文本聚类中基于密度聚类算法的研究与改进

苏 喻,郑 诚,封 军

(安徽大学 计算智能与信号处理教育部重点实验室,安徽 合肥 230039)

摘 要:文本聚类在很多领域都有广泛应用,而聚类算法作为文本聚类的核心直接决定了聚类的效果和效率。结合基于划分的聚类算法和基于密度的聚类算法的优点,提出了基于密度的聚类算法 DBCKNN。算法利用了 k 近邻和离群度等概念,能够迅速确定数据集中每类的中心及其类半径,在保证聚类效果的基础上提高了聚类效率。

关键词:文本聚类;基于密度;k 近邻;离群度

中图分类号:TP301 **文献标识码**:A **文章编号**:1674-7720(2011)01-0001-03

The research and improvement of density-based clustering algorithm in text clustering

Su Yu, Zheng Cheng, Feng Jun

(Educational Department Key Laboratory of Intelligent Computing & Signal Processing, Anhui University, Hefei 230039, China)

Abstract: Nowadays, the applications of text clustering are applied widely in many fields. the clustering algorithm as the core of text clustering directly determines the effectiveness and efficiency of clustering. In this paper, we combine the advantages between the partition-based clustering algorithm and the density-based clustering algorithm and propose a density-based clustering algorithm named DBCKNN. This algorithm by using the concepts of k-nearest neighbor and outlier degree can find the center and radius of each cluster from a data set rapidly, and improve the efficiency of clustering on the basis of kind effectiveness.

Key words: text clustering; density-based; k-nearest neighbor; outlier degree

文本聚类是指将 n 篇文章聚集成 k 类,使得每类内的样本相似度较大,每类间的样本相似度较小。文本聚类是一种特殊的数据聚类,有着自身的特点。文本的聚类对象维数较高,决定了聚类算法需要快速收敛,注重效率。国内外也围绕着文本聚类提出了很多理论和算法,采用的核心聚类算法一般分为两类,一类是基于划分的聚类算法,如 K-means 算法、CLARANS 算法等;另一类是基于密度的聚类算法,如 KNNCLUST 算法、DBSCAN 算法等。

1 基于密度聚类算法的研究与改进

1.1 现有算法的缺陷

K-means 算法与 DBSCAN 算法分别作为基于划分和基于密度的聚类算法的代表,被广泛应用于文本聚类中。其中 K-means 算法有实现简单、时间复杂度低等优点,但算法需要指定种类数,且对初始点依赖性过强,导致聚类效果不理想;DBSCAN 算法则有抗噪音性强、聚类准确性高等优点,但算法的主要阈值参数很难确定,且时间复杂度过高,导致聚类效果不理想。

本文将 K-means 算法的特性,融入到利用 k 近邻概念的基于密度的聚类算法中,提出了 DBCKNN 算法(Density-Based Clustering using K-Nearest Neighbor),在保证算法准确性的前提下,提高了算法效率。

1.2 相关标记与标识

定义 1 点 p 的 ε 邻域

$$\text{Neighbors}(p,\varepsilon)=\{q\in D|\text{dis}(p,q)\leqslant\varepsilon\} \quad (1)$$

其中 $\text{dis}(p,q)$ 是点 p 和点 q 之间的距离。

定义 2 点 p 的 k 近邻距离和点 p 的 k 平均近邻距离

点 p 的 k 近邻距离,记作 $\text{KNNdis}(p,k)$。$\text{KNNdis}(p,k)=\text{dis}(p,q)$。其中 q 是 p 的第 k 个最近邻居。

点 p 到其 $\text{KNNdis}(p,k)$ 邻域中每个点距离的均值称为点 p 的 k 平均近邻距离,记作 $\text{aveKNNdis}(p,k)$。

$$\text{aveKNNdis}(p,k)=\sum_{q_i}\text{dis}(p,q_i)/k \tag{2}$$

其中 $q_i\in\text{Neighbors}(p,\text{KNNdis}(p,k))$。

定义 3 点 p 在 ε 邻域的 n 边缘点集

$$\text{border}(p,\varepsilon,n)=\{q\in\text{Neighbors}(p,\varepsilon)\mid\text{dis}(p,q)\geqslant \text{mindis}(\text{Neighbors}(p,\varepsilon),p,n)\} \tag{3}$$

