Topic 1: Clustering

Welcome to Week 6

Dr Pavel Krivitsky gives you a brief overview of topics and concepts we'll be covering in this week.

Transcript

Weekly learning outcomes

- Explain different types of clustering and their advantages and disadvantages.
- Fit, visualise, and evaluate a K-Means, K-Medioids and hierarchical clustering to data.
- Translate substantive information about clusters into multivariate normal model assumptions.
- Fit, visualise, and evaluate model-based clustering to data.
- Select the optimal number of clusters and cluster model.
- Select a copula appropriate to the data.
- Estimate, visualise, diagnose, and simulate from a copula model.
- Explain the assumptions underlying the above inferential procedures and check them.

Topics we will cover are:

- Topic 1: Clustering
- Topic 2: Copula methods

Optional reading

An alternative presentation of the concepts for this week can be found in:

Johnson, R. A., & Wichern, D. (2008). *Applied Multivariate Statistical Analysis* (6th ed.). Pearson Prentice Hall.

- 12.1 - 12.5

Additional software demonstration of model-based clustering can be found in:

Scrucca, L., Fop, M., Murphy, T. B., & Raftery, A. E. (2016). mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1), 289.

All readings are available from the course Leganto reading list. Please keep in mind that you will need to be logged into Moodle to access the Leganto reading list.

Questions about this week's topics?

This week's topics were prepared by Dr P. Krivitsky. If you have any questions or comments, please post them under Discussion or email directly: p.krivitsky@unsw.edu.au

Cluster analysis

The goal of cluster analysis is to identify groups in data. In contrast to SVMs and discriminant analysis, no preexisting group labels are provided. This makes it an example of *unsupervised learning*.

The input of cluster analysis is therefore an *unlabelled* sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$, and the output is a *grouping* of observations such that more similar (in some sense) observations are placed in the group. That is, cluster analysis assigns to each $\boldsymbol{x}_i$ a group index $G_i \in 1, \ldots, K$ such that if $G_i = G_j$, $\boldsymbol{x}_i$ *and* $\boldsymbol{x}_j$ are "on average" more similar in some sense than if $G_i \neq G_j$.

Throughout this week, we will use the following additional notation.

$\boldsymbol{G} = (G_1, \ldots, G_n)^\top$: a column vector of cluster memberships.

$S_1, \ldots, S_K$: a *partitioning* of the observations $\{1, \ldots, n\}$ into K non-overlapping sets such that for every $i \in S_k$, $G_i = k$.

$\boldsymbol{S} = (S_1, \ldots, S_K)$: a shorthand for the clustering expressed in terms of sets.

We will consider a taxonomy of approaches to clustering. The "classical" approach is to specify an *algorithm* that assigns observations to clusters. (Often, but not always, an objective function may be defined that is optimised by the algorithm.) Classical approaches can be further subdivided into *hierarchical* clustering, which produces a hierarchy of nested clusterings in a tree which has observations as leaves; and *non-hierarchical*, which merely assigns a label to each point.

The *model-based* approach to clustering is to postulate a *mixture model*—a model consisting of a mixture of probability distributions with different location parameters. The parameters of this model embody information about the clusters (e.g., their means and frequencies), and estimating them enables probabilistic, or *soft* clusterings.

We discuss these approaches in this topic.

“Classical”

Defining a clustering

In order to cluster data—particularly multivariate data—we must first define a *proximity measure*: some function $d(\boldsymbol{x}_1, \boldsymbol{x}_2)$ that determines difference between two observations. (Equivalently we can define a similarity score and negate or invert it.) Here are some common metrics measures:

Euclidean: $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| = \sqrt{\sum_{k=1}^{p} (x_{1k} - x_{2k})^2}$, the "ordinary" straight-line distance.

Taxicab/Manhattan: $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_1 = \sum_{k=1}^{p} |x_{1k} - x_{2k}|$, distance if one is only allowed to travel parallel to the axes (like a taxicab on the Manhattan city grid).

Gower: $p^{-1} \sum_{k=1}^{p} \mathbb{I}(x_{1k} \neq x_{2k})$: for binary measures.

A metric should be substantively meaningful and appropriate for the data. It is also common to scale all of the dimensions (say, to have variance of 1 or to be between 0 and 1) before clustering.

Given these distances, we specify the algorithm that minimises within-cluster and maximises between-cluster distances in some sense — that sense often operationalised in an *objective function*.

Example: K-means

Perhaps the best known clustering algorithm is the K-means. It has the advantage of being simple and intuitive. The objective function that it ultimately minimises (over the partitioning $\boldsymbol{S} = (S_1, \ldots, S_K)$) is

$$\sum_{g=1}^{K} \frac{1}{2|S_g|} \sum_{i,j \in S_g} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2,$$

the sum of squared Euclidean distances between every distinct pair of observations within each cluster, scaled by twice $|S_g|$, the cardinality of (i.e., number of elements in) set S_g. It can be shown (using a decomposition similar to that of ANOVA) that this is equivalent to minimising

$$\sum_{g=1}^{K} \sum_{i \in S_g} \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_{S_g}\|^2, \; \bar{\boldsymbol{x}}_{S_g} = \frac{1}{|S_g|} \sum_{i \in S_g} \boldsymbol{x}_i,$$

which is simply the sum of the squared Euclidean distances between each data point and the mean of its cluster.

