CHAPTER 16 Spectral and Graph Clustering

In this chapter we consider clustering over graph data, that is, given a graph, the goal is to cluster the nodes by using the edges and their weights, which represent the similarity between the incident nodes. Graph clustering is related to divisive hierarchical clustering, as many methods partition the set of nodes to obtain the final clusters using the pairwise similarity matrix between nodes. As we shall see, graph clustering also has a very strong connection to spectral decomposition of graph-based matrices. Finally, if the similarity matrix is positive semidefinite, it can be considered as a kernel matrix, and graph clustering is therefore also related to kernel-based clustering.

16.1 GRAPHS AND MATRICES

Given a dataset $\mathbf{D} = \{\mathbf{x}_i\}_{i=1}^n$ consisting of n points in $\mathbb{R}^d$, let $\mathbf{A}$ denote the $n \times n$ symmetric *similarity matrix* between the points, given as

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \tag{16.1}$$

where $\mathbf{A}(i, j) = a_{ij}$ denotes the similarity or affinity between points $\mathbf{x}_i$ and $\mathbf{x}_j$. We require the similarity to be symmetric and non-negative, that is, $a_{ij} = a_{ji}$ and $a_{ij} \geq 0$, respectively. The matrix $\mathbf{A}$ may be considered to be a *weighted adjacency matrix* of the weighted (undirected) graph $G = (V, E)$, where each vertex is a point and each edge joins a pair of points, that is,

$$V = \{\mathbf{x}_i \mid i = 1, \ldots, n\}$$
$$E = \big\{(\mathbf{x}_i, \mathbf{x}_j) \mid 1 \leq i, j \leq n\big\}$$

Further, the similarity matrix $\mathbf{A}$ gives the weight on each edge, that is, a_{ij} denotes the weight of the edge $(\mathbf{x}_i, \mathbf{x}_j)$. If all affinities are 0 or 1, then $\mathbf{A}$ represents the regular adjacency relationship between the vertices.

For a vertex $\mathbf{x}_i$, let d_i denote the *degree* of the vertex, defined as

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

We define the *degree matrix* $\mathbf{\Delta}$ of graph G as the $n \times n$ diagonal matrix:

$$\mathbf{\Delta} = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{n} a_{1j} & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{n} a_{2j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{j=1}^{n} a_{nj} \end{pmatrix}$$

$\mathbf{\Delta}$ can be compactly written as $\mathbf{\Delta}(i,i) = d_i$ for all $1 \le i \le n$.

Example 16.1. Figure 16.1 shows the similarity graph for the Iris dataset, obtained as follows. Each of the $n = 150$ points $\mathbf{x}_i \in \mathbb{R}^4$ in the Iris dataset is represented by a node in G. To create the edges, we first compute the pairwise similarity between the points using the Gaussian kernel [Eq. (5.10)]:

$$a_{ij} = \exp\left\{-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right\}$$

using $\sigma = 1$. Each edge $(\mathbf{x}_i, \mathbf{x}_j)$ has the weight a_{ij}. Next, for each node $\mathbf{x}_i$ we compute the top q nearest neighbors in terms of the similarity value, given as

$$N_q(\mathbf{x}_i) = \left\{\mathbf{x}_j \in V : a_{ij} \le a_{iq}\right\}$$

where a_{iq} represents the similarity value between $\mathbf{x}_i$ and its qth nearest neighbor. We used a value of $q = 16$, as in this case each node records at least 15 nearest neighbors (not including the node itself), which corresponds to 10% of the nodes. An edge is added between nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ if and only if both nodes are *mutual nearest neighbors*, that is, if $\mathbf{x}_j \in N_q(\mathbf{x}_i)$ and $\mathbf{x}_i \in N_q(\mathbf{x}_j)$. Finally, if the resulting graph is disconnected, we add the top q most similar (i.e., highest weighted) edges between any two connected components.

The resulting Iris similarity graph is shown in Figure 16.1. It has $|V| = n = 150$ nodes and $|E| = m = 1730$ edges. Edges with similarity $a_{ij} \ge 0.9$ are shown in black, and the remaining edges are shown in gray. Although $a_{ii} = 1.0$ for all nodes, we do not show the self-edges or loops.

Normalized Adjacency Matrix

The normalized adjacency matrix is obtained by dividing each row of the adjacency matrix by the degree of the corresponding node. Given the weighted adjacency matrix

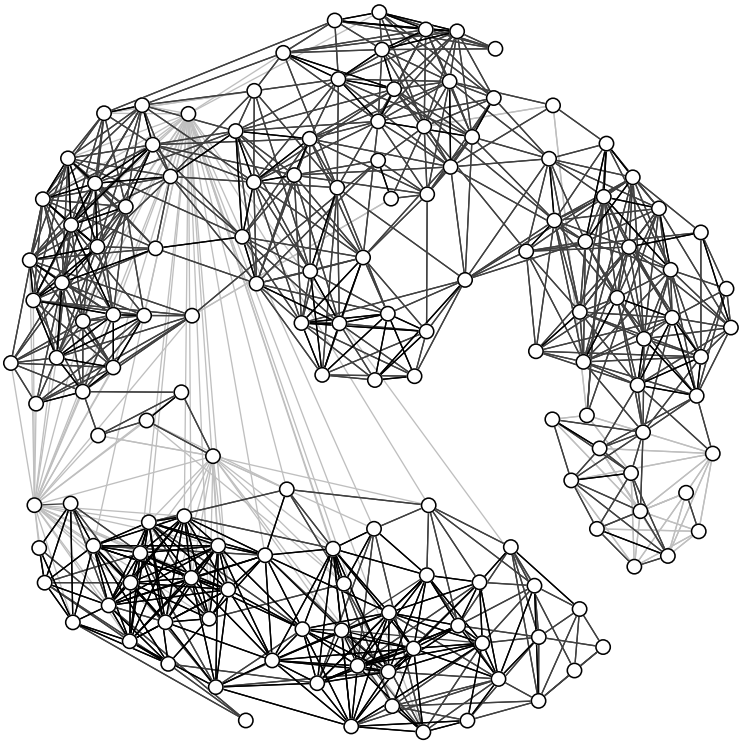


Figure 16.1. Iris similarity graph.

$\mathbf{A}$ for a graph G, its normalized adjacency matrix is defined as

$$\mathbf{M} = \mathbf{\Delta}^{-1}\mathbf{A} = \begin{pmatrix} \frac{a_{11}}{d_1} & \frac{a_{12}}{d_1} & \cdots & \frac{a_{1n}}{d_1} \\ \frac{a_{21}}{d_2} & \frac{a_{22}}{d_2} & \cdots & \frac{a_{2n}}{d_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{d_n} & \frac{a_{n2}}{d_n} & \cdots & \frac{a_{nn}}{d_n} \end{pmatrix} \tag{16.2}$$

Because $\mathbf{A}$ is assumed to have non-negative elements, this implies that each element of $\mathbf{M}$, namely m_{ij} is also non-negative, as $m_{ij} = \frac{a_{ij}}{d_i} \geq 0$. Consider the sum of the ith row in $\mathbf{M}$; we have

$$\sum_{j=1}^{n} m_{ij} = \sum_{j=1}^{n} \frac{a_{ij}}{d_i} = \frac{d_i}{d_i} = 1 \tag{16.3}$$

Thus, each row in $\mathbf{M}$ sums to 1. This implies that 1 is an eigenvalue of $\mathbf{M}$. In fact, $\lambda_1 = 1$ is the largest eigenvalue of $\mathbf{M}$, and the other eigenvalues satisfy the property that $|\lambda_i| \leq 1$. Also, if G is connected then the eigenvector corresponding to λ_1 is $\mathbf{u}_1 = \frac{1}{\sqrt{n}}(1, 1, \ldots, 1)^T = \frac{1}{\sqrt{n}}\mathbf{1}$. Because $\mathbf{M}$ is not symmetric, its eigenvectors are not necessarily orthogonal.

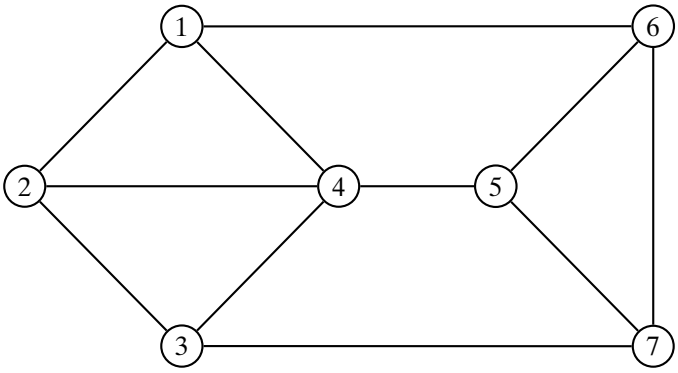


Figure 16.2. Example graph.

Example 16.2. Consider the graph in Figure 16.2. Its adjacency and degree matrices are given as

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \qquad \mathbf{\Delta} = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

The normalized adjacency matrix is as follows:

$$\mathbf{M} = \mathbf{\Delta}^{-1}\mathbf{A} = \begin{pmatrix} 0 & 0.33 & 0 & 0.33 & 0 & 0.33 & 0 \\ 0.33 & 0 & 0.33 & 0.33 & 0 & 0 & 0 \\ 0 & 0.33 & 0 & 0.33 & 0 & 0 & 0.33 \\ 0.25 & 0.25 & 0.25 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0.33 & 0 & 0.33 & 0.33 \\ 0.33 & 0 & 0 & 0 & 0.33 & 0 & 0.33 \\ 0 & 0 & 0.33 & 0 & 0.33 & 0.33 & 0 \end{pmatrix}$$

The eigenvalues of $\mathbf{M}$ sorted in decreasing order are as follows:

$$\lambda_1 = 1 \qquad \lambda_2 = 0.483 \qquad \lambda_3 = 0.206 \qquad \lambda_4 = -0.045$$
$$\lambda_5 = -0.405 \qquad \lambda_6 = -0.539 \qquad \lambda_7 = -0.7$$

The eigenvector corresponding to $\lambda_1 = 1$ is

$$\mathbf{u}_1 = \frac{1}{\sqrt{7}}(1,1,1,1,1,1,1)^T = (0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38)^T$$

Graph Laplacian Matrices

The *Laplacian matrix* of a graph is defined as

$$\begin{aligned}\mathbf{L} &= \mathbf{\Delta} - \mathbf{A} \\ &= \begin{pmatrix} \sum_{j=1}^{n} a_{1j} & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{n} a_{2j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{j=1}^{n} a_{nj} \end{pmatrix} - \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \\ &= \begin{pmatrix} \sum_{j\neq 1} a_{1j} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \sum_{j\neq 2} a_{2j} & \cdots & -a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \sum_{j\neq n} a_{nj} \end{pmatrix} \end{aligned} \tag{16.4}$$

It is interesting to note that $\mathbf{L}$ is a symmetric, positive semidefinite matrix, as for any $\mathbf{c} \in \mathbb{R}^n$, we have

$$\begin{aligned}\mathbf{c}^T\mathbf{L}\mathbf{c} &= \mathbf{c}^T(\mathbf{\Delta} - \mathbf{A})\mathbf{c} = \mathbf{c}^T\mathbf{\Delta}\mathbf{c} - \mathbf{c}^T\mathbf{A}\mathbf{c} \\ &= \sum_{i=1}^{n} d_i c_i^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j a_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} d_i c_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j a_{ij} + \sum_{j=1}^{n} d_j c_j^2\right) \\ &= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} c_i^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j a_{ij} + \sum_{i=j}^{n}\sum_{i=1}^{n} a_{ij} c_j^2\right) \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}(c_i - c_j)^2 \\ &\geq 0 \qquad \text{because } a_{ij} \geq 0 \text{ and } (c_i - c_j)^2 \geq 0 \end{aligned} \tag{16.5}$$

This means that $\mathbf{L}$ has n real, non-negative eigenvalues, which can be arranged in decreasing order as follows: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Because $\mathbf{L}$ is symmetric, its eigenvectors are orthonormal. Further, from Eq. (16.4) we can see that the first column (and the first row) is a linear combination of the remaining columns (rows). That is, if L_i denotes the ith column of $\mathbf{L}$, then we can observe that $L_1 + L_2 + L_3 + \cdots + L_n = \mathbf{0}$. This implies that the rank of $\mathbf{L}$ is at most $n-1$, and the smallest eigenvalue is $\lambda_n = 0$, with the corresponding eigenvector given as $\mathbf{u}_n = \frac{1}{\sqrt{n}}(1, 1, \ldots, 1)^T = \frac{1}{\sqrt{n}}\mathbf{1}$, provided the graph is connected. If the graph is disconnected, then the number of eigenvalues equal to zero specifies the number of connected components in the graph.

