Chapter 8

Spectral Clustering

Jialu Liu
University of Illinois at Urbana-Champaign
Urbana, IL
jliu64@illinois.edu

Jiawei Han
University of Illinois at Urbana-Champaign
Urbana, IL
hanj@illinois.edu

8.1 Introduction ................................................................. 177
8.2 Similarity Graph ......................................................... 179
8.3 Unnormalized Spectral Clustering ................................. 180
  8.3.1 Notation ................................................................. 180
  8.3.2 Unnormalized Graph Laplacian ............................... 180
  8.3.3 Spectrum Analysis ................................................... 181
  8.3.4 Unnormalized Spectral Clustering Algorithm ............... 182
8.4 Normalized Spectral Clustering ........................................ 182
  8.4.1 Normalized Graph Laplacian ..................................... 183
  8.4.2 Spectrum Analysis ................................................... 184
  8.4.3 Normalized Spectral Clustering Algorithm ................... 184
8.5 Graph Cut View ........................................................... 185
  8.5.1 Ratio Cut Relaxation ................................................ 186
  8.5.2 Normalized Cut Relaxation ....................................... 187
8.6 Random Walks View ..................................................... 188
8.7 Connection to Laplacian Eigenmap .................................. 189
8.8 Connection to Kernel $k$-Means and Nonnegative Matrix Factorization .............................................. 191
8.9 Large Scale Spectral Clustering ...................................... 192
8.10 Further Reading .......................................................... 193
Bibliography ..................................................................... 195

8.1 Introduction

In this chapter, we introduce the family of spectral clustering algorithms which have seen increasing popularity over the past few years. Starting with the seminal works in [37] and [43], a large number of papers has been published along this line of work. As opposed to “traditional clustering algorithms” such as $k$-means and generative mixture models which always result in clusters with convex geometric shape, spectral clustering can solve problems in much more complex scenarios, such as intertwined spirals, or other arbitrary nonlinear shapes, because it does not make assumptions on the shapes of clusters. Another disadvantage of the previous algorithms is related to the inherent challenges in the Expectation Maximization (EM) framework, which is often used to learn
a mixture model for clustering. This framework is essentially an iterative process of finding local minima, and therefore multiple restarts are required to find a good solution.

Many data sets can be notoriously difficult to cluster with traditional methods. Figure 8.1 demonstrates a number of toy data sets [60], which are difficult for traditional clustering algorithms. These data sets have been constructed in order to generate clusters of different shapes. On such datasets, algorithms which implicitly assume specific shapes of clusters cannot achieve good results. For example, the Euclidean distance metrics assume a convex shape to the underlying clusters. Obviously such assumptions can impact the quality of the clustering in arbitrary data sets. As will be evident, the spectral clustering method is able to handle such data sets effectively.

The history of spectral clustering can be traced back to [14, 15], in which it was suggested that the eigenvectors of the adjacency matrix could be used in order to determine the underlying partitions. The main difference between spectral clustering algorithms is whether they use normalized or unnormalized “graph Laplacian” [52] which will be introduced in the later sections. Different versions of spectral clustering have been successfully applied to image segmentation [43], text mining [11], speech processing [1], and general purpose methods for data analysis and clustering [37, 60, 13, 12]. An excellent review on the history of spectral clustering can be found in [46].

The spectral clustering family can be viewed as a three-step algorithm:

- The first step is to construct a similarity graph for all the data points.
- The data points are embedded in a space, in which the clusters are more “obvious,” with the use of the eigenvectors of the graph Laplacian.
- Finally, a classical clustering algorithm such as k-means is applied to partition the embedding.

The low-dimensional representation obtained in the second step is also referred to as “spectral embedding” and has applications beyond the clustering context, such as dimensionality reduction [2]. The word spectral is used to denote the fact that the clustering results are obtained by analyzing the spectrum of the graph Laplacian.

Figure 8.2 shows\(^1\) the application of the spectral clustering on an example data set. The figure in the middle is a K-nearest neighbor (KNN) graph built in term of the Euclidean distance. The clustering result is illustrated in the adjacent figure on the right.

\(^1\)The figure is generated by GraphDemo: http://www.ml.uni-saarland.de/GraphDemo/GraphDemo.html.
8.2 Similarity Graph

Let the set of data points, which we wish to partition into \( k \) subsets, be denoted by \( X = \{x_1, \ldots, x_n\} \) in \( \mathbb{R}^m \). In order to perform spectral clustering, we first need to represent this data in the form of an undirected “similarity graph” \( G = (V, E) \). Here each data point \( x_i \) is represented by a vertex \( v_i \), and \( E \) refers to the edges between vertices. Note that \( x_i \) is a vector which denotes the data point while \( v_i \) is a vertex without any attributes. Then we can use a nonnegative weighted \( n \) by \( n \) adjacency matrix (or affinity matrix) \( W \) to describe \( G \), where \( W = \{W_{ij}\}_{i,j=1,\ldots,n} \). Note that \( W_{ij} \) equals 0 when vertices \( v_i \) and \( v_j \) are not connected. Since spectral clustering algorithms aim at partitioning the vertices to let those in the same cluster have high similarity and those in different clusters have low similarity, it becomes critical to choose an effective method to construct such an adjacency matrix.

Recent studies in spectral graph theory [8] and manifold learning theory [50, 2] have demonstrated that the adjacency matrix should model the local geometric structure of the data points. This issue will be discussed in some more detail in Section 8.7. Based on this rule, we introduce three ways to construct \( W \) [2, 31]:

1. **\( K \)-nearest neighbor graphs**: The idea is that \( v_i \) is connected with \( v_j \) when \( v_j \) is among the \( K \)-nearest neighbors of \( v_i \), or \( v_i \) is among the \( K \)-nearest neighbors of \( v_j \). The distance is computed based on the original representation of the data points \( x_i \) and \( x_j \). Some examples include \( \ell_1 \)-norm, \( \ell_2 \)-norm, and the cosine distance. The resulting graph is usually called the \( K \)-nearest neighbor graph. The alternative is to connect \( v_i \) and \( v_j \) when they are mutually in the neighborhood of each other. This graph is referred to as the mutual \( K \)-nearest neighbor graph or symmetric \( K \)-nearest neighbor graph. In both cases, after adding the edges according to the neighborhood of each vertex, we can assign weights to the edges by the similarity of their endpoints or simply adopt a 0–1 weight.

2. **\( \epsilon \)-neighborhood graph**: In this kind of graph, vertices are connected only when the pairwise distance \( ||x_i - x_j||^2 \) is smaller than \( \epsilon \). However, this method often leads to graphs with disconnected components if \( \epsilon \) is not carefully chosen.

