Clustering Part II:
k-means and Related Methods

Let's begin with a few examples.

Example 1 *Figures 1 and 2 shows some synthetic examples where the clusters are meant to be intuitively clear. In Figure 1 there are two blob-like clusters. Identifying clusters like this is easy. Figure 2 shows four clusters: a blob, two rings and a half ring. Identifying clusters with unusual shapes like this is not quite as easy. To the human eye, these certainly look like clusters. But what makes them clusters? In this chapter we will make the definition of a cluster precise.*

Example 2 (Gene Clustering) *In genomic studies, it is common to measure the expression levels of d genes on n people using microarrays (or gene chips). The data (after much simplification) can be represented as an $n \times d$ matrix X where X_{ij} is the expression level of gene j for subject i. Typically d is much larger than n. For example, we might have $d \approx 5{,}000$ and $n \approx 50$. Clustering can be done on genes or subjects. To find groups of similar people, regard each row as a data vector so we have n vectors $X_1, \ldots, X_n$ each of length d. Clustering can then be used to place the subjects into similar groups.*

Example 3 (Curve Clustering) *Sometimes the data consist of a set of curves $f_1, \ldots, f_n$ and the goal is to cluster similarly shaped clusters together. For example, Figure 3 shows a small sample of curves a from a dataset of 472 curves from Frappart (2003). Each curve is a radar waveform from the Topex/Poseidon satellite which used to map the surface topography of the oceans.*[1] *One question is whether the 472 curves can be put into groups of similar shape.*

Example 4 (Supernova Clustering) *Figure 4 shows another example of curve clustering. The data are described in detail in the case studies in the chapter on linear classification. Briefly, each data point is a light curve, essentially brightness versus time. The top two plots show the light curves for two types of supernovae called "Type Ia" and "other." The bottom two plots show what happens if we throw away the labels ("Type Ia" and "other") and apply a clustering algorithm (k-means clustering). We see that the clustering algorithm almost completely recovers the two types of supernovae.*

[1] See http://topex-www.jpl.nasa.gov/overview/overview.html. The data are available at "Working Group on Functional and Operator-based Statistics" a web site run by Frederic Ferrarty and Philippe Vieu. The address is http://www.math.univ-toulouse.fr/staph/npfda/. See also `http://podaac.jpl.nasa.gov/DATA_CATALOG/topexPoseidoninfo.html`.

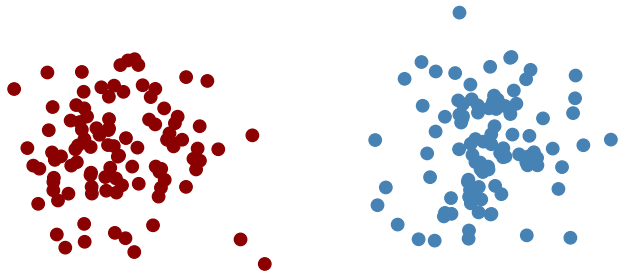

Figure 1: A synthetic example with two "blob-like" clusters.

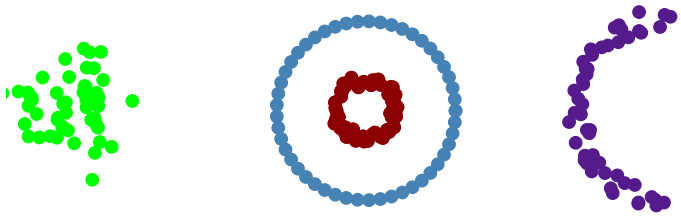

Figure 2: A synthetic example with four clusters with a variety of different shapes.

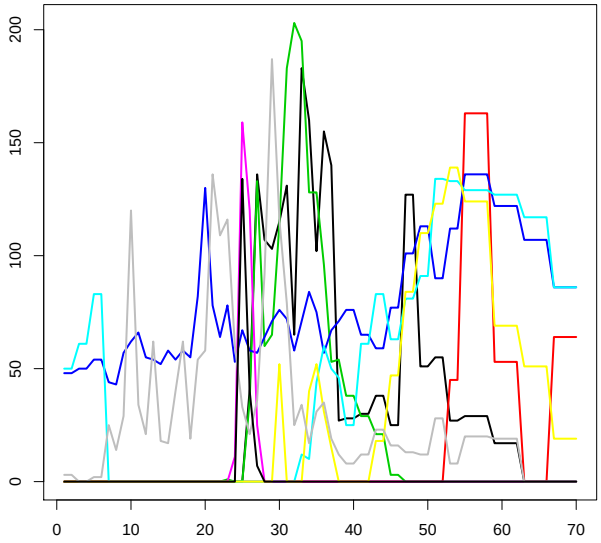

Figure 3: Some curves from a dataset of 472 curves. Each curve is a radar waveform from the Topex/Poseidon satellite.

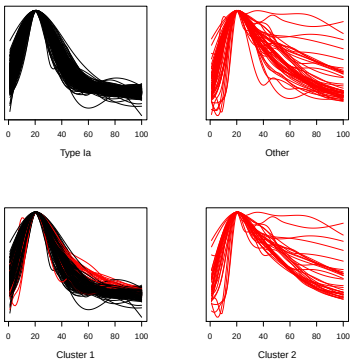

Figure 4: Light curves for supernovae. The top two plots show the light curves for two types of supernovae. The bottom two plots show the results of clustering the curves into two groups, without using knowledge of their labels.

1 k-means (Vector Quantization)

One approach to clustering is to find k representative points, called prototypes or cluster centers, and then divide the data into groups based on which prototype they are closest to. The set of prototypes is called a codebook. For now, we asume that k is given. Later we discuss how to choose k.

Let $X_1,\ldots,X_n \sim P$ where $X_i \in \mathbb{R}^d$. Let $C = \{c_1,\ldots,c_k\}$ where each $c_j \in \mathbb{R}^d$. We call C a codebook. Let $\Pi_C[X]$ be the projection of X onto C:

$$\Pi_C[X] = \text{argmin}_{c\in C}||c - X||^2. \tag{1}$$

Define the empirical clustering risk of a codebook C by

$$R_n(C) = \frac{1}{n}\sum_{i=1}^{n}\Big|\Big|X_i - \Pi_C[X_i]\Big|\Big|^2 = \frac{1}{n}\sum_{i=1}^{n}\min_{1\le j\le k}||X_i - c_j||^2. \tag{2}$$

Let $\mathscr{C}_k$ denote all codebooks of length k. The optimal codebook $\widehat{C} = \{\widehat{c}_1,\ldots,\widehat{c}_k\} \in \mathscr{C}_k$ minimizes $R_n(C)$:

$$\widehat{C} = \text{argmin}_{C\in\mathscr{C}_k} R_n(C). \tag{3}$$

The empirical risk is an estimate of the population clustering risk defined by

$$R(C) = \mathbb{E}\Big|\Big|X - \Pi_C[X]\Big|\Big|^2 = \mathbb{E}\min_{1\le j\le k}||X - c_j||^2 \tag{4}$$

where $X \sim P$. The optimal population quantization $C^* = \{c_1^*,\ldots,c_k^*\} \in \mathscr{C}_k$ minimizes $R(C)$. We can think of $\widehat{C}$ as an estimate of C^*. This method is called k-means clustering or vector quantization.

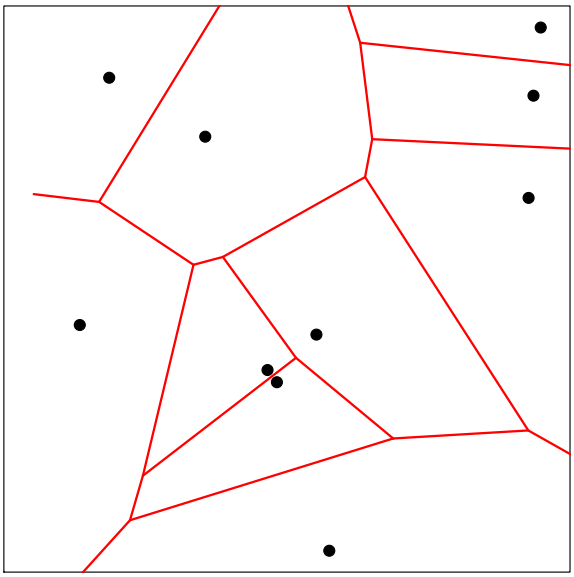

Figure 5: The Voronoi tesselation formed by 10 cluster centers $c_1, \ldots, c_{10}$. The cluster centers are indicated by dots. The corresponding Voronoi cells $T_1, \ldots, T_{10}$ are defined as follows: a point x is in T_j if x is closer to c_j than c_i for $i \neq j$.

A codebook $C = \{c_1, \ldots, c_k\}$ defines a set of cells known as a tesselation. Let

$$T_j = \Big\{ x : \ ||x - c_j|| \leq ||x - c_s||, \ \text{ for all } s \neq j \Big\}. \tag{5}$$

The set T_j is known as a Voronoi cell and consists of all points closer to c_j than any other point in the codebook. See Figure 5.

The usual algorithm to minimize $R_n(C)$ and find $\widehat{C}$ is the k-means clustering algorithm— also known as Lloyd's algorithm— see Figure 6. The risk $R_n(C)$ has multiple minima. The algorithm will only find a local minimum and the solution depends on the starting values. A common way to choose the starting values is to select k data points at random. We will discuss better methods for choosing starting values in Section 1.1.

Example 5 *Figure 7 shows synthetic data inspired by the Mickey Mouse example from*

`http://en.wikipedia.org/wiki/K-means_clustering.`

The data in the top left plot form three clearly defined clusters. k-means easily finds in the clusters (top right). The bottom shows the same example except that we now make the groups very unbalanced. The lack of balance causes k-means to produce a poor clustering.

1. Choose k centers $c_1, \ldots, c_k$ as starting values.
2. Form the clusters $C_1, \ldots, C_k$ as follows. Let $g = (g_1, \ldots, g_n)$ where $g_i = \text{argmin}_j ||X_i - c_j||$. Then $C_j = \{X_i : \ g_i = j\}$.
3. For $j = 1, \ldots, k$, let n_j denote the number of points in C_j and set

$$c_j \longleftarrow \frac{1}{n_j} \sum_{i:\ X_i \in C_j} X_i.$$

4. Repeat steps 2 and 3 until convergence.
5. Output: centers $\widehat{C} = \{c_1, \ldots, c_k\}$ and clusters $C_1, \ldots, C_k$.

Figure 6: The k-means (Lloyd's) clustering algorithm.

Example 6 *We applied k-means clustering to the Topex data with $k = 9$. The data are discretized so we treated each curve as one vector of length 70. The resulting nine clusters are shown in Figure 8.*

Example 7 (Supernova Clustering) *Returning to the supernova data, Figure 9 shows what happens if we apply k-means clustering with $k = 4$. The type Ia supernovae get split into two groups although the groups are very similar. The other type also gets split into two groups which look qualitatively different.*

k-means works best when clusters are at least roughly spherical. Otherwise, it can produce very non-intuitive clusters.

