Lecture 11

IDS575: Statistical Models and Methods
Theja Tulabandhula

Notes derived from the book 'Elements of Statistical Learning' [2nd Ed.] (Sections 14.3,14.5-14.7)

We will discuss two unsupervised learning methods: clustering and principal components.

1 Unsupervised Learning with Clustering

The idea of clustering is to segment observations into separate sets such that the observations in the same set are closer to each other according to a metric/measure compared to observations in other sets.

Note 1. We do not necessarily have to work with observations. Even information about how similar each observation is to other observations is enough to cluster them.

Thus, a notion of *dissimilarity* (or similarity) is key to cluster analysis. This is typically defined by an expert using domain knowledge, and is not data driven.

1.1 Defining Dissimilarity

Dissimilarity can be represented using a $N \times N$ matrix $\mathbf{D}$, where N is the number of observations (e.g., x_i for $i = 1, ..., N$). Each entry of this matrix $d_{ii'}$ is the similarity between observation i and observation i'.

Note 2. If we have the similarity value between two observations, we can convert it to a value dissimilarity by using a function such as $1 - z$. For example, if similarity is $0 \le s \le 1$, then dissimilarity can be defined as $1 - s$.

One way to define dissimilarity is to define it coordinate wise. That is, dissimilarity

$$D(x_i, x_i') = d_{ii'} = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j}),$$

where $d_j(x_{ij}, x_{i'j})$ is the coordinate wise dissimilarity[1].

Example 1. For example, $d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$.

Example 2. Define correlation between *standardized* observations x_i and $x_{i'}$ as:

$$\rho = \frac{\sum_{j=1}^{p} x_{ij} x_{i'j}}{\sqrt{\sum_{j=1}^{p} x_{ij}^2 \sum_{j=1}^{p} x_{i'j}^2}}.$$

This correlation (similarity/dissimilarity) information can be used to cluster. In fact, this is equivalent to defining clustering using squared distance because:

$$\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \propto 2(1-\rho).$$

Example 3. For categorical variables, you can define $d_j(x_{ij}, x_{i'j})$ using exogenous information about how one category is different from another.

Specifying $d_{ii'}$ is generally very important compared to the clustering method of choice.

1.2 Algorithms for Clustering

There are two main ways to cluster:

- optimization formulations (there is no statistical or probability distribution)
- mixture models (like Gaussian mixtures)

Since we have looked at mixture models before, lets investigate the optimization formulation route. Here, we want to find the cluster label of each observation in our dataset given the dissimilarities matrix $\mathbf{D}$. This cluster labeling should minimize some loss.

Example 4. An example loss function is $W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'})$, where we are optimizing for cluster labels given by $C(i)$ for $i = 1, ..., N$. This loss function is high if observations that are not close in similarity are assigned to the same cluster.

Note 3. Optimizing the cluster assignments over a loss function such as the above is typically a difficult problem. Enumeration is not possible for even small datasets. Hence iterative greedy methods are our only computationally tractable choice.

[1] One can question why dissimilarity in each attribute should be added equally. There is no right answer here.

In particular, lets look at the *k-means* clustering method. Assume that all X_j are quantitative. Let $d_{ii'} = \|x_i - x_{i'}\|_2^2$. Then, $W(C)$ can be written as:

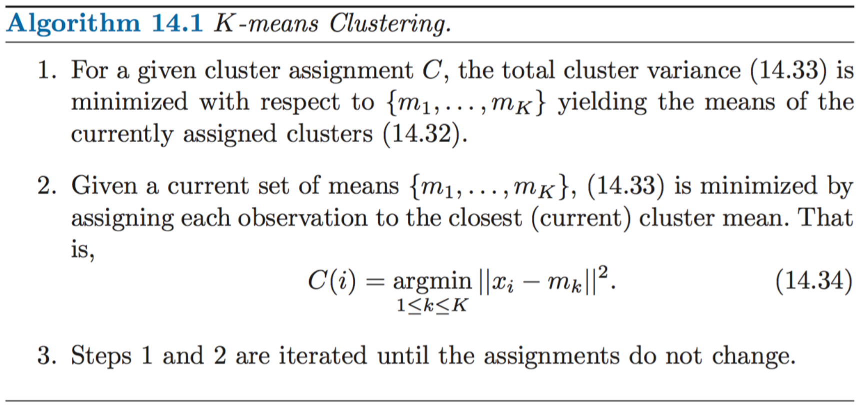
$$W(C) = \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|x_i - \bar{x}_k\|_2^2.$$

Here, $\bar{x}_k$ is the mean of the k^{th} cluster and N_k is the number of observations in the k^{th} cluster.

