第35卷 第14期 计 算 机 工 程 2009年7月
Vol.35 No.14 Computer Engineering July 2009
·博士论文· 文章编号：1000—3428(2009)14—0004—03 文献标识码：A 中图分类号：TP311

# 基于 $k$ 最相似聚类的子空间聚类算法

单世民，闫 妍，张宪超
(大连理工大学软件学院，大连 116621)

**摘 要：**子空间聚类是聚类研究领域的一个重要分支和研究热点，用于解决高维聚类分析面临的数据稀疏问题。提出一种基于 $k$ 最相似聚类的子空间聚类算法。该算法使用一种聚类间相似度度量方法保留 $k$ 最相似聚类，在不同子空间上采用不同局部密度阈值，通过 $k$ 最相似聚类确定子空间搜索方向。将处理的数据类型扩展到连续型和分类型，可以有效处理高维数据聚类问题。实验结果证明，与 CLIQUE 和 SUBCLU 相比，该算法具有更好的聚类效果。

**关键词：**聚类算法；子空间聚类；高维数据

# Subspace Clustering Algorithm Based on $k$ Most Similar Clustering

SHAN Shi-min, YAN Yan, ZHANG Xian-chao
(School of Software, Dalian University of Technology, Dalian 116621)

**【Abstract】** Subspace clustering is an important part and research hotspot in clustering research, which resolves the problem of clustering sparse data in high dimensional data environment. A subspace clustering algorithm based on $k$ most similar clustering is presented. This algorithm holds the $k$ most similar clustering by the similarity of the clusters, discovers the different subspace through the different local density threshold, ascertains the subspace search direction by the $k$ most similar clustering and clusters both continuous data and categorical data. The high dimensional data can be effectively clustered in this algorithm. Experimental results show that this algorithm is more effective in clustering than CLIQUE and SUBCLU.

**【Key words】** clustering algorithm; subspace clustering; high dimensional data

## 1 概述

高维聚类分析是聚类分析的研究难点[1]，具体表现为：(1)高维数据中存在大量无关属性，导致在全维空间中存在簇的可能性几乎为零；(2)在高维空间中，数据间距几乎相等的情况普遍存在。传统聚类算法在低维数据集上表现良好，但在对高维数据对象聚类时会遇到困难。为解决上述问题，文献[2]首先提出子空间聚类的概念。

CLIQUE算法是最早且最具代表性的子空间聚类算法[3]，它搜索最大的密集超立方体，并以此为基础发现子空间聚类。另一个具有代表性的算法 SUBCLU[4]具有更好的聚类精度，它采用 DBSCAN 算法中的密度连通概念形成子空间聚类。以上 2 个算法通过全局密度阈值确定子空间聚类，如果子空间维数的不同，子空间密集程度也会不同。它们没有考虑数据类型的多样性。用于分类型数据的聚类算法 ROCK[5]虽然取得了较好的聚类结果，但全局方法的使用导致算法复杂度增加。

针对上述算法存在的问题，本文算法的主要工作集中在以下 3 个方面：(1)采用一种聚类间相似度计算方法，通过其确定 $k$ 最相似聚类，缩小搜索空间，提高了算法可伸缩性和执行效率；(2)在子空间聚类搜索过程中摒弃传统算法采用的全局密度阈值，在不同子空间中采用不同密度阈值，提高了子空间聚类算法聚类结果的精度；(3)将子空间聚类算法处理的数据对象扩展为连续型数据和分类型数据。

## 2 问题描述及算法策略

### 2.1 问题描述

设 $A=\{A_1,A_2,\cdots,A_d\}$ 代表数据维集合，$A=A_1\times A_2\times\cdots\times A_d$ 代表一个 $d$ 维数据空间，其中，$d$ 为正整数。设 $DB$ 代表由 $n$ 个位于 $d$ 维特征空间的数据对象组成的集合，记为 $DB=\{X_i|i\in[1,n],x_{ij}=X_i.A_j\}$，其中，点 $X_i=(x_{i1},x_{i2},\cdots,x_{ij},\cdots,x_{id})$，$X_i$ 的第 $j$ 个属性值 $x_{ij}$ 为其在 $A_j$ 维上的取值。