Example 8 *The top left plot of Figure 10 shows a dataset with two ring-shaped clusters. The remaining plots show the clusters obtained using k-means clustering with $k = 2, 3, 4$. Clearly, k-means does not capture the right structure in this case.*

1.1 Starting Values for k-means

Since $\widehat{R}_n(C)$ has multiple minima, Lloyd's algorithm is not guaranteed to minimize $R_n(C)$. The clustering one obtains will depend on the starting values. The simplest way to choose starting values is to use k randomly chosen points. But this often leads to poor clustering.

Example 9 *Figure 11 shows data from a distribution with nine clusters. The raw data are in the top left plot. The top right plot shows the results of running the k-means algorithm with*

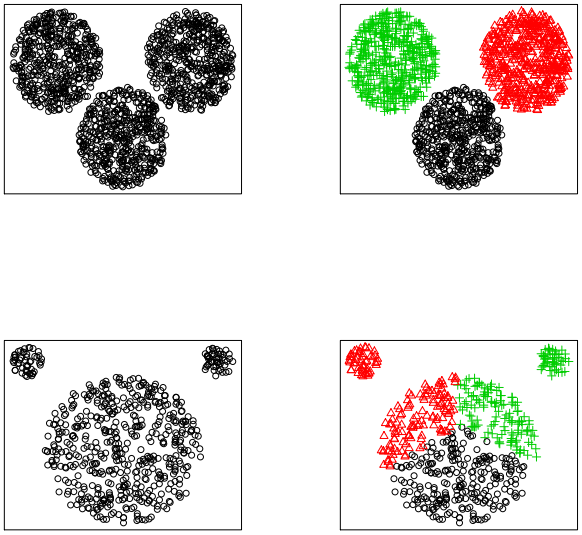

Figure 7: Synthetic data inspired by the "Mickey Mouse" example from wikipedia. Top left: three balanced clusters. Top right: result from running k means with $k = 3$. Bottom left: three unbalanced clusters. Bottom right: result from running k means with $k = 3$ on the unbalanced clusters. k-means does not work well here because the clusters are very unbalanced.

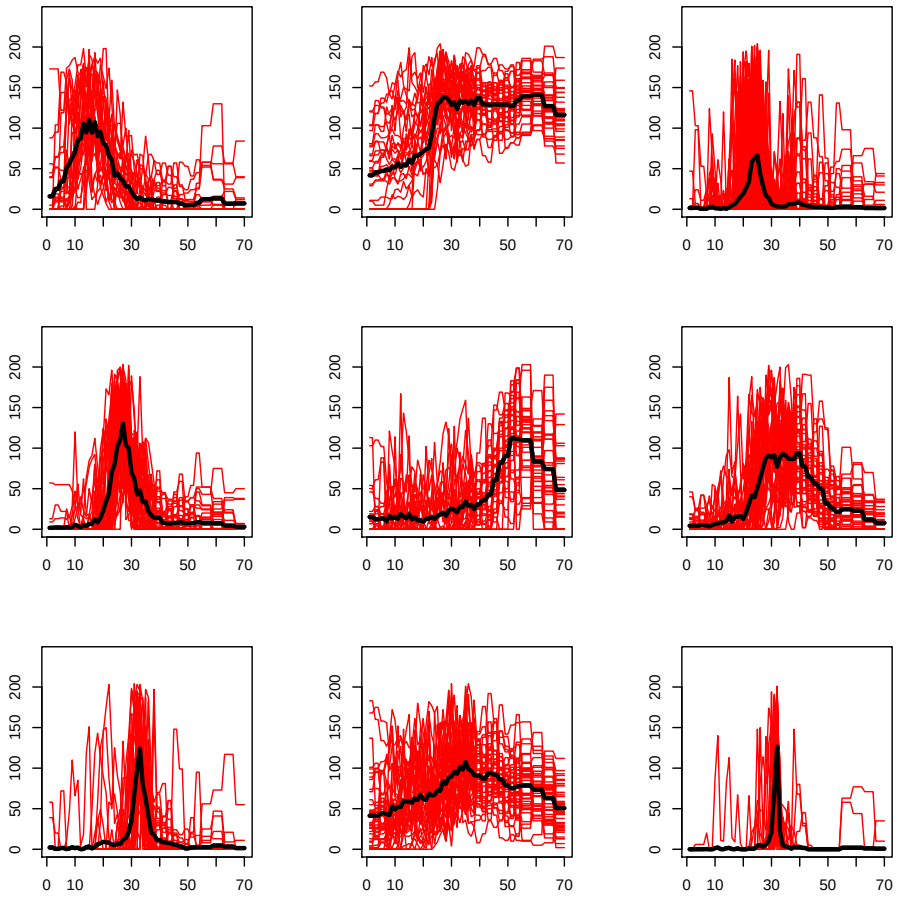

Figure 8: The nine clusters found in the Topex data using k-means clustering with $k = 9$. Each plot show the curves in that cluster together with the mean of the curves in that cluster.

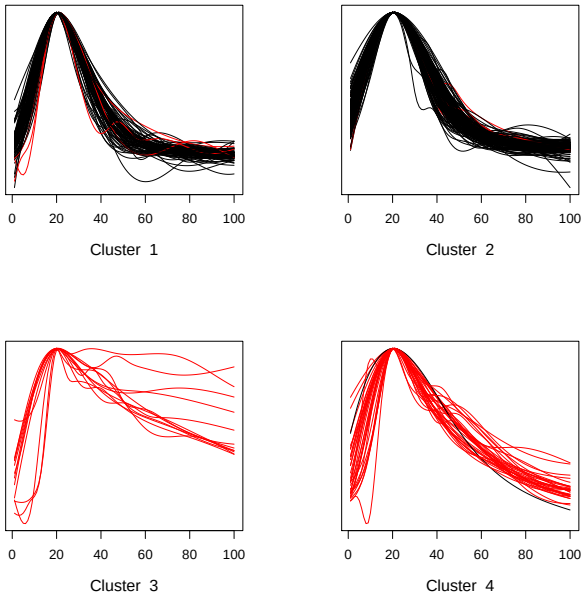

Figure 9: Clustering of the supernova light curves with $k = 4$.

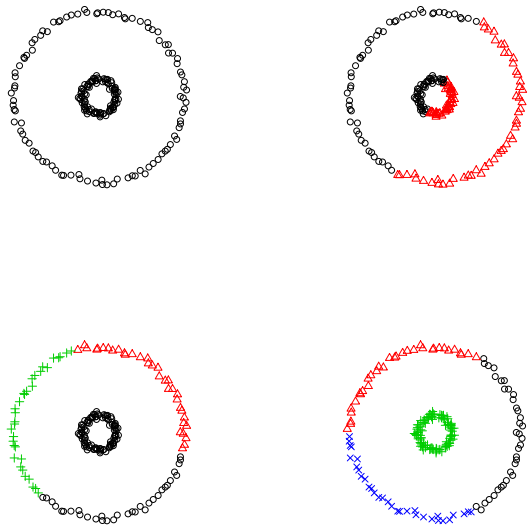

Figure 10: Top left: a dataset with two ring-shaped clusters. Top right: k-means with $k = 2$. Bottom left: k-means with $k = 3$. Bottom right: k-means with $k = 4$.

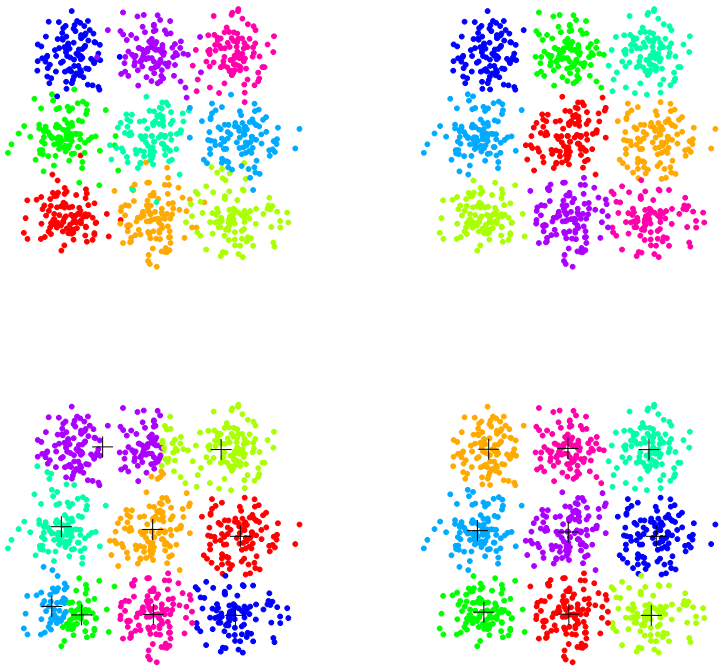

Figure 11: An example with 9 clusters. Top left: data. Top right: k-means with random starting values. Bottom left: k-means using starting values from hierarchical clustering. Bottom right: the k-means^{++} algorithm.

$k = 9$ *using random points as starting values. The clustering is quite poor. This is because we have not found the global minimum of the empirical risk function. The two bottom plots show better methods for selecting starting values that we will describe below.*

Hierarchical Starting Values. Tseng and Wong (2005) suggest the following method for choosing staring values for k-means. Run single-linkage hierarchical clustering (which we describe in Section 3) to obtains $p \times k$ clusters. They suggest using $p = 3$ as a default. Now take the centers of the k-largest of the $p \times k$ clusters and use these as starting values. See the bottom left plot in Figure 11.

k-means^{++}. Arthur and Vassilvitskii (2007) invented an algorithm called k-means^{++} to get good starting values. They show that if the starting points are chosen in a certain way, then we can get close to the minimum with high probability. In fact the starting points themselves — which we call seed points — are already close to minimizing $R_n(C)$. The

1. Input: Data $X = \{X_1, \ldots, X_n\}$ and an integer k.
2. Choose c_1 randomly from $X = \{X_1, \ldots, X_n\}$. Let $C = \{c_1\}$.
3. For $j = 2, \ldots, k$:
   (a) Compute $D(X_i) = \min_{c \in C} ||X_i - c||$ for each X_i.
   (b) Choose a point X_i from X with probability
   $$p_i = \frac{D^2(X_i)}{\sum_{j=1}^{n} D^2(X_j)}.$$
   (c) Call this randomly chosen point c_j. Update $C \longleftarrow C \cup \{c_j\}$.
4. Run Lloyd's algorithm using the **seed points** $C = \{c_1, \ldots, c_k\}$ as starting points and output the result.

