谱聚类

谱聚类（spectral clustering）是广泛使用的聚类算法，比起传统的K-Means算法，谱聚类对数据分布的适应性更强，聚类效果也很优秀，同时聚类的计算量也小很多，更加难能可贵的是实现起来也不复杂。谱聚类演化于图论，后由于其表现出优秀的性能被广泛应用于聚类中，对比其他无监督聚类（如kmeans），spectral clustering的优点主要有以下：

- 过程对数据结构并没有太多的假设要求，如kmeans则要求数据为凸集。
- 可以通过构造稀疏similarity graph，使得对于更大的数据集表现出明显优于其他算法的计算速度。
- 由于spectral clustering是对图切割处理，不会存在像kmesns聚类时将离散的小簇聚合在一起的情况。
- 无需像GMM一样对数据的概率分布做假设。（假设属于Gaussian Mixture Model）

同样，spectral clustering也有自己的缺点，主要存在于构图步骤，有如下：

- 对于选择不同的similarity graph比较敏感（如 epsilon-neighborhood， k-nearest neighborhood，fully connected等）。
- 对于参数的选择也比较敏感（如 epsilon-neighborhood的epsilon，k-nearest neighborhood的k，fully connected的）。

1 无向权重图

图$G = (V, E)$, 其中$V = (v_1, v_2, ...v_n)$。定义权重w_{ij}为点v_i和点v_j之间的权重。由于我们是无向图，所以$w_{ij} = w_{ji}$。节点v_i的度d_i定义为和它相连的所有边的权重之和，即：

$$d_i = \sum_{j=1}^{n} w_{ij} \tag{1}$$

度矩阵：

$$\mathbf{D} = \begin{pmatrix} d_1 & \dots & \dots \\ \dots & d_2 & \dots \\ \vdots & \vdots & \ddots \\ \dots & \dots & d_n \end{pmatrix} \tag{2}$$

利用所有点之间的权重值，我们可以得到图的邻接矩阵W，它也是一个$n \times n$的矩阵，第i行的第j个值对应我们的权重w_{ij}。除此之外，对于点集V的的一个子集$A \subset V$，我们定义：

$$|A| := \#nodes \quad in \quad A \tag{3}$$

A中节点度之和

$$vol(A) := \sum_{i \in A} d_i \tag{4}$$

2 相似度矩阵

基本思想是，距离较远的两个点之间的边权重值较低，而距离较近的两个点之间的边权重值较高，不过这仅仅是定性，我们需要定量的权重值。一般来说，我们可以通过样本点距离度量的相似矩阵S来获得邻接矩阵W。构建邻接矩阵W的方法有三类。ϵ-邻近法，K邻近法和全连接法。

2.1 ϵ-邻近法

对于ϵ-邻近法，它设置了一个距离阈值ϵ，然后用欧式距离s_{ij}度量任意两点x_i和x_j的距离。即相似矩阵的$s_{ij} = ||x_i - x_j||_2^2$ （平方和开根号）, 然后根据s_{ij}和ϵ的大小关系，来定义邻接矩阵W如下：

$$w_{ij} = \begin{cases} 0 & s_{ij} > \epsilon \\ \epsilon & s_{ij} \leq \epsilon \end{cases} \tag{5}$$

从上式可见，两点间的权重要不就是ϵ,要不就是0，没有其他的信息了。距离远近度量很不精确，因此在实际应用中，我们**很少使用ϵ-邻近法**。

2.2 K邻近法

第二种定义邻接矩阵W的方法是K邻近法，利用KNN算法遍历所有的样本点，取每个样本最近的k个点作为近邻，只有和样本距离最近的k个点之间的$w_{ij} > 0$。但是这种方法会造成重构之后的邻接矩阵W非对称，我们后面的算法需要对称邻接矩阵。为了解决这种问题，一般采取下面两种方法之一：第一种K邻近法是只要一个点在另一个点的K近邻中，则保留s_{ij}:

$$w_{ij} = w_{ji} = \begin{cases} 0 & x_i \notin KNN(x_j) \; and \; x_j \notin KNN(x_i) \\ \exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2}) & x_i \in KNN(x_j) \; or \; x_j \in KNN(x_i) \end{cases} \tag{6}$$