Example 16.3. Consider the graph in Figure 16.2, whose adjacency and degree matrices are shown in Example 16.2. The graph Laplacian is given as

$$\mathbf{L} = \mathbf{\Delta} - \mathbf{A} = \begin{pmatrix} 3 & -1 & 0 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & -1 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 & 3 \end{pmatrix}$$

The eigenvalues of $\mathbf{L}$ are as follows:

$$\lambda_1 = 5.618 \qquad \lambda_2 = 4.618 \qquad \lambda_3 = 4.414 \qquad \lambda_4 = 3.382$$
$$\lambda_5 = 2.382 \qquad \lambda_6 = 1.586 \qquad \lambda_7 = 0$$

The eigenvector corresponding to $\lambda_7 = 0$ is

$$\mathbf{u}_7 = \frac{1}{\sqrt{7}}(1,1,1,1,1,1,1)^T = (0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38)^T$$

The *normalized symmetric Laplacian matrix* of a graph is defined as

$$\begin{aligned} \mathbf{L}^s &= \mathbf{\Delta}^{-1/2}\mathbf{L}\mathbf{\Delta}^{-1/2} \\ &= \mathbf{\Delta}^{-1/2}(\mathbf{\Delta} - \mathbf{A})\mathbf{\Delta}^{-1/2} = \mathbf{\Delta}^{-1/2}\mathbf{\Delta}\mathbf{\Delta}^{-1/2} - \mathbf{\Delta}^{-1/2}\mathbf{A}\mathbf{\Delta}^{-1/2} \\ &= \mathbf{I} - \mathbf{\Delta}^{-1/2}\mathbf{A}\mathbf{\Delta}^{-1/2} \end{aligned} \tag{16.6}$$

where $\mathbf{\Delta}^{1/2}$ is the diagonal matrix given as $\mathbf{\Delta}^{1/2}(i,i) = \sqrt{d_i}$, and $\mathbf{\Delta}^{-1/2}$ is the diagonal matrix given as $\mathbf{\Delta}^{-1/2}(i,i) = \frac{1}{\sqrt{d_i}}$ (assuming that $d_i \neq 0$), for $1 \leq i \leq n$. In other words, the normalized Laplacian is given as

$$\begin{aligned} \mathbf{L}^s &= \mathbf{\Delta}^{-1/2}\mathbf{L}\mathbf{\Delta}^{-1/2} \\ &= \begin{pmatrix} \frac{\sum_{j\neq 1} a_{1j}}{\sqrt{d_1 d_1}} & -\frac{a_{12}}{\sqrt{d_1 d_2}} & \cdots & -\frac{a_{1n}}{\sqrt{d_1 d_n}} \\ -\frac{a_{21}}{\sqrt{d_2 d_1}} & \frac{\sum_{j\neq 2} a_{2j}}{\sqrt{d_2 d_2}} & \cdots & -\frac{a_{2n}}{\sqrt{d_2 d_n}} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{\sqrt{d_n d_1}} & -\frac{a_{n2}}{\sqrt{d_n d_2}} & \cdots & \frac{\sum_{j\neq n} a_{nj}}{\sqrt{d_n d_n}} \end{pmatrix} \end{aligned} \tag{16.7}$$

Like the derivation in Eq. (16.5), we can show that $\mathbf{L}^s$ is also positive semidefinite because for any $\mathbf{c} \in \mathbb{R}^d$, we get

$$\mathbf{c}^T\mathbf{L}^s\mathbf{c} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\left(\frac{c_i}{\sqrt{d_i}} - \frac{c_j}{\sqrt{d_j}}\right)^2 \geq 0 \tag{16.8}$$

Further, if L_i^s denotes the ith column of $\mathbf{L}^s$, then from Eq. (16.7) we can see that

$$\sqrt{d_1}L_1^s + \sqrt{d_2}L_2^s + \sqrt{d_3}L_3^s + \cdots + \sqrt{d_n}L_n^s = \mathbf{0}$$

That is, the first column is a linear combination of the other columns, which means that $\mathbf{L}^s$ has rank at most $n-1$, with the smallest eigenvalue $\lambda_n = 0$, and the corresponding eigenvector $\frac{1}{\sqrt{\sum_i d_i}}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})^T = \frac{1}{\sqrt{\sum_i d_i}}\boldsymbol{\Delta}^{1/2}\mathbf{1}$. Combined with the fact that $\mathbf{L}^s$ is positive semidefinite, we conclude that $\mathbf{L}^s$ has n (not necessarily distinct) real, positive eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n = 0$.

Example 16.4. We continue with Example 16.3. For the graph in Figure 16.2, its normalized symmetric Laplacian is given as

$$\mathbf{L}^s = \begin{pmatrix} 1 & -0.33 & 0 & -0.29 & 0 & -0.33 & 0 \\ -0.33 & 1 & -0.33 & -0.29 & 0 & 0 & 0 \\ 0 & -0.33 & 1 & -0.29 & 0 & 0 & -0.33 \\ -0.29 & -0.29 & -0.29 & 1 & -0.29 & 0 & 0 \\ 0 & 0 & 0 & -0.29 & 1 & -0.33 & -0.33 \\ -0.33 & 0 & 0 & 0 & -0.33 & 1 & -0.33 \\ 0 & 0 & -0.33 & 0 & -0.33 & -0.33 & 1 \end{pmatrix}$$

The eigenvalues of $\mathbf{L}^s$ are as follows:

$$\lambda_1 = 1.7 \qquad \lambda_2 = 1.539 \qquad \lambda_3 = 1.405 \qquad \lambda_4 = 1.045$$
$$\lambda_5 = 0.794 \qquad \lambda_6 = 0.517 \qquad \lambda_7 = 0$$

The eigenvector corresponding to $\lambda_7 = 0$ is

$$\begin{aligned} \mathbf{u}_7 &= \frac{1}{\sqrt{22}}(\sqrt{3}, \sqrt{3}, \sqrt{3}, \sqrt{4}, \sqrt{3}, \sqrt{3}, \sqrt{3})^T \\ &= (0.37, 0.37, 0.37, 0.43, 0.37, 0.37, 0.37)^T \end{aligned}$$

The *normalized asymmetric Laplacian* matrix is defined as

$$\begin{aligned} \mathbf{L}^a &= \boldsymbol{\Delta}^{-1}\mathbf{L} \\ &= \boldsymbol{\Delta}^{-1}(\boldsymbol{\Delta} - \mathbf{A}) = \mathbf{I} - \boldsymbol{\Delta}^{-1}\mathbf{A} \\ &= \begin{pmatrix} \frac{\sum_{j\neq 1} a_{1j}}{d_1} & -\frac{a_{12}}{d_1} & \cdots & -\frac{a_{1n}}{d_1} \\ -\frac{a_{21}}{d_2} & \frac{\sum_{j\neq 2} a_{2j}}{d_2} & \cdots & -\frac{a_{2n}}{d_2} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{d_n} & -\frac{a_{n2}}{d_n} & \cdots & \frac{\sum_{j\neq n} a_{nj}}{d_n} \end{pmatrix} \end{aligned} \tag{16.9}$$

Consider the eigenvalue equation for the symmetric Laplacian $\mathbf{L}^s$:

$$\mathbf{L}^s\mathbf{u} = \lambda\mathbf{u}$$

Left multiplying by $\mathbf{\Delta}^{-1/2}$ on both sides, we get

$$\begin{aligned}
\mathbf{\Delta}^{-1/2}\mathbf{L}^s\mathbf{u} &= \lambda\mathbf{\Delta}^{-1/2}\mathbf{u} \\
\mathbf{\Delta}^{-1/2}\left(\mathbf{\Delta}^{-1/2}\mathbf{L}\mathbf{\Delta}^{-1/2}\right)\mathbf{u} &= \lambda\mathbf{\Delta}^{-1/2}\mathbf{u} \\
\mathbf{\Delta}^{-1}\mathbf{L}\left(\mathbf{\Delta}^{-1/2}\mathbf{u}\right) &= \lambda\left(\mathbf{\Delta}^{-1/2}\mathbf{u}\right) \\
\mathbf{L}^a\mathbf{v} &= \lambda\mathbf{v}
\end{aligned}$$

where $\mathbf{v} = \mathbf{\Delta}^{-1/2}\mathbf{u}$ is an eigenvector of $\mathbf{L}^a$, and $\mathbf{u}$ is an eigenvector of $\mathbf{L}^s$. Further, $\mathbf{L}^a$ has the same set of eigenvalues as $\mathbf{L}^s$, which means that $\mathbf{L}^a$ is a positive semi-definite matrix with n real eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n = 0$. From Eq. (16.9) we can see that if L_i^a denotes the ith column of $\mathbf{L}^a$, then $L_1^a + L_2^a + \cdots + L_n^a = 0$, which implies that $\mathbf{v}_n = \frac{1}{\sqrt{n}}\mathbf{1}$ is the eigenvector corresponding to the smallest eigenvalue $\lambda_n = 0$.

Example 16.5. For the graph in Figure 16.2, its normalized asymmetric Laplacian matrix is given as

$$\mathbf{L}^a = \mathbf{\Delta}^{-1}\mathbf{L} = \begin{pmatrix}
1 & -0.33 & 0 & -0.33 & 0 & -0.33 & 0 \\
-0.33 & 1 & -0.33 & -0.33 & 0 & 0 & 0 \\
0 & -0.33 & 1 & -0.33 & 0 & 0 & -0.33 \\
-0.25 & -0.25 & -0.25 & 1 & -0.25 & 0 & 0 \\
0 & 0 & 0 & -0.33 & 1 & -0.33 & -0.33 \\
-0.33 & 0 & 0 & 0 & -0.33 & 1 & -0.33 \\
0 & 0 & -0.33 & 0 & -0.33 & -0.33 & 1
\end{pmatrix}$$

The eigenvalues of $\mathbf{L}^a$ are identical to those for $\mathbf{L}^s$, namely

$$\lambda_1 = 1.7 \qquad \lambda_2 = 1.539 \qquad \lambda_3 = 1.405 \qquad \lambda_4 = 1.045$$
$$\lambda_5 = 0.794 \qquad \lambda_6 = 0.517 \qquad \lambda_7 = 0$$

The eigenvector corresponding to $\lambda_7 = 0$ is

$$\mathbf{u}_7 = \frac{1}{\sqrt{7}}(1, 1, 1, 1, 1, 1, 1)^T = (0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38)^T$$

16.2 CLUSTERING AS GRAPH CUTS

A *k-way cut* in a graph is a partitioning or clustering of the vertex set, given as $\mathcal{C} = \{C_1, \ldots, C_k\}$, such that $C_i \neq \emptyset$ for all i, $C_i \cap C_j = \emptyset$ for all i, j, and $V = \bigcup_i C_i$. We require $\mathcal{C}$ to optimize some objective function that captures the intuition that nodes within a cluster should have high similarity, and nodes from different clusters should have low similarity.