Figure 12: The k-means^{++} algorithm.

algorithm is described in Figure 12. See the bottom right plot in Figure 11 for an example.

Theorem 10 *(Arthur and Vassilvitskii, 2007). Let $C = \{c_1, \ldots, c_k\}$ be the seed points from the k-means^{++} algorithm. Then,*

$$\mathbb{E}\big(R_n(C)\big) \leq 8(\log k + 2) \left(\min_C R_n(C)\right) \tag{6}$$

where the expectation is over the randomness of the algorithm.

See Arthur and Vassilvitskii (2007) for a proof. They also show that the Euclidean distance can be replaced with the ℓ_p norm in the algorithm. The result is the same except that the constant 8 gets replaced by 2^{p+2}. It is possible to improve the k-means^{++} algorithm. Ailon, Jaiswal and Monteleoni (2009) showed that, by choosing $3 \log k$ points instead of one point, at each step of the algorithm, the $\log k$ term in (6) can be replaced by a constant. They call the algorithm, k-means#.

Global k-means. This method is due to Likas and Vlassis (2003). The idea is to run k-means for $k = 1, 2, \ldots,$. The cluster centers obtained from $(k-1)$-means, together with one of the data points, are used as starting values for k-means. All such starting values are considered and the best one is used. To describe this more formally, let $\{c_1^{(k)}, \ldots, c_k^{(k)}\}$ denote the cluster centers from running the k-means algorithm.

Input: maximum number of clusters K.

1. Set $k = 1$ and set $c_1^{(1)} = \overline{X}$.
2. For $k = 2$ to K.
   (a) Run k means using starting values $\{c_1^{(k-1)}, \ldots, c_{k-1}^{(k-1)}, X_i\}$, for $i = 1, \ldots, n$.
   (b) From the n runs, choose the solution with lowest empirical risk. Denote the solution by $\{c_1^{(k)}, \ldots, c_k^{(k)}\}$.

As described above, this method will be quite slow. See Likas and Vlassis (2003) for ways to speed up the implementation.

1.2 Choosing k

In k-means clustering we must choose a value for k. This is still an active area of research and there are no definitive answers. The problem is much different than choosing a tuning parameter in regression or classification because there is no observable label to predict. Indeed, for k-means clustering, both the true risk R and estimated risk R_n decrease to 0 as k increases. This is in contrast to classification where the true risk gets large for high complexity classifiers even though the empirical risk decreases. Hence, minimizing risk does not make sense. There are so many proposals for choosing tuning parameters in clustering that we cannot possibly consider all of them here. Instead, we highlight a few methods.

Elbow Methods. One approach is to look for sharp drops in estimated risk. Let R_k denote the minimal risk among all possible clusterings and let $\widehat{R}_k$ be the empirical risk. It is easy to see that R_k is a nonincreasing function of k so minimizing R_k does not make sense. Instead, we can look for the first k such that the improvement $R_k - R_{k+1}$ is small, sometimes called an elbow. This can be done informally by looking at a plot of $\widehat{R}_k$. We can try to make this more formal by fixing a small number $\alpha > 0$ and defining

$$k_\alpha = \min\left\{k : \ \frac{R_k - R_{k+1}}{\sigma^2} \leq \alpha\right\} \tag{7}$$

where $\sigma^2 = \mathbb{E}(\|X - \mu\|^2)$ and $\mu = \mathbb{E}(X)$. An estimate of k_α is

$$\widehat{k}_\alpha = \min\left\{k : \ \frac{\widehat{R}_k - \widehat{R}_{k+1}}{\widehat{\sigma}^2} \leq \alpha\right\} \tag{8}$$

where $\widehat{\sigma}^2 = n^{-1}\sum_{i=1}^n \|X_i - \overline{X}\|^2$.

Unfortunately, the elbow method often does not work well in practice because there may not be a well-defined elbow.

Hypothesis Testing. A more formal way to choose k is by way of hypothesis testing. For each k we test

$$H_k : \text{the number of clusters is } k \quad \text{versus} \quad H_{k+1} : \text{the number of clusters is } > k.$$

We begin $k = 1$. If the test rejects, then we repeat the test for $k = 2$. We continue until the first k that is not rejected. In summary, $\widehat{k}$ is the first k for which k is not rejected.

To implement this idea, we need to choose a specific test statistic. There are numerous suggestions in the literature. Since k-means is designed for roughly spherical clusters, a reasonable approach is to test if the data in each cluster are approximately Gaussian. In one dimension, a popular test for Normality is the Shapiro-Wilk (1965) test. The test is described in the appendix. The test results in a p-value. Small p-values suggest the data do not look Normal. For one-dimensional data, the steps for using hypothesis testing to choose k are as follows:

Initialize: Choose significance level α. Set $k = 1$.

1. Perform k means clustering. Let p_j be the p-value a test for Normality for the data in the j^{th} cluster. Define the adjusted p-value $P_k = \min_j\{kp_j, 1\}$.
2. If $P_k < \alpha$ set $k = k + 1$ and go to step 1. Otherwise stop and output k.

The adjustment $P_k = \min\{kp_j, 1\}$ is to account for the fact that we are conducting several hypothesis tests. The hypothesis test is being used in an informal manner. The test does not account for the fact that the groups have been determined by the data so, technically, the test is not valid. Nonetheless, the method often yields reasonable values for k.

For multivariate data, the procedure is the same except that the one-dimensional test has to be replaced by a multi-dimensional test. There are many ways to do this. One way is to test several one-dimension projections.

Initialize: Choose significance level α. Choose number of projections B (default $B = 10$.) Set $k = 1$.

1. Perform k means clustering.
2. Select $U_1, \ldots, U_B$ randomly from the d-sphere.
3. Define $\mathscr{D}_1, \ldots, \mathscr{D}_B$ where $\mathscr{D}_j = \{Y_{j1}, \ldots, Y_{jn}\}$ and $Y_{ji} = U_j^T X_i$.
4. Let p_{js} be the p-value a test for Normality for the data in the j^{th} cluster using data $\mathscr{D}_s$.
5. Define the combined p-value $P_k = \min_{s,j}\{kBp_{js}, 1\}$.
6. If $P_k < \alpha$ set $k = k + 1$ and go to step 1. Otherwise stop and output k.

A similar procedure, called PG means is described in Feng and Hammerly (2007).

Example 11 *Figure 13 shows a two-dimensional example. The top left plot shows a single cluster. The p-values are shown as a function of k in the top right plot. The first k for which the p-value is larger than $\alpha = .05$ is $k = 1$. The bottom left plot shows a dataset with three clusters. The p-values are shown as a function of k in the bottom right plot. The first k for which the p-value is larger than $\alpha = .05$ is $k = 3$.*

Stability. Another class of methods are based on the idea of stability. The idea is to find the largest number of clusters than can be estimated with low variability.

We start with a high level description of the idea and then we will discuss the details. Suppose that $Y = (Y_1, \ldots, Y_n)$ and $Z = (Z_1, \ldots, Z_n)$ are two independent samples from P. Let A_k be any clustering algorithm that takes the data as input and outputs k clusters. Define the *stability*

$$\Omega(k) = \mathbb{E}[s(A_k(Y), A_k(Z))] \tag{9}$$

where $s(\cdot, \cdot)$ is some measure of the similarity of two clusterings. To estimate Ω we use random subsampling. Suppose that the original data are $X = (X_1, \ldots, X_{2n})$. Randomly split the data into two equal sets Y and Z of size n. This process if repeated N times. Denote the random split obtained in the j^{th} trial by Y^j, Z^j. Define

$$\widehat{\Omega}(k) = \frac{1}{N} \sum_{j=1}^{N} \left[s(A_k(Y^j), A_k(Z^j)) \right].$$

For large N, $\widehat{\Omega}(k)$ will approximate $\Omega(k)$. There are two ways to choose k. We can choose a small k with high stability. Alternatively, we can choose k to maximize $\widehat{\Omega}(k)$ if we somehow standardize $\widehat{\Omega}(k)$.

Now we discuss the details. First, we need to define the similarity between two clusterings. We face two problems. The first is that the cluster labels are arbitrary: the clustering $(1,1,1,2,2,2)$ is the same as the clustering $(4,4,4,8,8,8)$. Second, the clusterings $A_k(Y)$ and $A_k(Z)$ refer to different data sets.

The first problem is easily solved. We can insist the labels take values in $\{1, \ldots, k\}$ and then we can maximize the similarity over all permutations of the labels. Another way to solve the problem is the following. Any clustering method can be regarded as a function ψ that takes two points x and y and outputs a 0 or a 1. The interpretation is that $\psi(x, y) = 1$ if x and y are in the same cluster while $\psi(x, y) = 0$ if x and y are in a different cluster. Using this representation of the clustering renders the particular choice of labels moot. This is the approach we will take.

Let ψ_Y and ψ_Z be clusterings derived from Y and Z. Let us think of Y as training data and Z as test data. Now ψ_Y returns a clustering for Y and ψ_Z returns a clustering for Z. We'd

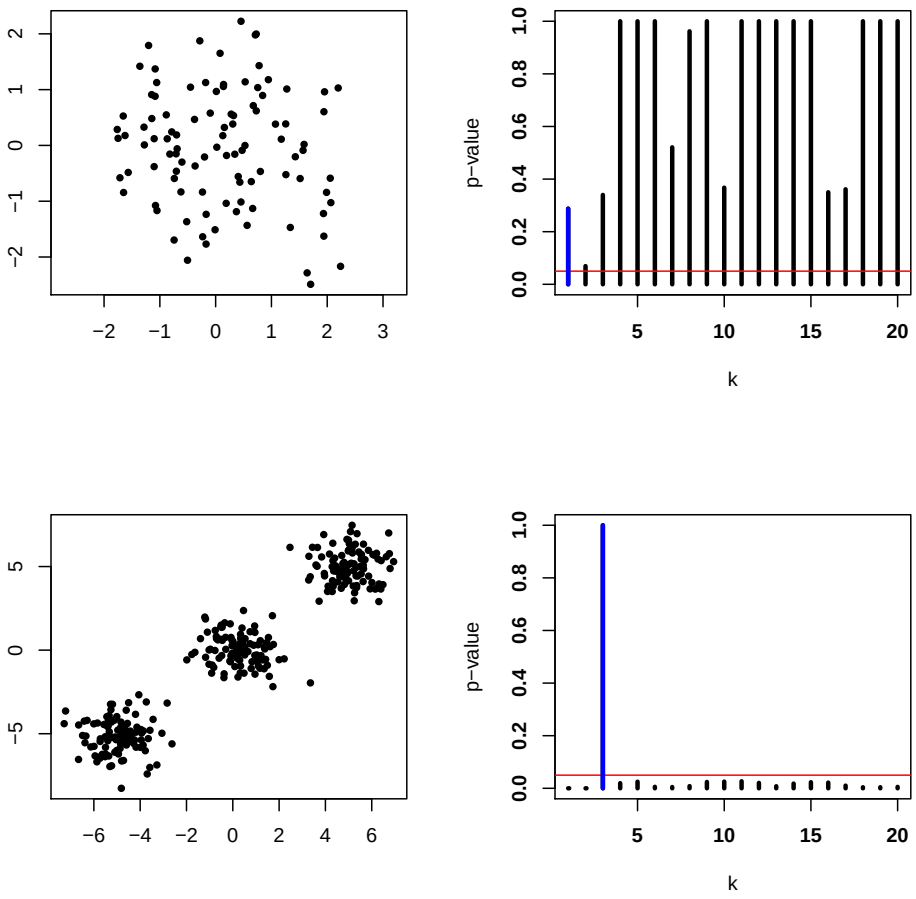

Figure 13: Top left: a single cluster. Top right: p-values for various k. The first k for which the p-value is larger than .05 is $k = 1$. Bottom left: three clusters. Bottom right: p-values for various k. The first k for which the p-value is larger than .05 is $k = 3$.