3. **Fully connected graph**: In this case, we connect all vertices with positive similarity. Since the adjacency matrix should model the local neighborhood, the selection of the similarity function itself becomes tricky. An example is the heat kernel function:

\[
W_{ij} = e^{-\frac{||x_i - x_j||^2}{\sigma^2}}
\]

where \( \sigma \) controls the width of the neighborhoods. It is suggested in [60] that \( \sigma \) could be tuned locally with respect to the pair of vertices. Besides the problem of choosing an appropriate
value of $\sigma$, the construction of this kind of graph also suffers from the efficiency problem. It is reported in [5] that the most commonly used approach for addressing the computational and memory challenge is to zero out some elements in the similarity matrix, or to sparsify the matrix. From this point of view, once the graph is fully connected, it will increase both the time and space complexity. In Section 8.9, we will carefully discuss this issue.

The selection of the specific method for constructing the similarity graph can sometimes be a complex and confusing problem. To the best of our knowledge, there are no theoretical studies regarding the best choice under different circumstances. Besides these three methods, some recent papers have been published to propose new graph construction techniques which will be covered in Section 8.10.

## 8.3 Unnormalized Spectral Clustering

Given the similarity graph $G$, the main step for spectral clustering is then to compute graph Laplacian matrices [8]. The actual construction of this very important intermediate representation is often not defined uniquely in the literature. Every author just calls “his” matrix the graph Laplacian [31]. To distinguish them, we will carefully study and discuss their properties in this and the next section. In this section, we first introduce the unnormalized graph Laplacian.

### 8.3.1 Notation

In the previous section, we defined the nonnegative weight $W_{ij}$ for each pair of vertices $v_i$ and $v_j$ in the undirected similarity graph $G$. Since $G$ is undirected, we have $W_{ij} = W_{ji}$.

For each vertex, its degree is then computed as the sum of the weights incident on it:

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

Correspondingly, the degree matrix $D$ is defined as the diagonal matrix satisfying $D_{ii} = d_i$.

For a subset $A$ of vertices $V$, its indicator vector is denoted by $1_A = (f_1, \ldots, f_n)^T$, where $f_i = 1$ if vertex $v_i$ belongs to $A$ and $f_i = 0$ otherwise.

### 8.3.2 Unnormalized Graph Laplacian

The unnormalized graph Laplacian $L$ is defined as follows:

$$L = D - W$$

We list several important properties of $L$ [33, 34, 31]:

**Proposition 8.3.1** The unnormalized graph Laplacian $L$ satisfies the following four properties:

1. For an arbitrary vector $f \in \mathbb{R}^n$, we have

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

2. $L$ is symmetric and positive semidefinite.
3. The smallest eigenvalue of $L$ is 0, with eigenvector $1$.

4. $L$ has $n$ nonnegative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

**Proof 1** We can verify the first property from the following equation:

\[
T Lf = T(D - W)f = \sum_{i=1}^{n} f_i^2 d_i - \sum_{i,j=1}^{n} f_i f_j W_{ij}
\]

\[
= \frac{1}{2} \left( \sum_{i=1}^{n} f_i^2 d_i + \sum_{j=1}^{n} f_j^2 d_j - 2 \sum_{i,j=1}^{n} f_i f_j W_{ij} \right)
\]

\[
= \frac{1}{2} \sum_{i,j=1}^{n} W_{ij}(f_i - f_j)^2
\]

Since $f^T Lf \geq 0$ and both $D$ and $W$ are symmetric, $L$ is symmetric and positive semidefinite. Property 3 can be proven directly by plugging the unit vector $1$ into Equation (8.1). Due to the fact that $L$ is a real symmetric matrix, its eigenvalues should be also real numbers. Then property 4 is proved based on property 3.

### 8.3.3 Spectrum Analysis

We first analyze the spectrum of $L$ in the context of the unnormalized algorithm. Assume that $G$ is an undirected similarity graph with $k$ connected components $A_1, A_2, \ldots, A_k$. The corresponding graph Laplacians for these subsets are represented as $L_1, L_2, \ldots, L_k$. Without loss of generality, $L$ can be represented as a block-diagonal structure:

\[
L = \begin{bmatrix}
L_1 & & \\
& L_2 & \\
& & \ddots \\
& & & L_k
\end{bmatrix}
\]

Since $L$ is a block diagonal matrix, its eigenvalues and eigenvectors are the union of eigenvalues and eigenvectors of its blocks (i.e., $L_1, L_2, \ldots, L_k$). This implies that the multiplicity of eigenvalue 0 of $L$ should be at least $k$. The corresponding eigenvectors can be represented as indicator vectors $1_{A_1}, 1_{A_2}, \ldots, 1_{A_k}$. As the weights $W_{ij}$ are nonnegative, it can also be seen from Equation 8.1 that the next eigenvalue is strictly bigger than 0 because this sum can vanish if expressions $W_{ij}(f_i - f_j)^2$ for any pairs of vertices $v_i$ and $v_j$ vanish. So we can claim that the multiplicity $k$ of the eigenvalue 0 of $L$ equals the number of connected components $A_1, A_2, \ldots, A_k$.

How about the eigenvectors? Since 0 is a repeated eigenvalue in $L$, any other $k$ orthogonal vectors spanning the same subspace as the eigenspace of the eigenvectors obtained above can also be the eigenvectors. Therefore, we have the following proposition [31].

**Proposition 8.3.2** The multiplicity $k$ of eigenvalue 0 of $L$ equals the number of connected components $A_1, A_2, \ldots, A_k$ in the graph. And the eigenspace of eigenvalue 0 is spanned by the indicator vectors $1_{A_1}, 1_{A_2}, \ldots, 1_{A_k}$ of those components.

This proposition provides theoretical foundation for the spectral clustering algorithm proposed for partitioning vertices into different subsets in the “ideal” case.
8.3.4 Unnormalized Spectral Clustering Algorithm

We will introduce the unnormalized spectral clustering algorithm based on unnormalized graph Laplacian $L$. Assume that $F \in \mathbb{R}^{n \times k}$ is a matrix containing the $k$ relevant orthonormal vectors $f_1, f_2, \ldots, f_k$. It is desired to determine these orthonormal vectors $f_1, f_2, \ldots, f_k$ with the following objective function:

$$
\min_F Tr(F^T L F) \quad \text{s.t.} \quad F^T F = I \quad (8.2)
$$

Here, $Tr(\cdot)$ denotes the trace of the matrix.