Given a weighted graph G defined by its similarity matrix [Eq. (16.1)], let $S, T \subseteq V$ be any two subsets of the vertices. We denote by $W(S, T)$ the sum of the weights on all

edges with one vertex in S and the other in T, given as

$$W(S,T) = \sum_{v_i \in S} \sum_{v_j \in T} a_{ij}$$

Given $S \subseteq V$, we denote by $\overline{S}$ the complementary set of vertices, that is, $\overline{S} = V - S$. A *(vertex) cut* in a graph is defined as a partitioning of V into $S \subset V$ and $\overline{S}$. The *weight of the cut* or *cut weight* is defined as the sum of all the weights on edges between vertices in S and $\overline{S}$, given as $W(S, \overline{S})$.

Given a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ comprising k clusters, the *size* of a cluster C_i is the number of nodes in the cluster, given as $|C_i|$. The *volume* of a cluster C_i is defined as the sum of all the weights on edges with one end in cluster C_i:

$$vol(C_i) = \sum_{v_j \in C_i} d_j = \sum_{v_j \in C_i} \sum_{v_r \in V} a_{jr} = W(C_i, V)$$

Let $\mathbf{c}_i \in \{0,1\}^n$ be the *cluster indicator vector* that records the cluster membership for cluster C_i, defined as

$$c_{ij} = \begin{cases} 1 & \text{if } v_j \in C_i \\ 0 & \text{if } v_j \notin C_i \end{cases}$$

Because a clustering creates pairwise disjoint clusters, we immediately have

$$\mathbf{c}_i^T \mathbf{c}_j = 0$$

Further, the cluster size can be written as

$$|C_i| = \mathbf{c}_i^T \mathbf{c}_i = \|\mathbf{c}_i\|^2$$

The following identities allow us to express the weight of a cut in terms of matrix operations. Let us derive an expression for the sum of the weights for all edges with one end in C_i. These edges include internal cluster edges (with both ends in C_i), as well as external cluster edges (with the other end in another cluster $C_{j \neq i}$).

$$\begin{aligned} vol(C_i) = W(C_i, V) &= \sum_{v_r \in C_i} d_r = \sum_{v_r \in C_i} c_{ir} d_r c_{ir} \\ &= \sum_{r=1}^{n} \sum_{s=1}^{n} c_{ir} \mathbf{\Delta}_{rs} c_{is} = \mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i \end{aligned} \tag{16.10}$$

Consider the sum of weights of all internal edges:

$$\begin{aligned} W(C_i, C_i) &= \sum_{v_r \in C_i} \sum_{v_s \in C_i} a_{rs} \\ &= \sum_{r=1}^{n} \sum_{s=1}^{n} c_{ir} a_{rs} c_{is} = \mathbf{c}_i^T \mathbf{A} \mathbf{c}_i \end{aligned} \tag{16.11}$$

We can get the sum of weights for all the external edges, or the cut weight by subtracting Eq. (16.11) from Eq. (16.10), as follows:

$$W(C_i, \overline{C_i}) = \sum_{v_r \in C_i} \sum_{v_s \in V - C_i} a_{rs} = W(C_i, V) - W(C_i, C_i)$$
$$= \mathbf{c}_i(\mathbf{\Delta} - \mathbf{A})\mathbf{c}_i = \mathbf{c}_i^T \mathbf{L} \mathbf{c}_i \tag{16.12}$$

Example 16.6. Consider the graph in Figure 16.2. Assume that $C_1 = \{1, 2, 3, 4\}$ and $C_2 = \{5, 6, 7\}$ are two clusters. Their cluster indicator vectors are given as

$$\mathbf{c}_1 = (1, 1, 1, 1, 0, 0, 0)^T \qquad \mathbf{c}_2 = (0, 0, 0, 0, 1, 1, 1)^T$$

As required, we have $\mathbf{c}_1^T \mathbf{c}_2 = 0$, and $\mathbf{c}_1^T \mathbf{c}_1 = \|\mathbf{c}_1\|^2 = 4$ and $\mathbf{c}_2^T \mathbf{c}_2 = 3$ give the cluster sizes. Consider the cut weight between C_1 and C_2. Because there are three edges between the two clusters, we have $W(C_1, \overline{C_1}) = W(C_1, C_2) = 3$. Using the Laplacian matrix from Example 16.3, by Eq. (16.12) we have

$$W(C_1, \overline{C_1}) = \mathbf{c}_1^T \mathbf{L} \mathbf{c}_1$$
$$= (1, 1, 1, 1, 0, 0, 0) \begin{pmatrix} 3 & -1 & 0 & -1 & 0 & -1 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & -1 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$
$$= (1, 0, 1, 1, -1, -1, -1)(1, 1, 1, 1, 0, 0, 0)^T = 3$$

16.2.1 Clustering Objective Functions: Ratio and Normalized Cut

The clustering objective function can be formulated as an optimization problem over the k-way cut $\mathcal{C} = \{C_1, \ldots, C_k\}$. We consider two common minimization objectives, namely ratio and normalized cut. We consider maximization objectives in Section 16.2.3, after describing the spectral clustering algorithm.

Ratio Cut

The *ratio cut* objective is defined over a k-way cut as follows:

$$\min_{\mathcal{C}} J_{rc}(\mathcal{C}) = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{|C_i|} = \sum_{i=1}^{k} \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{c}_i} = \sum_{i=1}^{k} \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\|\mathbf{c}_i\|^2} \tag{16.13}$$

where we make use of Eq. (16.12), that is, $W(C_i, \overline{C_i}) = \mathbf{c}_i^T \mathbf{L} \mathbf{c}_i$.

Ratio cut tries to minimize the sum of the similarities from a cluster C_i to other points not in the cluster $\overline{C_i}$, taking into account the size of each cluster. One can observe that the objective function has a lower value when the cut weight is minimized and when the cluster size is large.

Unfortunately, for binary cluster indicator vectors $\mathbf{c}_i$, the ratio cut objective is NP-hard. An obvious relaxation is to allow $\mathbf{c}_i$ to take on any real value. In this case, we can rewrite the objective as

$$\min_{\mathcal{C}} J_{rc}(\mathcal{C}) = \sum_{i=1}^{k} \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\|\mathbf{c}_i\|^2} = \sum_{i=1}^{k} \left(\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}\right)^T \mathbf{L} \left(\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}\right) = \sum_{i=1}^{k} \mathbf{u}_i^T \mathbf{L} \mathbf{u}_i \tag{16.14}$$

where $\mathbf{u}_i = \frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}$ is the unit vector in the direction of $\mathbf{c}_i \in \mathbb{R}^n$, that is, $\mathbf{c}_i$ is assumed to be an arbitrary real vector.

To minimize J_{rc} we take its derivative with respect to $\mathbf{u}_i$ and set it to the zero vector. To incorporate the constraint that $\mathbf{u}_i^T \mathbf{u}_i = 1$, we introduce the Lagrange multiplier λ_i for each cluster C_i. We have

$$\frac{\partial}{\partial \mathbf{u}_i} \left(\sum_{i=1}^{k} \mathbf{u}_i^T \mathbf{L} \mathbf{u}_i + \sum_{i=1}^{n} \lambda_i (1 - \mathbf{u}_i^T \mathbf{u}_i) \right) = \mathbf{0}, \text{ which implies that}$$

$$2\mathbf{L}\mathbf{u}_i - 2\lambda_i \mathbf{u}_i = \mathbf{0}, \text{ and thus}$$

$$\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{u}_i \tag{16.15}$$

This implies that $\mathbf{u}_i$ is one of the eigenvectors of the Laplacian matrix $\mathbf{L}$, corresponding to the eigenvalue λ_i. Using Eq. (16.15), we can see that

$$\mathbf{u}_i^T \mathbf{L} \mathbf{u}_i = \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \lambda_i$$

which in turn implies that to minimize the ratio cut objective [Eq. (16.14)], we should choose the k smallest eigenvalues, and the corresponding eigenvectors, so that

$$\begin{aligned} \min_{\mathcal{C}} J_{rc}(\mathcal{C}) &= \mathbf{u}_n^T \mathbf{L} \mathbf{u}_n + \cdots + \mathbf{u}_{n-k+1}^T \mathbf{L} \mathbf{u}_{n-k+1} \\ &= \lambda_n + \cdots + \lambda_{n-k+1} \end{aligned} \tag{16.16}$$

where we assume that the eigenvalues have been sorted so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Noting that the smallest eigenvalue of $\mathbf{L}$ is $\lambda_n = 0$, the k smallest eigenvalues are as follows: $0 = \lambda_n \leq \lambda_{n-1} \leq \lambda_{n-k+1}$. The corresponding eigenvectors $\mathbf{u}_n, \mathbf{u}_{n-1}, \ldots, \mathbf{u}_{n-k+1}$ represent the relaxed cluster indicator vectors. However, because $\mathbf{u}_n = \frac{1}{\sqrt{n}}\mathbf{1}$, it does not provide any guidance on how to separate the graph nodes if the graph is connected.

Normalized Cut

Normalized cut is similar to ratio cut, except that it divides the cut weight of each cluster by the volume of a cluster instead of its size. The objective function is given as

$$\min_{\mathcal{C}} J_{nc}(\mathcal{C}) = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{vol(C_i)} = \sum_{i=1}^{k} \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i} \tag{16.17}$$

where we use Eqs. (16.12) and (16.10), that is, $W(C_i, \overline{C_i}) = \mathbf{c}_i^T \mathbf{L} \mathbf{c}_i$ and $vol(C_i) = \mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i$, respectively. The J_{nc} objective function has lower values when the cut weight is low and when the cluster volume is high, as desired.

As in the case of ratio cut, we can obtain an optimal solution to the normalized cut objective if we relax the condition that $\mathbf{c}_i$ be a binary cluster indicator vector. Instead we assume $\mathbf{c}_i$ to be an arbitrary real vector. Using the observation that the diagonal degree matrix $\mathbf{\Delta}$ can be written as $\mathbf{\Delta} = \mathbf{\Delta}^{1/2}\mathbf{\Delta}^{1/2}$, and using the fact that $\mathbf{I} = \mathbf{\Delta}^{1/2}\mathbf{\Delta}^{-1/2}$ and $\mathbf{\Delta}^T = \mathbf{\Delta}$ (because $\mathbf{\Delta}$ is diagonal), we can rewrite the normalized cut objective in terms of the normalized symmetric Laplacian, as follows:

$$\begin{aligned}
\min_{\mathcal{C}} J_{nc}(\mathcal{C}) &= \sum_{i=1}^{k} \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i} \\
&= \sum_{i=1}^{k} \frac{\mathbf{c}_i^T \big(\mathbf{\Delta}^{1/2}\mathbf{\Delta}^{-1/2}\big)\mathbf{L}\big(\mathbf{\Delta}^{-1/2}\mathbf{\Delta}^{1/2}\big)\mathbf{c}_i}{\mathbf{c}_i^T \big(\mathbf{\Delta}^{1/2}\mathbf{\Delta}^{1/2}\big)\mathbf{c}_i} \\
&= \sum_{i=1}^{k} \frac{(\mathbf{\Delta}^{1/2}\mathbf{c}_i)^T(\mathbf{\Delta}^{-1/2}\mathbf{L}\mathbf{\Delta}^{-1/2})(\mathbf{\Delta}^{1/2}\mathbf{c}_i)}{(\mathbf{\Delta}^{1/2}\mathbf{c}_i)^T(\mathbf{\Delta}^{1/2}\mathbf{c}_i)} \\
&= \sum_{i=1}^{k} \left(\frac{\mathbf{\Delta}^{1/2}\mathbf{c}_i}{\big\|\mathbf{\Delta}^{1/2}\mathbf{c}_i\big\|}\right)^T \mathbf{L}^s \left(\frac{\mathbf{\Delta}^{1/2}\mathbf{c}_i}{\big\|\mathbf{\Delta}^{1/2}\mathbf{c}_i\big\|}\right) \\
&= \sum_{i=1}^{k} \mathbf{u}_i^T \mathbf{L}^s \mathbf{u}_i
\end{aligned}$$

where $\mathbf{u}_i = \frac{\mathbf{\Delta}^{1/2}\mathbf{c}_i}{\|\mathbf{\Delta}^{1/2}\mathbf{c}_i\|}$ is the unit vector in the direction of $\mathbf{\Delta}^{1/2}\mathbf{c}_i$. Following the same approach as in Eq. (16.15), we conclude that the normalized cut objective is optimized by selecting the k smallest eigenvalues of the normalized Laplacian matrix $\mathbf{L}^s$, namely $0 = \lambda_n \leq \cdots \leq \lambda_{n-k+1}$.