like to somehow apply ψ_Y to Z. Then we would have two clusterings for Z which we could then compare. There is no unique way to do this. A simple and fairly general approach is to define

$$\psi_{Y,Z}(Z_j, Z_k) = \psi_Y(Y'_j, Y'_k) \tag{10}$$

where Y'_j is the closest point in Y to Z_j and Y'_k is the closest point in Y to Z_k. (More generally, we can use Y and the cluster assignment to Y as input to a classifier; see Lange et al 2004). The notation $\psi_{Y,Z}$ indicates that ψ is trained on Y but returns a clustering for Z. Define

$$s(\psi_{Y,Z}, \psi_Z) = \frac{1}{\binom{n}{2}} \sum_{s \neq t} I\left(\psi_{Y,Z}(Z_s, Z_t) = \psi_Z(Z_s, Z_t)\right).$$

Thus s is the fraction of pairs of points in Z on which the two clusterings $\psi_{Y,Z}$ and ψ_Z agree. Finally, we define

$$\widehat{\Omega}(k) = \frac{1}{N} \sum_{j=1}^{N} s(\psi_{Y^j,Z^j}, \psi_{Z^j}).$$

Now we need to decide how to use $\widehat{\Omega}(k)$ to choose k. The interpretation of $\widehat{\Omega}(k)$ requires some care. First, note that $0 \leq \widehat{\Omega}(k) \leq 1$ and $\widehat{\Omega}(1) = \widehat{\Omega}(n) = 1$. So simply maximizing $\widehat{\Omega}(k)$ does not make sense. One possibility is to look for a small k larger than $k > 1$ with a high stability. Alternatively, we could try to normalize $\widehat{\Omega}(k)$. Lange et al (2004) suggest dividing by the value of $\widehat{\Omega}(k)$ obtained when cluster labels are assigned randomly. The theoretical justification for this choice is not clear. Tibshirani, Walther, Botstein and Brown (2001) suggest that we should compute the stability separately over each cluster and then take the minimum. However, this can sometimes lead to very low stability for all $k > 1$.

Many authors have considered schemes of this form, including Breckenridge (1989), Lange, Roth, Braun and Buhmann (2004), Ben-Hur, Elisseeff and Guyron (2002), Dudoit and Fridlyand (2002), Levine and Domany (2001), Buhmann (2010), Tibshirani, Walther, Botstein and Brown (2001) and Rinaldo and Wasserman (2009).

It is important to interpret stability correctly. These methods choose the largest number of stable clusters. That does not mean they choose "the true k." Indeed, Ben-David, von Luxburg and Pál (2006), Ben-David and von Luxburg Tübingen (2008) and Rakhlin (2007) have shown that trying to use stability to choose "the true k" — even if that is well-defined — will not work. To explain this point further, we consider some examples from Ben-David, von Luxburg and Pál (2006). Figure 14 shows the four examples. The first example (top left plot) shows a case where we fit $k = 2$ clusters. Here, stability analysis will correctly show that k is too small. The top right plot has $k = 3$. Stability analysis will correctly show that k is too large. The bottom two plots show potential failures of stability analysis. Both cases are stable but $k = 2$ is too small in the bottom left plot and $k = 3$ is too big in the bottom right plot. Stability is subtle. There is much potential for this approach but more work needs to be done.

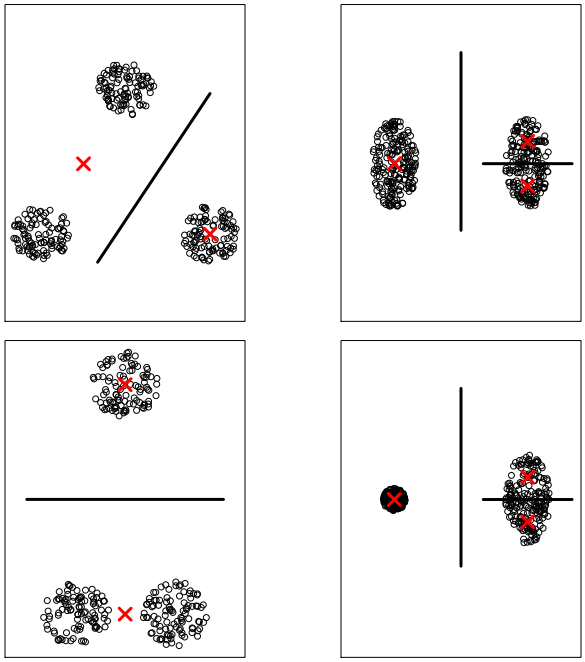

Figure 14: Examples from Ben-David, von Luxburg and Pál (2006). The first example (top left plot) shows a case where we fit $k = 2$ clusters. Stability analysis will correctly show that k is too small. The top right plot has $k = 3$. Stability analysis will correctly show that k is too large. The bottom two plots show potential failures of stability analysis. Both cases are stable but $k = 2$ is too small in the bottom left plot and $k = 3$ is too big in the bottom right plot.

1.3 Theoretical Properties

A theoretical property of the k-means method is given in the following result. Recall that $C^* = \{c_1^*, \ldots, c_k^*\}$ minimizes $R(C) = \mathbb{E}||X - \Pi_C[X]||^2$.

Theorem 12 *Suppose that* $\mathbb{P}(||X_i||^2 \leq B) = 1$ *for some* $B < \infty$. *Then*

$$\mathbb{E}(R(\widehat{C})) - R(C^*) \leq c\sqrt{\frac{k(d+1)\log n}{n}} \tag{11}$$

for some $c > 0$.

Warning! The fact that $R(\widehat{C})$ is close to $R(C_*)$ does not imply that $\widehat{C}$ is close to C_*.

This proof is due to Linder, Lugosi and Zeger (1994). The proof uses techniques from a later lecture on VC theory so you may want to return to the proof later.

Proof. Note that $R(\widehat{C}) - R(C^*) = R(\widehat{C}) - R_n(\widehat{C}) + R_n(\widehat{C}) - R(C^*) \leq R(\widehat{C}) - R_n(\widehat{C}) + R_n(C^*) - R(C^*) \leq 2\sup_{C \in \mathscr{C}_k} |R(\widehat{C}) - R_n(\widehat{C})|$. For each C define a function f_C by $f_C(x) = ||x - \Pi_C[x]||^2$. Note that $\sup_x |f_C(x)| \leq 4B$ for all C. Now, using the fact that $\mathbb{E}(Y) = \int_0^\infty \mathbb{P}(Y \geq t)dt$ whenever $Y \geq 0$, we have

$$\begin{aligned}
2\sup_{C \in \mathscr{C}_k} |R(\widehat{C}) - R_n(\widehat{C})| &= 2\sup_C \left| \frac{1}{n}\sum_{i=1}^n f_C(X_i) - \mathbb{E}(f_C(X)) \right| \\
&= 2\sup_C \left| \int_0^\infty \left(\frac{1}{n}\sum_{i=1}^n I(f_C(X_i) > u) - \mathbb{P}(f_C(Z) > u) \right) du \right| \\
&\leq 8B \sup_{C,u} \left| \frac{1}{n}\sum_{i=1}^n I(f_C(X_i) > u) - \mathbb{P}(f_C(Z) > u) \right| \\
&= 8B \sup_A \left| \frac{1}{n}\sum_{i=1}^n I(X_i \in A) - \mathbb{P}(A) \right|
\end{aligned}$$

where A varies over all sets $\mathscr{A}$ of the form $\{f_C(x) > u\}$. The shattering number of $\mathscr{A}$ is $s(\mathscr{A}, n) \leq n^{k(d+1)}$. This follows since each set $\{f_C(x) > u\}$ is a union of the complements of k spheres. By the VC Theorem,

$$\begin{aligned}
\mathbb{P}(R(\widehat{C}) - R(C^*) > \epsilon) &\leq \mathbb{P}\left(8B \sup_A \left| \frac{1}{n}\sum_{i=1}^n I(X_i \in A) - \mathbb{P}(A) \right| > \epsilon \right) \\
&= \mathbb{P}\left(\sup_A \left| \frac{1}{n}\sum_{i=1}^n I(X_i \in A) - \mathbb{P}(A) \right| > \frac{\epsilon}{8B} \right) \\
&\leq 4(2n)^{k(d+1)} e^{-n\epsilon^2/(512B^2)}.
\end{aligned}$$

Now conclude that $\mathbb{E}(R(\widehat{C}) - R(C^*)) \leq C\sqrt{k(d+1)}\sqrt{\frac{\log n}{n}}$. $\square$

A sharper result, together with a lower bound is the following.

Theorem 13 (Bartlett, Linder and Lugosi 1997) *Suppose that* $\mathbb{P}\left(\|X\|^2 \leq 1\right) = 1$ *and that* $n \geq k^{4/d}$, $\sqrt{dk^{1-2/d}\log n} \geq 15$, $kd \geq 8$, $n \geq 8d$ *and* $n/\log n \geq dk^{1+2/d}$. *Then,*

$$\mathbb{E}(R(\widehat{C})) - R(C^*) \leq 32\sqrt{\frac{dk^{1-2/d}\log n}{n}} = O\left(\sqrt{\frac{dk\log n}{n}}\right).$$

Also, if $k \geq 3$, $n \geq 16k/(2\Phi^2(-2))$ *then, for any method* $\widehat{C}$ *that selects* k *centers, there exists* P *such that*

$$\mathbb{E}(R(\widehat{C})) - R(C^*) \geq c_0\sqrt{\frac{k^{1-4/d}}{n}}$$

where $c_0 = \Phi^4(-2)2^{-12}/\sqrt{6}$ *and* Φ *is the standard Gaussian distribution function.*

See Bartlett, Linder and Lugosi (1997) for a proof. It follows that k-means is risk consistent in the sense that $R(\widehat{C}) - R(C^*) \xrightarrow{P} 0$, as long as $k = o(n/(d^3\log n))$. Moreover, the lower bound implies that we cannot find any other method that improves much over the k-means approach, at least with respect to this loss function.

