北京大学学报(自然科学版)
Acta Scientiarum Naturalium Universitatis Pekinensis

# 适用于大规模文本处理的动态密度聚类算法

李霞† 蒋盛益 张倩生 朱靖

广东外语外贸大学思科信息学院, 广州 510006; † 通信作者, E-mail: shelly_lx@126.com

**摘要** 针对传统基于密度的聚类算法对海量数据处理时, 存在参数输入复杂及时间复杂度高的问题, 给出新的密度定义方法, 并在此基础上提出一种只需一个简单输入参数就能动态识别密度不均匀聚类簇的聚类算法, 同时将其扩充为可以处理海量数据的两阶段动态密度聚类算法。在人造数据集、大规模数据集以及中英文文本语料数据集上的实验表明, 所提出的算法具有输入参数简单和聚类效率高的特点, 可以应用于海量文本数据的聚类处理。

**关键词** 文本挖掘; 聚类; 海量数据; 动态密度

**中图分类号** TP391

# A Dynamic Density-Based Clustering Algorithm Appropriate to Large-Scale Text Processing

LI Xia†, JIANG Shengyi, ZHANG Qiansheng , ZHU Jing

Cisco School of Informatics, Guangdong University of Foreign Studies, Guangzhou 510006;
† Corresponding author, E-mail: shelly_lx@126.com

**Abstract** Because of the high time complexity and complicated parameter setting in traditional density-based clustering algorithm, a new density definition is proposed, which just needs one parameter and can find clusters with different densities. The authors also expand the algorithm to a two-stage dynamic density-based clustering algorithm, which can process large-scale text corpus data. Experiments on synthetic dataset, large-scale dataset from UCI, English text corpus and Chinese text corpus show that TSDDBCA algorithm has the characteristic of easy parameter setting and high clustering efficiency, and can be applied to clustering processing to large-scale text data.

**Key words** text mining; clustering; large-scale data; dynamic density

随着互联网的快速发展和普及, 面向海量数据的自然语言处理和数据挖掘技术正逐渐成为新的研究热点, 如微博情感倾向性分析研究、面向互联网新闻及评论内容的网络舆情监测及预警研究、面向海量数据的社会网络分析等。在这些应用研究中, 聚类算法被广泛使用, 然而海量文本数据所具有的大规模和高维特征、类分布倾斜和不平衡问题以及对算法的高效率要求, 对传统聚类算法提出了新的要求。

从聚类的角度看, 文本的分布倾斜和不平衡问题可以看成是数据的密度不均匀问题, 传统的基于密度的优秀聚类算法可以发现不同大小和不同形状的聚类, 如 DBSCAN[1], CURE[2], Chameleon[3]和 SNN[4]等算法, 在处理密度不均匀的数据集上 SNN 的聚类效果相对具有优势。然而以上算法在处理海量大规模高维文本数据时, 存在一些实际应用问题: 1) 参数输入复杂, 算法的输入参数均为 2 个以上, 且设置方法复杂对用户具有较高要求; 2) 算法的时

国家自然科学基金(61070061)、教育部人文社会科学研究青年基金(11YJCZH086, 12YJCZH281)、广州社会科学规划课题(11Q20, 2012GJ31)资助

收稿日期: 2012–06–06; 修回日期: 2012–08–13; 网络出版时间: 2012-10-26 17:55

网络出版地址: http://www.cnki.net/kcms/detail/11.2442.N.20121026.1755.024.html

间复杂度高, 算法在大规模高维数据集上时间复杂度为 $O(n^2)$, 这难以应对海量文本数据的挖掘; 3) 不能处理混合属性的数据, 而实际领域中很多数据具有混合属性, 这就限制了算法的使用。

基于以上问题, 本文提出一种新的动态密度定义方法, 并在此基础上给出一种简化用户输入参数的动态密度聚类算法(a dynamic density-based clustering algorithm, DDBCA), 此算法可以发现不同大小、不同形状和不同密度的空间聚类, 但同时时间复杂度为 $O(n^2)$。为此, 继续对 DDBCA 算法进行了扩充, 实现了能够处理混合属性数据、时间复杂度为近似线性时间复杂度的两阶段动态密度聚类算法(two-step dynamic density- based Clustering algorithm, TSDDBCA), 在人造 2 维数据、UCI 高维数据[5]、Reuters-21578 英文文本语料[6]以及 sougou[7] 中文文本语料上进行的实验表明 TSDDBCA 算法对海量文本数据的聚类是可行的。