The following algorithm often does a good job finding such a clustering:

1. Randomly assign a cluster index to each element of $\boldsymbol{G}^{(0)}$.
2. Calculate cluster means (centroids):

$$\bar{\boldsymbol{x}}_{S_g^{(t-1)}} = \frac{1}{|S_g^{(t-1)}|} \sum_{i \in S_g^{(t-1)}} \boldsymbol{x}_i,\ g = 1, \ldots, K.$$

3. Calculate distances of each data point from each mean:

$$d_{ig} = \|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_{S_g^{(t-1)}}\|,\ i = 1, \ldots, n,\ g = 1, \ldots, K.$$

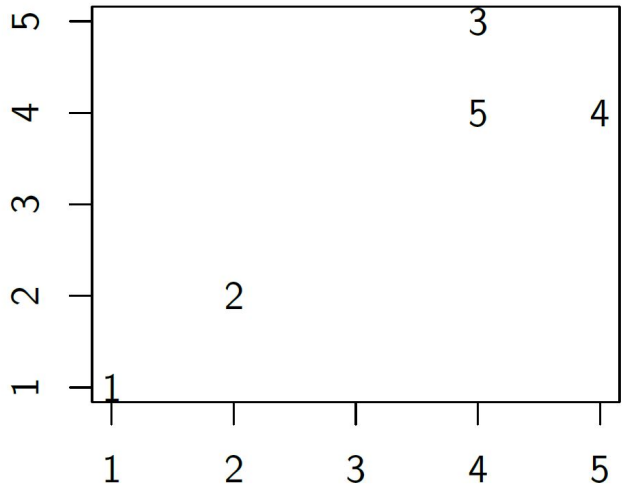
4. Reassign each point to its nearest mean:

$$G_i^{(t)} = \arg\min_g d_{ig}.$$

   (Here and elsewhere, $\arg\min_a h(a)$ is that a which minimises the value of $h(a)$.)
5. Repeat from Step 2 until $\boldsymbol{G}^{(t)} = \boldsymbol{G}^{(t-1)}$.

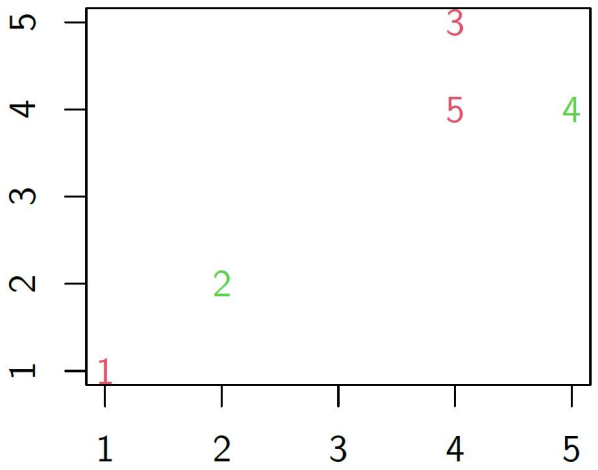
Example 6.1. Consider the following dataset:

Index	V1	V2
1	1	1
2	2	2
3	4	5
4	5	4
5	4	4



We begin by creating an initial clustering at random:

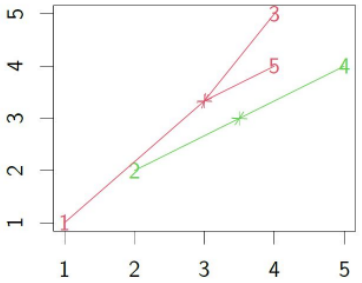
Index	V1	V2	C
1	1	1	1
2	2	2	2
3	4	5	1
4	5	4	2
5	4	4	1



Calculate the centroids:

Index	V1	V2	C
1	1	1	1
2	2	2	2
3	4	5	1
4	5	4	2
5	4	4	1

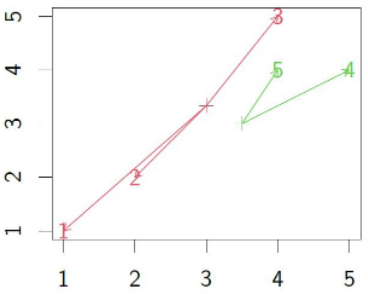
C	V1	V2
1	3.0	3.333333
2	3.5	3.000000



Update the clustering:

Index	V1	V2	C
1	1	1	1
2	2	2	1
3	4	5	1
4	5	4	2
5	4	4	2

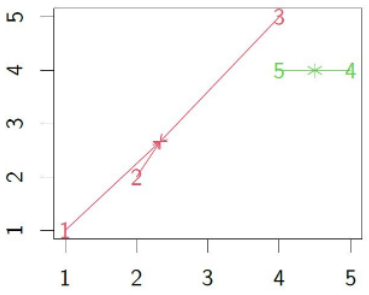
Index	1	2
1	3.073182	3.201562
2	1.666667	1.802776
3	1.943651	2.061553
4	2.108185	1.802776
5	1.201850	1.118034



Calculate the centroids:

Index	V1	V2	C
1	1	1	1
2	2	2	1
3	4	5	1
4	5	4	2
5	4	4	2

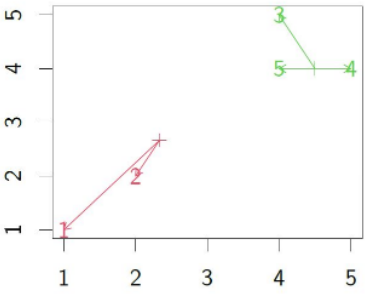
C	V1	V2
1	2.333333	2.666667
2	4.500000	4.000000



Update the clustering:

Index	V1	V2	C
1	1	1	1
2	2	2	1
3	4	5	2
4	5	4	2
5	4	4	2

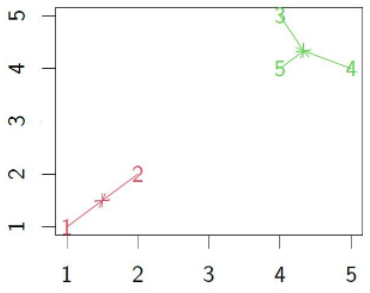
Index	1	2
1	2.134375	4.609772
2	0.745356	3.201562
3	2.867442	1.118034
4	2.981424	0.500000
5	2.134375	0.500000



Calculate the centroids:

Index	V1	V2	C
1	1	1	1
2	2	2	1
3	4	5	2
4	5	4	2
5	4	4	2

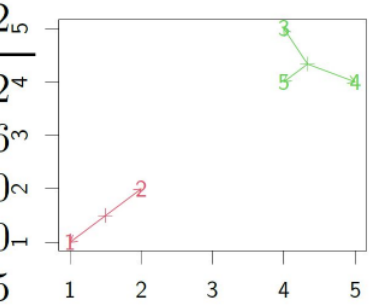
C	V1	V2
1	1.500000	1.500000
2	4.333333	4.333333



Update the clustering:

Index	V1	V2	C
1	1	1	1
2	2	2	1
3	4	5	2
4	5	4	2
5	4	4	2

Index	1	2
1	0.7071068	4.7140452
2	0.7071068	3.2998316
3	4.3011626	0.7453560
4	4.3011626	0.7453560
5	3.5355339	0.4714045



The clustering is unchanged from the previous iteration, so we declare convergence.

Extension: K-medioids

A generalisation of K-means is the K-medioids technique. We define a *medioid* $\tilde{\boldsymbol{x}}_g$ *of cluster* g to be a *specific observation* that has the closest summed distance (however defined) to all other observations in S_g:

$$\tilde{\boldsymbol{x}}_{S_g} = \arg\min_{\boldsymbol{x}_i} \sum_{i \in S_g} d(\boldsymbol{x}_j, \boldsymbol{x}_i).$$

The *Method of* K *– medioids* or *partitioning around medioids (PAM)* minimises the sum of these distances:

$$\arg\min_{\boldsymbol{S}} \sum_{g=1}^{K} \sum_{i \in S_g} d(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_{S_g}).$$

This method is much more expensive computationally than K-means, but it is also more robust to outliers. It is typically fit as follows:

1. Randomly assign a cluster index to each element of $\boldsymbol{G}^{(0)}$.
2. Calculate cluster medioids:

$$\tilde{\boldsymbol{x}}_{S_g^{(t-1)}} = \arg\min_{\boldsymbol{x}_i} \sum_{j \in S_g^{(t-1)}} d(\boldsymbol{x}_j, \boldsymbol{x}_i),\ g = 1, \ldots, K.$$

3. Calculate distances of each data point from each medioid:

$$d_{ig} = d(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_{S_g^{(t-1)}}),\ i = 1, \ldots, n,\ g = 1, \ldots, K.$$

4. Reassign each point to its nearest medioid:

$$G_i^{(t)} = \arg\min_g d_{ig}.$$

5. Repeat from Step 2 until $\boldsymbol{G}^{(t)} = \boldsymbol{G}^{(t-1)}$.

Hierarchical clustering

Hierarchical clustering, instead of partitioning the data into K groups, produces a hierarchy of clusterings whose sizes range from 1 (no splits) to as high as n (every observation its own cluster). This clustering is typically visualised in a *dendrogram*, a tree diagram whose branching represents subdivisions of the data into clusters and whose height represents the distances between points or clusters.

The algorithms for producing these clusterings are either *agglomerative*, in that they start with each observation in its own cluster, then combine nearest observations into clusters, nearest clusters into bigger clusters, etc.; or *divisive*, starting with the whole dataset, then splitting it into a small number of clusters, those clusters into smaller clusters, etc..