设 $k$ 维子空间 $S_k\subseteq A$，其中，$k\leq n$。在 $S_k$ 子空间中的元组集合记做 $T_{Sk}(x_{i1},x_{i2},\cdots,x_{ik})=\{X_i|X_i\in S_k,x_{ij}=X_i.A_j|j\in[1,k]\}$。将 $S_k$ 子空间投影到每一维产生的不同元组的数目记为 $\pi_{Sk}(DB)$。

### 2.2 算法基本策略

2.2.1 算法策略及相关定义

本文以所有一维空间聚类为基础，通过发现包含相同数据对象的相似聚类来确定子空间搜索方向，进而将不同维度中的相似聚类进行合并，最终发现子空间聚类。

为了处理不同类型的数据，对于连续型数据，采用 DBSCAN 方法在每一维上对所有数据对象进行聚类。对于分类数据，由于其自身的无序性特点，在每一维上将属性值相同的数据对象作为一个自然聚类。

**定义 1**(基本聚类) 将以上方法产生的所有一维空间的聚类的集合称为基本聚类，记为 $C^1$。

**定义 2**(基本聚类相似度) 给定 $c_1,c_2\in C^1$，其中，$c_1$ 在 $A_i$ 维，$c_2$ 在 $A_j$ 维，且 $i\neq j$，其相似度定义为基本聚类 $c_1,c_2$ 所包含数据对象集合的交集的元素数目，记为 $sim(c_1,c_2)=|c_1\cap c_2|$。

**定义 3**(最相似聚类) 给定聚类 $c\in C^1$，$c$ 的最相似聚类 $MSC(c)\in C^1$ 满足条件：$\forall c_p\in C^1, sim(c,MSC(c))\geq sim(c,c_p)$，

**基金项目：**国家自然科学基金资助项目(70671016, 60873180, 60673066)
**作者简介：**单世民(1978－)，男，讲师、博士，主研方向：知识发现；闫 妍，硕士研究生；张宪超，副教授、博士
**收稿日期：**2009-02-25 **E-mail：**ssm@dlut.edu.cn

其中，$c \neq MSC(c)$。

**性质 1** 给定 $c_a \in C^1$，若 $MSC(c_a)=c_b$，则被 $c_a$, $c_b$ 同时包含的数据对象的数目大于等于与 $c_a$ 与基本聚类中除 $c_b$ 外任何基本聚类的相似度。

上述最相似聚类的定义有助于发现大的聚类，但该定义可能忽略较小但有意义的聚类，且由于子空间聚类的特点，一个基本聚类中的数据对象可能被多个子空间聚类包含，因此给出以下定义：

**定义 4**($k$ 最相似聚类) 给定聚类 $c \in C^1$ 和正整数 $k$，$c$ 的第 $k$ 个最相似聚类定义为 $k$ 最相似聚类，记为 $MSC_k(c)$，$c$ 的第 $k$ 个最相似聚类满足：

$\forall\, c_p \in MSC_k(c), \forall\, c_q \in C^1\text{-}MSC_1(c)\text{-}MSC_2(c)\text{-}\cdots\text{-}MSC_k(c)$：$sim(c,c_p)>sim(c,c_q)$。

**性质 2** 给定聚类 $c \in C^1$，设定的正整数 $k$ 越大，$sim(c, MSC_k(c))$越小。

算法通过给定 $k$ 值并保留 $k$ 个最相似聚类，确定子空间聚类的搜索方向。在此过程中，如果采用合适的 $k$ 值，就可以覆盖大部分数据对象，从而确保不会忽略小而有意义的聚类，并缩减搜索空间。

2.2.2 局部密度阈值的选取

给定 $m$ 个不相关属性组成的子空间 $S_m$，在传统算法如 CLIQUE[2]中，本文认为当 $n_m/n>\mu$ 时，子空间 $S_m$ 是密集的，其中，$n_m$ 是 $S_m$ 子空间中包含数据对象的数目，$\mu$ 是用户给定的全局密度阈值。由于子空间聚类的特性，子空间密度随着数据维数的增高而降低，在全局应用相同密度阈值是不合理的，因此本文根据子空间的期望密度为不同的子空间应用不同的密度阈值。

