

基于密度加权的粗糙 K-均值聚类改进算法

郑超 苗夺谦 王睿智

(同济大学计算科学与技术系 上海 201804)

摘要 针对粗糙 K-均值聚类算法中类均值计算式的特点,提出了一种改进的粗糙 K-均值算法。改进后的算法基于数据对象所在区域的密度,在类的均值计算过程中对每个对象赋以不同的权重。不同测试数据集的实验结果表明,改进后的粗糙 K-均值算法提高了聚类的准确性,降低了迭代次数,并且可以有效地减小孤立点对聚类的影响。

关键词 聚类算法, 粗糙 K-均值, 密度, 孤立点

中图分类号 TP311 文献标识码 A

Improved Rough K-means Clustering Algorithm with Weight Based on Density

ZHENG Chao MIAO Duo-qian WANG Rui-zhi

(Department of Computer Science and Technology, Tongji University, Shanghai 201804, China)

Abstract According to the feature of the calculation of means in Rough K-means algorithm, an improved Rough K-means algorithm was proposed. The new algorithm introduces weights to the calculation of means, which is based on the density of each point. The experiments show that the new algorithm improves the clustering accuracy and reduces the iteration times as well as the outliers' influence.

Keywords Clustering algorithm, Rough K-means, Density, Outlier

聚类就是按照某个相似性测度将未标记的样本集分成若干个类的过程,并使同一类中的样本尽可能的相似,不同类中的样本尽可能的不相似。聚类是数据挖掘、模式识别等研究方向的重要研究内容之一,在识别数据的内在结构方面具有极其重要的作用^[1]。

目前,国内外学者已经提出许多聚类算法,常见的有层次型聚类、划分型聚类、密度型聚类、网格型聚类等。在划分型聚类算法中,传统的硬划分方法将每个数据对象严格地划分到某一个类中,并取欧氏距离为对象之间的距离度量。而现实中类与类之间不一定存在确定的边界,对象的类属存在着一定的过渡,即某个对象对于类并非只有属于和不属于两种状态。

粗糙集理论^[2]作为研究不确定知识的数学工具,具有处理聚类不确定性的能力。Lingras^[3]最早将粗糙集理论引入到聚类算法中,提出了粗糙 K-均值算法。该算法将确定属于某类的对象放到该类的下近似中,而将可能属于某类的对象放到该类的上近似中。粗糙 K-均值算法能较好地处理不确定性对象,已被广泛地应用到了生物信息学等领域^[4]。

针对粗糙 K-均值算法计算过程的特点,文献[5]以相对距离作为判定对象是否属于类的下近似的条件对粗糙 K-均值算法进行了改进。文献[6]通过进化算法决定了粗糙 K-均值算法中的最优输入参数。文献[7]基于粗糙 K-均值算法,结合经典的 K-中心算法,提出了粗糙 K-中心算法。但现有的改进算法都没有考虑对象所有区域的密度,并且认定属于

同一上近似或下近似中的每个数据对象的权重是相同的。本文根据数据对象所在区域的密度,对粗糙 K-均值算法中类均值的计算式加以改进,并通过实验与改进算法前的聚类结果进行比较。

1 粗糙 K-均值算法

粗糙集理论的核心思想就是用上近似和下近似来描述不确定事物^[2]。在经典粗糙集理论中,如果集合 X 可以精确定义,当且仅当 X 能表示为基本等价类的并集,否则 X 通过下近似和上近似方式刻画。集合 X 关于属性集 R 的下近似定义为 $\underline{R}X = \{a: a \in U, [a] \in X\}$,表示确定属于 X 的对象集合;上近似为 $\overline{R}X = \{a: a \in U, [a] \cap X \neq \emptyset\}$,表示可能属于 X 的对象集合。

Lingras 在基于划分的经典 K-均值算法的基础上,结合了粗糙集中的上近似和下近似的思想,提出了粗糙 K-均值算法,其算法有以下特性^[3]:

- 1) 一个数据对象最多只能属于一个类的下近似;
- 2) 如果一个对象不属于某一个类的下近似,那么它一定属于两个或两个以上个类的上近似;
- 3) 每个类的下近似是它的上近似的子集。

设数据集 $U = \{X_i, i = 1, \dots, N\}$, \overline{C}_k 和 \underline{C}_k 分别代表类 C_k 的上近似和下近似集合, C_k^B 代表类 C_k 的边界区域集合,即该类的上近似和下近似的差集。 C_k 的均值 m_k 按式(1)计算:

到稿日期:2008-04-28 本文受国家自然科学基金项目(60775036, 60475019),高等学校博士学科点专项科研基金(20060247039)资助。

