一种更有效的 K-means 聚类算法①

A More Effective K-Means Clustering Algorithm

单玉双　邢长征　(辽宁工程技术大学 电子与信息工程学院 辽宁 葫芦岛 125105)

摘　要： 一个好的聚类算法不仅要考虑“同类内尽可能的相似，不同类间尽可能的相异”，而且也要考虑算法的时间复杂度。针对 K-means 算法依赖于初始聚类中心而影响聚类结果，提出了一种基于样本分布选取初始聚类中心的方法；针对 K-means 算法中每次调整聚类中心后指定聚类所需要的大量的距离计算，提出了三角不等式原理避免冗余计算的方法。将两种方法结合进行实验，结果表明新的方法更加有效，不仅较原算法有良好的聚类划分，而且加快了原算法的运行速度。

关键词： 聚类算法 时间复杂度 样本分布 冗余计算 聚类划分

K-means 算法是一种最常用的基于划分的聚类算法，该算法目前主要存在两个问题：算法对初始聚类中心敏感，从不同的初始聚类中心出发，得到的聚类结果也不一样，并且一般不会得到全局最优解；算法在每次调整了中心点之后，需要大量的距离计算确定新的聚类，其中的许多计算是冗余的。

1　引言

1.1 文章安排

本文第 2 节介绍原 K-means 算法。第 3 节给出初始中心选择的优化。第 4 节给出收敛速度的优化。第 5 节给出一种更有效的 K-means 聚类算法。第 6 节给出结论以及未来工作。

1.1.1 基本介绍

聚类分析是数据挖掘中的一个非常活跃的研究领域，也是一种重要的数据分析技术。聚类分析也已经广泛地应用到诸多领域中，包括模式识别、数据分析、图像处理以及市场研究等[1]。然而，面对大规模的、高维的数据，如何建立有效的聚类算法是当今的一个研究热点。对聚类算法的进一步优化研究不仅有助于算法理论的完善，更有助于算法的推广和应用。

为了实现对数据对象的聚类，人们提出了很多种不同的算法。K-means 算法是一种最常用的基于划分的聚类算法，该算法主要存在两个问题：算法对初始聚类中心敏感，从不同的初始聚类中心出发，得到的聚类结果也不一样，并且一般不会得到全局最优解；算法在每次调整了中心点之后，需要大量的距离计算确定新的聚类，其中的许多计算是冗余的。本文针对这两个问题对原 K-means 算法进行改进，提出了一种更有效的 K-means 聚类算法，不仅较原算法有良好的聚类划分，而且加快了原算法的运行速度。

2　K-means算法

2.1 K-means 算法

K-means 算法如下[2]：

输入：聚类个数 k，以及包含 n 个数据对象的数据库。

输出：满足方差最小标准的 k 个聚类。

处理流程：

1) 从 n 个数据对象任意选择 k 个对象作为初始聚类中心；

2) 循环 3 到 4 直到每个聚类不再发生变化为止；

3) 根据每个聚类对象的均值(中心对象)，计算每个对象与这些中心对象的距离，并根据最小距离重新对相应对象进行划分；

4) 重新计算每个(有变化)聚类的的平均值(中心对象)；

K-means 算法的工作过程说明如下：首先从 n

① 收稿时间:2008-12-08

个数据对象任意选择 k 个对象作为初始聚类中心；而对于所剩下其它对象，则根据它们与这些聚类中心的相似度(距离)，分别将它们分配给与其最相似的(聚类中心所代表的)聚类；然后再计算每个所获新聚类的聚类中心(该聚类中所有对象的均值)；不断重复这一过程直到标准测度函数开始收敛为止。一般采用均方差作为标准测度函数，具体如公式 1：

$$E=\sum_{i=1}^{k}\sum_{p\in c_i}\left|p-m_i\right|^2 \qquad (1)$$

其中，E 为数据库中所有对象的均方差之和；p 代表对象的空间中的一个点；m_i 为聚类 C_i 的均值(p 和 C_i 都是多维的)。K-means 算法的时间复杂度为 O(nkt)，n 为对象的个数；k 为聚类个数；而 t 为循环次数。K-means 算法常常终于局部最优。

2.2 K-means 算法存在的两个问题

l) 在 K-means 算法中，初始聚类中心的选择对聚类结果有较大的影响，一旦初始值选择的不好，不但无法得到有效的聚类结果，而且会增加迭代的次数，影响收敛速度[3]。所以需要对算法的初始中心的选择进行优化。