设 $th(S_m)$为 $S_m$ 子空间中的密度阈值，$n_m$ 为 $S_m$ 子空间中的数据对象数目，则对于 $S_m$ 子空间和已合并的 $m$ 个基本聚类 $c \in C^1$，其期望密度为

$$E\left[T_{S_m}(c_1,c_2,...,c_m)\right]=\frac{|DB|}{\pi_{S_m}(DB)} \tag{1}$$

而对于 $S_{m+1}$ 子空间，其期望密度为

$$E\left[T_{S_m}(c_1,c_2,...,c_{m+1})\right]=\frac{|DB|}{\pi_{S_{m+1}}(DB)} \tag{2}$$

本文中使用的 $S_m$ 子空间密度阈值可以根据子空间期望密度获得，如式(3)所示。

$$th(S_m)=r\times E\left[\left|T_{S_m}(c_1,c_2,\cdots,c_m)\right|\right] \tag{3}$$

其中，$r$ 为用户给定值，算法中 $r$ 取值范围为 $0<r<1$。

**定义 5**(子空间合并候选) 给定 $S_m$ 子空间和已合并的 $m$ 个基本聚类 $c_1, c_2,\cdots, c_m \in C^1$，这 $m$ 个基本聚类的 $k$ 最相似聚类的交集为 $S_m$ 子空间的合并候选，记为 $SC_p$。

**定义 6**(子空间可合并聚类) 给定聚类 $c_p$，已知 $S_m$ 子空间和已合并的 $m$ 个基本聚类 $c_1, c_2,\cdots, c_m \in C^1$，$c_p$ 是 $S_m$ 子空间的可合并聚类，当且仅当$|T_{Sm+1}(c_1, c_2,\cdots, c_m, c_p)| \geq th(S_{m+1})$。

## 3 算法描述

### 3.1 本文算法

本文算法步骤如下：

**输入** 数据点集

**输出** 簇

(1)数据预处理，在每一维上形成基本聚类。

(2)计算所有基本聚类间相似度。

(3)计算每个基本聚类的 $k$ 个最相似聚类，作为该基本聚类的合并候选。

(4)搜索子空间产生子空间聚类。

(5)将剩余数据点分配到相应聚类中或认定其为噪声。

### 3.2 子空间聚类

在本文算法第(4)步采用深度优先搜索的方法产生子空间聚类。在搜索过程中，允许对搜索空间做适度裁减。先找到基本聚类 $c \in C^1$，形成 $S_1$ 子空间，计算 $C^1$ 中各个基本聚类间相似度，根据基本聚类间相似度保留每个基本聚类的 $k$ 个最相似聚类，对基本聚类相似度大于等于 $th(S_2)$的基本聚类进行合并，形成 $S_2$ 子空间。然后用递归方式搜索产生子空间聚类。对于某些特定数据集，深度优先策略的使用会导致过多子空间聚类。为了解决该问题，算法增加了子空间合并步骤。子空间搜索算法描述如下：

**输入** 候选子空间 $S_p$, $\delta$, $k$

**输出** 子空间结果集

(1)For each $c_x \in SC_p$;

(2) $S_{p+1} \Leftarrow S_p \cup c_x$;

(3) $PC_{p+1} \Leftarrow \{c_y | c_y \in SC_p, c_y > c_x\}$;

(4) $C_{p+1} \Leftarrow$ can-merge($S_{p+1}$,$PC_{p+1}$, $\delta$);// 根据定义 6 判断是否//为可合并聚类

(5) If $C_{p+1}=\phi$

(6) If $S_{p+1}$ 不是结果集中任意子空间的子集

(7) MS $\Leftarrow$ MS $\cup$ $S_{p+1}$;

(8) End if

(9) Else

(10) 搜索子空间;

(11) End if

(12) End for

(13)子空间合并

(14)返回结果集;

### 3.3 算法评价

本文算法采用自底向上的子空间搜索方法，生成子空间聚类的数目不依赖外界条件，完全由数据对象的分布决定，体现了较小的外界依赖。算法主要时间开销集中在子空间搜索过程。令 $N$ 为基本聚类规模，则计算基本聚类间相似度的时间复杂度为 $O(N^2)$，子空间搜索的时间复杂度为 $O(N^2)$。因此，算法总时间复杂度为 $O(N^2)$。

