一种最小生成树的聚类方法

齐建鹏

烟台大学 计算机与控制工程学院,山东 烟台 264005

摘 要: 通过最小生成树实现了一种聚类方法，该方法分为两个阶段进行，首先对数据生成最小生成树，然后在最小生成树的基础上根据给定的阈值*threshold*进行聚类. 并在6个来自网络的数据集上进行了验证，实验证明，基于最小生成树的聚类具有良好的效果，同时，该算法能够根据不同的*threshold*快速的将数据划分，在效率上也具有不错的效果。

关键词: 聚类;最小生成树;Prim算法

A Clustering Method based on MST

QI Jianpeng

School of Computer and Control Engineering, Yantai University, Yantai 264005, China)

**Abstract**: This paper implements a clustering method based on Multiple Spanning Tree(MST). Through 2 stages, we can clustering data into serval clusters. First, using MST generated by dataset with Prim algorithm. Second, give a *threshold* and partition data into clusters. According 6 datasets from Internet, we find that clustering method based on MST can get a good result, and after first stage, we can easily obtain different results using various *threshold*.

**Key words**: clustering; MST; Prim

# 引言

随着互联网的发展，大量的数据不断产生。人们不再局限于信息的表面含义，而是开始挖掘一些数字背后的隐藏信息。IDC与EMC发布了一份题为“2020年的数字宇宙”的报告中，预测到2020年，世界信息量将会膨胀到40000EB，或40万亿GB。这无疑说明了大数据时代已经来临。人工对庞大的数据进行有效管理是不可能的，如何发现具有巨大效益的潜在信息，这是一个十分火热的话题。对于大数据的处理，计算机的参与就成为了必然。

聚类在数据分析与处理中占据着重要的地位，通过聚类可以把不同类别的数据进行区分。很多领域都涉及到了聚类，比如模式识别，数据汇总，模式发现以及图像处理等[1]。传统的聚类方式大致分为三类，基于划分的聚类，基于密度的聚类以及层次聚类[2]。

基于化分的聚类通常需要预先知道数据中存在类别的数量，根据给定的数量进行聚类，比如采用*k*个初始点初始化然后根据距离远近进行划分的*k*-means方法[3]，*k*-medoids[4]，CLARANS[5]以及FCM[6]等。

基于密度的聚类是较为广泛的一种方法，通过密度聚类可以发现任意形状的簇。常见的密度聚类算法有才用邻域半径与临域最小点数概念的DBSCAN算法[7]，使用增广的簇排序对聚类进行优化的OPTICS方法[8]等。

而基于层次聚类的方法又可以分类两类，即自上而下（分解）和自下而上（合并），例如，利用树的结构对对象集进行划分，然后再利用其它聚类方法对这些聚类进行优化的BIRCH[9]方法；利用固定数目代表对象来表示相应聚类，然后对各聚类按照指定量（向聚类中心）进行收缩的CURE[10]方法以及利用聚类间的连接进行合并的ROCK[11]等。

本文主要通过最小生成树（MST）进行聚类，首先通过Prim算法对数据生成最小生成树，然后根据不同的阈值进行动态的聚类。

# Prim算法获取最小生成树

## 最小生成树

在介绍Prim算法前，首先对最小生成树进行描述。

**定义1(最小生成树).** 设G=(V, E)是无向连通带权图，即一个网络。E中每条边(v, w)的权为w(v, w)。如果G的一个子图G’是一棵包含G的所有顶点的树，则称G’为G的生成树。生成树上各边权的总和称为该生成树的消耗。在G的所有生成树中，耗费最小的生成树称为G的最小生成树(MST)。

最小生成树有很多应用，比如要在n个城市之间建立通信联络网,则连接n个城市只需要n-1条线路。这时,自然会考虑这样一个问题,如何在节省费用的前提下建立这个通信网? n个城市之间最多可以设置n(n-1)/2条线路,那么如何在这些可能的线路中选择n-1条使总的代价最小呢?可以用连通网来表示n个城市以及n个城市之间可能设置的通信线路,其中网的顶点表示城市,边表示两个城市之间的线路,赋予边的权值表示相应的代价。对于n个顶点的连通网可以建立许多不同的生成树,每一个生成树都可以是一个通信网。使总的代价最小的树便是最小代价生成树(简称最小生成树)。

同时，最小生成树还具有特殊的性质。

**性质1(MST性质).** 设G=(V, E)是无向连通带权图，U是V的真子集。如果(u, v)∈E，且u∈U，v∈V-U，且在所有这样的边中，(u, v)的权w(u,v)最小，那么一定存在G的一棵最小生成树，它以(u,v)为其中一条边。