其中 $\text{mindis}(A,p,n)$ 表示点集 A 中离点 p 第 n 小的点。

定义 4 点 p 在 ε 邻域的 n 边缘点集的 k 离群度

$$\text{degree}(p,k,\varepsilon,n)=\frac{1}{n}\sum_{q_i}\frac{\text{aveKNNdis}(q_i,k)}{\text{aveKNNdis}(p,k)} \tag{4}$$

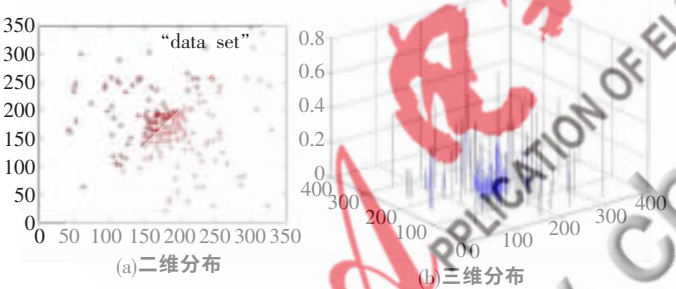
其中 $q_i\in\text{border}(p,\varepsilon,n)$。

当 $n=k$，$\varepsilon=\text{KNNdis}(p,k)$，$q_i\in\text{Neighbors}(p,\text{KNNdis}(p,k))$ 时，$\text{degree}(p,\varepsilon,k,n)$ 转化为 $\text{degree}'(p,k)$，称为点 p 的 k 离群度。

定义 5 点 p 的绝对离群度

$$\text{absdegree}(p,k)=|\text{degree}'(p,k)-1|^{\theta} \tag{5}$$

点的绝对离群度域值为 $(0,+\infty)$，且值越小，表示点越有可能为类中点，反之，表示点越有可能为噪音点。θ 一般取不大于 3 的正整数，θ 越大，不同对象的绝对离群度分布越离散。如图 1(a)中类近似为高斯分布，图 1(b)中 z 轴为点的绝对离群度值，其 θ 取 1。可以看出，越是类中心的点，其绝对离群度越小，而类边缘和噪音点都有相对高的绝对离群度值。



(a)二维分布 (b)三维分布

图 1 数据集的绝对离群度分布图

定义 6 边缘点集的 k 平均近邻距离的均值

$$\text{avedis}(p,\varepsilon,k,n)=\sum_{q_i}\text{aveKNNdis}(q_i,k)/|\text{border}(p,\varepsilon,n)| \tag{6}$$

其中 $q_i\in\text{border}(p,\varepsilon,n)$。

1.3 算法改进

1.3.1 确定类初始中心核心子算法 FINDCENTER(ε)

现有的基于密度的聚类算法中，通常会对全体点进行一次密度值扫描，导致算法复杂性和空间复杂性过高。改进后的算法，利用划分算法中迭代并更新中心点的思想，可以对定半径的超球体的移动具有指导性，使得落入超球体内部的点相对更多，即超球体的密度相对更大。

算法 1 FINDCENTER(ε)

算法输入：p，ε；**算法输出**：p

(1)以 ε 为邻域半径，求 $|\text{Neighbors}(p,\varepsilon)|$ 和 $\text{absdegree}(p,k)$。

(2)基于 $\text{Neighbors}(p,\varepsilon)$ 重新计算类中心点 q，并求 $|\text{Neighbors}(q,\varepsilon)|$，$\text{absdegree}(p,k)$。如果 $|\text{Neighbors}(p,\varepsilon)|<|\text{Neighbors}(q,\varepsilon)|$，且 $\text{absdegree}(q,k)<\text{absdegree}(p,k)$，则 $p=q$，返回(1)；否则，输出中心点 p，算法结束。

1.3.2 调整类半径核心子算法 ADJUSTRADIUS(p,ε)

现有聚类算法中探测类半径一般会取定区间定步长的连续值，然后通过一个评价函数确定最优的类半径。通过对点离群度分析得知，当 p 的绝对离群度接近 0 时，若 $\text{degree}(p,k,\varepsilon,n)$ 小于阈值 α，说明点 p 的密度与其边缘点集的密度较融洽，可以视为一类，应扩大半径继续考察周边点；若 $\text{degree}(p,k,\varepsilon,n)$ 大于阈值 β，说明点 p 的密度大于其边缘点集的密度，应缩小半径。而半径变化的步长取 $\text{avedis}(p,\varepsilon,n,k)$ 的 λ 倍。步长为已部分探测过的环形区域内外半径差，避免步长取值的盲目性；且探测半径为离散值，降低了时间复杂度。算法中标记 f 保证了算法的收敛。

