Clustering
10716, Spring 2020
Pradeep Ravikumar (amending notes from Larry Wasserman)

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 an example of *unsupervised learning*. We will study the following approaches:

1. k-means
2. Mixture models
3. Density-based Clustering I: Level Sets and Trees
4. Density-based Clustering II: Modes
5. Hierarchical Clustering
6. Spectral Clustering

Some issues that we will address are:

1. Rates of convergence
2. Choosing tuning parameters
3. Variable selection
4. High Dimensional Clustering

Example 1 *Figures 19 and 20 show some synthetic examples where the clusters are meant to be intuitively clear. In Figure 19 there are two blob-like clusters. Identifying clusters like this is easy. Figure 20 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 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. For now, we assume that k is given. Later we discuss how to choose k.

Warning! My view is that k is a tuning parameter; it is **not** the number of clusters. Usually we want to choose k to be larger than the number of clusters.

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

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

Define the empirical clustering risk of a codebook C by

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

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

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

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

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

where $X \sim P$. The optimal population quantization $C^* = \{c_1^*, \ldots, c_k^*\} \in \mathcal{C}_k$ 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 *Voronoi tesselation*. Let

$$V_j = \Big\{ x : \ ||x - c_j|| \le ||x - c_s||, \ \text{ for all } s \ne j \Big\}. \tag{5}$$

The set V_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 1.

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 2. 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 2.1.

Example 2 *Figure 3 shows synthetic data inspired by the Mickey Mouse example from* `http://en.wikipedia.org/wiki/K-means_clustering`. *The data in the top left plot form three clearly defined clusters. k-means easily finds in the clusters (top right). The bottom shows the same example except that we now make the groups very unbalanced. The lack of balance causes k-means to produce a poor clustering. But note that, if we "overfit then merge" then there is no problem.*

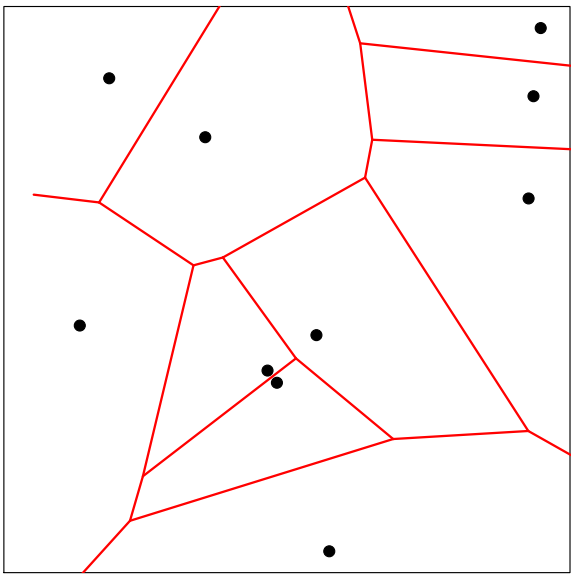


Figure 1: *The Voronoi tesselation formed by 10 cluster centers* $c_1, \ldots, c_{10}$*. The cluster centers are indicated by dots. The corresponding Voronoi cells* $T_1, \ldots, T_{10}$ *are defined as follows: a point* x *is in* T_j *if* x *is closer to* c_j *than* c_i *for* $i \neq j$*.*

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

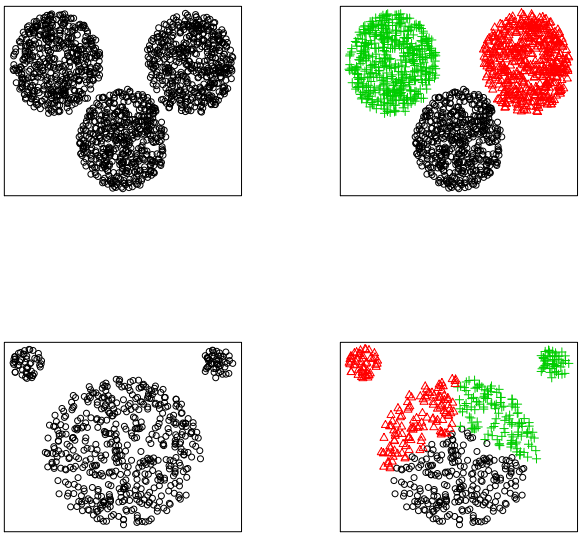


Figure 3: *Synthetic data inspired by the "Mickey Mouse" example from wikipedia. Top left: three balanced clusters. Top right: result from running k means with $k = 3$. Bottom left: three unbalanced clusters. Bottom right: result from running k means with $k = 3$ on the unbalanced clusters. k-means does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

Example 3 *The top left plot of Figure 4 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 unless we overfit then merge.*

2.1 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 4 *Figure 5 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

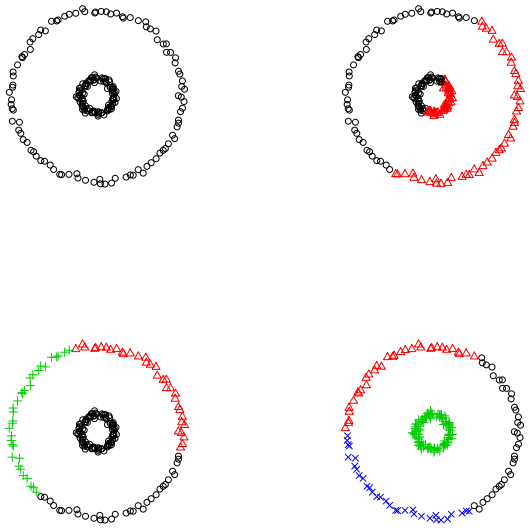


Figure 4: Top left: a dataset with two ring-shaped clusters. Top right: k-means with $k = 2$. Bottom left: k-means with $k = 3$. Bottom right: k-means with $k = 4$.

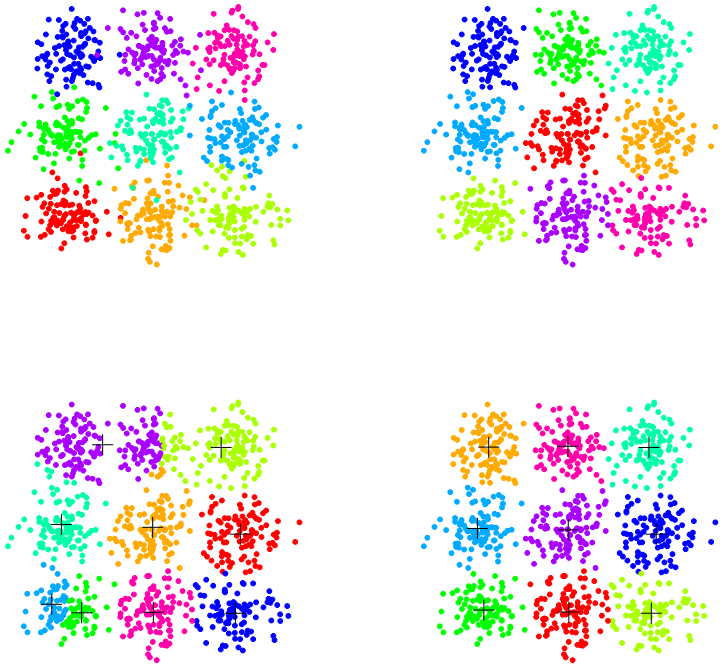


Figure 5: An example with 9 clusters. Top left: data. Top right: k-means with random starting values. Bottom left: k-means using starting values from hierarchical clustering. Bottom right: the k-means^{++} algorithm.

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

describe in Section 7) 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 5.

k-means^{++}. Arthur and Vassilvitskii (2007) invented an algorithm called k-means^{++} to get good starting values. They show that if the starting points are chosen in a certain way, then we can get close to the minimum with high probability. In fact the starting points themselves — which we call seed points — are already close to minimizing $R_n(C)$. The algorithm is described in Figure 6. See the bottom right plot in Figure 5 for an example.

Theorem 5 *(Arthur and Vassilvitskii, 2007). Let $C = \{c_1, \ldots, c_k\}$ be the seed points from the k-means^{++} algorithm. Then,*

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

where the expectation is over the randomness of the algorithm.

See Arthur and Vassilvitskii (2007) for a proof. They also show that the Euclidean distance can be replaced with the ℓ_p norm in the algorithm. The result is the same except that the constant 8 gets replaced by 2^{p+2}. It is possible to improve the k-means^{++} algorithm.

2.2 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.

2.2.1 Elbow Methods

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

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

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

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

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

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

2.2.2 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.