郑超(1985-),男,硕士生,主要研究方向为聚类分析、模式识别, E-mail: senchat85@yahoo.com.cn; 苗夺谦(1964-),男,教授,博士生导师, CCF 会员,主要研究方向为模式识别、数据挖掘、粗糙集、主曲线等; 王睿智(1972-),女,博士生,主要研究方向为聚类分析、模式识别、粒计算等。

$$m_k = \begin{cases} \frac{\omega_l \sum_{x_i \in \underline{C}_k} X_i + \omega_b \sum_{x_i \in \overline{C}_k^b} X_i}{|\underline{C}_k| + |\overline{C}_k^b|} & (C_k^b \neq \phi) \\ \frac{\sum_{x_i \in \underline{C}_k} X_i}{|\underline{C}_k|} & (C_k^b = \phi) \end{cases} \quad (1)$$

其中, $|\underline{C}_k|$ 是类 C_k 中下近似集合的对象个数, $|\overline{C}_k^b| = |\overline{C}_k - \underline{C}_k|$ 是类 C_k 的边界区域的对象个数。 ω_l, ω_b 分别为上近似和边界区域的权重值, 且 $\omega_l + \omega_b = 1$ 。

粗糙 K-均值算法首先将数据集中每个对象划分到任意一个类的下近似集合中, 然后根据式(1)计算每个类的均值。接下来对于每个数据对象找出与它距离最近的均值点, 并计算其最小距离。如果存在其它类的均值点和该对象之间的距离与最小距离之差小于阈值 ϵ , 则将该对象加入到这些类的上近似集合中, 否则将该对象加入到最近均值点所对应的类的下近似中。接下来重新计算每个类的均值, 以同样的方法将对象指派到相应的类中。不断重复这个过程, 直到每个类的均值不变。

2 改进的粗糙 K-均值算法

在粗糙 K-均值算法的迭选过程中, 均值点的计算依据类边界是否为空分两种情况讨论。如果边界不为空, 则分别计算下近似的均值和边界区域的均值, 最后再对两个均值赋以不同权重, 求出类的均值。在下近似和边界区域的均值的计算过程中, 算法都只是将对象相加再除以相应区域内的对象个数, 即认定每个数据对象的权重是相同的。

但在实际中, 位于数据密集区域和位于非密集区域的对象对类均值计算的重要性是不同的。如类中有一点是离同类较远的孤立点, 很可能会使类均值点与类中大多数对象有较大偏离, 必然会影响算法的迭代效果。若在对类均值的计算过程中根据每个数据点所在区域的密度调整权重, 得出的均值点将能更好地代表这个类, 算法的准确性也将会得到改善。

2.1 新的均值计算式

对于一个有 N 个样本的数据集 $U = \{X_i, i=1, \dots, N\}$, 定义点 X_i 处的密度函数为:

$$f_i = \frac{1}{\sum_{j=1}^N e^{-\frac{\|X_i - X_j\|^2}{\delta^2}}} \quad (2)$$

其中 δ 表示邻域有效半径, 这里我们取 N 个样本的均方根距离的二分之一, 即:

$$\delta = \frac{1}{2} \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^N \sum_{j=1}^N \|X_i - X_j\|^2} \quad (3)$$

由式(2)和式(3)可见, X_i 周围聚集的样本点越多, 则对象 X_i 的密度值 f_i 越大。

由于每个数据对象对类的影响是不同的, 所以在类均值的计算过程中, 我们将密度值较小的对象赋予较小的权重, 将密度值较大的对象赋予较大的权重。对象 X_i 的权重 p_i 定义为:

$$p_i = \begin{cases} \frac{f_i}{\sum_{x_j \in \underline{C}_k} f_j} & (X_i \in \underline{C}_k) \\ \frac{f_i}{\sum_{x_j \in \overline{C}_k^b} f_j} & (X_i \in \overline{C}_k^b) \end{cases} \quad (4)$$

所以, 重新定义的类 C_k 的均值计算式为

$$m_k = \begin{cases} \frac{\omega_l \sum_{x_i \in \underline{C}_k} p_i X_i + \omega_b \sum_{x_i \in \overline{C}_k^b} p_i X_i}{\sum_{x_i \in \underline{C}_k} p_i + \sum_{x_i \in \overline{C}_k^b} p_i} & (C_k^b \neq \phi) \\ \frac{\sum_{x_i \in \underline{C}_k} p_i X_i}{\sum_{x_i \in \underline{C}_k} p_i} & (C_k^b = \phi) \end{cases} \quad (5)$$

2.2 聚类算法描述

输入: 数据集 U , 权值 ω_l, ω_b , 聚类数目 K 以及阈值 ϵ ;

