第 42 卷 第 1 期 计 算 机 工 程 2016 年 1 月
Vol. 42 No. 1 Computer Engineering January 2016

·人工智能及识别技术· 文章编号：1000-3428(2016)01-0203-07 文献标识码：A 中图分类号：TP301

初始点优化与参数自适应的密度聚类算法

戴阳阳，李朝锋，徐 华

（江南大学物联网工程学院，江苏 无锡 214122）

摘 要： 针对密度聚类算法 DBSCAN 无法处理变化密度的问题，提出一种初始点优化与参数自适应的改进算法。利用初始点优化方法确定全局密度最大的点，结合该点和数据集自身的特征，自适应得到 DBSCAN 算法聚类出当前簇所需要的合适参数。该算法能够为不同密度的簇自适应设置不同的参数，而且优先对高密度簇进行聚类，即能对变化密度的数据集进行聚类。实验结果表明，该算法可以发现任意形状、大小和变化密度的簇，解决数据重叠和簇内密度不均匀问题，具有较高的聚类准确率。

关键词： 初始点优化；自适应；变化密度；聚类；数据挖掘

中文引用格式： 戴阳阳，李朝锋，徐 华. 初始点优化与参数自适应的密度空间聚类算法[J]. 计算机工程，2016，42(1)：203-209.

英文引用格式： Dai Yangyang，Li Chaofeng，Xu Hua. Density Spatial Clustering Algorithm with Initial Point Optimization and Parameter Self-adaption[J]. Computer Engineering，2016，42(1)：203-209.

Density Clustering Algorithm with Initial Point Optimization and Parameter Self-adaption

DAI Yangyang，LI Chaofeng，XU Hua

（School of Internet of Things Engineering，Jiangnan University，Wuxi，Jiangsu 214122，China）

【Abstract】 Aiming at the problem that the Density Based Spatial Clustering algorithm of Application with Noise (DBSCAN) can not find clusters of varied densities，this paper proposes a density clustering algorithm with initial points optimization and parameter self-adaption. It uses the method of optimization initial points to find the maximum density point in the current global datasets，and adaptively calculates the parameters of DBSCAN for the current cluster with the features of the current maximum density point and current datasets. These parameters are found to be different with the other clusters' parameters，and the high-density cluster gets priority processed，so this algorithm can find clusters of varied density. Experimental results demonstrate that the improved algorithm can find clusters of arbitrary shape，size and density，enhance the ability to deal with overlapping data and uneven density in the cluster，and get clustering in higher accuracy.

【Key words】 initial point optimization；self-adaption；varied density；clustering；data mining

DOI： 10.3969/j.issn.1000-3428.2016.01.036

1 概述

聚类是一种流行的数据挖掘技术，聚类算法被广泛地应用在许多领域，包括模式识别、机器学习、图像处理、信息检索等，在数据挖掘中起到重要的作用[1]。它能将大量数据根据其相互间的相似度划分成若干个类或簇，簇内数据间的相似度相对较大，而不同簇的数据间的相似度相对较小[2]。许多学者对聚类进行了深入的研究，提出了许多不同的类型的聚类算法，包括层次聚类、划分聚类、基于密度的聚类等。

现有的聚类算法都有其各自的应用领域，同时也存在各自的不足。基于层次聚类的算法，可以发现任意形状的簇，但是对噪声敏感[3]；基于划分的聚类算法，如 K-means 算法[4]简单、效率高，但它只能发现球状簇，对初始点非常敏感；基于密度的聚类算法，如 DBSCAN（Density Based Spatial Clustering algorithm of Application with Noise）算法[5]可以发现

基金项目： 国家留学基金资助项目（201308320030）；江苏省自然科学基金资助项目（BK20140165）。

作者简介： 戴阳阳（1986－），男，硕士、CCF 会员，主研方向为人工智能、数据挖掘；李朝锋，教授；徐 华，副教授、博士后。

收稿日期： 2015-01-04 **修回日期：** 2015-03-09 **E-mail：** yydai168@qq.com

任意形状的簇,并且能够过滤掉噪声,但它对输入参数敏感,尤其是当数据密度有变化时,无法取得适合于不同密度数据的全局参数,此时就很难对数据进行正确的聚类。

本文针对 DBSCAN 算法不能对变化密度数据进行聚类的问题,在该算法的基础上加入初始点优化与参数自适应步骤,提出一种改进算法(OS-DBSCAN)。该算法可以聚类任意形状和任意大小的变化密度的数据集,并且参数的设置易于用户理解。