The normalized cut objective [Eq. (16.17)], can also be expressed in terms of the normalized asymmetric Laplacian, by differentiating Eq. (16.17) with respect to $\mathbf{c}_i$ and setting the result to the zero vector. Noting that all terms other than that for $\mathbf{c}_i$ are constant with respect to $\mathbf{c}_i$, we have:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{c}_i}\left(\sum_{j=1}^{k} \frac{\mathbf{c}_j^T \mathbf{L} \mathbf{c}_j}{\mathbf{c}_j^T \mathbf{\Delta} \mathbf{c}_j}\right) = \frac{\partial}{\partial \mathbf{c}_i}\left(\frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i}\right) &= \mathbf{0} \\
\frac{\mathbf{L}\mathbf{c}_i(\mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i) - \mathbf{\Delta}\mathbf{c}_i(\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i)}{(\mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i)^2} &= \mathbf{0} \\
\mathbf{L}\mathbf{c}_i = \left(\frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i}\right)\mathbf{\Delta}\mathbf{c}_i & \\
\mathbf{\Delta}^{-1}\mathbf{L}\mathbf{c}_i = \lambda_i \mathbf{c}_i & \\
\mathbf{L}^a \mathbf{c}_i = \lambda_i \mathbf{c}_i &
\end{aligned}$$

where $\lambda_i = \frac{\mathbf{c}_i^T \mathbf{L} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{\Delta} \mathbf{c}_i}$ is the eigenvalue corresponding to the ith eigenvector $\mathbf{c}_i$ of the asymmetric Laplacian matrix $\mathbf{L}^a$. To minimize the normalized cut objective we therefore choose the k smallest eigenvalues of $\mathbf{L}^a$, namely, $0 = \lambda_n \leq \cdots \leq \lambda_{n-k+1}$.

To derive the clustering, for $\mathbf{L}^a$, we can use the corresponding eigenvectors $\mathbf{u}_n, \ldots, \mathbf{u}_{n-k+1}$, with $\mathbf{c}_i = \mathbf{u}_i$ representing the real-valued cluster indicator vectors.

However, note that for $\mathbf{L}^a$, we have $\mathbf{c}_n = \mathbf{u}_n = \frac{1}{\sqrt{n}}\mathbf{1}$. Further, for the normalized symmetric Laplacian $\mathbf{L}^s$, the real-valued cluster indicator vectors are given as $\mathbf{c}_i = \mathbf{\Delta}^{-1/2}\mathbf{u}_i$, which again implies that $\mathbf{c}_n = \frac{1}{\sqrt{n}}\mathbf{1}$. This means that the eigenvector $\mathbf{u}_n$ corresponding to the smallest eigenvalue $\lambda_n = 0$ does not by itself contain any useful information for clustering if the graph is connected.

16.2.2 Spectral Clustering Algorithm

Algorithm 16.1 gives the pseudo-code for the spectral clustering approach. We assume that the underlying graph is connected. The method takes a dataset $\mathbf{D}$ as input and computes the similarity matrix $\mathbf{A}$. Alternatively, the matrix $\mathbf{A}$ may be directly input as well. Depending on the objective function, we choose the corresponding matrix $\mathbf{B}$. For instance, for normalized cut $\mathbf{B}$ is chosen to be either $\mathbf{L}^s$ or $\mathbf{L}^a$, whereas for ratio cut we choose $\mathbf{B} = \mathbf{L}$. Next, we compute the k smallest eigenvalues and eigenvectors of $\mathbf{B}$. However, the main problem we face is that the eigenvectors $\mathbf{u}_i$ are not binary, and thus it is not immediately clear how we can assign points to clusters. One solution to this problem is to treat the $n \times k$ matrix of eigenvectors as a new data matrix:

$$\mathbf{U} = \begin{pmatrix} | & | & & | \\ \mathbf{u}_n & \mathbf{u}_{n-1} & \cdots & \mathbf{u}_{n-k+1} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} u_{n,1} & u_{n-1,1} & \cdots & u_{n-k+1,1} \\ u_{n2} & u_{n-1,2} & \cdots & u_{n-k+1,2} \\ | & | & \cdots & | \\ u_{n,n} & u_{n-1,n} & \cdots & u_{n-k+1,n} \end{pmatrix} \tag{16.18}$$

Next, we normalize each row of $\mathbf{U}$ to obtain the unit vector:

$$\mathbf{y}_i = \frac{1}{\sqrt{\sum_{j=1}^{k} u_{n-j+1,i}^2}} (u_{n,i},\ u_{n-1,i},\ \ldots,\ u_{n-k+1,i})^T \tag{16.19}$$

which yields the new normalized data matrix $\mathbf{Y} \in \mathbb{R}^{n \times k}$ comprising n points in a reduced k dimensional space:

$$\mathbf{Y} = \begin{pmatrix} — & \mathbf{y}_1^T & — \\ — & \mathbf{y}_2^T & — \\ & \vdots & \\ — & \mathbf{y}_n^T & — \end{pmatrix}$$

ALGORITHM 16.1. Spectral Clustering Algorithm

SPECTRAL CLUSTERING ($\mathbf{D}, k$):

1 Compute the similarity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
2 **if** *ratio cut* **then** $\mathbf{B} \leftarrow \mathbf{L}$
3 **else if** *normalized cut* **then** $\mathbf{B} \leftarrow \mathbf{L}^s$ or $\mathbf{L}^a$
4 Solve $\mathbf{B}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ for $i = n, \ldots, n-k+1$, where $\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_{n-k+1}$
5 $\mathbf{U} \leftarrow \begin{pmatrix} \mathbf{u}_n & \mathbf{u}_{n-1} & \cdots & \mathbf{u}_{n-k+1} \end{pmatrix}$
6 $\mathbf{Y} \leftarrow$ normalize rows of $\mathbf{U}$ using Eq. (16.19)
7 $\mathcal{C} \leftarrow \{C_1, \ldots, C_k\}$ via K-means on $\mathbf{Y}$

We can now cluster the new points in $\mathbf{Y}$ into k clusters via the K-means algorithm or any other fast clustering method, as it is expected that the clusters are well-separated in the k-dimensional eigen-space. Note that for $\mathbf{L}$, $\mathbf{L}^s$, and $\mathbf{L}^a$, the cluster indicator vector corresponding to the smallest eigenvalue $\lambda_n = 0$ is a vector of all 1's, which does not provide any information about how to separate the nodes. The real information for clustering is contained in eigenvectors starting from the second smallest eigenvalue. However, if the graph is disconnected, then even the eigenvector corresponding to λ_n can contain information valuable for clustering. Thus, we retain all k eigenvectors in $\mathbf{U}$ in Eq. (16.18).

Strictly speaking, the normalization step [Eq. (16.19)] is recommended only for the normalized symmetric Laplacian $\mathbf{L}^s$. This is because the eigenvectors of $\mathbf{L}^s$ and the cluster indicator vectors are related as $\mathbf{\Delta}^{1/2}\mathbf{c}_i = \mathbf{u}_i$. The jth entry of $\mathbf{u}_i$, corresponding to vertex v_j, is given as

$$u_{ij} = \frac{\sqrt{d_j}c_{ij}}{\sqrt{\sum_{r=1}^{n} d_r c_{ir}^2}}$$

If vertex degrees vary a lot, vertices with small degrees would have very small values u_{ij}. This can cause problems for K-means for correctly clustering these vertices. The normalization step helps alleviate this problem for $\mathbf{L}^s$, though it can also help other objectives.

Computational Complexity

The computational complexity of the spectral clustering algorithm is $O(n^3)$, because computing the eigenvectors takes that much time. However, if the graph is sparse, the complexity to compute the eigenvectors is $O(mn)$ where m is the number of edges in the graph. In particular, if $m = O(n)$, then the complexity reduces to $O(n^2)$. Running the K-means method on $\mathbf{Y}$ takes $O(tnk^2)$ time, where t is the number of iterations K-means takes to converge.

Example 16.7. Consider the normalized cut approach applied to the graph in Figure 16.2. Assume that we want to find $k = 2$ clusters. For the normalized asymmetric Laplacian matrix from Example 16.5, we compute the eigenvectors, $\mathbf{v}_7$ and $\mathbf{v}_6$, corresponding to the two smallest eigenvalues, $\lambda_7 = 0$ and $\lambda_6 = 0.517$. The matrix composed of both the eigenvectors is given as

$$\mathbf{U} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 \\ \hline -0.378 & -0.226 \\ -0.378 & -0.499 \\ -0.378 & -0.226 \\ -0.378 & -0.272 \\ -0.378 & 0.425 \\ -0.378 & 0.444 \\ -0.378 & 0.444 \end{pmatrix}$$

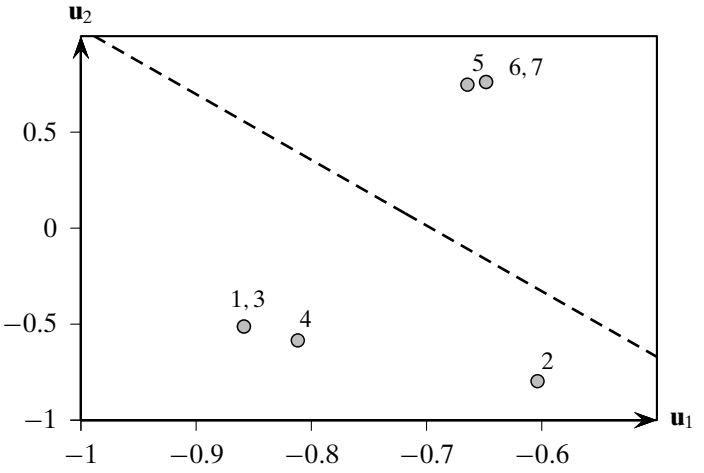


Figure 16.3. K-means on spectral dataset $\mathbf{Y}$.

We treat the ith component of $\mathbf{u}_1$ and $\mathbf{u}_2$ as the ith point $(u_{1i}, u_{2i}) \in \mathbb{R}^2$, and after normalizing all points to have unit length we obtain the new dataset:

$$\mathbf{Y} = \begin{pmatrix} -0.859 & -0.513 \\ -0.604 & -0.797 \\ -0.859 & -0.513 \\ -0.812 & -0.584 \\ -0.664 & 0.747 \\ -0.648 & 0.761 \\ -0.648 & 0.761 \end{pmatrix}$$

For instance the first point is computed as

$$\mathbf{y}_1 = \frac{1}{\sqrt{(-0.378)^2 + (-0.226^2)}}(-0.378, -0.226)^T = (-0.859, -0.513)^T$$

Figure 16.3 plots the new dataset $\mathbf{Y}$. Clustering the points into $k = 2$ groups using K-means yields the two clusters $C_1 = \{1, 2, 3, 4\}$ and $C_2 = \{5, 6, 7\}$.