第二种K邻近法是必须两个点互为K近邻中，才能保留s_{ij}:

$$w_{ij} = w_{ji} = \begin{cases} 0 & x_i \notin KNN(x_j) \; or \; x_j \notin KNN(x_i) \\ \exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2}) & x_i \in KNN(x_j) \; and \; x_j \in KNN(x_i) \end{cases} \tag{7}$$

第三种定义邻接矩阵W的方法是全连接法，相比前两种方法，第三种方法所有的点之间的权重值都大于0，因此称之为全连接法。可以选择不同的核函数来定义边权重，常用的有多项式核函数，高斯核函数和Sigmoid核函数。最常用的是高斯核函数RBF，此时相似矩阵和邻接矩阵相同：

$$w_{ij} = s_{ij} = exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2}) \tag{8}$$

在实际的应用中，使用第三种全连接法来建立邻接矩阵是最普遍的，而在全连接法中使用高斯径向核RBF是最普遍的。

3 拉普拉斯矩阵

单独把拉普拉斯矩阵(Graph Laplacians)拿出来介绍是因为后面的算法和这个矩阵的性质息息相关。它的定义很简单，拉普拉斯矩阵$L = D` W$。D即为度矩阵，它是一个对角矩阵。而W即为邻接矩阵，它可以由我

们第二节的方法构建出。拉普拉斯矩阵是对称半正定矩阵，且对应的n个实数特征值都大于等于0, 且最小特征值为0 ：

$$0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n \tag{9}$$

对于任意向量f, 有：

$$f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \tag{10}$$

推导如下：

$$\begin{aligned} f^T L f = f^T D f - f^T W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j \\ = \frac{1}{2}(\sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n} d_j f_j^2) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \end{aligned} \tag{11}$$

4 无向图切图

对于无向图G的切图, 目标是将图$G = (V, E)$切成相互没有连接的k个子图, 每个子图点的集合为：$A_1, A_2, ..A_k$，它们满足$A_i \cap A_j = \emptyset$,且$A_1 \cup A_2 \cup ... \cup A_k = V$。对于任意两个子图点的集合$A, B \subset V$, 且$A \cap B = \emptyset$, 我们定义$A$和$B$之间的切图权重为：

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij} \tag{12}$$

那么对于我们k个子图点的集合：$A_1, A_2, ..A_k$, 我们定义切图cut为：

$$cut(A_1, A_2, ...A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \overline{A}_i) \tag{13}$$

其中$\overline{A}_i$为A_i的补集，意为除A_i子集外其他V的子集的并集。那么如何切图可以让子图内的点权重和高，子图间的点权重和低呢？一个自然的想法就是最小化$cut(A_1, A_2, ...A_k)$, 但是可以发现，这种极小化的切图存在问题，如下图： 我们选择一个权重最小的边缘的点，比如C和H之间进行cut，这样可以最小化$cut(A_1, A_2, ...A_k)$, 但