2 相关研究

DBSCAN 算法是经典的基于密度的聚类算法。它引入密度的概念,将稀疏数据与稠密数据隔开,稠密数据区域就是簇,其中密度定义是以一个数据点为中心,在其领域阈值(*Eps*)内的数据点的个数。密度若大于密度阈值(*MinPts*),则该中心点就是核心点。DBSCAN 可以发现任意大小和形状的簇,并有一定的去噪声能力,但是当簇之间的密度变化较大时,为 DBSCAN 算法设置合适的参数较困难。文献[6]提出利用 k-dist 图和 DK 分析,对数据集中的不同密度层次自动选择一组 *Eps* 值,分别调用 DBSCAN 算法,通过不同的 *Eps* 值能够找到不同密度的簇。但是需要事先绘制 k-dist 图,并且 *k* 值不容易确定。文献[3]针对聚类密度不均匀的问题,提出了一种基于划分改进的蚂蚁聚类算法和基于数据分区的 DBSCAN 相混合的算法,由于引入蚂蚁聚类算法,因此不能快速地计算出 *Eps* 和 *MinPts*,影响了算法的效率。文献[7]针对 DBSCAN 无法处理变化密度问题,提出一种基于变化密度的自适应空间聚类方法,采用密度变化率来识别不同密度的簇之间的边界,且运行时自动调整参数的值。该算法可以发现变化密度的簇,参数也有一定的自适应性,但是对簇内数据点分布不均匀的簇,聚类效果不佳,容易将一个自然簇分为多个簇,并且算法运行时间随着 *k* 的增加而变长。

以上各种聚类算法都对 DBSCAN 算法进行了相应的改进,尝试着解决不同形状、不同大小和不同密度聚类的问题,但是依然存在各种不足,并且算法的适用范围有限,参数不易设置。

3 OS-DBSCAN 算法

3.1 设计思想

DBSCAN 算法不能处理变化密度的数据集,最主要的原因是它的 2 个参数 *Eps* 和 *MinPts* 是全局参数,都是由用户设置的,并且在运行时不会改变。假设 *Eps* 不变化,当处理变化密度数据集时,*MinPts* 太小就可能将噪声当做簇中的数据,或者 *MinPts* 太大就可能将低密度的数据当做噪声,所以,需要对每个簇设置适合各个簇的参数。本文算法最重要的思想是 DBSCAN 的初始点优化,即在一定的参数设置下寻找到密度最大的簇中的密度最大的点,并以此作为初始点。由于空间数据中同一簇的点的密度大致相同,它们近似高斯分布[8],因此全局密度最大的点所在的簇就可能是密度最大的簇,也就找到了最大密度的簇和该簇的初始点。然后通过初始点结合自适应方法产生出适合所在簇的 *Eps* 和 *MinPts*,再使用改进的 DBSCAN 算法扩张和聚类,在聚类出一个簇之后停止并将这个簇的所有的点从数据集中删除,然后在剩下的数据集中寻找下一个密度最大簇的初始点,直到找到 *k* 个簇,再进入多核心集凝聚,就是将未聚类的点按照就近原则找到离自己最近的核心点并加入该簇,输出聚类结果,算法结束。这样就能解决变化密度聚类的问题,从密度最大的簇开始聚类,密度小的都当做噪声,第一个簇处理完之后再根据剩下的数据集计算出适合下一个簇的不同的参数,直到聚类出 *k* 个簇为止。

3.2 初始点优化

初始点优化一般在 K-means 算法的改进中会经常被使用到,因为初始点是随机选择的,所以对于 K-means 的改进来说优化初始点是很重要的。对于 DBSCAN 来说,初始点的选择与算法结果的相关性不太明显,所以,学者们没有对此有较多的研究。受文献[9]启发,本文提出改进的初始点优化来产生更适合变化密度的簇的参数。

例如一个包含 100 个点的数据集 *A* 的点图,其中包括 3 个簇,它们各自簇的密度不同,而且簇之间的距离很近,使用传统的 DBSCAN 算法是无法聚类出 3 个簇的,图中的聚类结果是使用的 OS-DBSCAN 算法。初始点优化的具体步骤如下:

Setp1 求得各个数据点之间的相互距离,然后将所有的距离数据从小到大排序。

Setp2 得到所有距离序列中最小的部分所占全部序列的百分比。

Setp3 统计出最小距离数据中出现次数最多的数据点,该点就是当前簇的初始点。

通过上述方法得到的当前簇的初始点的实际意义是,在以 *percent* 比例对应的距离值为半径的圆,当以初始点为圆心时,该圆所能包含的数据点最多,也就是说该初始点是 *percent* 条件下的全局密度最大点。