由性质1可知，G的子图中最小权w(u,v)在G中最小生成树一定存在，那么最小生成树问题可通过贪心算法进行求解。即，通过选择最优的点或边来不断地构造最小生成树。

## Prim算法

通过2.1节对最小生成树定义及其性质的了解，可以利用贪心算法进行求解。较为经典的是Prim与Kruskal算法。Prim算法是通过每次选择权值较小的边来构造MST。

首先，设G=(V, E)是连通带权图，V={1, 2, …, n}。则算法思想如算法1所示。

**算法1**：Prim(V, E)

输入: V, E

输出: S, E’

01: E’ ← ∅;

02: S ← {1};

03: while(S ≠ V){

04: (i, j) ← min{w(i,j) | i∈S and j∈V-S};

05: E’ ← E’ ∪{i, j};

06: S ← S ∪ {j};

07:}

如算法1，首先选择一个点作为开始点并放入已生成的最小生成树集合(S,E’)中，如算法第02行，然后根据在V集合中还未选择的点到已选择S中的权重，选取最小的点j归纳到V中，直到所有的点都被扫描，算法结束。此时得到的(S, E’)即为一棵最小生成树。

通过以上算法，可以分析得出，Prim算法的时间复杂度为O(n2)。

## 最小生成树聚类

通过前两节对最小生成树以及Prim算法的描述，可知最小生成树是通过选取最小权重边构造的。在聚类问题上根据MST的一些性质，则可以很快的将数据进行聚类。

在介绍算法之前，首先对几个定义进行阐述，下述定义与DBSCAN算法中的一些定义相类似。

**定义2(*threshold*-邻域).** 对象*p*的*threshold*-邻域是与*p*为中心，*threshold*为半径的空间。参数*threshold*，是用户指定每个对象的领域半径值。

**定义3(直接可达).** 如果对象p在象q的*threshold*-邻域内，则p是从q直接可达的。

**定义4(间接可达).** p是从q间接可达的，如果存在对象链，使得,是从关于*threshold*直接可达的，即在的*threshold*-邻域内，则到间接可达。

为了更好地理解定义2~4，下面给出图示，如图1。

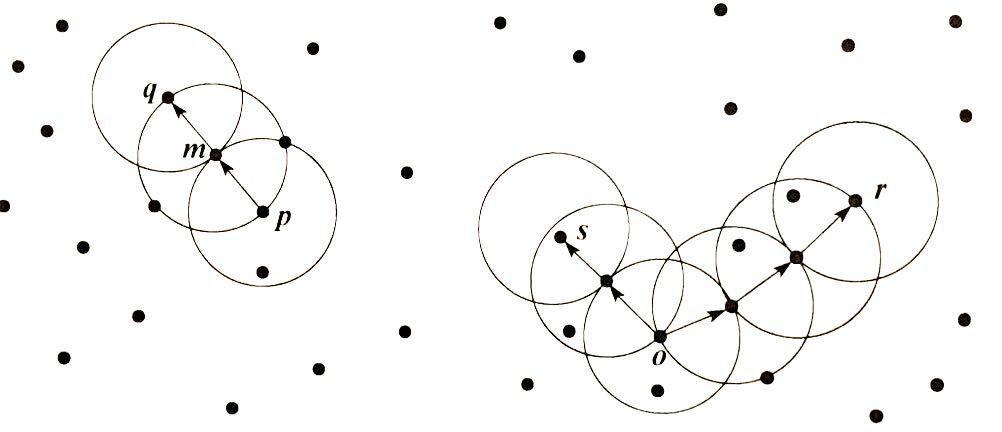


图1基于最小生成树的聚类中的直接可达和间接可达

图1中，m与p，q是直接可达的，o与s，r是间接可达的。

考虑数据集D={pi | i = 1, 2, …, n}, pi∈d 。在最小生成树中，权重采用欧式距离，即w。

最小生成树的聚类思想是比较简单的，首先通过Prim算法对点集生成最小生成树，然后根据给定的阈值*threshold*对最小生成树的边进行扫描，当w时，将当前的链接切断。对所有数据执行上述操作后，剩下的还相互链接的部分即为相同的簇。伪代码如算法2所示。为了更好地利用空间，在Prim算法中，最小生成树E’是对距离矩阵E修改而来。