# 1 动态密度聚类算法 DDBCA

## 1.1 算法描述

经典 DBSCAN 算法通过检查数据集中每个对象的 eps 邻域来寻找类簇, 但由于半径阈值 eps 和密度阈值 MinPts 均为全局参数, 因此 DBSCAN 无法识别密度不均匀的簇。SNN 算法改进了密度的定义方法, 将对象所有 $k$ 最近邻中与其共享的邻居个数总和定义为密度, 该密度定义能够动态反应簇的密度变化, 因此能够识别密度不均匀的簇。然而 SNN 算法需要 3 个输入参数: 最近邻个数 $k$、半径阈值 eps 和密度阈值 MinPts, 虽然算法本身提供了参数设置的方法, 但对普通用户要求较高, 且较为繁复。

为了简化参数输入, 同时能够发现空间密度不同的簇, 本文提出一种新的密度定义方法, 对某个对象 $p$, 将所有 $k$ 最近邻与其距离的和的平均值的逆定义为对象 $p$ 的密度, 具体定义如下。

**定义 1** 给定阈值 $k$, $k$ 为最近邻对象个数。

1) 对象 $p$ 和 $q$ 的最近邻分别定义为 $N_k(p)$ 和 $N_k(q)$;

2) 对象 $p$ 和 $q$ 的距离定义为 $\mathrm{dist}(p,q)$;

3) 对象 $p$ 的密度定义为 $\mathrm{density}(p)=\dfrac{|N_k(p)|}{\sum_{q\in N_k(p)}\mathrm{dist}(p,q)}$。

对象 $p$ 和 $q$ 的距离在不同数据集下计算方法可以不同, 对于文本数据, 距离 $\mathrm{dist}(p,q)$ 定义为 $1-\mathrm{sim}(p,q)$, $\mathrm{sim}(p,q)$ 为两个文本对象的余弦相似度或其它相似度定义函数的值。

定义 1 直观反应了密度不同的空间聚类, 当 $k$ 值一定时, 低密度区域对象到其 $k$ 个最近邻的距离和的平均值, 要大于高密度区域对象到其 $k$ 个最近邻的距离和的平均值, 所对应的逆值为前者小于后者, 即低密度区域对象的密度值要低于高密度区域对象的密度值。为了能够自动区分密度不同的簇, 在定义 1 的基础上, 我们对每个对象依据其密度值进行排序, 从密度最大的对象开始建立一个新簇, 并迭代地聚集从该对象直接密度可达的对象, 直到无新的对象可以被添加进来。继续下一个未被处理的对象, 继续该过程, 直到全部对象处理完毕。DDBCA 的详细算法描述如下。

输入: 要聚类的数据集 $D$, 参数 $k$, $k$ 为最近邻个数
输出: 聚类好的簇列表 $S$
  对数据集 $D$, 计算所有对象的距离矩阵 $\boldsymbol{M}$
  依据定义 1 计算每个对象的密度, 记为 $\zeta$
  依据每个对象的密度值 $\zeta$ 对所有对象进行降序排序
for 排序后所有对象 $p, p\in D$ do
  以对象 $p$ 建立一个新簇 $C$
  遍历距离矩阵 $\boldsymbol{M}$, 找到所有与对象 $p$ 距离小于 $p_\zeta$ 的对象 $q$, $p_\zeta$ 为对象 $p$ 的密度
  如果对象 $q$ 未纳入其它簇中, 则纳入簇 $C$ 中
  对簇 $C$ 内新加入的对象 $o$, $o\in C$, 递归将与其距离小于 $o_\zeta$ 的对象继续聚到簇 $C$ 中, 直到没有新的对象加入, $o_\zeta$ 为对象 $o$ 的密度
end for
  输出聚类好的簇列表 S=$\{C_1, C_2, \ldots, C_m\}$。

算法中直接密度可达的定义类似于 DBSCAN, 具体定义如下。

**定义 2** 给定数据集合 $D$ 及阈值 $k$, 如果对象 $q$ 满足 $q\in N_k(p)$, 且 $\mathrm{dist}(p,q)<\mathrm{density}(p)$, 则称对象 $q$ 从对象 $p$ 出发时是直接密度可达的。

## 1.2 时间复杂度分析及在二维数据集上的实验结果

DDBCA 算法的主要时间用于计算距离矩阵和数据对象的排序, 总的时间复杂度为 $O(n^2)$, $n$ 为对象的个数, 这与 DBSCAN 以及 SNN 算法的时间复杂度平行。DDBCA 的优势在于只需要一个输入参数, 即最近邻个数 $k$。用户对参数 $k$ 值的确定比较简单, 对同一个数据集, $k$ 值越小, 对密度的要求更严