## 4 实验结果与分析

对本文算法进行性能测试，实验平台如下：Pentium D 2.6 GHz CPU，主存 512 MB，操作系统为 Windows XP pro sp2。

### 4.1 实验数据集

为了全面测试本文算法性能，在实验中使用人造和真实数据集。人造数据集为连续型数据集，共 35 维，10 000 个数据对象，包括 5 个不同维度、不同密度的子空间聚类。其中，第 2 个和第 4 个子空间聚类中数据对象分布最稀疏，且不包括全维上的聚类。实验中采用的真实数据集包括 Pendigits, Mushroom 和 Zoo。

### 4.2 算法准确度

将聚类结果的准确度定义为聚类结果中属于正确聚类的数据对象所占的比例。将本文算法与 CLIQUE[2], SUBCLU[4], ROCK[5]算法在人造和真实数据集上的聚类准确度作比较。对于上述 3 个算法，均采用取得最好聚类结果的参数，得到基于人造数据集的准确度，如图 1 所示。可以看出，本文算法在密度较低的第 2 个 10 维子空间和第 4 个 15 维子空间的准确度均高于 CLIQUE 和 SUBCLU 算法。基于真实数据集的准

确度如表 1 所示，可以看出，本文算法更适用于处理不同类型数据，且聚类准确度高于其他 3 种算法。

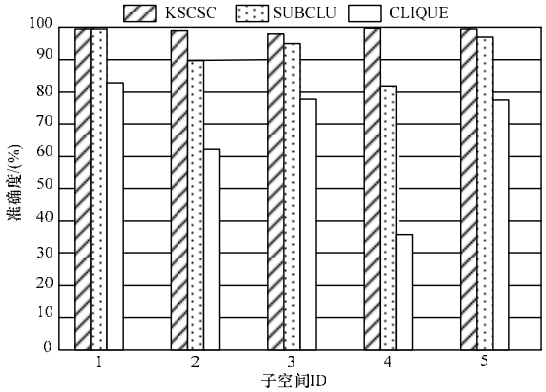


**图 1　基于人造数据集的准确度**

**表 1　4 种算法基于真实数据集的准确度　(%)**

| 数据集 | 本文算法 | SUBCLU | CLIQUE | ROCK |
|---|---|---|---|---|
| pendigits | 93.75 | 85.73 | 76.42 | 不可用 |
| mushroom | 96.87 | 不可用 | 不可用 | 96.06 |
| zoo | 97.03 | 不可用 | 不可用 | 96.04 |

### 4.3　算法伸缩性

对比本文算法与 CLIQUE, SUBCLU 算法在人造数据集和真实数据集上的伸缩性。图 2 为以上 3 种算法对数据对象数目的伸缩性的比较结果。可以看出，SUBCLU 算法与数据对象数目呈超线性关系，而 CLIQUE 算法和本文算法与数据对象数目呈线性关系。图 3 比较了 3 种算法对数据维数的伸缩性。可以看出，3 种算法与数据维数都呈超线性关系，但本文算法在数据维数的伸缩性上优于其他 2 种算法。