找到当前簇的初始点之后,求得 *percent* 比例中的距离的平均值 *roughEps*,再求得所有与初始点距离小于 *roughEps* 的数据点的距离的平均值,该平均

值就是当前簇的 *Eps*。而此时以 *Eps* 为半径,以密度最大点为圆心,通过包含的数据点的数目就能计算出 *MinPts*,如图 1 所示。这是经过大量实验之后得到的一种确定 *Eps* 和 *MinPts* 相对比较合适的方法。

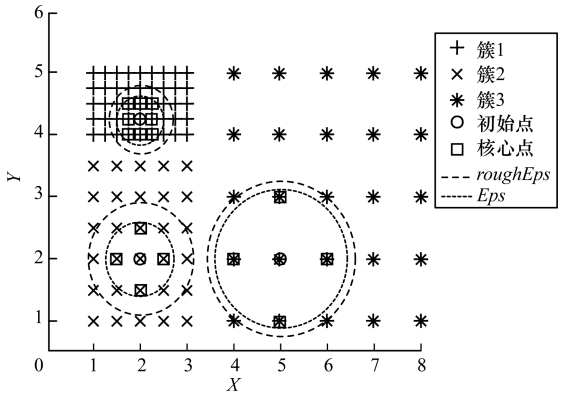


图 1　数据集 *A* 数据点聚类结果

初始点优化可以优先寻找到密度最高的簇的初始点,并且只将密度一致的数据点标记为同一簇。由于密度是从高到低的,并不会影响其他簇的聚类,因此可以对变化密度的数据集聚类。数据集 *A* 首次距离表排序结果和数据点统计次数如图 2 和图 3 所示,其中,*Dist* 为 2 个点之间的距离值;All distacnes 表示所有两点之间距离值组成的曲线。

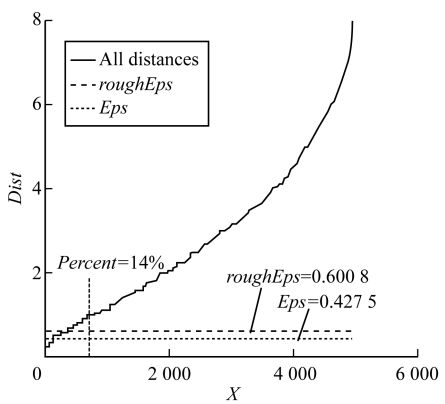


图 2　数据集 *A* 首次距离表排序结果

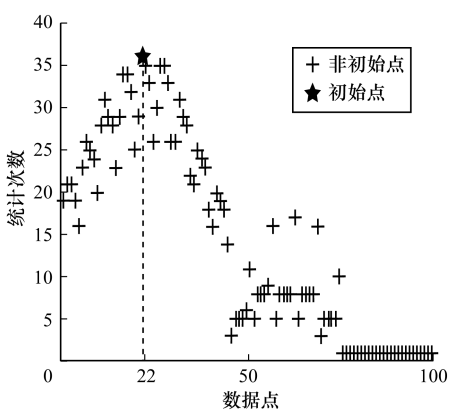


图 3　数据集 *A* 首次数据点统计次数

3.3　参数自适应

参数的自适应能够在程序运行中,动态地产生更加精准的 *Eps* 和 *MinPts*,使 OS-DBSCAN 算法更加精准地对变化密度数据集进行聚类,获得更好的聚类结果。

数据集 *A* 有 100 个数据点,$A=\{a_1,a_2,\cdots,a_n\}$ $(n=100)$。数据集 *A* 的距离表记为 *distTable*,$distTable=\{dist(a_1,a_2),dist(a_1,a_3),dist(a_i,a_j)\}$ $(i=1,2,\cdots,n;j=1,2,\cdots,n;i\neq j)$,$n$ 为数据集中的数据点个数,距离表中值的数目为 $n\times(n-1)/2$,*distTable* 排序后的距离排序记为 *Array*,$Array(percent)=\{distArray_1,distArray_2,\cdots,distArray_{roughNum}\}$ $(roughNum=percent\times n\times(n-1)/2)$,$Array(percent)$ 记为序列中值最小的 *percent* 比例部分,每一个 *distArray* 对应 2 个不同的数据点 a。

percent 由式(1)所得,是因为要确保在距离最小的部分中统计产生初始点。其中,引入倍率参数 β 对 *Eps* 进行调节,获得的 *percent* 部分数据的个数是 n 的 β 倍,β 由用户设定,$\beta>0$。