格, 聚类的结果簇会更紧密, 聚类精度更高, 聚类簇的个数更多; $k$ 值越大, 对密度的要求较松, 聚类的结果簇相对稀疏, 聚类精度更低, 聚类簇的个数更少。

为了验证 DDBCA 算法的有效性, 分别用 SNN 算法、DBSCAN 算法以及 DDBCA 算法对表 1 的 3 个人造 2 维数据集进行聚类, 聚类结果如表 2 所示。该三个数据的二维可视化图形如图 1~3 所示。从聚类结果可以看出, 对于密度均匀、形状不同的数据集 Dataset1 和 Dataset2, DDBCA, DBSCAN 算法以及 SNN 算法均能有效识别数据的自然形状, 但 DDBCA 的参数输入相对简单, 对用户没有复杂的要求, 不需要做 K-DIST 图来获取复杂的输入参数。对于密度不均匀的人造数据集 Dataset3, SNN 和 DDBCA 算法均可以识别出密度不均匀的自然簇。

**表 1　3 个人造 2 维数据集的特征**

Table1　Characters of the five synthetic datasets

| 数据集 | 自然簇个数 | 数据集大小 |
|---|---|---|
| Dataset1 | 4 | 217 |
| Dataset2 | 24 | 3400 |
| Dataset3 | 6 | 939 |

**表 2　不同聚类算法在人造数据集 Dataset1~Dataset3 上的聚类结果**

Table 2　Result of different clustering algorithms on dataset1- dataset3

| 数据集 | 算法名称 | 参数设置 | 簇个数 | 聚类精度/% |
|---|---|---|---|---|
| Dataset1 | SNN | $K$=4,Eps=1,minpts=4 | 4 | 100 |
| | DBSCAN | Eps=0.05,minpts=2 | 4 | 100 |
| | DDBCA | $K$=8 | 4 | 100 |
| Dataset2 | SNN | K=5,Eps=1,minpts=5 | 24 | 100 |
| | DBSCAN | Eps=0.03,minpts=2 | 24 | 99.8235 |
| | DDBCA | $K$=10 | 24 | 99.8235 |
| Dataset3 | SNN | K=6,Eps=3,minpts=6 | 6 | 98.6223 |
| | DBSCAN | Eps=9.75,minpts=4 | 3 | — |
| | DDBCA | $K$=8 | 8 | 98.0851 |
| | | $K$=12 | 6 | 95.0638 |

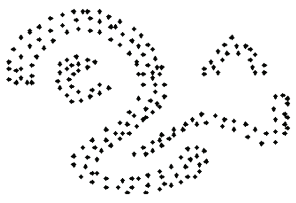

**图 1　Dataset1**

Fig.1　Dataset1

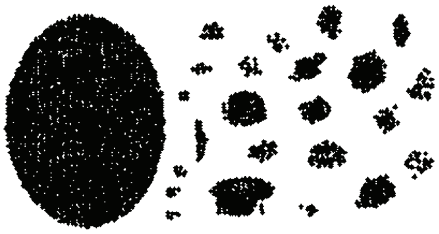

**图 2　Dataset2**

Fig.2　Dataset2

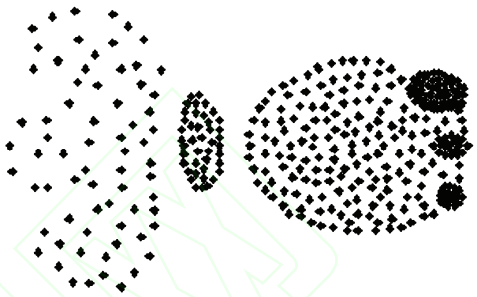

**图 3　Dataset3**

Fig.3　Dataset3

## 2　两阶段动态密度聚类算法

为了让算法适用于海量数据的实际使用以及面向更加通用数据的处理, 对 DDBCA 算法进一步改进, 使其能够处理混合型属性数据, 并结合两阶段聚类思想, 降低算法总的时间复杂度。

### 2.1　DDBCA 算法混合属性的处理

为了方便定义, 引入一组记号, 用 $N$ 表示数据集 $D$ 包含的记录总数, 每条记录有 $m$ 个属性, 其中有 $m_C$ 个分类属性和 $m_N$ 个数值属性, $m = m_C + m_N$, 第 $i$ 个属性 $D_i$ 有 $n_i$ 个不同的取值。

