(K=2)

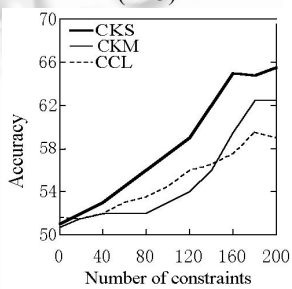

Fig.9 Accuracy on Sonar data (K=3)

图 9 处理 Sonar 数据集的结果(K=3)

5 算法分析

5.1 收敛性分析

5.1.1 结合两类限制的 K-means 类算法在收敛时的主要问题

(1) 插入矛盾

在算法的执行过程中,严格按照关联限制,很可能会出现某一数据对象无法插入任何一个类的情况:若有 3 个数据对象 A,B,C,要将其分入两类中;现有限制 Cannot-link(A,B)=true 和 Cannot-link(B,C)=true.这样,若 A 和 B 先被分别归入不同类中,则 C 无法加入,这种情况可称为插入矛盾.CKM 解决插入矛盾的方法是重新选择起始点.在限制数较多,而可归入的类较少时,选择初始点常需要花费很多时间,整个算法的执行时间往往会大大延长.CKS 解决插入矛盾的方法是将该数据对象插入第 1 类的主子类,希望在迭代过程中自然解决问题,实验结果较好.

(2) 无法终止

在结合两类限制的 K-means 类型算法执行过程中,偶尔会碰到无法终止的情况.在实验中的已出现的形式均为:在两个或两个以上不同的分类方案之间无限循环.以简单的 R^1 空间的一个例子来说明:

图 10 中 A,B,C 三点各表示一个数据对象,坐标(属性值)分别为 0,10,20,S 是一个数据对象集,其中有 N 个(N>100)数据对象 $X_1,X_2,\ldots,X_N$,对$\forall O\in S$,设 O 坐标为 x,总有$|x-8|<0.01$,并且 Cannot-link(O,A)=true;另外还有 Cannot-link(B,C)=true.

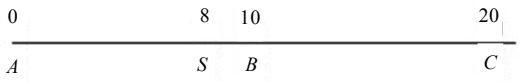

Fig.10 Partition a group of data objects in R^1 space
图 10 分隔 R^1 空间中的一组数据对象

现要将以上数据对象分入两个类,数据对象的加入顺序为 $X_1,A,X_2,\ldots,X_N,B,C$.开始时中心为 X_1 和 A 的坐标.记第 i 次迭代结束时:第 1 个类为 $C_{i,1}$,第 2 个类为 $C_{i,2}$;第 1 个类的中心点坐标为 $\text{Center}_{i,1}$,第 2 个类的中心点坐标为 $\text{Center}_{i,2}$,则:

第 1 次迭代结束时:两个类 $C_{1,1}=S\cup\{B\},C_{1,2}=\{A,C\},|\text{Center}_{1,1}-8|<1,\text{Center}_{1,2}=10$.

第 2 次迭代结束时:两个类 $C_{2,1}=S\cup\{C\},C_{2,2}=\{A,B\},|\text{Center}_{2,1}-8|<1,\text{Center}_{2,2}=5$.

第 3 次迭代结束时:两个类 $C_{3,1}=S\cup\{B\}=C_{1,1},C_{3,2}=\{A,C\}=C_{1,2}$.

第 3 次迭代的结果与第 1 次迭代结束时完全相同.显然,第 4 次迭代的结果将与第 2 次相同.算法无法终止.

对于无法终止情况,CKM 没有考虑[5](在实验中,CKM 也出现了不终止的情况,见表 1,在对比实验时,CKM 规定了最大迭代次数),而 CKS 规定是最大迭代次数,达到这一次数时的结果为最终结果.在实验中,出现无法中止情况时,从准确率的角度分析,多个不停转换的状态其实相差不大,在任何一个状态终止对整体准确率影响较小.

Table 1 The convergence properties of CKS and CKM on Iris data set

表 1 对 Iris 数据集处理时的收敛情况

K	Number of constraints	Convergence	Iterations before convergence	memo
3	10	87(100)	9(7)	
	20	84(98)	13(7)	Twice CKM can not stop
	100	72(34)	29(2)	For 66 times, CKM fail to find appropriate initial centers
5	10	90(100)	9(6)	
	20	84(99)	10(6)	Once CKM can not stop
	100	77(94)	35(6)	For 6 times, CKM fail to find appropriate initial centers

5.1.2 理论上的收敛性分析

基于数据对象间两类关联限制的聚类算法的收敛性至今仍然是一个问题[2],CKS 和 CKM 均不收敛,但在满足很强的条件下,CKM 及 CKS 是可能收敛的.