As we have discussed in the last subsection, these vectors are the top $k$ eigenvectors of $L$. As discussed earlier, these eigenvectors are useful for clustering the vertices. However, unlike non-negative matrix factorization methods, these eigenvectors cannot be directly used for inferring the cluster labels because of the following two reasons. First, only in the ideal case where no edges exist between different connected components can the eigenspace be spanned by the indicator vectors $\mathbb{1}_{A_1}, \mathbb{1}_{A_2}, \ldots, \mathbb{1}_{A_k}$. Second, the eigenvectors could be any orthogonal transformation of the indicator vectors, which are not necessarily indicative of the cluster labels. Fortunately, if we view these eigenvectors as the low-dimensional representations of the original data points $X$, then any off-the-shelf clustering algorithms such as $k$-means and mixture models can be used to partition the embedding.

The overall algorithm for spectral clustering is illustrated below.

### Unnormalized Spectral Clustering

1. Construct the similarity graph with one of the methods described in Section 8.2. Let $W$ be the adjacency matrix and $D$ be the degree matrix.
2. Compute the unnormalized graph Laplacian $L$ where $L = D - W$.
3. Determine $f_1, f_2, \ldots, f_k$, the top $k$ eigenvectors of $L$.
4. Construct the matrix $F \in \mathbb{R}^{n \times k}$ from $f_1, f_2, \ldots, f_k$.
5. Treat each row of $F$ as a vertex in $\mathbb{R}^k$, partition these vertices into $k$ clusters via any off-the-shelf method such as the $k$-means algorithm.

8.4 Normalized Spectral Clustering

In the literature, there are two versions of normalized graph Laplacian: $L_{\text{sym}}$ and $L_{\text{rm}}$. The former refers to a symmetric matrix, and the latter can be explained from a random walk perspective. Close relationships exist between these two alternatives of normalized spectral clustering. These relationships will be discussed carefully in this section.

---

2Here, by top, we refer to the eigenvectors with the smallest eigenvalues.
8.4.1 Normalized Graph Laplacian

The two variations of the normalized graph Laplacian are defined as follows:

\[ L_{\text{sym}} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}} \]
\[ L_{\text{rw}} = D^{-1}L = I - D^{-1}W \]

We list some properties of these two variations \( L_{\text{sym}} \) and \( L_{\text{rw}} \) below.

**Proposition 8.4.1** The normalized graph Laplacians satisfy the following six properties [31]:

1. For arbitrary vector \( f \in \mathbb{R}^n \), we have

\[ f^T L_{\text{sym}} f = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} \left( \frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 \] \hspace{1cm}(8.3)

2. \( \lambda \) is an eigenvalue of \( L_{\text{sym}} \) with eigenvector \( u \) if and only if \( \lambda \) is also an eigenvalue of \( L_{\text{rw}} \) with eigenvector \( w \), satisfying that \( u = D^{\frac{1}{2}}w \).

3. \( \lambda \) is an eigenvalue of \( L_{\text{rw}} \) with eigenvector \( w \) if and only if \( \lambda \) and \( w \) together solve the generalized eigen-problem \( Lw = \lambda Dw \).

4. Both \( L_{\text{sym}} \) and \( L_{\text{rw}} \) are positive semidefinite.

5. The smallest eigenvalue of both \( L_{\text{sym}} \) and \( L_{\text{rw}} \) is 0, with eigenvector \( D^{\frac{1}{2}}1 \) for \( L_{\text{sym}} \) and 1 for \( L_{\text{rw}} \).

6. Both \( L_{\text{sym}} \) and \( L_{\text{rw}} \) have \( n \) nonnegative, real-valued eigenvalues \( 0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \).

**Proof 2** Similar to the proof for Proposition 8.3.1, we can verify the first property from the following equation:

\[ f^T L_{\text{sym}} f = f^T (I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}})f = \sum_{i=1}^{n} f_i^2 - \sum_{i,j=1}^{n} f_i f_j \frac{W_{ij}}{\sqrt{d_i d_j}} \]
\[ = \frac{1}{2} \sum_{i=1}^{n} f_i^2 + \sum_{j=1}^{n} f_j^2 - 2 \sum_{i,j=1}^{n} f_i f_j \frac{W_{ij}}{\sqrt{d_i d_j}} \]
\[ = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} \left( \frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 \]

To prove property 2, we replace \( u \) with \( D^{\frac{1}{2}}w \) in the eigenvalue equation of \( L_{\text{sym}} \):

\[ L_{\text{sym}} u = L_{\text{sym}} D^{\frac{1}{2}}w = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}D^{\frac{1}{2}}w = D^{-\frac{1}{2}}Lw = D^{\frac{1}{2}}\lambda w = \lambda u \]

Property 3 follows directly by multiplying the eigenvalue equation of \( L_{\text{rm}} \) with \( D \) from the left:

\[ DL_{\text{rm}}w = D\lambda w \]
\[ Lw = \lambda Dw \]

Since \( f^T L_{\text{sym}} f \geq 0 \) and both \( D \) and \( W \) are symmetric, \( L_{\text{sym}} \) is then symmetric and positive semidefinite. This property also fits \( L_{\text{rm}} \) due to property 2. Property 5 can be proved directly as \( L_{\text{rw}} 1 = 0 \), and the same statement for \( L_{\text{sym}} \) follows from property 2. Due to the fact that \( L_{\text{sym}} \) and \( L_{\text{rm}} \) are both real symmetric matrices, their eigenvalues should be also real numbers. Then property 6 is proved based on property 5.

\[ ^{3}{ \text{The same case does not hold for } L_{\text{rw}}.} \]
8.4.2 Spectrum Analysis

As in the case of the spectrum analysis for unnormalized spectral clustering, we have a similar proposition [31] on the multiplicity of the 0 eigenvalues. The following proposition relies on Proposition 8.4.1:

**Proposition 8.4.2** The multiplicity \( k \) of eigenvalue 0 of both \( L_{\text{sym}} \) and \( L_{\text{rw}} \) equals the number of connected components \( A_1, A_2, \ldots, A_k \) in the graph. For \( L_{\text{rw}} \), the eigenspace of eigenvalue 0 is spanned by the indicator vectors \( 1_A_1, 1_A_2, \ldots, 1_A_k \) of those components. For \( L_{\text{sym}} \), the eigenspace of eigenvalue 0 is spanned by \( D^{\frac{1}{2}}1_A_i \).