**定义 3**　簇 $C$ 的摘要信息 CSI (cluster summary information) 定义为 $\mathrm{CSI} = \{n, \mathrm{ClusterID}, \mathrm{summary}\}$, 其中 $n$ 为簇 $C$ 的大小($n = |C|$); ClusterID 为簇中对象标识号 ID 的集合; summary 由分类属性中不同取值的频度信息和数值型属性的质心构成, 即 $\mathrm{summary} = \{\langle \mathrm{Stat}_1, ..., \mathrm{Stat}_{m_C}, c_{m_C+1}, c_{m_C+2}, ..., c_{m_C+m_N}\rangle\}$。这里 $\mathrm{Stat}_i = \{(a, \mathrm{Freq}_{C|D_i}(t)) \,|\, a \in C \,|\, D_i\}$ 表示属性 $D_i$ ($1 \leqslant i \leqslant m_C$) 上不同取值的频度集合; $C \,|\, D_i$ 表示 $D_i$ 在 $C$ 上的投影; $\mathrm{Freq}_{C|Di}{}^{(a)}$ 表示属性值 $a$ 在 $C \,|\, D_i$ 中出现的频度, 若 $a \notin C \,|\, D_i$, 则 $\mathrm{Freq}_{C|Di}{}^{(a)} = 0$; $c_j$ 表示数值属性 $D_j$ ($m_C + 1 \leqslant j \leqslant m_C + m_N$) 的质心。

**定义 4**　簇 $C_1$ 与 $C_2$ 间的距离定义为 $d(C_1, C_2) =$

$\sqrt{\sum_{i=1}^{m}\mathrm{dif}(C_i^{(1)},C_i^{(2)})^2/m}$，这里 $\mathrm{dif}(C_i^{(1)},C_i^{(2)})$ 为 $C_1$ 与 $C_2$ 在属性 $D_i$ 上的差异。对于分类属性 $D_i$，其值定义为 $C_1$ 与 $C_2$ 中 $|C_1|\cdot|C_2|$ 对对象在属性 $D_i(1\leqslant i\leqslant m_C)$ 上的距离平均值: $\mathrm{dif}(C_i^{(1)},C_i^{(2)})=1-\frac{1}{|C_1|\cdot|C_2|}\sum_{p\in C_1}\mathrm{Freq}_{C_1|D_i}(p_i)\cdot\mathrm{Freq}_{C_2|D_i}(p_i)=1-\frac{1}{|C_1|\cdot|C_2|}\sum_{q\in C_2}\mathrm{Freq}_{C_1|D_i}(q_i)\cdot\mathrm{Freq}_{C_2|D_i}(q_i)$，这里 $p=[p_1,p_2,\cdots,p_m]$，$q=[q_1,q_2,...,q_m]$ 是 $D$ 中两个对象。对于数值属性 $D_i$，$\mathrm{dif}(C_i^{(1)},C_i^{(2)})$ 定义为两质心间的绝对偏差: $\mathrm{dif}(C_i^{(1)},C_i^{(2)})=\left|c_i^{(1)}-c_i^{(2)}\right|$ $(m_C+1\leqslant j\leqslant m_C+m_N)$，$c_i^{(1)},c_i^{(2)}$ 分别为 $C_1$、$C_2$ 对应于属性 $D_i$ 的质心。

## 2.2 两阶段动态密度聚类算法

使用定义 3 和定义 4 对 DDBCA 算法进行推广，使其可以处理含分类属性的数据，结合一趟聚类算法[8]快速划分的优势，以及 DDBCA 算法对密度不同的簇的动态识别能力，实现了能够处理混合属性且时间复杂度为近似线性时间复杂度的两阶段动态密度聚类算法(two-step dynamic density-based clustering algorithm, TSDDBCA)。TSDDBCA 先用一趟聚类算法对原数据集进行聚类，得到大小几乎相同的初始划分簇，然后利用推广后的 DDBCA 算法对初始划分进行归并，从而在快速聚类的基础上同时能够识别大小不同、形状不同以及密度不同的空间聚类，算法为 2 个步骤，具体描述如下。

**第一步** 原始数据的初始划分。

使用一趟聚类算法将数据集 $D$ 划分为 $K_1$ 个簇列表 $S=\{C_1,C_2,...,C_{K_1}\}$，每个簇的半径相同，$r$ 表示一趟聚类算法的阈值，聚类步骤描述如下:

初始时，簇集合 $S$ 为空，读入一个新的对象 $p$

以这个对象 $p$ 构造一个新的簇

若已到数据集末尾，则转 6，否则读入一个新对象 $p$，在簇集 $S$ 中找到与对象 $p$ 最近的簇 $C^*$，也即对 $S$ 中的所有 $\overline{C}$，有 $d(p,C^*)\leqslant d(p,\overline{C})$