输出: 聚类结果 $\{C_1, C_2, \dots, C_K\}$;

Step 1: 在数据集中任取 K 个点作为每个类的初始均值点;

Step 2: 对 $\forall X_i \in U$, 根据式(2)求得相应的 f_i ;

Step 3: 对 $\forall X_i \in U$, 找出与其距离最小的均值 m_k , 即 $d(X_i, m_k) = \min\{d(X_i, m_j), j=1, \dots, K\}$, 并且使 $X_i \in \overline{C}_k$ 。

如果 $\exists m_j$, 使得 $d(X_i, m_j) - d(X_i, m_k) \leq \epsilon$, 则使 $X_i \in \overline{C}_j$ 。否则使 $X_i \in \underline{C}_k$;

Step 4: 根据式(5)更新每个类的均值;

Step 5: 如果每个类的均值没有改变, 则聚类结束。否则转 Step 3。

3 实验

3.1 人工数据实验

实验中的人工数据由 15 个二维数据点组成, 其中包含一个孤立点。在给定聚类数目 $K=2, \omega_l=0.8, \omega_b=0.2$, 当 $\epsilon \in [0.05, 0.5]$ 时, 粗糙 K-均值算法改进前后的聚类结果如图 1 和图 2 所示。图中 \blacksquare 和 \bullet 代表两类各自下近似中的对象点, ∇ 代表同属于两类边界区域的对象点, $+$ 代表类的均值点。

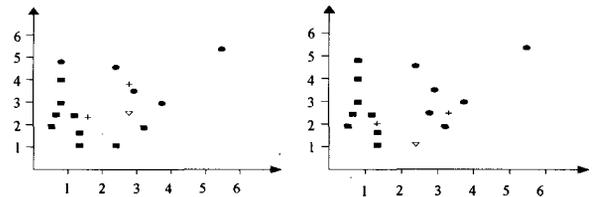


图 1 粗糙 K-均值算法的聚类结果 图 2 改进的粗糙 K-均值算法的聚类结果

这里我们用文献[8]提出的 DB(Davies-Bouldin) 指标来衡量人工数据的实验效果。DB 指标值即是类内距和类间距函数的比值, 好的聚类结果应使得比值尽可能地小。计算得到粗糙 K-均值算法改进前的聚类结果的 DB 值为 1.43, 算法改进后聚类结果 DB 值为 1.22。由实验结果不难看出, 改进后的粗糙 K-均值算法能够减小孤立点对类均值计算的影响, 提高了聚类效果。

3.2 UCI^[9] 数据集实验

为了进一步验证改进算法的聚类效果, 本文采用 UCI^[9] 数据集中的 Iris, Wine 和 Soybean 3 个常用数据集进行对比实验, 各数据集的特征如表 1 所示。由于这 3 个数据集都有已知的分类结果, 可以准确率来直观地评价聚类效果^[10]。

表 1 实验中涉及到的数据集

数据集名称	对象数目	分类属性	类别
Iris	150	4	3
Soybean	47	35	4
Wine	178	13	3

在给定 3 个数据集的聚类类别数目 $K, \omega_l=0.8, \omega_b=0.2$ 的条件下, 进行 10 次实验。粗糙 K-均值算法改进前后聚类的平均准确率如表 2 所示。

表 2 平均准确率比较

	粗糙 K-均值	改进的粗糙 K-均值
Iris	85.32%	87.12%
Soybean	78.90%	79.91%
Wine	69.11%	71.70%

由表 2 可以看出,改进后的粗糙 K-均值算法比改进前在聚类的准确率上平均提高了 1~2 个百分点。

为了验证改进后算法的效率,表 3 比较了算法改进前后 10 次实验的平均迭代次数。结果表明改进后的算法具有更快的收敛速度。

表 3 平均迭代次数比较

	粗糙 K-均值	改进的粗糙 K-均值
Iris	8.95	8.05
Soybean	7.20	5.87
Wine	11.02	9.67

结束语 本文基于数据对象所在区域的密度,提出了一种基于密度加权的粗糙 K-均值改进算法。新算法提高了聚类的准确性,降低了迭代次数,并减小了孤立点对聚类的影响,是一种有效的算法。但是计算所有数据对象的密度的时间复杂度是 $O(N^2)$,其中 N 是数据对象的个数。进一步的工作是如何降低对象密度计算的时间复杂度,提高计算速度,使算法能够运用到特大型数据的聚类中。

参考文献

[1] 孙吉贵,刘杰,赵连宇. 聚类算法研究. 软件学报,2008,1(19): 48-49

[2] Pawlak Z. Rough sets. International Journal of Information and Computer Sciences,1982,11:145-172

