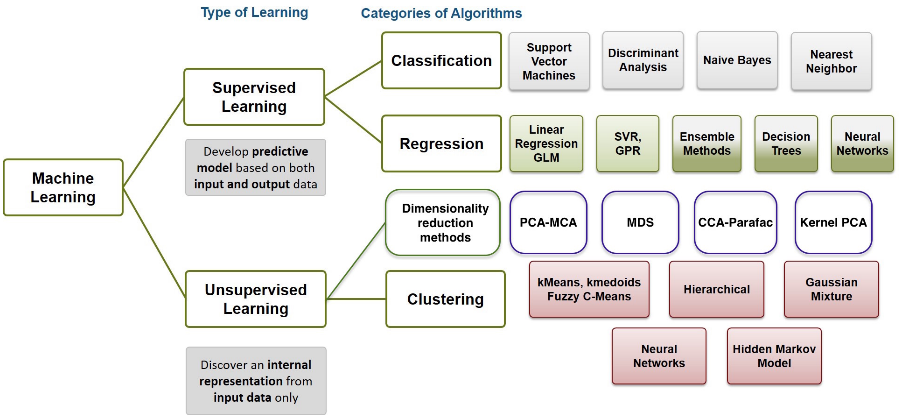
Unsupervised Learning
Introduction to clustering

Polytechnique MAP 573, 2019 – Julien Chiquet

Autumn semester, 2019

`https://github.com/jchiquet/CourseUnsupervisedLearningX`

Machine Learning



Supervised vs Unsupervised Learning

Supervised Learning

- Training data $\mathcal{D}_n = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}, X_i \sim^{\text{i.i.d}} \mathbb{P}$
- Construct a predictor $\hat{f} : \mathcal{X} \to \mathcal{Y}$ using $\mathcal{D}_n$
- Loss $\ell(y, f(x))$ measures how well $f(x)$ predicts y
- Aim: minimize the generalization error
- Task: Regression, Classification

$\rightsquigarrow$ The goal is clear: predict y based on x (regression, classification)

Unsupervised Learning

- Training data $\mathcal{D} = \{x_1, \dots, x_n\}$
- Loss? , Aim?
- Task: Dimension reduction, Clustering

$\rightsquigarrow$ The goal is less well defined, and *validation* is questionable

Outline

Outline

Outline

Companion data set

Morphological Measurements on Leptograpsus Crabs

Description

The crabs data frame has 200 rows and 8 columns, describing 5 morphological measurements on 50 crabs each of two colour forms and both sexes, of the species *Leptograpsus variegatus* collected at Fremantle, W. Australia.

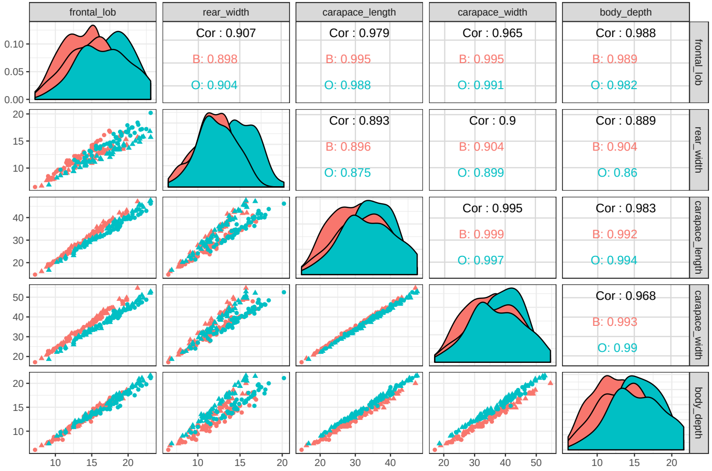
```
crabs <- MASS::crabs %>% select(-index) %>%
  rename(sex = sex,
         species        = sp,
         frontal_lob    = FL,
         rear_width     = RW,
         carapace_length = CL,
         carapace_width  = CW,
         body_depth     = BD)
crabs %>% select(sex, species) %>% summary() %>% knitr::kable("latex")
```

	sex	species
	F:100	B:100
	M:100	O:100

Companion data set II

Pairs plot of attributes

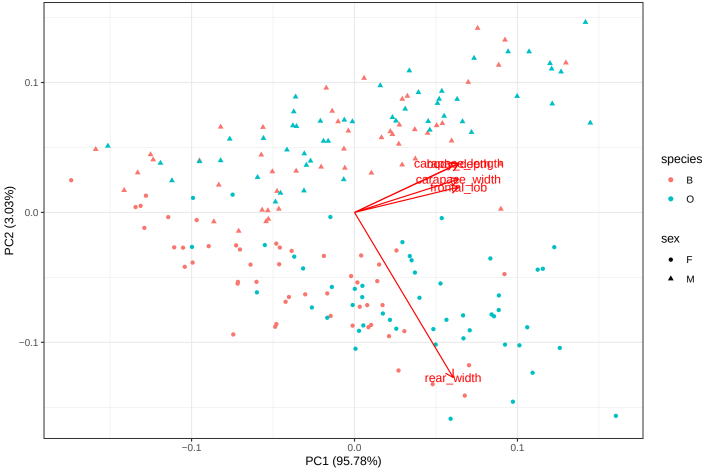
```r
ggpairs(crabs, columns = 3:7, aes(colour = species, shape = sex))
```