A nice 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 6 *Figure 7 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 $k = 3$.*

2.2.3 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*

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

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

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

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

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

The first problem is easily solved. We can insist the labels take values in $\{1, \ldots, k\}$ and then we can maximize the similarity over all permutations of the labels. Another way to solve

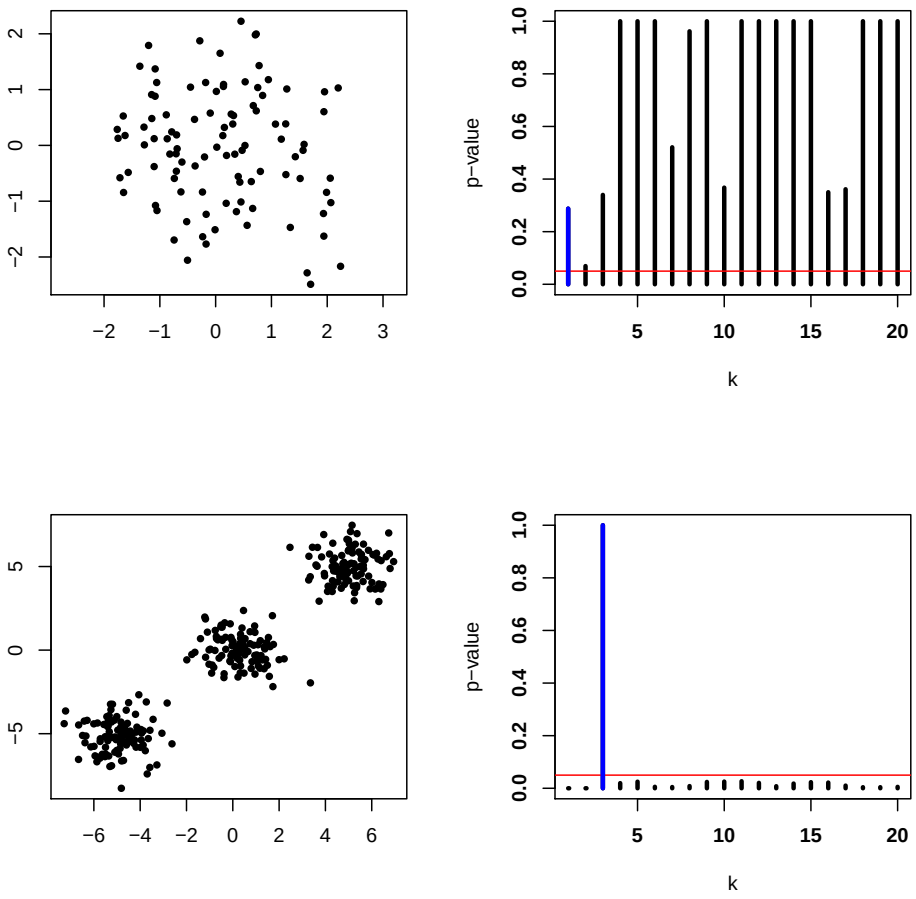


Figure 7: Top left: a single cluster. Top right: p-values for various k. The first k for which the p-value is larger than .05 is $k = 1$. Bottom left: three clusters. Bottom right: p-values for various k. The first k for which the p-value is larger than .05 is $k = 3$.

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

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

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

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

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

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

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

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

It is important to interpret stability correctly. These methods choose the largest number of stable clusters. That does not mean they choose "the true k." Indeed, Ben-David, von Luxburg and Pál (2006), Ben-David and von Luxburg Tübingen (2008) and Rakhlin (2007) have shown that trying to use stability to choose "the true k" — even if that is well-defined

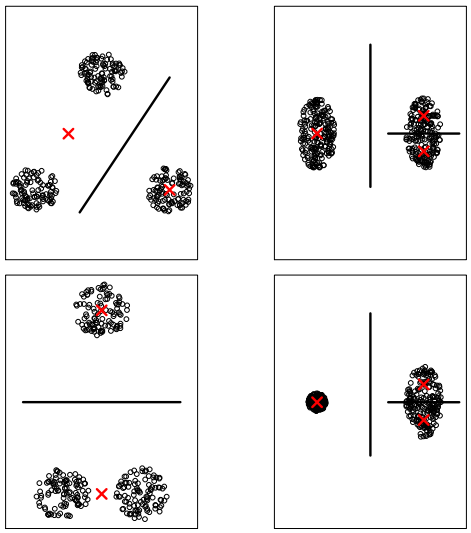


Figure 8: Examples from Ben-David, von Luxburg and Pál (2006). The first example (top left plot) shows a case where we fit $k = 2$ clusters. Stability analysis will correctly show that k is too small. The top right plot has $k = 3$. Stability analysis will correctly show that k is too large. The bottom 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.

— will not work. To explain this point further, we consider some examples from Ben-David, von Luxburg and Pál (2006). Figure 8 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.

2.2.4 Silhouette Score

A practically useful approach is based on the silhouette scores, and silhoutte graphs (Rousseeuw 1987).

For each point X_i in any cluster C_j, denote its intra-cluster fit as:

$$\text{ic}_i = \frac{1}{|C_j| - 1} \sum_{s \in C_j, s \neq i} \|X_i - X_s\|,$$

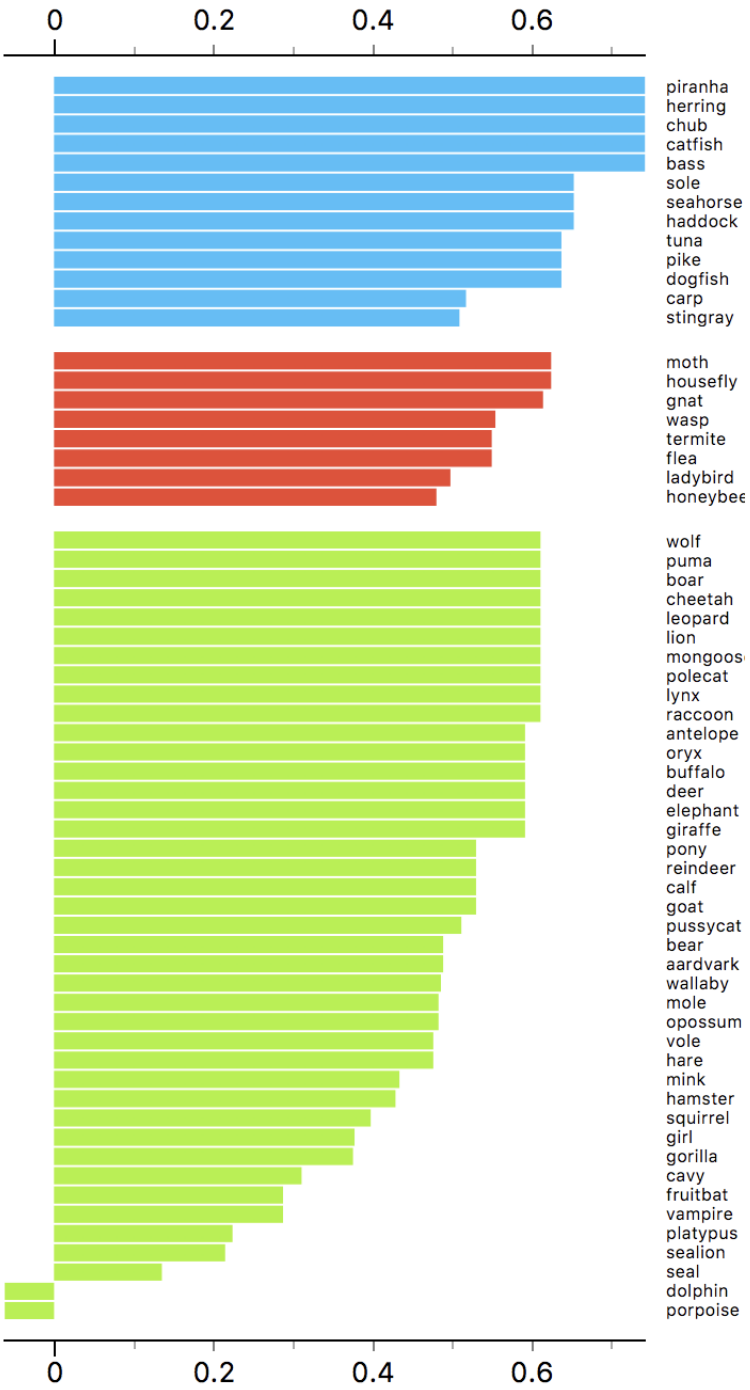


Figure 9: (From Wikipedia) A plot of the silhouette scores of all the data points in each of three clusters. Note that some scores are negative, indicating lack of fit with that cluster.

and denote its extra-cluster fit with any of the other clusters as:

$$\mathrm{xc}_i = \min_{j' \neq j} \frac{1}{|C_{j'}| - 1} \sum_{s \in C_{j'}} \|X_i - X_s\|.$$

We can then define the silhouette value for X_i as:

$$s_i = \frac{\mathrm{xc}_i - \mathrm{ic}_i}{\max\{\mathrm{xc}_i, \mathrm{ic}_i\}}, \quad \text{if } |C_j| > 1,$$

where C_j is the cluster to which X_i belongs to. Denote $s_i = 0$ if $|C_j| = 1$.

From the definition, it can be seen that $-1 \leq s_i \leq 1$. If X_i is much much closer to points in its own clusters as compared to the other clusters, then s_i will be closer to one. On the other hand, when it is closer to points in any other cluster on average compared to points in its own cluster, then s_i will be negative. The overall silhouette score of the clustering can then be written as:

$$s(\mathcal{C}) = \frac{1}{n} \sum_{i=1}^{n} s_i.$$

But more information can be gleaned by looking at the overall silhoutte graph as in Figure 9. Visual inspection of this silhouette graph might provide more insight into the goodness of a clustering compared to the single numeric silhoutte score, for instance, by indicating if there are many points with poor fit to their clusters.

2.3 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 7 *Suppose that* $\mathbb{P}(||X_i||^2 \leq B) = 1$ *for some* $B < \infty$. *Then*

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

for some $c > 0$.

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

This proof is due to Linder, Lugosi and Zeger (1994), and follows along standard VC theory techniques.

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

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

Theorem 8 (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 (global optimum) k-means is risk consistent in the sense that $R(\widehat{C}) - R(C^*) \xrightarrow{P} 0$, as long as $k = o(n/(d \log n))$. 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 k-means algorithm can be generalized in many ways. For example, if we replace the L_2 norm with the L_1 norm we get k-medians clustering. We will not discuss these extensions here.

2.4 Overfitting and Merging

The best way to use k-means clustering is to "overfit then merge." Don't think of the k in k-means as the number of clusters. Think of it as a tuning parameter. k-means clustering works much better if we:

1. Choose k large
2. merge close clusters

This eliminates the sensitivity to the choice of k and it allows k-means to fit clusters with arbitrary shapes. For a theoretical underpinning of this approach, see Aragam, Chen, Xing, Ravikumar, Annals of Statistics, 2019 (more on this in the next section).

2.5 k-Means: Population Perspective

If we think about what K-means does at the population or distribution level, it could be viewed as quantization or discretization: obtaining k centroids around each of which there is a lot of probability mass, hopefully tightly concentrated. A density model with a similar perspective is a Gaussian mixture model, where the mixture component means could be viewed as centroids around which the density locally concentrates. Indeed, k-means could be derived as an asymptotic limit (with variance going to zero) of an algorithm to estimate Gaussian mixture models. We will thus study mixture models next.

3 Mixture Models

Simple cluster structure can be discovered using mixture models. We start with a simple example. We flip a coin with success probability π. 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

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

which is called a mixture of two densities p_1 and p_0. Figure 10 shows a mixture of two Gaussians distribution.

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

$$p(x) = \sum_{z=0,1} p(x, z) = \sum_{z=0,1} p(x|z)p(z) \tag{12}$$

where $p(x|Z = 0) := p_0(x)$, $p(x|Z = 1) := p_1(x)$ and $p(z) = \pi^z(1 - \pi)^{1-z}$. Equation (12) 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

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

where the mixing coefficients $\pi_j \geq 0$, $\sum_{j=0}^{K-1} \pi_j = 1$ and $\psi = (\pi_0, \ldots, \pi_{K-1}, \theta_0, \ldots, \theta_{K-1})$ 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.

3.1 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

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

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

A mixture of d-dimensional multivariate Gaussians is

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

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.

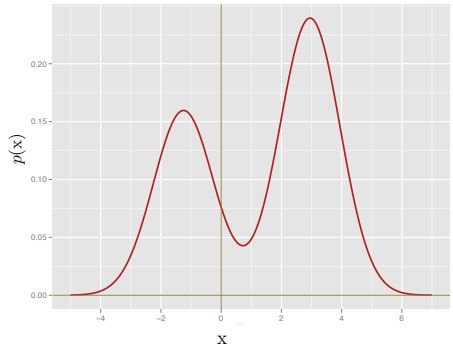


Figure 10: A mixture of two Gaussians, $p(x) = \frac{2}{5}\phi(x; -1.25, 1) + \frac{3}{5}\phi(x; 2.95, 1)$.

3.2 Maximum Likelihood Estimation

A finite mixture model $p_\psi(x)$ has parameters $\psi = (\pi_0, \ldots, \pi_{K-1}, \theta_0, \ldots, \theta_{K-1})$. The likelihood of ψ based on the observations $X_1, \ldots, X_n$ is

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

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 π_j. However, for fixed $\pi_0, \ldots, \pi_{K-1}$, 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.

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

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. To discuss this algorithm, we will re-write the statistical model for a mixture of two Gaussians in terms of latent variables as

$$Z \sim \text{Bernouli}(\pi), \tag{13}$$

$$X|Z = j \sim N(\mu_j, \Sigma_j) \quad \text{for } j = 0, 1. \tag{14}$$

Define

$$p(x|Z = 1) := p_{\mu_1, \Sigma_1}(x) \quad \text{and} \quad p(x|Z = 0) := p_{\mu_0, \Sigma_0}(x).$$

Let $X_1, \ldots, X_n$ be the observed data and let $Z_1, \ldots, Z_n$ be the "missing data". There are two types of unknowns: (i) the parameter vector $\psi = (\pi, \mu_0, \mu_1)^T$, 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 then 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).

Let us consider the simpler setting where $\Sigma_0 = \Sigma_1 = I$.

The Expectation-Maximization Algorithm for the Mixture of Two Gaussians

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

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

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

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

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

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

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

until convergence.

It can be seen that this is a "softer" version of k-means, where γ_i is a softer cluster assignment of each point to the k centroids, while the corresponding centroids also average the softer cluster assignments.

While the algorithm can be slow to converge, its simplicity, flexibility, and the fact that it doesn't require a choice of step size make it a convenient choice for many estimation problems. Nonetheless, 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 principle, there are polynomial time algorithms for finding good estimates of ψ based on spectral methods and the method of moments. It appears that, at least so far, these methods are not yet practical enough to be used in routine data analysis.

Example. The data are measurements on duration and waiting time of eruptions of the Old Faithful geyser 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 11 illustrates the fitted densities for all the six steps. We see that even though the starting density is unimodal, it quickly becomes bimodal.

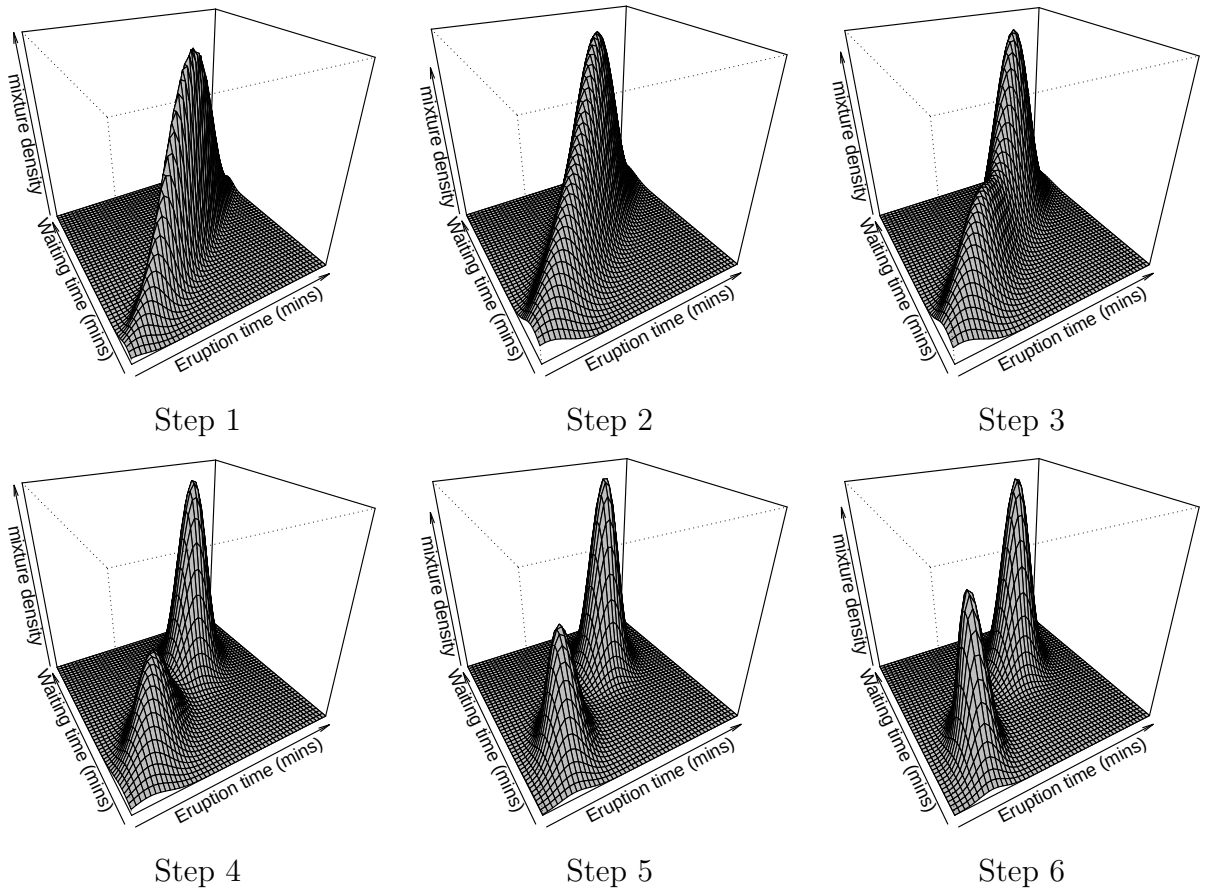


Figure 11: Fitting a mixture of two Gaussians on the Old Faithful Geyser data. The initial values are $\pi_0 = \pi_1 = 0.5$. $u_0 = (4, 70)^T$, $u_1 = (3, 60)^T$, $\Sigma_1 = \Sigma_2 = \begin{pmatrix} 0.8 & 7 \\ 7 & 70 \end{pmatrix}$. We see that even though the starting density is not bimodal, the EM algorithm converges quickly to a bimodal density.

3.3 The Twilight Zone

Mixtures models are conceptually simple but they have some strange properties.

Computation. Finding the mle is NP-hard.

Infinite Likelihood. Let $p_\psi(x) = \sum_{j=1}^k \pi_j \phi(x; \mu_j, \sigma_j^2)$, be a mixture of Gaussians. Let $\mathcal{L}(\psi) = \prod_{i=1}^n p_\psi(X_i)$ be the likelihood function based on a sample of size n. Then $\sup_\psi \mathcal{L}(\psi) = \infty$. To see this, set $\mu_j = X_1$ for some j. Then $\phi(X_1; \mu_j, \sigma_j^2) = (\sqrt{2\pi}\sigma_j)^{-1}$. Now let $\sigma_j \to 0$. We have $\phi(X_1; \mu_j, \sigma_j^2) \to \infty$. Therefore, the log-likelihood is unbounded. This behavior is very different from a typical parametric model. Fortunately, if we define the maximum likelihood estimate to be a mode of $\mathcal{L}(\psi)$ in the interior of the parameter space, we get a well-defined estimator.

Multimodality of the Density. Consider the mixture of two Gaussians

$$p(x) = (1-\pi)\phi(x; \mu_1, \sigma^2) + \pi\phi(x; \mu_0, \sigma^2).$$

You would expect $p(x)$ to be multimodal but this is not necessarily true. The density $p(x)$ is unimodal when $|\mu_1 - \mu_2| \leq 2\sigma$ and bimodal when $|\mu_1 - \mu_2| > 2\sigma$. One might expect that the maximum number of modes of a mixture of k Gaussians would be k. However, there are examples where a mixture of k Gaussians has more than k modes. In fact, Edelsbrunner, Fasy and Rote (2012) show that the relationship between the number of modes of p and the number of components in the mixture is very complex.