Example 16.8. We apply spectral clustering on the Iris graph in Figure 16.1 using the normalized cut objective with the asymmetric Laplacian matrix $\mathbf{L}^a$. Figure 16.4 shows the $k = 3$ clusters. Comparing them with the true Iris classes (not used in the clustering), we obtain the contingency table shown in Table 16.1, indicating the number of points clustered correctly (on the main diagonal) and incorrectly (off-diagonal). We can see that cluster C_1 corresponds mainly to `iris-setosa`, C_2 to `iris-virginica`, and C_3 to `iris-versicolor`. The latter two are more difficult to separate. In total there are 18 points that are misclustered when compared to the true Iris types.

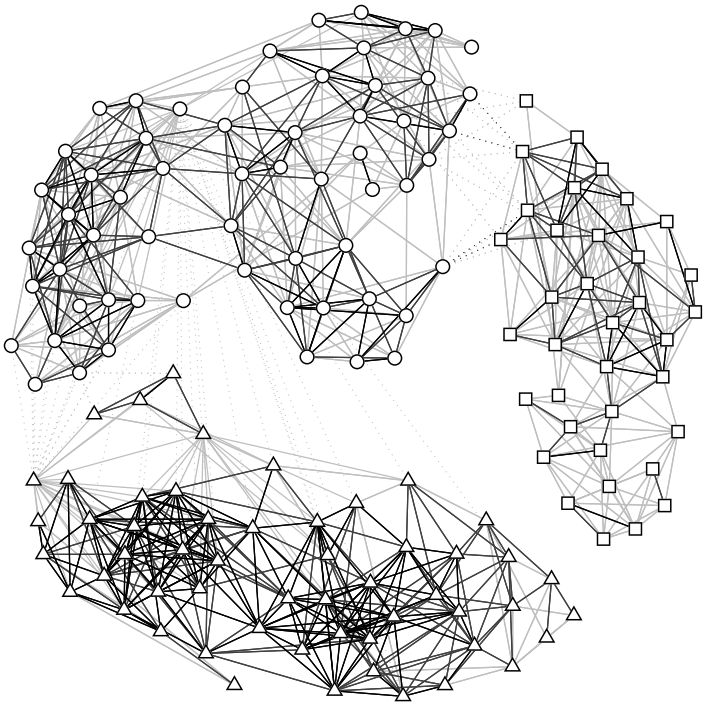


Figure 16.4. Normalized cut on Iris graph.

Table 16.1. Contingency table: clusters versus Iris types

	iris-setosa	iris-virginica	iris-versicolor
C_1 (triangle)	50	0	4
C_2 (square)	0	36	0
C_3 (circle)	0	14	46

16.2.3 Maximization Objectives: Average Cut and Modularity

We now discuss two clustering objective functions that can be formulated as maximization problems over the k-way cut $\mathcal{C} = \{C_1, \ldots, C_k\}$. These include average weight and modularity. We also explore their connections with normalized cut and kernel K-means.

Average Weight

The *average weight* objective is defined as

$$\max_{\mathcal{C}} J_{aw}(\mathcal{C}) = \sum_{i=1}^{k} \frac{W(C_i, C_i)}{|C_i|} = \sum_{i=1}^{k} \frac{\mathbf{c}_i^T \mathbf{A} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{c}_i} \tag{16.20}$$

where we used the equivalence $W(C_i, C_i) = \mathbf{c}_i^T \mathbf{A} \mathbf{c}_i$ established in Eq. (16.11). Instead of trying to minimize the weights on edges between clusters as in ratio cut, average weight tries to maximize the within cluster weights. The problem of maximizing J_{aw} for binary cluster indicator vectors is also NP-hard; we can obtain a solution by relaxing

the constraint on $\mathbf{c}_i$, by assuming that it can take on any real values for its elements. This leads to the relaxed objective

$$\max_{\mathcal{C}} J_{aw}(\mathcal{C}) = \sum_{i=1}^{k} \mathbf{u}_i^T \mathbf{A} \mathbf{u}_i \tag{16.21}$$

where $\mathbf{u}_i = \frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}$. Following the same approach as in Eq. (16.15), we can maximize the objective by selecting the k largest eigenvalues of $\mathbf{A}$, and the corresponding eigenvectors

$$\begin{aligned}\max_{\mathcal{C}} J_{aw}(\mathcal{C}) &= \mathbf{u}_1^T \mathbf{A} \mathbf{u}_1 + \cdots + \mathbf{u}_k^T \mathbf{A} \mathbf{u}_k \\ &= \lambda_1 + \cdots + \lambda_k\end{aligned}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

If we assume that $\mathbf{A}$ is the weighted adjacency matrix obtained from a symmetric and positive semidefinite kernel, that is, with $a_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, then $\mathbf{A}$ will be positive semidefinite and will have non-negative real eigenvalues. In general, if we threshold $\mathbf{A}$ or if $\mathbf{A}$ is the unweighted adjacency matrix for an undirected graph, then even though $\mathbf{A}$ is symmetric, it may not be positive semidefinite. This means that in general $\mathbf{A}$ can have negative eigenvalues, though they are all real. Because J_{aw} is a maximization problem, this means that we must consider only the positive eigenvalues and the corresponding eigenvectors.

Example 16.9. For the graph in Figure 16.2, with the adjacency matrix shown in Example 16.3, its eigenvalues are as follows:

$$\lambda_1 = 3.18 \qquad \lambda_2 = 1.49 \qquad \lambda_3 = 0.62 \qquad \lambda_4 = -0.15$$
$$\lambda_5 = -1.27 \qquad \lambda_6 = -1.62 \qquad \lambda_7 = -2.25$$

We can see that the eigenvalues can be negative, as $\mathbf{A}$ is the adjacency graph and is not positive semidefinite.

Average Weight and Kernel K-means The average weight objective leads to an interesting connection between kernel K-means and graph cuts. If the weighted adjacency matrix $\mathbf{A}$ represents the kernel value between a pair of points, so that $a_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, then we may use the sum of squared errors objective [Eq. (13.3)] of kernel K-means for graph clustering. The SSE objective is given as

$$\begin{aligned}\min_{\mathcal{C}} J_{sse}(\mathcal{C}) &= \sum_{j=1}^{n} K(\mathbf{x}_j, \mathbf{x}_j) - \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{\mathbf{x}_r \in C_i} \sum_{\mathbf{x}_s \in C_i} K(\mathbf{x}_r, \mathbf{x}_s) \\ &= \sum_{j=1}^{n} a_{jj} - \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{v_r \in C_i} \sum_{v_s \in C_i} a_{rs}\end{aligned}$$

$$= \sum_{j=1}^{n} a_{jj} - \sum_{i=1}^{k} \frac{\mathbf{c}_i^T \mathbf{A} \mathbf{c}_i}{\mathbf{c}_i^T \mathbf{c}_i}$$

$$= \sum_{j=1}^{n} a_{jj} - J_{aw}(\mathcal{C}) \tag{16.22}$$

We can observe that because $\sum_{j=1}^{n} a_{jj}$ is independent of the clustering, minimizing the SSE objective is the same as maximizing the average weight objective. In particular, if a_{ij} represents the linear kernel $\mathbf{x}_i^T \mathbf{x}_j$ between the nodes, then maximizing the average weight objective [Eq. (16.20)] is equivalent to minimizing the regular K-means SSE objective [Eq. (13.1)]. Thus, spectral clustering using J_{aw} and kernel K-means represent two different approaches to solve the same problem. Kernel K-means tries to solve the NP-hard problem by using a greedy iterative approach to directly optimize the SSE objective, whereas the graph cut formulation tries to solve the same NP-hard problem by optimally solving a relaxed problem.

Modularity

Informally, modularity is defined as the difference between the observed and expected fraction of edges within a cluster. It measures the extent to which nodes of the same type (in our case, the same cluster) are linked to each other.

Unweighted Graphs Let us assume for the moment that the graph G is unweighted, and that $\mathbf{A}$ is its binary adjacency matrix. The number of edges within a cluster C_i is given as

$$\frac{1}{2} \sum_{v_r \in C_i} \sum_{v_s \in C_i} a_{rs}$$

where we divide by $\frac{1}{2}$ because each edge is counted twice in the summation. Over all the clusters, the observed number of edges within the same cluster is given as

$$\frac{1}{2} \sum_{i=1}^{k} \sum_{v_r \in C_i} \sum_{v_s \in C_i} a_{rs} \tag{16.23}$$

Let us compute the expected number of edges between any two vertices v_r and v_s, assuming that edges are placed at random, and allowing multiple edges between the same pair of vertices. Let $|E| = m$ be the total number of edges in the graph. The probability that one end of an edge is v_r is given as $\frac{d_r}{2m}$, where d_r is the degree of v_r. The probability that one end is v_r and the other v_s is then given as

$$\frac{1}{2} p_{rs} = \frac{d_r}{2m} \cdot \frac{d_s}{2m} = \frac{d_r d_s}{4m^2}$$

The number of edges between v_r and v_s follows a binomial distribution with success probability p_{rs} over $2m$ trials (because we are selecting the two ends of m edges). The expected number of edges between v_r and v_s is given as

$$2m \cdot p_{rs} = \frac{d_r d_s}{2m}$$

The expected number of edges within a cluster C_i is then

$$\frac{1}{2}\sum_{v_r \in C_i}\sum_{v_s \in C_i}\frac{d_r d_s}{2m}$$

and the expected number of edges within the same cluster, summed over all k clusters, is given as

$$\frac{1}{2}\sum_{i=1}^{k}\sum_{v_r \in C_i}\sum_{v_s \in C_i}\frac{d_r d_s}{2m} \tag{16.24}$$

where we divide by 2 because each edge is counted twice. The *modularity* of the clustering $\mathcal{C}$ is defined as the difference between the observed and expected fraction of edges within the same cluster, obtained by subtracting Eq. (16.24) from Eq. (16.23), and dividing by the number of edges:

$$Q = \frac{1}{2m}\sum_{i=1}^{k}\sum_{v_r \in C_i}\sum_{v_s \in C_i}\left(a_{rs} - \frac{d_r d_s}{2m}\right)$$

Because $2m = \sum_{i=1}^{n} d_i$, we can rewrite modularity as follows:

$$Q = \sum_{i=1}^{k}\sum_{v_r \in C_i}\sum_{v_s \in C_i}\left(\frac{a_{rs}}{\sum_{j=1}^{n} d_j} - \frac{d_r d_s}{\left(\sum_{j=1}^{n} d_j\right)^2}\right) \tag{16.25}$$

Weighted Graphs One advantage of the modularity formulation in Eq. (16.25) is that it directly generalizes to weighted graphs. Assume that $\mathbf{A}$ is the weighted adjacency matrix; we interpret the modularity of a clustering as the difference between the observed and expected fraction of weights on edges within the clusters.

From Eq. (16.11) we have

$$\sum_{v_r \in C_i}\sum_{v_s \in C_i} a_{rs} = W(C_i, C_i)$$

and from Eq. (16.10) we have

$$\sum_{v_r \in C_i}\sum_{v_s \in C_i} d_r d_s = \left(\sum_{v_r \in C_i} d_r\right)\left(\sum_{v_s \in C_i} d_s\right) = W(C_i, V)^2$$

Further, note that

$$\sum_{j=1}^{n} d_j = W(V, V)$$

Using the above equivalences, can write the modularity objective [Eq. (16.25)] in terms of the weight function W as follows:

$$\max_{\mathcal{C}} J_Q(\mathcal{C}) = \sum_{i=1}^{k}\left(\frac{W(C_i, C_i)}{W(V, V)} - \left(\frac{W(C_i, V)}{W(V, V)}\right)^2\right) \tag{16.26}$$

We now express the modularity objective [Eq. (16.26)] in matrix terms. From Eq. (16.11), we have

$$W(C_i, C_i) = \mathbf{c}_i^T \mathbf{A} \mathbf{c}_i$$

Also note that

$$W(C_i, V) = \sum_{v_r \in C_i} d_r = \sum_{v_r \in C_i} d_r c_{ir} = \sum_{j=1}^{n} d_j c_{ij} = \sum_{j=1}^{n} \mathbf{d}^T \mathbf{c}_i$$

where $\mathbf{d} = (d_1, d_2, \ldots, d_n)^T$ is the vector of vertex degrees. Further, we have

$$W(V, V) = \sum_{j=1}^{n} d_j = tr(\mathbf{\Delta})$$

where $tr(\mathbf{\Delta})$ is the trace of $\mathbf{\Delta}$, that is, sum of the diagonal entries of $\mathbf{\Delta}$.