如果 $d(p,C^*)>r$，转 2

把对象 $p$ 合并到簇 $C^*$ 中，并更新该簇 $C^*$ 的摘要 $CSI$，然后转 3

结束。

**第二步** 归并初始聚类簇。

使用推广后的 DDBCA 算法，对第一步划分出的初始簇进行再次聚类。假定第一步中初始划分为 $K_1$ 个簇 $S=\{C_1,C_2,...,C_{K_1}\}$，经过推广 DDBCA 算法，再次聚类后，得到 $K_2$ 个最终聚类 $S'=\{C_1',C_2',...,C_{K2}'\}$，聚类过程描述如下:

将第一步中初始划分的每个簇视为一个聚类对象

计算 $K_1$ 个聚类对象的距离矩阵，记为 $M$

依据定义 1 计算每个聚类对象的密度，记为 $\zeta$

依据每个聚类对象的密度值 $\zeta$ 对所有聚类对象进行降序排序

for 排序后所有聚类对象 $C, C\in S$ do

以聚类对象 $C$ 建立一个新簇 $C'$

遍历距离矩阵 $M$，找到所有与聚类对象 $C$ 距离小于 $\zeta$ 的聚类对象 $C^*$

如果聚类对象 $C^*$ 未纳入其它簇中，则纳入到簇 $C'$ 中

对簇 $C'$ 内新加入的聚类对象，递归将与其距离小于该对象密度值 $\zeta$ 的聚类对象纳入到簇 $C'$ 中，直到没有新的聚类对象加入

end for

输出聚类好的簇列表 $S'=\{C_1',C_2',...,C_{K2}'\}$。

## 2.3 时间复杂度分析及参数设置

TSDDBCA 算法的时间复杂度由两个部分组成，即第一步初始聚类时间复杂度和第二步归并聚类时间复杂度的和。第一步对原始数据集进行聚类得到初始簇，期望的时间复杂度为 $O(N\cdot K_1\cdot m)$，$K_1$ 最终聚类簇的个数，$N$ 为数据集的大小，$m$ 为数据集的维数。第二步的时间复杂度决定于 DDBCA 聚类算法的运行时间复杂度，为 $O(K_1^2)$，$K_1$ 为第一步中聚类的初始簇的个数。这样 TSDDBCA 总的时间复杂度为 $O(N\cdot K_1\cdot m)+O(K_1^2)$。对于大规模数据集，第一步聚类簇的个数将远远小于原始数据集的大小 $N$，即 $K_1<<N$，因此 $(K_1)^2<N$，第二步的时间复杂度低于第一步的时间复杂度。TSDDBCA 算法的时间复杂度主要由第一步决定，期望时间复杂度为 $O(N\cdot K_1\cdot m)$，近似等价于 $O(N)$，为近似线性时间复杂度，可用于大规模数据集的聚类。

第一步中一趟聚类算法的半径阈值 $r$ 的选择采用抽样计算，具体操作步骤为在数据集 $D$ 中随机选择若干对对象，计算每对对象间的距离及其均值 EX，阈值 $r$ 取为 EX 的倍数。第二步的 DDBCA 算法的阈值为最近邻个数 $k$，依据用户对聚类结果的精度要求来决定 $k$ 值的大小。

## 2.4 实验结果

为了检验 TSDDBCA 算法的有效性，在 UCI Machine Learning Repository[5]提供的机器学习数据集以及 Reuters-21578[6]和搜狗中英文文本语料[7]数

据上对算法进行了测试。整个实验是在一台配置 32 位 Win7 操作系统, 安装内存为 2G, 处理器为 Intel Celeron G530 2.4 GHz 的台式机器上进行, 程序用 VC++6.0 实现。实验结果中的符号“/”表示相关文献没有给出相关结果。

### 2.4.1 在中小规模混合数据集上的性能

Musroom 数据集[5]有 8124 条记录, 每条记录由 22 个分类属性来刻画。每条记录用来描述一种蘑菇的特性, 并被标识为是有毒 p(Poisonous)还是可食用 e(Edible), 其中有毒记录 3916 条, 可食用记录 4208 条。表 3 给出了算法 TSDDBCA 与一趟聚类算法[8]、ROCK 算法[9]和 Squeezer 算法[10]在 mushroom 数据集上聚类结果的对照表。mushroom 数据集的属性都是非数值型的, 文献中传统的 SNN, DBSCAN 算法等均无法处理。TSDDBCA 算法在 $r$=0.3, $k$=8 时得到的结果与 ROCK 和 Squeezer 相当, 聚类簇的个数为 22, 聚类精度为 99.90153%,聚类时间为 2 s。