By noting that $\bar{x}_S = \arg\min_m \sum_{i \in S} \|x_i - m\|_2^2$, we can solve for cluster assignments by solving the following problem:

$$\min_{C, \{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|x_i - m_k\|_2^2.$$

The above can be minimized by alternative over C and the $\{m_k\}$ variables, giving us the Algorithm shown in Figure 1.



Algorithm 14.1 *K-means Clustering.*

1. For a given cluster assignment C, the total cluster variance (14.33) is minimized with respect to $\{m_1, \dots, m_K\}$ yielding the means of the currently assigned clusters (14.32).

2. Given a current set of means $\{m_1, \dots, m_K\}$, (14.33) is minimized by assigning each observation to the closest (current) cluster mean. That is,

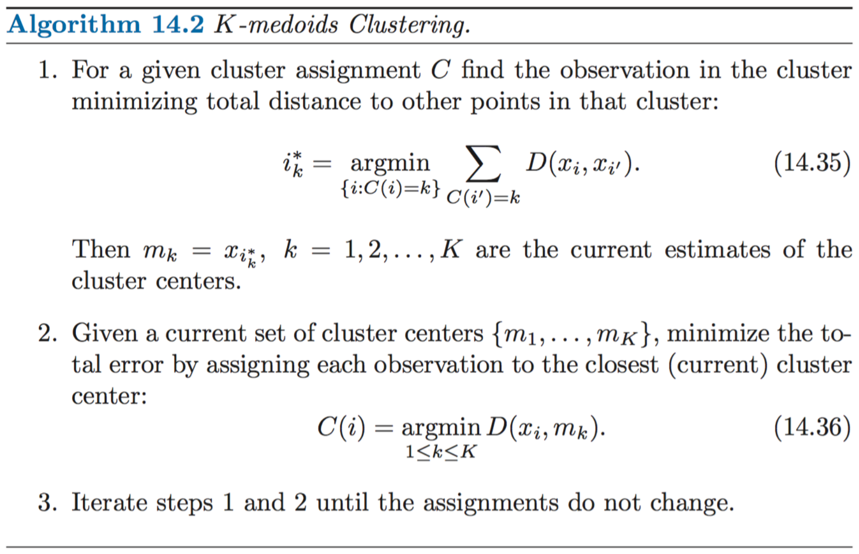
$$C(i) = \underset{1 \le k \le K}{\text{argmin}} \, ||x_i - m_k||^2. \qquad (14.34)$$

3. Steps 1 and 2 are iterated until the assignments do not change.

Figure 1: The k-means clustering method.

The EM algorithm for Gaussian mixture model is closely related to the k-means algorithm. The E-step finds soft assignments (recall the $q(|x)$ conditional probability values) of observations to mixture components. The M-step finds the mixture parameters given the soft assignments. If we assume K mixture components with covariances $\sigma^2\mathbf{I}$, then given mixture soft assignments, the likelihood of data is proportional to Euclidean distance to the mixture center. This is very similar to the k-means method above.

A variation of k-means, called k-medoids, extends it to non-Euclidean dissimilarity measures. The Algorithm is given in Figure 2. Other than replacing the squared dissimilarity measure, an additional restriction imposed here is that each cluster center has to be one of the observation itself.



Algorithm 14.2 *K-medoids Clustering.*

1. For a given cluster assignment C find the observation in the cluster minimizing total distance to other points in that cluster:

$$i_k^* = \underset{\{i:C(i)=k\}}{\operatorname{argmin}} \sum_{C(i')=k} D(x_i, x_{i'}). \tag{14.35}$$

   Then $m_k = x_{i_k^*}$, $k = 1, 2, \ldots, K$ are the current estimates of the cluster centers.

2. Given a current set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} D(x_i, m_k). \tag{14.36}$$

3. Iterate steps 1 and 2 until the assignments do not change.

Figure 2: The k-medoid clustering method.

Note 4. The choice of K: Larger K implies a lower objective value. Hence, as a heuristic, we can search for that K for which the decrease in the objective drastically changes from large values to small values. This is known as identifying a *kink* in the objective versus K plot.