8.4.3 Normalized Spectral Clustering Algorithm

We will introduce two normalized spectral clustering algorithms based on the two normalized graph Laplacian variants. These are aimed at finding orthonormal vectors \( f_1, f_2, \ldots, f_k \) with the following objective functions (assume \( F \in \mathbb{R}^{n \times k} \) is a matrix consisting of orthogonal vectors), respectively:

\[
\begin{align*}
\min_{F} & \quad \text{Tr}(F^T L_{\text{sym}} F) \quad \text{s.t.} \quad F^T F = I \quad (8.4) \\
\min_{F} & \quad \text{Tr}(F^T L F) \quad \text{s.t.} \quad F^T D F = I \quad (8.5)
\end{align*}
\]

We first introduce the normalized spectral clustering algorithm based on \( L_{\text{sym}} \). In this case, the eigenspace of eigenvalue 0 is spanned by \( D^{\frac{1}{2}}1_A_i \) instead of \( 1_A_i \). Therefore, before applying \( k \)-means, one row normalization step is usually adopted for \( L_{\text{sym}} \) in order to make each row comparable in the Euclidean space. This is an additional step, which needs to be applied, beyond what is discussed for the case of unnormalized spectral clustering.

The algorithm is shown below.

**Normalized Spectral Clustering (symmetric version)**

1. Construct similarity graph by one of the methods described in Section 8.2. Let \( W \) be the adjacency matrix and \( D \) be the degree matrix.
2. Compute symmetric normalized graph Laplacian \( L_{\text{sym}} \) where \( L_{\text{sym}} = D^{-\frac{1}{2}}L D^{-\frac{1}{2}} \).
3. Determine \( f_1, f_2, \ldots, f_k \), the top \( k \) eigenvectors of \( L_{\text{sym}} \).
4. Construct the matrix \( F \in \mathbb{R}^{n \times k} \) from \( f_1, f_2, \ldots, f_k \).
5. Normalize the rows of \( F \) to 1 such that \( \forall i \leq n, \sum_j F_{ij}^2 = 1 \).
6. Treat each row of \( F \) as a vertex in \( \mathbb{R}^k \), partition these vertices into \( k \) clusters via \( k \)-means algorithm.

The other version of normalized spectral clustering algorithms is based on \( L_{\text{rw}} \). This version is more similar to the unnormalized case, because their eigenspaces are both spanned by \( 1_A_i \) in the ideal case where there are \( k \) connected components. Therefore, their procedures are almost the same except for the computation step for the graph Laplacian.
Normalized Spectral Clustering (random walk version)

1. Construct similarity graph by one of the methods described in Section 8.2. Let $W$ be the adjacency matrix and $D$ be the degree matrix.

2. Compute unnormalized graph Laplacian $L$ where $L = D - W$.

3. Find $f_1, f_2, \ldots, f_k$, the top $k$ eigenvectors of the generalized eigenproblem $Lf = \lambda Df$.

4. Form the matrix $F \in \mathbb{R}^{n \times k}$ from $f_1, f_2, \ldots, f_k$.

5. Treat each row of $F$ as a vertex in $\mathbb{R}^k$, partition these vertices into $k$ clusters via $k$-means algorithm.

8.5 Graph Cut View

In the last two sections, we introduced the definitions for different formulations of graph Laplacians and specific procedures for all three kinds of spectral clustering algorithms. Although we have analyzed the spectrum of both unnormalized and normalized Laplacian matrices, the intuition behind the graph Laplacians is not completely clear. Therefore, in this section, the spectral clustering method will be presented from the perspective of the graph cut.

Given a graph $G = (V, E)$, suppose we have two subsets of vertices $A_1, A_2$, satisfying $A_1 \cap A_2 = \emptyset$ and $A_1 \cup A_2 \subseteq V$. We then introduce the definition of the cut as the sum of weights of the edges across the two subsets:

$$\text{cut}(A_1, A_2) = \sum_{v_i \in A_1, v_j \in A_2} W_{ij}$$

Here, $W_{ij}$ is the weight of the edge between vertices $i$ and $j$. It is evident that the cut describes the closeness between these two subsets in the graph. In other words, a smaller value of the cut indicates greater separability of the two subsets.

In graph theory, a minimum cut (MinCut) of a graph is a cut with the smallest possible value under the condition $A_1 \cup A_2 = V$. This is also the most direct way to construct the partition. MinCut is a classical problem in the literature and relatively easy to solve. More details of MinCut may be found in [49].

By extending the cut definition from two sets to the multi-set situation, we can reformulate the cut definition as follows:

$$\text{cut}(A_1, A_2, \ldots, A_k) = \sum_{i=1}^{k} \text{cut}(A_i, \overline{A_i})$$

where $\overline{A_i}$ stands for the set of vertices $\{v_j | v_j \notin A_i\}$.

Although MinCut is an intuitive way to partition the graph, it suffers from a critical problem, which is quite common in graph partitioning. In many cases, MinCut separates an individual vertex or a small set of vertices from the remaining graph, overlooking the balance between the sizes of different partitions. Some normalization strategies can be incorporated into the objective function of MinCut to circumvent this problem. Two examples of such normalized objective functions are the Ratio Cut (RatioCut) [19] and Normalized Cut (NCut) [43].
8.5.1 Ratio Cut Relaxation

Suppose we already have $k$ vectors $\{f_i\}_{i=1}^k$ indicating the cluster labels for the vertices in the graph where $f_i = (f_{i1}, \ldots, f_{in})^T$. Define that:

$$f_{ij} = \begin{cases} 
\frac{1}{\sqrt{|A_i|}} & \text{if } v_j \in A_i \\
0 & \text{otherwise}
\end{cases} \quad (8.6)$$

Then, under this condition, for each cluster $i$ among all $k$ clusters, we can rewrite Equation 8.1 of the unnormalized graph Laplacian as follows:

$$f_i^T L f_i = \frac{1}{2} \sum_{j,l=1}^n W_{jl} (f_{ij} - f_{il})^2$$

$$= \frac{1}{2} \left( \sum_{v_j \in A_i, v_l \not\in A_i} W_{jl} \left( \frac{1}{\sqrt{|A_i|}} - 0 \right)^2 + \sum_{v_j \not\in A_i, v_l \in A_i} W_{jl} \left( 0 - \frac{1}{\sqrt{|A_i|}} \right)^2 \right)$$

$$= \frac{1}{2} \frac{\text{cut}(A_i, \overline{A_i})}{|A_i|} + \frac{1}{2} \frac{\text{cut}(\overline{A_i}, A_i)}{|A_i|}$$

$$= \frac{\text{cut}(A_i, \overline{A_i})}{|A_i|}$$

The RatioCut measure is linked to the unnormalized graph Laplacian because of the following calculation:

$$\sum_{i=1}^k f_i^T L f_i = \text{Tr}(F^T L F)$$

$$= \sum_{i=1}^k \frac{\text{cut}(A_i, \overline{A_i})}{|A_i|}$$

$$= \text{RatioCut}(A_1, \ldots, A_k)$$

where $F \in \mathbb{R}^{n \times k}$ is formed from $f_1, f_2, \ldots, f_k$. 