算法 2 ADJUSTRADIUS(p,ε)

算法输入：p，ε；**算法输出**：ε

(1)$f=-1$；

(2)计算 $c=\text{degree}(p,k,\varepsilon,n)$，记录 $\text{avedis}(p,\varepsilon,n,k)$。

(3)若 $f=-1$，则到(4)；若 $f!=-1$，则到(5)。

(4)若 $c<\alpha$ 且 $|\text{Neighbors}(p,\varepsilon)|<=n$，则当前类半径扩大，$f=1$，$\varepsilon=\varepsilon+\text{avedis}(p,\varepsilon,n,k)\times\lambda$，返回(2)；若 $c>\beta$ 且 $|\text{Neighbors}(p,\varepsilon)|>n$，则当前类半径缩小，$f=0$，$\varepsilon=\varepsilon-\text{avedis}(p,\varepsilon,n,k)\times\lambda$，返回(2)；否则，输出半径 ε，算法结束。

(5)若 $c<\alpha$ 且 $|\text{Neighbors}(p,\varepsilon)|<=n$，则到(6)；若 $c>\beta$ 且 $|\text{Neighbors}(p,\varepsilon)|>n$，则到(7)。

(6)若 $f=1$，则当前类半径扩大，$\varepsilon=\varepsilon+\text{avedis}(p,\varepsilon,n,k)\times\lambda$，返回(2)；若 $f=0$，则当前类半径扩大，$\varepsilon=\varepsilon+\text{avedis}(p,\varepsilon,n,k)\times\lambda/2$，算法结束。

(7)若 $f=0$，则当前类半径缩小，$\varepsilon=\varepsilon-\text{avedis}(p,\varepsilon,n,k)\times\lambda$，返回(2)；若 $f=1$，则当前类半径缩小，$\varepsilon=\varepsilon-\text{avedis}(p,\varepsilon,n,k)\times\lambda/2$，算法结束；否则，输出半径 ε，算法结束。

算法中的 λ 取值一般为 0~1。λ 取值越小，半径变化越小，迭代次数越多，但最终得到类半径的值越准确。算法中的 (α,β) 域越宽，聚类粒度越大。算法中的 n 是不大于 k 的正整数，一般取值和 k 相同。n 取值越大，则时间复杂度越高，但最终得到类半径的值越准确。

1.3.3 DBSCAN 算法思想

在数据对象集中找到 $\text{absdegree}(p,k)$ 大于阈值的对象 p 后，通过反复迭代 FINDCENTER(ε)和 ADJUSTRADIUS(p,ε)，找出初始类 C_i，并将 C_i 排除出数据对象集。重复上述过程，生成初始类集。通过初始类集中各类间的包含关系和评价函数，将噪音点集从初始类集中提取出。最后将噪音点集中的对象按与类集中各类中心点的距

离分配给各个类。

2 实验

通过实验对 DBCKNN 算法的聚类效果和时间效率进行对比和分析。数据集采用两个著名的数据集 Iris 和 KDDCUP1000。测试数据集信息如表 1。这两个数据集每一类的数据映射到高维空间中近似为正凸型的超球体，符合文本聚类中所提取的文本特征向量的分布情况，其中 Iris 为 3 类，KDDCUP1000 为 5 类。实验用 VC6.0 编写，在配置 Pentium Ⅳ 2.4 GB CPU、内存 1 GB、80 GB 硬盘的计算机上运行。

表 1 测试数据集

数据集	数据个数	数据维度
Iris 数据集	150	4
KDDCUP1000 数据集	1 500	34

本文对聚类效果的评判标准采用参考文献[4]中提出的聚类质量判定式：$S\_Dbw(c)=Scat(c)+Dens\_bw(c)$，其中 c 为类集，$Dens_bw(c)$评价的是各类间的平均密度，值越小表示类间区分度越好；$Scat(c)$评价的是类内元素的相似性，值越小表示类越内聚。

对经典算法 K-means、DBSCAN 和本文提出的 DBCKNN 算法在聚类效果和效率上做对比。其中 K-means 算法的 k 分别取 3 和 5，并对初始点集做预处理，尽量使初始点集分散且局部密度相对大，DBSCAN 算法的 minpts、eps 取 2.5，DBCKNN 算法的 λ 取 0.5，k、n 取 20，θ 取 2，(α,β)取(0.9，1.1)。