Note 5. One issue with k-means and k-medoids is that empirically, the clusters may change drastically as K is changed. This does not happen for a different clustering method called *hierarchical clustering.*

Hierarchical clustering requires user to specify a dissimilarity measure between groups of observations.

It produces a hierarchical representation of clusters, where clusters at one level are obtained by splitting the clusters one level above (*divisive*) or merging the clusters from one level below (*agglomerative*).

Such cluster representations can be represented as *trees* with nodes representing clusters. Sometimes these hierarchical clusters have a monotonicity property: dissimilarity between merged clusters is monotonically increasing with the level of the merger. In terms of plot, the height of a node can be proportional to the dissimilarity of its children nodes. This is called a *dendrogram.*

Note 6. Hierarchical clustering will impose a hierarchical representation whether such pattern exists in the dataset or not.

For agglomerative clustering, the following are some measures for dissimilarity between groups:

- Single Linkage (SL): $d_{SL}(G,H) = \min_{i\in G, i'\in H} d_{ii'}$.
- Complete Linkage (CL): $d_{CL}(G,H) = \max_{i\in G, i'\in H} d_{ii'}$.
- Group Average (GA): $d_{GA}(G,H) = \frac{1}{N_G N_H} \sum_{i\in G} \sum_{i'\in H} d_{ii'}$.

Note 7. SL has a tendency to combine observations linked by a series of close intermediate observations, and this phenomena is called *chaining*.

2 Principal Components

Recall that we used principal components to understand ridge regression. So, what are these?

Intuitively, they are projections (linear approximations) of data ($x_i \in \mathbb{R}^p$ for $i = 1, .., N$) that are uncorrelated with each other and capture data variance in order.

For simplicity subtract from each observation a constant vector defined as $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$.

Say we want a rank $q < p$ linear approximation to this data. This means, that we want for each vector x_i an equivalent q-dimensional vector λ_i such that a linear transformation of x_i is close to λ_i for all i. Let $\mathbf{V}_q$ be a $p \times q$-dimensional matrix with q orthogonal unit column vectors.

It turns out that $\mathbf{V}_q \mathbf{V}_q^T$ is called a projection matrix that maps x_i to a rank q reconstruction of itself.

The search for $\mathbf{V}_q$ ($\lambda_i = \mathbf{V}_q^T x_i$) is done by minimizing reconstruction error:

$$\min_{V_q} \sum_{i=1}^{N} \|x_i - \mathbf{V}_q \mathbf{V}_q^T x_i\|_2^2.$$

The matrix $\mathbf{V}_q \mathbf{V}_q^T$ projects x_i onto the columnspace of $\mathbf{V}_q$, which is of rank q.

The solution to the above problem is given by taking the SVD of $\mathbf{X}$ ($N \times p$-dimensional), i.e., $X = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T$[2]and set $\mathbf{V}_q$ to be equal to the first q columns of $\mathbf{V}$.

[2]Assume that the entries of the diagonal matrix D are ordered $|d_1| \geq |d_2| \geq ....$

The columns of $\mathbf{U}\,\mathbf{D}$ are called the principal components of $\mathbf{X}$.

The N λ_is are given by the first q principal components (the first q columns of $\mathbf{U}\,\mathbf{D}$).

Example 5. One and two dimensional principal components are shown in Figures 3 and 4 respectively.

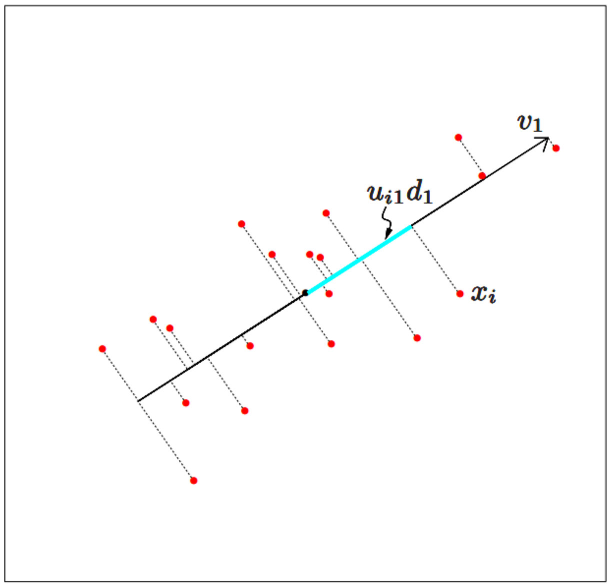


Figure 3: The 1-dimensional principal components of an example 2-dimensional dataset.