$$percent=\begin{cases}\dfrac{n\times\beta}{n\times(n-1)/2} & \dfrac{n\times\beta}{n\times(n-1)/2}\leqslant 0.2\\ 0.2 & \dfrac{n\times\beta}{n\times(n-1)/2}>0.2\end{cases}\tag{1}$$

roughEps 由式(2)所得。

$$roughEps=\frac{\sum_{i=1}^{roughNum}distArray_i}{roughNum}\tag{2}$$

数据集 *A* 簇 1 的初始点由图 3 可知是第 22 个点,记 a_{init} 为当前簇的初始点,则当前簇的 *Eps* 由式(3)所得,*epsNum* 记为与初始点与数据集其他各点之间距离小于等于 *roughEps* 的数据点的个数。

$$Eps=\frac{\sum_{i=1}^{n}dist(a_{init},a_i)}{epsNum}$$

$$dist(a_{init},a_i)\leqslant roughEps,i\neq init\tag{3}$$

此处需要说明的是,每一个簇的 *Eps* 是根据当前数据集自适应的得出的适合于当前簇的 *Eps*。随着一个簇被聚类出来后,该簇的所有数据将从数据集中删除,由于每个被除去的簇的数据点的数量和密度都不同,因此不能简单地用单一的 *Eps*。但是用百分比就能很好地找到当前密度最高的簇,并且自适应地找到合适的 *Eps*,因为密度高的数据点之间的距离相较于密度低的要小,所以在这些高密度数据的总是排在距离排序中的前面,由 *roughEps* 就能统计出其中出现最多的初始点,继而得出适合于每个簇不同的 *Eps*。当一个簇聚类结束,则用重新生成的数据集使用以上方法自适应更新 *Eps*。

得到当前簇的 *Eps* 之后，由 *Eps*、初始点和数据集 *A* 自适应的产生 *MinPts*。本文提出不同的 *MinPts* 概念，是以初始点为中心，*Eps* 为半径的圆中含有的其他数据点的个数再乘以参数 α。引入参数 α 的目的是由于此时的 *MinPts* 是由处在密度最高出的初始点产生的，这个 *MinPts* 是明显高于当前簇的平均密度的，因此引入密度参数 α 对 *MinPts* 进行比例调节，α 由用户设定，$0<\alpha\leqslant 1$。记 $ptsNum(init,i)$ 用于表示初始点 a_{init} 与第 i 个数据点的距离是否在初始点的 *Eps* 领域内，如式(4)所示。

$$ptsNum(init,i)=\begin{cases}1 & dist(a_{init},a_i)\leqslant Eps\\ 0 & dist(a_{init},a_i)>Eps\end{cases},i\neq init \tag{4}$$

当前簇的初始 *MinPts* 由式(5)所得。

$$MinPts=\alpha\times\sum_{i=1}^{n}ptsNum(init,i) \tag{5}$$

当算法运行到扩展核心点之前需要对 *MinPts* 进行自适应更新，就是通过将当前簇之前得到的所有核心点的密度除以核心点的个数再乘以参数 α。当第 i 个数据点为当前第 k 簇的核心点时，记 $coreNum_i=k$；当不是核心点时，记 $coreNum_i=0$。则更新后的 *MinPts* 由式(6)所得。

$$MinPts=\alpha\times\frac{\sum_{j=1}^{n}coreNum_j\times\sum_{i=1}^{n}ptsNum(j,i)}{\sum_{l=1}^{n}coreNum_l} \tag{6}$$

在算法运行中根据当前数据的情况自适应地产生适合于当前簇的 *MinPts*，使其针对变化密度的簇产生不同的适合于当前簇的 *MinPts*，再结合不同的 *Eps* 就可以将不同密度的簇聚类出来，也就是给每一个不同密度的簇不同的 *Eps* 和 *MinPts*。

3.4 算法描述

因为 OS-DBSCAN 算法是基于 DBSCAN 的基础改进的，所以继承了 DBSCAN 的相关定义。假设空间数据集 *U* 含有 *n* 个样本数据，它是 *m* 维空间 $\mathbf{R}^m$ 中点的集合，则数据 *x* 可以表示为 $x=(x_1,x_2,\cdots,x_m)$。