The former require defining a notion of a *distance between clusters*. The latter require to defining a criterion based on which a cluster is split. Some common examples of distances are provided in the following list:

- Single linkage $d(S_1, S_2) = \min\{d(\boldsymbol{x}_i, \boldsymbol{x}_j) : i \in S_1, j \in S_2\}$
- Complete linkage $d(S_1, S_2) = \max\{d(\boldsymbol{x}_i, \boldsymbol{x}_j) : i \in S_1, j \in S_2\}$
- Average linkage (unweighted) $d(S_1, S_2) = \frac{1}{|S_1||S_2|} \sum_{i \in S_1} \sum_{j \in S_2} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- Average linkage (weighted) $d(S_1 \cup S_2, S_3) = \frac{d(S_1, S_3) + d(S_2, S_3)}{2}$
- Centroid $d(S_1, S_2) = \|\bar{\boldsymbol{x}}_{S1} - \bar{\boldsymbol{x}}_{S_2}\|$
- Ward $d\left(S_1, S_2\right) = \sum_{i \in S_1 \cup S_2} \left|\boldsymbol{x}_i - \overline{\boldsymbol{x}}_{S_1 \cup S_2}\right|^2 - \sum_{i \in S_1} \left|\boldsymbol{x}_i - \overline{\boldsymbol{x}}_{S_1}\right|^2 - \sum_{i \in S_2} \left|\boldsymbol{x}_i - \overline{\boldsymbol{x}}_{S_2}\right|^2 = \frac{|S_1||S_2|}{|S_1|+|S_2|} \left|\overline{\boldsymbol{x}}_{S_1} - \overline{\boldsymbol{x}}_{S_2}\right|^2$

A framework that is useful for expressing different between-cluster distances is the *Lance–Williams* framework. Given three clusters, S_1, S_2, and S_3, and suppose that we have some metric for evaluating pairwise distances between them, i.e., $d(S_1, S_2)$, $d(S_1, S_3)$, and $d(S_2, S_3)$. Then, we define the distance resulting from combining S_1 and S_2 in terms of these pairwise distances and coefficients α_1, α_2, β, and γ:

$$d(S_1 \cup S_2, S_3) = \alpha_1 d(S_1, S_3) + \alpha_2 d(S_2, S_3) + \beta d(S_1, S_2) + \gamma |d(S_1, S_3) - d(S_2, S_3)|.$$

This, plus the distance metric between individual points (which applies when the clusters have only one observation in them), allows us to define and efficiently calculate distances between clusters.

For example, the unweighted average linkage can be expressed in this framework as follows:

$$
\begin{aligned}
d\left(S_1 \cup S_2, S_3\right) &= \frac{1}{\left|S_1 \cup S_2\right| \left|S_3\right|} \sum_{i \in S_1 \cup S_2} \sum_{j \in S_3} d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \\
&= \frac{1}{\left(\left|S_1\right| + \left|S_2\right|\right) \left|S_3\right|} \left(\sum_{i \in S_1} \sum_{j \in S_3} d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) + \sum_{i \in S_2} \sum_{j \in S_3} d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \right) \\
&= \frac{\left|S_1\right| \left|S_3\right| d\left(S_1, S_3\right) + \left|S_2\right| \left|S_3\right| d\left(S_2, S_3\right)}{\left(\left|S_1\right| + \left|S_2\right|\right) \left|S_3\right|}
\end{aligned}
$$

$$
\implies \alpha_1 = \frac{\left|S_1\right|}{\left|S_1\right| + \left|S_2\right|}, \alpha_2 = \frac{\left|S_2\right|}{\left|S_1\right| + \left|S_2\right|} \beta = \gamma = 0
$$

Ward's method—the most popular hierarchical clustering criterion— similarly, uses the squared Euclidean distances $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2$ between points and then

$$
\begin{aligned}
\alpha_1 &= \frac{\left|S_1\right| + \left|S_3\right|}{\left|S_1\right| + \left|S_2\right| + \left|S_3\right|}, \alpha_2 = \frac{\left|S_2\right| + \left|S_3\right|}{\left|S_1\right| + \left|S_2\right| + \left|S_3\right|} \\
\beta &= \frac{-\left|S_3\right|}{\left|S_1\right| + \left|S_2\right| + \left|S_3\right|}, \gamma = 0
\end{aligned}
$$

Ward's method joins the groups that will increase the within-group variance least.

Assessing

We now briefly discuss how a clustering $\boldsymbol{G}$ may be assessed. Ideally, this measurement should be "fair" to the number of clusters K. For example, in K-means clustering, splitting a cluster will *always* reduce the within-cluster variances, and so those cannot be used as a criterion.

A popular method, inspired by K-medioid clustering, is the *silhouettes*. For each $i = 1, \dots, n$, let

$$a(i) = \frac{1}{|S_{G_i}| - 1} \sum_{j \in S_{G_i}} d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$$

$$b(i) = \min_{g \neq G_i} \frac{1}{|S_g|} \sum_{j \in S_g} d\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$$

Observe that $a(i)$ is the distance between i and other observations its own cluster and $b(i)$ is the distance between i and observations in the cluster nearest to i to which i does not belong. In a good clustering each observation will be much closer to its own cluster than to its neighbouring cluster, so $b(i) \gg a(i)$.

Then, *silhouette of* i is a value between -1 and $+1$ calculated as follows:

$$s(i) = \begin{cases} \frac{b(i) - a(i)}{\max(a(i), b(i))} & \text{if } |S_{G_i}| > 1 \\ 0 & \text{otherwise} \end{cases}.$$

That is $s(i)$ evaluates how much closer is i to the rest of its cluster than it is to its nearest cluster, and a higher silhouette indicates a better clustering for point i. Mean silhouette $n^{-1} \sum_{i=1}^{n} s(i)$ then measures the overall quality of clustering.

