基于 Hub 的高维数据初始聚类中心的选择策略①

张巧达, 何振峰

(福州大学 数学与计算机科学学院, 福州 350108)

摘 要: 针对基于 Hub 的聚类算法 K-hubs 算法存在对初始聚类中心敏感的问题, 提出一种基于 Hub 的初始中心选择策略. 该策略充分利用高维数据普遍存在的 Hubness 现象, 选择相距最远的 K 个 Hub 点作为初始的聚类中心. 实验表明采用该策略的 K-hubs 算法与原来采用随机初始中心的 K-hubs 算法相比, 前者拥有较好的初始中心分布, 能够提高聚类准确率, 而且初始中心所在的位置倾向于接近最终簇中心, 有利于加快算法收敛.

关键词: Hubness; 初始中心; 最大最小距离方法; 高维数据; 聚类

Hub-Based Initialization for K-hubs

ZHANG Qiao-Da, HE Zhen-Feng

(School of Mathematics and Computer Science, Fuzhou University, Fujian 350108, China)

Abstract: K-hubs is a Hub-based clustering algorithm that is very sensitive to initialization. Therefore, this paper proposes an initialization method based on Hub to solve this problem. The initialization method takes full use of the feature of the Hubness phenomenon by selecting initial centers that are the most remote Hub points with each other. The experimental results show that compared with the random initialization of ordinary K-hubs algorithm, the proposed initialization method can obtain a better distribution of initial centers, which could enhance the clustering accuracy; moreover, the selected initial centers can appear near the cluster centers, which could speed up the convergence of the clustering algorithm.

Key words: Hubness; initial center; maximin method; high-dimensional data; clustering

聚类分析作为数据挖掘研究的重要手段之一, 已被广泛应用于机器学习、统计分析、模式识别、图像处理等领域. 随着数据采集技术的进步, 现实生活中各个领域的数据呈现高维度化的趋势. 高维数据存在着不同于低维数据的特征, 使得许多适用于低维空间的传统聚类算法, 在高维空间中的表现不能令人满意. 处理高维数据面临的困难, 被称为“维度灾难”[1]. “维度灾难”有两个突出的现象: 空空间现象和距离集中现象. 前者源于高维数据的稀疏性: 随着维度的增加, 数据点在高维空间的分布变得越来越稀疏. 后者表现为: 在高维空间里, 任意两个数据点间的距离趋于一致. 这两个现象导致了使用欧式距离等作为相似度度量的聚类算法在高维空间里失效.

2009 至 2012 年, 一系列研究高维数据另一个现象—Hubness 现象—的论文陆续出现[2-4], 提供了一种新思路来解决“维度灾难”带来的挑战. Hubness 现象是指高维数据里某些数据点更频繁出现在其他数据点的 k 最近邻列表中, 而且这一趋势会随着维度的增加而更加明显. 这种现象是高维空间的一种内在特征, 对机器学习和数据挖掘很多方面都产生了影响, 比如分类[5]、时间序列的实例选择[6,7]、信息检索和聚类等. Tomasev 等人[8]在 2011 年提出了利用高维数据的 Hubness 现象进行聚类的 K-hubs 算法. K-hubs 算法表明高维数据中的 Hub 点能作为一种局部中心, 提高聚类质量. 尽管实验结果显示 K-hubs 算法对高维数据集的聚类性能优于传统的聚类算法, 而且高噪声环境下算法的性能更稳定, 但 K-hubs 算法存在着对初始中心敏感, 容易陷入局部极值的问题. 初始聚类中心选择不好, 会极大

① 收稿时间:2014-07-31;收到修改稿时间:2014-09-28

影响 K-hubs 算法的聚类效果.