2) 从 K-means 算法框架可以看出，该算法需要不断地进行样本分类调整，并不断地计算调整后的新的聚类中心，使算法的时间开销是非常大的。所以需要对算法的时间复杂度进行分析、改进，提高算法应用效率。

下文从上述两个方面对K-means算法进行优化。

3　初始中心选择的优化

3.1 基本概念

为了便于说明问题，给出如下基本概念：

定义 1. 数据对象 $x=(x_1,x_2,\cdots,x_p)$和 $y=(y_1,y_2,\cdots,y_p)$ 之间的距离为[4]：

$$d(x,y)=\sqrt{(x_1-x_2)^2+(x_2-y_2)^2+\cdots+(x_p-y_p)^2}$$

定义 2. 一个数据对象 x 和一个数据对象集合 V 之间的距离，定义为这个数据对象与这个数据对象集合中所有数据对象当中最近的距离：

$$d(x,V)=\min\left(d(x,y),y\in V\right)$$

定义 3. 两个对象集合 S 和 V 之间的距离，定义为两个集合 S 和 V 中最近的两个数据对象 x 和 y 之间的距离：

$$d(S,V)=\min\left(d(x,y),x\in S,y\in V\right)$$

3.2 样本分布优化初始中心选择

设数据对象集合 U 有 n 个数据对象，要将其聚为 k 类，m 的初值为 1，(1≤m≤k)。则算法描述如下：

输入：聚类个数 k 包含 n 个数据对象的数据样本集；

输出：k 个初始聚类中心

处理流程：

1) 计算任意两个数据对象间的距离 d(x,y)，找到集合 U 中距离最近的两个数据对象，形成集合 Am，并从集合 U 中删除这两个对象；

2) 在 U 中找到距离集合 Am 最近的数据对象，将其加入集合 Am，并从集合 U 中删除该对象；

3) 重复 2 直到集合中的数据对象个数大于等于 $\alpha*n/k$，$(0<\alpha\leq 1)$；

4) 如果 m<k,则 m←m+1，再从集合 U 中找到距离最近的两个数据对象，形成新的集合 Am，并从集合 U 中删除这两个数据对象，返回 2 执行；

5) 将最终形成的 k 个集合中的数据对象分别进行算术平均，从而形成 k 个聚类中心。

4　收敛速度优化

4.1 基本概念

为了便于说明，给出如下基本概念：

定义 4. 假设存在数据集 $X(x_1,x_2,\cdots,x_n)$，若 X 中存在划分，使得 $X=\bigcup_{1\leq i\leq k}(S_i)(k<n)$，其中 S_i 成为其中的一个聚类。

定义 5. 假设 C 为所有已聚类中心的集合，当 $\forall x\in X,\exists c\in C$，使得 d(x,c)=min d(x, C)，那么 c 为向量 x 的聚类中心。

定理 1. 任一个三角行都有两边之和大于第三边，两边之差小于第三边[5]。欧式距离满足三角不等式特性，将它扩展到多维的欧几里得空间。欧几里得空间中任意 3 个向量 x，b 和 c 满足：d(b,c)+d(x,b)≥d(x,c)，d(b,c) -d(x,b)≤d(x,c)。

推论 1. 设 x 代表一个点，b 和 c 代表两个中心点。如果 2d(x, b) ≤d(b,c),那么 d(x,b) ≤d(x,c)。

证明：由于假设有 2d(x, b) ≤d(b,c)，同时在不等式两边减去 d(x,b)，则有：2d(x, b)-d(x,b) ≤d(b,c)- d(x,b)，即 d(x,b) ≤d(b,c)- d(x,b)；又由于定理 1：d(b,c) -d(x,b)≤d(x,c)，因此，我们得到 d(x,b) ≤d(x,c)；所以，如果 2d(x, b)≤d(b,c),那么

d(x,b) ≤d(x,c)。

根据推论 1 可以减少冗余的距离计算，应用如下：当在迭代过程中计算将点 x 分配到中心点 b 还是 c 时，如果已知 d(x,b) ≤d(x,c)时，此时再计算 x 和 c 的距离已无实际意义，因为存在距离该点较近的中心点 b。同样，若已知某一个点 x 距离某一个中心点 b 最近，也没必要再计算 x 与 b 的距离。为减少距离计算次数，我们设定一个上界和下界函数用来约束某点到各个中心点的距离，使得迭代计算时减少距离次数，提高运算速度。