算法描述如下：

输入 有 *n* 个数据点的数据集 *U*，聚类个数 *k*，密度参数 α，倍率参数 β

输出 *k* 个簇的集合

Setp1 计算数据集 *U* 的所有对象距离数据表 *distTable*，得到 *distTable* 从小到大排序的 *Array*。

Setp2 通过 *Array* 的 *percent* 范围内出现最多的数据点标记，得到初始点 *init*。

Setp3 根据初始点 *init* 计算出当前簇的 *Eps* 和初始 *MinPts*，得到当前簇的以 *init* 为圆心的初始簇点。

Setp4 计算当前簇的每一个点的密度，若大于 *MinPts*，则标记为核心点，核心点的 *Eps* 领域内的点标记为当前簇标。

Setp5 根据当前簇核心点的平均 *MinPts*，更新 *MinPts*，重复 Setp4 直到当前簇的点个数不再增加。

Setp6 从数据集中去掉当前簇的点，当前簇标加 1，重复 Setp1 ~ Setp5 直到当前簇标为 $k+1$。

Setp7 给每一个没有被标记的点附上与其最近的标记的点的簇标，聚类算法结束。

此算法在运行完之后并不会出现孤立点，每个点都会有属于的簇，这样做的理由是，有些数据集是不存在噪声点，但是有些数据采集中会有噪声点，所以，本文算法将这个选择权交给了用户。对于有噪声的数据集，只需要在算法结束后，加一个过滤噪声步骤，即采用算法过程中产生的 *k* 对 *Eps* 和 *MinPts* 中最小的一对，对所有数据进行 DBSCAN 算法，就可以得到的噪声点。

在 DBSCAN 算法中，初始点没有优化选取，簇的聚类没有先后顺序，只能依据数据点的空间间隙用于区别不同的簇。本文算法的目的是改进 DBSCAN 算法使其适用于变化密度的数据。初始点优化能够让密度最高的簇从剩余数据集中优先聚类出，并且即使与周围低密度的簇相连也会由更加精准的参数加以过滤而只聚类属于最高密度的簇。通过初始点优化和参数自适应方法，可以得到更精准的贴近于各个不同密度簇的参数，并且将变化密度簇依据簇自身密度从高到低逐个聚类出来，从而能够应对变化密度的问题，并且提高了聚类的准确率，确保类内相似度最大，类间相似度最小。用实例来阐述算法，数据如图 1 所示。

根据本文算法，首先计算所有数据点之间的距离并且排序，然后计算得初始点，如该实例图中簇 1 中被小圆圈围住的数据点，再计算出 *Eps* 和 *MinPts* 并以此进行聚类，如图 1 中内环的虚线围住的数据，接着更新参数并且扩展当前簇直到不再增加，如图 1 中所示，簇 1 的所有数据都变成了“小加号”，将得到的簇从数据集中删除，最后重复以上的步骤直到聚类出 *k* 个簇为止，如图 1 中形成的 3 种符号所表示的聚类结果。而若是用 DBSCAN 算法只能得到一个簇。

3.5 算法分析

本文算法由于优化初始中心点时需要计算全局

的所有对象之间的距离,因此时间主要花费在生成 *distTable* 上,时间复杂度为 $O(n^2)$。

3.6　参数选择

算法有 4 个输入参数:*dataset*,k,α 和 β。其中,*dataset* 是数据集。参数 k 是簇类的个数,当用户知道具体的簇类个数时,可以直接设置为对应的值,当未知时,可以相对设置的偏大一些,因为密度大的内聚高的簇依然会优先聚类。倍率参数 β 决定 *percent* 的值,主要影响 *Eps* 大小。β 选择偏大则 *Eps* 越大,β 偏小则 *Eps* 越小,所以,一般设置为 $\beta \leqslant n/10$。密度参数 α 主要作用于数据集有不同密度的簇相连接的时候,去分离开不同密度的簇。当簇之间重叠程度高时可以相应增加 α 的值,使当前簇只扩展与当前簇密度相近的数据点。若增大 α 时,扩展能力降低,容易将一个自然簇分分裂,若 α 过小,则容易合并低密度的相连接的自然簇。所以,一般情况下 α 设置在 0.8 左右。本文算法由于有自适应参数部分,因此对于输入参数的设置不是特别敏感。

4　实验结果与分析

本节通过实验对 OS-DBSCAN 算法进行评估,分别采用人工数据集、CHAMELEON[10] 和 SAVDBSCAN[7] 进行比较实验,检验其对形状、大小、变化密度和重叠的数据集的聚类能力,通过真实数据集检验其有效性。本文实验环境为:Intel i7 2.4 GHz,8 GB 内存,Mac OS X 10.10 操作系统,Matlab 2014a。