Nonintinuitive Group Membership. Our motivation for studying mixture modes in this chapter was clustering. But one should be aware that mixtures can exhibit unexpected behavior with respect to clustering. Let

$$p(x) = (1-\pi)\phi(x; \mu_1, \sigma_1^2) + \pi\phi(x; \mu_2, \sigma_2^2).$$

Suppose that $\mu_1 < \mu_2$. We can classify an observation as being from cluster 1 or cluster 2 by computing the probability of being from the first or second component, denoted $Z = 0$ and $Z = 1$. We get

$$\mathbb{P}(Z = 0|X = x) = \frac{(1-\pi)\phi(x; \mu_1, \sigma_1^2)}{(1-\pi)\phi(x; \mu_1, \sigma_1^2) + \pi\phi(x; \mu_2, \sigma_2^2)}.$$

Define $Z(x) = 0$ if $\mathbb{P}(Z = 0|X = x) > 1/2$ and $Z(x) = 1$ otherwise. When σ_1 is much larger than σ_2, Figure 12 shows $Z(x)$. We end up classifying all the observations with large X_i to the leftmost component. Technically this is correct, yet it seems to be an unintended consequence of the model and does not capture what we mean by a cluster.

Improper Posteriors. Bayesian inference is based on the posterior distribution $p(\psi|X_1, \ldots, X_n) \propto \mathcal{L}(\psi)\pi(\psi)$. Here, $\pi(\psi)$ is the prior distribution that represents our knowledge of ψ before seeing the data. Often, the prior is improper, meaning that it does not have a finite integral.

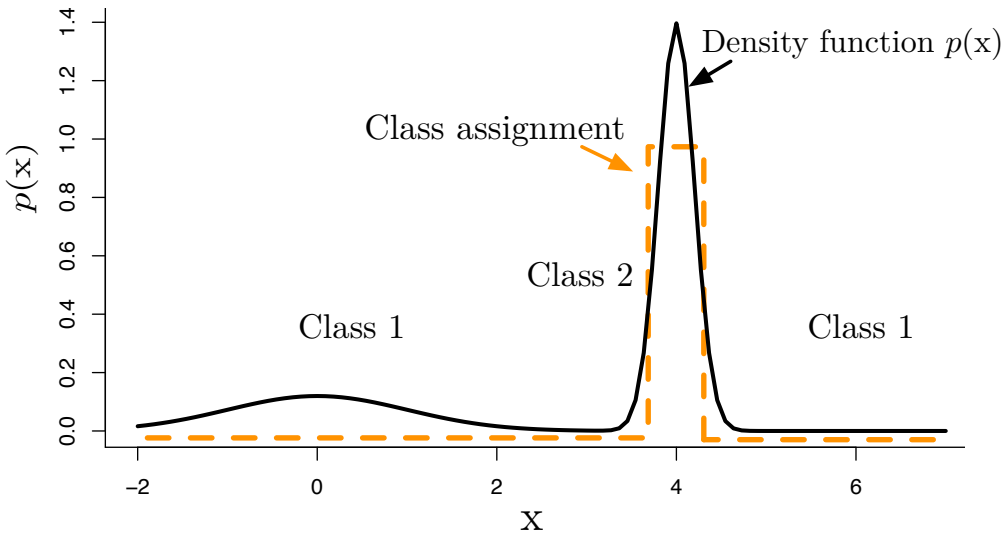


Figure 12: Mixtures are used as a parametric method for finding clusters. Observations with $x = 0$ and $x = 6$ are both classified into the first component.

For example, suppose that $X_1, \ldots, X_n \sim N(\mu, 1)$. It is common to use an improper prior $\pi(\mu) = 1$. This is improper because

$$\int \pi(\mu) d\mu = \infty.$$

Nevertheless, the posterior $p(\mu|\mathcal{D}_n) \propto \mathcal{L}(\mu)\pi(\mu)$ is a proper distribution, where $\mathcal{L}(\mu)$ is the data likelihood of μ. In fact, the posterior for μ is $N(\overline{X}, 1/\sqrt{n})$ where $\overline{x}$ is the sample mean. The posterior inferences in this case coincide exactly with the frequentist inferences. In many parametric models, the posterior inferences are well defined even if the prior is improper and usually they approximate the frequentist inferences. Not so with mixtures. Let

$$p(x; \mu) = \frac{1}{2}\phi(x; 0, 1) + \frac{1}{2}\phi(x; \mu, 1). \tag{15}$$

If $\pi(\mu)$ is improper then so is the posterior. Moreover, Wasserman (2000) shows that the only priors that yield posteriors in close agreement to frequentist methods are data-dependent priors.

Nonidentifiability. A model $\{p_\theta(x) : \theta \in \Theta\}$ is identifiable if

$$\theta_1 \neq \theta_2 \quad \text{implies} \quad P_{\theta_1} \neq P_{\theta_2}$$

where P_θ is the distribution corresponding to the density p_θ. Mixture models are nonidentifiable in two different ways. First, there is nonidentifiability due to permutation of labels. For example, consider a mixture of two univariate Gaussians,

$$p_{\psi_1}(x) = 0.3\phi(x; 0, 1) + 0.7\phi(x; 2, 1)$$

and

$$p_{\psi_2}(x) = 0.7\phi(x; 2, 1) + 0.3\phi(x; 0, 1),$$

then $p_{\psi_1}(x) = p_{\psi_2}(x)$ even though $\psi_1 = (0.3, 0.7, 0, 2, 1)^T \neq (0.7, 0.3, 2, 0, 1)^T = \psi_2$. This is not a serious problem although it does contribute to the multimodality of the likelihood.

A more serious problem is local nonidentifiability. Suppose that

$$p(x; \pi, \mu_1, \mu_2) = (1 - \pi)\phi(x; \mu_1, 1) + \pi\phi(x; \mu_2, 1). \tag{16}$$

When $\mu_1 = \mu_2 = \mu$, we see that $p(x; \pi, \mu_1, \mu_2) = \phi(x; \mu)$. The parameter π has disappeared. Similarly, when $\pi = 1$, the parameter μ_2 disappears. This means that there are subspaces of the parameter space where the family is not identifiable. This local nonidentifiability causes many of the usual theoretical properties— such as asymptotic Normality of the maximum likelihood estimator and the limiting χ^2 behavior of the likelihood ratio test— to break down. For the model (16), there is no simple theory to describe the distribution of the likelihood ratio test for $H_0 : \mu_1 = \mu_2$ versus $H_1 : \mu_1 \neq \mu_2$. The best available theory is very complicated. However, some progress has been made lately using ideas from algebraic geometry (Yamazaki and Watanabe 2003, Watanabe 2010).

The lack of local identifiabilty causes other problems too. For example, we usually have that the Fisher information is non-zero and that $\widehat{\theta} - \theta = O_P(n^{-1/2})$ where $\widehat{\theta}$ is the maximum likelihood estimator. Mixture models are, in general, irregular: they do not satisfy the usual regularity conditions that make parametric models so easy to deal with. Here is an example from Chen (1995).

Consider a univariate mixture of two Gaussians distribution:

$$p_\theta(x) = \frac{2}{3}\phi(x; -\theta, 1) + \frac{1}{3}\phi(x; 2\theta, 1).$$

Then it is easy to check that $I(0) = 0$ where $I(\theta)$ is the Fisher information. Moreover, no estimator of θ can converge faster than $n^{-1/4}$ if the number of components is not known in advance. Compare this to a Normal family $\phi(x; \theta, 1)$ where the Fisher information is $I(\theta) = n$ and the maximum likelihood estimator converges at rate $n^{-1/2}$. Moreover, the distribution of the mle is not even well understood for mixture models. The same applies to the likelihood ratio test.

Mixture Models: Use With Caution. Mixture models can have very unusual and unexpected behavior. This does not mean that we should not use mixture modes. Indeed, mixture models are extremely useful. However, when you use mixture models, it is important to keep in mind that many of the properties of models that we often take for granted, may not hold.

If you are going to use mixture models, it is worthwhile remembering the words of Rod Serling:

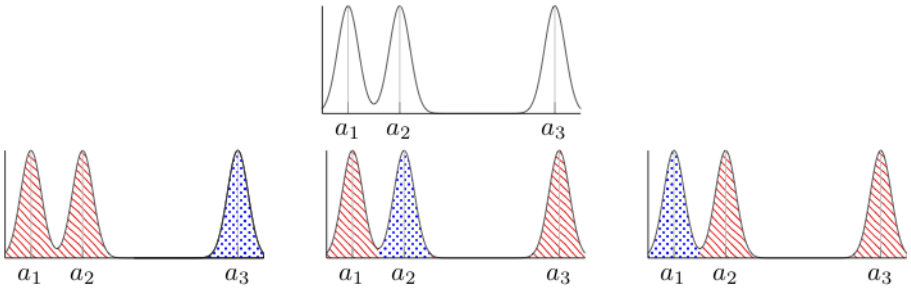


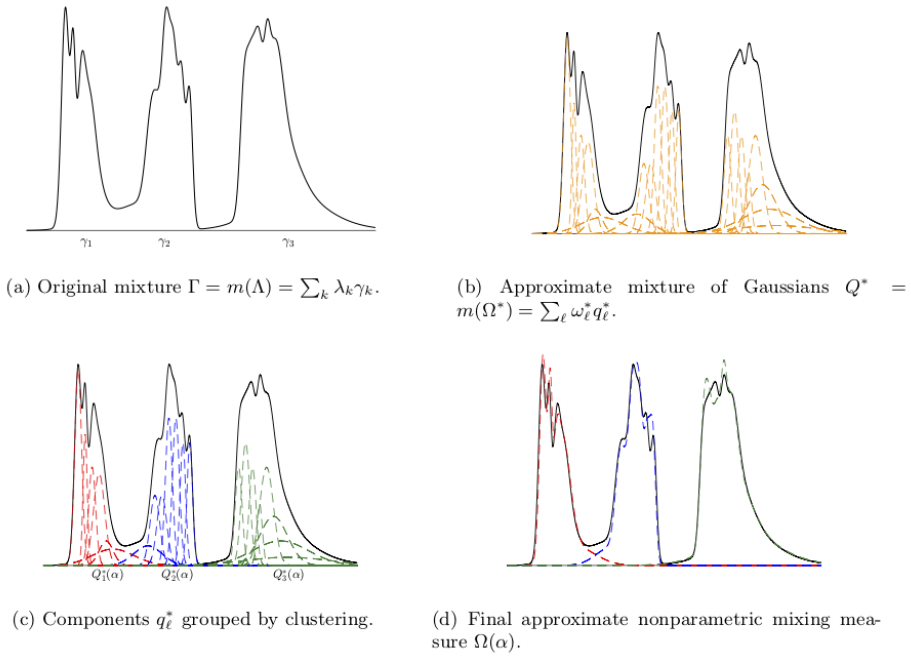
Figure 13: A distribution that is identifiable as a mixture of Gaussians, but not identifiable as a mixture of two sub-Gaussians.

> There is a fifth dimension beyond that which is known to man. It is a dimension as vast as space and as timeless as infinity. It is the middle ground between light and shadow, between science and superstition, and it lies between the pit of man's fears and the summit of his knowledge. This is the dimension of imagination. It is an area which we call the Twilight Zone.

4 Nonparametric Mixture Models

Viewed from an information-theoretic lens, in clustering, we are asking: when can we recover the unseen cluster assignment Y given just the input X? Without much loss of generality, this can be cast as a non-parametric mixture model estimation problem. To see this, suppose we are given some random vector X, and denote the latent clustering assignment variable as Y, that say takes k values. It can then be seen that Y specifies a mixture model: $P(X) = \sum_{j=1}^{k} P(Y=j)\, P(X|Y=j)$, so if we are able to estimate the mixture components, we would be able to recover the clustering corresponding to Y. This is an identifiability question: given the mixture components $\{P(X|Y)\}$ (and mixture weights) there is obviously a unique input distribution $P(X)$. When is there a unique set of mixture components (and is there a practical recover these) given just $P(X)$? Obviously, without any information about the mixture components, the answer is no, since there are many possible mixture models that could have given rise to $P(X)$.

With parametric mixture models, we are given the additional side information that the mixture component distributions lie in a specific parametric family.Classical results on identifiability of mixture models (Yakowitz and Spragins, 1968; Teicher, 1963) state that so long as the specific family of distributions is such that the CDFs of the individual distributions are linearly independent over $\mathbb{R}$, the corresponding mixture models are identifiable. This



(a) Original mixture $\Gamma = m(\Lambda) = \sum_k \lambda_k \gamma_k$.

(b) Approximate mixture of Gaussians $Q^* = m(\Omega^*) = \sum_\ell \omega_\ell^* q_\ell^*$.

(c) Components q_ℓ^* grouped by clustering.

(d) Final approximate nonparametric mixing measure $\Omega(\alpha)$.

Figure 14: Estimating Non-parametric Mixture Models via Overfitted Gaussian Mixtures

holds true for typical parametric models (e.g. Gaussian mixture models), so that the corresponding mixture model is indeed identifiable: just given $P(X)$, we can in principle recover the mixture components. But the moment we go to non-parametric models, identifiability typicaly fails to hold, even for very simple non-parametric classes.

For instance, consider the mixture of three Gaussians in Figure 13. While we can write the distribution uniquely as a mixture of three Gaussians, we can also write it as three *different* equally valid representations as a mixture of two sub-Gaussians. Thus, the distribution is not identifiable with respect to a mixture model over sub-Gaussian distributions. Note that even if we assume the number of components are known , and the component means are well-separated, this would still remain non-identifiable: consider two components, and where the third component is arbitrarily far to the right.

(Aragam, Chen, Xing, Ravikumar, Annals of Statistics 2020) provide conditions under which non-parametric mixture models can indeed be identifiable. Loosely: Gaussian mixture models are identifiable, and moreover are dense in the space of all distributions. Thus, each non-parametric mixture component can be approximated well via a mixture of Gaussians, and consequently, the overall distribution as a mixture of Gaussians in turn. Thus, one can project any non-parametric distribution onto a mixture of a large number of Gaussians. Then so long as one could cluster these Gaussian components, then one could identify the

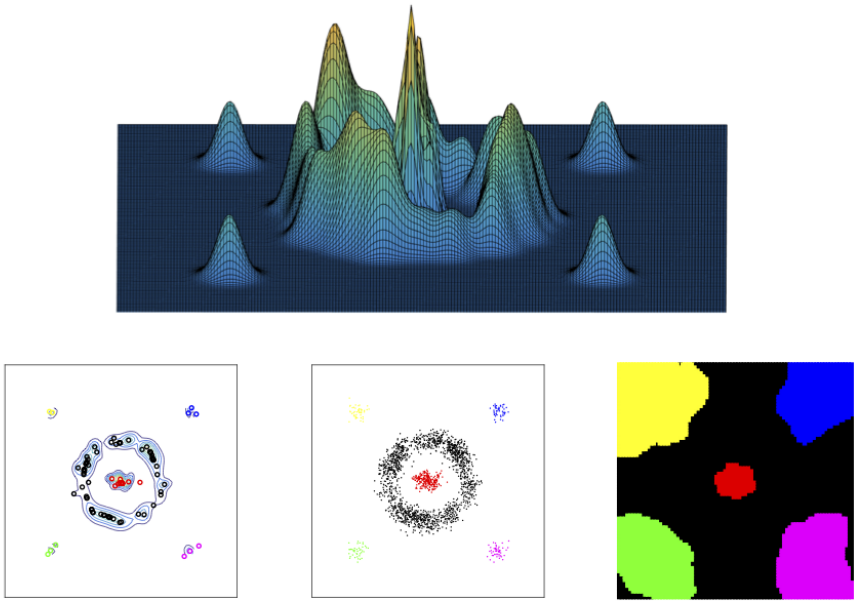


Figure 15: (Top) Density plot of the original mixture density. (Left) Contour plot of overfitted Gaussian mixture approximation, centers marked with small circles. (Middle) Original data color coded by the approximate Bayes optimal partition. (Right) Estimated Bayes optimal partition, visualized by color-coding the input space by estimated cluster membership.

individual non-parametric mixture components with each cluster of Gaussian components. Their paper essentially provided the regularity conditions under which the above very natural procedure is guaranteed to identify the non-parametric mixture components.

The overall algorithm, as shown in Figure 14, then is:

1. Estimate an overfitted mixture of Gaussians to the given data
2. Cluster the Gaussian densities with respect to say the Hellinger metric
3. Identify each cluster with a non-parametric mixture component

Figure 15 provides an example of a difficult clustering example, where the above approach works well.

Given the connection between Gaussian mixture models and K-means, this also suggestive of a theoretical underpinning for the overfitted Kmeans strategy mentioned in the previous section: fit a large number of clusters via K-means, and then merge the (sub)-clusters.

5 Density-Based Clustering I: Level Set Clustering

Let p be the density if the data. 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 (2014) 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 16 and 17.

5.1 Theory

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

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

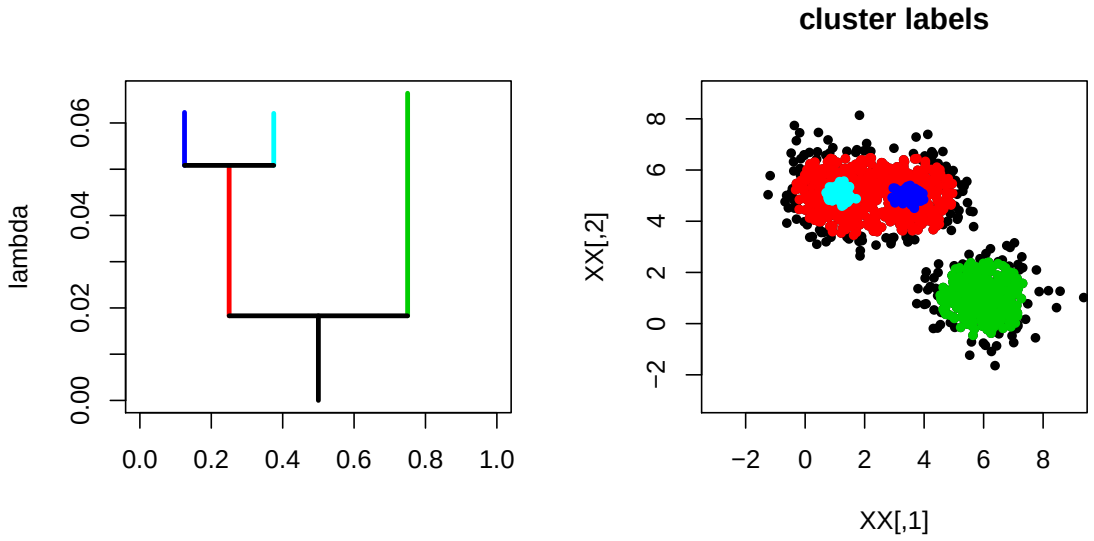


Figure 16: DeBaClR in two dimensions.

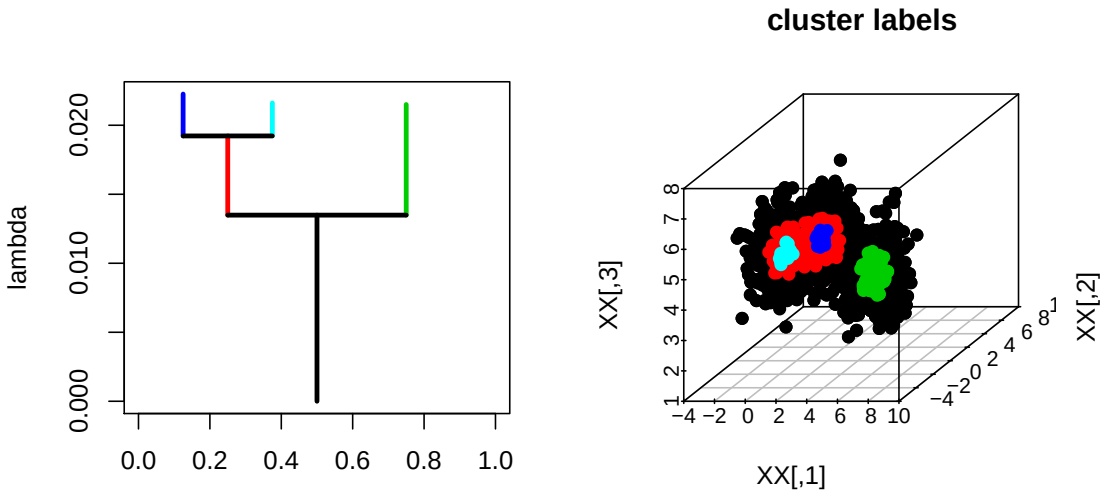


Figure 17: DeBaClR in three dimensions.

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 $\delta < a$,

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

Theorem 9 *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 $||x - y|| \leq Cr_n$. □

5.2 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 18.

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 18. Points near the diagonal correspond to modes with short lifetimes. We might kill modes with lifetimes smaller than the noise

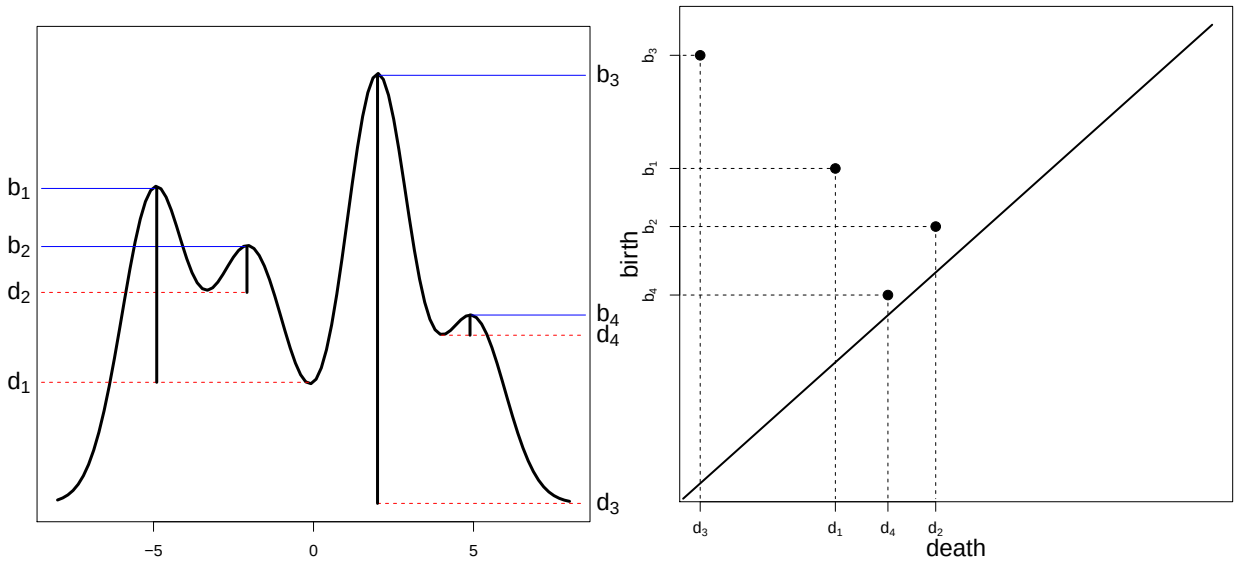


Figure 18: *Starting at the top of the density and moving down, each mode has a birth time b and a death time d. The persistence diagram (right) plots the points $(d_1, b_1), \ldots, (d_4, b_4)$. Modes with a long lifetime are far from the diagonal.*

level, as captured by the deviation of the density estimate, and the true density. We measure this via the bootstrap quantile ϵ_α defined by

$$\epsilon_\alpha = \inf\left\{ z : \ \frac{1}{B}\sum_{b=1}^{B} I\Big(||\widehat{p}_h^{*b} - \widehat{p}_h||_\infty > z\Big) \leq \alpha \right\}. \tag{17}$$

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 (2014).)

6 Density-Based Clustering II: Modes

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.

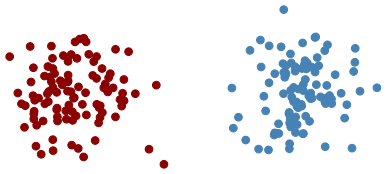


Figure 19: A synthetic example with two "blob-like" clusters.

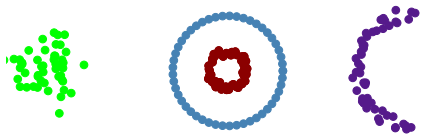


Figure 20: A synthetic example with four clusters with a variety of different shapes.

6.1 Mode Clustering

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

$$\pi_x'(t) = \nabla p(\pi_x(t)). \tag{18}$$

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

Equation (18) 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

$$\text{dest}(x) = \lim_{t \to \infty} \pi_x(t). \tag{19}$$

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

$$\mathcal{A}_j = \Big\{ x : \ \text{dest}(x) = m_j \Big\}. \tag{20}$$

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 21. 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 $C_j = \big\{ X_i : \ X_i \in \widehat{\mathcal{A}}_j \big\}$.

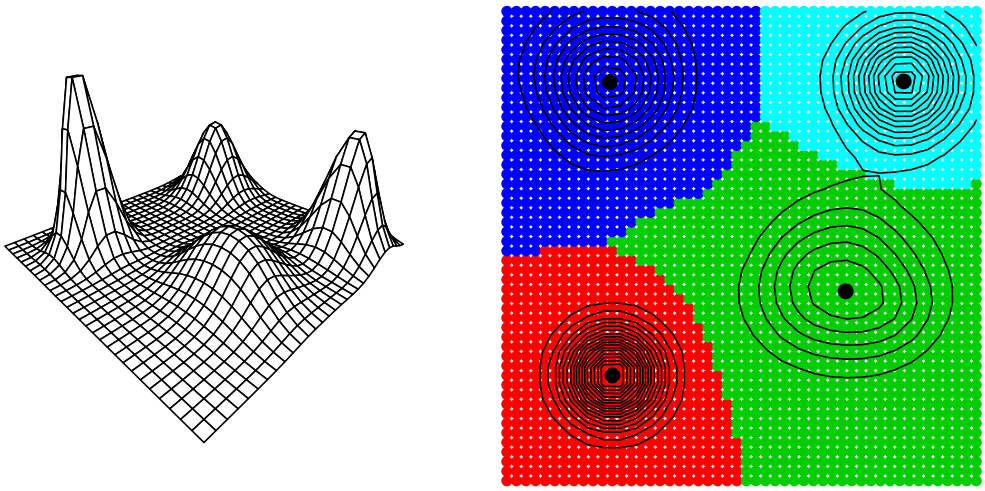


Figure 21: *The left plot shows a function with four modes. The right plot shows the ascending manifolds (basins of attraction) corresponding to the four modes.*

Recall that the kernel density estimator is

$$\widehat{p}(x) \equiv \widehat{p}_h(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^d} K\left(\frac{||x - X_i||}{h}\right) \tag{21}$$

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

$$p_h(x) = \mathbb{E}[\widehat{p}_h(x)] = \int K(t)p(x + th)dt. \tag{22}$$

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 22, and is essentially a fixed point iteration obtained by setting the gradient of the kernel estimate to zero. 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 23.

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 24 and 25.

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 $\phi(x, 0) = x$

[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 $K_H(x) = |H|^{-\frac{1}{2}} K(H^{-\frac{1}{2}}x)$.

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 22: *The Mean Shift Algorithm.*

and $\phi(\phi(x,t),s) = \phi(x, s+t)$. The latter is called the semi-group property.

6.2 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.

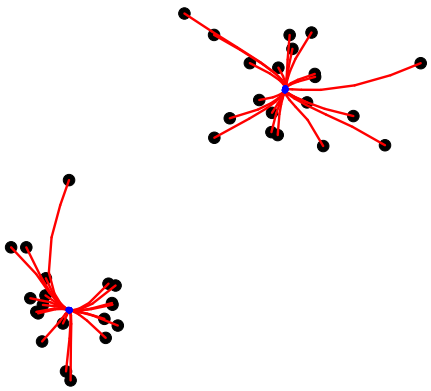


Figure 23: A simple example of the mean shift algorithm.

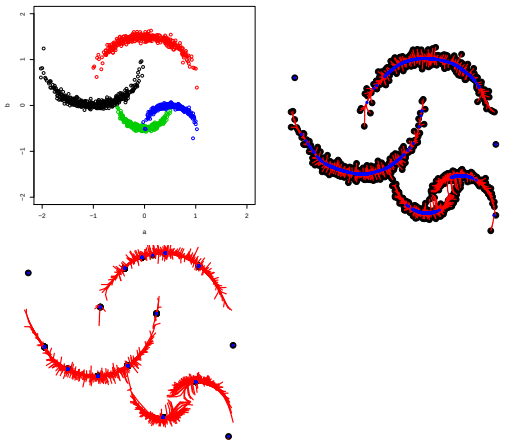


Figure 24: The crescent data example. Top left: data. Top right: a few steps of mean-shift. Bottom left: a few steps of blurred mean-shift.

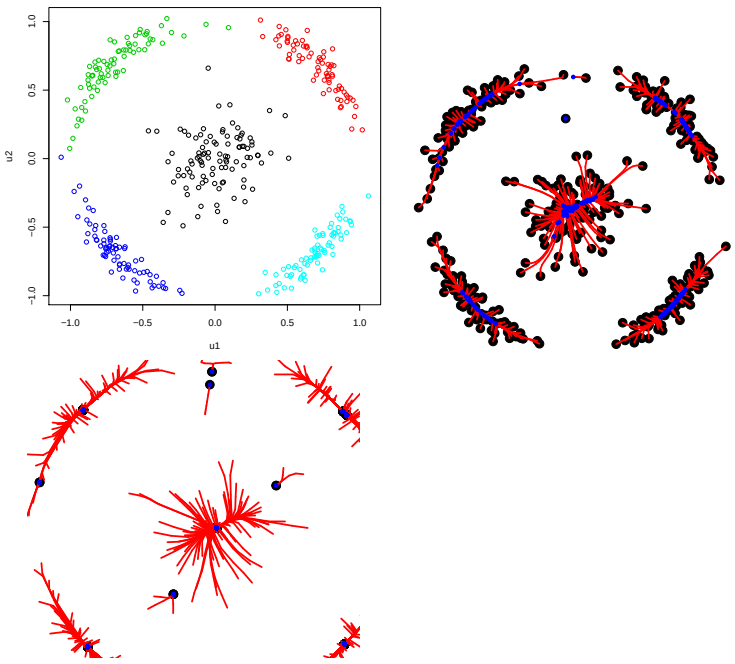


Figure 25: The Broken Ring example. Top left: data. Top right: a few steps of mean-shift. Bottom left: a few steps of blurred mean-shift.

Theorem 10 (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

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

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

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

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

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

6.3 Theoretical Analysis

How well can we estimate the modes?

Theorem 11 *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.

7 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 12 *Figure 26 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.*

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.

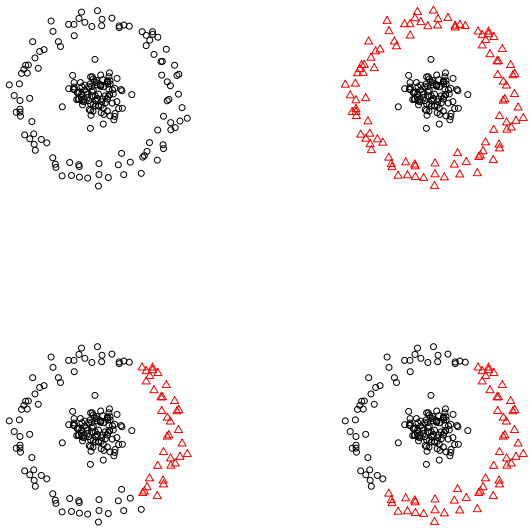


Figure 26: Hierarchical clustering applied to two noisy rings. Top left: the data. Top right: two clusters from hierarchical clustering using single linkage. Bottom left: average linkage. Bottom right: complete linkage.

7.1 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. Loosely, higher up in the hierarchical clustering tree, clusters get large enough that they might intersect with multiple connected components of the density level sets. So Hartigan (1981) suggested that to obtain high density clusters from single linkage clustering, we need to make it more "robust," and remove points from lower density regions that would not have made the corresponding density level set.

Chaudhuri and DasGupta (2010) formalized this, and suggested the following modified version of hierarchical clustering that attempts to fix this problem, and which can be seen to be very similar to density clustering introduced earlier:

1. For each x_i, let $r_k(x_i) = \{\inf r : B(x_i, r) \text{ contains } k \text{ data points}\}$.
2. As r increases from 0 to ∞:
   (a) Construct a graph G_r with nodes $\{x_i : r_k(x_i) \leq r\}$. Include edge (x_i, x_j) if $\|x_i - x_j\| \leq \alpha r$.
   (b) Output $\widehat{C}(r)$ as the connected components of G_r.

Here, the neighborhood radius parameter r ranging from 0 to ∞ corresponds to the level set parameter λ in density clustering ranging from 0 to ∞. In both cases, we remove points corresponding to lower density regions: in this case, by checking if the point has at least k nearest neighbors within that radius. There are two parameters here: k and α. It can be seen that single linkage is obtained by setting $\alpha = 1, k = 2$. More robust versions can be obtained by setting k to be much larger. Chaudhuri and DasGupta (2010) showed that $\alpha \geq \sqrt{2}$, and $k \sim d \log n$ suffices to obtain consistent estimates of high density clusters.

Single linkage hierarchical clustering could also be viewed 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.

8 Spectral Clustering

Spectral clustering refers to a class of clustering methods that use ideas related to eigenvectors of appropriately constructed similarity matrices. An excellent tutorial on spectral clustering, and on which this section heavily relies on, is von Luxburg (2006). 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

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

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

$$L = D - W. \tag{23}$$

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.

Theorem 13 *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 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 connected, for any pair of nodes i, j, there is a path $i_0 = i, i_1, \ldots, i_m = j$, such that $W_{i_\ell i_{\ell+1}} > 0$, which entails that:

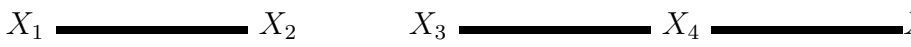
$v(i_\ell) = v(i_{\ell'})$ for all nodes in the path, and consequently that $v_i = v_j$. Since this holds for all i, j, v is constant, and lies in the span of v_1.

Now suppose that K has k components. Let n_j be the number of nodes in components j. We can relabel 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 the connected components of the graph. It is easy to see that for $j = 1, \ldots, k$, $L\, v_j = 0$ so that each v_j is an eigenvector with zero eigenvalue. Suppose that v is any eigenvector with 0 eigenvalue. Arguing as before, v must be constant over each component, so that it lies in the span of $\{v_j\}_{j=1}^k$. □

Example 14 *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

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

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.

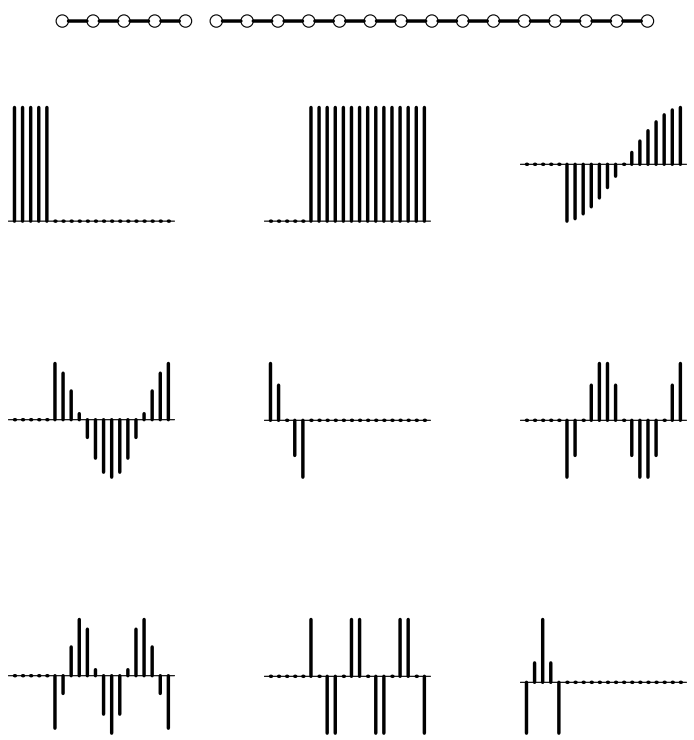


Figure 27: The top shows a simple graph. The remaining plots are the eigenvectors of the graph Laplacian. Note that the first two eigenvectors correspond to the two connected components of the graph.

Smoothness with respect to 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 15 *Figure 27 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 also called geometric graph clustering. Thus one perspective of spectral clustering is 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.

8.1 Spectral Clustering: Feature Transformations

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 16 *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.*

To get further intuition for the random walk Laplacian $\mathcal{L}$, note that it 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

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

The random walk Laplacian $\mathcal{L}(i, j)$ captures how easy it is to move from X_i to X_j. 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 linear operator $\widehat{A} : f \to \widetilde{f}$ is

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

is in turn an estimate of the population linear operator $A : f \to \widetilde{f}$ where

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

Given the form of this linear operator, it can be seen that the lower order eigenvectors of $\mathcal{L}$ are vectors that are smooth relative to the density P. Thus, projecting onto the first few eigenvectors parameterizes in terms of closeness with respect to the underlying density.

Thus, one approach to spectral clustering would be to use the random walk Laplacian to transform the data $\{X_i\}$ to within into a new coordinate system. Denoting the transformed data as $\{\widehat{X}_i\}$, we would like to ensure that if $\widehat{X}_i$ and $\widehat{X}_j$ are close in Euclidean distance, then they are connected by many high density paths through the data.

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 $\widehat{X}_i = (\sqrt{\lambda_j}\, v_j(i))_{j \in [k]}$.
5. Cluster the points $\widehat{X}_1, \ldots, \widehat{X}_n$ using any standard clustering algorithm.

The numbers h (bandwidth for kernel) and k (number of eigenvectors) are tuning parameters. The hope is that clusters are easier to find in the new parameterization.

In this view of 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 17 *Figure 28 shows a simple synthetic example. The top left plot shows the data. We apply spectral clustering with Gaussian weights and bandwidth $h = 3$. The top middle plot shows the first 20 eigenvalues. The top right plot shows the the first versus the second eigenvector. The two clusters are clearly separated. (Because the clusters are so separated, 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 18 *Figure 29 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*

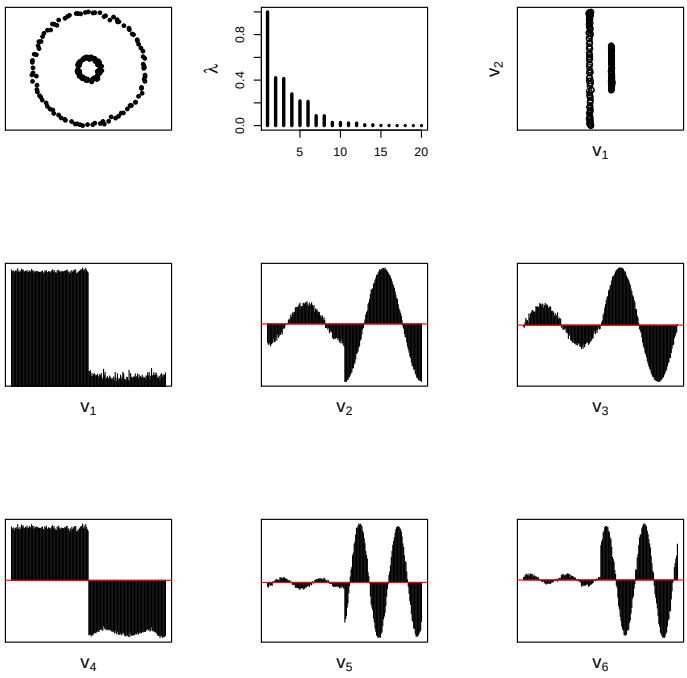


Figure 28: Top left: data. Top middle: eigenvalues. Top right: second versus third eigenvectors. Remaining plots: first six eigenvectors.

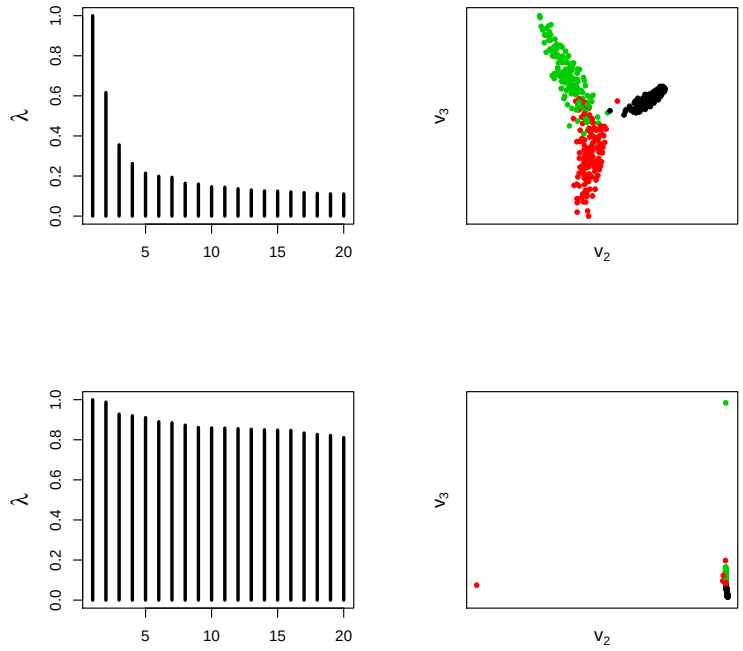


Figure 29: Spectral analysis of some zipcode data. Top: $h = 6$. Bottom: $h = 4$. The plots on the right show the second versus third eigenvector. The three colors correspond to the three digits 1, 2 and 3.

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

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?

9 High-Dimensional Clustering

As usual, interesting and unexpected things happen in high dimensions. The usual methods may break down and even the meaning of a cluster may not be clear.

We will begin by discussing some recent results from Sarkar and Ghosh (arXiv:1612.09121). Suppose we have data coming from k distributions $P_1, \ldots, P_k$. Let μ_r be the mean of P_r and Σ_r be the covariance matrix. Most clustering methods depend on the pairwise distances $||X_i - X_j||^2$. Now,

$$||X_i - X_j||^2 = \sum_{a=1}^{d} \delta(a)$$

where $\delta_a = (X_i(a) - X_j(a))^2$. This is a sum. As d increases, by the law of large numbers we might expect this sum to converge to a number (assuming the features are not too dependent). Indeed, suppose that X is from P_r and Y is from P_s then

$$\frac{1}{\sqrt{d}}||X - Y|| \xrightarrow{P} \sqrt{\sigma_r^2 + \sigma_s^2 + \nu_{rs}}$$

where

$$\nu_{rs} = \lim_{d\to\infty} \frac{1}{d} \sum_{a=1}^{d} ||\mu_r(a) - \mu_s(a)||^2$$

and

$$\sigma_r^2 = \lim_{d\to\infty} \frac{1}{d} \text{trace}(\Sigma_r).$$

Note that $\nu_{rr} = 0$.

Consider two clusters, C_1 and C_2:

X	Y	$\|\|X - Y\|\|$
$X \in C_1$	$Y \in C_1$	$\|\|X - Y\|\| = 2\sigma_1^2$
$X \in C_2$	$Y \in C_2$	$\|\|X - Y\|\| = 2\sigma_2^2$
$X \in C_1$	$Y \in C_2$	$\|\|X - Y\|\| = \sigma_1^2 + \sigma_2^2 + \nu_{12}$

If

$$\sigma_1^2 + \nu_{12} < \sigma_2^2$$

then **every point in cluster 2 is closer to a point in cluster 1 than to other points in cluster 2.** Indeed, if you simulate high dimensional Gaussians, you will see that all the standard clustering methods fail terribly.

What's really going on is that high dimensional data tend to cluster on rings. Pairwise distance methods don't respect rings.

10 Summary

The main clustering methods are:

1. k-means
2. mixture models
3. density-based
4. hierarchical
5. spectral

The multiplicity of methods is because each method corresponds to different if related notions of what a cluster means. They all involve tuning parameters that must be chosen carefully.

11 References

Arias-Castro, E., Mason, D. and Pelletier, B. On the estimation of the gradient lines of a density and the consistency of the mean-shift algorithm. Unpublished Manuscript, 2013.

Chacon, J. Clusters and water flows: a novel approach to modal clustering through Morse theory. arXiv preprint arXiv:1212.1384, 2012.

Chacon, J. and Monfort, P. A comparison of bandwidth selectors for mean shift clustering. arXiv preprint arXiv:1310.7855, 2013.

Chacon, J. and Duong, T. Multivariate plug-in bandwidth selection with unconstrained pilot bandwidth matrices. Test, 19(2):375-398, 2010.

Chacon, J. and Duong, T. and Wand, M. Asymptotics for general multivariate kernel density derivative estimators. Statistica Sinica, 21:807-840, 2011.

Chacon, J. and Duong, T. Data-driven density derivative estimation, with applications to nonparametric clustering and bump hunting. Electronic Journal of Statistics, 7:1935-2524, 2013.

Chazal, F., Guibas, L.J., Oudot, S.Y. and Skraba, P. Persistence-based clustering in riemannian manifolds. In Proceedings of the 27th annual ACM symposium on Computational geometry, pages 97-106. ACM, 2011.

Comaniciu, D. and Meer, P. Mean shift: a robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(5):603-619, may 2002. ISSN 0162-8828. doi: 10.1109/34.1000236.

Donoho, D. and Liu, R. Geometrizing rates of convergence, III. The Annals of Statistics, pages 668-701, 1991.

Carreira-Perpinan, M. (2006). Fast nonparametric clustering with Gaussian blurring mean-shift. Proceedings of the 23rd international conference on Machine learning. 153–160.

Silverman, B. (1981). Using kernel density estimates to investigate multimodality. Journal of the Royal Statistical Society. Series B (Methodological), pages 97-99, 1981.

Fasy, Lecci, Rinaldo, Wasserman, Balakrishnan and Singh (2013). Statistical Inference For Persistent Homology. arXiv:1303.7117.

Wasserman, L. (2000). Asymptotic inference for mixture models by using data-dependent priors. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 62, 159–180.

Rousseeuw, P. J. (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. Computational and Applied Mathematics, 20: 5365.

B. Aragam, C. Dan, E. Xing, P. Ravikumar (2020) Identifiability of Nonparametric Mixture Models and Bayes Optimal Clustering. Annals of Statistics, 2019.

```
@inproceedings{edelsbrunner2012add,
  title={Add isotropic Gaussian kernels at own risk:
  More and more resilient modes in higher dimensions},
  author={Edelsbrunner, Herbert and Fasy, Brittany Terese and Rote, G{\"u}nter},
  booktitle={Proceedings of the 2012 symposuim on Computational Geometry},
  pages={91--100},
  year={2012},
  organization={ACM}
}
```

```
@article{yamazaki2003singularities,
  title={Singularities in mixture models and
    upper bounds of stochastic complexity},
  author={Yamazaki, Keisuke and Watanabe, Sumio},
  journal={Neural networks},
  volume={16},
  number={7},
  pages={1029--1038},
  year={2003},
```

```
  publisher={Elsevier}
}

@book{watanabe2009algebraic,
  title={Algebraic geometry and statistical learning theory},
  author={Watanabe, Sumio},
  volume={25},
  year={2009},
  publisher={Cambridge University Press}
}




@article{chen1995optimal,
  title={Optimal rate of convergence for finite mixture models},
  author={Chen, Jiahua},
  journal={The Annals of Statistics},
  pages={221--233},
  year={1995},
  publisher={JSTOR}
}




@article{dacunha1999testing,
  title={Testing the order of a model using locally conic parametrization: population mi
  author={Dacunha-Castelle, Didier and Gassiat, Elisabeth},
  journal={The Annals of Statistics},
  volume={27},
  number={4},
  pages={1178--1209},
  year={1999},
  publisher={Institute of Mathematical Statistics}
}

@article{dacunha1997estimation,
  title={The estimation of the order of a mixture model},
  author={Dacunha-Castelle, Didier and Gassiat, Elisabeth},
  journal={Bernoulli},
```

```
  pages={279--299},
  year={1997},
  publisher={JSTOR}
}

@article{keribin2000consistent,
  title={Consistent estimation of the order of mixture models},
  author={Keribin, Christine},
  journal={Sankhy{\=a}: The Indian Journal of Statistics, Series A},
  pages={49--66},
  year={2000},
  publisher={JSTOR}
}
```