The RatioCut is defined as follows:

$$\text{RatioCut}(A_1, \ldots, A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, \overline{A_i})}{|A_i|}$$

Here, $|A_i|$ denotes the number of vertices belonging to $A_i$. By minimizing this measure, the partitions with more balanced cluster sizes are preferred.

The objective function for NCut is defined as follows:

$$\text{NCut}(A_1, \ldots, A_k) = \sum_{i=1}^k \frac{\text{cut}(A_i, \overline{A_i})}{\text{assoc}(A_i, V)}$$

where $\text{assoc}(A_i, V) = \sum_{v_j \in A_i} (d_j)$. Different from RatioCut which measures the balance by the cluster size, NCut focuses on the total degree within different clusters.

The incorporation of normalization into the objective function makes it NP-hard [53]. However, by relaxing some constraints, spectral clustering is a way to give the approximate solution. Specifically, unnormalized spectral clustering can be used to solve the relaxed RatioCut problem, and normalized spectral clusterings are designed to solve the relaxed NCut problem.
Moreover, according to Equation 8.6, \( f_i \) satisfies the following:

\[
f_i^T f_i = \sum_{j=1}^{n} f_{ij}^2 = |A_i| \frac{1}{|A_i|} = 1
\]

It is also easy to check that any two different column vectors of \( F \) are orthogonal.

Now the RatioCut problem is equivalent to the following:

\[
\min_{A_1, \ldots, A_k} \text{Tr}(F^T L F) \quad \text{s.t.} \quad F^T F = I
\]  

(8.7)

Note that \( f_{ij} \) can only be 0 or \( 1/\sqrt{|A_i|} \). The objective function is therefore discrete, which is known to be NP-hard in the general case. A relaxation of this problem allows \( f_{ij} \) to take on arbitrary real values. This implies the following:

\[
\min_{F \in \mathbb{R}^{n \times k}} \text{Tr}(F^T L F) \quad \text{s.t.} \quad F^T F = I
\]

The aforementioned objective function is exactly the same as the objective function in Equation (8.2) for unnormalized spectral clustering.

According to Rayleigh-Ritz theorem [30], to minimize the refixed RatioCut objective function is equivalent to minimizing Rayleigh quotient of the unnormalized spectral clustering, where the Rayleigh quotient \( R(L, f) \) is defined as follows:

\[
R(L, f) = \frac{f^T L f}{f^T f}
\]

The Rayleigh quotient (objective function value) reaches its minimum value (the smallest eigenvalue of \( L \) ) when \( f \) is the corresponding eigenvector\(^4\). Therefore, the solution of Equation (8.2) is obtained by choosing the top \( k \) eigenvectors of \( L \). Later, we need to convert the real valued matrix \( F \) back to discrete indicators. As mentioned in Section 8.3.4, the \( k \)-means approach can be the candidate algorithm to be adopted on the rows of \( F \).

### 8.5.2 Normalized Cut Relaxation

Similar to RatioCut, we first define the indicator vectors \( f_i = (f_{i1}, \ldots, f_{in})^T \) where \( 1 \leq i \leq k \) such that:

\[
f_{ij} = \begin{cases} 
\frac{1}{\sqrt{\text{assoc}(A_i, V)}} & \text{if } v_j \in A_i \\
0 & \text{otherwise.}
\end{cases}
\]  

(8.8)

Under this condition, we can rewrite Equation 8.1 of the unnormalized graph Laplacian as follows:

\[
f_i^T L f_i = \frac{1}{2} \sum_{j=1}^{n} \sum_{l=1}^{n} W_{jl}(f_{ij} - f_{il})^2
\]

\[
= \frac{1}{2} \left( \sum_{j \in A_i, v_j \notin A_i} W_{ij} \frac{1}{\text{assoc}(A_i, V)} + \sum_{j \notin A_i, v_j \in A_i} W_{ij} \frac{1}{\text{assoc}(A_i, V)} \right)
\]

\[
= \frac{1}{2} \left( \text{cut}(A_i, \overline{A_i}) + \text{cut}(\overline{A_i}, A_i) \right) + \frac{1}{2} \left( \frac{\text{cut}(A_i, V)}{\text{assoc}(A_i, V)} \right)
\]

\[
= \frac{\text{cut}(A_i, A_i)}{\text{assoc}(A_i, V)}
\]

\(^4\)The other eigenvalues and corresponding eigenvectors of \( L \) follow the same way.
The NCut measure can be linked with the graph Laplacian, based on the following calculation:

\[ \sum_{i=1}^{k} f_i^T L f_i = \text{Tr}(F^T L F) \]

\[ = \sum_{i=1}^{k} \text{cut}(A_i, \overline{A}_i) \]

\[ = \text{assoc}(A_i, V) \]

\[ = \text{NCut}(A_1, \ldots, A_k) \]

The matrix \( F \in \mathbb{R}^{n \times k} \) can be constructed with the \( k \) column vectors \( f_1, f_2, \ldots, f_k \).

According to Equation 8.8, \( f_i \) satisfies the following:

\[ f_i^T D f_i = \sum_{j=1}^{n} d_j f_{ij}^2 = \frac{\sum_{v_j \in A_i} d_j}{\text{assoc}(A_i, V)} = \frac{\text{assoc}(A_i, V)}{\text{assoc}(A_i, V)} = 1 \]

For any pair of columns \( f_i, f_j \) from \( F \), it is easy to show that they are orthogonal.

The NCut objective function is equivalent to the following:

\[ \min_{A_1, \ldots, A_k} \text{Tr}(F^T L F) \quad \text{s.t.} \quad F^T D F = I \]

This is exactly the same as the objective function in Equation (8.5) for normalized spectral clustering.

If we define \( P = D^{1/2} F \) according to Proposition 8.4.1, and substitute \( F \) by \( P \) in the above relaxed objective function for NCut, we have

\[ \min_{P \in \mathbb{R}^{n \times k}} \text{Tr}(P^T D^{1/2} L D^{1/2} P) \quad \text{s.t.} \quad P^T P = I \]

This is exactly the same as the symmetric version of objective function in Equation (8.4) for normalized spectral clustering. The relaxed NCut problem can be solved by simply computing the first \( k \) eigenvectors of \( L_{rw} \). The first \( k \) eigenvectors of the generalized eigenproblem \( L f = \lambda D f \) is one method adopted in [43]. To convert the real value matrix \( F \) (the first \( k \) eigenvectors of \( L_{rw} \)) back to discrete indicators, \( k \)-means could be adopted.

We can also compute the first \( k \) eigenvectors of \( L_{sym} \) instead of \( L_{rw} \) according to [37]. For \( P \) (the first \( k \) eigenvectors of \( L_{sym} \)), we need row normalization before discretization because \( P = D^{1/2} F \).

### 8.6 Random Walks View

A random walk on a graph is a Markov chain which can be described by an \( n \times n \) square matrix \( M \), where \( n \) is the number of vertices in the graph. The matrix \( M \) denotes the transition probabilities. Therefore, the conditional probability of the next state being vertex \( v_j \), given the current state \( v_i \), is given by \( 0 \leq P(v_i | v_j) = M_{ij} \leq 1 \). If there is no edge from vertex \( v_i \) to vertex \( v_j \), then \( M_{ij} = 0 \). The matrix \( M \) is a stochastic matrix. Therefore, all its entries are nonnegative and every row adds up to one.
A formal relationship analysis between NCut and random walks has been provided in [32]. Specifically, the transition probability $M_{ij}$ is proportional to the edge weight $W_{ij}$ and is computed by the following:

$$M_{ij} = \frac{W_{ij}}{d_i}$$

If we use matrix operations to represent this computation, we have:

$$M = D^{-1}W$$

In the following, we will use the transition matrix $M$ to achieve a better understanding of the normalized spectral clustering algorithm. First, define $\pi = (\pi_1, \ldots, \pi_n)^T$ to be a vector with length $n$ as follows:

$$\pi_i = \frac{d_i}{\text{assoc}(V, V)}$$

Here, $\text{assoc}(V, V)$ represents the total degrees in the graph. It is easy to verify that $M^T \pi = \pi$ and, thus, that $\pi$ is a stationary distribution of the Markov chain.

Assume in the graph $G = (V, E)$, we have two disjoint subsets $A$ and $\overline{A}$. Define $M_{AA} = P(A \rightarrow \overline{A}|A)$ as the probability of the random walk transiting from set $A$ to set $\overline{A}$ in one step if the current state is in $A$ and the random walk is started from its stationary distribution.

$$M_{AA} = \frac{\sum_{i \in A, j \in \overline{A}} \pi_i M_{ij}}{\sum_{i \in A} \pi_i}$$

$$= \frac{\sum_{i \in A, j \in \overline{A}} \pi_i \frac{W_{ij}}{d_i}}{\sum_{i \in A} \pi_i}$$

$$= \frac{\sum_{i \in A, j \in \overline{A}} \frac{W_{ij}}{\text{assoc}(V, V)}}{\sum_{i \in A} \frac{d_i}{\text{assoc}(V, V)}}$$

$$= \frac{\text{cut}(A, \overline{A})}{\text{assoc}(A, V)}$$

From this, we now have:

$$\text{NCut}(A, \overline{A}) = M_{AA} + M_{\overline{A}}$$

If the NCut measure is small for a certain partition $A$ and $\overline{A}$, then the probability of the walk moving from one partition to the other is small. In other words, when minimizing NCut, we are trying to partition the set $V$ into different groups such that the random walk, once in one of the parts, tends to remain in it [32].

This random walk view gives a new and intuitive characterization of the NCut algorithm. The NCut algorithm is just another view of spectral clustering. Moreover, as the random walk is essentially a Markov chain on the graph, it provides spectral clustering algorithm with a probabilistic foundation.

### 8.7 Connection to Laplacian Eigenmap

The Laplacian eigenmap algorithm [2] is a very successful method for dimensionality reduction and is closely related to spectral clustering. Dimensionality reduction is also called subspace learning or feature extraction, aiming at extracting low-dimensional structure dimensionality from high-dimensional data. One of the most popular dimensionality reduction algorithms is Principal Component Analysis (PCA) which gives the solution by projecting the original high-dimensional
data onto the low-dimensional linear subspace spanned by the leading eigenvectors of the data’s covariance matrix.

In many real-world problems, the data is not always sampled from a linear subspace. For example, it is always believed that the face images are sampled from a nonlinear low-dimensional manifold which is embedded in the high-dimensional ambient space. This has motivated researchers to consider manifold-based techniques for dimensionality reduction and Laplacian eigenmap is just one of them. The basic idea of Laplacian eigenmap is also based on spectral graph theory [8]. Its objective function is similar to that in normalized spectral clustering:

\[
\arg\min_F \text{Tr}(F^T L F) \quad \text{s.t.} \quad F^T D F = I
\]

where \( F \) is considered to be the new low-dimensional representation.

We can reconsider the following equation from the dimensionality reduction point of view:

\[
f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} (f_i - f_j)^2
\]

The intuition is that for any pair of data points, if they are near to each other on the high-dimensional space (a space in which a low-dimensional manifold is embedded), they should still be close in the mapped low-dimensional space. In real life, the underlying manifold is often unknown. Similar to [50], the geodesic distance between all pairs of points on the manifold can be reflected by their shortest path distance on the adjacency graph. Therefore, the minimization of the aforementioned cost function based on the graph preserves the local distance between every pair of data points. The additional constraint \( F^T D F = I \) fixes the arbitrary scaling factor in the embedding.

Figure 8.3 is used in [50] to illustrate how geodesic paths work for nonlinear dimensionality reduction. In the left figure, for two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high-dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). The figure on the right shows an approximation to the true geodesic path, as the shortest path in adjacency matrix \( W \).

We can see that the only difference between these two algorithms is that spectral clustering makes one step further to cluster the information in the new dimension. It is discussed in [3] that for the local approaches to do dimensionality reduction, although they attempt only to approximate or preserve neighborhood information, they may also be interpreted as imposing a soft clustering of the data (which may be converted to a hard clustering by a variety of heuristic techniques such as \( k \)-means). In this sense, the local approaches to dimensionality reduction impose a natural clustering on the data.

**FIGURE 8.3:** The “swiss roll” data set, illustrating how geodesic paths work for nonlinear dimensionality reduction.
8.8 Connection to Kernel $k$-Means and Nonnegative Matrix Factorization

Recent studies [12, 13] show that spectral clustering is theoretically closely related to two other popular clustering algorithms: kernel $k$-means and nonnegative matrix factorization [25]. It is proved that these three are unified in a simple way: they are different prescriptions of the same problem with slightly different constraints and optimization solutions.

It is known that $k$-means uses $k$ centroids to characterize the data. The objective function is to minimize the sum of squared errors [13]:

$$ J = \sum_{i=1}^{k} \sum_{j \in A_i} ||x_j - m_i||^2 = c - \sum_{i=1}^{k} \frac{1}{|A_i|} \sum_{j \in A_i} x_j^T x_j $$

(8.9)

where $m_i = \sum_{j \in A_i} x_j / |A_i|$ is the centroid of cluster $A_i$ of $|A_i|$ points, and $c = \sum_{j} ||x_j||^2$.

We can represent the solution by $k$ nonnegative indicator vectors:

$$ F = (f_1, \cdots, f_k), \quad f_i^T f_j = \delta_{ij} $$

where

$$ f_{ij} = \begin{cases} 
\frac{1}{\sqrt{|A_i|}} & \text{if } x_j \in A_i \\
0 & \text{otherwise.} 
\end{cases} $$

We can rewrite Equation 8.9 as $J = Tr(X^T X) - Tr(F^T X^T X F)$. As the first term is a constant, minimizing $J$ is equal to the following:

$$ \max Tr(F^T G F) $$

Here, $G = X^T X$ is a pairwise similarity matrix called inner-product linear kernel matrix which can be extended to any other kernels. It is clear that if we set the kernel matrix $G$ to be the negative of unnormalized graph Laplacian $L = W - D$, the objective function of kernel $k$-means clustering is equivalent to that of RatioCut in Equation 8.7. Therefore, for the same objective function, spectral clustering first relaxes $F$ to be real value and computes it through solving an eigen-decomposition problem. The clustering labels are finally obtained by running $k$-means on $F$. Different from this “global” minimum after relaxation of $F$, kernel $k$-means uses an iterative algorithm similar to $k$-means to directly partition all data points, which is essentially a local search. It is worth noting that in [12], a weighted kernel $k$-means is proposed and the authors also prove it equivalent to spectral clustering where kernel matrix $G$ is set to $D^{-1} W D^{-1}$ and the weight matrix for data points equals to $D$.

Nonnegative matrix factorization (NMF) aims at factorizing $X$ into two nonnegative matrices,

$$ X \approx UV^T $$

where $U \in \mathbb{R}^{n \times k}_+$ and $V \in \mathbb{R}^{m \times k}_+$. As $k \ll \min(m, n)$, $UV^T$ can be viewed as a low-rank approximation of $X$. The clustering aspect of NMF is studied in [57, 18].

Now we consider the symmetric NMF of

$$ G \approx FF^T $$

5If $G$ is set to be $I - L_{sym}$, then the objective function is equivalent to NCut, which is proved in [13].
The factorizations can be obtained by the least squares minimization:

\[ F = \arg\min_{F \geq 0} \|G - FF^T\|^2 \]

\[ = \arg\min_{F \geq 0} \|G\|^2 - 2\text{Tr}(F^T GF) + \|F^T F\|^2 \]

\[ = \arg\max_{F \geq 0} \text{Tr}(F^T GF) \]

Again if we set \( G \) to be \(-L = W - D\), the objective function of NMF is very similar to unnormalized spectral clustering except that for NMF we require \( F \) to be nonnegative while for spectral clustering \( F \) is expected to be orthonormal.

8.9 Large Scale Spectral Clustering

The general spectral clustering method needs to construct an adjacency matrix and calculate the eigen-decomposition of the corresponding Laplacian matrix. Both of these two steps are computational expensive. For a data set consisting of \( n \) data points with \( m \) dimensions, the above two steps will have time complexities of \( O(n^2m) \) and \( O(n^3) \), respectively. This is an unreasonable burden for large-scale applications.

Fortunately, if we only want to construct an approximate adjacency matrix with a few number of nearest neighbors, there is a shortcut for reducing the complexity. This is called approximate nearest neighbor search. We first introduce randomized kd-trees [45, 36]. A kd-tree partitions the space by hyperplanes that are perpendicular to the coordinate axes. At each node of the tree, the corresponding data points are partitioned into two halves with a hyperplane through a specified dimension which exhibits the greatest variance. As kd-tree’s performance rapidly degrades with the number of dimensions increasing, Silpa-Anan and Hartley [45] recently have proposed an improved version of the kd-tree algorithm in which multiple randomized kd-trees are created. By comparison, the randomized trees are built by choosing the split dimension randomly from the first few dimensions on which data has the greatest variance. The other available technique is called locality-sensitive hashing (LSH) [10] which offers probabilistic guarantees of retrieving data points similar to the query. The basic idea is to compute a set of functions that reduce the dimensionality of the data points through certain mathematical operations. By concatenating several such functions to form a hash function, data points mapped in the same bin as the query are returned as candidates of nearest neighbors with probabilistic guarantees. Moreover, several hash tables can be built in the same way to further enhance the performance. A few libraries for approximate nearest neighbor search are available, including FLANN [36] and ANN [35].

For the problem of eigen-decomposition, state-of-the-art eigensolvers can be categorized into three classes: local, global, and extreme solvers [27]. The power method is the simplest local solver, which only computes the maximum eigenvalue and its associated eigenvector. QR-decomposition method is a typical global solver, which computes all the eigenvalues and eigenvectors. Extreme eigensolvers compute several extreme (smallest or largest) eigenvalues and the corresponding eigenvectors, which is obviously the most suitable for spectral clustering since the number of clusters \( k \) usually is not too large. Typical examples of extreme eigensolvers include Lanczos, preconditioned conjugate gradient (PCG), and Jacobi-Davidson (JD) methods. Interested readers can refer to [41] for details. Several software implementations of the Lanczos method are publicly available and are well maintained, e.g., ARPACK [26]. PCG and JD methods also have some implementations available [22, 48]. If the Laplacian matrix is sparse, i.e., the similarity graph is not fully connected, these state-of-art eigensolvers can benefit from this property because most elements are zero and could be removed from the computation.
Besides this, many works have been proposed to accelerate spectral clustering itself instead of focusing on the general eigensolvers. [16] adopts the classical Nyström method which was originally proposed for finding numerical approximations to eigenfunction problems. It first chooses samples randomly among the data points to obtain the small-size eigenvectors and then extrapolates the solution for the whole dataset with weights. The work in [44] reduces the original data set to a relatively small number of data points in a biased way. The first step is to apply an efficient clustering algorithm such as $k$-means to get the initial partitions. Then data points near the same centroid are merged and viewed as one. By some additional similarity computations for such new representations, the algorithm can operate directly on this smaller graph. Similar to this idea, in [58], all data points are collapsed into centroids through $k$-means or random projection trees so that the later eigen-decomposition needs to be applied only on the centroids. [42] uses random projection for another purpose which is to reduce the dimensionality of $X$ (i.e., $m$ mentioned at the beginning) and therefore to help save time in constructing the adjacency matrix. Additionally, random sampling is later applied to drastically reduce the size of data points within the eigen-decomposition step. [4, 27] introduce early stop strategies to speed up eigen-decomposition. It is based on observation that well-separated data points will converge to the final embedding more quickly, and hence, we can sequentially depress the scale of the Laplacian matrix in the iteration steps of conjugate-based eigensolvers without inferringing the final clustering results by much. In [6], landmark points are first selected randomly or through $k$-means among all the data points to serve as codebook. Later each data point can be encoded in the form of linear combinations of these landmarks. After constructing the affinity matrix by inner product using such landmark representations, acceleration can be achieved since the complexity now is related to the number of landmark points which is far less than $n$. Different from the above methods, authors in [21] work on resistance distance embedding which has an idea similar to spectral clustering and also has quite similar clustering capability. The proposed method does not require using any sampling technique or computing any eigenvector. Instead it uses random projection and a linear time solver of Spielman and Teng [47] to find the approximate and accurate embedding.

To address resource bottlenecks of both memory use and computation time, another promising direction is to make use of distributed system. In [5], a parallel framework is proposed which first distributes $n$ data instances onto $p$ distributed machine nodes. On each node, similarities between local data and the whole set are computed in a way that uses minimal disk I/O. The eigenvector matrices are stored on distributed nodes to reduce per-node memory use. Finally, together with parallel eigensolver and $k$-means, the proposed parallel spectral clustering achieves good speedup with large data sets. Additionally, for constructing the sparse similarity matrix, the authors choose MapReduce because this step may be the most time consuming and having a fault tolerant mechanism is essential. For eigen-decomposition and the $k$-means step, MPI-based open source softwares are adopted. Gao et al. [17] use a different approach to first map data points into buckets and then implement spectral clustering for each bucket simultaneously. The main idea is within the preprocess step, which is to adopt locality sensitive hashing (LSH) [10] to create signatures (i.e., $M$-bit binary numbers) for data points and later to project near-duplicate signatures\(^6\) into one bucket. The hash function they use to generate the signatures falls into the family of random projection. Besides this, they propose the MapReduce algorithm for LSH and call the spectral clustering module in Apache Mahout [40] which is also implemented on top of Apache Hadoop using the MapReduce paradigm.

\(^6\)Two $M$-bit signatures are considered to be near-duplicate if they have at least $M - \epsilon$ bits in common.
8.10 Further Reading

The family of spectral clustering has become popular especially since the works in [37] and [43]. Since then, several works have been proposed to slightly change the original framework and achieve better clustering results. In [59], the final \(k\)-means step on the eigenvectors shared by the family of spectral clustering is explained not to be optimal. The authors clarify the role of eigenvectors obtained from graph Laplacian as a generator of all optimal solutions through orthonormal transforms. A better solution is then proposed by rotating normalized eigenvectors to obtain an optimal segmentation. Their method iterates between nonmaximum suppression and using SVD to recover the rotation. Later in [60], it is pointed out that this iterative method can easily get stuck in local minima and thus does not reliably find the optimal alignment. To find the best alignment for a set of eigenvectors, the authors define a new cost function which measures the alignment quality of a set of vectors and adopts a gradient descent scheme which provably converges. However, this paper also mentions that such alignments might only be effective when the data is not highly noisy. Another insight of this paper concerns the construction of adjacency matrix \(W\). It suggests that in the fully connected graph, any two vertices are connected and the value of \(W_{ij}\) is defined to be \(e^{-\frac{||x_i-x_j||^2}{\sigma_i\sigma_j}}\). Here \(\sigma_i\) is called a local scale and is set to be the distance between vertex \(i\) and its certain neighbor (7th nearest neighbor in the paper). The intuition behind this method is that in reality data resides in multiple scales (one cluster is tight and the other is sparse) and local scaling automatically finds the two scales and results in high affinities within clusters and low affinities across clusters. Later a more natural parameterization of the neighborhood density based on the empty region graphs is proposed in [9] which is claimed to be not sensitive to parameters and data perturbation.

Many papers have also been published to discover the theoretical relations of spectral clustering to other algorithms. As discussed in Section 8.8, there exist close links among spectral clustering, kernel \(k\)-means, and nonnegative matrix factorization. Besides this, relations between spectral clustering and kernel principal component analysis (KPCA) are built on the fact that the Laplacian eigenmap can be interpreted as KPCA with kernel matrix set to be the Moore-Penrose inverse of \(L\) [20]. From Section 8.7 we know that Laplacian eigenmap can be essentially viewed as a nonlinear dimensionality reduction method which is almost the same as spectral clustering except for the final \(k\)-means step.

Spectral clustering algorithms are extended to many nonstandard data types compared to those introduced at the beginning of this chapter. In [11], a co-clustering algorithm is proposed to the bipartite graph which is a kind of graph that can be divided into two sets such that no two data points in the same set are connected, e.g., document–word graph. [28] was later published to address the clustering problem through collective factorization for heterogeneous data types which is more complicated and general than the bipartite graph case. For network data where only nodes and links are observed, a spectral clustering-like algorithm [56] is designed for optimizing “modularity” measure instead of “cut” measure because the former is also popular in the community detection. Another interesting extension to incorporate is temporal information. Chi et al. [7] present two frameworks in evolutionary spectral clustering through regularizing one additional cost on temporal smoothness. If the data points are observed in a stream-like way, [51] has been proposed to update the clustering directly without evaluating the entire affinity matrix. In [38], a more general algorithm allowing similarity changes between existing data points has been developed based on incrementally updating the eigenvectors.

Besides all these, in the machine learning community, spectral clustering techniques usually appear with other learning tasks. For example, in the semisupervised scenario, pairwise constraints, i.e., must-link and cannot-link, are incorporated into spectral clustering as the user guidance for the partition [29]. Different from this, in [61], a few class labels are directly provided and a semi-
supervised classification function similar to spectral clustering is proposed requiring the smoothness of inference between known labeled and unlabeled data. Multi-view learning is another hot topic where multiple complementary representations of the data are presented. In [62], the spectral clustering approach is generalized to multiple views via a random walk formulation. The work in [23] approaches the problem by adopting a co-training framework such that the similarity matrix in one view is affected by the similarity estimated based on the eigenvectors of graph Laplacian in the other view. In [24], a co-regularization framework is proposed to enforce the pairwise similarities computed from the eigenvectors learned from different views to be close with each other. Opposite to multi-view algorithms which learn the latent consensus from multiple representations, the work in [39] tries to find multiple nonredundant clustering results from a single data view through penalizing for redundancy between different results. There are also extensions of spectral clustering algorithms in the field of active learning. The work in [54] formulates this problem as incrementally querying the oracle about pairwise relations mentioned above between chosen data points. In [55], the “active” part lies in querying the pairwise similarity under the assumption that the adjacency matrix \( W \) is expensive to obtain or compute.

In the last few years, many papers have been published concerning combinations of spectral clustering with different data types, learning tasks, and applications. Interested readers are encouraged to further explore the literature in these directions.

Bibliography


Data Clustering: Algorithms and Applications


Spectral Clustering