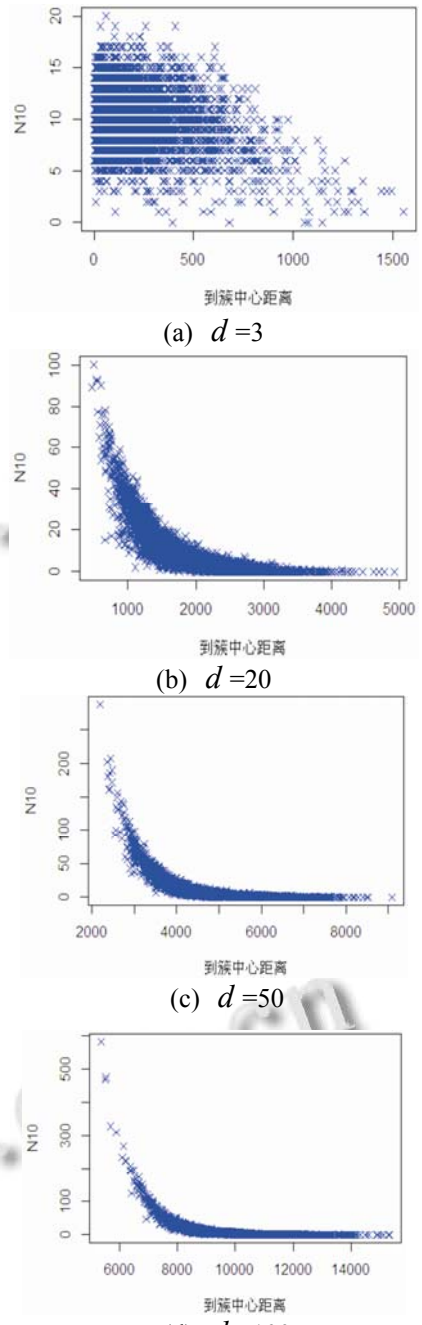
为了解决这一问题, 本文提出一种基于 Hub 的初始聚类中心选择策略. 该优化策略利用 Hubness 现象里 Hub 点比其他数据点更倾向于靠近簇中心这一特征, 从数据集中选择一定数目的 Hub 点集合, 初始聚类中心只能从这一集合里选出, 以使初始聚类中心能尽可能靠近簇中心. 在此基础上, 采用最大最小距离思想, 使任意两个初始聚类中心尽量不出现在一个簇中.

1　Hubness现象

令$D \in R^d$表示一个 d 维的数据集, $N_k(x)$ 表示数据点 x 在其他所有数据点的 k 最近邻列表中出现的次数. Hubness现象表现为: 随着维度的增加, $N_k(x)$ 的分布会呈现出明显的右偏, 即某些数据点的 $N_k(x)$ 值远远大于其他数据点. 这些具有高 $N_k(x)$ 值的数据点称为 Hub点.

Hub 点的出现与距离集中现象有关. 当高维数据出现距离集中现象时, 大部分数据点将落在以数据质心为中心的超球面上[9]. 然而, 不管数据具有多少数目的维度, 各个数据点到数据质心的距离变化都不能忽略不计[10]. 高维数据里仍然会存在一些数据点比其他数据点更靠近数据质心. 一般来说, 数据质心附近的数据点周围存在着更多的数据点. 在高维空间里, 这一趋势更加明显. 这一特征使得靠近质心的数据点更有可能出现在其他数据点的 k 最近邻列表中, 即这些数据点具有更高的 $N_k(x)$ 值. 若数据集产生自多个分布, 那么大部分数据点将落在以相应分布的质心为中心的超球面上.

图 1(a-d)显示 5000 个随机抽取自(0, 1)均匀分布的 d 维数据点, 在簇类数 K =6, 近邻数 k =10, 维度分别为: (a) d =3, (b) d =20, (c) d =50, (d) d =100 情况下, 通过 K-means 算法得到聚类结果簇中心, 求得数据点到簇中心距离与该数据点的 $N_k(x)$ 值的关系. 横坐标表示数据点到簇中心的距离, 纵坐标表示该数据点的 $N_k(x)$ 值. 从图中可以看到, 当 d =3 时, 数据点的 $N_k(x)$ 值与其到簇中心的距离没有多大关系. 随着维度增加, 当维度 d =20 时, 距离簇中心越近的点, 其 $N_k(x)$ 值越高. 换句话说, $N_k(x)$ 值越高的点越可能靠近簇中心. 当维度 d 到达 50, 100 时, 这一趋势更加明显. 这意味着, 在聚类算法中选择 Hub 点作为簇中心, 有利于缩短初始中心到最终簇中心的迭代次数.



(a) d =3

(b) d =20

(c) d =50

(d) d =100

图 1　数据点 x 到簇中心的距离与 $N_k(x)$ 值的关系

2　基于Hub的聚类算法K-hubs

2.1 K-hubs 算法

K-hubs算法过程类似K-means算法, 将每个数据点分配给距离其最近的簇中心点, 以此形成簇. 不同之处在于每轮迭代更新簇中心时, K-means算法选择簇质心作为簇中心, K-hubs算法选择簇内具有最高 $N_k(x)$ 值的点作为簇中心. 簇内数据点的 $N_k(x)$ 值定义为: 数

据点 x 在整个数据集内其他数据点的 k 最近邻列表中出现的次数.