Model-based clustering

Lastly, we turn to model-based clustering. We will discuss the theoretical underpinnings of this approach—mixture models—and an important special case of Gaussian clustering and its parametrisation. The Expectation–Maximisation algorithm, often used to estimate these models will also be described, as it is useful in a wide variety of circumstances, but it is not examinable.

Mixture Models

A *finite mixture model* is a probability model under which each observation comes from one of several distributions, but we do not observe from which one. (Infinite mixture models exist as well, but they are outside of the scope of this class).

A mixture model is specified as follows. We set K to be the number of distributions (clusters), and a collection of K density functions on the support of $\boldsymbol{x}_i$, $f_g(\boldsymbol{x}_i; \boldsymbol{\theta}_g)$ (for $g = 1, \ldots, K$) each having a parameter vectors $\boldsymbol{\theta}_g$ (e.g., its expectation), which we do not know and must estimate. We also postulate K (unknown) probabilities π_g that an observation (any observation) comes from cluster g. (Standard restrictions apply: $0 \leq \pi_g \leq 1, \sum_{g=1}^K \pi_g$.)

For brevity, we define $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)^\top$, a vector of these probabilities; and $\Psi = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\pi}\}$, the collection of all model parameters. Then, we assume the following data-generating process: for each $i = 1, \ldots, n$,

1. Sample $G_i \in 1, \ldots, K$ with $\Pr(G_i = g; \boldsymbol{\pi}) = \pi_g$.
2. Sample $\boldsymbol{X}_i | G_i \sim f_{G_i}(\cdot; \boldsymbol{\theta}_{G_i})$.
3. Observe $\boldsymbol{X}_i$, and "forget" G_i.

The pdf of this *mixture density* is

$$f_{\boldsymbol{X}_i}(\boldsymbol{x}_i; \Psi) = \sum_{g=1}^K \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\theta}_g). \tag{6.1}$$

We wish to estimate the parameters Ψ from the sample of $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$. This leads to the likelihood

$$L_{\boldsymbol{x}}(\Psi) = \prod_{i=1}^n \sum_{g=1}^K \pi_g f_g(\boldsymbol{x}_i; \boldsymbol{\theta}_g). \tag{6.2}$$

This formulation is convenient for a number of reasons. It is a probability model for the $\boldsymbol{X}_i$s, and

therefore we can use it to obtain a *soft clustering* rather than a *hard clustering* that assigns a point to a single cluster, we can apportion an observation's membership by how likely it to have come from each cluster. An application of Bayes's rule and (6.1) gives

$$\Pr(G_i = g|\boldsymbol{x}_i; \Psi) = \frac{\pi g f_g(\boldsymbol{x}_i; \boldsymbol{\theta}_g)}{\sum_{g'=1}^{K} \pi_{g'} f_{g'}(\boldsymbol{x}_i; \boldsymbol{\theta}_{g'})}.$$

We can also embed it into a *hierarchical model* (a meaning distinct from the hierarchical clustering above), in which either $\boldsymbol{x}_i$s are parameters for some model for the data or for the observation process or $\boldsymbol{\theta}$s are functions of some *hyper–parameters*. Lastly, the fact that we have a well-defined likelihood facilitates model selection.

Multivariate normal clusters

As with other analysis scenarios discussed in this course, the multivariate normal distribution provides a useful formulation for the clusters. Consider the following parametrisation:

$$f_g(\boldsymbol{x}_i; \boldsymbol{\theta}_g) = \frac{1}{(2\pi)^{p/2}|\Sigma(\boldsymbol{\theta}_g)|^{1/2}} \, \mathrm{e}^{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}(\boldsymbol{\theta}_g))^\top \{\Sigma(\boldsymbol{\theta}_g)\}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}(\boldsymbol{\theta}_g))} \, .$$

Here, $\boldsymbol{\mu}(\boldsymbol{\theta}_g)$ is the mean vector of cluster g (e.g., first p elements of $\boldsymbol{\theta}_g$), and $\Sigma(\boldsymbol{\theta}_g)$ is the model for the variances. We may also have different clusters "share" elements of $\boldsymbol{\theta}$, and a more general case is

$$f_g(\boldsymbol{x}_i; \boldsymbol{\theta}) = \frac{1}{(2\pi)^{p/2}|\Sigma_g(\boldsymbol{\theta})|^{1/2}} \, \mathrm{e}^{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_g(\boldsymbol{\theta}))^\top \{\Sigma_g(\boldsymbol{\theta})\}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_g(\boldsymbol{\theta}))}, \tag{6.3}$$

where $\boldsymbol{\mu}_g(\boldsymbol{\theta})$ and $\Sigma_g(\boldsymbol{\theta})$ "extract" the appropriate elements from $\boldsymbol{\theta}$.

One advantage of multivariate normal clusters is in its flexibility in specifying cluster size and shape. Recall the eigendecomposition of the covariance matrix $\Sigma = P\Lambda P^\top$, with P orthogonal and Λ diagonal and nonnegative. Let us further parametrise it as

$$\Sigma = \lambda P A P^\top,$$

with $P \in \mathcal{M}_{p,p}$ orthogonal, $A \in \mathcal{M}_{p,p}$ diagonal and nonnegative *with* $|A| = 1$ (*unimodular*), and scalar $\lambda > 0$. This allows us to interpret the structure of the matrix in simple, substantive terms.