4.1　任意形状和大小的人工数据集

此实验数据为 CHAMELEON 算法的实验数据,分别为数据集 t7.10k.dat 和 t8.8k.dat[10]。图 4 和图 5 是 OS-DBSCAN 算法聚类结果,与文献[7]和文献[10]中的结果一致。可以发现,本文算法可以对任意形状和大小的数据集进行聚类。OS-DBSCAN 对 t7.10k 和 t8.8k 数据集的参数 k,α 和 β 分别为 18,0.6,30 以及 10,0.5,30。

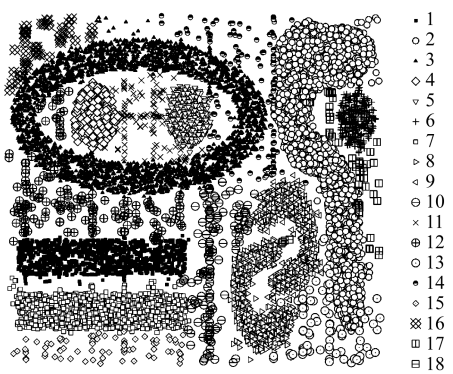
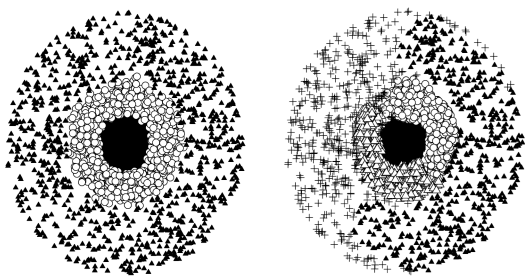


图 4　t7.10k.dat 聚类结果

图 5　t8.8k.dat 聚类结果

4.2　变化密度和簇连接的人工数据集

此实验数据为自己产生的数据集,由 3 000 个点组成的 3 个嵌套的同心圆,每个圆环有 1 000 个点,密度由中心向外渐渐变大,同一环的密度基本一致,但是同一环中的数据点的分布并不均匀。图 6 是 OS-DBSCAN 和 SAVDBSCAN 算法的结果。可以发现,在文献[7]中,SAVDBSCAN 算法的聚类效果很好,但是当数据点个数从 2 000 减少到 1 000 后,簇内数据点分布变得不均匀,效果就变差了,而本文算法的聚类结果依然很好。这表明本文算法可以对变化密度并且簇连接的数据集进行聚类,并且对簇内数据点分布不均有一定的适应能力。OS-DBSCAN 算法参数 k,α 和 β 分别是 3,0.5,30;SAVDBSCAN 算法参数 k 和 α 分别是 50,0.5。



(a)OS-DBSCAN算法　　(b)SAVDBSCAN算法

图 6　OS-DBSCAN 和 SAVDBSCAN 算法的聚类结果

4.3　有重叠的人工数据集

此实验数据为自己产生的数据集,由 550 个数据点组成的重叠程度不同的 2 组数据集,第 2 行各个簇之间的重叠程度比第 1 行大。当各簇数据重叠程度变大时,不同算法的聚类结果如图 7 所示。可以发现,K-means 和本文算法呈现出对数据重叠较好的鲁棒性,而 DBSCAN 和文献[7]算法对于数据重叠增大时的聚类效果不佳。DBSCAN 算法的参数分别是 0.07,10 和 0.06,10,文献[7]算法的参数分别是 30,0.5 和 40,0.2,本文算法参数分别是 3,0.7,20 和 3,0.7,20。