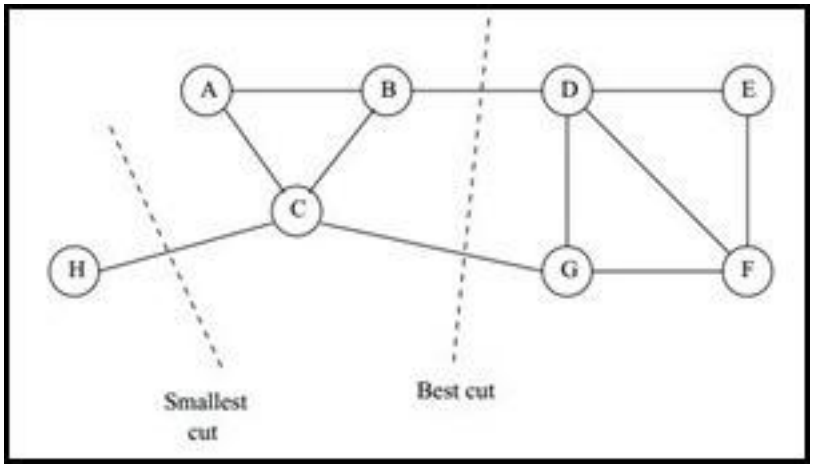

图 1: this is a figure demo

是却不是最优的切图，如何避免这种切图，并且找到类似图中"Best Cut"这样的最优切图呢？我们下一节就来看看谱聚类使用的切图方法。

5 谱聚类之切图聚类

谱聚类切图存在两种主流的方式：RatioCut和Ncut，目的是找到一条权重最小，又能平衡切出子图大小的边，下面详细说明这两种切法。

5.1 RatioCut切图

RatioCut切图为了避免上一节的最小切图，对每个切图，不光考虑最小化$cut(A_1, A_2, ...A_k)$，它还同时考虑最大化每个子图点的个数，即：

$$RatioCut(A_1, A_2, ...A_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(A_i, \overline{A}_i)}{|A_i|} \tag{14}$$

那么怎么最小化这个RatioCut函数呢？牛人们发现，RatioCut函数可以通过如下方式表示。我们引入指示向量$h_j \in \{h_1, h_2, ..h_k\}\ j = 1, 2, ...k$，对于任意一个向量$h_j$，它是一个$n$维向量（$n$为样本数），我们定义$h_{ij}$为：

$$h_{ij} = \begin{cases} 0 & v_i \notin A_j \\ \frac{1}{\sqrt{|A_j|}} & v_i \in A_j \end{cases} \tag{15}$$

那么我们对于$h_i^T L h_i$有：

$$\begin{aligned} h_i^T L h_i &= \frac{1}{2}\sum_{m=1}\sum_{n=1} w_{mn}(h_{im} - h_{in})^2 \\ &= \frac{1}{2}(\sum_{m\in A_i, n\notin A_i} w_{mn}(\frac{1}{\sqrt{|A_i|}} - 0)^2 + \sum_{m\notin A_i, n\in A_i} w_{mn}(0 - \frac{1}{\sqrt{|A_i|}})^2 \\ &= \frac{1}{2}(\sum_{m\in A_i, n\notin A_i} w_{mn}\frac{1}{|A_i|} + \sum_{m\notin A_i, n\in A_i} w_{mn}\frac{1}{|A_i|} \\ &= \frac{1}{2}(cut(A_i, \overline{A}_i)\frac{1}{|A_i|} + cut(\overline{A}_i, A_i)\frac{1}{|A_i|}) \\ &= \frac{cut(A_i, \overline{A}_i)}{|A_i|} \end{aligned} \tag{16}$$

上述第（1）式用了上面第四节的拉普拉斯矩阵的性质3. 第二式用到了指示向量的定义。可以看出，对于某一个子图i，它的RatioCut对应于$h_i^T L h_i$，那么我们的k个子图呢？对应的RatioCut函数表达式为：

$$RatioCut(A_1, A_2, ...A_k) = \sum_{i=1}^{k} h_i^T L h_i = \sum_{i=1}^{k}(H^T LH)_{ii} = tr(H^T LH) \tag{17}$$