K-hubs 算法过程如下:

```
Algorithm 1.  K-hubs
initializeClusterCenters();
Cluster[] clusters = formClusters();
repeat
   for all Cluster c ∈ clusters do
      DataPoint h = findClusterHub(c)
      setClusterCenter(c, h);
   end for
   clusters = formClusters();
until noReassignments
return clusters
```

2.2 K-hubs 算法的不足

K-hubs 算法能使聚类过程快速收敛, 然而同 K-means 算法一样, 由于随机选择初始中心点, 算法容易陷入局部极小. 若随机选择出的初始中心分布不够均衡, K-hubs 算法以簇内最高 $N_k(x)$ 值的点作为簇中心, 这一更新簇中心方式更会加剧算法陷入局部极小. 而且这种现象在 $N_k(x)$ 值分布差异较大的数据集里尤其严重. 如图 2 所示的数据集包含三个簇, 右上角的簇里数据点较多且分布密集, 因此 $N_k(x)$ 值也普遍比另外两个簇高. 图中小的实心圆是初始中心, 大的实心圆是运行 K-hubs 算法后得到的最终簇中心. 可以看到, 由于采取了随机方法, 三个初始中心都出现在右上方的簇里. 根据 K-hubs 算法的更新簇中心方式, 右上角的簇里 $N_k(x)$ 值普遍比较高, 经过不断迭代得到的最终簇中心都仍将集中在右上角的簇里, 造成原本应该归于同一个簇的数据点被划分到三个簇里. 从该例子可以看到, 初始点的选择对 K-hubs 算法的聚类效果有极大影响.

3 基于Hub的初始中心选择策略HBI

良好的初始中心选择策略应该尽可能满足两个目标: (1)所选的初始中心尽可能靠近簇中心; (2)每个初始中心尽可能分布在不同簇中.

正如图 1 所示, 维度越高, Hub 点越倾向于出现在簇中心附近. 这一现象启发我们优先选择 Hub 点作为聚类算法的初始中心. 受这一启发, 本文提出基于 Hub 的初始中心选择策略 HBI(Hub-based Initialization Method). 该策略的主要思想是从 Hub 点集合中尽可能选择分布在不同簇的 Hub 点, 以达到上面提到的两个目标.

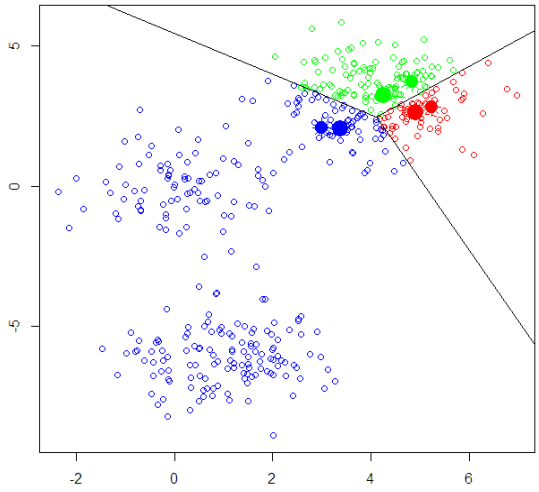


图 2 包含三个簇的数据集例子

首先, 对 Hub 点集合进行定义:

定义 1. Hub 点集合. 计算数据集每个数据点 x 的 $N_k(x)$ 值, 按从大到小排序, 取前 $\beta \times n$ 部分的数据点作为 Hub 点集合. 其中, $\beta \in (0,1)$, β 值的选择要满足 $\beta \times n >> K$, K 为簇数, n 为数据集大小.

为了尽可能达到上述目标(1), HBI 策略将只从 Hub 点集合里选择初始中心.

为了达到目标(2), HBI 策略采用最大最小距离思想[11], 从 Hub 点集合里选择相距最远的 K 个点. 而根据最大最小距离思想选出来的 K 个初始中心, 很可能是 Hub 点集合中 $N_k(x)$ 值较小的点. 为了使得 K 个初始中心相距尽量分开, 同时又使得这 K 个点有尽可能高的 $N_k(x)$ 值, 如图 3 所示, 本策略在用最大最小距离思想求得 K 个点后, 对每个点再遍历它们 k 个最近邻, 找到具有最高 $N_k(x)$ 值的点作为初始中心.