Seeded-K-means(简称 SKM)[10]是一种基于约束的 K-means 算法,它预先设定 M 个点所属的类,在以后的执行过程中,此 M 个点所属的类不变,其他点在每次循环时,加入中心与其最接近的类,再重新计算中心.可以借鉴[8]来证明此 SKM 与 K-means 一样是局部收敛的,并且其收敛速度也是超线性的.

对于 CKM,设有限制条件指标集 I,即对所有的 $i\in I$,存在 $j\in I$,使得有 Must-link(X_i,X_j)=true 或者 Cannot-link(X_i,X_j)成立.显然,如果一个数据对象的下标不在指标集 I 中,则其聚类时的决策将仅依照其与各类中心的距离来决定.如果从某一步之后,迭代过程中总满足条件 A:

条件 A:所有下标出现在下标集中的数据对象不改变它的分类.

则这些数据对象可以看成种子,此后的整个迭代过程与 SKM 相同,即算法将超线性收敛.

结论 1. 如果在某一步之后,可以保证所有下标出现在限制下标集中的数据对象不改变它们的分类,则 CKM 将超线性收敛.

就 CKS 来说,如果至某一步之后,可以保证出现在限制下标集中的数据对象不改变分类的话,设当前一共有 M 个子类(显然 $M>=k$),则问题可以转化为共有 M 个类的 SKM(区别只是在于对结果的解释,通过 Must-link 关联起来的子类最后将被归入一个大类来加以分析),故其收敛性也可预知.

结论 2. 如果在某一步之后,可以保证所有下标出现在限制下标集中的数据对象不改变它们的分类,则其后 CKS 将超线性收敛.

5.1.3 实验中的收敛情况

表 1 是 CKM 和 CKS 对 Iris 数据集进行处理时的收敛情况,“收敛次数”一列给出两种算法收敛的次数,括号外为 CKS 的收敛次数,括号内为 CKM 的收敛次数.“收敛前迭代次数”给出了两种算法收敛前的迭代次数,括号外为 CKS 算法收敛前迭代次数,括号内为 CKM 收敛前迭代次数.

从收敛次数上看,除了在 K=3,限制数为 100 时,CKS 好于 CKM 以外,其余 CKM 均好于 CKS.尽管在测试时每次最多可 5000 次随机发生种子点,在限制数增加时,CKM 仍有多次找不到种子点,如当限制数为 100 时(分类数为 3),它有 66 次找不到合适的种子点,只能得到 34 个有效分隔.不规定最大迭代次数时,CKM 会出现不终止现象,如在第 2 行,CKM 只有 98 次收敛,余下 2 次不终止.

考察收敛速度(在 K=3 时,K-means 收敛前迭代 9 次,在 K=5 时,K-means 收敛前迭代 5 次),CKM 与 K-means 相当,并且收敛速度随限制数的增加有加快的倾向.CKS 收敛速度要慢于 K-means,随着限制数的增加,收敛速度放慢.这可能是由于大量限制的引入,使得分类更加精细,出现了更多的子集,子集中心的调整可能需要更多次的迭代过程.

5.2 时间复杂性分析

K-means 算法的时间复杂度为 $O(nkl)$,其中 n 为数据对象数,K 为要分入的类数,l 为迭代的次数.CKM 在插入矛盾时,不重新选择初始点,即直接返回的时间复杂度为 $O(nkl+c)$,其中 c 为 Cannot-link 的数目[2].事实上,如果在出现插入矛盾时选择直接返回,则问题仍然没有解决.在实际运用中,时间复杂度应该为 $O((nkl+c)\times R)$,R 为需要选择初始点的次数;如果只考虑计算距离的次数,则时间复杂度为 $O(nklR)$.

CKS 在一次迭代过程中,有两次加入数据对象的过程(第 2 次加入的数据对象会少一些).加入一个数据对象的操作主要有两部分:计算与各子集中心的距离以及判断该数据对象与各子集的关系.若只考虑计算与各子集中心距离的时间,加入一个数据对象的时间复杂度为 $O(T)$,T 为最大的子集数,不大于 n,故加入一个数据对象的时间复杂度为 $O(n)$;每一次迭代最多进行 $2n$ 次数据对象加入操作,则每次迭代的时间复杂度为 $O(2n^2)$.设迭代次数为 l,则总的时间复杂度为 $O(2n^{2l})$.