注意到我们H矩阵里面的每一个指示向量都是n维的，向量中每个变量的取值为0或者$\frac{1}{\sqrt{|A_j|}}$, 就有2^n种取值，有k个子图的话就有k个指示向量，共有$k2^n$种H，因此找到满足上面优化目标的H是一个NP难的问题。那么是不是就没有办法了呢？　　注意观察$tr(H^T LH)$中每一个优化子目标$h_i^T L h_i$，其中h是单位正交基，L为对称矩阵，此时$h_i^T L h_i$的最大值为L的最大特征值，最小值是L的最小特征值。如果你对主成分分析PCA很熟悉的话，这里很好理解。在PCA中，我们的目标是找到协方差矩阵（对应此处的拉普拉斯矩阵L）的最大的特征值，而在我们的谱聚类中，我们的目标是找到目标的最小的特征值，得到对应的特征向量，此时对应二分切图效果最佳。也就是说，我们这里要用到维度规约的思想来近似去解决这个NP难的问题。

对于$h_i^T L h_i$, 我们的目标是找到最小的L的特征值，而对于$tr(H^T LH) = \sum_{i=1}^{k} h_i^T L h_i$, 则我们的目标就是找到$k$个最小的特征值，一般来说，$k$远远小于$n$，也就是说，此时我们进行了维度规约，将维度从$n$降到了$k$，从而近似可以解决这个NP难的问题。

通过找到L的最小的k个特征值，可以得到对应的k个特征向量，这k个特征向量组成一个$n \times k$维度的矩阵，即为我们的H。一般需要对H矩阵按行做标准化，即

$$h_{ij}^{*} = \frac{h_{ij}}{(\sum_{t=1}^{k} h_{it}^{2})^{1/2}} \tag{18}$$

由于我们在使用维度规约的时候损失了少量信息，导致得到的优化后的指示向量h对应的H现在不能完全指示各样本的归属，因此一般在得到$n \times k$维度的矩阵H后还需要对每一行进行一次传统的聚类，比如使用K-Means聚类。

5.2 Ncut切图

Ncut切图和RatioCut切图很类似，但是把Ratiocut的分母$|Ai|$换成$vol(A_i)$. 由于子图样本的个数多并不一定权重就大，我们切图时基于权重也更合我们的目标，因此一般来说Ncut切图优于RatioCut切图。

$$NCut(A_1, A_2, ...A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{vol(A_i)} \tag{19}$$

对应的，Ncut切图对指示向量h做了改进。注意到RatioCut切图的指示向量使用的是$\frac{1}{\sqrt{|A_j|}}$标示样本归属，而Ncut切图使用了子图权重$\frac{1}{\sqrt{vol(A_j)}}$来标示指示向量$h$，定义如下:

$$h_{ij} = \begin{cases} 0 & v_i \notin A_j \\ \frac{1}{\sqrt{vol(A_j)}} & v_i \in A_j \end{cases} \tag{20}$$

那么我们对于$h_i^T L h_i$有：

$$\begin{aligned} h_i^T L h_i &= \frac{1}{2}\sum_{m=1}\sum_{n=1} w_{mn}(h_{im} - h_{in})^2 \\ &= \frac{1}{2}(\sum_{m\in A_i, n\notin A_i} w_{mn}(\frac{1}{\sqrt{vol(A_i)}} - 0)^2 + \sum_{m\notin A_i, n\in A_i} w_{mn}(0 - \frac{1}{\sqrt{vol(A_i)}})^2 \\ &= \frac{1}{2}(\sum_{m\in A_i, n\notin A_i} w_{mn}\frac{1}{vol(A_i)} + \sum_{m\notin A_i, n\in A_i} w_{mn}\frac{1}{vol(A_i)} \\ &= \frac{1}{2}(cut(A_i, \overline{A}_i)\frac{1}{vol(A_i)} + cut(\overline{A}_i, A_i)\frac{1}{vol(A_i)}) \\ &= \frac{cut(A_i, \overline{A}_i)}{vol(A_i)} \end{aligned} \tag{21}$$

推导方式和RatioCut完全一致。也就是说，我们的优化目标仍然是

$$NCut(A_1, A_2, ...A_k) = \sum_{i=1}^{k} h_i^T L h_i = \sum_{i=1}^{k} (H^T L H)_{ii} = tr(H^T L H) \tag{22}$$