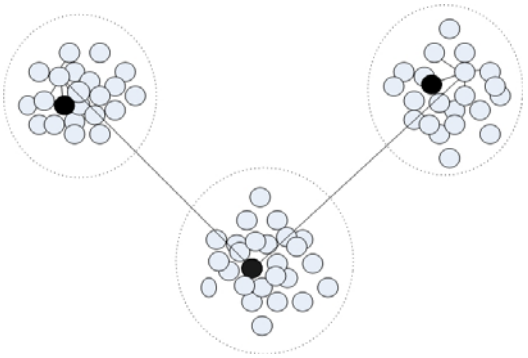


图 3 HBI 策略效果图

HBI 策略的具体步骤如下:

输入: 数据集 $\boldsymbol{D}$, 类数 K, 近邻数 k, β;

输出: 选出的 K 个初始中心;

1) 计算数据集 $\boldsymbol{D}$ 中每个数据点 x 的 $N_k(x)$ 值, 从大到小排序后, 取前 $\beta \times n$ 部分的数据点作为 Hub 点集合 H, 其中 $\boldsymbol{n}$ 是数据集 $\boldsymbol{D}$ 的样本数;

2) 从 H 中选择具有最高 $N_k(x)$ 值的点作为第 1 个初始中心 c_1, $c_1 \in H$;

3) 选择 H 中与 c_1 距离最远的数据点作为第 2 个初始中心 c_2, $c_2 \in H$;

4) 求 H 中每个点到距其最近的初始中心的距离, 找到距离最远的那个点作为第 3 个初始中心 c_3, $c_3 \in H$;

5) 重复步骤(4), 直到取到 K 个点.

6) 分别从这 K 点的每个点及其 k 个最近邻中, 选择具有最高 $N_k(x)$ 值的点作为最终初始中心.

4 实验结果与分析

4.1 实验环境及评价方法

实验选择了 11 个 UCI 数据集, 对 K-hubs 算法和使用 HBI 策略改进后的算法 HBI-K-hubs 进行比较. 实验使用十折交叉验证, 并用准确率(Rand Index)对聚类结果进行评价. 为了确保实验结果更加准确, 每个实验独立重复进行 10 次, 取 100 次结果的均值作为评价数据. 实验的运行环境为: 操作系统为 Windows XP, 处理器为 Pentium(R) Dual-Core 2.50GHz, 内存 1.99GB, 编程语言为 R 2.15.3.

Rand Index 的定义为

$$RandIndex=\frac{TP+TN}{TP+FP+FN+TN},$$

其中, TP(True-positive)表示事实上属于同一类且聚类后仍属于同一类的样本对数; TN(True-negative)表示事实上不属于同一类且聚类后也不属于同一类的样本对数; FP(False-positive)表示事实上不属于同一类但聚类后属于同一类的样本对数; FN(False-negative)表示事实上属于同一类但聚类后不属于同一类的样本对数. Rand Index 值越大说明聚类结果越好.

4.2 算法参数设置

实验中近邻数 k 设置为 10. 由于高维数据里 Hub 点仅占小部分, 而同时为了确保能够选到尽够多的 Hub 点以从中选出良好的初始中心分布, 实验中将 Hub 点集合的数目在整个数据集的比例(即 β)设置为 10%.

4.3 实验结果和分析

表 1　两种算法在 11 个 UCI 数据集上的比较

数据集	大小	维度	类数	准确率(%)		迭代次数		时间(s)	
				K-hubs	HBI-K-hubs	K-hubs	HBI-K-hubs	K-hubs	HBI-K-hubs
ionoshpere	351	34	2	55.14	**57.07**	2.48	**1.76**	0.057	**0.0506**
wdbc	569	30	2	79.86	**87.52**	2.71	**1.91**	0.1173	**0.1058**
iris	150	4	3	78.27	**85.61**	2.51	**2.05**	0.0230	**0.0206**
wine	178	13	3	79.98	**86.27**	2.78	**1.85**	0.0292	**0.0258**
pageblocks	5473	10	5	42.85	**65.56**	3.19	**3.03**	**5.772**	5.9261
segment	2310	19	7	81.14	**84.01**	3.25	**2.64**	1.6092	**1.5904**
mfeat-fac	2000	216	10	90.04	**91.34**	3.58	**2.61**	3.3021	**3.1027**
mfeat-fou	2000	76	10	85.73	**85.95**	4.28	**3.98**	**2.6762**	2.8443
mfeat-kar	2000	64	10	85.94	**87.08**	3.66	**3.07**	**2.4316**	2.4664
mfeat-pix	2000	240	10	89.47	**90.84**	3.89	**3.19**	3.0602	**3.0576**
optdigits	5620	64	10	**88.99**	88.12	**3.1**	3.51	**6.2931**	6.6552

1)聚类结果准确率比较

从表 1 可以看到, 使用 HBI 改进后的算法 HBI-K-hubs 在大部分数据集里都能使聚类准确率得到提升. 尽管 optdigits 数据集的聚类准确率略有下降, 但下降不超过 1%, 属于可接受范围. 实验结果说明利用 Hubness 信息结合最大最小距离思想, 选择出来的初始聚类中心能得到更好的初始中心分布, 有助于避免 K-hubs 算法频繁陷入局部极值, 通过不断迭代更容易获得正确的聚类结果.