4.2 三角不等式原理避免冗余计算

对于每个数据点 x 和中心 c，置下界约束 l(x,c)=0。对于每个点计算 x 到其中心点距离，将 x 分配给最近中心点 c，定义 c(x)为服务于 x 的中心点。在计算 x 到中心点距离时，可利用推论 1 避免冗余计算。每当 d(x,c)被计算时，置 l(x,c)=d(x,c)，同时求出上界约束 u(x)=min$_c$d(x,c)。

处理流程如下：

1) 对所有的中心 c 和 c’，计算 d(c,c’)。对所有的中心 c，计算 s(c)=min $_{c\neq c'}$ d(c,c’)。

2) 识别所有满足 u(x) ≤s(c(x))的数据向量。

3) 对所有剩下的同时满足条件 c≠c(x)；u(x)>l(x,c)；2u(x)>d(c(x),c)的数据点 x 和中心 c：

a. 如果 r(x)=true，那么计算 d(c(x),x)并且置 r(x)=false。否则 d(c(x),x)=u(x)。

b. 如果 d(c(x),x)>l(x,c) 或者 d(c(x),x)>d(c(x),c)/2，那么计算 d(x,c)，如果 d(x,c)<d(x,c(x))，那么置 c(x)= c。

4) 为每一个中心 c，让 m(c)是以 c 为中心的所有数据点的平均值。

5) 为每一个数据点 x 和中心 c，置 l(x,c)= max{ l(x,c)-d(c,m(c)),0}。

6) 为每个数据点 x，置 u(x)=u(x)+d(m(c(x)),c(x))，r(x)=true。

7) 将每个中心点 c 以 m(c)替换。

在步骤 3)对于任何 x 和 c，每当 d(x,c)被计算时，它的下界约束被更新通过置 l(x,c)=d(x,c)。同样，只要 c(x)改变或者 d(x,c(x))被计算，上界约束 u(x)就被更新。在 3)a 步骤中，如果 r(x)为真，那么 u(x)是过期的，即可 d(x,c(x))≠u(x)。否则，没必要计算 d(x,c(x))。步骤 3)b 重复核对步骤 3)中的条件，为了尽可能避免计算 d(x,c)。而该算法能有效地减少距离计算次数的基本原因是，在每次循环的开始，对于许多数据点 x 和中心 c 来说，上界约束 u(x)和下界约束 l(x,c)是相当紧凑的。如果在一次循环的开始这些约束是紧的，上界约束在下一次循环的开始仍然是紧的，因为许多中心只是稍稍改变了一下，因此，约束的改变也是轻微的。

5 一种更有效的K-means聚类算法

将上述两种算法相结合得出一种更有效的 K-means 聚类算法，处理流程如下：

输入：聚类个数 k，以及包含 n 个数据对象的数据库。

输出：满足方差最小标准的 k 个聚类。

处理流程：

1) 计算任意两个数据对象间的距离 d(x,y)，找到集合 U 中距离最近的两个数据对象，形成集合 Am，并从集合 U 中删除这两个对象；

2) 在 U 中找到距离集合 Am 最近的数据对象，将其加入集合 Am，并从集合 U 中删除该对象；

3) 重复 2 直到集合中的数据对象个数大于等于 α*n/k，(1< α ≤1)；

4) 如果 m<k,则 m←m+1，再从集合 U 中找到距离最近的两个数据对象，形成新的集合 Am，并从集合 U 中删除这两个数据对象，返回 2 执行；

5) 将最终形成的 k 个集合中的数据对象分别进行算术平均，从而形成 k 个聚类中心。

6) 循环 7 到 14 直到每个聚类不再发生变化为止；

7) 对于每个数据点 x 和中心 c，置下界约束 l(x,c)=0。将 x 归为与它最近的中心 c(x)，(c(x)为服务于 x 的中心点)。

8) 对所有的中心 c 和 c’，计算 d(c,c’)。对所有的中心 c，计算 s(c)=min$_{c\neq c'}$ d(c,c’)。

9) 识别所有满足 u(x) ≤s(c(x))的数据向量。

10) 对所有剩下的同时满足条件 c≠c(x)；u(x)>l(x,c)；2u(x)>d(c(x),c)的数据点 x 和中心 c：

a. 如果 r(x)=true，那么计算 d(c(x),x)并且置 r(x)=false。否则 d(c(x),x)=u(x)。

b. 如果 d(c(x),x)>l(x,c)或者 d(c(x),x)>d(c(x),c)/2，那么计算 d(x,c)，如果 d(x,c)<d(x,c(x))，那么置 c(x)= c。