**算法2**：MST\_Clustering(*D*)

输入: *D*, 数据集; *threshold*, 阈值

输出: *clusters*, 聚类结果

01: 生成点之间的距离矩阵*E*

02: (*S*, *E’*) = Prim(*D*, *E*)

03: *clusters* ← bfs(*S*, *E’*, *threshold*)

**算法3：**bfs(*S*, *E’*, *threshold*)

01: while(*rest* > 0){//1. 选择一个点p作为初始点

02: 选择一个未扫描的点*entry*∈*S*作为入口;

03: *queue*.add(*entry*);

04: while(!*queue*.isEmpty()){//扫描与entry相连的点（直接／间接可达）

05: *element* = *queue*.poll();

06: *rest* -- ;

07: *cluster*.add(*element*);

08: 在*E’*中根据*threshold*添加与*element*直接可达的点到*queue*中;

09: }

10: *clusters*.add(*cluster*);

11: }

为了便于最大限度利用最小生成树的信息，算法在广度优先遍历（算法3）时才使用了阈值*threshold*。这是因为最小生成树的代价较大，而在聚类的过程中，只需要对最小生成树进行一次扫描即可，所以可以根据不同的阈值进行动态的调整。算法3使用了队列对生成树进行遍历，复杂度为O(|S| + |E|)。

下面给出一个实例，如图2，对所有点生成最小生成树后将得到类似的无向图。对于该无向图，给定阈值*threshold*，将每条权值大于*threshold*的边去掉，剩下的每一部分将是一个完整的簇。



图2 最小生成树聚类

# 实验

程序采用Java编写，JDK版本为1.8，操作系统为MacOS 10.11.5。

## 实验数据

实验数据来自于相关研究的数据集，数据源见[13]中的Shape sets，如表1。

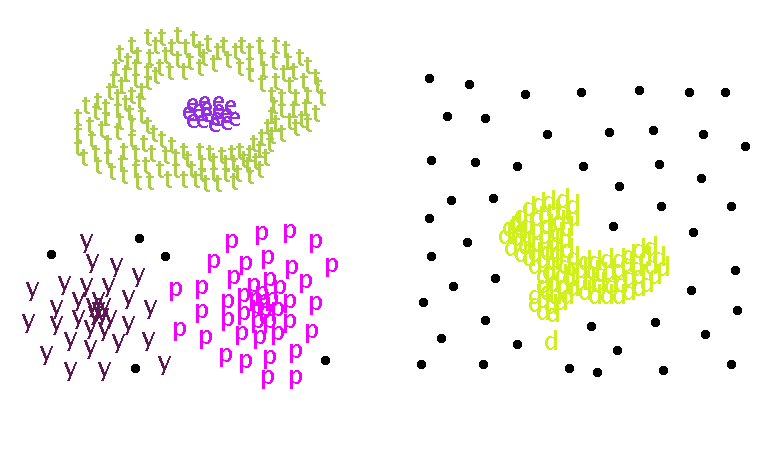
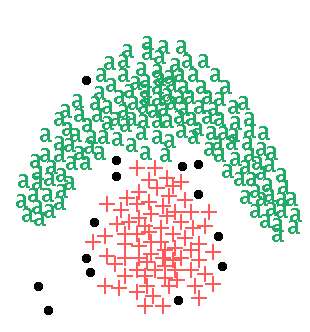
表1 数据集

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 点数 | 维数 | 类别 |
| Compound | 399 | 2 | 6 |
| Flame | 240 | 2 | 2 |
| R15 | 600 | 2 | 15 |
| Rings | 281 | 2 | 2 |
| Spiral | 312 | 2 | 3 |
| t4.8k | 8000 | 2 | 6 |