The clustering objective based on modularity can then be written as

$$\begin{aligned}
\max_{\mathcal{C}} J_Q(\mathcal{C}) &= \sum_{i=1}^{k} \left(\frac{\mathbf{c}_i^T \mathbf{A} \mathbf{c}_i}{tr(\mathbf{\Delta})} - \frac{(\mathbf{d}_i^T \mathbf{c}_i)^2}{tr(\mathbf{\Delta})^2} \right) \\
&= \sum_{i=1}^{k} \left(\mathbf{c}_i^T \left(\frac{\mathbf{A}}{tr(\mathbf{\Delta})} \right) \mathbf{c}_i - \mathbf{c}_i^T \left(\frac{\mathbf{d} \cdot \mathbf{d}^T}{tr(\mathbf{\Delta})^2} \right) \mathbf{c}_i \right) \\
&= \sum_{i=1}^{k} \mathbf{c}_i^T \mathbf{Q} \mathbf{c}_i
\end{aligned} \tag{16.27}$$

where $\mathbf{Q}$ is the *modularity matrix*:

$$\mathbf{Q} = \frac{1}{tr(\mathbf{\Delta})} \left(\mathbf{A} - \frac{\mathbf{d} \cdot \mathbf{d}^T}{tr(\mathbf{\Delta})} \right)$$

Directly maximizing objective Eq. (16.27) for binary cluster vectors $\mathbf{c}_i$ is hard. We resort to the approximation that elements of $\mathbf{c}_i$ can take on real values. Further, we require that $\mathbf{c}_i^T \mathbf{c}_i = \|\mathbf{c}_i\|^2 = 1$ to ensure that J_Q does not increase without bound. Following the approach in Eq. (16.15), we conclude that $\mathbf{c}_i$ is an eigenvector of $\mathbf{Q}$. However, because this a maximization problem, instead of selecting the k smallest eigenvalues, we select the k largest eigenvalues and the corresponding eigenvectors to obtain

$$\begin{aligned}
\max_{\mathcal{C}} J_Q(\mathcal{C}) &= \mathbf{u}_1^T \mathbf{Q} \mathbf{u}_1 + \cdots + \mathbf{u}_k^T \mathbf{Q} \mathbf{u}_k \\
&= \lambda_1 + \cdots + \lambda_k
\end{aligned}$$

where $\mathbf{u}_i$ is the eigenvector corresponding to λ_i, and the eigenvalues are sorted so that $\lambda_1 \geq \cdots \geq \lambda_n$. The relaxed cluster indicator vectors are given as $\mathbf{c}_i = \mathbf{u}_i$. Note that the modularity matrix $\mathbf{Q}$ is symmetric, but it is not positive semidefinite. This means that although it has real eigenvalues, they may be negative too. Also note that if Q_i denotes the ith column of $\mathbf{Q}$, then we have $Q_1 + Q_2 + \cdots + Q_n = \mathbf{0}$, which implies that 0 is an eigenvalue of $\mathbf{Q}$ with the corresponding eigenvector $\frac{1}{\sqrt{n}}\mathbf{1}$. Thus, for maximizing the modularity one should use only the positive eigenvalues.

Example 16.10. Consider the graph in Figure 16.2. The degree vector is $\mathbf{d} = (3,3,3,4,3,3,3)^T$, and the sum of degrees is $tr(\mathbf{\Delta}) = 22$. The modularity matrix is given as

$$\begin{aligned}
\mathbf{Q} &= \frac{1}{tr(\mathbf{\Delta})}\mathbf{A} - \frac{1}{tr(\mathbf{\Delta})^2}\mathbf{d}\cdot\mathbf{d}^T \\
&= \frac{1}{22}\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} - \frac{1}{484}\begin{pmatrix} 9 & 9 & 9 & 12 & 9 & 9 & 9 \\ 9 & 9 & 9 & 12 & 9 & 9 & 9 \\ 9 & 9 & 9 & 12 & 9 & 9 & 9 \\ 12 & 12 & 12 & 16 & 12 & 12 & 12 \\ 9 & 9 & 9 & 12 & 9 & 9 & 9 \\ 9 & 9 & 9 & 12 & 9 & 9 & 9 \\ 9 & 9 & 9 & 12 & 9 & 9 & 9 \end{pmatrix} \\
&= \begin{pmatrix} -0.019 & 0.027 & -0.019 & 0.021 & -0.019 & 0.027 & -0.019 \\ 0.027 & -0.019 & 0.027 & 0.021 & -0.019 & -0.019 & -0.019 \\ -0.019 & 0.027 & -0.019 & 0.021 & -0.019 & -0.019 & 0.027 \\ 0.021 & 0.021 & 0.021 & -0.033 & 0.021 & -0.025 & -0.025 \\ -0.019 & -0.019 & -0.019 & 0.021 & -0.019 & 0.027 & 0.027 \\ 0.027 & -0.019 & -0.019 & -0.025 & 0.027 & -0.019 & 0.027 \\ -0.019 & -0.019 & 0.027 & -0.025 & 0.027 & 0.027 & -0.019 \end{pmatrix}
\end{aligned}$$

The eigenvalues of $\mathbf{Q}$ are as follows:

$$\lambda_1 = 0.0678 \qquad \lambda_2 = 0.0281 \qquad \lambda_3 = 0 \qquad \lambda_4 = -0.0068$$
$$\lambda_5 = -0.0579 \qquad \lambda_6 = -0.0736 \qquad \lambda_7 = -0.1024$$

The eigenvector corresponding to $\lambda_3 = 0$ is

$$\mathbf{u}_3 = \frac{1}{\sqrt{7}}(1,1,1,1,1,1,1)^T = (0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38)^T$$

Modularity as Average Weight Consider what happens to the modularity matrix $\mathbf{Q}$ if we use the normalized adjacency matrix $\mathbf{M} = \mathbf{\Delta}^{-1}\mathbf{A}$ in place of the standard adjacency matrix $\mathbf{A}$ in Eq. (16.27). In this case, we know by Eq. (16.3) that each row of $\mathbf{M}$ sums to 1, that is,

$$\sum_{j=1}^{n} m_{ij} = d_i = 1, \text{ for all } i = 1, \ldots, n$$

We thus have $tr(\mathbf{\Delta}) = \sum_{i=1}^{n} d_i = n$, and further $\mathbf{d}\cdot\mathbf{d}^T = \mathbf{1}_{n\times n}$, where $\mathbf{1}_{n\times n}$ is the $n \times n$ matrix of all 1's. The modularity matrix can then be written as

$$\mathbf{Q} = \frac{1}{n}\mathbf{M} - \frac{1}{n^2}\mathbf{1}_{n\times n}$$

For large graphs with many nodes, n is large and the second term practically vanishes, as $\frac{1}{n^2}$ will be very small. Thus, the modularity matrix can be reasonably

approximated as

$$\mathbf{Q} \simeq \frac{1}{n}\mathbf{M} \tag{16.28}$$

Substituting the above in the modularity objective [Eq. (16.27)], we get

$$\max_{\mathcal{C}} J_Q(\mathcal{C}) = \sum_{i=1}^{k} \mathbf{c}_i^T \mathbf{Q} \mathbf{c}_i = \sum_{i=1}^{k} \mathbf{c}_i^T \mathbf{M} \mathbf{c}_i \tag{16.29}$$

where we dropped the $\frac{1}{n}$ factor because it is a constant for a given graph; it only scales the eigenvalues without effecting the eigenvectors.

In conclusion, if we use the normalized adjacency matrix, maximizing the modularity is equivalent to selecting the k largest eigenvalues and the corresponding eigenvectors of the normalized adjacency matrix $\mathbf{M}$. Note that in this case modularity is also equivalent to the average weight objective and kernel K-means as established in Eq. (16.22).

Normalized Modularity as Normalized Cut Define the *normalized modularity* objective as follows:

$$\max_{\mathcal{C}} J_{nQ}(\mathcal{C}) = \sum_{i=1}^{k} \frac{1}{W(C_i, V)} \left(\frac{W(C_i, C_i)}{W(V, V)} - \left(\frac{W(C_i, V)}{W(V, V)} \right)^2 \right) \tag{16.30}$$

We can observe that the main difference from the modularity objective [Eq. (16.26)] is that we divide by $vol(C_i) = W(C, V_i)$ for each cluster. Simplifying the above, we obtain

$$\begin{aligned} J_{nQ}(\mathcal{C}) &= \frac{1}{W(V, V)} \sum_{i=1}^{k} \left(\frac{W(C_i, C_i)}{W(C_i, V)} - \frac{W(C_i, V)}{W(V, V)} \right) \\ &= \frac{1}{W(V, V)} \left(\sum_{i=1}^{k} \left(\frac{W(C_i, C_i)}{W(C_i, V)} \right) - \sum_{i=1}^{k} \left(\frac{W(C_i, V)}{W(V, V)} \right) \right) \\ &= \frac{1}{W(V, V)} \left(\sum_{i=1}^{k} \left(\frac{W(C_i, C_i)}{W(C_i, V)} \right) - 1 \right) \end{aligned}$$

Now consider the expression $(k-1) - W(V, V) \cdot J_{nQ}(\mathcal{C})$, we have

$$\begin{aligned} (k-1) - W(V, V) J_{nQ}(\mathcal{C}) &= (k-1) - \left(\sum_{i=1}^{k} \left(\frac{W(C_i, C_i)}{W(C_i, V)} \right) - 1 \right) \\ &= k - \sum_{i=1}^{k} \frac{W(C_i, C_i)}{W(C_i, V)} \\ &= \sum_{i=1}^{k} 1 - \frac{W(C_i, C_i)}{W(C_i, V)} \\ &= \sum_{i=1}^{k} \frac{W(C_i, V) - W(C_i, C_i)}{W(C_i, V)} \end{aligned}$$

$$= \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{W(C_i, V)}$$

$$= \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{vol(C_i)}$$

$$= J_{nc}(\mathcal{C})$$

In other words the normalized cut objective [Eq. (16.17)] is related to the normalized modularity objective [Eq. (16.30)] by the following equation:

$$J_{nc}(\mathcal{C}) = (k-1) - W(V,V) \cdot J_{nQ}(\mathcal{C})$$

Since $W(V, V)$ is a constant for a given graph, we observe that minimizing normalized cut is equivalent to maximizing normalized modularity.

Spectral Clustering Algorithm

Both average weight and modularity are maximization objectives; therefore we have to slightly modify Algorithm 16.1 for spectral clustering to use these objectives. The matrix $\mathbf{B}$ is chosen to be $\mathbf{A}$ if we are maximizing average weight or $\mathbf{Q}$ for the modularity objective. Next, instead of computing the k smallest eigenvalues we have to select the k largest eigenvalues and their corresponding eigenvectors. Because both $\mathbf{A}$ and $\mathbf{Q}$ can have negative eigenvalues, we must select only the positive eigenvalues. The rest of the algorithm remains the same.

16.3 MARKOV CLUSTERING

We now consider a graph clustering method based on simulating a random walk on a weighted graph. The basic intuition is that if node transitions reflect the weights on the edges, then transitions from one node to another within a cluster are much more likely than transitions between nodes from different clusters. This is because nodes within a cluster have higher similarities or weights, and nodes across clusters have lower similarities.