Companion data set III

PCA on the attributes

```r
prcomp(select(crabs, -species, -sex), scale. = TRUE) %>%
  autoplot(loadings = TRUE, loadings.label = TRUE,
           data = crabs, colour = 'species', shape = 'sex')
```



Remove size effect I

Carried by 1st principal components

PCA is solved by SVD

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top.$$

We remove the best rank-1 approximation of $\mathbf{X}$ to remove the *size effect*, carried by the first axis, that is,

$$\tilde{\mathbf{X}}^{(1)} = \mathbf{U}_{\bullet 1} d_{11} \mathbf{v}_{\bullet 1}^\top.$$

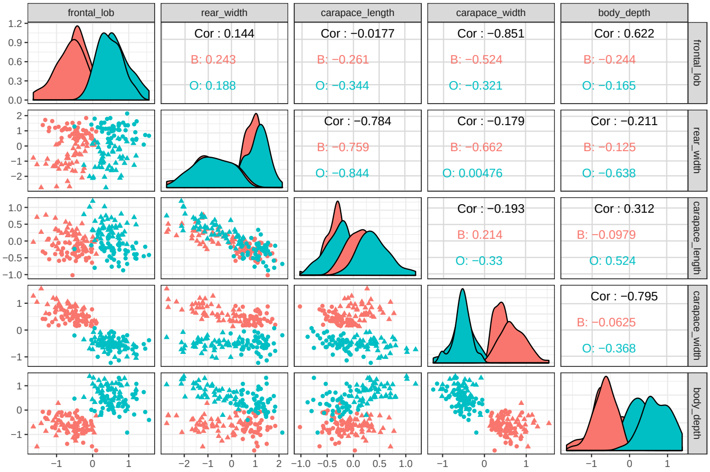
```
attributes <- select(crabs, -sex, -species)
SVD <- svd(attributes)
attributes_rank1 <- tcrossprod(SVD$u[, 1] * SVD$d[1], SVD$v[, 1])
crabs_corrected <- crabs
crabs_corrected[, 3:7] <- attributes - attributes_rank1
```

$\rightsquigarrow$ Axis 1 explains a latent effect, here the size in the case at hand, common to all attributes.

Remove size effect II

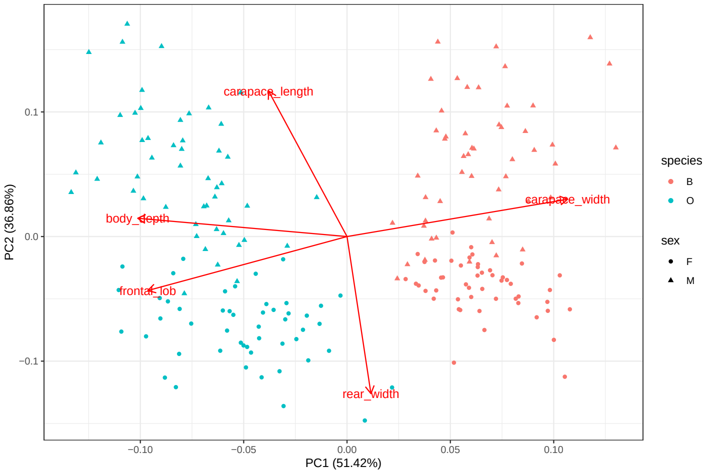
Carried by 1st principal components

```
ggpairs(crabs_corrected, columns = 3:7, aes(colour = species, shape = sex))
```



PCA on corrected data

```r
prcomp(select(crabs_corrected, -species, -sex), scale. = TRUE) %>%
  autoplot(loadings = TRUE, loadings.label = TRUE,
           data = crabs_corrected, colour = 'species', shape = 'sex')
```



Questions

1. Could we automatically identify some grouping (clustering) between samples ?
2. Would this clustering correspond to some known labels (sex, species)?
3. Does it matter?

Outline

Clustering: general goals

Objective: construct a map f from $\mathcal{D}$ to $\{1, \ldots, K\}$ where K is a fixed number of clusters.

Careful! classification $\neq$ clustering

- Classification presupposes the existence of classes
- Clustering labels only elements of the dataset
  - $\rightsquigarrow$ no ground truth (no given labels)
  - $\rightsquigarrow$ discover a structure "natural" to the data

Motivations

- describe large masses of data in a simplified way,
- structure a set of knowledge,
- reveal structures, hidden causes,
- use of the groups in further processing, . . .

Clustering: challenges

Clustering quality

No obvious measure to define the quality of the clusters.

- Inner homogeneity: samples in the same group should be similar
- Outer inhomogeneity: samples in different groups should be different

Number of clusters

Choice of the number of cluster K often complex

- No ground truth in unsupervised learning!
- Several solution might be equally good

Two general approaches

- distance-based: require a distance/disimilarity between $\{\mathbf{x}_i\}$
- model-based: require assumptions on the distribution $\mathbb{P}$

Outline

Dissimilarity and Distance

Clustering requires a measure of ressemblance between object

Definition ((dis)similarity)

Similarity (*resp. Dissimilarity*) measures the ressemblance (*resp. discrepancy*) between objects based on several features.