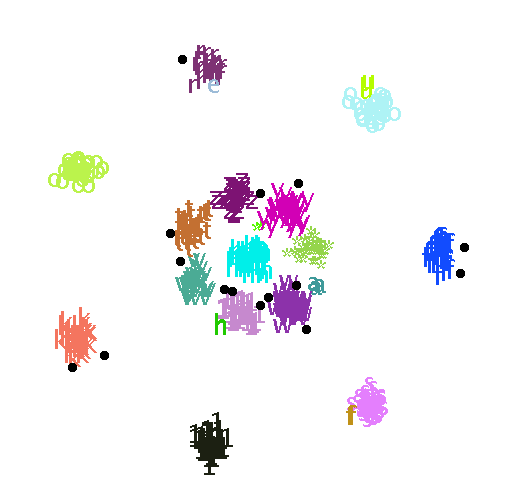
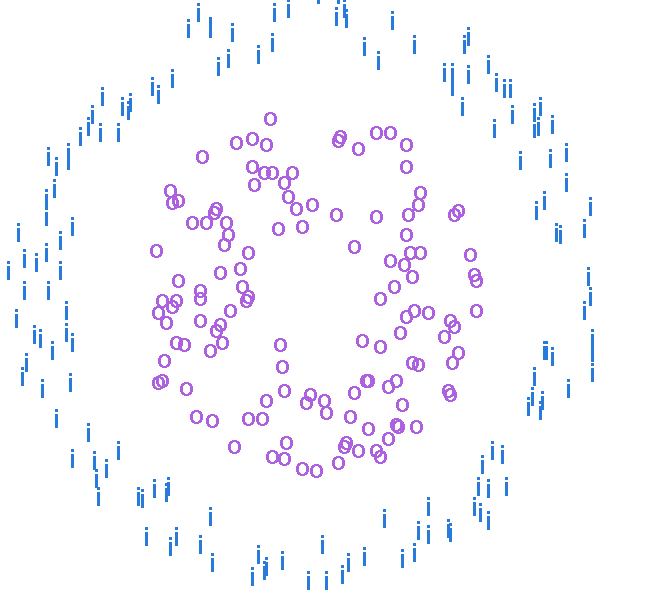
## 实验结果

如图3，分别展示了3.1节中数据的实验结果。为了更好的进行显示，针对每一个数据集引进了噪声显示，当某个数据簇的样本数小于*Noise*时，则将该簇识别为噪声，如图中圆心实点所示。

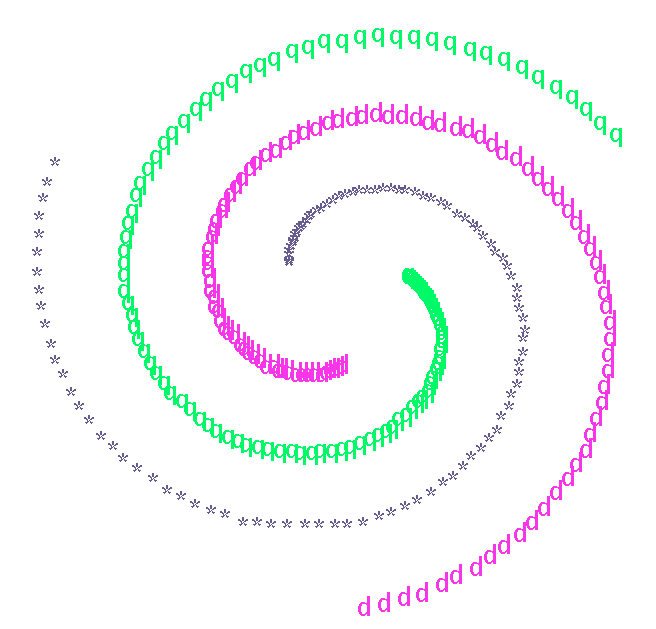
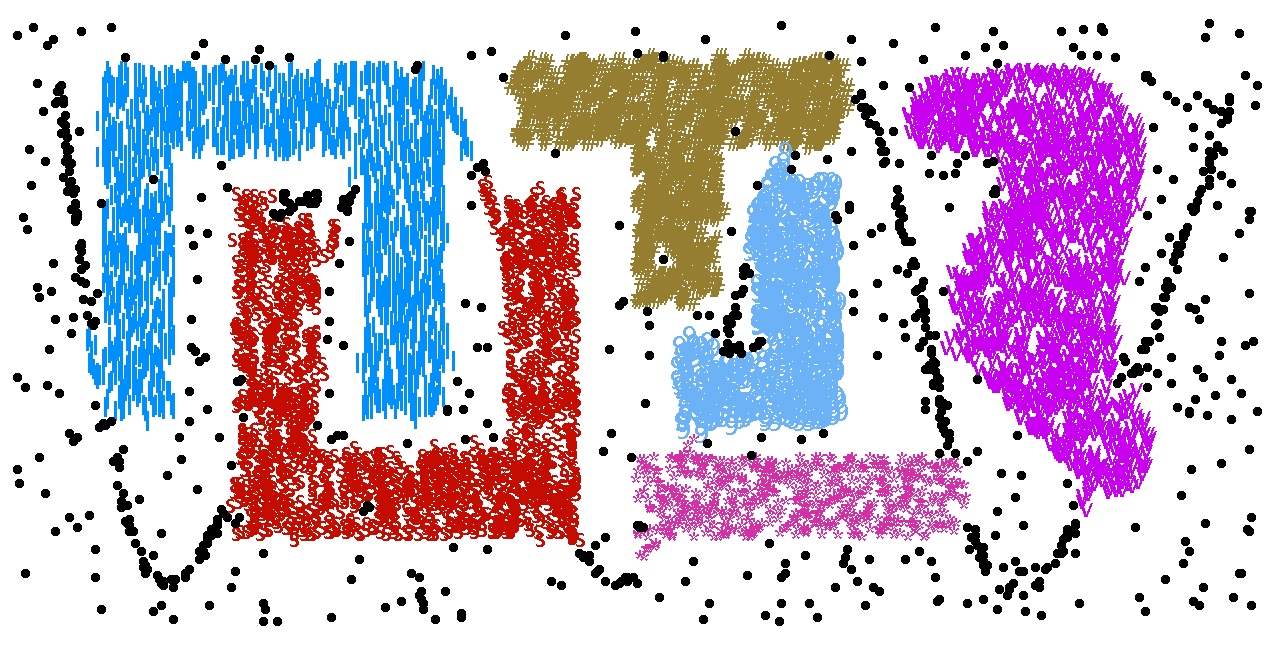
实验证明，基于最小生成树的聚类算法具有良好的效果。