Given the weighted adjacency matrix $\mathbf{A}$ for a graph G, the normalized adjacency matrix [Eq. (16.2)] is given as $\mathbf{M} = \mathbf{\Delta}^{-1}\mathbf{A}$. The matrix $\mathbf{M}$ can be interpreted as the $n \times n$ *transition matrix* where the entry $m_{ij} = \frac{a_{ij}}{d_i}$ can be interpreted as the probability of transitioning or jumping from node i to node j in the graph G. This is because $\mathbf{M}$ is a *row stochastic* or *Markov* matrix, which satisfies the following conditions: (1) elements of the matrix are non-negative, that is, $m_{ij} \geq 0$, which follows from the fact that $\mathbf{A}$ is non-negative, and (2) rows of $\mathbf{M}$ are probability vectors, that is, row elements add to 1, because

$$\sum_{j=1}^{n} m_{ij} = \sum_{j=1}^{n} \frac{a_{ij}}{d_i} = 1$$

The matrix $\mathbf{M}$ is thus the transition matrix for a *Markov chain* or a Markov random walk on graph G. A Markov chain is a discrete-time stochastic process over a set of

states, in our case the set of vertices V. The Markov chain makes a transition from one node to another at discrete timesteps $t = 1, 2, \ldots$, with the probability of making a transition from node i to node j given as m_{ij}. Let the random variable X_t denote the state at time t. The Markov property means that the probability distribution of X_t over the states at time t depends only on the probability distribution of X_{t-1}, that is,

$$P(X_t = i | X_0, X_1, \ldots, X_{t-1}) = P(X_t = i | X_{t-1})$$

Further, we assume that the Markov chain is *homogeneous*, that is, the transition probability

$$P(X_t = j | X_{t-1} = i) = m_{ij}$$

is independent of the time step t.

Given node i the transition matrix $\mathbf{M}$ specifies the probabilities of reaching any other node j in one time step. Starting from node i at $t = 0$, let us consider the probability of being at node j at $t = 2$, that is, after two steps. We denote by $m_{ij}(2)$ the probability of reaching j from i in two time steps. We can compute this as follows:

$$\begin{aligned} m_{ij}(2) &= P(X_2 = j | X_0 = i) = \sum_{a=1}^{n} P(X_1 = a | X_0 = i) P(X_2 = j | X_1 = a) \\ &= \sum_{a=1}^{n} m_{ia} m_{aj} = \mathbf{m}_i^T M_j \end{aligned} \tag{16.31}$$

where $\mathbf{m}_i = (m_{i1}, m_{i2}, \ldots, m_{in})^T$ denotes the vector corresponding to the ith row of $\mathbf{M}$ and $M_j = (m_{1j}, m_{2j}, \ldots, m_{nj})^T$ denotes the vector corresponding to the jth column of $\mathbf{M}$.

Consider the product of $\mathbf{M}$ with itself:

$$\begin{aligned} \mathbf{M}^2 = \mathbf{M} \cdot \mathbf{M} &= \begin{pmatrix} — \mathbf{m}_1^T — \\ — \mathbf{m}_2^T — \\ \vdots \\ — \mathbf{m}_n^T — \end{pmatrix} \begin{pmatrix} | & | & & | \\ M_1 & M_2 & \cdots & M_n \\ | & | & & | \end{pmatrix} \\ &= \left\{ \mathbf{m}_i^T M_j \right\}_{i,j=1}^{n} = \left\{ m_{ij}(2) \right\}_{i,j=1}^{n} \end{aligned} \tag{16.32}$$

Equations (16.31) and (16.32) imply that $\mathbf{M}^2$ is precisely the transition probability matrix for the Markov chain over two time-steps. Likewise, the three-step transition matrix is $\mathbf{M}^2 \cdot \mathbf{M} = \mathbf{M}^3$. In general, the transition probability matrix for t time steps is given as

$$\mathbf{M}^{t-1} \cdot \mathbf{M} = \mathbf{M}^t \tag{16.33}$$

A random walk on G thus corresponds to taking successive powers of the transition matrix $\mathbf{M}$. Let $\boldsymbol{\pi}_0$ specify the initial state probability vector at time $t = 0$, that is, $\boldsymbol{\pi}_{0i} = P(X_0 = i)$ is the probability of starting at node i, for all $i = 1, \ldots, n$. Starting

from $\boldsymbol{\pi}_0$, we can obtain the state probability vector for X_t, that is, the probability of being at node i at time-step t, as follows

$$\begin{aligned}\boldsymbol{\pi}_t^T &= \boldsymbol{\pi}_{t-1}^T \mathbf{M} \\ &= \left(\boldsymbol{\pi}_{t-2}^T \mathbf{M}\right) \cdot \mathbf{M} = \boldsymbol{\pi}_{t-2}^T \mathbf{M}^2 \\ &= \left(\boldsymbol{\pi}_{t-3}^T \mathbf{M}^2\right) \cdot \mathbf{M} = \boldsymbol{\pi}_{t-3}^T \mathbf{M}^3 \\ &= \vdots \\ &= \boldsymbol{\pi}_0^T \mathbf{M}^t\end{aligned}$$

Equivalently, taking transpose on both sides, we get

$$\boldsymbol{\pi}_t = (M^t)^T \boldsymbol{\pi}_0 = (\mathbf{M}^T)^t \boldsymbol{\pi}_0$$

The state probability vector thus converges to the dominant eigenvector of $\mathbf{M}^T$, reflecting the steady-state probability of reaching any node in the graph, regardless of the starting node. Note that if the graph is directed, then the steady-state vector is equivalent to the normalized prestige vector [Eq. (4.6)].

Transition Probability Inflation

We now consider a variation of the random walk, where the probability of transitioning from node i to j is inflated by taking each element m_{ij} to the power $r \geq 1$. Given a transition matrix $\mathbf{M}$, define the inflation operator Υ as follows:

$$\Upsilon(\mathbf{M}, r) = \left\{ \frac{(m_{ij})^r}{\sum_{a=1}^{n} (m_{ia})^r} \right\}_{i,j=1}^{n} \tag{16.34}$$

The inflation operation results in a transformed or inflated transition probability matrix because the elements remain non-negative, and each row is normalized to sum to 1. The net effect of the inflation operator is to increase the higher probability transitions and decrease the lower probability transitions.

16.3.1 Markov Clustering Algorithm

The Markov clustering algorithm (MCL) is an iterative method that interleaves matrix expansion and inflation steps. Matrix expansion corresponds to taking successive powers of the transition matrix, leading to random walks of longer lengths. On the other hand, matrix inflation makes the higher probability transitions even more likely and reduces the lower probability transitions. Because nodes in the same cluster are expected to have higher weights, and consequently higher transition probabilities between them, the inflation operator makes it more likely to stay within the cluster. It thus limits the extent of the random walk.

The pseudo-code for MCL is given in Algorithm 16.2. The method works on the weighted adjacency matrix for a graph. Instead of relying on a user-specified value for k, the number of output clusters, MCL takes as input the inflation parameter $r \geq 1$. Higher values lead to more, smaller clusters, whereas smaller values lead to fewer, but larger clusters. However, the exact number of clusters cannot be pre-determined. Given the adjacency matrix $\mathbf{A}$, MCL first adds *loops* or self-edges to $\mathbf{A}$ if they do

ALGORITHM 16.2. Markov Clustering Algorithm (MCL)

```
MARKOV CLUSTERING (A, r, ε):
t ← 0
Add self-edges to A if they do not exist
M_t ← Δ^{-1}A
repeat
    t ← t + 1
    M_t ← M_{t-1} · M_{t-1}
    M_t ← Υ(M_t, r)
until ‖M_t − M_{t-1}‖_F ≤ ε
G_t ← directed graph induced by M_t
C ← {weakly connected components in G_t}
```

not exist. If $\mathbf{A}$ is a similarity matrix, then this is not required, as a node is most similar to itself, and thus $\mathbf{A}$ should have high values on the diagonals. For simple, undirected graphs, if $\mathbf{A}$ is the adjacency matrix, then adding self-edges associates return probabilities with each node.

The iterative MCL expansion and inflation process stops when the transition matrix converges, that is, when the difference between the transition matrix from two successive iterations falls below some threshold $\epsilon \geq 0$. The matrix difference is given in terms of the *Frobenius norm*:

$$\|\mathbf{M}_t - \mathbf{M}_{t-1}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\Big(\mathbf{M}_t(i,j) - \mathbf{M}_{t-1}(i,j)\Big)^2}$$

The MCL process stops when $\|\mathbf{M}_t - \mathbf{M}_{t-1}\|_F \leq \epsilon$.

MCL Graph

The final clusters are found by enumerating the weakly connected components in the directed graph induced by the converged transition matrix $\mathbf{M}_t$. The directed graph induced by $\mathbf{M}_t$ is denoted as $G_t = (V_t, E_t)$. The vertex set is the same as the set of nodes in the original graph, that is, $V_t = V$, and the edge set is given as

$$E_t = \big\{(i,j) \mid \mathbf{M}_t(i,j) > 0\big\}$$

In other words, a directed edge (i,j) exists only if node i can transition to node j within t steps of the expansion and inflation process. A node j is called an *attractor* if $\mathbf{M}_t(j,j) > 0$, and we say that node i is attracted to attractor j if $\mathbf{M}_t(i,j) > 0$. The MCL process yields a set of attractor nodes, $V_a \subseteq V$, such that other nodes are attracted to at least one attractor in V_a. That is, for all nodes i there exists a node $j \in V_a$, such that $(i,j) \in E_t$. A strongly connected component in a directed graph

is defined a maximal subgraph such that there exists a directed path between all pairs of vertices in the subgraph. To extract the clusters from G_t, MCL first finds the strongly connected components $S_1, S_2, \ldots, S_q$ over the set of attractors V_a. Next, for each strongly connected set of attractors S_j, MCL finds the weakly connected components consisting of all nodes $i \in V_t - V_a$ attracted to an attractor in S_j. If a node i is attracted to multiple strongly connected components, it is added to each such cluster, resulting in possibly overlapping clusters.

Example 16.11. We apply the MCL method to find $k = 2$ clusters for the graph shown in Figure 16.2. We add the self-loops to the graph to obtain the adjacency matrix:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

The corresponding Markov matrix is given as

$$\mathbf{M}_0 = \boldsymbol{\Delta}^{-1}\mathbf{A} = \begin{pmatrix} 0.25 & 0.25 & 0 & 0.25 & 0 & 0.25 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0.25 & 0.25 & 0 & 0 & 0.25 \\ 0.20 & 0.20 & 0.20 & 0.20 & 0.20 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0 & 0 & 0 & 0.25 & 0.25 & 0.25 \\ 0 & 0 & 0.25 & 0 & 0.25 & 0.25 & 0.25 \end{pmatrix}$$

In the first iteration, we apply expansion and then inflation (with $r = 2.5$) to obtain

$$\mathbf{M}_1 = \mathbf{M}_0 \cdot \mathbf{M}_0 = \begin{pmatrix} 0.237 & 0.175 & 0.113 & 0.175 & 0.113 & 0.125 & 0.062 \\ 0.175 & 0.237 & 0.175 & 0.237 & 0.050 & 0.062 & 0.062 \\ 0.113 & 0.175 & 0.237 & 0.175 & 0.113 & 0.062 & 0.125 \\ 0.140 & 0.190 & 0.140 & 0.240 & 0.090 & 0.100 & 0.100 \\ 0.113 & 0.050 & 0.113 & 0.113 & 0.237 & 0.188 & 0.188 \\ 0.125 & 0.062 & 0.062 & 0.125 & 0.188 & 0.250 & 0.188 \\ 0.062 & 0.062 & 0.125 & 0.125 & 0.188 & 0.188 & 0.250 \end{pmatrix}$$

$$\mathbf{M}_1 = \Upsilon(\mathbf{M}_1, 2.5) = \begin{pmatrix} 0.404 & 0.188 & 0.062 & 0.188 & 0.062 & 0.081 & 0.014 \\ 0.154 & 0.331 & 0.154 & 0.331 & 0.007 & 0.012 & 0.012 \\ 0.062 & 0.188 & 0.404 & 0.188 & 0.062 & 0.014 & 0.081 \\ 0.109 & 0.234 & 0.109 & 0.419 & 0.036 & 0.047 & 0.047 \\ 0.060 & 0.008 & 0.060 & 0.060 & 0.386 & 0.214 & 0.214 \\ 0.074 & 0.013 & 0.013 & 0.074 & 0.204 & 0.418 & 0.204 \\ 0.013 & 0.013 & 0.074 & 0.074 & 0.204 & 0.204 & 0.418 \end{pmatrix}$$

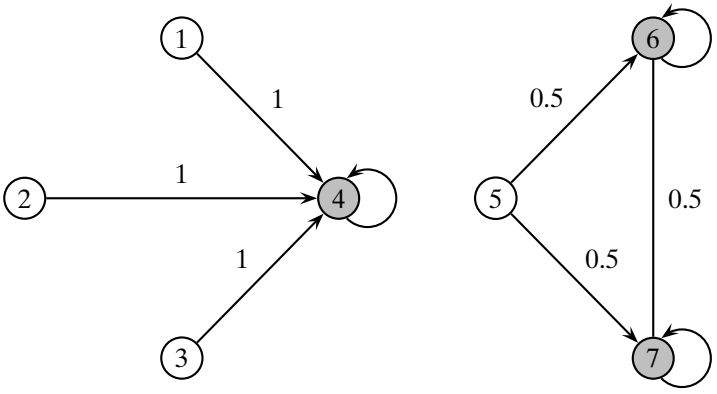


Figure 16.5. MCL attractors and clusters.