**表 3 不同聚类算法在 mushroom 数据集上的聚类结果**

Table 3 Result of different clustering algorithms on mushroom

| 聚类方法 | 参数设置 | 簇个数 | 聚类精度/% | 聚类时间 |
|---|---|---|---|---|
| TSDDBCA | $r$=0.3, $K$=8 | 22 | 99.0153 | 2 s |
| ROCK | — | 21 | 99.61 | — |
| Squeezer | — | 24 | 99.99 | — |

### 2.4.2 在大规模数据集上的实验效果

KDDCUP99[5]包含了约 4900000 条网络记录, 这些记录分为 22 种攻击记录和正常记录共 23 个类别, 每条记录由 7 个分类属性和 34 个数值型属性刻画。由于整个数据集太大, 通常使用一个 10%的子集 KDDCUP99_10 (包含 494020 条记录)来测试算法的性能。

对于 KDDCUP99_10 数据集, 由于具有混合属性及规模太大, 传统 SNN 类算法仅仅对其中的数值型数据抽样, 选取部分数据进行测试。而本文算法 TSDDBCA 是对整个 10%的 KDDCUP99 数据集的全部属性和全部数据进行测试。表 4 给出了 TSDDBCA 算法和一趟聚类算法对 KDDCUP99_10 数据集的实验结果。在阈值 $r$=0.25, $k$ 取 4 时的聚类精度为 98.8917%, 仅比一趟聚类算法在同一参数下的聚类精度低 2.776%, 但簇个数由原来的 98 个减少为 45 个, 得到的簇个数更接近于数据集的类别数。

**表 4 不同聚类算法在 KDDCUP99 上的聚类结果对比**

Table 4 Result of different clustering algorithms on KDDCUP99

| 聚类方法 | 参数设置 | 簇个数 | 聚类精度/% | 聚类时间/ s |
|---|---|---|---|---|
| 一趟聚类算法 | $r$=0.25 | 98 | 99.1693 | 141 |
| TSDDBCA | $r$= 0.25, $k$=3 | 55 | 99.0375 | 141 |
| | $r$=0.25, $k$=4 | 45 | 98.8917 | 141 |

### 2.4.3 在文本数据集上的实验结果

为了验证算法 TSDDBCA 在大规模文本处理中的聚类效果, 分别在 Reuters-21578 英文文本语料[6]和搜狗中文文本语料[7]上进行实验验证, 实验中所有文本特征采用信息增益[11]方法提取。为了实验结果的可比性, 所有语料均提取 100 维特征, 文本使用向量空间模型[12]表示, 向量权重值采用经典的 TF-IDF[13]方法。

Reuters-21578[6]是文本自动分类的公开英文基准语料库, 包含 1987 年在路透社报纸上的 21578 篇新闻报道, 由 Dobbins 等进行人工分类标注, 总共包含 135 个类别。选取其中使用较为频繁的来自 acq, crude, earn, interest 和 trade 共 5 个类别的子集文本作为实验语料进行聚类处理。

图 4 为一趟聚类算法和 TSDDBCA 算法在该子集上的聚类结果对比, 参数设置分别为 $r$=2EX 和 $r$=2EX, $k$=3。实验结果表明 TSDDBCA 算法的聚类精度与一趟聚类算法的聚类精度基本保持一致。随着聚类文本数量的增加, 一趟聚类算法的聚类簇的个数会不断增加, 而 TSDDBCA 的聚类簇个数保持在一定范围内, 这表明 TSDDBCA 算法对文本语料的聚类结果趋于稳定。

搜狗中文文本分类语料[7]来源于搜狐新闻网站保存的大量经过编辑手工整理与分类的新闻语料与对应的分类信息, 选取其中的财经、IT、军事、旅游及体育 5 个类别子集共 9950 篇文章进行聚类实验, 聚类参数设置分别为 $r$=2EX 和 $r$=2EX, $k$=3, 实验结果如图 5 所示。图 5 结果表明 TSDDBCA 算法在搜狗中文文本语料上的聚类精度略微低于在 Reuters-21578 上的聚类精度, 同时随着聚类文本的不断增加, TSDDBCA 算法的聚类结果趋于稳定, 聚类簇的个数保持在 20 个簇左右。