Starting with λ, recall recalling that the determinant of a matrix can be viewed as its volume. Then,

$$|\Sigma| = \lambda^p |P||A||P^\top| = \lambda^p,$$

which makes λ is the "spread", "size", or "volume" of the cluster.

To interpret the diagonal, unimodular matrix A, observe that if $A = I_p$, then

$$\Sigma = \lambda P A P^\top = \lambda P P^\top = \lambda I_p,$$

making the cluster spherical—equal variances on all dimensions. Similarly, if some diagonal elements of A are much larger than others, then the cluster will be an ellipsoid more stretched in one direction than in others.

Lastly, observe that if $P = I_p$, then

$$\Sigma = \lambda P A P^\top = \lambda A,$$

an ellipsoid whose axes are parallel to coordinate axes, implying the elements of $\boldsymbol{X}_i$ within each cluster are uncorrelated with unequal variances. More generally, P controls the rotation of ellipsoid--- the correlation between the dimensions and the orientation of the cluster.

When it comes to estimating K clusters, we can permit the λs, the As, and the Ps to vary between the clusters, be constant between the clusters, or, for A and P, be fixed at the identity. Each combination embodies different assumption about the shape and the relationship between clusters; and, in general, the more we permit to vary, the more parameters we must estimate and the more data we therefore require. Generally,

1. For a mixture of K clusters, we must, invariably, estimate the cluster membership probabilities $\pi_1, \ldots, \pi_K$ ($K - 1$ parameters) and cluster means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ (Kp parameters).
2. Then, λ can be constrained $\lambda_1 = \lambda_2 = \cdots = \lambda_K$ (1 parameter) or allowed to vary (K parameters).
3. Then, A can be fixed $A_1 = A_2 = \cdots = A_K = I_d$ (0 parameters), constrained $A_1 = A_2 = \cdots = A_K$ ($p - 1$ parameters), or allowed to ($K(p - 1)$ parameters).
4. Lastly, if A is not fixed at the identity matrix, P can either be fixed $P_1 = P_2 = \cdots = P_K = I_d$ (0 parameters), constrained $P_1 = P_2 = \cdots = P_K$ ($\binom{p}{2}$ parameters), or allowed to vary ($K\binom{p}{2}$ parameters)

The different cluster shapes identified by their constraint tripple (λ, A, P) encoding being fixed at identity as I, being constrained to equality between clusters as E, and being allowed to vary freely as V are given in the following figure:

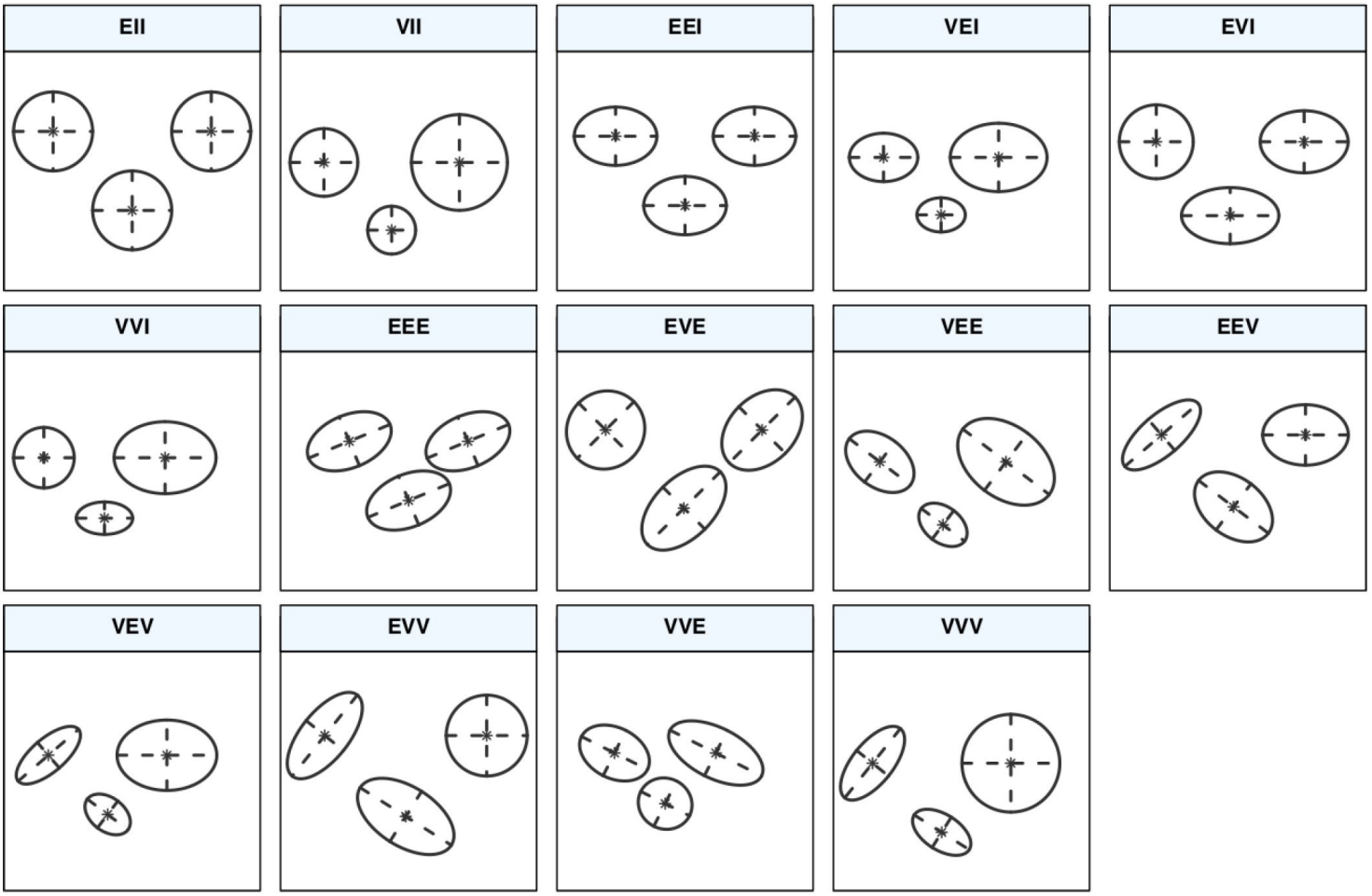


Figure 2 of: Luca Scrucca, Michael Fop, T. Brendan Murphy, and Adrian E. Raftery (2016). `mclust` 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R Journal* 8:1, pages 289-317.

Model selection

As mentioned before, model-based clustering requires one to specify both the number of clusters K and the within-cluster models $f_g(\boldsymbol{x}_i; \Psi)$. In the case of multivariate normal clustering, we have a large number of possible specifications for the Σ_gs, and the number of parameters can grow quickly for "XXV" models in particular.

At the same time, because it is likelihood-based, a variety of standard model-selection techniques can be used. For example, BIC is recommended:

$$\mathrm{BIC}_\nu = -2 \log L_{\boldsymbol{x}}(\hat{\Psi}) + \nu \log n,$$

where ν the number of parameters estimated. (Here, lower BIC is better, but some authors and software packages use $2 \log L_{\boldsymbol{x}}(\hat{\Psi}) - \nu \log n$, with higher BIC being better.)

Substantive considerations also matter. For example, how many clusters does our research hypothesis predict? Do we expect correlations between dimensions to vary between clusters?

Expectation–Maximisation Algorithm (optional)

In this final topic, we discuss the typical computational approach for estimating these mixture models. The $\log L(\Psi)$ in (6.2) is computationally tractable, but it does not simplify much, because while the logarithm of a product is a sum of the logarithms, the logarithm of a sum does not, in general, simplify further. Thus, we introduce the *Expectation–Maximisation (EM)* algorithm.

1. Introduce an unobserved (latent) variable G_i, $i = 1, \ldots, n$ giving the cluster membership of i.
2. Suppose that $G_1, \ldots, G_n$ *are* observed; then, this *complete-data likelihood*

$$L_{\boldsymbol{x},G_1,\ldots,G_n}(\Psi) = \prod_{i=1}^{n} \pi_{G_i} f_{G_i}(\boldsymbol{x}_i; \boldsymbol{\theta}_{G_i}) :$$

   we "know" the exact cluster from which each observation came, so we no longer have to sum over the possible clusters. Then, the log-likelihood decomposes into two summations

$$\log L_{\boldsymbol{x},G_1,\ldots,G_n}(\Psi) = \sum_{i=1}^{n} \log \pi G_i + \sum_{i=1}^{n} \log f_{G_i}(\boldsymbol{x}_i; \boldsymbol{\theta} G_i), \tag{6.4}$$

   one that depends only on the π_gs and the other only on the $\boldsymbol{\theta}_g$s.
3. Start with an initial guess $\Psi^{(0)}$.
4. Iterate **E-step** and **M-step** described below to convergence.

E-step

The *Expectation step* consists of starting with a parameter guess $\Psi^{(t-1)}$ and evaluating

$$Q(\Psi|\Psi^{(t-1)}) = \mathrm{E}_{G_1,\ldots,G_n|\boldsymbol{x};\Psi^{(t-1)}}(\log L_{\boldsymbol{x},G_1,\ldots,G_n}(\Psi)) :$$

the expected value of the complete-data log-likelihood. We can evaluate it by calculating (using the Bayes's rule)

$$q_{ig}^{(t-1)} = \Pr(G_i = g|\boldsymbol{x}; \Psi^{(t-1)}) = \frac{\pi_g^{(t-1)} f_g(\boldsymbol{x}_i; \boldsymbol{\theta} g^{(t-1)})}{\sum_{g'=1}^{K} \pi_{g'}^{(t-1)} f_{g'}(\boldsymbol{x}_i; \boldsymbol{\theta}_{g'}^{(t-1)})}$$
$$i = 1, \ldots, n,\ g = 1, \ldots, K,$$

then substituting them in as

$$Q(\Psi|\Psi^{(t-1)}) = \sum_{i=1}^{n}\sum_{g=1}^{K} q_{ig}^{(t-1)} \log \pi_g + \sum_{i=1}^{n}\sum_{g=1}^{K} q_{ig}^{(t-1)} \log f_g(\boldsymbol{x}_i;\boldsymbol{\theta}_g). \tag{6.5}$$

Observe that, like (6.4), (6.5) decomposes into a summation that depends only on the π_gs and a summation that depends only on the $\boldsymbol{\theta}_g$s.