表 2、表 3 给出了 3 种算法对 Iris 和 Sonar 数据集聚类分析的运行时间(单位:s):它是对 100 组限制测试的总时间,运行环境为 CPU,PIII,1G,128M 内存,CKM 算法每次最多 5000 次重新选择开始点.为了简单起见,CKM 和 CKS 没有判断分隔是否变化,而是限制迭代 100 次.CCL 采用的是较小的 K 值时的运行时间.可以看出:CKM 的运行速度在限制数较少时最快,在限制数很多时,由于严重的插入矛盾,会造成运行速度急剧下降(如 K=3,处理 Iris 数据集时);记录数对于总的时间耗费有显著的影响,相对来说,CCL 受到的影响最大;在限制数不太多时,K 对 CKS 及 CKM 的影响几乎是线性的(如用 CKS 算法处理 Iris 数据集时,当限制数为 10 时,时间分别为 47 和 72,两者比例为 0.65;当限制数为 100 时,分别是 54 和 85,两者的比例为 0.63,这与 3 与 5 的比值都很接近).

Table 2 Running time for CKM, CKS and CCL on Iris data set (150 instances)

表 2 3 种算法的执行时间(Iris dataset,150 条记录)

Number of constraints	CKM (K=3)	CKM (K=5)	CKS (K=3)	CKS (K=5)	CCL
10	10	15	47	72	52
20	10	15	49	74	57
50	53	17	51	78	68
100	576	24	54	85	80

Table 3 Running time for CKM, CKS and CCL on Glass data set (214 instances)

表 3 3 种算法的执行时间(Glass 数据集,214 条记录)

Number of constraints	CKM (K=6)	CKM (K=10)	CKS (K=6)	CKS (K=10)	CCL
10	32	50	83	120	259
20	32	49	87	125	255
50	34	51	103	141	260
100	37	55	124	163	270

6 结束语

随着知识发现技术应用的扩展,结合背景知识的聚类方法引人关注,数据对象间的两类限制由于其广泛的应用前景吸引了众多的注意力.但是,结合关联限制的 *K*-means 类算法也面临着由其表示模型带来的固有问题.不同于以往的工作,本文尝试对分隔模型上作一些改进,给出了一个基于限制的分隔模型.在进行分隔时,可以结合关联限制,引入子集,突破原有模型的局限.同时也给出一个搜索这一分隔模型的算法 CKS,对一些 UCI 数据集的测试结果显示,CKS 可以比 CCL 和 CKM 更好地利用两类限制来提高准确率.

References:

[1] Jain A, Murty M, Flynn P. Data clustering: A review. ACM Computing Surveys, 1999,31(3):264−323.

[2] Wagstaff K. Intelligent clustering with instance-level constraints [Ph.D. Thesis]. Cornell University, 2002.

[3] Wagstaff K, Cardie C. Clustering with instance-level constraints. In: Langley P, ed. Proc. of the 17th Int'l Conf. on Machine Learning (ICML 2000). San Francisco: Morgan Kaufmann Publishers, 2000. 1103−1110.

[4] Basu S, Banerjee A, Mooney R. Comparing and unifying search-based and similarity-based approaches to semi-supervised clustering. In: Fawcett T, Mishra N, eds. Proc. of the 20th Int'l Conf. on Machine Learning (ICML 2003) Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining. New Orleans: AAAI Press, 2003. 42−49.

[5] Wagstaff K, Cardie C, Rogers S, Schroedl S. Constrained *K*-means clustering with background knowledge. In: Brodley C, Danyluk AP, eds. Proc. of the 18th Int'l Conf. on Machine Learning (ICML 2001). San Francisco: Morgan Kaufmann Publishers, 2001. 577−584.

[6] Blake C, Merz J. UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html

[7] Klein D, Kamvar S, Manning C. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In: Sammut C, Hoffmann A, eds. Proc. of the 19th Int'l Conf. on Machine Learning (ICML 2002). San Francisco: Morgan Kaufmann Publishers, 2002. 307−314.

[8] Bottou L, Bengio Y. Convergence properties of the *K*-means algorithm. In: Tesauro G, Touretzky DS, Leen TK, eds. Advances in Neural Information Processing Systems 7. Cambridge: MIT Press, 1995. 585−592.

[9] Halkidi M, Batistakis Y, Vazirgiannis M. Cluster validity methods: Part I. SIGMOD Record, 2002,31(2):40−45.

[10] Basu S, Banerjee A, Mooney R. Semi-Supervised clustering by seeding. In: Sammut C, Hoffmann A, eds. Proc. of the 19th Int'l Conf. on Machine Learning (ICML 2002). San Francisco: Morgan Kaufmann Publishers, 2002. 19−26.