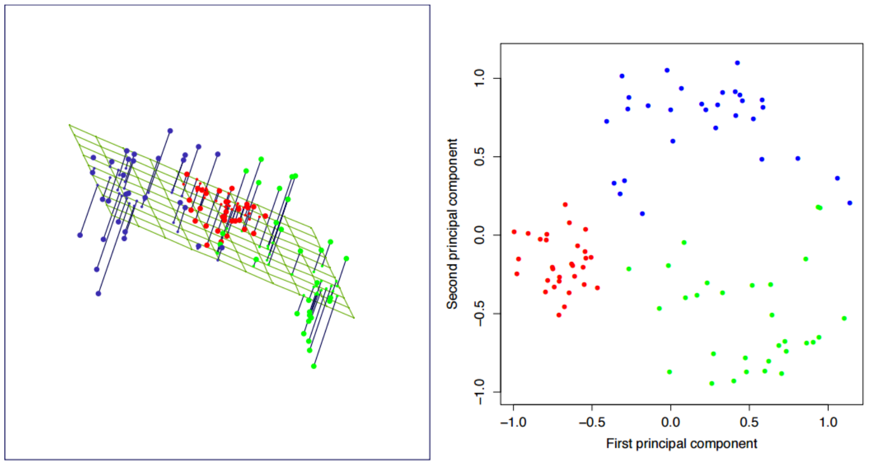


Figure 4: The 2-dimensional principal components (right) of an example dataset.

There is an interesting property that principal components capture with regard to *variance* in the data. For example, the N-dimensional vector (linear combination) $\mathbf{X}\,v_1$ has the highest variance among all linear combinations of columns of $\mathbf{X}$. Similarly $\mathbf{X}\,v_2$ has the next highest variance.

Example 6. They capture low dimensional aspects of the data well. For example, consider the 16-dimensional images of the number '3' in Figure 5.

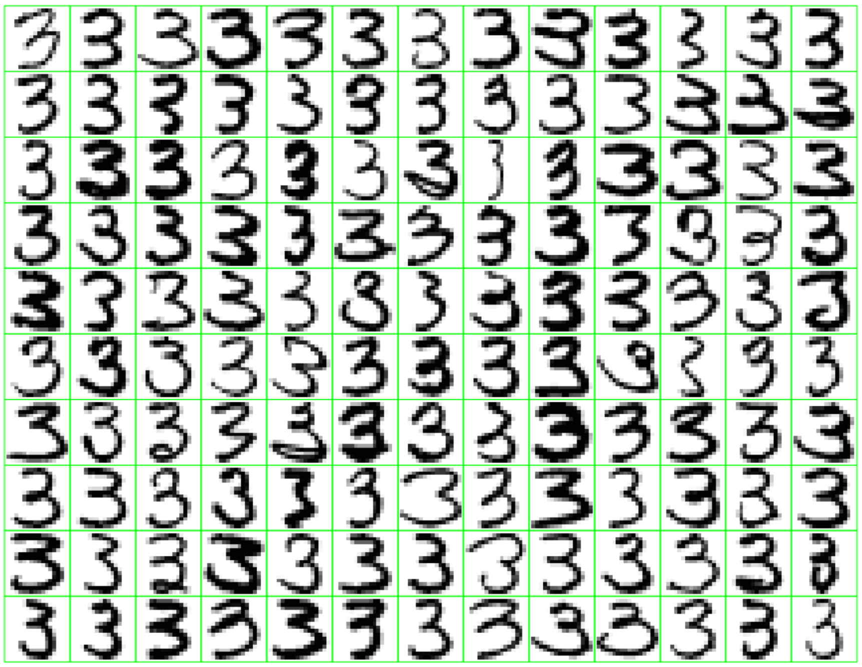


Figure 5: The handwritten '3' dataset.

Note 8. *Kernel PCA* computes principal components with the kernel matrix and is more flexible than using the gram matrix ($\mathbf{X}^T\,\mathbf{X} = \mathbf{U}\,\mathbf{D}^2\,\mathbf{U}^T$).

2.1 Spectral Clustering

The idea here to cluster, taking into account a different metric to define dissimilarity/similarity.