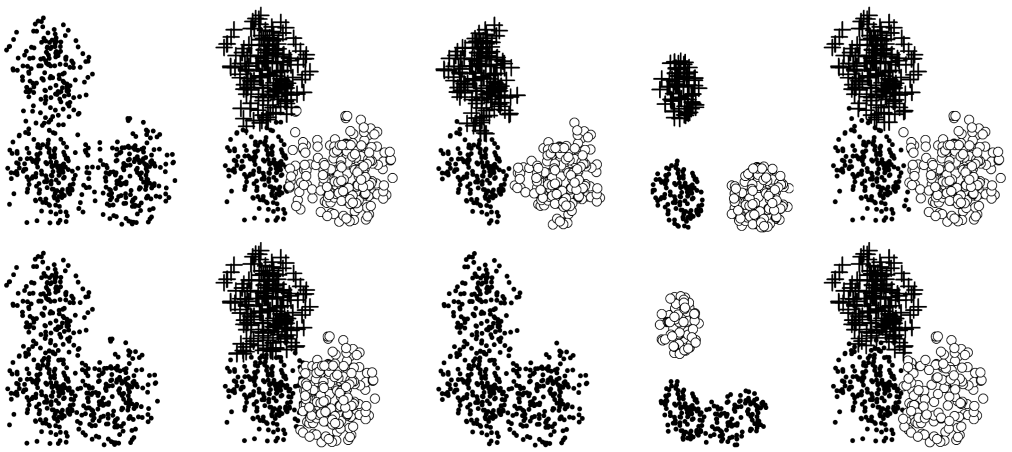


图 7 不同重叠程度的聚类结果比较

4.4 真实数据集

为了进一步说明 DBSCANN 算法的有效性,本节将采用真实数据集进行聚类实验。真实数据集分别来源于 USPS(the United States Postal Service,由 16×16 维灰度图像构成,每个样本为 256 维向量,共包含 7 291 个训练样本,2 007 个测试样本,本节的实验只取用其测试样本)和 UCI 数据库(http://archive.ics.uci.edu/ml,本节选取其中的 wine 和 iris 数据集),实验所用数据集具体如下:

(1)usps(0,1):USPS 测试样本中数字 0 和 1,共 623 个数据。

(2)usps(0,2):USPS 测试样本中数字 0 和 2,共 557 个数据。

(3)usps(4,9):USPS 测试样本中数字 4 和 9,共 377 个数据。

(4)usps(5,8):USPS 测试样本中数字 5 和 8,共 326 个数据。

(5)usps(0,1,5):USPS 测试样本中数字 0,1,5,共 783 个数据。

(6)wine:UCI 数据库,包含 178 个 13 维数据,分 3 类。

(7)iris:UCI 数据库,包含 150 个 4 维数据,分 3 类。

不同算法的实验结果如表 1 所示,其中,OS-DBSCAN 算法参数(k,α,β)、MulCA 算法[11]参数(α,σ_0)、SAVDBSCAN 算法参数(k,α)、K-means 算法参数(k)和 DBSCAN 算法的参数(*Eps*,*MinPts*)分别标注在准确率的后面括号中,加粗数据表示其中的最佳值。由表 1 可以看出,DBSCAN 算法在数据集 usps(0,2),usps(4,9),usps(5,8),usps(0,1,5),wine 和 iris 上的结果比其他算法都好,只有在 usps(0,1)数据集上稍逊于 MulCA 算法,显示了其出色的聚类能力。

表 1 真实数据聚类准确率 %

数据集	OS-DBSCAN 算法	MulCA 算法	SAVDBSCAN 算法	K-means 算法	DBSCAN 算法
usps(0,1)	99.20(2,0.7,30)	**99.84**(0.3,0.08)	94.86(38.0.8)	98.56(2)	83.31(7.6,11)
usps(0,2)	**96.59**(2,0.7,30)	94.61(0.4,0.03)	92.10(24,0.3)	82.23(2)	64.27(13.7,12)
usps(4,9)	**87.27**(2,0.7,22)	75.60(0.5,0.2)	68.70(11,0.3)	80.11(2)	52.52(12.5,11)
usps(5,8)	**89.26**(2,0.9,15)	88.96(0.4,0.1)	57.06(22,0.2)	86.81(2)	50.31(13.5,11)
usps(0,1,5)	**96.42**(3,0.7,26)	81.48(0.6,0.04)	69.22(31,0.4)	81.48(3)	66.28(7.6,11)
wine	**73.03**(3,0.7,10)	71.91(0.3,0.05)	62.92(31,0.4)	72.20(3)	60.67(37,5)
iris	**98.00**(3,0.9,7)	96.00(0.3,0.05)	68.67(9,1)	91.33(3)	80.67(0.4,5)

4.5 结果分析

实验结果表明,结合优化初始点和自适应参数的 DBSCAN 算法可以在对任意形状和任意大小的数据集进行聚类,并且可以对变化密度的数据集进行聚类,还可以应对一定的数据重叠和簇内密度不均匀,提高聚算法的准确率。

5 结束语

本文针对 DBSCAN 算法不能处理变化密度的缺点,提出了改进算法 OS-DBSCAN。将初始点优化引入 DBSCAN 算法,同时引入倍率参数 β 和密度参数 α 用于参数自适应。参数 β 和 α 都是比例值,

方便了参数的理解和设置。通过此自适应机制,本文算法在运行时可以根据数据的分布情况,自动调节 DBSCAN 参数的值,从而发现更好的自然簇。实验结果表明,OS-DBSCAN 可以发现任意形状、大小和密度的簇,并且对簇内不均匀、簇间重叠有一定的聚类能力,同时提高聚类算法的准确率。下一步工作将对算法进行改进,使其能通过分析数据自身的情况自动选择合适的簇类个数 k,并将增量数据动态增加到数据集中,避免重复计算整个 *distTable*,降低运行时间。

参考文献

[1] 孙吉贵,刘　杰,赵连宇. 聚类算法研究[J]. 软件学报,2008,19(1):48-61.