MCL converges in 10 iterations (using $\epsilon = 0.001$), with the final transition matrix

$$\mathbf{M} = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \end{array}$$

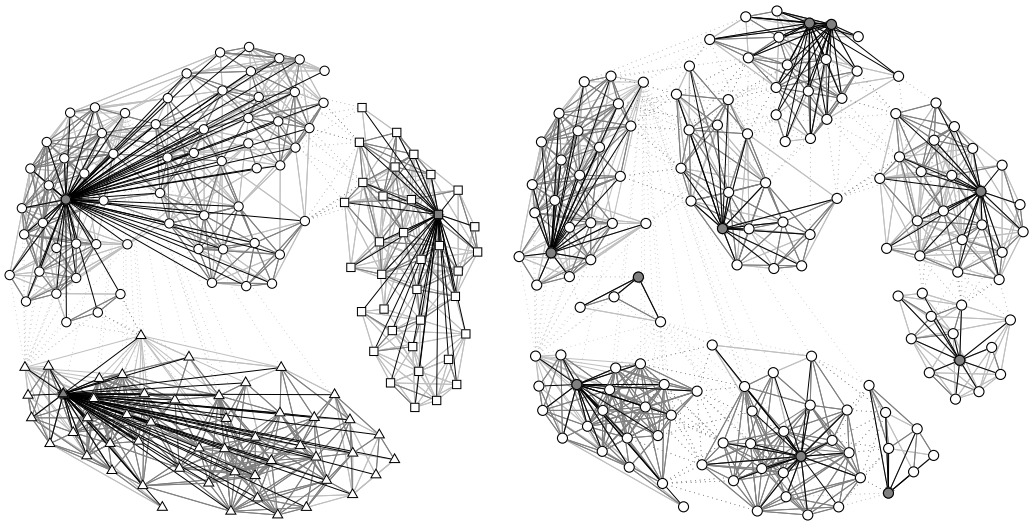
Figure 16.5 shows the directed graph induced by the converged $\mathbf{M}$ matrix, where an edge (i, j) exists if and only if $\mathbf{M}(i, j) > 0$. The nonzero diagonal elements of $\mathbf{M}$ are the attractors (nodes with self-loops, shown in gray). We can observe that $\mathbf{M}(4, 4)$, $\mathbf{M}(6, 6)$, and $\mathbf{M}(7, 7)$ are all greater than zero, making nodes 4, 6, and 7 the three attractors. Because both 6 and 7 can reach each other, the equivalence classes of attractors are $\{4\}$ and $\{6, 7\}$. Nodes $1, 2$, and 3 are attracted to 4, and node 5 is attracted to both 6 and 7. Thus, the two weakly connected components that make up the two clusters are $C_1 = \{1, 2, 3, 4\}$ and $C_2 = \{5, 6, 7\}$.

Example 16.12. Figure 16.6a shows the clusters obtained via the MCL algorithm on the Iris graph from Figure 16.1, using $r = 1.3$ in the inflation step. MCL yields three attractors (shown as gray nodes; self-loops omitted), which separate the graph into three clusters. The contingency table for the discovered clusters versus the true Iris types is given in Table 16.2. One point with class `iris-versicolor` is (wrongly) grouped with `iris-setosa` in C_1, but 14 points from `iris-virginica` are misclustered.

Notice that the only parameter for MCL is r, the exponent for the inflation step. The number of clusters is not explicitly specified, but higher values of r result in more clusters. The value of $r = 1.3$ was used above because it resulted in three clusters. Figure 16.6b shows the results for $r = 2$. MCL yields nine clusters, where one of the clusters (top-most) has two attractors.

Table 16.2. Contingency table: MCL clusters versus Iris types

	iris-setosa	iris-virginica	iris-versicolor
C_1 (triangle)	50	0	1
C_2 (square)	0	36	0
C_3 (circle)	0	14	49



(a) $r = 1.3$ (b) $r = 2$

Figure 16.6. MCL on Iris graph.

Computational Complexity

The computational complexity of the MCL algorithm is $O(tn^3)$, where t is the number of iterations until convergence. This follows from the fact that whereas the inflation operation takes $O(n^2)$ time, the expansion operation requires matrix multiplication, which takes $O(n^3)$ time. However, the matrices become sparse very quickly, and it is possible to use sparse matrix multiplication to obtain $O(n^2)$ complexity for expansion in later iterations. On convergence, the weakly connected components in G_t can be found in $O(n + m)$ time, where m is the number of edges. Because G_t is very sparse, with $m = O(n)$, the final clustering step takes $O(n)$ time.

16.4 FURTHER READING

Spectral partitioning of graphs was first proposed in Donath and Hoffman (1973). Properties of the second smallest eigenvalue of the Laplacian matrix, also called *algebraic connectivity*, were studied in Fiedler (1973). A recursive bipartitioning approach to find k clusters using the normalized cut objective was given in Shi and Malik (2000). The direct k-way partitioning approach for normalized cut, using the normalized symmetric Laplacian matrix, was proposed in Ng, Jordan, and Weiss (2001). The connection between spectral clustering objective and kernel K-means was established

in Dhillon, Guan, and Kulis (2007). The modularity objective was introduced in Newman (2003), where it was called *assortativity coefficient*. The spectral algorithm using the modularity matrix was first proposed in Smyth and White (2005). The relationship between modularity and normalized cut was shown in Yu and Ding (2010). For an excellent tutorial on spectral clustering techniques see Luxburg (2007). The Markov clustering algorithm was originally proposed in van Dongen (2000). For an extensive review of graph clustering methods see Fortunato (2010).

Dhillon, I. S., Guan, Y., and Kulis, B. (2007). "Weighted graph cuts without eigenvectors: A multilevel approach." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29 (11): 1944–1957.

Donath, W. E. and Hoffman, A. J. (September 1973)."Lower bounds for the partitioning of graphs." *IBM Journal of Research and Development*, 17 (5): 420–425.

Fiedler, M. (1973). "Algebraic connectivity of graphs." *Czechoslovak Mathematical Journal*, 23 (2): 298–305.

Fortunato, S. (2010). "Community detection in graphs." *Physics Reports*, 486 (3): 75–174.

Luxburg, U. (December 2007). "A tutorial on spectral clustering." *Statistics and Computing*, 17 (4): 395–416.

Newman, M. E. (2003). "Mixing patterns in networks." *Physical Review E*, 67 (2): 026126.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). "On spectral clustering: Analysis and an algorithm." *Advances in Neural Information Processing Systems 14* (pp. 849–856). Cambridge, MA: MIT Press.

Shi, J. and Malik, J. (August 2000). "Normalized cuts and image segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (8): 888–905.

Smyth, S. and White, S. (2005). "A spectral clustering approach to finding communities in graphs." *In Proceedings of the 5th SIAM International Conference on Data Mining,* vol. 119, p. 274.

van Dongen, S. M. (2000). "Graph clustering by flow simulation." PhD thesis. The University of Utrecht, The Netherlands.

Yu, L. and Ding, C. (2010). "Network community discovery: solving modularity clustering via normalized cut." *In Proceedings of the 8th Workshop on Mining and Learning with Graphs*. ACM pp. 34–36.

16.5 EXERCISES

Q1. Show that if Q_i denotes the ith column of the modularity matrix $\mathbf{Q}$, then $\sum_{i=1}^{n} Q_i = \mathbf{0}$.

Q2. Prove that both the normalized symmetric and asymmetric Laplacian matrices $\mathbf{L}^s$ [Eq. (16.6)] and $\mathbf{L}^a$ [Eq. (16.9)] are positive semidefinite. Also show that the smallest eigenvalue is $\lambda_n = 0$ for both.

Q3. Prove that the largest eigenvalue of the normalized adjacency matrix $\mathbf{M}$ [Eq. (16.2)] is 1, and further that all eigenvalues satisfy the condition that $|\lambda_i| \leq 1$.

Q4. Show that $\sum_{v_r \in C_i} c_{ir} d_r c_{ir} = \sum_{r=1}^{n} \sum_{s=1}^{n} c_{ir} \mathbf{\Delta}_{rs} c_{is}$, where $\mathbf{c}_i$ is the cluster indicator vector for cluster C_i and $\mathbf{\Delta}$ is the degree matrix for the graph.

Q5. For the normalized symmetric Laplacian $\mathbf{L}^s$, show that for the normalized cut objective the real-valued cluster indicator vector corresponding to the smallest eigenvalue $\lambda_n = 0$ is given as $\mathbf{c}_n = \frac{1}{\sqrt{n}}\mathbf{1}$.

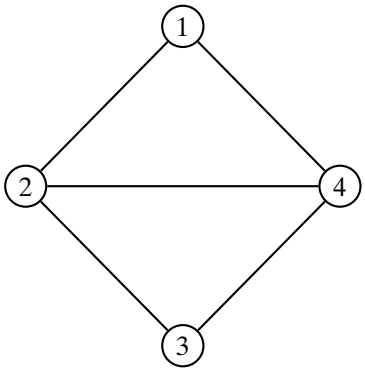


Figure 16.7. Graph for Q6.

Q6. Given the graph in Figure 16.7, answer the following questions:

(a) Cluster the graph into two clusters using ratio cut and normalized cut.

(b) Use the normalized adjacency matrix $\mathbf{M}$ for the graph and cluster it into two clusters using average weight and kernel K-means, using $\mathbf{K} = \mathbf{M}$.

(c) Cluster the graph using the MCL algorithm with inflation parameters $r = 2$ and $r = 2.5$.

Table 16.3. Data for Q7

	X_1	X_2	X_3
$\mathbf{x}_1$	0.4	0.9	0.6
$\mathbf{x}_2$	0.5	0.1	0.6
$\mathbf{x}_3$	0.6	0.3	0.6
$\mathbf{x}_4$	0.4	0.8	0.5

Q7. Consider Table 16.3. Assuming these are nodes in a graph, define the weighted adjacency matrix $\mathbf{A}$ using the linear kernel

$$\mathbf{A}(i, j) = 1 + \mathbf{x}_i^T \mathbf{x}_j$$

Cluster the data into two groups using the modularity objective.