为了对比本文算法与 SNN 算法在聚类精度和聚类时间上的差异, 文章分别在 Reuters-21578 来自 5 个类别的 7000 篇英文语料和搜狗来自 5 个类别

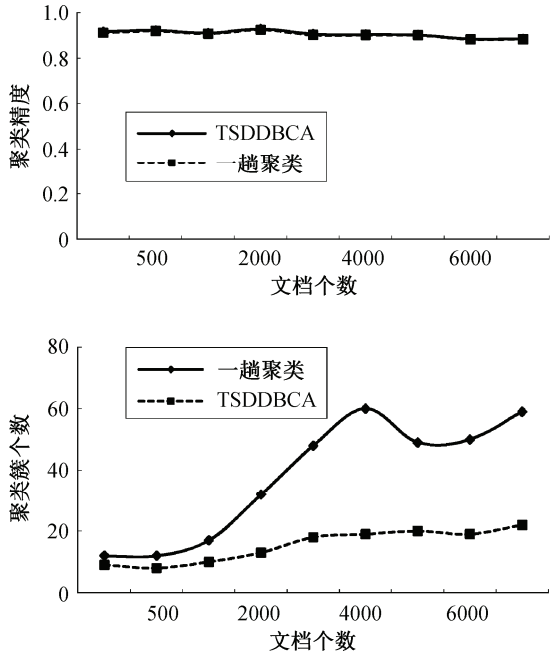


**图 4　TSDDBCA 算法和一趟聚类算法在 Reuters-21578 英文语料上的实验结果对比**

Fig. 4　Comparision of clustering result of TSDDBCA and one-pass clustering algorithm on Reuters-21578

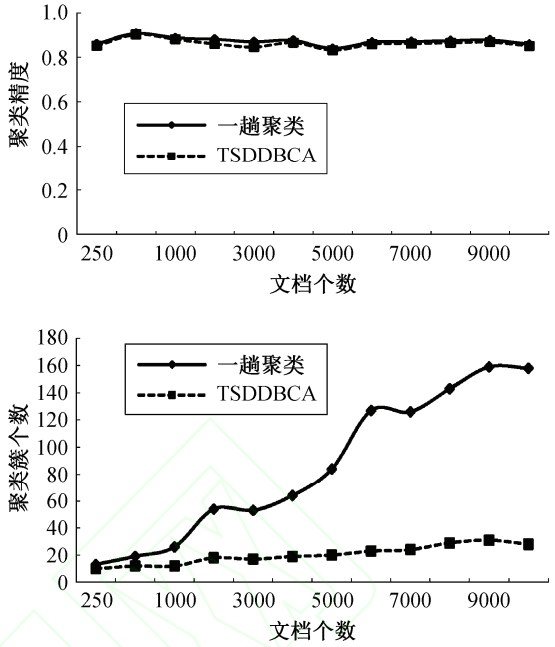


**图 5　TSDDBCA 算法和一趟聚类算法在搜狗中文语料上的实验结果对比**

Fig.5　Comparision of clustering result of TSDDBCA and one-pass clustering algorithm on Sogou Chinese text corpus

9950 篇中文语料上进行了聚类对比，实验中 SNN 算法的参数设置为 $k$=2, eps=0, minpts=2, DDBCA 算法的参数设置为 $k$=6, TSDDBCA 算法的参数设置为 $r$=2EX, $k$=3，实验结果如表 5 所示。实验结果表明随着聚类文本的增加，DDBCA 算法的聚类精度在高于 SNN 的基础上，聚类簇个数略多于 SNN 算法，同时两者的时间都随着样本个数呈指数增长。而扩充算法 TSDDBCA 在精度略低于 SNN 和 DDBCA 算法的基础上，聚类结果趋于稳定，但聚类簇个数要远远小于 SNN 算法。同时聚类时间是近似线性的，在 Reuters-21578 数据子集和搜狗数据集上的平均聚类时间分别为 1.045 s 和 1.5408 s。