The previous results depend on the dimension d. It is possible to get a dimension-free result at the expense of replacing $\sqrt{k}$ with k. In fact, the following result even applies to functions instead of vectors. In that case, we interpret $||\cdot||$ to be the norm in a Hilbert space.

Theorem 14 *(Biau, Devroye and Lugosi 2008). Suppose that* $\mathbb{P}(||X_i|| \leq B) = 1$. *Then*

$$\mathbb{E}(R(\widehat{C})) - R(C^*) \leq \frac{12B^2k}{\sqrt{n}}.$$

Proof. Define $W(C,P) = \mathbb{E}_P\left(\min_{1\leq j\leq k}\left[-2\langle X, c_j\rangle + ||c_j||^2\right]\right)$. Minimizing $R(C)$ is equivalent to minimizing $W(C,P)$ and minimizing $R_n(C)$ is equivalent to minimizing $W(C,P_n)$ where P_n is the empirical measure that puts mass $1/n$ at each X_i. Arguing as in the proof of Theorem 12,

$$\mathbb{E}(W(\widehat{C},P)) - W(C^*,P) \leq 2\mathbb{E}\Big(\sup_C W(C,P) - W(C,P_n)\Big).$$

Let $\sigma_1,\ldots,\sigma_n$ be Rademacher random variables. That is, $\sigma_1,\ldots,\sigma_n$ are iid and $\mathbb{P}(\sigma_i = +1) = \mathbb{P}(\sigma_i = -1) = 1/2$. Let $X_1',\ldots,X_n'$ be a second independent sample. Let $\ell_c(x) = -2\langle x, c\rangle + ||c||^2$.

Then,

$$\begin{aligned}
\mathbb{E}\Big(\sup_C W(C,P) - W(C,P_n)\Big) &\leq \mathbb{E}\left(\sup_C \frac{1}{n}\sum_{i=1}^n \sigma_i \left[\min_{1\leq j\leq n} \ell_{c_j}(X_i) - \min_{1\leq j\leq n} \ell_{c_j}(X_i')\right]\right) \\
&\leq \mathbb{E}\left(\sup_C \frac{1}{n}\sum_{i=1}^n \sigma_i \left[\min_{1\leq j\leq n} \ell_{c_j}(X_i)\right]\right) \\
&\quad + \mathbb{E}\left(\sup_C \frac{1}{n}\sum_{i=1}^n (-\sigma_i) \left[\min_{1\leq j\leq n} \ell_{c_j}(X_i)\right]\right) \\
&= 2\mathbb{E}\left(\sup_C \frac{1}{n}\sum_{i=1}^n \sigma_i \left[\min_{1\leq j\leq n} \ell_{c_j}(X_i)\right]\right).
\end{aligned}$$

An inductive argument shows that

$$2\mathbb{E}\left(\sup_C \frac{1}{n}\sum_{i=1}^n \sigma_i \left[\min_{1\leq j\leq n} \ell_{c_j}(X_i)\right]\right) \leq 4k\left[\mathbb{E}\sup_{c\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle X_i, c\rangle + \frac{B^2}{2\sqrt{n}}\right]$$

Also,

$$\begin{aligned}
\mathbb{E}\left(\sup_{c\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n \sigma_i \langle X_i, c\rangle\right) &= \mathbb{E}\left(\sup_{c\in\mathbb{R}^d} \frac{1}{n}\left\langle \sum_{i=1}^n \sigma_i X_i, c\right\rangle\right) = \frac{B}{n}\mathbb{E}\left|\left|\sum_{i=1}^n \sigma_i X_i\right|\right| \\
&\leq \frac{B}{n}\sqrt{\mathbb{E}\left|\left|\sum_{i=1}^n \sigma_i X_i\right|\right|^2} = B\sqrt{\frac{\mathbb{E}||X||^2}{n}} \leq \frac{B^2}{\sqrt{n}}.
\end{aligned}$$

□

The k-means algorithm can be generalized in many ways. For example, if we replace the L_2 norm with the L_1 norm we get k-medians clustering. We will not discuss these extensions here.

2 Geometric Graph Clustering

In geometric graph clustering we form a graph G_ϵ with one node for each observation X_i. We put an edge between X_i and X_j if and only if $d(X_i, X_j) \leq \epsilon$ where $\epsilon \geq 0$ is a tuning parameter. Here, d is nay measure of similarity. We will use the Euclidean metric $d(X_i, X_j) = ||X_i - X_j||$. The vertices and edges define the graph G_ϵ. Define the clusters to be the connected components of G_ϵ. We shall see that this type of clustering is related to hierarchical clustering (Section 3) and spectral clustering (Section 4).

To find the connected components of the graph, we mention two algorithms: depth-first search and the spectral method. The latter is discussed in Section 4. The depth-first algorithm is as follows:

1. Pick a starting node u.
2. Run traverse to find the connected component containing u.
3. Remove this component and repeat (until there are no more nodes).

traverse: In this routine, nodes are marked white (unvisited), gray (in progress) and black (finished).

1. Mark all nodes to *white*.
2. Mark u as *gray*.
3. If there is an edge (u,v), where v is white: go to v and mark v gray.
4. Continue until you reach a node w such that there is no edge (w,x) where x is white. Mark w as *black* and backtrack to parent.
5. Continue until all nodes are black.

Example 15 *Figure 15 shows a simple example with two clusters. The top row corresponds to* $\epsilon = 1$ *and the bottom row corresponds to* $\epsilon = 3$*. The left column shows the graph. The right column shows a ball of radius* $\epsilon/2$ *around each* X_i*. In this way, the connected components of the graph show up as connected sets. The top plot of Figure 16 shows the number of connected components versus* ϵ*. There is a fairly large range of values of* ϵ *that yields two connected components. The bottom plot shows the values of* ϵ *at which* $k(\epsilon)$ *changes.*

Example 16 *The top left plot of Figure 17 shows data with two ring-shaped clusters. The remaining plots show geometric graph clustering with* $\epsilon = .2, 1$ *and 6. Clearly, the clustering is quite sensitive to the choice of* ϵ*.*

Example 17 *Figure 18 shows data with two ring-shaped clusters. In this case, we have also added background noise. The noise consist of points drawn from a uniform distribution. Here we see that geometric graph clustering fails.*

As the last example shows, background noise makes geometric graph clustering fail. It is tempting to fix this by trying to remove the background noise first. But this is essentially what density based clustering does; see Section **??**.

3 Hierarchical Clustering

Hierarchical clustering methods build a set of nested clusters at different resolutions. The are two types of hierarchical clustering: agglomerative (bottom-up) and divisive (top-down).

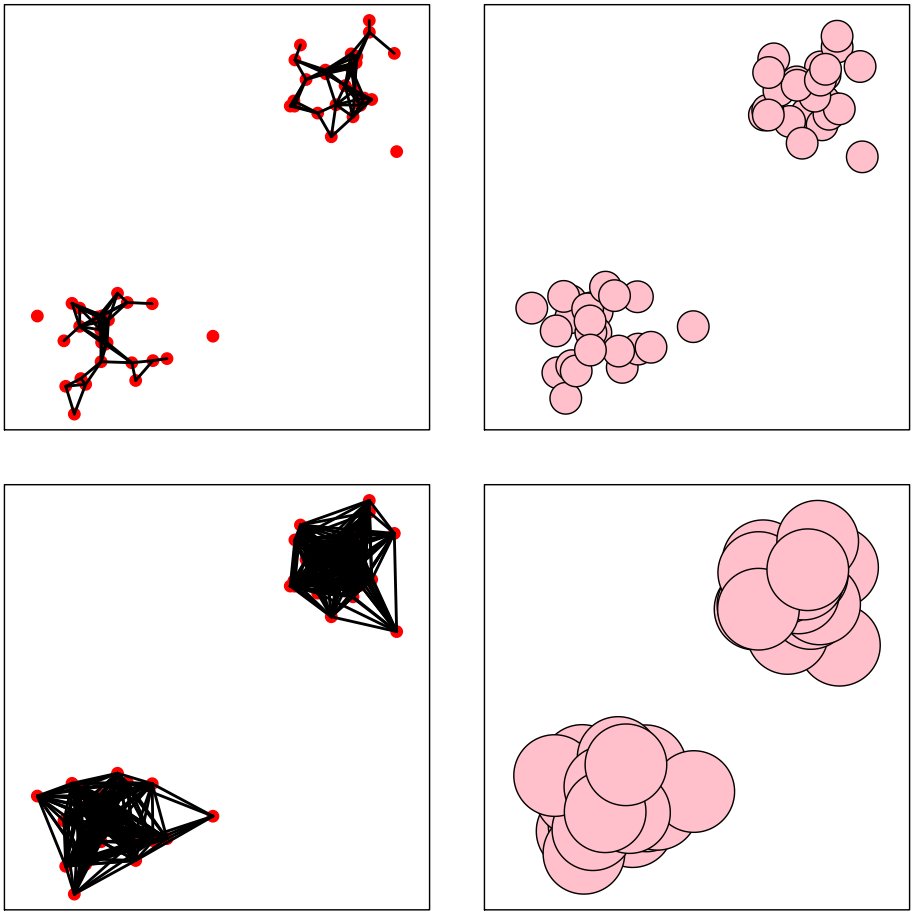

Figure 15: Geometric graph clustering. Top row: $\epsilon = 1$. Bottom row: $\epsilon = 3$. Left column: the graphs. Right column: a ball of radius $\epsilon/2$ around each X_i.