[3] Lingras P, West C. Interval set clustering of web users with rough k-means. Journal of Intelligent Information Systems,2004,23(1):5-1643

[4] Wang Ruizhi, Miao Duoqian, Li Gang, et al. Rough Overlapping Biclustering of Gene Expression Data // Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering, 2007:828-834

[5] Peters G. Some refinements of rough k-means clustering. Pattern Recognition,2006,39(8):1481-1491

[6] Mitra S. An evolutionary rough partitive clustering. Pattern Recognition Letters,2004,25(12):1429-1449

[7] Peters G, Lampart M. A Partitive Rough Clustering Algorithm. Rough Sets and Current Trends in Computing,2006,4259(1):658

[8] Davies D, Bouldin D. A Cluster Separation Measure. IEEE Trans, Pattern Anal,1979,1(2):224-227

[9] Blake C L, Merz C J. UCI repository of learning databases. <http://www.ics.uci.edu/~mllearn/MLRepository.html>

[10] Sun Y, Zhu Q M, Chen Z X. An iterative initial-points refinement algorithm for categorical data clustering. Pattern Recognition Letters,2002,23(7):880-883

(上接第 214 页)

下面给出结论:

定理 4.3 MPSO 的有效搜索时间复杂度为 $O(dt)$,要优于基本 PSO 的 $O(mt)$ 。

我们所说的有效搜索是指微粒沿着流形或者局部沿着切空间飞行。如果目标函数 E 仅定义在流形上,则很容易实现有效搜索。或者说,如果目标函数对非流形区域很敏感,则 MPSO 就需要更好的健壮性和收敛性。从这个角度讲,本文提出的新算法要优于非流形 PSO。

5 实验数据

在这一部分,我们给出一个例子来证明 MPSO 的性能。为了简化问题,我们利用一个一维流形 $y = \sin(\pi x)$, $x \in [-1, 1]$ 。目标函数为

$$\begin{cases} E(x, y) = x^2 + y^2 + |y - \sin(\pi x)| & H(x, y) \\ H(x, y) = 100(y^2 - x) + (1 - x)^2 \end{cases} \quad (20)$$

该函数曲面如图 3 所示。

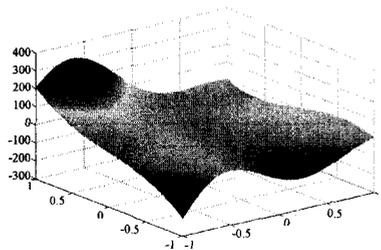


图 3 Energy function surface

种群大小设为 10, $c_1 = 3$, $c_2 = 2$, $\omega = 0.5$,进化代数为 500,由图 4 的能量曲线可以看出,MPSO 比传统 PSO 有更快的递减速度。这个简单的实验验证了第 4 部分的理论分析。图 4 给出了开始和第 7 代之后的能量曲线。

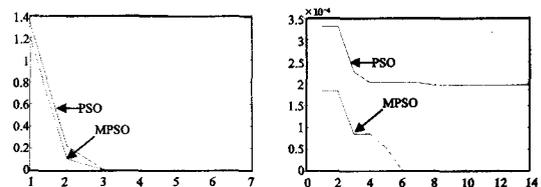


图 4 Energy curve of PSO versus MPSO. The second image shows the energy curve after 7 generations.

结束语 本文提出了基于流形的微粒群优化算法——MPSO,将 PSO 与流形相结合以示更好的性能,并对其进行了收敛性分析实验,从而验证了理论分析。

参考文献

[1] Bonabeau E, Dorigo M, Theraulaz G. Swarm Intelligence From Natural to Artificial Systems[M]. New York:Oxford University Press,1999

[2] van den B F, Engelbrecht A P. A cooperative approach to partial swarm optimization[J]. IEEE Trans. Evolutionary Computation,2004,8(3):225-239

[3] Kennedy J, Eberhart R C. Particle Swarm Optimization[C]// Proceedings on Neural Networks, IV. Piscataway, NJ: IEEE Service Center,1995:1942-1948

[4] Tenenbaum J, Silva D V, Langford J. A global geometric framework for nonlinear dimension reduction[J]. Science,2000,290:2319-2323

[5] Yang J, Li F X, Wang J. A Better Scaled Local Tangent Space Alignment Algorithm[J]. Journal of Software,16(9):1584-1590

[6] Zeng J, Jie J, Cui Z. Partical Swarm Optimization Algorithm. Beijing[M]. Science Press,2004

[7] Kohonen T. Self-organizing Maps. Springer-Verlag, 3rd Edition, 2000

[8] Zhang Z Y, Zha H Y. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment[J]. SIAM Journal of Scientific Computing,2004,26(1):313-338