M-step

The *Maximisation step* then consists of maximising the $Q(\Psi|\Psi^{(t-1)})$ with respect to Ψ to obtain the next parameter guess:

$$\Psi^{(t)} = \arg\max_{\Psi} Q(\Psi|\Psi^{(t-1)}), \text{ s.t. } \sum_{g=1}^{K} \pi_g = 1.$$

Conveniently, the form (6.5) separates the π_gs from the $\boldsymbol{\theta}_g$s, and so we can maximise them separately (i.e., if we differentiate with respect to one, the summation involving the other will vanish).

Maximising (6.5) with respect to $\boldsymbol{\theta}_g$s, we take the derivative

$$\frac{\partial Q(\Psi|\Psi^{(t-1)})}{\partial \boldsymbol{\theta}_g} = \sum_{i=1}^{n} q_{ig}^{(t-1)} \frac{\partial \log f_g(\boldsymbol{x}_i;\boldsymbol{\theta}_g)}{\partial \boldsymbol{\theta}_g},$$

and set to 0. This is a *weighted* maximum likelihood estimator.

Maximising (6.5) with respect to to π_gs is also straightforward. We will use Lagrange Multipliers to do so:

$$\text{Lag}(\boldsymbol{\pi}) = \sum_{i=1}^{n}\sum_{g=1}^{K} q_{ig}^{(t-1)} \log \pi_g - \alpha\left(\sum_{g=1}^{K} \pi_g - 1\right).$$

Differentiating,

$$\text{Lag}'_g(\boldsymbol{\pi}) = \sum_{i=1}^{n} q_{ig}^{(t-1)} \pi_g^{-1} - \alpha.$$

Setting to 0,

$$\pi_g = \sum_{i=1}^{n} q_{ig}^{(t-1)} / \alpha.$$

Summing and solving for α,

$$\sum_{g=1}^{K} \pi_g = \frac{1}{\alpha} \sum_{g=1}^{K} \sum_{i=1}^{n} q_{ig}^{(t-1)} = 1,$$

$$\alpha = \sum_{g=1}^{K} \sum_{i=1}^{n} q_{ig}^{(t-1)}.$$

Therefore,

$$\pi_g^{(t)} = \frac{\sum_{i=1}^{n} q_{ig}^{(t-1)}}{\sum_{g=1}^{K} \sum_{i=1}^{n} q_{ig}^{(t-1)}}.$$

"Sharing" $\boldsymbol{\theta}$s

Lastly, recall that when we select one of the "$\mathbf{E}$" models and (6.3) in the 'Model-based clustering' section, we no longer have a separate $\boldsymbol{\theta}_g$ for every f_g. We may then need to redefine $\boldsymbol{\theta} \in \mathbb{R}^{Kp+1 \text{ or more}}$ to contain parameters for all groups (separate means, distinct variance parameters, etc.), and $f_g(\boldsymbol{x}_i; \boldsymbol{\theta})$ to "extract" those elements of $\boldsymbol{\theta}$ that it needs, with $\Psi = (\boldsymbol{\theta}, \boldsymbol{\pi})$.

Inferentially, $\boldsymbol{\theta}$ replaces $\boldsymbol{\theta}_g$ in all derivations above. In particular,

$$Q(\Psi|\Psi^{(t-1)}) \sum_{i=1}^{n} \sum_{g=1}^{K} q_{ig}^{(t-1)} \log \pi_g + \sum_{i=1}^{n} \sum_{g=1}^{K} q_{ig}^{(t-1)} \log f_g(\boldsymbol{x}_i; \boldsymbol{\theta}),$$

so

$$\frac{\partial Q(\Psi|\Psi^{(t-1)})}{\partial \boldsymbol{\theta}} = \sum_{i=1}^{n} \sum_{g=1}^{K} q_{ig}^{(t-1)} \frac{\partial \log f_g(\boldsymbol{x}_i; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}},$$

which is still a weighted MLE, but now it is joint for all groups, and without simplification.

Demonstration: Cluster analysis

Please begin by watching the following demonstration by Dr Krivitsky.

Transcript

This demonstration can be completed using the provided RStudio or your own RStudio.

To complete this task select the 'Cluster_Examples.demo.Rmd' in the 'Files' section of RStudio. Follow the demonstration contained within the RMD file.

If you choose to complete the example in your own RStudio, upload the following file:

Cluster_Examples.demo.Rmd

The output of the RMD file is also displayed below:

Challenge: Cluster analysis

If you choose to complete this task in your own RStudio, upload the following file:

Cluster_Examples.challenge.Rmd

Click on the 'Cluster_Examples.challenge.Rmd' in the 'Files' section to begin. Enter your response to the tasks in the 'Enter your code here' section.

This activity and the solution will be discussed at the Collaborate session this week. In the meantime, share and discuss your results in the 'Tutorials' discussion forum.

The solution will also be available here on Friday of this week by clicking on the 'Solution' tab in the top right corner.

The output of the RMD file is also displayed below: