Clustering
10/26-702 Spring 2016

1 The Clustering Problem

In a clustering problem we aim to find groups in the data. Unlike classification, the data are not labeled, and so clustering is considered an example of unsupervised learning. In many cases, clustering methods are presented as heuristic algorithms without a clearly stated goal. We will cast clustering problems as estimating some population quantities.

Example 1 *Figures 1 and 2 show 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. In fact, finding clusters of this type requires nonparametric methods.*

2 Mode-Based Clustering

Let p be the density of $X \in \mathbb{R}^d$. Assume that p has modes $m_1, \ldots, m_{k_0}$ and that p is a *Morse function*, which means that the Hessian of p at each stationary point is non-degenerate. We can use the modes to define clusters as follows.

Given any point $x \in \mathbb{R}^d$, there is a unique gradient ascent path, or integral curve, passing through x that eventually leads to one of the modes. We define the clusters to be the "basins of attraction" of the modes, the equivalence classes of points whose ascent paths lead to the same mode. Formally, an *integral curve* through x is a path $\pi_x : \mathbb{R} \to \mathbb{R}^d$ such that $\pi_x(0) = x$ and

$$\pi_x'(t) = \nabla p(\pi_x(t)). \tag{1}$$

Integral curves never intersect (except at stationary points) and they partition the space.

Equation (1) means that the path π follows the direction of steepest ascent of p through x. The destination of the integral curve π through a (non-mode) point x is defined by

$$\text{dest}(x) = \lim_{t \to \infty} \pi_x(t). \tag{2}$$

It can then be shown that for all x, $\text{dest}(x) = m_j$ for some mode m_j. That is: all integral curves lead to modes. For each mode m_j, define the sets

$$\mathcal{A}_j = \Big\{ x : \ \text{dest}(x) = m_j \Big\}. \tag{3}$$

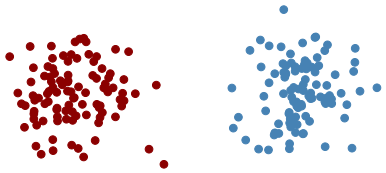

Figure 1: A synthetic example with two "blob-like" clusters.

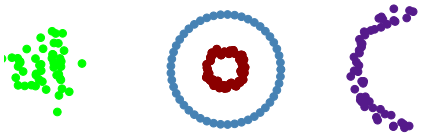

Figure 2: A synthetic example with four clusters with a variety of different shapes.

These sets are known as the *ascending manifolds*, and also known as the cluster associated with m_j, or the basin of attraction of m_j. The $\mathcal{A}_j$'s partition the space. See Figure 3. The collection of ascending manifolds is called the *Morse complex*.

Given data $X_1, \ldots, X_n$ we construct an estimate $\widehat{p}$ of the density. Let $\widehat{m}_1, \ldots, \widehat{m}_k$ be the estimated modes and let $\widehat{\mathcal{A}}_1, \ldots, \widehat{\mathcal{A}}_k$ be the corresponding ascending manifolds derived from $\widehat{p}$. The sample clusters $C_1, \ldots, C_k$ are defined to be $C_j = \{X_i : \ X_i \in \widehat{\mathcal{A}}_j\}$.

Recall that the kernel density estimator is

$$\widehat{p}(x) \equiv \widehat{p}_h(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{||x - X_i||}{h}\right) \tag{4}$$

where K is a smooth, symmetric kernel and $h > 0$ is the bandwidth.[1] The mean of the estimator is

$$p_h(x) = \mathbb{E}[\widehat{p}_h(x)] = \int K(t)p(x + th)dt. \tag{5}$$

To locate the modes of $\widehat{p}_h$ we use the *mean shift algorithm* which finds modes by approximating the steepest ascent paths. The algorithm is given in Figure 4. The result of this process is the set of estimated modes $\widehat{\mathcal{M}} = \{\widehat{m}_1, \ldots, \widehat{m}_k\}$. We also get the clustering for free: the mean shift algorithm shows us what mode each point is attracted to. See Figure 5.

A modified version of the algorithm is the blurred mean-shift algorithm (Carreira-Perpinan, 2006). Here, we use the data as the mesh and we replace the data with the mean-shifted data at each step. This converges very quickly but must be stopped before everything converges to a single point; see Figures 6 and 7.

[1] In general, we can use a bandwidth matrix H in the estimator, with $\widehat{p}(x) \equiv \widehat{p}_H(x) = \frac{1}{n}\sum_{i=1}^{n} K_H(x - X_i)$ where $K_H(x) = |H|^{-\frac{1}{2}} K(H^{-\frac{1}{2}}x)$.

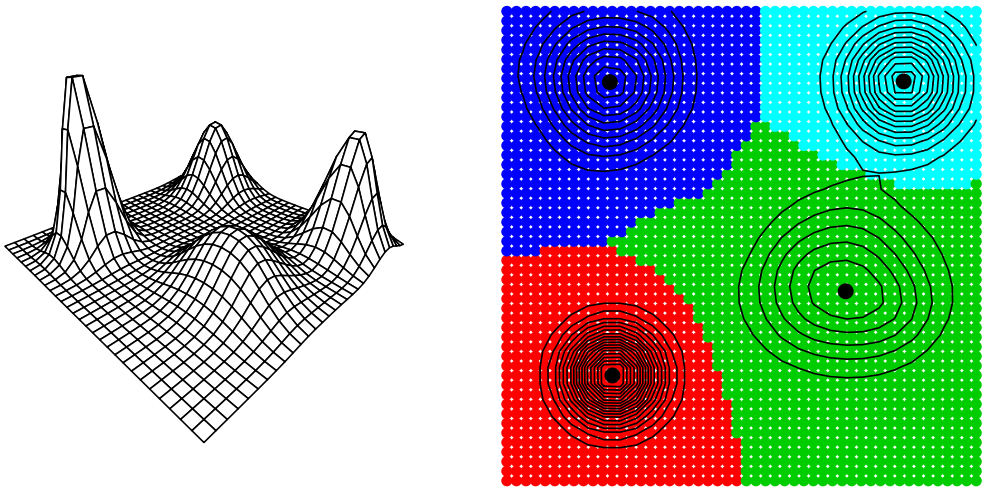

Figure 3: *The left plot shows a function with four modes. The right plot shows the ascending manifolds (basins of attraction) corresponding to the four modes.*

What we are doing is tracing out the *gradient flow*. The flow lines lead to the modes and they define the clusters. In general, a flow is a map $\phi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ such that $\phi(x, 0) = x$ and $\phi(\phi(x, t), s) = \phi(x, s + t)$. The latter is called the semi-group property.

2.1 Choosing the Bandwidth

As usual, choosing a good bandwidth is crucial. You might wonder if increasing the bandwidth, decreases the number of modes. Silverman (1981) showed that the answer is yes if you use a Normal kernel.

Theorem 2 (Silverman 1981) *Let $\widehat{p}_h$ be a kernel density estimator using a Gaussian kernel in one dimension. Then the number of modes of $\widehat{p}_h$ is a non-increasing function of h. The Gaussian kernel is the unique kernel with this property.*

We still need a way to pick h. We can use cross-validation as before. One could argue that we should choose h so that we estimate the gradient $g(x) = \nabla p(x)$ well since the clustering is based on the gradient flow.

How can we estimate the loss of the gradient? Consider, first the scalar case. Note that

$$\int (\widehat{p}' - p')^2 = \int (\widehat{p}')^2 - 2\int \widehat{p}'\, p' + \int (p')^2.$$

Mean Shift Algorithm

1. Input: $\widehat{p}(x)$ and a mesh of points $A = \{a_1, \ldots, a_N\}$ (often taken to be the data points).
2. For each mesh point a_j, set $a_j^{(0)} = a_j$ and iterate the following equation until convergence:

$$a_j^{(s+1)} \longleftarrow \frac{\sum_{i=1}^n X_i K\left(\frac{||a_j^{(s)} - X_i||}{h}\right)}{\sum_{i=1}^n K\left(\frac{||a_j^{(s)} - X_i||}{h}\right)}.$$

3. Let $\widehat{\mathcal{M}}$ be the unique values of the set $\{a_1^{(\infty)}, \ldots, a_N^{(\infty)}\}$.
4. Output: $\widehat{\mathcal{M}}$.

Figure 4: *The Mean Shift Algorithm.*

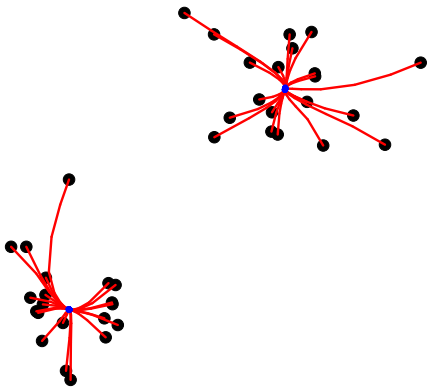

Figure 5: A simple example of the mean shift algorithm.

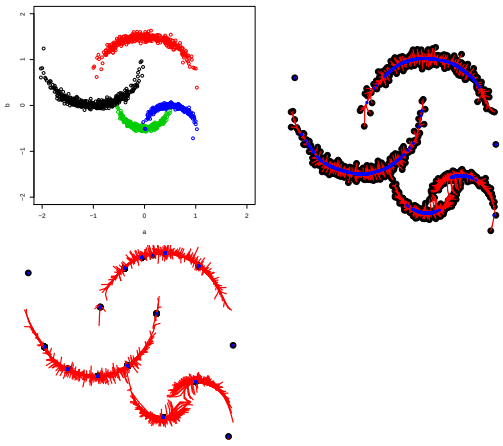

Figure 6: The crescent data example. Top left: data. Top right: a few steps of mean-shift. Bottom left: a few steps of blurred mean-shift.

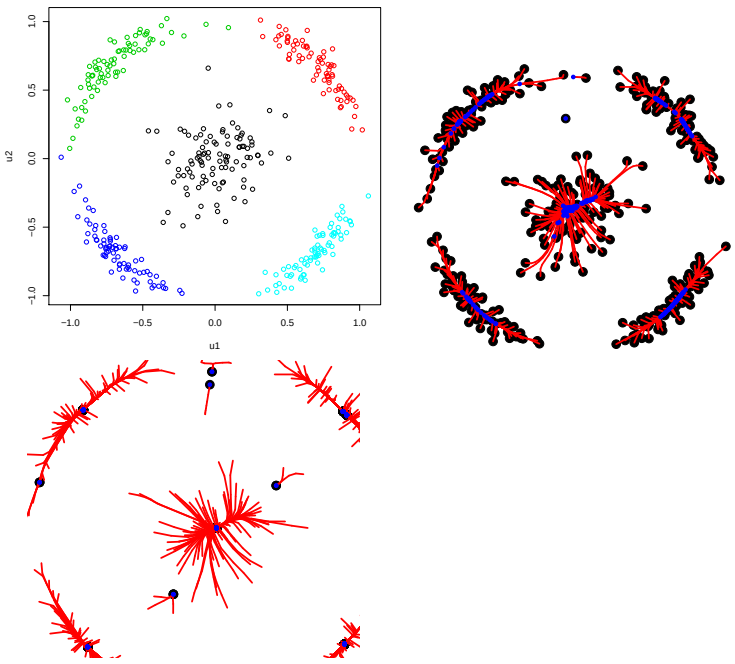

Figure 7: The Broken Ring example. Top left: data. Top right: a few steps of mean-shift. Bottom left: a few steps of blurred mean-shift.

We can ignore the last term. The first term is known. To estimate the middle term, we use integration by parts to get

$$\int \widehat{p}'\, p' = -\int p'' p$$

suggesting the cross-validation estimator

$$\int (\widehat{p}'(x))^2 dx + \frac{2}{n}\sum_i \widehat{p}_i''(X_i)$$

where $\widehat{p}_i''$ is the leave-one-out second derivative. More generally, by repeated integration by parts, we can estimate the loss for the r^{th} derivative by

$$\text{CV}_r(h) = \int (\widehat{p}^{(r)}(x))^2 dx - \frac{2}{n}(-1)^r \sum_i \widehat{p}_i^{(2r)}(X_i).$$

Let's now discuss estimating derivatives more generally following Chacon and Duong (2013). Let

$$\widehat{p}_H(x) = \frac{1}{n}\sum_{i=1}^n K_H(x - X_i)$$

where $K_H(x) = |H|^{-1/2}K(H^{-1/2}x)$. Let $D = \partial/\partial x = (\partial/\partial x_1, \ldots, \partial/\partial x_d)$ be the gradient operator. Let $H(x)$ be the Hessian of $p(x)$ whose entries are $\partial^2 p/(\partial x_j \partial x_k)$. Let

$$D^{\otimes r} p = (Dp)^{\otimes r} = \partial^r p/\partial x^{\otimes r} \in \mathbb{R}^{d^r}$$

denote the r^{th} derivatives, organized into a vector. Thus

$$D^{\otimes 0} p = p, \qquad D^{\otimes 1} p = Dp, \qquad D^{\otimes 2} p = \text{vec}(H)$$

where vec takes a matrix and stacks the columns into a vector.

The estimate of $D^{\otimes r} p$ is

$$\widehat{p}^{(r)}(x) = D^{\otimes r}\widehat{p}_H(x) = \frac{1}{n}\sum_{i=1}^n D^{\otimes r} K_H(x - X_i) = \frac{1}{n}\sum_{i=1}^n |H|^{-1/2}(H^{-1/2})^{\otimes r} D^{\otimes r} K(H^{-1/2}(x - X_i).$$

The integrated squared error is

$$L = \int ||D^{\otimes r}\widehat{p}_H(x) - D^{\otimes r} p(x)||^2 dx.$$

Chacon, Duong and Wand shows that $\mathbb{E}[L]$ is minimized by choosing H so that each entry has order $n^{-2/(d+2r+4)}$ leading to a risk of order $O(n^{-4/(d+2r+4)})$. In fact, it may be shown that

$$\begin{aligned}\mathbb{E}[L] = \frac{1}{n}|H|^{-1/2}\text{tr}((H^{-1})^{\otimes r} R(D^{\otimes r} K)) - \frac{1}{n}\text{tr}R^*(K_H \star K_H, D^{\otimes r} p) \\ + \text{tr}R^*(K_H \star K_H, D^{\otimes r} p) - 2\text{tr}R^*(K_H, D^{\otimes r} p) + \text{tr}R(D^{\otimes r} p)\end{aligned}$$

where

$$R(g) = \int g(x)g^T(x)dx$$

$$R^*(a,g) = \int (a \star g)(x)g^T(x)dx$$

and $(a \star g)$ is componentwise convolution.

To estimate the loss, we expand L as

$$L = \int ||D^{\otimes r}\widehat{p}_H(x)||^2 dx - 2\int \langle D^{\otimes r}\widehat{p}_H(x), D^{\otimes r}p(x)\rangle dx + \text{constant}.$$

Using some high-voltage calculations, Chacon and Duong (2013) derived the following leave-one-out approximation to the first two terms:

$$\text{CV}_r(H) = (-1)^r |H|^{-1/2}(\text{vec}(H^{-1})^{\otimes r})^T B(H)$$

where

$$B(H) = \frac{1}{n^2}\sum_{i,j} D^{\otimes 2r}\overline{K}(H^{-1/2}(X_i - X_j)) - \frac{2}{n(n-1)}\sum_{i\neq j} D^{\otimes 2r}K(H^{-1/2}(X_i - X_j))$$

and $\overline{K} = K \star K$ In practice, the minimization is easy if we restrict to matrices of the form $H = h^2 I$.

A better idea is to used fixed (non-decreasing h). We don't need h to go to 0 to find the clusters. More on this when we discuss persistence.

2.2 Theoretical Analysis

How well can we estimate the modes?

Theorem 3 *Assume that p is Morse with finitely many modes $m_1, \ldots, m_k$. Then for $h > 0$ and not too large, p_h is Morse with modes $m_{h1}, \ldots, m_{hk}$ and (possibly after relabelling),*

$$\max_j ||m_j - m_{jh}|| = O(h^2).$$

With probability tending to 1, $\widehat{p}_h$ has the same number of modes which we denote by $\widehat{m}_{h1}, \ldots, \widehat{m}_{hk}$. Furthermore,

$$\max_j ||\widehat{m}_{jh} - m_{jh}|| = O_P\left(\sqrt{\frac{1}{nh^{d+2}}}\right)$$

and

$$\max_j ||\widehat{m}_{jh} - m_j|| = O(h^2) + O_P\left(\sqrt{\frac{1}{nh^{d+2}}}\right).$$

Remark: Setting $h \asymp n^{-1/(d+6)}$ gives the rate $n^{-2/(d+6)}$ which is minimax (Tsyabkov 1990) under smoothness assumptions. See also Romano (1988). However, if we take the fixed h point if view, then we have a $n^{-1/2}$ rate.

Proof Outline. But a small ball B_j around each m_{jh}. We will skip the first step, which is to show that there is one (and only one) local mode in B_j. Let's focus on showing

$$\max_j ||\widehat{m}_{jh} - m_{jh}|| = O_P\left(\sqrt{\frac{1}{nh^{d+2}}}\right).$$

For simplicity, write $m = m_{jh}$ and $x = \widehat{m}_{jh}$. Let $g(x)$ and $H(x)$ be the gradient and Hessian of $p_h(x)$ and let $\widehat{g}(x)$ and $\widehat{H}(x)$ be the gradient Hessian of $\widehat{p}_h(x)$. Then

$$(0,\ldots,0)^T = \widehat{g}(x) = \widehat{g}(m) + (x-m)^T \int_0^1 \widehat{H}(m + u(x-m))du$$

and so

$$(x-m)^T \int_0^1 \widehat{H}(m + u(x-m))du = (g(m) - \widehat{g}(m))$$

where we used the fact that $\mathbf{0} = g(m)$. Multiplying on the right by $x - m$ we have

$$(x-m)^T \int_0^1 \widehat{H}(m + u(x-m))(x-m)du = (\widehat{g}(m) - \widehat{g}(m))^T(x-m).$$

Let $\lambda = \inf_{0\leq u\leq 1} \lambda_{\min}(H(m + u(x-m)))$. Then $\lambda = \lambda_{\min}(H(m)) + o_P(1)$ and

$$(x-m)^T \int_0^1 \widehat{H}(x + u(m-x))(x-m)du \geq \lambda||x-m||^2.$$

Hence, using Cauchy-Schwartz,

$$\lambda||x-m||^2 \leq ||\widehat{g}(m)-g(m)||\,||x-m|| \leq ||x-m||\sup_y ||\widehat{g}(y)-\widehat{g}(y)|| \leq ||x-m|| O_P\left(\sqrt{\frac{1}{nh^{d+2}}}\right)$$

and so $||x-m|| = O_P\left(\sqrt{\frac{1}{nh^{d+2}}}\right)$. □

Remark: If we treat h as fixed (not decreasing) then the rate is $O_P(\sqrt{1/n})$ independent of dimension.

3 Level Set Clustering

An alternative to mode clustering is *level set clustering*. Let $L_t = \{x : \ p_h(x) > t\}$ denote an upper level set of p. Suppose that L_t can be decomposed into finitely many disjoint sets: $L_t = C_1 \bigcup \cdots \bigcup C_{k_t}$. We call $\mathcal{C}_t = \{C_1, \ldots, C_{k_t}\}$ the level set clusters at level t.

Let $\mathcal{C} = \bigcup_{t \geq 0} \mathcal{C}_t$. The clusters in $\mathcal{C}$ form a tree: if $A, B \in \mathcal{C}$, the either (i) $A \subset B$ or (ii)$B \subset A$ or (iii) $A \cap B = \emptyset$. We call $\mathcal{C}$ the *level set cluster tree*.

The level sets can be estimated in the obvious way: $\widehat{L}_t = \{x : \ \widehat{p}_h(x) > t\}$. How do we decompose $\widehat{L}_t$ into its connected components? This can be done as follows. For each t let

$$\mathcal{X}_t = \{X_i : \ \widehat{p}_h(X_i) > t\}.$$

Now construct a graph G_t where each $X_i \in \mathcal{X}_t$ is a vertex and there is an edge between X_i and X_j if and only if $||X_i - X_j|| \leq \epsilon$ where $\epsilon > 0$ is a tuning parameter. Bobrowski et al (2104) show that we can take $\epsilon = h$. G_t is a called a Rips graphs. The clusters at level t are estimated by taking the connected components of the graph G_t. In summary:

1. Compute $\widehat{p}_h$.
2. For each t, let $\mathcal{X}_t = \{X_i : \ \widehat{p}_h(X_i) > t\}$.
3. Form a graph G_t for the points in $\mathcal{X}_t$ by connecting X_i and X_j if $||X_i - X_j|| \leq h$.
4. The clusters at level t are the connected components of G_t.

A Python package, called DeBaCl, written by Brian Kent, can be found at

`http://www.brianpkent.com/projects.html`.

Fabrizio Lecci has written an R implementation, include in his R package: TDA (topological data analysis). You can get it at:

`http://cran.r-project.org/web/packages/TDA/index.html`

Two examples are shown in Figures 8 and 9.

3.1 Theory

How well does this work? Define the Hausdorff distance between two sets by

$$H(U, V) = \inf\left\{\epsilon : \ U \subset V \oplus \epsilon \ \text{ and } \ V \subset U \oplus \epsilon\right\}$$

where

$$V \oplus \epsilon = \bigcup_{x \in V} B(x, \epsilon)$$

and $B(x, \epsilon)$ denotes a ball of radius ϵ centered at x. We would like to say that L_t and $\widehat{L}_t$ are close. In general this is not true. Sometimes L_t and $L_{t+\delta}$ are drastically different even for small δ. (Think of the case where a mode has height t.) But we can estimate stable level sets. Let us say that L_t is stable if there exists $a > 0$ and $C > 0$ such that, for all $\delta < a$,

$$H(L_{t-\delta}, L_{t+\delta}) \leq C\delta.$$

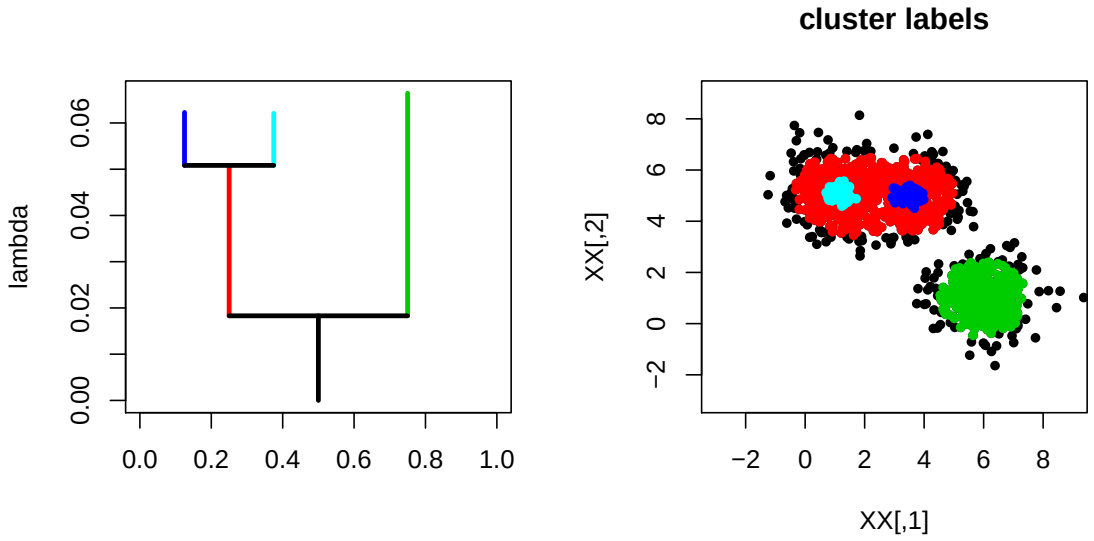

Figure 8: DeBaClR in two dimensions.

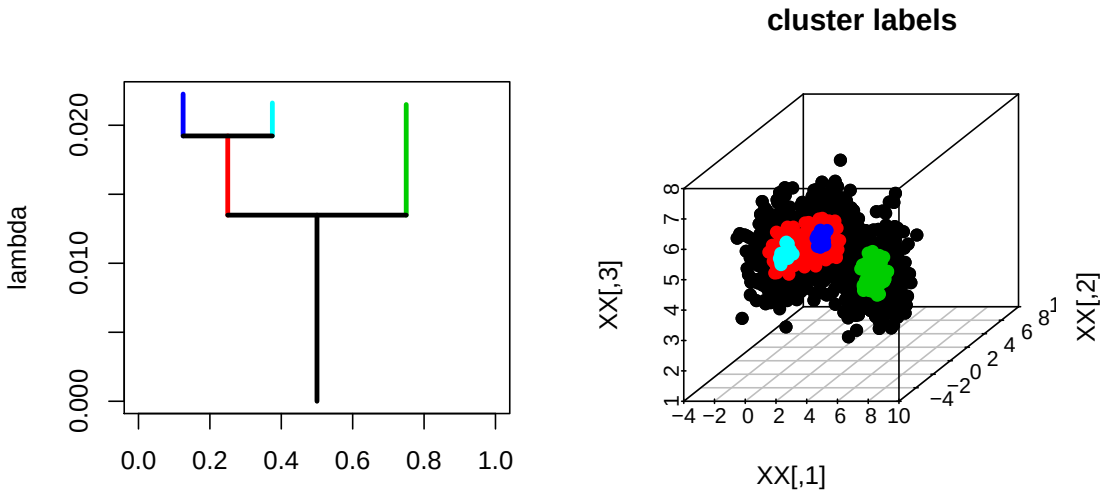

Figure 9: DeBaClR in three dimensions.

Theorem 4 *Suppose that L_t is stable. Then $H(\widehat{L}_t, L_t) = O_P(\sqrt{\log n/(nh^d)})$.*

Proof. Let $r_n = \sqrt{\log n/(nh^d)})$. We need to show two things: (i) for every $x \in L_t$ there exists $y \in \widehat{L}_t$ such that $||x - y|| = O_P(r_n)$ and (ii) for every $x \in \widehat{L}_t$ there exists $y \in L_t$ such that $||x - y|| = O_P(r_n)$. First, we note that, by earlier results, $||\widehat{p}_h - p_h||_\infty = O_P(r_n)$. To show (i), suppose that $x \in L_t$. By the stability assumption, there exists $y \in L_{t+r_n}$ such that $||x - y|| \leq Cr_n$. Then $p_h(y) > t + r_n$ which implies that $\widehat{p}_h(y) > t$ and so $y \in \widehat{L}_t$. To show (ii), let $x \in \widehat{L}_t$ so that $\widehat{p}_h(x) > t$. Thus $p_h(x) > t - r_n$. By stability, there is a $y \in L_t$ such that $||x - y|| \leq Cr_n$. □

3.2 Persistence

Consider a smooth density p with $M = \sup_x p(x) < \infty$. The t-level set clusters are the connected components of the set $L_t = \{x : \ p(x) \geq t\}$. Suppose we find the upper level sets $L_t = \{x : \ p(x) \geq t\}$ as we vary t from M to 0. *Persistent homology* measures how the topology of L_t varies as we decrease t. In our case, we are only interested in the modes, which correspond to the zeroth order homology. (Higher order homology refers to holes, tunnels etc.) The idea of using persistence to study clustering was introduced by Chazal, Guibas, Oudot and Skraba (2013).

Imagine setting $t = M$ and then gradually decreasing t. Whenever we hit a mode, a new level set cluster is born. As we decrease t further, some clusters may merge and we say that one of the clusters (the one born most recently) has died. See Figure 10.

In summary, each mode m_j has a death time and a birth time denoted by (d_j, b_j). (Note that the birth time is larger than the death time because we start at high density and move to lower density.) The modes can be summarized with a persistence diagram where we plot the points $(d_1, b_1), \ldots, (d_k, b_k)$ in the plane. See Figure 10. Points near the diagonal correspond to modes with short lifetimes. We might kill modes with lifetimes smaller than the bootstrap quantile ϵ_α defined by

$$\epsilon_\alpha = \inf\left\{ z : \ \frac{1}{B}\sum_{b=1}^{B} I\Big(||\widehat{p}_h^{*b} - \widehat{p}_h||_\infty > z\Big) \leq \alpha \right\}. \tag{6}$$

Here, $\widehat{p}_h^{*b}$ is the density estimator based on the b^{th} bootstrap sample. This corresponds to killing a mode if it is in a $2\epsilon_\alpha$ band around the diagonal. See Fasy, Lecci, Rinaldo, Wasserman, Balakrishnan and Singh (2014). Note that the starting and ending points of the vertical bars on the level set tree are precisely the coordinates of the persistence diagram. (A more precise bootstrap approach was introduced in Chazal, Fasy, Lecci, Michel, Rinaldo and Wasserman (2104).)

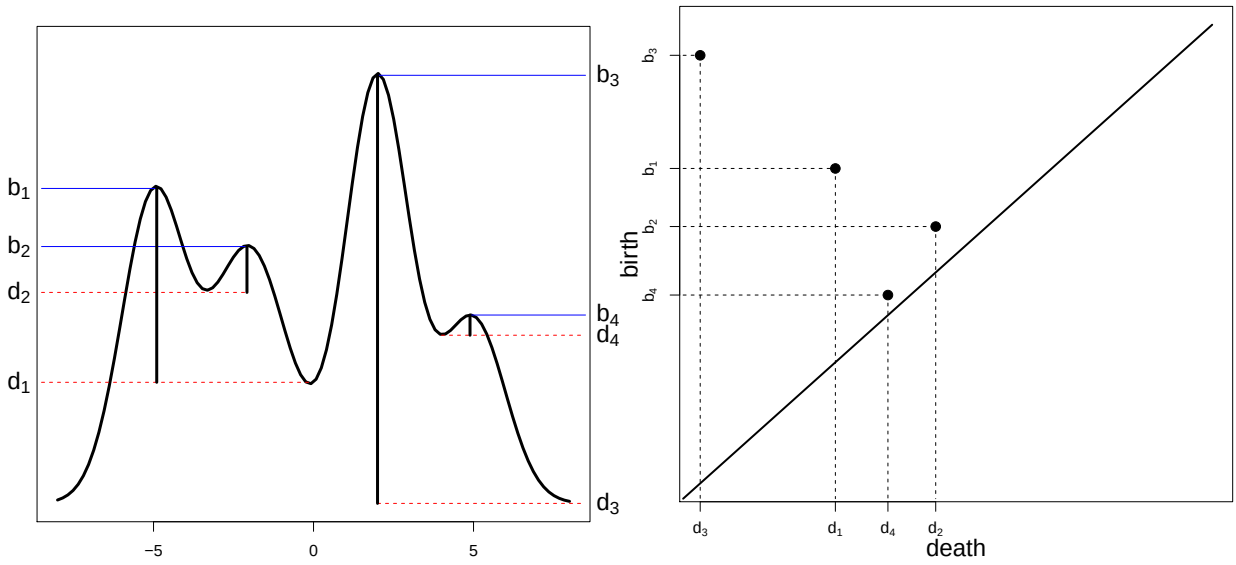

Figure 10: *Starting at the top of the density and moving down, each mode has a birth time b and a death time d. The persistence diagram (right) plots the points $(d_1, b_1), \ldots, (d_4, b_4)$. Modes with a long lifetime are far from the diagonal.*

4 k-means (Vector Quantization)

One of the oldest approaches 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:

$$\Pi_C[X] = \text{argmin}_{c \in C} ||c - X||^2. \tag{7}$$

Define the empirical clustering risk of a codebook C by

$$R_n(C) = \frac{1}{n} \sum_{i=1}^{n} \Big|\Big| X_i - \Pi_C[X_i] \Big|\Big|^2 = \frac{1}{n} \sum_{i=1}^{n} \min_{1 \le j \le k} ||X_i - c_j||^2. \tag{8}$$

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

$$\widehat{C} = \text{argmin}_{C \in \mathcal{C}_k} R_n(C). \tag{9}$$

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

$$R(C) = \mathbb{E}\Big|\Big| X - \Pi_C[X] \Big|\Big|^2 = \mathbb{E} \min_{1 \le j \le k} ||X - c_j||^2 \tag{10}$$

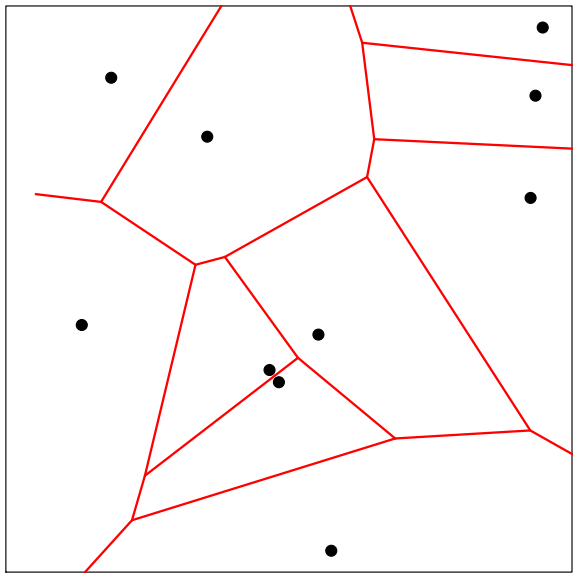

Figure 11: The Voronoi tesselation formed by 10 cluster centers $c_1, \ldots, c_{10}$. The cluster centers are indicated by dots. The corresponding Voronoi cells $T_1, \ldots, T_{10}$ are defined as follows: a point x is in T_j if x is closer to c_j than c_i for $i \neq j$.

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

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

$$T_j = \Big\{ x : \ ||x - c_j|| \leq ||x - c_s||, \quad \text{for all } s \neq j \Big\}. \tag{11}$$

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 11.

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 12. 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 4.1.

Example 5 *Figure 13 shows synthetic data inspired by the Mickey Mouse example from*

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

`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.

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 14.*

Example 7 (Supernova Clustering) *Figure 15 shows supernova data where 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 16 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.*

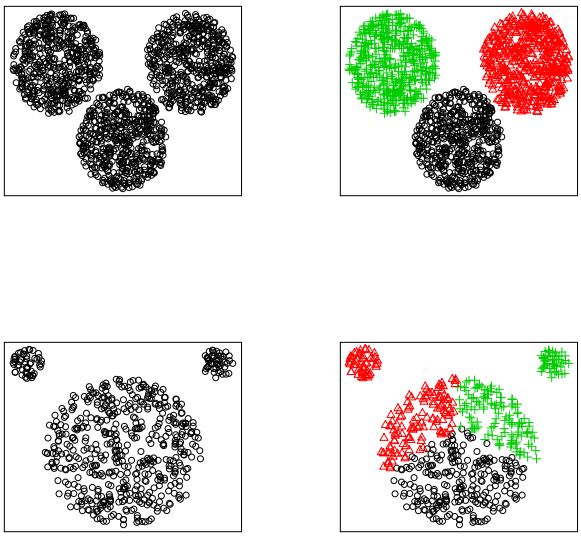

Figure 13: Synthetic data inspired by the "Mickey Mouse" example from wikipedia. Top left: three balanced clusters. Top right: result from running k means with $k = 3$. Bottom left: three unbalanced clusters. Bottom right: result from running k means with $k = 3$ on the unbalanced clusters. k-means does not work well here because the clusters are very unbalanced.

4.1 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 17 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 $k = 9$ 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 6) 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 17.

k-means^{++}. Arthur and Vassilvitskii (2007) invented an algorithm called k-means^{++} to get

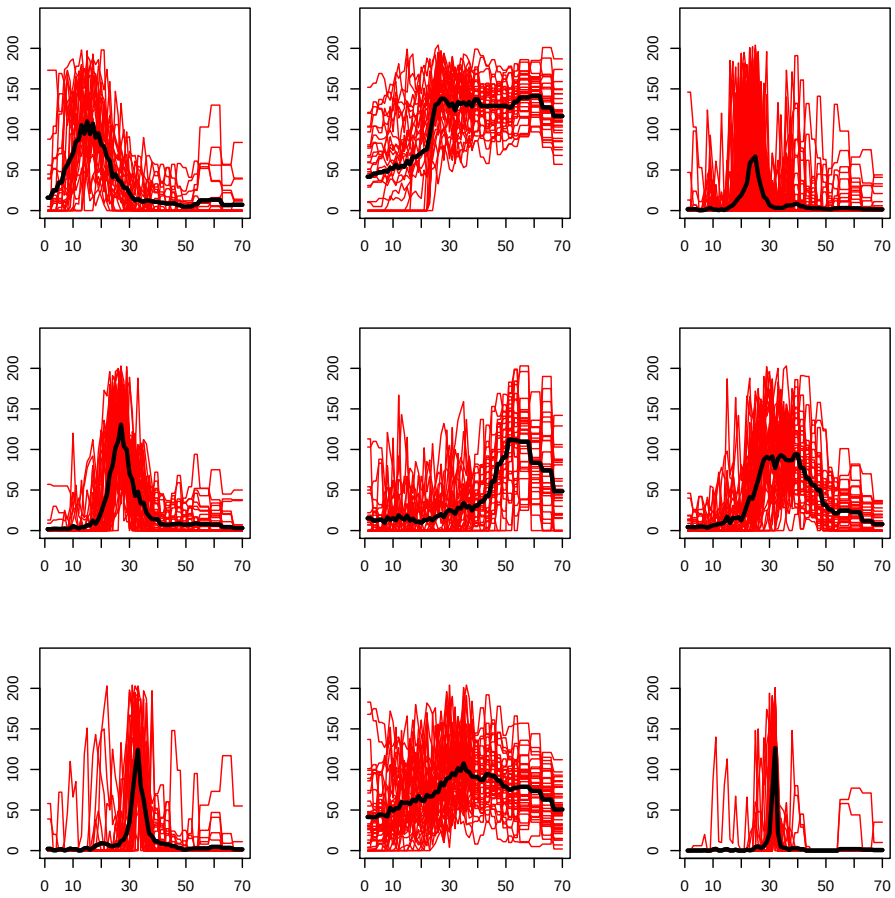

Figure 14: The nine clusters found in the Topex data using k-means clustering with $k = 9$. Each plot show the curves in that cluster together with the mean of the curves in that cluster.

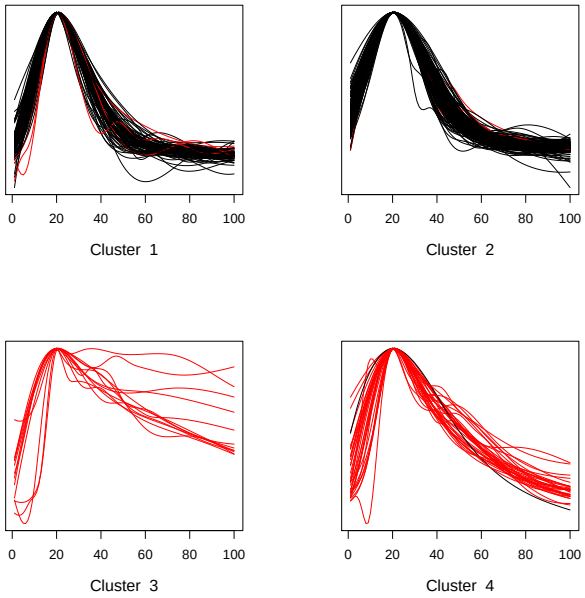

Figure 15: Clustering of the supernova light curves with $k = 4$.

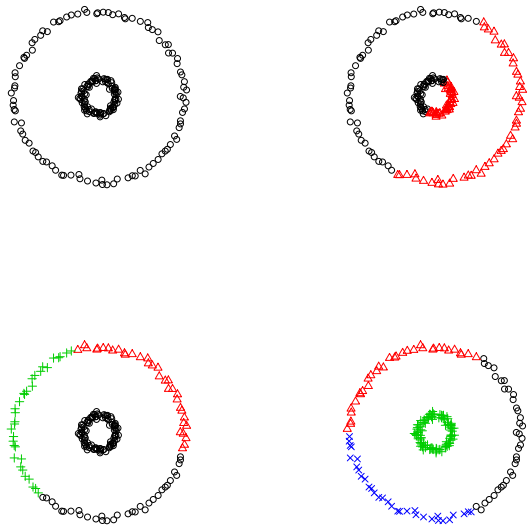

Figure 16: Top left: a dataset with two ring-shaped clusters. Top right: k-means with $k = 2$. Bottom left: k-means with $k = 3$. Bottom right: k-means with $k = 4$.

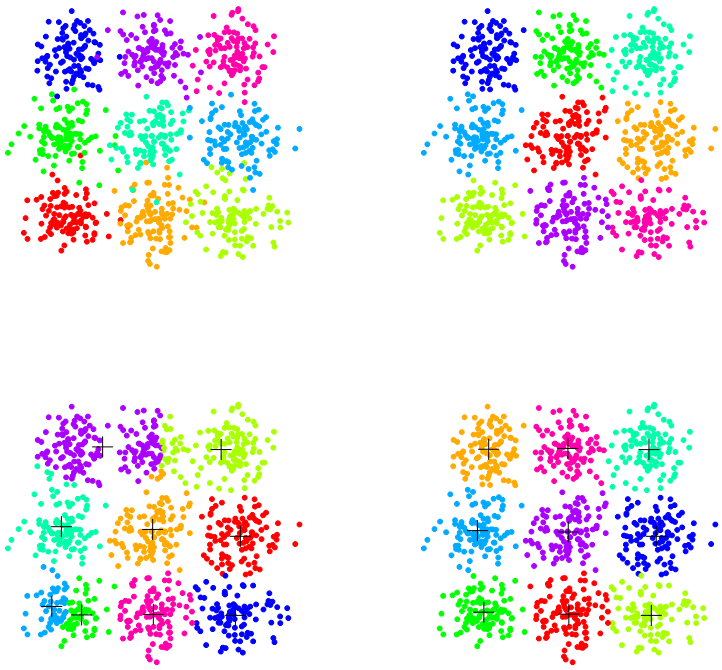

Figure 17: An example with 9 clusters. Top left: data. Top right: k-means with random starting values. Bottom left: k-means using starting values from hierarchical clustering. Bottom right: the k-means^{++} algorithm.

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 algorithm is described in Figure 18. See the bottom right plot in Figure 17 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,*

$$\mathbb{E}\big(R_n(C)\big) \leq 8(\log k + 2) \ \Big(\min_C R_n(C)\Big) \tag{12}$$

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 (12) can be replaced by a constant. They call the algorithm, k-means#.

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 18: The k-means^{++} algorithm.

4.2 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

$$k_\alpha = \min \left\{ k : \ \frac{R_k - R_{k+1}}{\sigma^2} \leq \alpha \right\} \tag{13}$$

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

$$\widehat{k}_\alpha = \min\left\{k : \ \frac{\widehat{R}_k - \widehat{R}_{k+1}}{\widehat{\sigma}^2} \leq \alpha\right\} \tag{14}$$

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.

Currently, my favorite approach is the one in Liu, Hayes, Andrew Nobel and Marron (2012). (JASA, 2102, 1281-1293). They simply test if the data are multivariate Normal. If this rejects, they split into two clusters and repeat. The have an R package `sigclust` for this. A similar procedure, called PG means is described in Feng and Hammerly (2007).

Example 11 *Figure 19 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*

$$\Omega(k) = \mathbb{E}\left[s(A_k(Y), A_k(Z))\right] \tag{15}$$

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

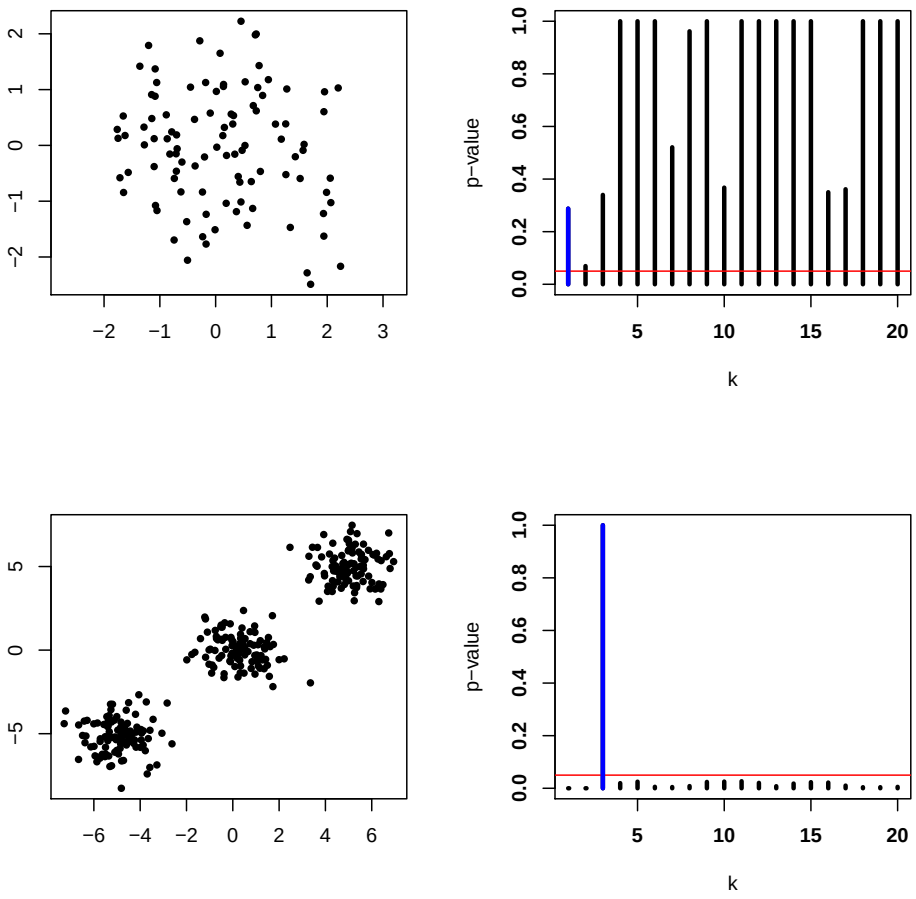

Figure 19: Top left: a single cluster. Top right: p-values for various k. The first k for which the p-value is larger than .05 is $k = 1$. Bottom left: three clusters. Bottom right: p-values for various k. The first k for which the p-value is larger than .05 is $k = 3$.

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 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

$$\psi_{Y,Z}(Z_j, Z_k) = \psi_Y(Y'_j, Y'_k) \tag{16}$$

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 20 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.

4.3 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*

$$\mathbb{E}(R(\widehat{C})) - R(C^*) \leq c\sqrt{\frac{k(d+1)\log n}{n}} \tag{17}$$

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.

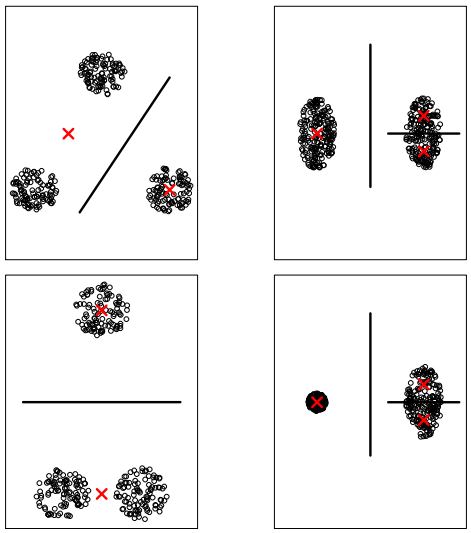

Figure 20: Examples from Ben-David, von Luxburg and Pál (2006). The first example (top left plot) shows a case where we fit $k = 2$ clusters. Stability analysis will correctly show that k is too small. The top right plot has $k = 3$. Stability analysis will correctly show that k is too large. The bottom 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.

Proof. Note that $R(\widehat{C}) - R(C^*) = R(\widehat{C}) - R_n(\widehat{C}) + R_n(\widehat{C}) - R(C^*) \leq R(\widehat{C}) - R_n(\widehat{C}) + R_n(C^*) - R(C^*) \leq 2\sup_{C\in\mathcal{C}_k} |R(\widehat{C}) - R_n(\widehat{C})|$. 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

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

where A varies over all sets $\mathcal{A}$ of the form $\{f_C(x) > u\}$. The shattering number of $\mathcal{A}$ is $s(\mathcal{A}, n) \leq n^{k(d+1)}$. 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* $\mathbb{P}(\|X\|^2 \leq 1) = 1$ *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*

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

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.

5 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 6) and spectral clustering (Section 7).

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 7. 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 21 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 22 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 23 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 24 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 **??**.

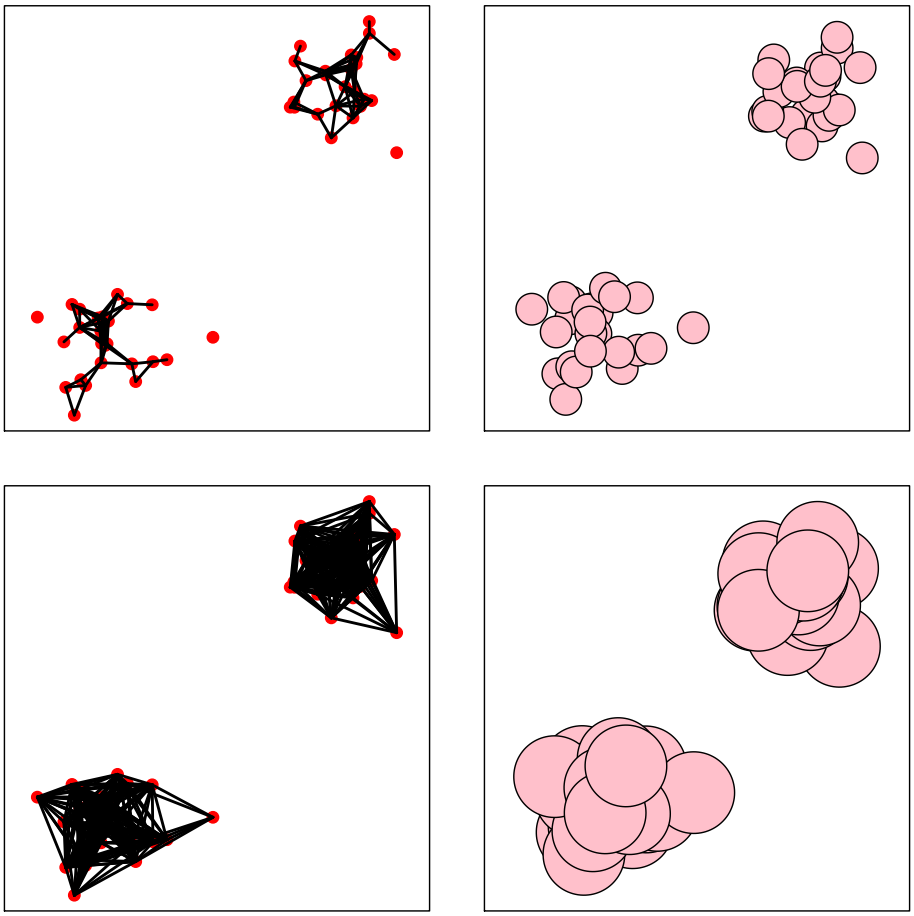

Figure 21: Geometric graph clustering. Top row: $\epsilon = 1$. Bottom row: $\epsilon = 3$. Left column: the graphs. Right column: a ball of radius $\epsilon/2$ around each X_i.

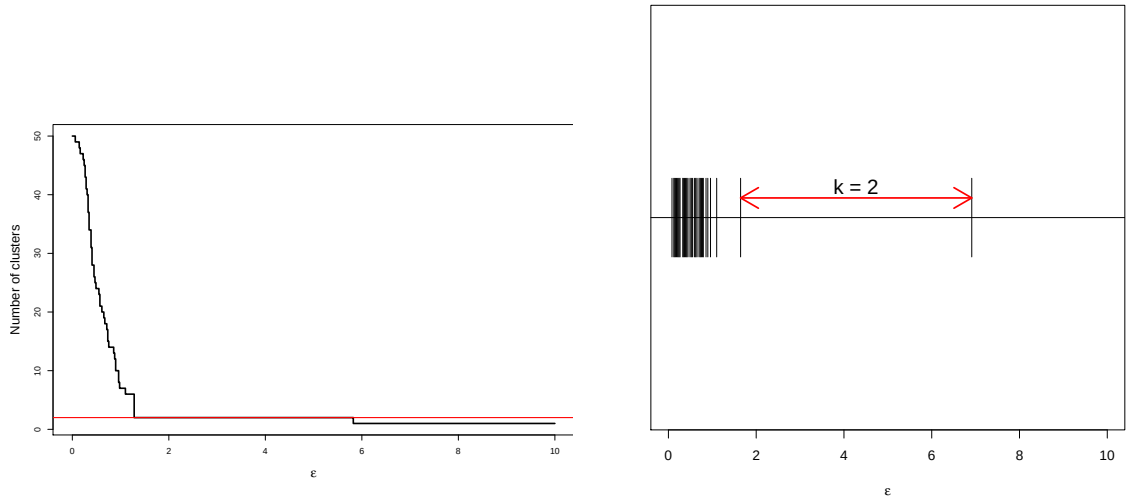

Figure 22: Geometric graph clustering. The top plot shows the number of connected components $k(\epsilon)$ versus ϵ. There is a fairly large range of values of ϵ that yields two connected components. The bottom plot shows the values of ϵ at which $k(\epsilon)$ changes.

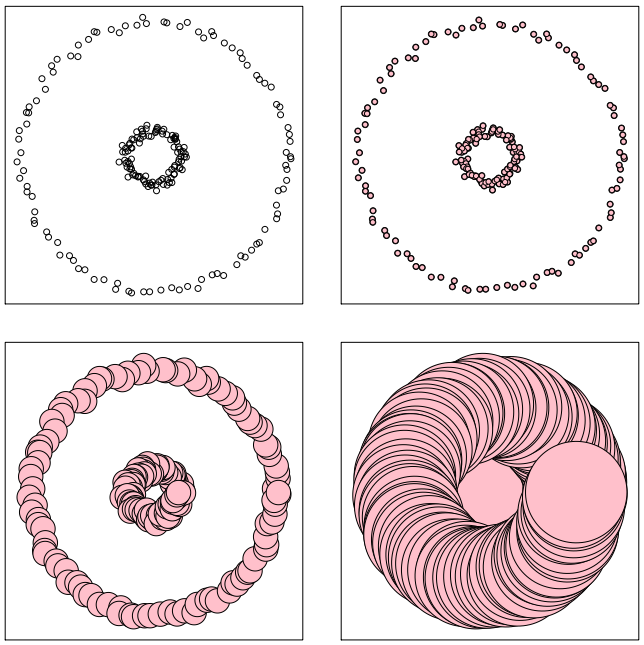

Figure 23: Geometric graph clustering applied to two ring-shaped clusters. Top left: data. Top right: $\epsilon = .2$. Bottom left: $\epsilon = 1.0$. Bottom right: $\epsilon = 6$.

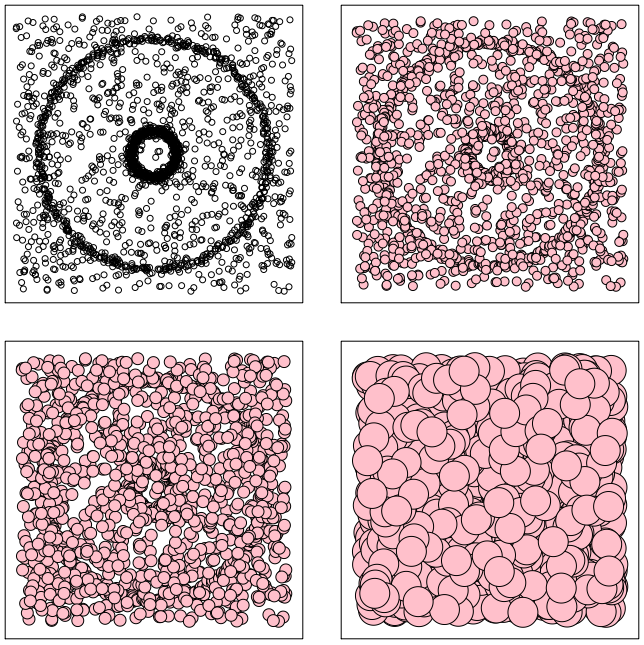

Figure 24: Geometric graph clustering applied to two ring-shaped clusters with background noise. Top left: data. Top right: $\epsilon = 0.40$. Bottom left: $\epsilon = 0.54$. Bottom right: $\epsilon = 1.46$.

6 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). 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:

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	$d(A,B) = \max_{x \in A, y \in B} d(x,y)$

The algorithm is:

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 $T_1, \ldots, T_n$.

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 25 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 $X_1, \ldots, X_n$ 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.

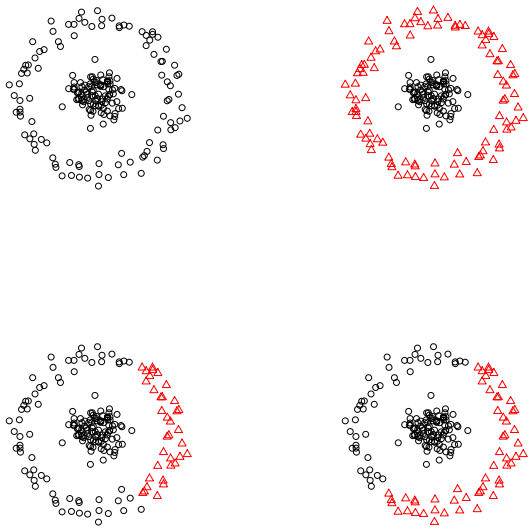

Figure 25: Hierarchical clustering applied to two noisy rings. Top left: the data. Top right: two clusters from hierarchical clustering using single linkage. Bottom left: average linkage. Bottom right: complete linkage.

7 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 5. 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

$$L = D - W. \tag{18}$$

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 $\mathcal{L}(v) = \{cv : \ c \in \mathbb{R}, \ c \neq 0\}$ 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, $\mathcal{L}(v)$ 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 $v \in \mathcal{L}(v_1)$.

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, $v \in \mathcal{L}(v_j)$ for some $1 \leq j \leq k$. □

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

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

and the Laplacian is

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

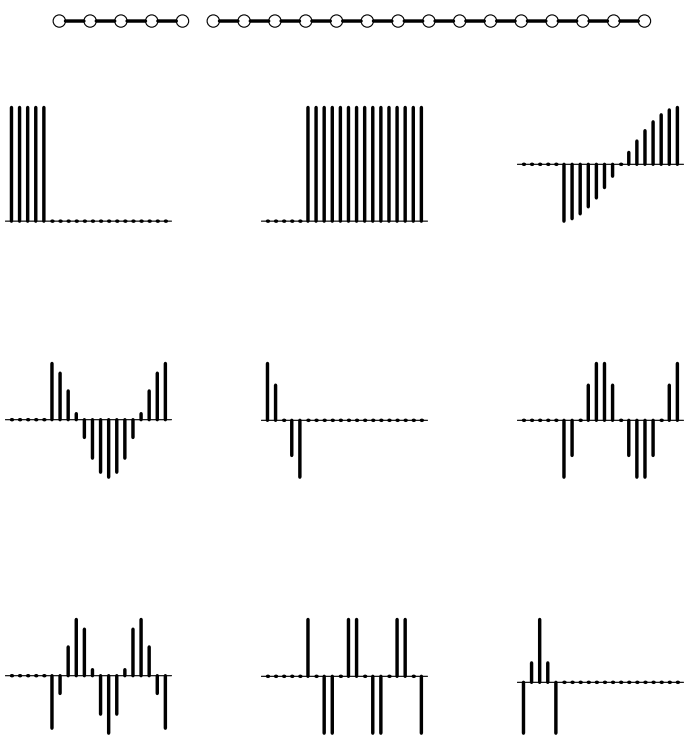

Figure 26: The top shows a simple graph. The remaining plots are the eigenvectors of the graph Laplacian. Note that the first two eigenvectors correspond to the two connected components of the graph.

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

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

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

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 26 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 5. 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 $\mathcal{L} = D^{-1/2}WD^{-1/2}$ and the random walk Laplacian $\mathcal{L} = D^{-1}W$. (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 $\mathcal{L} = D^{-1}W$ 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 $\mathcal{L}$ be the random walk Laplacian.*

1. *λ is an eigenvalue of $\mathcal{L}$ with eigenvector v if and only if $Lv = (1 - \lambda)Dv$.*
2. *1 is an eigenvalue of $\mathcal{L}$ with eigenvector $(1, \ldots, 1)^T$.*
3. *$\mathcal{L}$ is positive semidefinite with n non-negative real-valued eigenvalues.*
4. *The number of eigenvalues of $\mathcal{L}$ 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. □ **H**

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathcal{L}$ 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 $\mathcal{L}$ 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

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

The Laplacian $\mathcal{L}(i,j)$ 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 $\mathcal{L}$ 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 $\mathcal{L} = D^{-1}W$.
3. Find first k eigenvectors $v_1, \ldots, v_k$ of $\mathcal{L}$.
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 27 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*

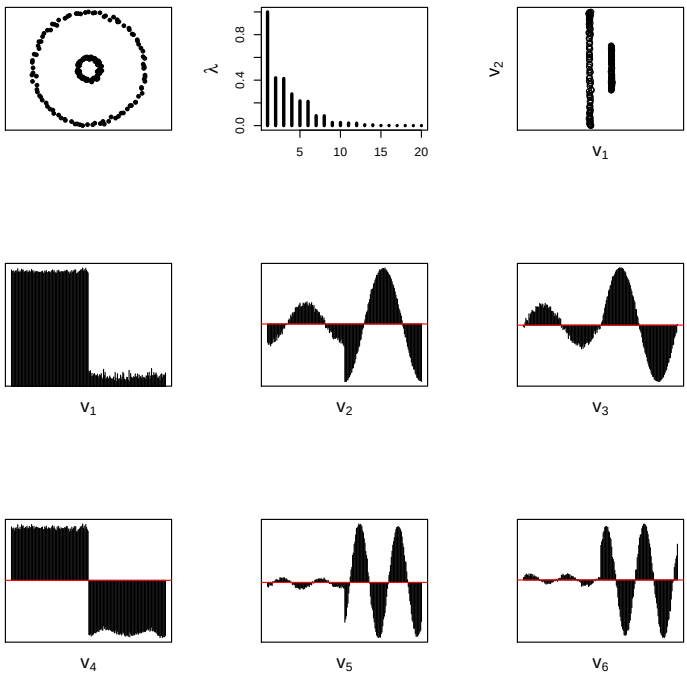

Figure 27: Top left: data. Top middle: eigenvalues. Top right: second versus third eigenvectors. Remaining plots: first six eigenvectors.

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, 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 28 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*

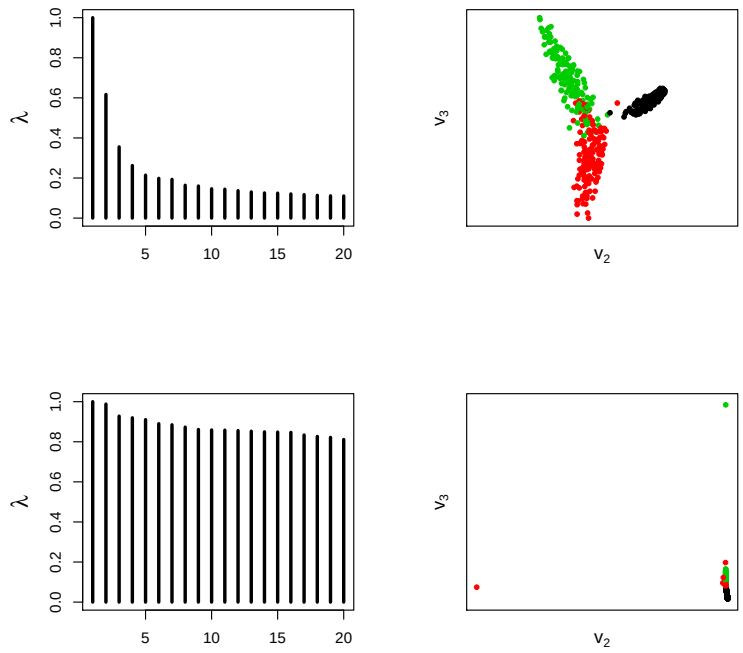

Figure 28: Spectral analysis of some zipcode data. Top: $h = 6$. Bottom: $h = 4$. The plots on the right show the second versus third eigenvector. The three colors correspond to the three digits 1, 2 and 3.

by two eignvectors so the effective dimension is 2. This example also shows that the choice of h is crucial.

Spectral methods are interesting. 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?

8 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.

8.1 Marginal Selection

In marginal selection, we look for variables that marginally look 'clustery." This idea was used in Chan and Hall (2010) and Wasserman, Azizyan and Singh (2014). We proceed as follows:

Test For Multi-Modality

1. Fix $0 < \alpha < 1$. Let $\widetilde{\alpha} = \alpha/(nd)$.
2. For each $1 \leq j \leq d$, compute $T_j = \text{Dip}(F_{nj})$ where F_{nj} is the empirical distribution function of the j^{th} feature and $\text{Dip}(F)$ is defined in (19).
3. Reject the null hypothesis that feature j is not multimodal if $T_j > c_{n,\widetilde{\alpha}}$ where $c_{n,\widetilde{\alpha}}$ is the critical value for the dip test.

Any test of multimodality may be used. Here we describe the *dip test* (Hartigan and Hartigan, 1985). Let $Z_1, \ldots, Z_n \in [0,1]$ be a sample from a distribution F. We want to test "$H_0 : F$ is unimodal" versus "$H_1 : F$ is not unimodal." Let $\mathcal{U}$ be the set of unimodal distributions. Hartigan and Hartigan (1985) define

$$\text{Dip}(F) = \inf_{G \in \mathcal{U}} \sup_x |F(x) - G(x)|. \tag{19}$$

If F has a density p we also write $\text{Dip}(F)$ as $\text{Dip}(p)$. Let F_n be the empirical distribution function. The dip statistic is $T_n = \text{Dip}(F_n)$. The dip test rejects H_0 if $T_n > c_{n,\alpha}$ where the critical value $c_{n,\alpha}$ is chosen so that, under H_0, $\mathbb{P}(T_n > c_{n,\alpha}) \leq \alpha$.[2]

Since we are conducting multiple tests, we cannot test at a fixed error rate α. Instead, we replace α with $\widetilde{\alpha} = \alpha/(nd)$. That is, we test each marginal and we reject H_0 if $T_n > c_{n,\widetilde{\alpha}}$. By the union bound, the chance of at least one false rejection of H_0 is at most $d\widetilde{\alpha} = \alpha/n$.

There are more refined tests such as the excess mass test given in Chan and Hall (2010), building on work by Muller and Sawitzki (1991). For simplicity, we use the dip test in this paper; a fast implementation of the test is available in R.

Marginal selection can obviously fail. See Figure 29 taken from Wasserman, Azizyan and Singh (2014).

[2]Specifically, $c_{n,\alpha}$ can be defined by $\sup_{G \in \mathcal{U}} P_G(T_n > c_{n,\alpha}) = \alpha$. In practice, $c_{n,\alpha}$ can be defined by $P_U(T_n > c_{n,\alpha}) = \alpha$ where U is Unif(0,1). Hartigan and Hartigan (1985) suggest that this suffices asymptotically.

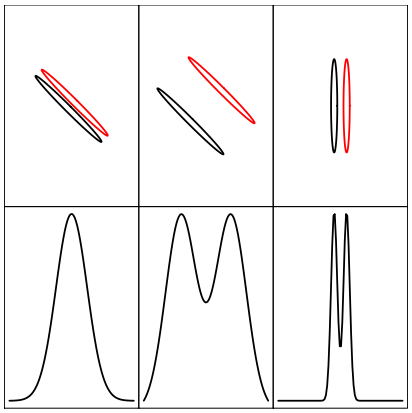

Figure 29: *Three examples, each showing two clusters and two features $X(1)$ and $X(2)$. The top plots show the clusters. The bottom plots show the marginal density of $X(1)$. Left: The marginal fails to reveal any clustering structure. This example violates the marginal signature assumption. Middle: The marginal is multimodal and hence correctly identifies $X(1)$ as a relevant feature. This example satisfies the marginal signature assumption. Right: In this case, $X(1)$ is relevant but $X(2)$ is not. Despite the fact that the clusters are close together, the marginal is multimodal and hence correctly identifies $X(1)$ as a relevant feature. This example satisfies the marginal signature assumption. (Figure from Wasserman, Azizyan and Singh, 2014).*

8.2 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

$$R_n(C) = \frac{1}{n}\sum_{i=1}^{n} ||X_i - \Pi_C[X_i]||^2 = \frac{1}{n}\sum_{i=1}^{n} \min_{1\leq j\leq k} ||X_i - c_j||^2. \tag{20}$$

This is equivalent to minimizing the within sums of squares

$$\sum_{j=1}^{k} \frac{1}{n_j} \sum_{s,t\in A_j} d^2(X_s, X_t) \tag{21}$$

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

$$B = \frac{1}{n}\sum_{s,t} d^2(X_s, X_t) - \sum_{j=1}^{k} \frac{1}{n_j} \sum_{s,t\in A_j} d^2(X_s, X_t). \tag{22}$$

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 (22) 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 (22) 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 30: The Witten-Tibshirani Sparse k-means Method

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

$$B = \frac{1}{n} \sum_{s,t} d_w^2(X_s, X_t) - \sum_{j=1}^k \frac{1}{n_j} \sum_{s,t \in A_j} d_w^2(X_s, X_t) \tag{23}$$

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. There is no theory that shows that this method works.

9 Finite Mixture Models

Simple cluster structure can be discovered using mixture models. We start with a simple example. We flip a coin with success probability η. If heads, we draw X from a density $p_1(x)$. If tails, we draw X from a density $p_0(x)$. Then the density of X is

$$p(x) = \eta p_1(x) + (1-\eta)p_0(x),$$

which is called a mixture of two densities p_1 and p_0. Figure 31 shows a mixture of two Gaussians distribution.

Let $Z \sim \text{Bernoulli}(\eta)$ be the unobserved coin flip. Then we can also write $p(x)$ as

$$p(x) = \sum_{z=0,1} p(x,z) = \sum_{z=0,1} p(x|z)p(z) \tag{24}$$

where $p(x|Z=0) := p_0(x)$, $p(x|Z=1) := p_1(x)$ and $p(z) = \eta^z(1-\eta)^{1-z}$. Equation (24) is called the hidden variable representation. A more formal definition of finite mixture models is as follows.

[Finite Mixture Models] Let $\{p_\theta(x) : \theta \in \Theta\}$ be a parametric class of densities. Define the mixture model

$$p_\psi(x) = \sum_{j=0}^{K-1} \eta_j p_{\theta_j}(x),$$

where the mixing coefficients $\eta_j \geq 0$, $\sum_{j=0}^{K-1} \eta_j = 1$ and $\psi = (\eta_0, \ldots, \eta_{K-1}, \theta_0, \ldots, \theta_{K-1})$ are the unknown parameters. We call $p_{\theta_0}, \ldots, p_{\theta_{K-1}}$ the component densities.

Generally, even if $\{p_\theta(x) : \theta \in \Theta\}$ is an exponential family model, the mixture may no longer be an exponential family.

9.1 Mixture of Gaussians

Let $\phi(x; \mu_j, \sigma_j^2)$ be the probability density function of a univariate Gaussian distribution with mean μ_j and variance σ_j^2. A typical finite mixture model is the mixture of Gaussians. In one dimension, we have

$$p_\psi(x) = \sum_{j=0}^{K-1} \eta_j \phi(x; \mu_j, \sigma_j^2),$$

which has $3K-1$ unknown parameters, due to the restriction $\sum_{j=0}^{K-1} \eta_j = 1$.

A mixture of d-dimensional multivariate Gaussians is

$$p(x) = \sum_{j=0}^{K-1} \frac{\eta_j}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left\{-\frac{1}{2}(x-u_j)^T\Sigma_j^{-1}(x-u_j)\right\}.$$

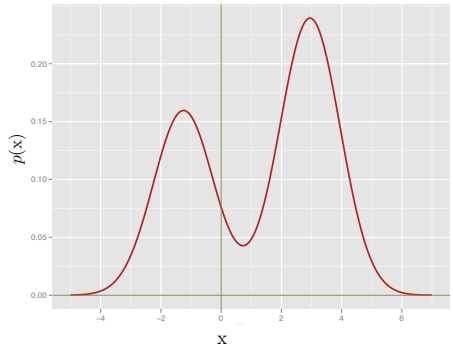

Figure 31: A mixture of two Gaussians, $p(x) = \frac{2}{5}\phi(x; -1.25, 1) + \frac{3}{5}\phi(x; 2.95, 1)$.

There are in total

$$K\Bigg(\underbrace{\frac{d(d+1)}{2}}_{\#\text{ of parameters in }\Sigma_j} + \underbrace{d}_{\#\text{ of parameters in }u_j}\Bigg) + \underbrace{(K-1)}_{\#\text{ of mixing coefficients}} = \frac{Kd(d+3)}{2} + K - 1$$

parameters in the mixture of K multivariate Gausssians.

9.2 Mixture of Multinomials

Another important mixture mode is a mixture of multinomials, which is useful for modeling count data. Let $X = (X_1, \ldots, X_m)^T \in \mathbb{Z}^m$ be the observed variable and $Z \in \{0, \ldots, K-1\}$ be the latent variable with $Z \sim \text{Multinomial}(1, \eta_0, \ldots, \eta_{K-1})$. We assume that

$$X|Z = j \sim \text{Multinomial}(n, \theta_{j1}, \ldots, \theta_{jm}).$$

Let $\theta_j = (\theta_{j1}, \ldots, \theta_{jm})^T$ and $x = (x_1, \ldots, x_m)$, we have

$$\begin{aligned} p(x) &= \sum_{j=0}^{K-1} \mathbb{P}(X = x|Z = j)\mathbb{P}(Z = j) \\ &= \frac{n!}{x_1! \cdots x_m!} \sum_{j=0}^{K-1} \eta_j \Big(\prod_{\ell=1}^{m} \theta_{j\ell}^{x_\ell}\Big). \end{aligned}$$

There are in total $K(m-1) + (K-1) = Km - 1$ parameters.

9.3 More General Mixture Models

We can write a finite mixture model more generally as

$$p_\psi(x) = \int p_\theta(x) dQ(\theta)$$

where Q is the distribution that puts mass η_j on θ_j. But, more genrally, Q can also be a continuous distribution.

Sometimes we can formally write a complicated distribution as a mixture of simpler distributions. For example, let $p(x)$ be the density function for a t random variable with ν degrees of freedom,

$$p(x) = \frac{\Gamma((\nu+1)/2)}{\sqrt{\pi\nu}\Gamma(\nu/2)(1+x^2)^{(\nu+1)/2}}.$$

Some calculations show that

$$p(x) = \int \phi(x; 0, 1/\sqrt{u}) dQ(u)$$

where Q is the distribution for a Gamma ($\nu/2$, $\nu/2$) random variable. Thus, a t distribution is a continuous mixture of Gaussians. However, this is a mixture of scale parameters instead of location parameters as in the previous section. We focus on finite mixture models.

9.4 Convexity for Mixture Models

Certain forms of mixture models result in convex estimation problems. It can be useful to recognize and exploit this convexity. First, suppose that we fix the density parameters θ_j in the parametric mixture and denote the component densities to be $p_0, p_1, \ldots, p_{K-1}$, the only unknown parameters in the finite mixture model

$$p_\eta(x) = \sum_{j=0}^{K-1} \eta_j \, p_j(x)$$

are the mixing coefficients $\eta_0, \ldots, \eta_{K-1}$. Let $X_1, \ldots, X_n$ be n random samples with $X_i = (X_{i1}, \ldots, X_{im})^T$, the log-likelihood function

$$\ell(\eta) = \sum_{i=1}^{n} \log\left(\sum_{j=0}^{K-1} \eta_j \, p_j(X_i)\right)$$

is a concave function of η. This can be seen since $\log\left(\sum_{j=0}^{K-1} \eta_j \, p_j(X_i)\right)$ is the logarithm of an affine function of η, and the logarithmic function is concave. Thus, optimizing the mixing coefficients, holding the component parameters fixed, is a convex optimization problem. For the same reason, if we hold the mixing parameters η fixed, and the model $p_{\theta_j}(x)$ is a multinomial, then the log-likelihood of $\theta_0, \ldots, \theta_{K-1}$

$$\ell(\theta_0, \ldots, \theta_{K-1}) = \sum_{i=1}^{n} \log\left(\sum_{j=0}^{K-1} \eta_j \prod_{\ell} \theta_{j\ell}^{x_{i\ell}}\right)$$

is concave in θ. However, it is not jointly concave in η and θ.

9.5 Maximum Likelihood Estimation

A finite mixture model $p_\psi(x)$ has parameters $\psi = (\eta_0, \ldots, \eta_{K-1}, \theta_0, \ldots, \theta_{K-1})$. The likelihood of ψ based on the observations $X_1, \ldots, X_n$ is

$$\mathcal{L}(\psi) = \prod_{i=1}^{n} p_\psi(X_i) = \prod_{i=1}^{n} \Big(\sum_{j=0}^{K-1} \eta_j p_{\theta_j}(X_i) \Big)$$

and, as usual, the maximum likelihood estimator is the value $\widehat{\psi}$ that maximizes $\mathcal{L}(\psi)$. Usually, the likelihood is multimodal and one seeks a local maximum instead if a global maximum.

For fixed $\theta_0, \ldots, \theta_{K-1}$, the log-likelihood is often a concave function of the mixing parameters η_j. However, for fixed $\eta_0, \ldots, \eta_{K-1}$, the log-likelihood is not generally concave with respect to $\theta_0, \ldots, \theta_{K-1}$.

One way to find $\widehat{\psi}$ is to apply your favorite optimizer directly to the log-likelihood.

$$\ell(\psi) = \sum_{i=1}^{n} \log \Big(\sum_{j=0}^{K-1} \eta_j p_{\theta_j}(X_i) \Big).$$

However, $\ell(\psi)$ is not jointly convex with respect to ψ. It is not clear which algorithm is the best to optimize such a nonconvex objective function.

A convenient and commonly used algorithm for finding the maximum likelihood estimates of a mixture model (or the more general latent variable models) is the *expectation-maximization (EM)* algorithm. The algorithm runs in an iterative fashion and alternates between the "E-step" which computes conditional expectations with respect to the current parameter estimate, and the "M-step" which adjusts the parameter to maximize a lower bound on the likelihood. While the algorithm can be slow to converge, its simplicity and the fact that it doesn't require a choice of step size make it a convenient choice for many estimation problems.

On the other hand, while simple and flexible, the EM algorithm is only one of many numerical procedures for obtaining a (local) maximum likelihood estimate of the latent variable models. In some cases procedures such as Newton's method or conjugate gradient may be more effective, and should be considered as alternatives to EM. In general the EM algorithm converges linearly, and may be extremely slow when the amount of missing information is large,

In principle, there are polynomial time algorithms for finding good estimates of ψ based on spectral methods (refxxxx) and the method of moments (refxxxx). It appears that, at least so far, these methods are not yet practical enough to be used in routine data analysis.

9.6 Expectation-Maximization (EM) Algorithm

The expectation-maximization (EM) algorithm is widely used for obtaining maximum likelihood estimates for models with latent variables or missing data. The advantages of EM are that it is conceptually simple and exploits the structure of latent variable models. For finite mixture models, the EM algorithm often leads to an easy-to-implement procedure that is guaranteed to increase the likelihood monotonically, without the need for line search methods. The monotonicity property can be extremely helpful for debugging the implementation.

In this section, we first introduce a deterministic optimization strategy, block coordinate ascent, as a motivation of the methodological development of the EM algorithm. We then use the mixture of Gaussians model as a special case to provide an informal derivation of the EM algorithm. A formal derivation with geometric interoperation is then provided.

9.7 Main Idea

The main intuition of the expectation-maximization (EM) algorithm comes from block coordinate descent optimization, which considers a maximization problem with an objective function

$$\max_{x,z} f(\underbrace{x_1, \ldots, x_m}_{x}, \underbrace{z_1, \ldots, z_n}_{z}). \tag{25}$$

Sometimes, maximizing the objection function $f(\cdot)$ jointly with respect to x and z is not easy. However, if z is fixed, optimizing f with respect to x is easy; and if x is fixed, optimizing f with respect to z is easy. Block coordinate descent is an iterative algorithm which alternately conducts two steps: (1) Optimizing the objective function with respect to x with z fixed; (2) optimizing the objective function with respect to z with x fixed. The block coordinate descent algorithm can be described as follows:

The Block Coordinate Descent Algorithm

Initialize $x^{(0)}$ and $z^{(0)}$.

For $t = 1, 2, \ldots$. {

- $x^{(t+1)} \leftarrow \text{argmax}_x \, f(x, z^{(t)})$
- $z^{(t+1)} \leftarrow \text{argmax}_z \, f(x^{(t+1)}, z)$

} until convergence.

It is easy to see that the objective function values of the block coordinate decent algorithm will be nondecreasing.

The EM algorithm is similar to (but not quite the same as) applying the block coordinate descent strategy to optimize the likelihood function of latent variable models. To do this, we need to examine the likelihood function of latent variable models and figure out what are the corresponding blocks of variables x and z as in Equation (25). We illustrate this intuition by considering a simple example of mixture of two univariate Gaussians.

The statistical model for a mixture of two Gaussians can be written as

$$Z \sim \text{Bernouli}(\eta), \tag{26}$$

$$X|Z = j \sim N(\mu_j, 1) \quad \text{for } j = 0, 1. \tag{27}$$

Define

$$p(x|Z = 1) := p_{\mu_1}(x) \quad \text{and} \quad p(x|Z = 0) := p_{\mu_0}(x).$$

Let $X_1, \ldots, X_n$ be the observed data and let $Z_1, \ldots, Z_n$ be the missing data. For a clustering problem, there are two types of unknowns: (i) the parameter vector $\psi = (\eta, \mu_0, \mu_1)^T$ and the latent samples $Z_1, \ldots, Z_n$. The latent variables $Z_1, \ldots, Z_n$ can be used for clustering while ψ can be used for evaluating the likelihood. The EM algorithm is similar to a block coordinate ascent procedure, which aims to maximize the log-likelihod function by alternatively inferring the information of $Z_1, \ldots, Z_n$ (Expectation-step) and estimating the parameter vector ψ (Maximization-step).

9.8 An Informal Derivation of the EM Algorithm

Using the mixture of Gaussians as an example, we illustrate the informal derivation of the EM algorithm. We start with the following question:

Question 1: If the missing values $Z_1, \ldots, Z_n$ are given, how do we find the parameter value ψ that maximizes the log-likelihood?

With $Z_1, \ldots, Z_n$ given, the *complete* log-likelihood of ψ under the mixture of two Gaussians

model is

$$\begin{aligned}
\ell^C(\psi) &= \sum_{i=1}^{n} \log p_\psi(X_i, Z_i) && (28)\\
&= \sum_{i=1}^{n} I(Z_i = 1) \log\big[p_{\mu_1}(X_i) \cdot \eta\big] + \sum_{i=1}^{n} I(Z_i = 0) \log\big[p_{\mu_0}(X_i) \cdot (1-\eta)\big] && (29)\\
&= -\sum_{i=1}^{n} I(Z_i = 1)\Big[\frac{X_i - \mu_1}{2}\Big]^2 + \log(\eta) \cdot \sum_{i=1}^{n} I(Z_i = 1)\\
&\quad -\sum_{i=1}^{n} I(Z_i = 0)\Big[\frac{X_i - \mu_0}{2}\Big]^2 + \log(1-\eta) \cdot \sum_{i=1}^{n} I(Z_i = 0)
\end{aligned}$$

We calculate the partial derivative of $\ell^C(\psi)$ with respect to η and set it to zero:

$$\frac{\partial \ell^C(\psi)}{\partial \eta} := \frac{\sum_{i=1}^n I(Z_i = 1)}{\eta} - \frac{\sum_{i=1}^n I(Z_i = 0)}{1-\eta} = 0,$$

which implies that

$$\widehat{\eta} = \frac{\sum_{i=1}^n I(Z_i = 1)}{n}.$$

We then calculate the partial derivate with respect to μ_1 and set it to be 0,

$$\frac{\partial \ell^C(\psi)}{\partial \mu_1} := \sum_{i=1}^{n} I(Z_i = 1) X_i - \mu_1 \sum_{i=1}^{n} I(Z_i = 1) = 0.$$

This implies that

$$\widehat{\mu}_1 = \frac{\sum_{i=1}^n I(Z_i = 1) X_i}{\sum_{i=1}^n I(Z_i = 1)}.$$

Similarly, we can get the updating rule for μ_0 as

$$\widehat{\mu}_0 = \frac{\sum_{i=1}^n I(Z_i = 0) X_i}{\sum_{i=1}^n I(Z_i = 0)}.$$

From the above discussion, we see that once the missing data are given, it is easy to update the parameters. The next question is, given the current parameter configuration, how shall we infer the missing data $Z_1, \ldots, Z_n$?

Question 2: Given the current estimate of the parameter ψ, how can we infer the missing values $Z_1, \ldots, Z_n$?

Question 2 is more subtle since $Z_1, \ldots, Z_n$ are random quantities. Since we are treating ψ as known in ths step, we can find the distribtion of Z_i given the observed data and then we

estimate Z_i with its mean with respect to its distribution. The distribution of Z_i given the observed data is

$$\mathbb{P}\big(Z_i = 1|X_1, \ldots, X_n\big) := \mathbb{E}\big[I(Z_i = 1)|X_1, \ldots, X_n\big].$$

For this, we use the Bayes formula (recalling that the parameter values are taken as known in this step)

$$\begin{aligned}\mathbb{P}\big(Z_i = 1|X_1, \ldots, X_n\big) &= \mathbb{P}(Z_i = 1|X_i) \\ &= \frac{p(X_i|Z_i = 1)\mathbb{P}(Z_i = 1)}{p(X_i)} \\ &= \frac{p_{\mu_1}(X_i)\eta}{p_{\mu_1}(X_i)\eta + p_{\mu_0}(X_i)(1-\eta)}.\end{aligned}$$

Similarly, we know that

$$\mathbb{P}\big(Z_i = 1|X_1, \ldots, X_n\big) = 1 - \mathbb{P}\big(Z_i = 0|X_1, \ldots, X_n\big) = \frac{p_{\mu_0}(X_i)(1-\eta)}{p_{\mu_1}(X_i)\eta + p_{\mu_0}(X_i)(1-\eta)}.$$

For the mixture of two Gaussians in (26) and (27), we have

$$\mathbb{P}\big(Z_i = 1|X_1, \ldots, X_n\big) = \frac{\eta \exp\big[-\frac{(X_i-\mu_1)^2}{2}\big]}{\eta \exp\big[-\frac{(X_i-\mu_1)^2}{2}\big] + (1-\eta)\exp\big[-\frac{(X_i-\mu_0)^2}{2}\big]}.$$

To summarize: given the current parameter configuration ψ, we infer the conditional distribution $\mathbb{P}(Z_i = 1|X_1, \ldots, X_n)$. But to answer Question 1, we need the exact realizations of $Z_1, \ldots, Z_n$ to update the parameter estimate; there is a gap here. To be able to alternate between the two steps of estimating the parameter ψ and inferring the missing values $Z_1, \ldots, Z_n$, we need to answer the next question:

Question 3: Instead of the exact realizations $Z_1, \ldots, Z_n$, we are only given the conditional distributions $\mathbb{P}(Z_i = 1|X_1, \ldots, X_n)$ for $i = 1, \ldots, n$, how shall we find the parameter value ψ that maximizes the log-likelihood?

Compared with the setup in Question 1, we see that we can no longer evaluate the complete log-likelihood $\ell^C(\psi)$ since we do not have the exact realizations of $Z_1, \ldots, Z_n$ (or more specifically, we are not able to evaluate the class assignment $I(Z_i = 1)$). However, we have the conditional distribution $\mathbb{P}(Z_i = 1|X_1, \ldots, X_n)$, which corresponds to the conditional expectation of $I(Z_i = 1)$ with respect to the conditional distribution of Z_i given $X_1, \ldots, X_n$, i.e.,

$$\mathbb{P}(Z_i = 1|X_1, \ldots, X_n) = \mathbb{E}\big[I(Z_i = 1)|X_1, \ldots, X_n\big]$$

One simple idea is to replace the indicator $I(Z_i = 1)$ in (29) by the available conditional expectation $\mathbb{E}\big[I(Z_i = 1)|X_1, \ldots, X_n\big]$. This is exactly what the EM algorithm does. Later,

we will explain that this corresponds to finding ψ that maximizes the expected complete log-likelihood with respect to the conditional distribution of $\mathbb{P}(Z_i = 1|X_1, \ldots, X_n)$.

More specifically, the *expected* complete log-likelihood of ψ under the mixture of two Gaussians model is

$$\begin{aligned}
\ell^{EC}(\psi) &= \sum_{i=1}^{n} \mathbb{E}_{Z_i|X_i} \log p_\psi(X_i, Z_i) \qquad (30)\\
&= -\sum_{i=1}^{n} \mathbb{P}(Z_i = 1|X_i)\Big[\frac{X_i - \mu_1}{2}\Big]^2 + \log(\eta) \cdot \sum_{i=1}^{n} I(Z_i = 1)\\
&\quad -\sum_{i=1}^{n} \mathbb{P}(Z_i = 0|X_i)\Big[\frac{X_i - \mu_0}{2}\Big]^2 + \log(1-\eta) \cdot \sum_{i=1}^{n} I(Z_i = 0).
\end{aligned}$$

We calculate the partial derivative of $\ell^{EC}(\psi)$ with respect to η and set it to zero:

$$\frac{\partial \ell^{EC}(\psi)}{\partial \eta} := \frac{\sum_{i=1}^{n} \mathbb{P}(Z_i = 1|X_i)}{\eta} - \frac{\sum_{i=1}^{n} \mathbb{P}(Z_i = 0|X_i)}{1-\eta} = 0,$$

which implies that

$$\widehat{\eta} = \frac{\sum_{i=1}^{n} \mathbb{P}(Z_i = 1|X_i)}{n}.$$

We then calculate the partial derivate of $\ell^{EC}(\psi)$ with respect to μ_1 and μ_0, set them to be 0, and get

$$\widehat{\mu}_1 = \frac{\sum_{i=1}^{n} \mathbb{P}(Z_i = 1|X_i) X_i}{\sum_{i=1}^{n} \mathbb{P}(Z_i = 1|X_i)} \quad \text{and} \quad \widehat{\mu}_0 = \frac{\sum_{i=1}^{n} \mathbb{P}(Z_i = 0|X_i) X_i}{\sum_{i=1}^{n} \mathbb{P}(Z_i = 0|X_i)}.$$

Combining the results from Question 2 and Question 3, we get the following procedure for iteratively optimizing the log-likelihood of mixture of two Gaussians in (26) and (27).

The Expectation-Maximization Algorithm for the Mixture of Two Gaussians

Initialize $\psi^{(0)} := (\eta^{(0)}, \mu_1^{(0)}, \mu_0^{(0)})^T$.

For $t = 1, 2, \ldots$. {

- **Expectation-Step** (E-Step): for $i = 1, \ldots, n$, calculate

$$\begin{aligned}
\gamma_i^{(t+1)} &:= \mathbb{P}_{\psi^{(t)}}\big(Z_i = 1|X_1, \ldots, X_n\big)\\
&= \frac{\eta^{(t)} \exp\big[-\frac{(X_i - \mu_1^{(t)})^2}{2}\big]}{\eta^{(t)} \exp\big[-\frac{(X_i - \mu_1^{(t)})^2}{2}\big] + (1-\eta^{(t)}) \exp\big[-\frac{(X_i - \mu_0^{(t)})^2}{2}\big]}.
\end{aligned}$$

- **Maximization-Step** (M-Step): Given $\gamma_i^{(t+1)}$, we update the parameter ψ by

$$\eta^{(t+1)} \leftarrow \frac{1}{n}\sum_{i=1}^{n}\gamma_i^{(t+1)};$$

$$\mu_1^{(t+1)} \leftarrow \frac{\sum_{i=1}^n \gamma_i^{(t+1)} X_i}{\sum_{i=1}^n \gamma_i^{(t+1)}} \quad \text{and} \quad \mu_0^{(t+1)} \leftarrow \frac{\sum_{i=1}^n (1-\gamma_i^{(t+1)}) X_i}{\sum_{i=1}^n (1-\gamma_i^{(t+1)})}.$$

} until convergence.

9.9 The EM Algorithm for Mixtures of Multivariate Gaussians

Consider the mixture of multivariate Gaussians model

$$Z \sim \text{Multinomial}(1, \eta_0, \ldots, \eta_{K-1}), \tag{31}$$

$$X|Z = j \sim N(u_j, \Sigma_j) \text{ for } j = 0, 1, \ldots, K-1. \tag{32}$$

The EM steps becomes

The Expectation-Maximization Algorithm for the Mixture of Multivariate Gaussians

Initialize $\psi^{(0)} := \big(\eta_0^{(0)}, \ldots, \eta_{K-1}^{(0)}, u_0^{(0)}, \ldots, u_{K-1}^{(0)}, \Sigma_0^{(0)}, \ldots, \Sigma_{K-1}^{(0)}\big)^T$.

For $t = 1, 2, \ldots$. {

- **Expectation-Step** (E-Step): for $i = 1, \ldots, n$ and $j = 0, \ldots, K-1$, calculate

$$\gamma_{ij}^{(t+1)} \;:=\; \mathbb{P}_{\psi^{(t)}}\big(Z_i = j | X_1, \ldots, X_n\big) = \frac{\eta_j^{(t)} p_{u_j^{(t)}, \Sigma_j^{(t)}}(X_i)}{\sum_{\ell=0}^{K-1} \eta_\ell^{(t)} p_{u_\ell^{(t)}, \Sigma_\ell^{(t)}}(X_i)}.$$

Here $p_{u_\ell^{(t)}, \Sigma_\ell^{(t)}}(X_i)$ is a Gaussian density with mean $u_j^{(t)}$ and covariance $\Sigma_j^{(t)}$.

- **Maximization-Step** (M-Step): Given $\gamma_{ij}^{(t+1)}$, we update the parameter ψ by

$$\text{For } j = 0, \ldots, K-1:$$

$$\eta_j^{(t+1)} \leftarrow \frac{1}{n}\sum_{i=1}^{n}\gamma_{ij}^{(t+1)};$$

$$u_j^{(t+1)} \leftarrow \frac{\sum_{i=1}^n \gamma_{ij}^{(t+1)} X_i}{\sum_{i=1}^n \gamma_{ij}^{(t+1)}};$$

$$\Sigma_j^{(t+1)} \leftarrow \frac{\sum_{i=1}^n \gamma_{ij}^{(t+1)} (X_i - u_j^{(t+1)})(X_i - u_j^{(t+1)})^T}{\sum_{i=1}^n \gamma_{ij}^{(t+1)}}.$$

} until convergence.

From the above description we see that in the M-step of the EM algorithm we calculate the weighted sample mean and covariance, where the weight γ_{ij} is the posterior probability that example i was generated from mixture component j. These weights are sometimes called "responsibilities," suggesting that it is the probability that the latent variable j was responsible for example i.

9.10 A Geometric Perspective on the EM Algorithm

The EM algorithm is an iterative procedure that has two main steps. As in the mixture of Gaussians example, in the E-step, it tries to "guess" the values of the missing data $Z_1, \ldots, Z_n$. In the M-step, it updates the parameters of our model based on our guesses.

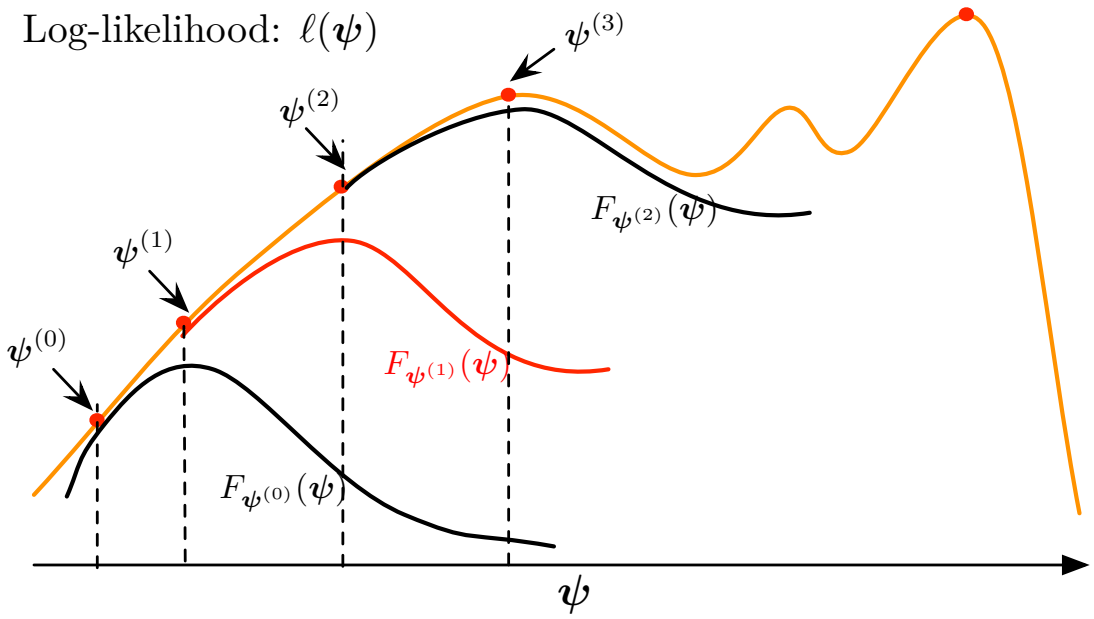

Figure 32: A geometric illustration of the EM Algorithm. Our goal is to find a local maxima of the log-likelihood function $\ell(\psi)$. In each iteration of the EM algorithm, we find a lower bound function and try to maximize the lower bound function.

Figure 32 illustrates the geometric intuition of the EM algorithm. Using the mixture of Gaussians as an example, we see that our goal is to find a local maxima of a nonconcave likelihood function

$$\ell(\psi) := \sum_{i=1}^{n} \log p_\psi(X_i) = \sum_{i=1}^{n} \log\Big(\sum_{j=0}^{K-1} \eta_j p_{u_j,\Sigma_j}(X_i|Z_i = j)\Big).$$

We first find an initial value $\psi^{(0)}$. Geometrically, the EM algorithm tries to find a lower bound function $F_{\psi^{(0)}}$, which satisfies

$$F_{\psi^{(0)}}(\psi) \leq \ell(\psi) \text{ for all } \psi,$$

and

$$F_{\psi^{(0)}}(\psi^{(0)}) = \ell(\psi^{(0)}).$$

The EM algorithm requires that the lower bound function $F_{\psi^{(0)}}(\psi)$ is easy to be optimized. To proceed, we maximize this lower bound function and get $\psi^{(1)}$:

$$\psi^{(1)} = \operatorname*{argmax}_{\psi} F_{\psi^{(0)}}(\psi).$$

We then continue with the same strategy to find a new lower bound function $F_{\psi^1}(\psi)$ which satisfies $F_{\psi^{(1)}}(\psi) \leq \ell(\psi)$ for all ψ and $F_{\psi^{(1)}}(\psi^{(1)}) = \ell(\psi^{(1)})$. The algorithm proceeds by maximizing the new lower bound function to get $\psi^{(2)}$:

$$\psi^{(2)} = \operatorname*{argmax}_{\psi} F_{\psi^{(1)}}(\psi).$$

We iterate the above process until the algorithm converges. In the next section, we will formally prove that such a procedure is guaranteed to converge to a local maxima of $\ell(\psi)$. By examining the above procedure, it is obvious that the initial value $\psi^{(0)}$ affects the final solution. In applications, a common practice is to run the EM algorithm for different initial values $\psi^{(0)}$ and use the one that gets the largest value of $\ell(\psi)$.

9.11 A Formal Derivation of the EM Algorithm

Let $X_1, \ldots, X_n$ be a random sample. We want to fit a mixture model with the density

$$p_\psi(x) = \sum_{j=0}^{K-1} p_{\theta_j}(x)\eta_j,$$

where $\eta_j = \mathbb{P}(Z = j)$ satisfying $\sum_{j=0}^{K-1} \eta_j = 1$, and $p_{\theta_j}(x) = p(x|Z = j)$. Our goal is to estimate the parameters $\psi = \left\{\theta_0, \eta_0, \ldots, \theta_{K-1}, \eta_{K-1}\right\}$ by maximum likelihood. We begin with some terminology:

- X: observed variable,
- Z: latent variable/unobserved varaible,
- $\ell^{\mathrm{M}}(\psi) = \log p_\psi(X_1, \ldots, X_n)$: marginal likelihood,
- $\ell^{\mathrm{C}}(\psi) = \log p_\psi(X_1, \ldots, X_n, Z_1, \ldots, Z_n)$: complete likelihood.

We also define

$$\gamma_{ij} = \mathbb{P}_\psi\big(Z_i = j | X_i\big).$$

It is easy to see that $\sum_{j=0}^{K-1} \gamma_{ij} = 1$. Therefore, $\gamma_{i1}, \ldots, \gamma_{i(K-1)}$ specify a probability distribution which indicates the posterior class assignment of a point X_i under the parameter configuration ψ.

The formal derivation of the EM algorithm hinges on the concavity of the logarithm, and unfolds by bounding the log-likelihood from below using Jensen's inequality. We derive the EM algorithm according to the following steps:

Task 1: Given the current parameter configuration $\psi^{(\text{old})}$, we want to derive a lower bound function $F(\psi)$ of $\ell^{\text{M}}(\psi)$ such that

$$F(\psi) \leq \ell^{\text{M}}(\psi) \text{ for all } \psi,$$

and $F(\psi^{(\text{old})}) = \ell^{\text{M}}(\psi^{(\text{old})})$. We define such a function to be $F_{\psi^{(\text{old})}}(\psi)$.

The key of this derivation is Jensen's inequality for the logarithmic function, which states that

$$\log \mathbb{E}X \geq \mathbb{E} \log X.$$

Let $\gamma = \{\gamma_{ij}\}_{i=1,\ldots,n;j=0,\ldots,K-1}$. We have

$$\begin{aligned}
\ell^{\text{M}}(\psi) &= \sum_{i=1}^{n} \log p_\psi(X_i) \\
&= \sum_{i=1}^{n} \log\Big[\sum_{j=0}^{K-1} p_\psi(X_i, Z_i = j)\Big] \\
&= \sum_{i=1}^{n} \log\Big[\sum_{j=0}^{K-1} \gamma_{ij} \frac{p_\psi(X_i, Z_i = j)}{\gamma_{ij}}\Big] \quad \text{(Mean field trick)} \\
&\geq \sum_{i=1}^{n} \sum_{j=0}^{K-1} \gamma_{ij} \log\Big[\frac{p_\psi(X_i, Z_i = j)}{\gamma_{ij}}\Big] \quad \text{(Jensen's inequality)} \\
&= \sum_{i=1}^{n} \sum_{j=0}^{K-1} \gamma_{ij} \log p_\psi(X_i, Z_i = j) - \sum_{i=1}^{n} \sum_{j=0}^{K-1} \gamma_{ij} \log \gamma_{ij} \\
&:= Q(\psi, \gamma).
\end{aligned}$$

Therefore, $Q(\psi, \gamma)$ is a function that lower bounds $\ell^{\text{M}}(\psi)$ for any ψ and γ, where ψ is the unknown parameter and γ carries the information of the missing data. The EM algorithm applies the block coordinate ascent strategy to maximize $Q(\psi, \gamma)$. More specifically, the EM algorithm iterates the following two steps until converge:

- **E-step**: $\gamma^{(\text{new})} \leftarrow \text{argmax}_\gamma \, Q(\psi^{(\text{old})}, \gamma)$,
- **M-step**: $\psi^{(\text{old})} \leftarrow \text{argmax}_\psi \, Q(\psi, \gamma^{(\text{new})})$.

To map the above updating rule back to that in Figure 32, we define

$$F_{\psi^{(\text{old})}}(\psi) := Q(\psi, \gamma^{(\text{new})}). \tag{33}$$

It is easy to see that $F_{\psi^{(\text{old})}}(\psi) \leq \ell^{\text{M}}(\psi)$ for any ψ and $F_{\psi^{(\text{old})}}(\psi^{(\text{old})}) = \ell^{\text{M}}(\psi^{(\text{old})})$. In the finite mixture model setting, the next theorem shows that the E-step update has a closed form solution.

Theorem 25 *For finite mixture models, the E-step update has a closed form solution*

$$\gamma_{ij}^{(\text{new})} = \mathbb{P}_{\psi^{(\text{old})}}\big(Z_i = j | X_i\big),$$

where $\psi^{(\text{old})}$ is the current parameter configuration.

Proof. In the above derivation, we know that $Q(\psi, \gamma) \leq \ell^{\text{M}}(\psi)$ for any ψ and γ. To show that $\gamma_{ij}^{(\text{new})} = \mathbb{P}_{\psi^{(\text{old})}}\big(Z_i = j | X_i\big)$ is the maximizer of $Q(\psi^{(\text{old})}, \gamma)$, it suffices to prove that

$$Q(\psi^{(\text{old})}, \gamma^{(\text{new})}) = \ell^{\text{M}}(\psi^{(\text{old})}).$$

To verify this, we have

$$\begin{aligned}
Q(\psi^{(\text{old})}, \gamma^{(\text{new})}) &= \sum_{i=1}^{n} \sum_{j=0}^{K-1} \gamma_{ij}^{(\text{new})} \log\Big[\frac{p_{\psi^{(\text{old})}}(X_i, Z_i = j)}{\gamma_{ij}^{(\text{new})}}\Big] \\
&= \sum_{i=1}^{n} \sum_{j=0}^{K-1} \mathbb{P}_{\psi^{(\text{old})}}\big(Z_i = j | X_i\big) \cdot \log\Big[\frac{\mathbb{P}_{\psi^{(\text{old})}}\big(Z_i = j | X_i\big) p_{\psi^{(\text{old})}}(X_i))}{\mathbb{P}_{\psi^{(\text{old})}}\big(Z_i = j | X_i\big)}\Big] \\
&= \sum_{i=1}^{n} \log p_{\psi^{(\text{old})}}(X_i) \underbrace{\Big[\sum_{j=0}^{K-1} \mathbb{P}_{\psi^{(\text{old})}}\big(Z_i = j | X_i\big)\Big]}_{=1} \\
&= \sum_{i=1}^{n} \log p_{\psi^{(\text{old})}}(X_i) \\
&= \ell^{\text{M}}(\psi^{(\text{old})}).
\end{aligned}$$

This finishes the proof. □ Once the lower bound function $F(\psi)$ is derived, our next task is to optimize it with the parameter ψ. For this, we need to handle the next task:

Task 2: Given the current lower bound function $F_{\psi^{(\text{old})}}(\psi)$ of $\ell^{\text{M}}(\psi)$, we update the parameter ψ by solving

$$\psi^{(\text{new})} = \operatorname*{argmax}_{\psi} F_{\psi^{(\text{old})}}(\psi).$$

The next theorem shows that $\psi^{(\text{new})}$ can be updated by maximizing the expected complete log-likelihood with respect to the conditional distribution of $\mathbb{P}_{\psi^{(\text{old})}}(Z_i = 1|X_1, \ldots, X_n)$.

Theorem 26 *Under a finite mixture model $p_\psi(x) = \sum_{j=0}^{K-1} p_{\theta_j}(x)\eta_j$, we have the following updating rule in the M-step:*

$$\theta_j^{(\text{new})} = \operatorname*{argmax}_{\psi_j} \sum_{i=1}^{n} \sum_{j=0}^{K-1} \gamma_{ij}^{(\text{old})} \log p_{\theta_j^{(\text{old})}}(X_i, Z_i = j), \tag{34}$$

$$\eta_j^{(\text{new})} = \frac{1}{n} \sum_{i=1}^{n} \gamma_{ij}^{(\text{new})}, \tag{35}$$

where $\gamma_{ij}^{(\text{new})} = \mathbb{P}_{\psi^{(\text{old})}}\big(Z_i = j|X_i\big)$.

Proof. From (33), we know that

$$\begin{aligned} F_{\psi^{(\text{old})}}(\psi) &= Q(\psi, \gamma^{(\text{new})}) \\ &= \sum_{i=1}^{n} \sum_{j=0}^{K-1} \gamma_{ij}^{(\text{new})} \log\Big[\frac{p_{\theta_j}(X_i, Z_i = j)}{\gamma_{ij}^{(\text{new})}}\Big] \\ &= \sum_{i=1}^{n} \sum_{j=0}^{K-1} \gamma_{ij}^{(\text{new})} \log\Big[\frac{p_{\theta_j}(X_i|Z_i = j)\eta_j}{\gamma_{ij}^{(\text{new})}}\Big] \end{aligned}$$

Optimizing $F_{\psi^{(\text{old})}}(\psi)$ with respect to θ_j, we get (34). To optimize $F_{\psi^{(\text{old})}}(\psi)$ with respect to η_j, we need to solve a constrained optimization

$$\max_{\eta_0, \ldots, \eta_{K-1}} \sum_{i=1}^{n} \sum_{j=0}^{K-1} \gamma_{ij}^{(\text{new})} \log \eta_j,$$

$$\text{subject to } \eta_0, \ldots, \eta_{K-1} \geq 0 \text{ and } \sum_{j=0}^{K-1} \eta_j = 1.$$

Solving the above optimization, we get the desired result. □

9.12 Convergence of the EM Algorithm

The next theorem shows that the convergence of the EM algorithm to a local maximizer is guaranteed.

Theorem 27 (Convergence of the EM Algorithm) *When the marginal likelihood $\ell^{\mathrm{M}}(\psi)$ is bounded from above, the EM algorithm is guaranteed to converge.*

Proof. From the derivation in the last section, we see that the EM algorithm is a block coordinate descent algorithm for maximizing the auxiliary function $Q(\psi, \gamma)$. The convergence of the EM algorithm follows from the convergence of the block coordinate descent algorithm under the bounded likelihood condition. □

Remark 28 *Since the marginal log-likelihood functions of the finite mixture models are in general nononvex, the EM algorithm may suffer from local maxima. In applications, we generally try different random initializations and pick the one that achieves the largest likelihood function.*

9.13 Old Faithful Geyser Data Example

We study a version of the eruptions data "Old Faithful" geyser in Yellowstone national park. The geyser was named by the Washburn expedition of 1870. They were impressed by its size and frequency. It is not the biggest or most regular geyser in Yellowstone but it is the biggest regular geyser. Furthermore, it has been erupting in nearly the same fashion throughout the recorded history of Yellowstone. Through the years, it has become one of the most studied geysers in the park.

The data in this study are continuous measurements from August 1 to August 15, 1985. There are two variables with 299 observations. The first variable ,"Duration", represents the numeric eruption time in minutes. The second variable, "waiting", represents the waiting time to next eruption. This data is believed to have two modes. We fit a mixture of two Gaussians using EM algorithm. To illustrate the EM step, we purposely choose a bad starting point. The EM algorithm quickly converges in six steps. Figure 33 illustrates the fitted densities for all the six steps. We see that even though the starting density is unimodal, it quickly becomes bimodal.

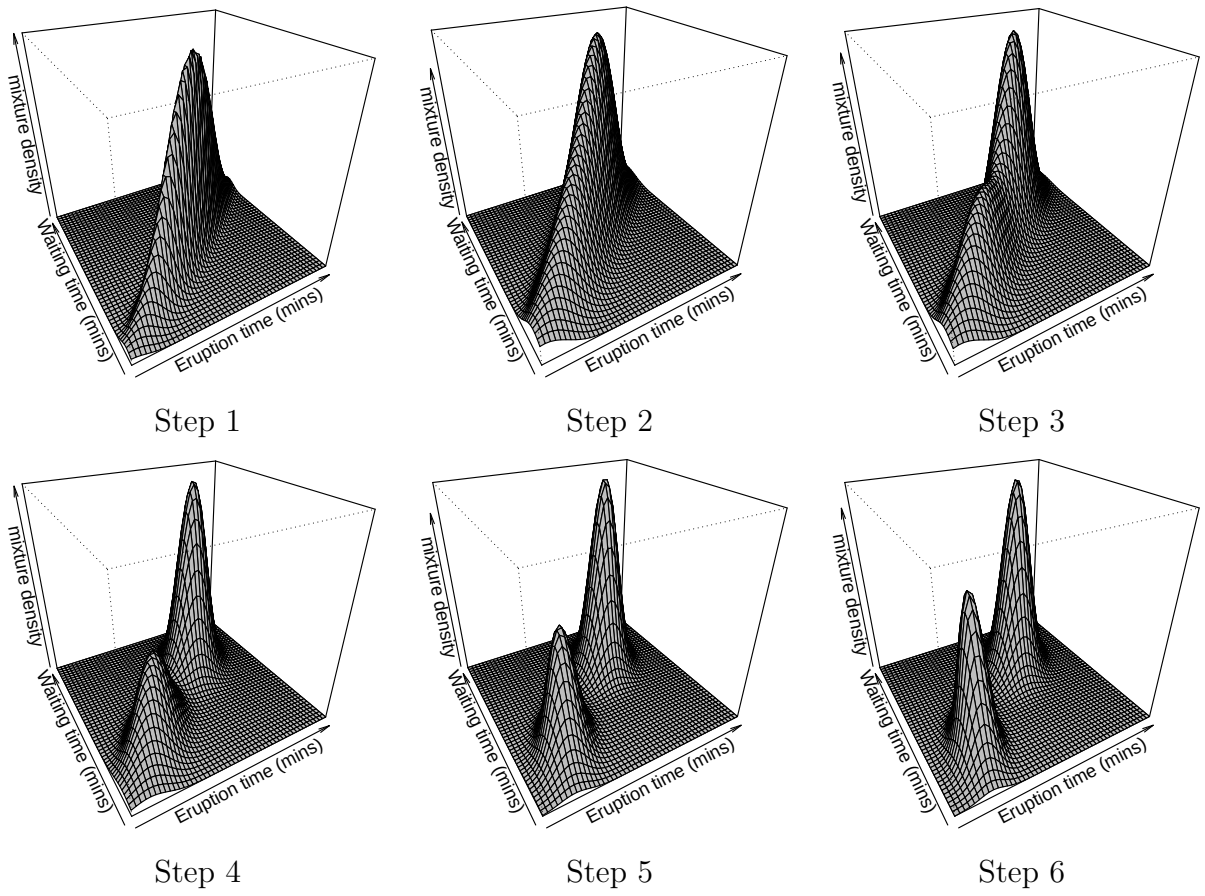

Figure 33: Fitting a mixture of two Gaussians on the Old Faithful Geyser data. The initial values are $\eta_0 = \eta_1 = 0.5$. $u_0 = (4, 70)^T$, $u_1 = (3, 60)^T$, $\Sigma_1 = \Sigma_2 = \begin{pmatrix} 0.8 & 7 \\ 7 & 70 \end{pmatrix}$. We see that even though the starting density is not bimodal, the EM algorithm converges quickly to a bimodal density.

10 Examples

Example 29 *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?*

Example 30 (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*

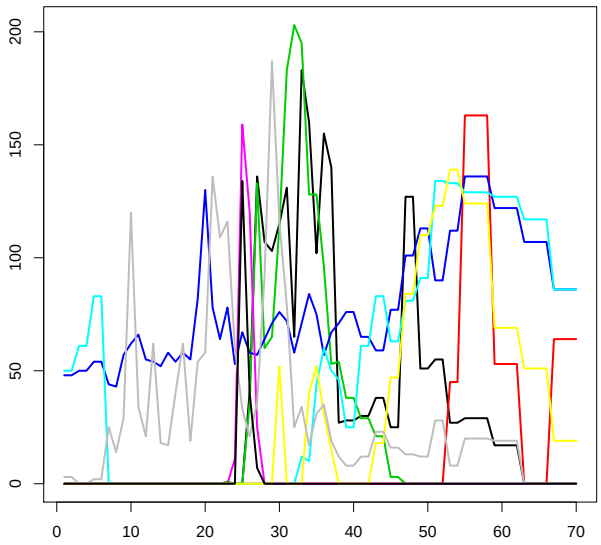

Figure 34: Some curves from a dataset of 472 curves. Each curve is a radar waveform from the Topex/Poseidon satellite.

$d \approx 5,000$ *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 31 (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 34 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.*[3] *One question is whether the 472 curves can be put into groups of similar shape.*

Example 32 (Supernova Clustering) *Figure 35 shows another example of curve clustering. 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*

[3]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`.

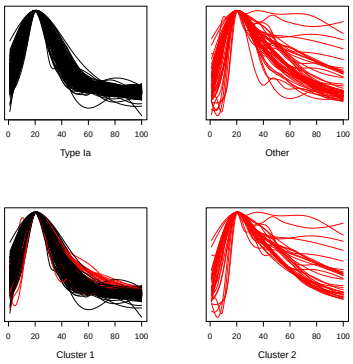

Figure 35: Light curves for supernovae. The top two plots show the light curves for two types of supernovae. The bottom two plots show the results of clustering the curves into two groups, without using knowledge of their labels.

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.

11 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.

12 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 [**?**].