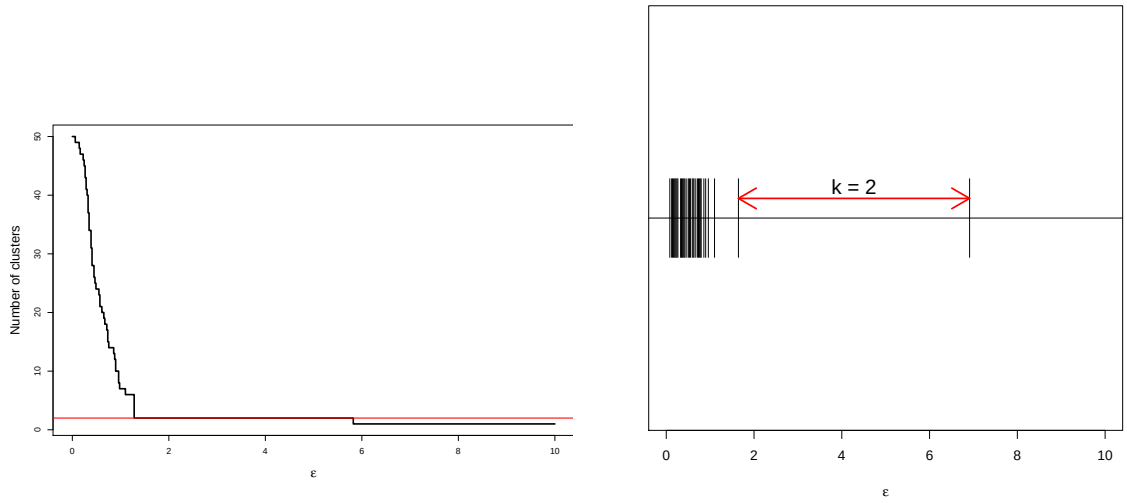

Figure 16: Geometric graph clustering. The top plot shows the number of connected components $k(\epsilon)$ versus ϵ. There is a fairly large range of values of ϵ that yields two connected components. The bottom plot shows the values of ϵ at which $k(\epsilon)$ changes.

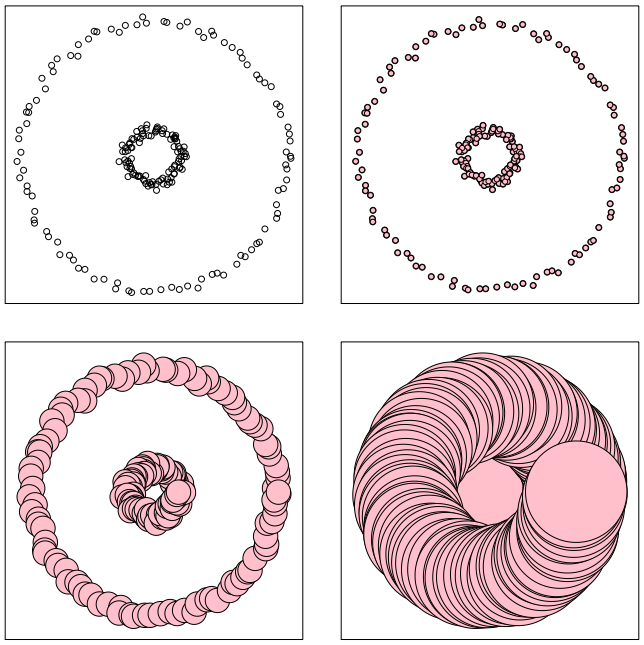

Figure 17: Geometric graph clustering applied to two ring-shaped clusters. Top left: data. Top right: $\epsilon = .2$. Bottom left: $\epsilon = 1.0$. Bottom right: $\epsilon = 6$.

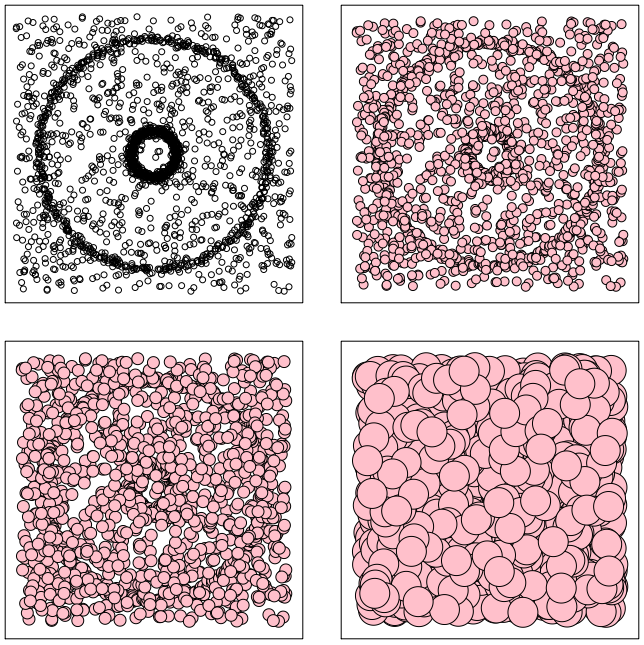

Figure 18: Geometric graph clustering applied to two ring-shaped clusters with background noise. Top left: data. Top right: $\epsilon = 0.40$. Bottom left: $\epsilon = 0.54$. Bottom right: $\epsilon = 1.46$.

With agglomerative clustering we start with some distance or dissimilarity $d(x, y)$ between points. We then extend this distance so that we can compute the distance $d(A, B)$ between to sets of points A and B.

The three most common ways of extending the distance are defined in Figure 19. The algorithm is given in Figure 20.

The result can be represented as a tree, called a dendogram. We can then cut the tree at different places to yield any number of clusters ranging from 1 to n. Single linkage often produces thin clusters while complete linkage is better at rounder clusters. Average linkage is in between.

Example 18 *Figure 21 shows agglomerative clustering applied to data generated from two rings plus noise. The noise is large enough so that the smaller ring looks like a blob. The data are show in the top left plot. The top right plot shows hierarchical clustering using single linkage. (The tree is cut to obtain two clusters.) The bottom left plot shows average linkage and the bottom right plot shows complete linkage. Single linkage works well while average and complete linkage do poorly.*

Let us now mention some theoretical properties of hierarchical clustering. Suppose that

Single Linkage	$d(A,B) = \min_{x \in A, y \in B} d(x,y)$
Average Linkage	$d(A,B) = \frac{1}{N_A N_B} \sum_{x \in A, y \in B} d(x,y)$
Complete Linkage	$d(A,B) = \max_{x \in A, y \in B} d(x,y)$

Figure 19: Three common distances between sets of points A and B used in hierarchical clustering. N_A is the number of points in A and similarly for N_B.

1. Input: data $X = \{X_1, \ldots, X_n\}$ and metric d giving distance between clusters.
2. Let $T_n = \{C_1, C_2, \ldots, C_n\}$ where $C_i = \{X_i\}$.
3. For $j = n-1$ to 1:
   (a) Find j, k to minimize $d(C_j, C_k)$ over all $C_j, C_k \in T_{j+1}$.
   (b) Let T_j be the same as T_{j+1} except that C_j and C_k are replaced with $C_j \cup C_k$.
4. Return the sets of clusters $T_1, \ldots, T_n$.

Figure 20: Agglomerative Hierarchical Clustering

.

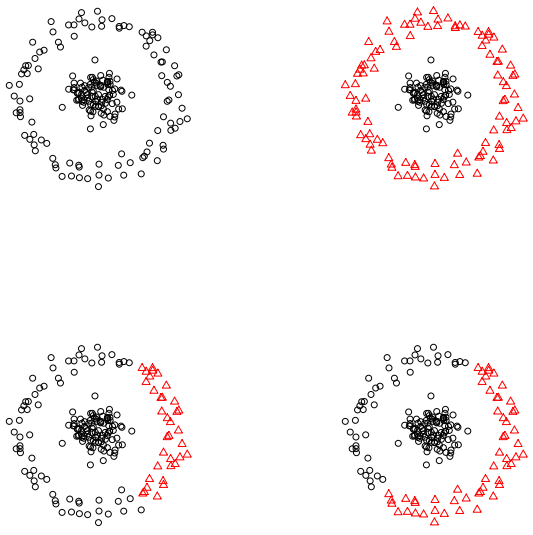

Figure 21: Hierarchical clustering applied to two noisy rings. Top left: the data. Top right: two clusters from hierarchical clustering using single linkage. Bottom left: average linkage. Bottom right: complete linkage.

$X_1,\ldots,X_n$ is a sample from a distribution P on $\mathbb{R}^d$ with density p. A high density cluster is a maximal connected component of a set of the form $\{x:\ p(x)\geq\lambda\}$. One might expect that single linkage clusters would correspond to high density clusters. This turns out not quite to be the case. See Hartigan (1981) for details. DasGupta (2010) has a modified version of hierarchical clustering that attempts to fix this problem. His method is very similar to density clustering.

Single linkage hierarchical clustering is the same as geometric graph clustering. Let $G=(V,E)$ be a graph where $V=\{X_1,\ldots,X_n\}$ and $E_{ij}=1$ if $||X_i-X_j||\leq\epsilon$ and $E_{ij}=0$ if $||X_i-X_j||>\epsilon$. Let $C_1,\ldots,C_k$ denote the connected components of the graph. As we vary ϵ we get exactly the hierarchical clustering tree.

Finally, we let us mention divisive clustering. This is a form of hierarchical clustering where we start with one large cluster and then break the cluster recursively into smaller and smaller pieces.

4 Spectral Clustering

Spectral clustering refers to a class of clustering methods that use ideas related to eigenvector. An excellent tutorial on spectral clustering is von Luxburg (2006) and some of this section relies heavily on that paper. More detail can be found in Chung (1997).

Let G be an undirected graph with n vertices. Typically these vertices correspond to observations $X_1,\ldots,X_n$. Let W be an $n\times n$ symmetric weight matrix. Say that X_i and X_j are connected if $W_{ij}>0$. The simplest type of weight matrix has entries that are either 0 or 1. For example, we could define

$$W_{ij}=I(||X_i-X_j||\leq\epsilon)$$

as we did in Section 2. An example of a more general weight matrix is $W_{ij}=e^{-||X_i-X_j||^2/(2h^2)}$.

The degree matrix D is the $n\times n$ diagonal matrix with $D_{ii}=\sum_{j=1}^n W_{ij}$. The graph Laplacian is

$$L=D-W. \tag{12}$$

The graph Laplacian has many interesting properties which we list in the following result. Recall that a vector v is an eigenvector of L if there is a scalar λ such that $Lv=\lambda v$ in which case we say that λ is the eigenvalue corresponding to v. Let $\mathscr{L}(v)=\{cv:\ c\in\mathbb{R},\ c\neq 0\}$ be the linear space generated by v. If v is an eigenvector with eigenvalue λ and c is any nonzero constant, then cv is an eigenvector with eigenvalue $c\lambda$. These eigenvectors are considered equivalent. In other words, $\mathscr{L}(v)$ is the set of vectors that are equivalent to v.

Theorem 19 *The graph Laplacian L has the following properties:*

1. *For any vector* $f = (f_1, \ldots, f_n)^T$,
$$f^T L f = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij}(f_i - f_j)^2.$$
2. *L is symmetric and positive semi-definite.*
3. *The smallest eigenvalue of L is 0. The corresponding eigenvector is* $(1, 1, \ldots, 1)^T$.
4. *L has n non-negative, real-valued eigenvalues* $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_k$.
5. *The number of eigenvalues that are equal to 0 is equal to the number of connected components of G. That is,* $0 = \lambda_1 = \ldots = \lambda_k$ *where k is the number of connected components of G. The corresponding eigenvectors $v_1, \ldots, v_k$ are orthogonal and each is constant over one of the connected components of the graph.*

Part 1 of the theorem says that L is like a derivative operator. The last part shows that we can use the graph Laplacian to find the connected components of the graph.

Proof.

(1) This follows from direct algebra.

(2) Since W and D are symmetric, it follow that L is symmetric. The fact that L is positive semi-definite folows from part (1).

(3) Let $v = (1, \ldots, 1)^T$. Then
$$Lv = Dv - Wv = \begin{pmatrix} D_{11} \\ \vdots \\ D_{nn} \end{pmatrix} - \begin{pmatrix} D_{11} \\ \vdots \\ D_{nn} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$
which equals $0 \times v$.

(4) This follows from parts (1)-(3).

(5) First suppose that $k = 1$ and thus that the graph is fully connected. We already know that $\lambda_1 = 0$ and $v_1 = (1, \ldots, 1)^T$. Suppose there were another eigenvector v with eigenvalue 0. Then
$$0 = v^T L v = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij}(v(i) - v(j))^2.$$

It follows that $W_{ij}(v(i)-v(j))^2 = 0$ for all i and j. Since G is fully connected, all $W_{ij} > 0$. Hence, $v(i) = v(j)$ for all i, j and so v is constant and thus $v \in \mathscr{L}(v_1)$.

Now suppose that K has k components. Let n_j be the number of nodes in components j. We can reliable the vertices so that the first n_1 nodes correspond to the first connected component, the second n_2 nodes correspond to the second connected component and so on. Let $v_1 = (1,\ldots,1,0,\ldots,0)$ where the 1's correspond to the first component. Let Let $v_2 = (0,\ldots,0,1,\ldots,1,0,\ldots,0)$ where the 1's correspond to the second component. Define $v_3,\ldots,v_k$ similarly. Due to the re-ordering of the vertices, L has block diagonal form:

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}.$$

Here, each L_i corresponds to one of th connected components of the graph. It is easy to see that $LV - j = 0$ for $j = 1,\ldots,k$. Thus, each v_j, for $j = 1,\ldots,k$ is an eigenvector with zero eigenvalue. Suppose that v is any eigenvector with 0 eigenvalue. Arguing as before, v must be constant over some component and 0 elsewhere. Hence, $v \in \mathscr{L}(v_j)$ for some $1 \le j \le k$. □

Example 20 *Consider the graph*

and suppose that $W_{ij} = 1$ if and only if there is an edge between X_i and X_j. Then

$$W = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and the Laplacian is

$$L = D - W = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 0 \end{pmatrix}.$$

The eigenvalues of W, from smallest to largest are $0,0,1,2,3$. *The eigenvectors are*

$$v_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \; v_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \; v_3 = \begin{pmatrix} 0 \\ 0 \\ -.71 \\ 0 \\ .71 \end{pmatrix} \; v_4 = \begin{pmatrix} -.71 \\ .71 \\ 0 \\ 0 \\ 0 \end{pmatrix} \; v_5 = \begin{pmatrix} 0 \\ 0 \\ -.41 \\ .82 \\ -.41 \end{pmatrix}$$

Note that the first two eigenvectors correspond to the connected components of the graph.

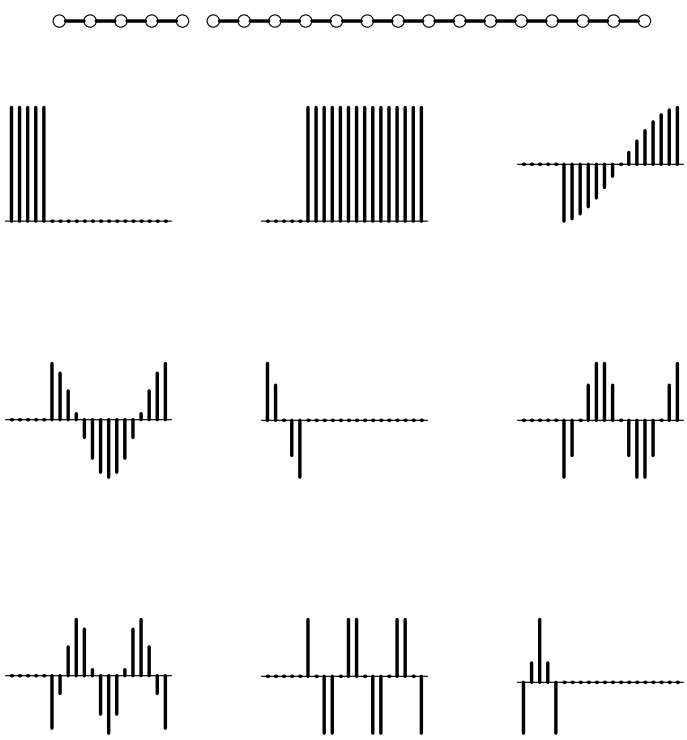

Figure 22: The top shows a simple graph. The remaining plots are the eigenvectors of the graph Laplacian. Note that the first two eigenvectors correspond to the two connected components of the graph.

Note $f^T L f$ measures the smoothness of f relative to the graph. This means that the higher order eigenvectors generate a basis where the first few basis elements are smooth (with respect to the graph) and the later basis elements become more wiggly.

Example 21 *Figure 22 shows a graph and the corresponding eigenvectors. The two eigenvectors correspond two the connected components of the graph. The other eignvectors can be thought of as forming bases vectors within the connected components.*

One approach to spectral clustering is to set

$$W_{ij} = I(||X_i - X_j|| \leq \epsilon)$$

for some $\epsilon > 0$ and then take the clusters to be the connected components of the graph which can be found by getting the eigenvectors of the Laplacian L. This is exactly equivalent to

geometric graph clustering from Section 2. In this case we have gained nothing except that we have a new algorithm to find the connected components of the graph. However, there are other ways to use spectral methods for clustering as we now explain.

The idea underlying the other spectral methods is to use the Laplacian to transform the data into a new coordinate system in which clusters are easier to find. For this purpose, one typically uses a modified form of the graph Laplacian. The most commonly used weights for this purpose are

$$W_{ij} = e^{-||X_i - X_j||^2/(2h^2)}.$$

Other kernels $K_h(X_i, X_j)$ can be used as well. We define the symmetrized Laplacian $\mathscr{L} = D^{-1/2}WD^{-1/2}$ and the random walk Laplacian $\mathscr{L} = D^{-1}W$. (We will explain the name shortly.) These are very similar and we will focus on the latter. Some authors define the random walk Laplacian to be $I - D^{-1}W$. We prefer to use the definition $\mathscr{L} = D^{-1}W$ because, as we shall see, it has a nice interpretation. The eigenvectors of $I - D^{-1}W$ and $D^{-1}W$ are the same so it makes little difference which definition is used. The main difference is that the connected components have eigenvalues 1 instead of 0.

Lemma 22 *Let L be the graph Laplacian of a graph G and let $\mathscr{L}$ be the random walk Laplacian.*

1. *λ is an eigenvalue of $\mathscr{L}$ with eigenvector v if and only if $Lv = (1-\lambda)Dv$.*
2. *1 is an eigenvalue of $\mathscr{L}$ with eigenvector $(1,\ldots,1)^T$.*
3. *$\mathscr{L}$ is positive semidefinite with n non-negative real-valued eigenvalues.*
4. *The number of eigenvalues of $\mathscr{L}$ equal to 1 equals the number of connected components of G. Let $v_1,\ldots,v_k$ denote the eigenvectors with eigenvalues equal to 1. The linear space spanned by $v_1,\ldots,v_k$ is spanned by the indicator functions of the connected components.*

Proof. Homework. □

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathscr{L}$ with eigenvectors $v_1,\ldots,v_n$. Define

$$Z_i \equiv T(X_i) = \sum_{j=1}^{r} \sqrt{\lambda_j}\, v_j(i).$$

The mapping $T : X \to Z$ transforms the data into a new coordinate system. The numbers h and r are tuning parameters. The hope is that clusters are easier to find in the new parameterization.

To get some intuition for this, note that $\mathscr{L}$ has a nice probabilistic interpretation (Coifman, Lafon, Lee 2006). Consider a Markov chain on $X_1,\ldots,X_n$ where we jump from X_i to X_j with

probability

$$\mathbb{P}(X_i \longrightarrow X_j) = \mathscr{L}(i,j) = \frac{K_h(X_i, X_j)}{\sum_s K_h(X_s, X_j)}.$$

The Laplacian $\mathscr{L}(i,j)$ captures how easy it is to move from X_i to X_j. If Z_i and Z_j are close in Euclidean distance, then they are connected by many high density paths through the data. This Markov chain is a discrete version of a continuous Markov chain with transition probability:

$$P(x \to A) = \frac{\int_A K_h(x,y)dP(y)}{\int K_h(x,y)dP(y)}.$$

The corresponding averaging operator $\widehat{A} : f \to \widetilde{f}$ is

$$(\widehat{A}f)(i) = \frac{\sum_j f(j) K_h(X_i, X_j)}{\sum_j K_h(X_i, X_j)}$$

which is an estimate of $A : f \to \widetilde{f}$ where

$$Af = \frac{\int_A f(y) K_h(x,y)dP(y)}{\int K_h(x,y)dP(y)}.$$

The lower order eigenvectors of $\mathscr{L}$ are vectors that are smooth relative to P. Thus, projecting onto the first few eigenvectors parameterizes in terms of closeness with respect to the underlying density.

The steps are:

Input: $n \times n$ similarity matrix W.

1. Let D be the $n \times n$ diagonal matrix with $D_{ii} = \sum_j W_{ij}$.
2. Compute the Laplacian $\mathscr{L} = D^{-1}W$.
3. Find first k eigenvectors $v_1, \ldots, v_k$ of $\mathscr{L}$.
4. Project each X_i onto the eigenvectors to get new points $\widehat{X}_i$.
5. Cluster the points $\widehat{X}_1, \ldots, \widehat{X}_n$ using any standard clustering algorithm.

There is another way to think about spectral clustering. Spectral methods are similar to multidimensional scaling. However, multidimensional scaling attempts to reduce dimension while preserving all pairwise distances. Spectral methods attempt instead to preserve local distances.

