Research Report - Spectral Clustering

Sourabh Antani

Chapter 1

Introduction

Nowadays, clustering has evolved into a class of most fundamental methods frequently applied in exploratory data analysis. Clustering helps the researchers understand the diversity of data and the number of different populations that exist. This knowledge, in turn, aids in subsequent tasks like the choice of modeling or classification techniques and the parameters for tuning them.

In this report, I will, first, list some of the commonly used clustering techniques before diving deeper into Spectral Clustering. In the next chapter, I will dig deeper into spectral clustering. Finally, I will list some of the possible avenues of research that I wish to explore.

Clustering Algorithms

Below are some examples of clustering algorithms used in practice, aside from spectral clustering.

Some common examples of algorithms that rely on euclidean distance and spatial density of points are:

- **k-means** This algorithm iteratively computes the centroids of the clusters using euclidean distances. The term was first used by J. Macqueen [1] but standard algorithm published by Lloyd [2].
- **BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)** This algorithm extracts centroids from building hierarchies in height balance trees [3]
- **Mean-shift** This algorithm initializes the centroids randomly but then iteratively refines them by updating the centroids to be the centroid of all the points in a neighborhood of each of them [4].
- **DBSCAN** This algorithm creates clusters by looking at the density of a neighborhood of a point and combining nearby areas density equal to or higher than a given threshold [5]

While these are generally simpler to understand, the 'shape' of the spatial distribution greatly affects the effectiveness of these algorithms.

Some examples of algorithms based on linear algebra/matrix factorization are:

- **Non-Negative Matrix Factorization** The main idea of this algorithm is to express a large matrix as the product of a tall and a wide matrix. The columns of the tall matrix represent the centroids of the clusters. The columns of the input matrix can be computed as a linear combination of the columns of the tall matrix with the coefficients from a column of the wide matrix [6] [7]. Later [8] studied the equivalence of NNMF with Spectral Clustering. This technique seeks to factorize the matrix and use one of the factors as the approximate basis for the matrix, thus effectively reducing the dimensionality of the problem. Another advantage of these methods is that it simple to gauge how 'good' the approximation is by looking at the difference between the product of approximated factors and the original matrix.
- **PCA Based clustering** The idea of this method is to apply PCA to a dataset and then plot the data on the first two principal components and apply k-means clustering. [9]

Chapter 2

Spectral Clustering

An excellent tutorial for Spectral Clustering can be found in [10]. Before we look at the various algorithms, let us go through some basic terms involved.

2.1 Some Terminology

Given an **undirected graph** $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with $\mathbf{V} = \{\mathbf{v_1}, \ldots, \mathbf{v_n}\}$ being the **set of vertices** and $\mathbf{E}$ being the **set of weighted edges** such that each edge between two vertices v_i and v_j carries a non-negative weight $w_{ij} \geq 0$ that is computed based on the similarity function.

The **Weighted *adjacency matrix*** of the graph is the matrix $\mathbf{W} = (\mathbf{w_{ij}})_{\mathbf{i,j=1,\ldots,n}}$. If $w_{ij} = 0$, vertices v_i and v_j are not connected. $w_{ij} = w_{ji}$ since G is undirected.

The **degree of a vertex** $v_i \in V$ is defined as $\mathbf{d_i} = \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{w_{ij}}$. ***Degree matrix*** $\mathbf{D}$ is defined as the diagonal matrix with the degrees $d_1, \ldots, d_n$ on the diagonal.

$\bar{\mathbf{A}}$ denotes the **Complement** $V \backslash A$ of a given subset $A \subset V$. **Indicator vector** $\mathbf{1}_A$ is a vector with entries $f_i = 1$ if $v_i \in A$, $f_i = 0$ otherwise.

Define $\mathbf{W(A, B)} = \sum_{\mathbf{i \in A, j \in B}} \mathbf{w_{ij}}$, for not necessarily disjoint sets $A, B \subset V$. The **'size'** of $A \subset V$ can be measured in two ways, $|\mathbf{A}|$:= **number of vertices in** A or $\mathbf{vol(A)} := \sum_{\mathbf{i \in A}} \mathbf{d_i}$ which is **the sum of edge weights** attached to the vertices in A

A subset $A \subset V$ of a graph is **connected** if any two vertices in A can be joined by a path such that all intermediate points also lie in A. A is called **Connected Component** if it is a connected subset such that A and $\bar{A}$ are disjoint. Finally, **Partitions** of a graph are defined as non-empty sets $A_1, \ldots, A_k$ form partition of graph if $A_i \cap A_j = \emptyset$ and $A_1 \cup \cdots \cup A_k = V$

2.2 Graph Laplacians and Graph cuts

2.2.1 Graph Laplacians

The following types of graph Laplacians have been defined in the literature:

$$\begin{aligned} \textbf{Unnormalized Graph Laplacian:} \quad & L = D - W \\ \textbf{Normalized Graph Laplacian (Symmetric):} \quad & L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \\ \textbf{Normalized Graph Laplacian (Random walk):} \quad & L_{rw} = D^{-1} L = I - D^{-1} W \end{aligned}$$

Below are some important properties of the three graph Laplacians:

- All three graph Laplacians are positive-semidefinte and have non-negative real-valued eigenvalues.
- 0 is an eigenvalue with multiplicity equal to the number of connected components of the graph. Thus for a fully connected graph one of the eigenvalues is 0.

- L and L_{sym} are symmetric.
- The eigenvectors of L_{rw} are the eigenvectors of L while $D^{1/2}u$ is eigenvector of L_{sym} if u is eigenvector of L.

For proofs of the above properties, refer to the appendix. As far as naming goes, L_{sym} follows from its structure. While L_{rw} is so named because its structure is derived from the point of view of random walk on the graph. Taking $p_j = w_{ij}/d_i$ as the probability of an edge between v_i and v_j being selected for random walk, the transition matrix for the random walk is defined by $P = D^{-1}W$. Thus $L_{rw} = I - P$. This is why the second form of normalized graph Laplacian is denoted by L_{rw}.

For information about unnormalized graph Laplacian, refer to [11] and [12] while the standard reference for normalized graph Laplacian is [13]. Formal equivalence between Ncut and transition probability of the random walk has been outlined in [14]

2.2.2 Graph Cuts

The intuition of clustering is to divide the graph into groups of vertices such that the edges between the vertices in the same group have high weight and the edges between vertices from different groups have zero or very low weight. Thus effective clustering is to solve the minicut problem which can be defined as, for a given number k subsets, choosing the partition $A_1, \ldots, A_k$ which minimizes

$$\text{cut}(A_1, \ldots, A_k) := \frac{1}{2}\sum_{i-1}^{k} W(A_i, \bar{A}_i)$$

To reduce the chances that minicut simply separates an individual vertex, two approaches have been suggested to make sure that the clusters are 'reasonably large'.

First is RatioCut [15] where the solution is to solve minicut problem while dividing the graph in components with roughly the same number of vertices. The second approach is Ncut [16] which tries to solve the minicut problem by keeping the volume of each component, i.e. sum of edge weights, roughly the same.

Thus, the objective functions that we seek to minimize are

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|}$$

$$Ncut(A_1, \ldots, A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)}$$

While the introduction of these balancing condition makes the mincut problem NP-hard, relaxing these conditions slightly leads Ncut and RatioCut to normalized and unnormalized spectral clustering respectively [10]

2.2.3 Perturbation theory point of view

Perturbation theory states that the eigenvalues and eigenvectors of a matrix A and a perturbed matrix $A + H$ is bounded by a constant times the norm of H. To apply this to spectral clustering, consider the ideal case where the graph is composed by k connected components. In this case, the first k eigenvalues will be 0 and corresponding eigenvectors will be the indicator vector of the cluster, where exactly one entry (corresponding to the cluster) will be 1 and rest will be 0. In this case, k-means will trivially converge to the indicator vectors and clustering will be ideal. With a small perturbation, the eigenvectors will also be slightly perturbed but k-means should still converge to the correct centroids. However, if the noise or perturbation is large, or the non-zero eigenvalues are very close to 0 or eigengap is very small, thus producing eigenvectors whose original values are very close to 0, the perturbation may be large enough for k-means to predictably converge to the correct centroids. The problem mentioned in the last section is not common for L or L_{rw} but for L_{sym} the eigenvectors are already left multiplied by $D^{1/2}$ and can cause trouble. Hence the third algorithm below [17] needs an extra normalization step.

2.3 Algorithms

Three classic spectral clustering algorithms can be found in literature.

All three algorithms essentially follow the same steps, using the first k eigenvectors of the graph Laplacian, create a matrix and then create k clusters from the rows of that matrix using k-Means algorithm. Finally, create the clusters of data points with the same indices as the indices of the matrix rows in the k clusters formed by k-means. For a proof of how the relaxation of the above balancing conditions to arrive at an approximation of NCut and RatioCut leads to the Normalized and Unnormalized Spectral Clustering respectively, see [10]

2.3.1 Unnormalized Spectral Clustering [15]

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
begin
Construct a similarity graph and its weighted adjacency matrix W based on similarity function
Compute the unnormalized Laplacian L
Compute the first k eigenvectors $u_1, \ldots, u_k$ of L
Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns
For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i^{th} row of U
Cluster the points $(y_i), i = 1, \ldots, n$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$
end
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{x_j | y_j \in C_i\}$

2.3.2 Normalized Spectral Clustering - Shi & Malik (2000)[16]

This algorithm uses generalized eigenvectors of L, which are the eigenvectors of normalized random-walk Laplacian L_{rw}

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct
begin
Construct a similarity graph and its weighted adjacency matrix W based on similarity function
Compute the unnormalized Laplacian L.
Compute the first k generalized eigenvectors $u_1, \ldots, u_k$ of generalized eigenproblem $Lu = \lambda Du$.
Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i^{th} row of U.
Cluster the points $(y_i), i = 1, \ldots, n$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.
end
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{x_j | y_j \in C_i\}$

2.3.3 Normalized Spectral Clustering - Ng, Jordan & Weiss (2002)[17]

This algorithm uses the eigenvectors of normalized symmetric Laplacian L_{sym}. Note that if $D^{1/2}u$ is an eigenvector of L_{sym} if u is an eigenvector of L hence an additional normalization step is needed.

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.
begin
Construct a similarity graph and its weighted adjacency matrix W based on similarity function
Compute the normalized Laplacian L_{sym}
Compute the first k eigenvectors $u_1, \ldots, u_k$ of L_{sym}
Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns
Form the matrix $T \in \mathbb{R}^{n \times k}$ from U by normalizing the norm to 1, $t_{ij} = u_{ij} / (\sum_k u_{ik}^2)^{1/2}$
For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i^{th} row of U
Cluster the points $(y_i), i = 1, \ldots, n$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$
end
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{x_j | y_j \in C_i\}$

2.3.4 Numerical Experiments

Using my implementation of the above algorithms, here are some results of my experiments

1. Mixture of 4 Gaussians in $\mathbb{R}^1$
   This data set consists of a random sample of 200 points drawn according to a mixture of four Gaussians. The similarity function used was a Gaussian kernel $e^{-|x_i - x_j|^2/(2\sigma^2)}$ with $\sigma = 1$. The distribution of points, pictorial representation of matrices W and D, and the final clustering results are shown in Figure 2.1

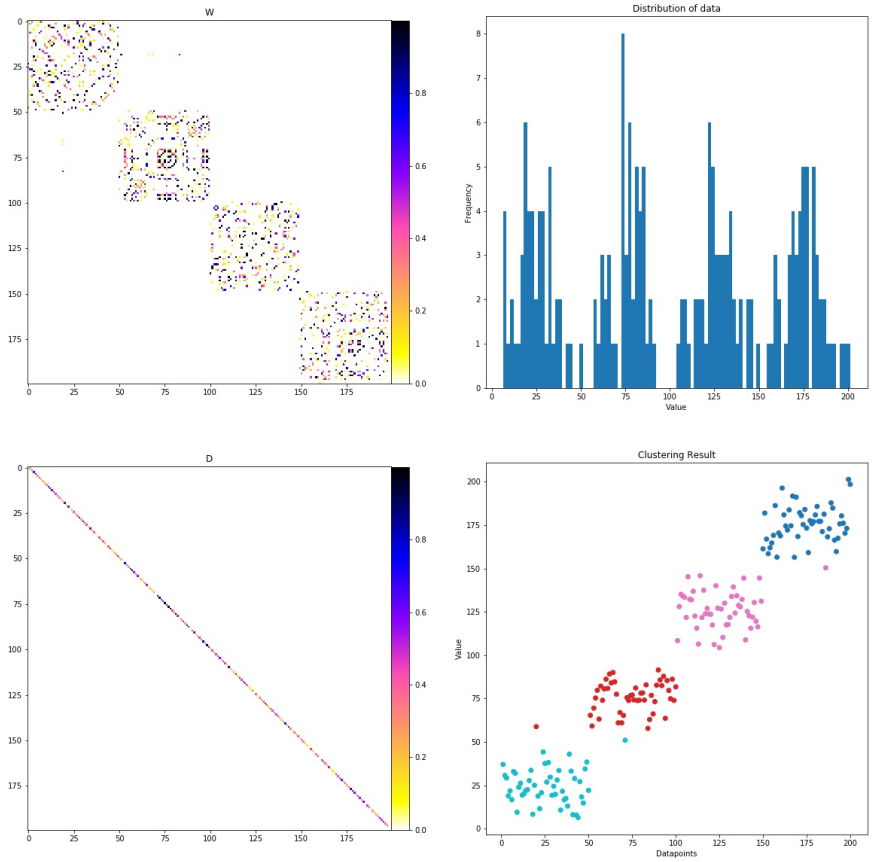


Figure 2.1: Clustering results for 200 points in $\mathbb{R}^1$

2. Points in $\mathbb{R}^2$
   This data set (the 'dim2' dataset from [18]) consists of 1351 points in $\mathbb{R}^2$ distributed in 9 clusters with one outlier. The figure below shows the clustering in 5, 8, and 10 clusters. This example shows that even though in theory, both Ncut and RatioCut try to keep the clusters reasonably large, that if a single outlier point would be clustered independently if it is sufficiently separated from the rest. The reason for this is that rest of the clusters are very tightly organized. The similarity function used in this case was scaled Euclidean distance. The clustering results into 5, 8, and 10 clusters is shown in Figure 2.2

3. Image Classification
   Image recognition is generally considered to be a supervised machine learning problem. A trained neural network can recognize and classify images with very high accuracy. However, this requires that the training set images need to be labeled. One such application arises in biology and medical

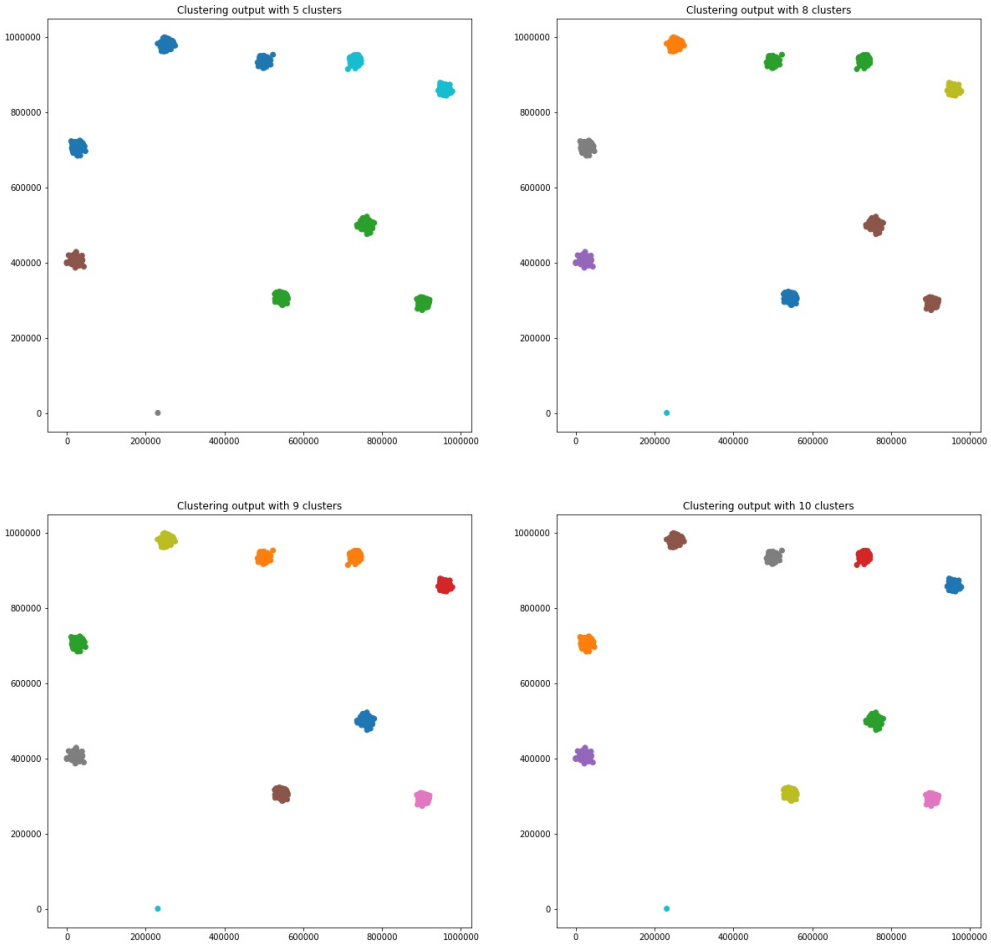


Figure 2.2: Clustering results for 1351 points in $\mathbb{R}^2$

imaging and manually labeling the training set of images may not be possible since trained personnel would be required for the purpose. One way to handle this type of situation is to cluster these images and then inspect a small sample of images from each cluster. If clustering was effective, the images in a cluster generally would fall under one category (in this case, the same digit) and the manual task gets reduced to labeling the cluster. To test this approach, I took 2000 images from the MNIST dataset. The MNIST dataset is a set of images of handwritten digits and each image is a 28×28-pixel single color image converted to an array of 784 (28×28) integers from 0 to 256 representing the intensity of the color. I took 2000 of the unlabeled images and used Normalized spectral clustering algorithm by Shi & Malik [16] to cluster the 2000 images.

Figure 2.3 shows the same dataset clustered into 10 clusters using three different similarity measures. The count of pixels that are non-zero in both images ($||\mathbf{v_i} \circ \mathbf{v_j}||_1$), reciprocal of 2-norm of the difference vector between two images ($||\mathbf{v_i} - \mathbf{v_j}||_2^{-1}$), and the count pixels that are zero or non-zero in both images were used as similarity measures respectively. 2.4 shows the weighted adjacency matrix using all the three similarity functions mentioned above. The weighted adjacency matrix, does not show clear clusters but the algorithm is able to cluster similar images to a certain extent. In all these three situations we see that some of the clusters cleanly group the images for the same

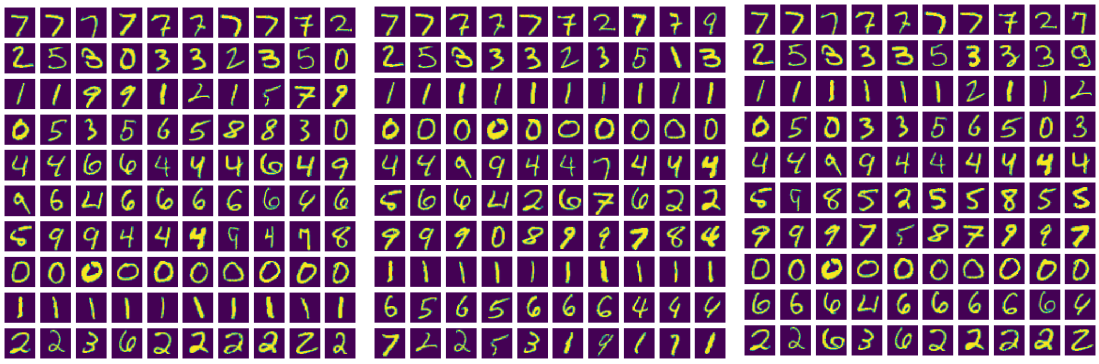
digit. However, some of the clusters are composed of a mixture of digits. Upon some inspection, it can be seen small changes in the handwriting style like the use of curves instead of straight lines for digits like 4 and 9 or use of the horizontal line for digit 7 cause these digits to be clustered with a different digit. The limitation here can be overcome with a more sophisticated similarity function that is robust to such changes, e.g. a function that counts the number of straight strokes, curves, and circles, this algorithm should be able to cluster the digits appropriately into 10 clusters.

Alternately, we can cluster the images into more clusters so that the algorithm will be able to create smaller and tighter clusters. Multiple clusters can be allocated to the same digits with different writing styles and strokes clustered into separate clusters. This reduces the number of images erroneously classified. Figure 2.5 shows the images clustered into 20 clusters shown as a row per cluster with the first 10 images in each cluster in 10 columns. As we can see, there is some noise in almost every cluster but overall, most clusters are pretty clean and only a few of the clusters need to be manually inspected.

Also, it can be noted that various similarity functions produce slightly different results in the case of 10 and 20 clusters.

The results above show the clustering with 10 and 20 clusters. However the optimal number of clusters can be bettween 10 and 20 or may even be beyond higher than 20. To find the optimum cluster count, one has to compute $Ncut$ for each run (since we use the algorithm by [16]) and findout the minima of $Ncut$

Finally, we note that the Unnormalized and the Symmetric Normalized Laplacian variants of the algorithms for this use case, usually generated worse results. In the next section, we will see how Ncut (which leads to Normalized spectral clustering with L_{rw}) uses the volume of the clusters in the denominator and, in our use case, is more likly to lead to balanced clusters compared to RatioCut.



(a) $||\mathbf{v_i} \circ \mathbf{v_j}||_1$ (b) $||\mathbf{v_i} - \mathbf{v_j}||_2^{-1}$ (c) Count of pixels that are zero or non-zero in both images

Figure 2.3: 2000 images from MNIST dataset were clustered into 10 clusters with similarity functions as shown below the clustering result. In the above figure, each row of images represent a cluster and show the first 10 images in that cluster

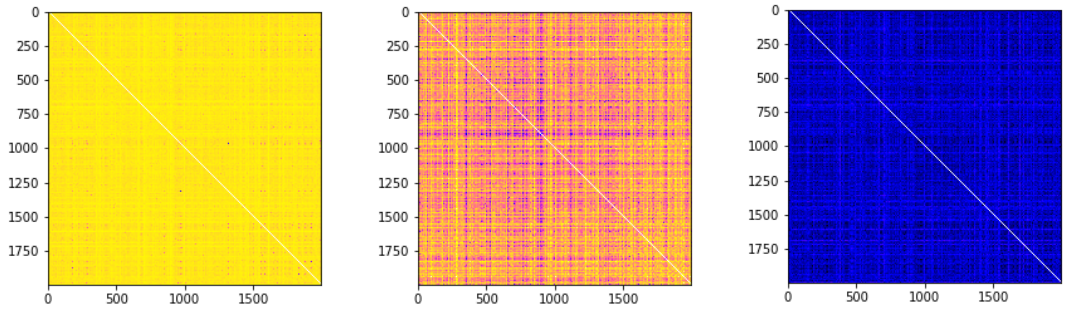
2.4 Practical considerations and challenges

While the algorithms are relatively simple to implement, there are some issues that arise in practice.

2.4.1 Parameters and Tuning Considerations

The first consideration is the choice of similarity function. The effectiveness of the algorithm depends upon the similarity matrix W which is defined by the similarity function. Although Gaussian similarity function $exp(-||x_I - x_j||^2)/(2\sigma^2)$ is generally a reasonable choice in Euclidean space, the choice of similarity function would generally be guided by the domain of application.

Next would be the choice of similarity graph and its parameters. The type of graph (k-nearest neighbor, ϵ-neighborhood, etc.) affects how the areas of different densities are treated. For example, a k-nearest



(a) $||\mathbf{v_i} \circ \mathbf{v_j}||_1$ (b) $||\mathbf{v_i} - \mathbf{v_j}||_2^{-1}$ (c) Count of pixels that are zero or non-zero in both images

Figure 2.4: Weighted adjecency matrix for 2000 images from MNIST dataset with similarity functions described under the images

neighbor graph may cause some points in a sparse cluster to be connected to a dense cluster or may break a high-density cluster into smaller clusters if k is not chosen appropriately. Also for ϵ neighborhood graph, the choice of ϵ dictates the sparsity of the matrix. While sparse matrix would be desirable since use of eigensolvers can speed up the eigenvector calculation, it can also lead to loss of information and hence a balance must be achieved and choice of the parameters should be guided by the domain of application.

Choice of Laplacian is the next concern. Recall that the goal of clustering is two-fold, to minimize inter-cluster similarity or $cut(A, \bar{A})$, and maximize within-cluster similarity or $W(A, A)$. Both Ncut and RatioCut will achieve the first objective since they both have $cut(A, \bar{A})$ in the numerator. However, note that $W(A, A) = W(A, V) - W(A, \bar{A}) = vol(A) - cut(A, \bar{A})$. This means that within-cluster similarity is maximized when $cut(A, \bar{A})$ is minimized and $vol(A)$ is maximized. This is where Ncut outperforms RatioCut since it has $vol(A)$ in the denominator. Hence L_{rw} is generally a good choice.

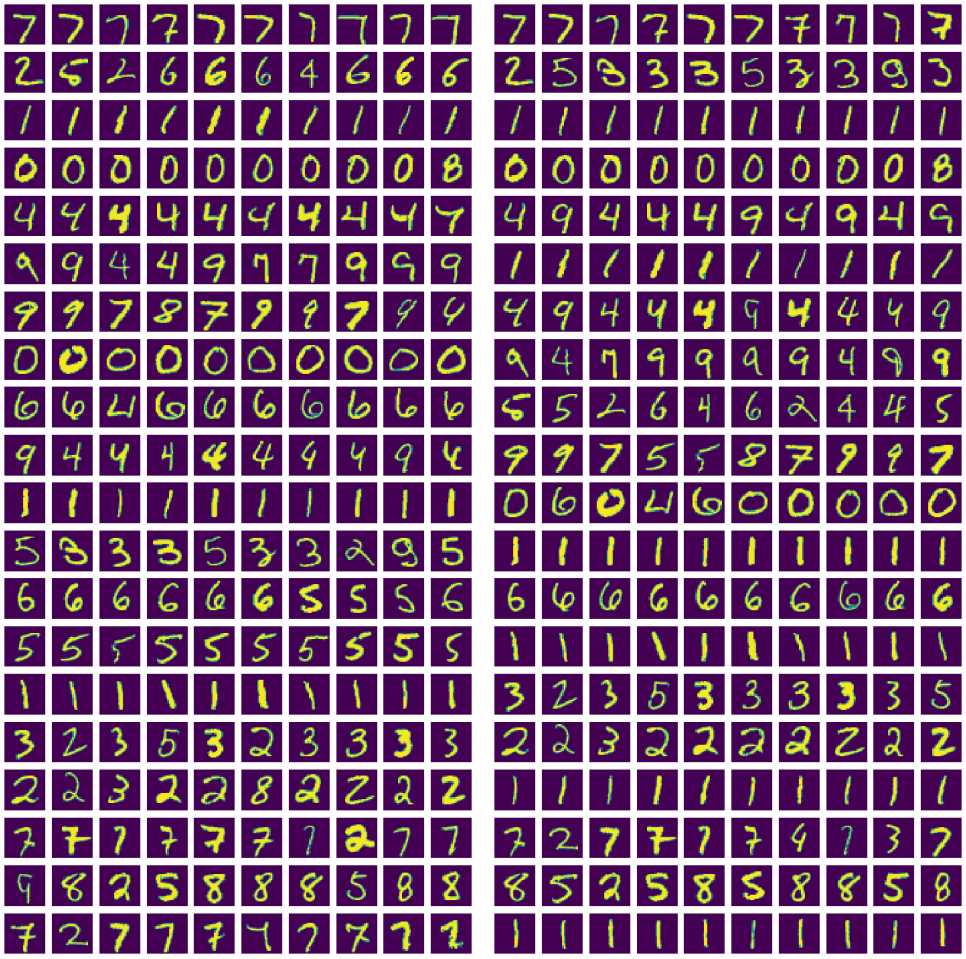
2.4.2 Challenges

The following are the challenges that frequently arise in practice.

Computing eigenvectors can be an expensive task, particularly if the matrix is large and dense. Fortunately a properly chosen graph and parameter(s) (e.g. k for k-means or ϵ for ϵ neighborhood), should lead to a sparse matrix, which, in turn, facilitates the choice of sparse eigensolvers and hence speed up the calculation. Additionally, the convergence speed depends upon spectral gap $\gamma_k = |\lambda_k - \lambda_{k+1}|$. Hence scaling and shifting or methods like MAPS described in [19] can be applied.

Next, challenge would be to choose the number of clusters. While the literature that describes the algorithms and their designs, usually takes the number of clusters, k, as input to the algorithms, frequently in practice, one needs to find the number of clusters that exist in a dataset. Since this is a general problem for all clustering algorithms, a variety of methods exist. Some methods use the ratio of within-cluster to inter-cluster similarity or distance as a measure of how good the clustering is, while other methods use a statistical calculation based on log-likelihood of data. Additionally, for spectral clustering, eigengap is an important heuristic. A good starting point is to choose k such that $\lambda_1, \ldots, \lambda_k$ are small but λ_{k+1} is relatively large. However, the effectiveness of this technique depends on how well the clusters are differentiated.

Finally, the k-means step itself can be expensive and in some cases, may not lead to ideal results based on the initial guess and/or the distribution of data points. Several attempts have been made to use other techniques of clustering. In [20], the authors show that k-means is efficient and minimizes the quantization error, thus highlight the reasons why k-means has become the choice for clustering applications.



(a) $||\mathbf{v_i} \circ \mathbf{v_j}||_1$ (b) $||\mathbf{v_i} - \mathbf{v_j}||_2^{-1}$

Figure 2.5: 2000 images from MNIST dataset were clustered into 20 clusters. In the above figure, each row of images represent a cluster and show the first 10 images in that cluster

2.5 Compressive Spectral Clustering

In Compressive Spectral Clustering [21], the authors propose an algorithm to sample $\mathcal{O}(\log(k))$ randomly filtered signals on the graph to serve as feature vectors instead of eigenvectors and by clustering random subset of $\mathcal{O}(k\log(k))$ nodes using random feature vectors and inferring the cluster label of all N nodes. This algorithm speeds up the clustering process by reducing the dimensionality and hence, the size of the problem.

As discussed above, for a graph with k connected components, 0 is the eigenvalue with multiplicity k and the corresponding eigenvectors would be the indicator vectors of the clusters, which would be orthogonal. Thus, the indicator vectors belong to $span\{U_k\}$. Perturbation theory says that for the general graph with slight perturbation to the ideal case, the first k eigenvectors still belong to $span\{U_k\}$. Thus it should be possible to sample a smaller number of features and still have k-means identify the k clusters correctly.

The algorithm for compressive spectral clustering is, however quite involved and is outlined briefly below.

- First, to approximate the first k eigenvectors,
  - Estimate the k^{th} eigenvalue using eigen-counting techniques detailed in [22]
  - Build polynomial estimation of the low-pass filter $H := h(L) = Uh(\Lambda)U^T$ where $h(\Lambda)$ is the diagonal matrix with 1 on primary diagonal for the first k rows/columns and 0 otherwise. U here is the matrix is the eigenvectors of L. However, L is not diagonalized, instead Hx is

approximated as $\sum_{l=0}^{p} \alpha_l L^l x$ by successive matrix-vector products with L.

- Generate d random Gaussian signals $r_1, \ldots, r_d$ with mean 0 and variance $1/d$, Let $S = (r_1|r_2|\ldots|r_d) \in \mathbb{R}^{N \times d}$
- Filter S with H and define for each node i, its feature vector $\bar{f}_i = \left[(HS)^T \delta_i\right] / ||(HS)^T \delta_i|| \in \mathbb{R}^d$
- Generate random sampling matrix $M \in \mathbb{R}^{n \times N}$ to keep $n = 2k \log k$ features.
- Run k-means on the reduced dataset with Euclidean vector distance to obtain k reduced indicator vectors, one for each cluster.
- Interpolate each reduced indicator vector to find the cluster assignment for each node on the graph.

Chapter 3

Future Work

In practice quite frequently the number of clusters is not known. Hence a clustering method is repeatedly applied with an increasing number of clusters until the desired accuracy is reached. With spectral clustering, since the number of clusters depends on the number of eigenvectors used, repeated computation of eigenvectors have to be performed.

This problem can be alleviated with a faster eigensolver. Also, if the information from a prior step can be used in a subseuent pass, using a method that allows hot-restart the repetitive work can be avoided and the we can quickly iterate of various cluster configurations. Based on prior work like [23], [24] and [25] Chebeshev Filters and Chebyshev-Davidson method look promising for future research.

Another possibility is to use dimensionality reduction methods, even if they may not fall strictly into spectral clustering methods. However, there are methods like Non-Negatve Matrix Factorization that try to extrat lower dimensional features that help in clustering. In spectral-clusterin methods, lower dimension would means faster k.means and eventually, quickly clustering.

Appendix A

Propositions related to Graph Laplacians

Proposition 1. *(Properties of L) The matrix L satisfies:*

1. *for every 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 semi-definite*
3. *The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant vector* $\mathbf{1}$
4. *L has n non-negative, real-values eigenvalues* $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

Proof:

1. By definition of $d_j = \sum_{i=1}^{n} w_{ij}$

$$\begin{aligned} f'Lf = f'Df - f'Wf &= \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij} \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{i=j}^{n} d_j f_j^2 \right) \\ &= \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \end{aligned}$$

2. Since W and D are symmetric $L = D - W$ is still symmetric. By result of part (a), since $w_{ij} \geq 0$ and $(f_i - f_j)^2 \geq 0$, $f'Lf \geq 0$, hence L is symmetric and positive semi-definite
3. See proposition 2
4. Since L is symmetric, all its eigenvalues are real. Also by part (c), the smallest eigenvalue is 0,

Note that the unnormalized graph Laplacian does not depend on the diagonal elements of W. $d_{ii} = \sum_{j=1}^{n} w_{ij}$, hence the diagonal elements are canceled by d_{ii} and the self-edges do not change the Laplacian.

Proposition 2. *(Number of connected components and the spectrum of L) Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue* 0 *of L equals the number of connected components* $A_1, \cdots, A_k$ *in the graph. The eigenspace of eigenvalue* 0 *is spanned by the indicator vectors* $\mathbf{1}_{A_1}, \cdots, \mathbf{1}_{A_k}$ *of those components.*

Proof: Starting with $k = 1$, i.e. graph is fully connected. Assume that f is the eigenvector with eigenvalue 0. Then,

$$0 = f'Lf = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \implies f_i - f_j = 0 \quad [\because w_{ij} > 0]$$

Since we assumed a fully connected graph, f has to be constant for the above argument to be true for any vertex in the connected component. Hence the vector $\mathbf{1}$, which is the indicator vector of the component is an eigenvector corresponding to eigenvalue 0.

Now, for $k > 1$, we can easily rearrange the vertices in the matrix to put all the vertices of a connected component together. This would cause the matrix L to be a block diagonal matrix. For each of these blocks, or components, the respective indicator vector is an eigenvector with eigenvalue 0. Thus the Matrix L has eigenvalue 0 with multiplicity equal to the number of components and the corresponding eigenvectors would be the indicator vectors of those components.