[2] Chen M S,Han Jiawei,Yu P S,et al. Data Mining:An Overview from a Database Perspective [J]. IEEE Transactions on Knowledge and Data Engineering,1996,8(6):866-883.

[3] Jiang Hua,Li Jing,Yi Shenghe,et al. A New Hybrid Method Based on Partitioning-based DBSCAN and Ant Clustering[J]. Expert Systems with Applications,2011,38(8):9373-9381.

[4] MacQueen J. Some Methods for Classification and Analysis of Multivariate Observations[C]//Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley,USA:University of California Press,1967:281-297.

[5] Ester M,Kriegel H P,Sander J,et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise [C]//Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland,USA:AAAI Press,1996:226-231.

[6] Liu Peng,Zhou Dong,Wu Naijun. VDBSCAN:Varied Density Based Spatial Clustering of Applications with Noise[C]//Proceedings of International Conference on Service Systems and Service Management. Washington D. C.,USA:IEEE Press,2007:1-4.

[7] 杨亚军,张坤龙,杨晓科. 基于变化密度的自适应空间聚类方法研究[J]. 计算机工程,2014,40(8):58-63,69.

[8] 陈　刚,刘秉权,吴　岩. 一种基于高斯分布的自适应 DBSCAN 算法[J]. 微电子学与计算机,2013,30(3):27-30.

[9] 汪　中,刘贵全,陈恩红. 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):299-304.

[10] Karypis G,Han E H,Kumar V. Chameleon:Hierarchical Clustering Using Dynamic Modeling [J]. Computer,1999,32(8):68-75.

[11] 马儒宁,王秀丽,丁军娣. 多层核心集凝聚算法[J]. 软件学报,2013,24(3):490-506.

编辑　金胡考

(上接第 202 页)

参考文献

[1] 刘建国,周　涛,汪秉宏. 个性化推荐系统的研究进展[J]. 自然科学进展,2009,19(1):1-4.

[2] 项　亮. 推荐系统实践[M]. 北京:人民邮电出版社,2012.

[3] Sarwar B,Karypis G,Konstan J,et al. Item-based Collaborative Filtering Recommendation Algorithms[C]//Proceedings of the 10th International Conference on World Wide Web. New York,USA:ACM Press,2001:285-295.

[4] Breese J S,Heckerman D,Kadie C. Empirical Analysis of Predictive Algorithms for Collaborative Filtering [C]//Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence. [S. 1.]:Morgan Kaufmann Publishers Inc.,1998:43-52.

[5] Karypis G. Evaluation of Item-based Top-n Recommendation Algorithms [C]//Proceedings of the 20th International Conference on Information and Knowledge Management. New York,USA:ACM Press,2001:247-254.

[6] 陈天昊,帅建梅,朱　明. 一种基于协作过滤的电影推荐方法[J]. 计算机工程,2014,40(1):55-58,62.

[7] 陈博文,刘功申,张浩霖,等. 融合标签传播和信任扩散的个性化推荐方法[J]. 计算机工程,2014,40(12):33-38.

[8] 韦素云,业　宁,杨旭兵. 结合项目类别和动态时间加权的协同过滤算法[J]. 计算机工程,2014,40(6):206-210.

[9] 邓爱林,朱扬勇,施伯乐. 基于项目评分预测的协同过滤推荐算法[J]. 软件学报,2003,14(9):1621-1628.

[10] 熊忠阳,刘　芹,张玉芳,等. 基于项目分类的协同过滤改进算法[J]. 计算机应用研究,2012,29(2):493-496.

[11] 王　茜,段双艳. 一种改进的基于二部图网络结构的推荐算法[J]. 计算机应用研究,2013,30(3):771-774.

[12] 高全力,高　岭,杨建锋,等. 融合影响因子的加权协同过滤算法[J]. 计算机工程,2014,40(8):38-42.

[13] Wikipedia. 倒排索引[EB/OL]. (2013-03-12). http://zh.wikipedia.org/wiki/%E5%80%92%E6%8E%92%E7%B4%A2%E5%BC%95.

[14] Wikipedia. 长尾效应[EB/OL]. (2015-01-11). http://zh.wikipedia.org/wiki/%E9%95%BF%E5%B0%BE.

编辑　索书志