11) 为每一个中心 c，让 m(c)是以 c 为中心的所

有数据点的平均值。

12) 为每一个数据点 x 和中心 c，置 l(x,c)= max{ l(x,c)-d(c,m(c)),0}。

13) 为每个数据点 x，置：u(x)=u(x)+d(m(c(x)),c(x))，r(x)=true。

14) 将每个中心点 c 以 m(c)替换。

对改进后的 K-means 算法的要求：

1) 原 K-means 算法能聚类的数据集，改进的 K-means 算法也能被应用；

2) 由于初始中心已经选定，所以在 k 值固定的情况下，改进的 K-means 算法只有一个稳定的划分结果，而且聚类的精度有所提高；

3) 改进的 K-means 算法比原算法运行的速度快；

6 实验结果

为评价算法有效性，采用 matlab 编写算法，机器配置为 Pentium 4 CPU，3.2GHz，IGB 内存，80GB 硬盘。选用 UCI 中 3 个数据集来测试该算法的有效性，这 3 个数据集的描述如表 1 所示。由于原算法为随机算法，所以采用运算 20 次该算法后所求的平均聚类值和最小聚类值与传统 K-means 算法所求结果相对应的值进行对比用来判断新算法的有效性，如表 2 所示。表 3 则根据每次迭代过程中，由于点的重新分配而进行的距离个数计算的平均值以及平均运算时间用来对比两个算法的运算时间。

注：新算法改进值公式=(1-新算法值/原算法值)*100%

表 1　选用数据集说明

数据集	点集个数	空间围数
Cloud Data	1024	10
Abalone	4177	8
Spam	4601	58

根据测试结果我们可以得出结果如表 2 和表 3：

通过表 2 和表 3 可以看出，新算法与传统算法相比，无论从聚类的精度还是从运算时间上都得到了较大提高，取得了较为令人满意的结果。当数据集的空间维数较大时如 Spam 数据集，本算法取得更佳的效果。

表 2 算法的聚类值对比

数据集	k 值	平均聚类值		最小聚类值	
		传统算法	新算法改进	传统算法	新算法改进
Cloud Data	10	80255640.62	28.65%	6434 641.07	11.56%
	30	3181423.42	46.76%	2013988.82	14.03%
	50	2037643.79	35.02%	1334073.92	18.20%
Abalone	10	2460.58	25.93%	1985.47	9.47%
	30	1022.23	58.29%	307.47	27.64%
	50	531.02	64.74%	307.47	45.67%
Spam	10	169 84 178.09	45.45%	169516116.51	49.30%
	30	149412154.74	87.23%	142 754999.33	80.19%
	50	149 678668.75	94.30%	140564888.51	92.37%

表 3　运行时间的对比

数据集	k 值	平均运算时间(ms)		平均每次迭代距离计算数	
		传统算法	新算法改进	传统算法	新算法改进
Cloud Data	10	117.2	73.96%	11264.	80. 65%
	30	178.90	74.26%	31744	83. 30%
	50	282.05	77.35%	52224	85. 02%
Abalone	10	101.60	43.39%	457947	69. 74%
	30	275.00	64.92%	129487	71. 46%
	50	407.85	63.47%	213027	79. 76%
Spam	10	3341.40	94. 54%	50611	79. 03%
	30	12832.85	96.32%	142631	88. 91%
	50	19165.65	95. 03%	234651	91. 73%

7 结论

新的算法对于数据集的空间维数较大情况时，无论从聚类的质量上还是运算效率上都具有非常大的提高。

本文结合传统 K-means 算法求解过程，分别从初始点选取、收敛过程加速等方面提出一种新的有效的改进方法。实验表明本算法能够获得较好的划分结果。本算法结构简单，易于实现。文中初始中心优化过程的 α 值的选取有待进一步研究。

参考文献

1 朱明.数据挖掘.合肥:中国科学技术大学出版社, 2002:138－139.

2 谭斯坦巴赫.范明,范宏建译.数据挖掘导论.北京:人民邮电出版社, 2006:182－183.

3 李听,郑宇,江芳泽.用改进 RPCL 算法提取聚类的最佳数目.上海大学学报, 1999,5(5):409－413.

4 唐立新,王梦光.用遗传算法改进聚类分析中的 k 平均算法.数理统计与应用概率, 1997,6(4):45－46.

5 陈小云,王平.基于三角不等式原理的 TTSAS 聚类加速算法.兰州大学学报, 2006,32(17):1－2.