图 2 为预处理后的 K-means 算法、DBSCAN 算法和 DBCKNN 算法对测试数据集的聚类效果比较。

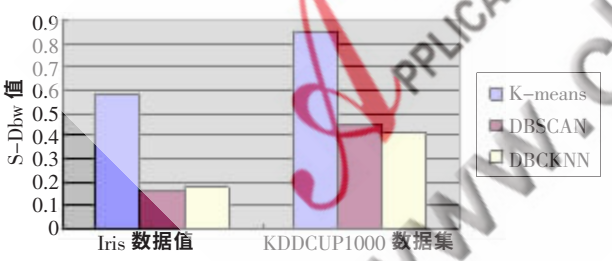


图 2 三种聚类算法的聚类效果比较

从图 2 可以看出，虽然已经对初始点集做了预处理，而且对 k 取了正确的值，但是 K-means 算法效果仍然不理想。由于 DBSCAN 算法在数据对象密度的处理上更精确，在数据对象维数较低时，效果略好于 DBCKNN 算法；而当数据对象维数较高时，高维空间中数据分布稀疏，DBSCAN 算法会误将部分数据对象视为噪音点，从而对聚类效果产生负面影响，而由于 DBCKNN 算法采用 k 近邻距离作为密度探测半径，对噪音点的处理更加合理，所以在数据对象维数较高时效果要略好于 DBSCAN 算法。

图 3 为 K-means 算法、DBSCAN 算法和 DBCKNN 算法对测试数据集的聚类效率。

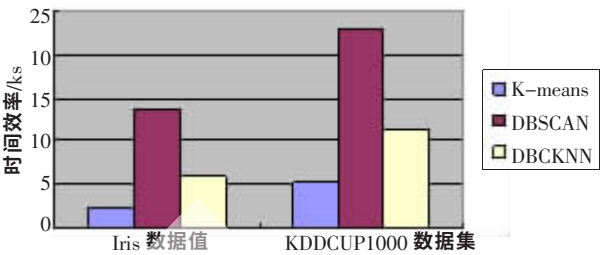


图 3 三种聚类算法的聚类效率比较

从图 3 可以看出，K-means 算法的效率非常高，在常数次迭代就得到聚类结果，数据规模对聚类效率的影响有限；DBSCAN 算法要对所有数据对象的密度进行一次以上的处理，聚类效率依赖于数据规模，导致效率相对低下；本文的 DBCKNN 算法会根据数据对象局部区域的密度信息来评价这个局部区域所有数据对象的密度信息，所以聚类效率比 K-means 算法低，但远高于 DBSCAN 算法。

本文结合了基于划分的聚类算法和基于密度的聚类算法各自的优点，提出了一种能够快速找到类中心并自适应类半径的聚类算法 DBCKNN 算法。DBCKNN 算法能在对高维空间下每类形似正凸形超球体的数据对象集进行相对准确的聚类情况下，提高算法效率。另外，本文通过分析及实验数据对比，从聚类效果和聚类效率两方面验证了这种改进方法的正确性和高效性。进一步将这种方法和基于语义的聚类方法相结合，应用于聚类搜索引擎等数据挖掘领域，是下一步研究的重点。

参考文献

[1] 孙吉贵.聚类算法研究[J].软件学报，2008，19(1)：48-61.
[2] KANUNGO T，MOUNT D M，NETANYAHU N，et al. A local search approximation algorithm for K-means clustering [J].Computational Geometry，2004(28)：89-112.
[3] 汪中.一种优化初始中心点的 K-means 算法[J].模式识别与人工智能，2009，22(2)：300-304.
[4] HALKIDI M，VAZIRGIANNIS M. Clustering validity assessment：finding the optimal partitioning of a data set [C]. In：Proc.of the 1st IEEE Int'I Conf.on Data Mining. 187-194.
[5] 谈恒贵.数据挖掘中的聚类算法[J].微型机与应用，2005(1).

(收稿日期：2010-09-15)

作者简介：

苏喻，男，1984年生，硕士研究生，主要研究方向：文本聚类，语义搜索。

郑诚，男，1964年生，副教授，硕士生导师，主要研究方向：数据挖掘，语义 Web。

封军，男，1983年生，硕士研究生，主要研究方向：数据挖掘，语义 Web。