Note 9. The name *spectral* represents the use of eigenvalues of matrices (such as those based on the dissimilarity matrix).

Example 7. For example, it is difficult to cluster the points shown in Figure **??** (top left) using k-means.

Say, we have N observations and let $d_{ii'}$ be the Euclidean distance between observation $x_i \in \mathbb{R}^p$ and observation $x_{i'}$. We will convert the dissimilarity to a similarity metric as: $s_{ii'} = \exp(-d_{ii'}/c)$ for some $c > 0$.

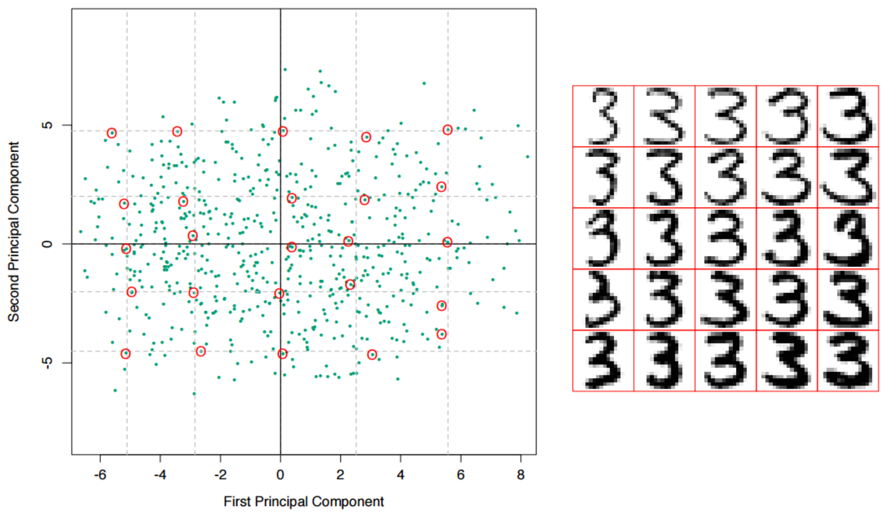


Figure 6: First two principal components and representative observations (corresponding to red circles on the left plot). The first component seems to encourage longer lower tail and the second encourages character thickness.

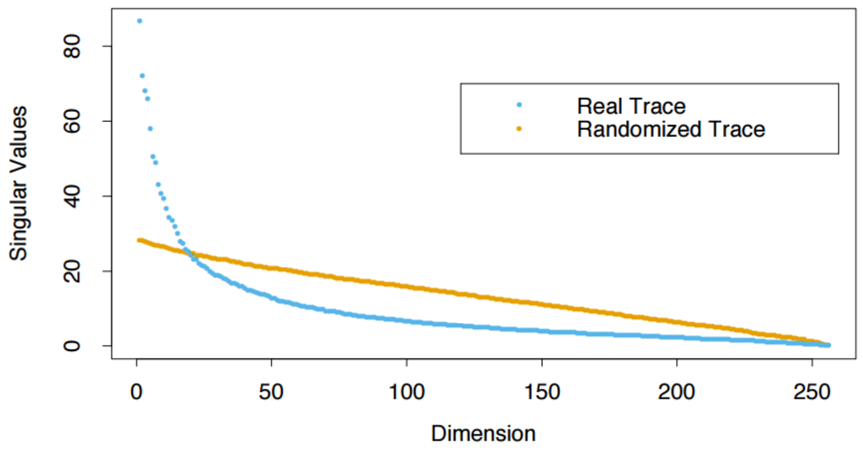


Figure 7: Evidence that principal components capture correlation. If you randomly scramble ech column of $\mathbf{X}$, less variance is captured in the lower principal directions.

We create a graph with observation indices as nodes and edge weights $s_{ii'}$. With this, we phrase a graph partitioning problem: break the graph into pieces such that edges between different clusters have low weight.