## 3　结论

本文提出了一种能够识别密度变化的动态密度聚类算法 DDBCA，该算法可以识别不同大小、不同

**表 5　不同算法在 Reuters-21578 和搜狗中英文文本语料上的聚类结果对比**

Table 5　Comparision of different clustering algorithms on Reuters-21578 and Sougou corpus

| 数据集 | | 聚类簇个数/个 | | | 聚类精度 | | | 执行时间/s | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | SNN | DDBCA | TSDDBCA | SNN | DDBCA | TSDDBCA | SNN | DDBCA | TSDDBCA |
| Reuters-21578 | 1000 | 355 | 421 | 6 | 0.898 | 0.952 | 0.909 | 11.696 | 11.512 | 0.655 |
| | 3000 | 1167 | 1280 | 15 | 0.914667 | 0.945667 | 0.903667 | 102.834 | 102.398 | 0.936 |
| | 5000 | 1948 | 2145 | 15 | 0.919 | 0.9504 | 0.901 | 290.583 | 285.938 | 1.17 |
| | 7000 | 2793 | 2906 | 19 | 0.92151 | 0.943286 | 0.903667 | 570.59 | 569.288 | 1.419 |
| | 平均 | 1565 | 1688 | 13 | 0.913294 | 0.947838 | 0.904334 | 243.925 | 242.284 | 1.045 |
| 搜狗语料 | 1000 | 355 | 382 | 11 | 0.908 | 0.942 | 0.882 | 12.722 | 12.88 | 0.708 |
| | 3000 | 1070 | 1170 | 13 | 0.928333 | 0.934 | 0.861 | 110.312 | 111.052 | 0.936 |
| | 5000 | 1803 | 2089 | 17 | 0.917 | 0.9356 | 0.867 | 311.489 | 307.254 | 1.31 |
| | 7000 | 2736 | 2957 | 18 | 0.925571 | 0.938857 | 0.862571 | 655.288 | 649.187 | 2.199 |
| | 9000 | 3520 | 3823 | 25 | 0.925111 | 0.942222 | 0.869333 | 1097.4 | 1076.34 | 2.551 |
| | 平均 | 1896 | 2084 | 16 | 0.920803 | 0.938536 | 0.868381 | 437.4422 | 431.3426 | 1.5408 |

形状以及不同密度的聚类, 同时参数是点的最近邻个数, 设置简单。考虑到算法的实际应用性, 对DDBCA 算法进行了扩充和改进, 得到两阶段动态密度聚类算法 TSDDBCA, 该算法可以处理混合属性的数据集, 并具有近似线性时间复杂度, 可以应用于更加广泛的实际应用领域和大规模数据的聚类。在中英文文本语料上的实验表明, TSDDBCA 算法输入参数简单, 同时随着聚类文本的规模逐渐增加, 聚类簇的大小趋于稳定, 能较好地识别不均匀密度的簇类, 并且 TSDDBCA 具有近似线性时间效率, 适用于海量文本的聚类处理。

## 参考文献


[1] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise // Proceedings of the 2nd International Conference on Knowledge Discovering in Databases and Data Mining (KDD-96). Massachusetts: AAAI Press, 1996: 226–232

[2] Guha S, Rastogi R, Shim K. CURE: an efficient clustering algorithm for large databases // Proceedings ACM SIGMOD International Conference on Management of Data (SIGMOD 1998). Washington, 1998: 73–84

[3] Karypis G, Han E, Kumar V. CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. IEEE Computer, 1999, 32(8): 68–75

[4] Ertoz L, Michael S, Kumar V. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data // Proceedings of the third SIAM International Conference on Data Mining (SIAM 2003). San Francisco, CA, 2003: 47–58

[5] Merz C J, Merphy P. UCI repository of machine learning databases [DB/OL]. (2000)[2012–05–30]. http://www.ics.uci.edu/~mlearn/MLRepository.html

[6] Lewis D D. Reuters-21578 text categorization collection data set [DB/OL]. (1997)[2012–05–30]. http://archive.ics.uci.edu/ml/datasets/Reuters21578 + Text+Categorization+collection

[7] 搜狐研发中心.搜狗文本分类语料库[DB/OL]. (2006)[2012–05–30].http://www.sogou. com/labs/dl/c. html

[8] Jiang Shengyi, Xu Yuming. An efficient clustering algorithm // Procedings of 2004 International Conference on Machine Learning and Cybernetics. Shanghai, 2004: 1513–1518

[9] Guha S,Rastogi R, Shim K. ROCK: a robust clustering algorithm for categorical attributes // Proceedings of the 15th ICDE. Sydney, 1999: 512–521

[10] He Zengyou, Xu Xiaofei, Deng Shengchun. Squeezer: an efficient algorithm for clustering categorical data. Journal of Computer Science and Technology, 2002, 17(5): 611–624

[11] Yang Yiming. A comparison study on feature selection in text categorization // Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997). Nashville, Tennessee, 1997: 412–420

[12] Salton G, Wong A, Yang C S. A vector space model for automatic indexing. Communications of ACM, 1975, 18(11): 613–620

[13] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing and Management, 1998, 24(5): 513–523