Compound(Thr=1.5, Noise=3, Time=7ms) Flame(Thr=0.83, Noise=2, Time=9ms)

R15(Thr=0.32, Noise=1, Time=22ms) Rings(Thr=1, Noise=1, Time=9ms)

Spiral(Thr=2, Noise=1, Time=8ms) t4.8k(Thr=5.1, Noise=40, Time=2301ms)

图3 在不同数据集上的聚类结果

## 数据集大小对算法性能的影响

为了测试算法在数据集大小上的表现，对t4.8k数据进行了处理，随机删除一部分点后，将数据分为了8组，数据量从1000~8000。实验结果如图4。



图4 数据量对算法的影响

通过图4，可以看出，算法的执行时间随着数据量的增大而增大，同时还可以看出Prim算法在生成最小生成树的过程中占用了大量的时间，这也符合上文对算法的复杂度分析。值得注意的是，在最小生成树生成后，对数据进行聚类时，所耗费的时间并不长，这是因为只需对生成树扫描一遍后就可以得到最终结果。

# 总结

通过使用最小生成树，对数据实现了聚类。最小生成树聚类算法在第一阶段（Prim算法）虽然会消耗一定的时间，但是在第二阶段（聚类）可以很快的对数据进行划分。因此，最小生成树聚类可以根据不同的阈值很快的得到结果。

参考文献:

1. Celebi M E, Kingravi H A, Vela P A. A comparative study of efficient initialization methods for the k-means clustering algorithm[J]. Expert Systems with Applications, 2012, 40(1):200-210.
2. JIAWEI HAN(加). 数据挖掘概念与技术[M]. 机械工业出版社, 2006.
3. Macqueen J. Some Methods for Classification and Analysis of MultiVariate Observations[C]// Berkeley Symposium on Math. 1966:281-297.
4. Kaufmann L, Rousseeuw P J. Clustering by Means of Medoids[C]// Statistical Data Analysis Based on the L1-norm & Related Methods. North-Holland, 1987:405-416.
5. Ng R T, Han J. CLARANS: A Method for Clustering Objects for Spatial Data Mining[J]. IEEE Transactions on Knowledge & Data Engineering, 2015, 14(5):1003-1016.
6. J. C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters[J]. Journal of Cybernetics, 1973, 3(3):32-57.
7. Ester M, Kriegel H P, Jiirg S, et al. A densitybased algorithm for discovering clusters in large spatial databases[C]// 2008:226--231.
8. Ordering Points To Identify the Clustering Structure[C]// International Conference on Management of Data. 1999.
9. Zhang T. BIRCH: an efficient data clustering method for very large databases[J]. Acm Sigmod Record, 1996, 25(2):103-114.
10. Guha S, Rastogi R, Shim K. Cure: an efficient clustering algorithm for large databases ☆[J]. Information Systems, 1998, 26(1):35-58.
11. Guha S, Rastogi R, Shim K. Rock: A robust clustering algorithm for categorical attributes ☆[J]. 1999, 25(5):512-521.
12. http://cs.joensuu.fi/sipu/datasets/[DB/OL].