但是此时我们的$H^T H \neq I$, 而是$H^T D H = I$。推导如下：

$$h_i^T D h_i = \sum_{j=1}^{n} h_{ij}^2 d_j = \frac{1}{vol(A_i)}\sum_{j\in A_i} d_j = \frac{1}{vol(A_i)} vol(A_i) = 1 \tag{23}$$

也就是说，此时我们的优化目标最终为：

$$\underbrace{arg\ min}_{H}\ tr(H^T L H)\ \ s.t.\ H^T D H = I \tag{24}$$

此时我们的H中的指示向量h并不是标准正交基，所以在RatioCut里面的降维思想不能直接用。怎么办呢？其实只需要将指示向量矩阵H做一个小小的转化即可。　　我们令$H = D^{-1/2}F$, 则：

$$\begin{aligned} H^T LH &= F^T D^{-1/2} L D^{-1/2} F \\ H^T DH &= F^T F = I \end{aligned} \tag{25}$$

也就是说优化目标变成了:

$$\underbrace{arg\ min}_{F}\ tr(F^T D^{-1/2} L D^{-1/2} F)\ \ s.t.\ F^T F = I \tag{26}$$

可以发现这个式子和RatioCut基本一致，只是中间的L变成了$D^{-1/2}LD^{-1/2}$，这样我们就可以继续按照RatioCut的思想，求出$D^{-1/2}LD^{-1/2}$的最小的前k个特征值，然后求出对应的特征向量，并标准化，得到最后的特征矩阵F, 最后对F进行一次传统的聚类（比如K-Means）即可。　一般来说，$D^{-1/2}LD^{-1/2}$相当于对拉普拉斯矩阵L做了一次标准化, 即：

$$\frac{L_{ij}}{\sqrt{d_i * d_j}} \tag{27}$$

6　谱聚类算法流程

最常用的相似矩阵的生成方式是基于高斯核距离的全连接方式，最常用的切图方式是Ncut。而到最后常用的聚类方法为K-Means。下面以Ncut总结谱聚类算法流程。

输入：样本集$D = (x_1, x_2, ..., x_n)$, 相似矩阵的生成方式, 降维后的维度k_1, 聚类方法，聚类后的维度k_2

输出：簇划分$C(c_1, c_2, ...c_{k_2})$。

1. 根据输入的相似矩阵的生成方式构建样本的相似矩阵S
2. 根据相似矩阵S构建邻接矩阵W，构建度矩阵D
3. 计算出拉普拉斯矩阵L
4. 构建标准化后的拉普拉斯矩阵$D^{-1/2}LD^{-1/2}$
5. 计算$D^{-1/2}LD^{-1/2}$最小的k_1个特征值所各自对应的特征向量f
6. 将各自对应的特征向量f组成的矩阵按行标准化，最终组成$n \times k_1$维的特征矩阵F
7. 对F中的每一行作为一个k_1维的样本，共n个样本，用输入的聚类方法进行聚类，聚类维数为k_2
8. 得到簇划分$C(c_1, c_2, ...c_{k_2})$

7　谱聚类算法总结

谱聚类算法是一个使用起来简单，但是讲清楚却不是那么容易的算法，它需要你有一定的数学基础。如果你掌握了谱聚类，相信你会对矩阵分析，图论有更深入的理解。同时对降维里的主成分分析也会加深理解。

下面总结下谱聚类算法的优缺点。

谱聚类算法的主要优点有：

1）谱聚类只需要数据之间的相似度矩阵，因此对于处理稀疏数据的聚类很有效。这点传统聚类算法比如K-Means很难做到

2）由于使用了降维，因此在处理高维数据聚类时的复杂度比传统聚类算法好。

谱聚类算法的主要缺点有：

1）如果最终聚类的维度非常高，则由于降维的幅度不够，谱聚类的运行速度和最后的聚类效果均不好。

2) 聚类效果依赖于相似矩阵，不同的相似矩阵得到的最终聚类效果可能很不同。