Example 23 *Figure 23 shows a simple synthetic example. The top left plot shows the data. We apply spectral clustering with Gaussian weights and bandwidth* $h = 3$. *The top middle plot shows the first 20 eigenvalues. The top right plot shows the the first versus the second eigenvector. The two clusters are clearly separated. (Because the clusters are so separated,*

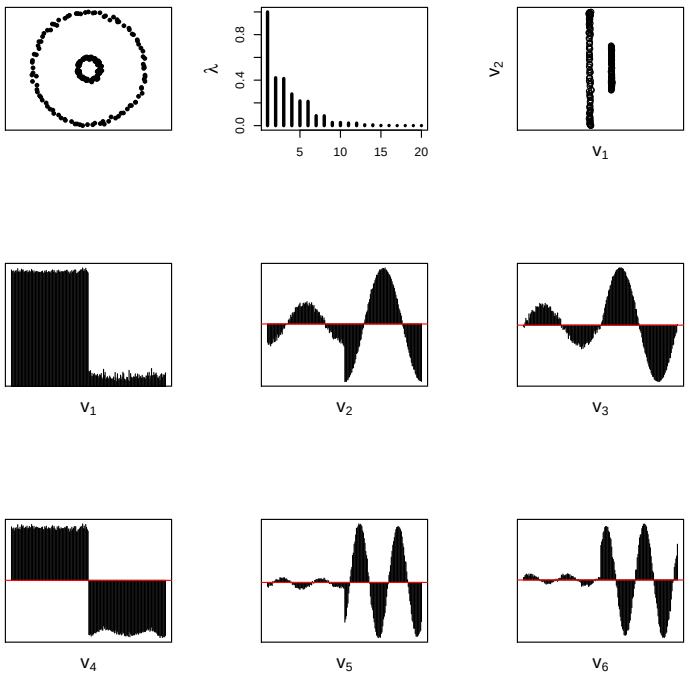

Figure 23: Top left: data. Top middle: eigenvalues. Top right: second versus third eigenvectors. Remaining plots: first six eigenvectors.

the graph is essentially disconnected and the first eigenvector is not constant. For large h, the graph becomes fully connected and v_1 is then constant.) The remaining six plots show the first six eigenvectors. We see that they form a Fourier-like basis within each cluster. Of course, single linkage clustering would work just as well with the original data as in the transformed data. The real advantage would come if the original data were high dimensional.

Example 24 *Figure 24 shows a spectral analysis of some zipcode data. Each datapoint is a 16 x 16 image of a handwritten number. We restrict ourselves to the digits 1, 2 and 3. We use Gaussian weights and the top plots correspond to h = 6 while the bottom plots correspond to h = 4. The left plots show the first 20 eigenvalues. The right plots show a scatterplot of the second versus the third eigenvector. The three colors correspond to the three digits. We see that with a good choice of h, namely h = 6, we can clearly see the digits in the plot. The original dimension of the problem is 16 x 16 =256. That is, each image can be represented by a point in $\mathbb{R}^{256}$. However, the spectral method shows that most of the information is captured by two eignvectors so the effective dimension is 2. This example also shows that the choice of h is crucial.*

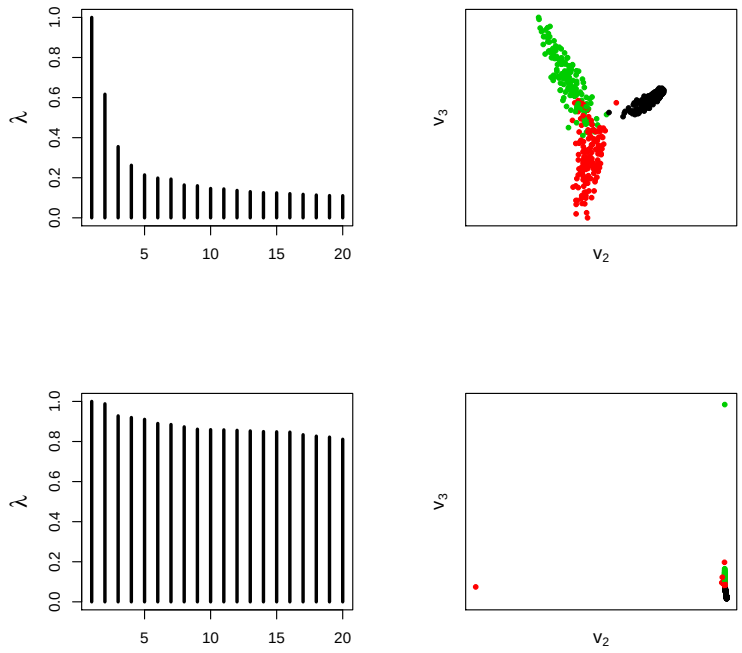

Figure 24: Spectral analysis of some zipcode data. Top: $h = 6$. Bottom: $h = 4$. The plots on the right show the second versus third eigenvector. The three colors correspond to the three digits 1, 2 and 3.

Spectral methods are very promising. However, there are some open questions:

1. There are tuning parameters (such as h) and the results are sensitive to these parameters. How do we choose these tuning parameters?
2. Does spectral clustering perform better than density clustering?

5 Variable Selection

If $X \in \mathbb{R}^d$ is high dimensional, then it makes sense to do variable selection before clustering. There are a number of methods for doing this. But, frankly, none are very convincing. This is, in my opinion, an open problem. Here are a couple of possibilities.

5.1 Marginal Selection

As in linear regression, a convenient strategy is to add variables one at a time in a greedy way. To do so, we need a method to measure the "clusteriness" of a random variable.

If X is a random variable with density p, one measure of how the clusteriness of p is the entropy

$$H(X) = H(p) = -\int p(x) \log p(x) dx. \tag{13}$$

Large values of $H(X)$ correspond to distributions that are more uniform (less clustered). Hence, we are interested in small values of $H(X)$. The plugin estimator of entropy is $-\int \widehat{p}(x) \log \widehat{p}(x) dx$ where $\widehat{p}$ is a estimate of p. However, we can estimate H without having to estimate the density. The nearest neighbor estimator of entropy (Kozachenko and Leonenko 1987) is

$$H_n = \frac{1}{n} \sum_{i=1}^{n} \log(nR_i) + \log 2 + C \tag{14}$$

where

$$R_i = \min_{j \neq i} ||X_i - X_j|| \tag{15}$$

and $C = -\int_0^\infty e^{-t} \log(t) dt$. We do not need an accurate estimate of H; we only need to rank the clusteriness of different variables. Hence, any monotone function of H_n will suffice. Thus we define

$$T(X) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \neq i} ||X_i - X_j||. \tag{16}$$

A greedy method for variable selection is to add the variables, one at a time, with the smallest T value. The algorithm is in Figure 25.

1. Standardize all variables to have mean 0 and variance 1.
2. Set $I = \emptyset$ and $A = \{1, \ldots, d\}$.
3. Compute $T_j = T(X_j)$ for $j = 1, \ldots, d$.
4. Let $j = \text{argmin}_s T_s$. Add j to I and remove j from A.
5. Repeat until A is empty:
   (a) Compute $T(X_I, X_s)$ for all $s \in A$, where $X_I = (X_i : i \in I)$.
   (b) Let $j = \text{argmin}_{s \in A} T(X_I, X_s)$.
   (c) Add j to I and remove it from A.

Figure 25: Forward Variable Selection For Clustering

The algorithm adds the variables one at a time. One must still decide on a stopping rule. The same methods for choosing the number of clusters can be used to choose the stopping point.

5.2 Sparse k-means

Here we discuss the approach in Witten and Tibshirani (2010). Recall that in k-means clustering we choose $C = \{c_1, \ldots, c_k\}$ to minimize

$$R_n(C) = \frac{1}{n} \sum_{i=1}^{n} ||X_i - \Pi_C[X_i]||^2 = \frac{1}{n} \sum_{i=1}^{n} \min_{1 \le j \le k} ||X_i - c_j||^2. \tag{17}$$

This is equivalent to minimizing the within sums of squares

$$\sum_{j=1}^{k} \frac{1}{n_j} \sum_{s,t \in A_j} d^2(X_s, X_t) \tag{18}$$

where A_j is the j^{th} cluster and $d^2(x, y) = \sum_{r=1}^{d} (x(r) - y(r))^2$ is squared Euclidean distance. Further, this is equivalent to maximizing the between sums of squares

$$B = \frac{1}{n} \sum_{s,t} d^2(X_s, X_t) - \sum_{j=1}^{k} \frac{1}{n_j} \sum_{s,t \in A_j} d^2(X_s, X_t). \tag{19}$$

Witten and Tibshirani propose replace the Euclidean norm with the weighted norm $d_w^2(x, y) = \sum_{r=1}^{d} w_r (x(r) - y(r))^2$. Then they propose to maximize

$$B = \frac{1}{n} \sum_{s,t} d_w^2(X_s, X_t) - \sum_{j=1}^{k} \frac{1}{n_j} \sum_{s,t \in A_j} d_w^2(X_s, X_t) \tag{20}$$

1. Input $X_1, \ldots, X_n$ and k.
2. Set $w = (w_1, \ldots, w_d)$ where $w_1 = \ldots = w_d = 1/\sqrt{d}$.
3. Iterate until convergence:
   (a) Optimize (19) over C holding w fixed. Find $c_1, \ldots, c_k$ from the k-means algorithm using distance $d_w(X_i, X_j)$. Let A_j denote the j^{th} cluster.
   (b) Optimize (19) over w holding $c_1, \ldots, c_k$ fixed. The solution is

$$w_r = \frac{s_r}{\sqrt{\sum_{t=1}^{d} s_t^2}}$$

   where

$$s_r = (a_r - \Delta)_+,$$

$$a_r = \left[\frac{1}{n} \sum_{s,t} w_r (X_s(r) - X_t(r))^2 - \sum_{j=1}^{k} \frac{1}{n_j} \sum_{s,t \in A_j} w_r (X_s(r) - X_t(r))^2 \right]_+$$

   and $\Delta = 0$ if $||w||_1 < s$ otherwise $\Delta > 0$ is chosen to that $||w||_1 = s$.

Figure 26: The Witten-Tibshirani Sparse k-means Method

over C and w subject to the constraints

$$||w||^2 \leq 1, \quad ||w||_1 \leq s, \quad w_j \geq 0$$

where $w = (w_1, \ldots, w_d)$. The optimization is done iteratively by optimizing over C, optimizing over w and repeating. See Figure **??**.

The ℓ_1 norm on the weights causes some of the components of w to be 0 which results in variable selection. It may seem odd that the procedure is applied to the between sums of squares rather than within sums of squares but, currently, this is the only version that is know to work.

6 Summary

The main clustering methods are:

1. density-based
2. k-means

3. hierarchical
4. spectral.

Each method has strengths and weaknesses. They all involve tuning parameters that must be chosen carefully.

7 Bibliographic Remarks

k-means clustering goes back to Stuart Lloyd who apparently came up with the algorithm in 1957 although he did not publish it until 1982. See [**?**]. Another key reference is [**?**]. Similar ideas appear in [**?**]. The related area of mixture models is discussed at length in McLachlan and Basford (1988). k-means is actually related to principal components analysis; see Ding and He (2004) and Zha, He, Ding, Simon and Gu (2001). The probabilistic behavior of random geometric graphs is discussed in detail in [**?**].