We can optionally do some preprocessing on the similarity matrix (such as setting low values to 0). Let this processed matrix be $\mathbf{W}$ (we will call it the *weighted adjacency matrix*). Spectral clustering has the following steps:

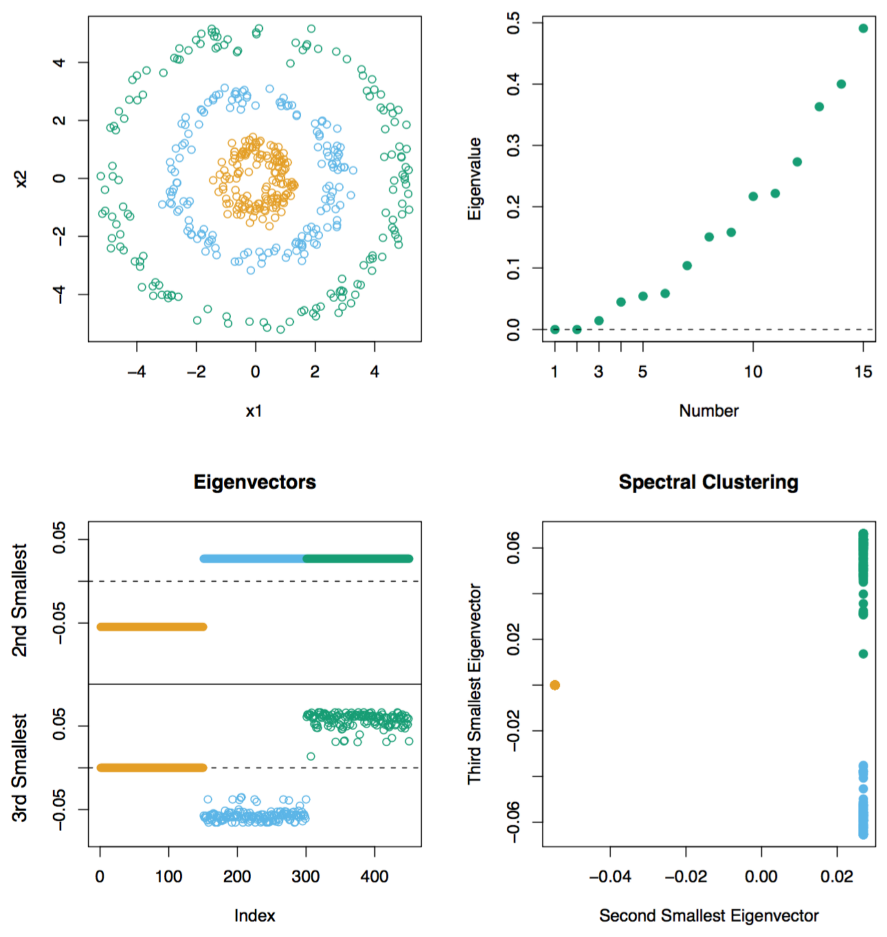


Figure 8: Spectral clustering example containing three concentric circular clusters with 150 points each.

- Compute $\mathbf{G}$ as a diagonal matrix with weighted degree value $g_i = \sum_{i'} w_{ii'}$ for each i.
- Compute the *un-normalized graph Laplacian* as $\mathbf{L} = \mathbf{G} - \mathbf{W}$.
- Find eigenvectors the $\mathbf{Z}$ ($N \times m$ dimensional) corresponding to the m smallest eigenvalues excluding 0.
- Use k-means to cluster the rows of $\mathbf{Z}$ to get a clustering of the original dataset.

Example 8. See Figure 8 for a visualization of how the smallest eigenvectors capture cluster information. The nodes of the graph were only connected to 10 nearest neighbors.

Why does this work? For any vector $\mathbf{f}$, $\mathbf{f}\,\mathbf{L}\,\mathbf{f} = \sum^N g_i f_i^2 - \sum_i^N \sum_{i'}^N f_i f_{i'} w_{ii'} = \frac{1}{2} \sum_i^N \sum_{i'}^N w_{ii'}(f_i - f_{i'})^2$. This takes small values for those vectors whose coordinates have close values if $w_{ii'}$ is large. Eigenvectors can be defined using this product as well. It turns out that if there is a single connected component, then $\mathbf{1}$ vector corresponds to the unique 0 eigenvalue. If there are more than one connected components, then there will be multiple such *indicator* eigenvectors corresponding to the 0 eigenvalue. In reality, this leads to the heuristic of finding smallest eigenvalues.

3 Summary

We learned the following things:

- Clustering methods: k-means and k-medoid.
- Principal components.

A Sample Exam Questions

1. What is the relation between k-means clustering and the EM algorithm for Gaussian mixture models?
2. Why do we cluster the left singular vectors corresponding to large singular values (in principal components) compared to clustering eigenvectors corresponding to small eigenvalues (in spectral clustering)?