2)时间复杂性分析

从算法分析上看, HBI-K-hubs 仅比 K-hubs 多花费一些时间在初始中心选择上. HBI 策略需要执行 K 次

迭代, 每次迭代平均进行 $\beta \times n \times \frac{K \times (K-1)}{2 \times K}$ 次向量距离计算, 因此 HBI 策略总共需要进行大约 $\frac{\beta \times K^2}{2} \times n$ 次向量距离计算. 本文实验 β 取 10%(在大数据集下可以取更小值), 通常情况下数据集的类数 K 也不会很大, 取 $K=10$, HBI 策略一共仅需进行大约 $5 \times n$ 次向量距离计算. 从表 1 迭代次数一栏的比较可以看出, 除了 optdigits 数据集, HBI-K-hubs 的迭代次数比 K-hubs 的更少. 这说明 HBI 策略能在一开始筛选出较好的初始中心, 因而 HBI-K-hubs 能更快收敛. 从表 1 上运行时间一栏的比较看, HBI-K-hubs 的运行时间也普遍比 K-hubs 的更少. 这是由于 HBI 策略能选出较好的初始中心从而获得了更快的收敛速度, 抵消了 HBI 策略的时间开销, 而其实 HBI 的策略时间开销并不多, 因此 HBI-K-hubs 最终还比原来的 K-hubs 算法更快.

5 结语

本文研究了基于 Hub 的聚类算法 K-hubs, 指出了该算法存在的缺点: 初始中心的选取对 K-hubs 算法的聚类效果有极大影响. K-hubs 算法仅在算法迭代过程中使用 Hubness 特征, 并没有在算法初始化时充分利用这一特征来筛选出更好的初始中心. 针对这一缺点, 本文提出 HBI 初始中心选择策略, 经过实验表明, 在 HBI 策略选出的初始中心基础上, 通过不断的聚类迭代, K-hubs 算法的聚类准确率能得到提高. 而且由于 HBI 策略选出的初始中心会出现在最终簇中心附近, 能加速聚类算法的收敛, 因此平衡了 HBI 策略的时间开销.

参考文献

1 Han J, Kamber M. Data Mining: Concepts and Techniques. 2nd ed, Morgan Kaufmann Publishers, 2006.

2 Radovanovic M, Nanopoulos A, Ivanovic M. Nearest neighbors in high-dmensional data: The emergence and influence of hubs. Proc. of the 26th International Conference on Machine Learning(ICML). 2009. 865–872.

3 Radovanovic M, Nanopoulos A, Ivanovic M. Hubs in space: popular nearestneighbors in high-dimensional data. Journal of Machine Learning Research 9999, 2010: 2487–2531.

4 Tomasev N, Mladenic D. Nearest neighbor voting in high dimensional data: Learning form past occurrences. Conputer Science and Information Systems, 9,2012: 691–712.

5 Tomasev N, Mladenic D. Hubness-aware shared neighbor distances for high-dimensional k-nearest neighbor classification. Hybrid Aritificial Intelligent Systems, 2012: 116–127.

6 翟婷婷,何振峰.基于 Hubness 的类别均衡的时间序列实例选择算法.计算机应用,2012,32(11):3034–3037.

7 Zhai TT, He ZF. Instance selection for time series classification based on immune binary particle swarm optimization. Knowledge-Based Systems. 2013, 49(9): 106–115.

8 Tomasev N, Radovanovic M, Mladenic D, Ivanovic M. The role of hubness in clustering high-dimensional data. Advances in Knowledge Discovery and Data Mining, 2011: 183–195.

9 Aggarwal CC, Hinneburg A, Kein DA. On the surprising behavior of distance metrics in high dimensional spaces. Proc. of the 8th International Conference on Database Theory (ICDT). 2001. 420–434.

10 Beyer K, Goldstein J, Ramakrishnan R, et al. When is “nearest neighbor” meaningful? Proc. of the 7th International Conference on Database Theory(ICDT). 1999. 217–235.

11 Katsavounidis I, Jay Kuo CC, Zhang Z. A new initialization technique for generalized Lloyd iteration. Signal Processing Letters, 1994, 1(10): 144–146.