Proposition 3. *(Properties of* L_{sym} *and* L_{rw}*)*

1. *For every* $f \in \mathbb{R}^n$

$$f'L_{sym}f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$$

2. *λ is an eigenvalue of L_{rw} with eigenvector u iff λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}u$.*

3. *λ is an eigenvalue of L_{rw} with eigenvector u iff λ and u solve the generalized eigenproblem $Lu = \lambda Du$*

4. *0 is an eigenvalue of L_{rw} with the constant one vector $\mathbf{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbf{1}$*

5. *L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$*

Proof:

1. This can be proved similar to part 1 of proposition 1.

2. Assume λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2}u$.

$$\begin{aligned} & L_{sym}D^{1/2}u = \lambda D^{1/2}u \\ \implies & D^{-1/2}LD^{-1/2}D^{1/2}u = \lambda D^{1/2}u \\ \implies & D^{-1}Lu = \lambda u \quad [\text{left multiplying with } D^{-1/2}] \\ \implies & L_{rw}u = \lambda u \end{aligned}$$

   In other words, u is eigenvector of L_{rw}. To prove the other direction, apply the above steps in reverse order.

3. This can be proven in similar manner to above by left multiplying the generalized eigenproblem $Lu = \lambda Du$ with D^{-1}

4. $L_{rw}\mathbf{1} = (I - D^{-1}W)\mathbf{1} = \mathbf{1} - \mathbf{1} = 0$.
   The above is true because $d_i = \sum_{j=1}^{n} wij \implies \left(D^{-1}W)\mathbf{1}\right)_i = \sum_{j=1}^{n} \frac{w_{ij}}{d_i}$, since $\mathbf{1}$ is the indicator vector. Then using part (b) leads to the second statement of this property.

5. The first part (for L_{sum}) can be proved from part (a), similar to Proposition 1. Part 2 essentially states that L_{sym} and L_{rw} have the same eigenvalues, hence second statement is also proven.

Proposition 4. *(Number of connected components and spectra of L_{sym} and L_{rw}) Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of both L_{sym} and L_{rw} equals to the number of connected components $A_1, \cdots, A_k$ in the graph. for L_{rw}, the eigenspace of 0 is spanned by the indicator vectors $\mathbf{1}_{\mathbf{A_i}}$ of these components. for L_{sym} the eigenspace of 0 is spanned by $D^{1/2}\mathbf{1}_{\mathbf{A_i}}$*

Proof: The proof of this is analogous to that of proposition 2, but using proposition 3.

Bibliography

[1] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[2] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[3] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA, 1996. Association for Computing Machinery.

[4] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[6] William H. Lawton and Edward A. Sylvestre. Self modeling curve resolution. *Technometrics*, 13(3):617–633, 1971.

[7] Pentti Paatero, Unto Tapper, Pasi Aalto, and Markku Kulmala. Matrix factorization methods for analysing diffusion battery data. *Journal of Aerosol Science*, 22:S273–S276, 1991.

[8] Chris Ding, Xiaofeng He, and Horst D. Simon. *On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering*, pages 606–610. Proceedings of the 2005 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics, April 21, 2005.

[9] Nian Zhang, Keenan Leatham, Jiang Xiong, and Jing Zhong. Pca-k-means based clustering algorithm for high dimensional and overlapping spectra signals. In *2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 349–354, 2018.

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

[11] B Mohar, Y Alavi, G Chartrand, Ortrud Oellermann, and Allen Schwenk. The laplacian spectrum of graphs. *Graph Theory, Combinatorics and Applications*, 2:871–898, 01 1991.

[12] Bojan Mohar. *Some applications of Laplace eigenvalues of graphs*, pages 225–275. Springer Netherlands, Dordrecht, 1997.

[13] F. R. K. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[14] Marina Meilă and Jianbo Shi. A random walks view of spectral segmentation. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, volume R3 of *Proceedings of Machine Learning Research*, pages 203–208. PMLR, 04–07 Jan 2001.

[15] L. Hagen and A.B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9), 1992.

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

[17] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Proc. NIPS*, pages 849–856, 2001.

[18] I. Kärkkäinen and P. Fränti. Gradual model generator for single-pass clustering. *Pattern Recognition*, 40(3):784–795, 2007.

[19] Songtao Lu and Zhengdao Wang. Accelerated algorithms for eigenvalue decomposition with application to spectral clustering. *2015 49th Asilomar Conference on Signals, Systems and Computers*, pages 355–359, 2015.

[20] Léon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*, pages 585–592. MIT Press, 1995.

[21] Nicolas Tremblay, Gilles Puy, Remi Gribonval, and Pierre Vandergheynst. Compressive spectral clustering, arXiv:1602.02018, 2016.

[22] Edoardo Di Napoli, Eric Polizzi, and Yousef Saad. Efficient estimation of eigenvalue counts in an interval. *CoRR*, abs/1308.4275, 2013.

[23] Y. Zhou and Y. Saad. A Chebyshev-Davidson algorithm for large symmetric eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 29(3):954–971, 2007.

[24] Y. Zhou. A block Chebyshev-Davidson method with inner-outer restart for large eigenvalue problems. *Journal of Computational Physics*, 229(24):9188–9200, 2010.

[25] Z. Wang. *Filtered Davidson-type Methods for Large-scale Eigen-related Problems*. PhD thesis, Dept. Mathematics, Southern Methodist University, Dallas TX, USA, August 2015.

[26] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[27] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Number 20 in Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1998.

[28] G. W. Stewart and J. G. Sun. *Matrix perturbation theory*. Academic Press, Boston, MA, 1990.

[29] G. Cheung, E. Magli, Y. Tanaka, and M. K. Ng. Graph spectral image processing. *Proceedings of the IEEE*, 106(5):907–930, 2018.