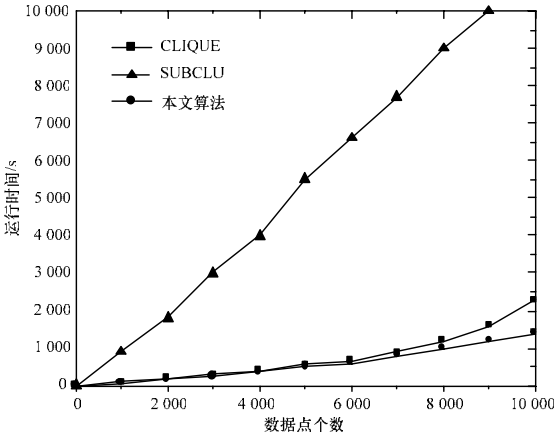


**图 2　数据对象数目的伸缩性对比**

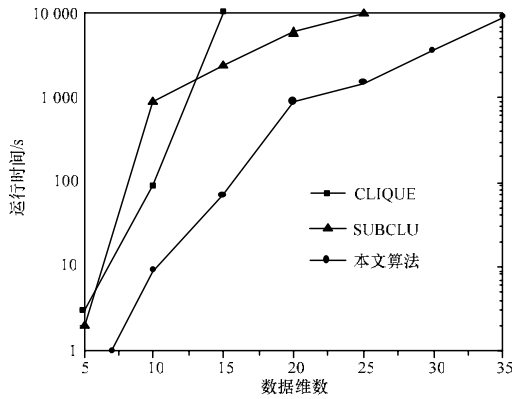


**图 3　数据维数的伸缩性对比**

### 4.4　参数对算法精度的影响

本文算法涉及 2 个参数 $k$, $m$，其中，$k$ 为最相似聚类数；$m$ 为密度阈值。实验中考察了参数的设定对算法聚类准确度的影响。图 4 给出了不同 $k$ 值对聚类精度的影响。随着 $k$ 值的增大，聚类精度有所改善，对于本文中的真实数据集来说，当 $k$ 取值在 10~15 之间时，可以达到最好的聚类精度。图 5 给出了密度阈值 $m$ 对聚类精度的影响。可以看出，当 $m$ 取值在 0.75~0.90 之间时，算法得到最好的聚类精度。

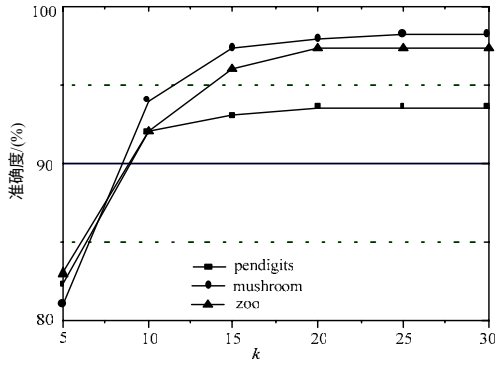


**图 4　参数 $k$ 对算法精度的影响**

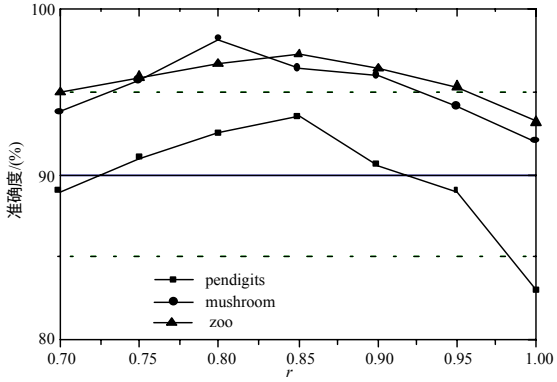


**图 5　参数 $r$ 对算法精度的影响**

## 5　结束语

针对现有子空间聚类算法处理数据类型单一、密度阈值不合理、聚类精度不理想的问题，结合基于密度算法思想，提出基于 $k$ 最相似聚类的子空间聚类算法，提高了子空间聚类算法的精度及伸缩性。下一步工作的重点是提高本文算法的精度和执行效率，并将其用于实际应用领域。

**参考文献**


[1] Jiawei H, Micheline K. 数据挖掘: 概念与技术[M]. 北京: 机械工业出版社, 2001.

[2] Rakesh A, Johannes G, Dimitrios G, et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications[C]//Proc. of SIGMOD'98. Washington, USA: ACM Press, 1998.

[3] Lance P, Ehtesham H, Huan L. Subspace Clustering for High Dimensional Data: A Review[C]//Proc. of SIAM'04. New York, USA: ACM Press, 2004.

[4] Karin K, Hans-p K, Peer K. Density-connected Subspace Clustering for High Dimensional Data[C]//Proc. of SIAM'04. New York, USA: ACM Press, 2004.

[5] Sudipto G, Rajeev R, Kyuseok S. ROCK: A Robust Clustering Algorithm for Categorical Attributes[C]//Proc. of ICDE'99. [S. l.]: IEEE Computer Society, 1999.


编辑　陈　晖