For instance, two objects are similar if

- they share a certain feature
- their features are close according measure of proximity

Definition (distance/metric)

Dissimilairty can be measuresd by distances, *i.e.* a function $d_{ii'}$ between pairs in $\{\mathbf{x}_i\}$ s.t.

- $d_{ij} \geq 0,$
- $d_{ij} = 0 \Leftrightarrow \mathbf{x}_i = \mathbf{x}_j,$
- $d_{ij} = d_{ji},$
- $d_{ik} \leq d_{ij} + d_{jk}.$

Dissimilarity and Distance

Clustering requires a measure of ressemblance between object

Definition ((dis)similarity)

Similarity (*resp. Dissimilarity*) measures the ressemblance (*resp. discrepancy*) between objects based on several features.

For instance, two objects are similar if

- they share a certain feature
- their features are close according measure of proximity

Definition (distance/metric)

Dissimilairty can be measuresd by distances, *i.e.* a function $d_{ii'}$ between pairs in $\{\mathbf{x}_i\}$ s.t.

- $d_{ij} \geq 0$,
- $d_{ij} = 0 \Leftrightarrow \mathbf{x}_i = \mathbf{x}_j$,
- $d_{ij} = d_{ji}$,
- $d_{ik} \leq d_{ij} + d_{jk}$.

Classification structures: Partition

Clustering leads to a grouping (or classification) of individuals into homogeneous classes

We consider two structures to describe this classification: partitions and hierarchies.

Definition (Partition)

A partition $\mathcal{P}$ is a decomposition $\mathcal{P} = \{P_1, \ldots, P_K\}$ of a finite ensemble Ω such that

- $P_k \cap P_{k'} = \emptyset$ for any $k \neq k'$
- $\bigcup_k P_k = \Omega$

In a set $\Omega = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ partitioned into K classes, each element of the set belongs to a class and only one.

Classification structures: Hierarchy

Definition (Hierarchy)

A hierarchy $\mathcal{H}$ is a is a non empty subsetof a finite ensemble Ω such that

- $\Omega \in \mathcal{H}$,
- $\forall \mathbf{x} \in \Omega, \{\mathbf{x}\} \in \mathcal{H}$,
- $\forall H, H' \in \mathcal{H}$, then either $H \cap H' = \emptyset$, $H \subset H'$ or $H' \subset H$.

Definition (Index of a Hierarchy)

The index is a function $i : \mathcal{H} \to \mathbb{R}_+$ such that

- if $H \subset H'$ then $i(H) < i(H')$;
- if $\mathbf{x} \in \Omega$ then $i(\mathbf{x}) = 0$.

Properties (Partition and Hierarchy)

- *Each level of an indexed hierarchy is a partition;*
- $\{\Omega, P_1, \ldots, P_K, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$ *is a hierachy.*

Classification structures: Hierarchy

Definition (Hierarchy)

A hierarchy $\mathcal{H}$ is a is a non empty subsetof a finite ensemble Ω such that

- $\Omega \in \mathcal{H}$,
- $\forall \mathbf{x} \in \Omega, \{\mathbf{x}\} \in \mathcal{H}$,
- $\forall H, H' \in \mathcal{H}$, then either $H \cap H' = \emptyset$, $H \subset H'$ or $H' \subset H$.

Definition (Index of a Hierarchy)

The index is a function $i : \mathcal{H} \to \mathbb{R}_+$ such that

- if $H \subset H'$ then $i(H) < i(H')$;
- if $\mathbf{x} \in \Omega$ then $i(\mathbf{x}) = 0$.

Properties (Partition and Hierarchy)

- *Each level of an indexed hierarchy is a partition;*
- $\{\Omega, P_1, \dots, P_K, \mathbf{x}_1, \dots, \mathbf{x}_n\}$ *is a hierachy.*

Classification structures: Hierarchy

Definition (Hierarchy)

A hierarchy $\mathcal{H}$ is a is a non empty subsetof a finite ensemble Ω such that

- $\Omega \in \mathcal{H}$,
- $\forall \mathbf{x} \in \Omega, \{\mathbf{x}\} \in \mathcal{H}$,
- $\forall H, H' \in \mathcal{H}$, then either $H \cap H' = \emptyset$, $H \subset H'$ or $H' \subset H$.

Definition (Index of a Hierarchy)

The index is a function $i : \mathcal{H} \to \mathbb{R}_+$ such that

- if $H \subset H'$ then $i(H) < i(H')$;
- if $\mathbf{x} \in \Omega$ then $i(\mathbf{x}) = 0$.

Properties (Partition and Hierarchy)

- *Each level of an indexed hierarchy is a partition;*
- $\{\Omega, P_1, \ldots, P_K, \mathbf{x}_1, \ldots, \mathbf{x}_n\}$ *is a hierachy.*

Clusterings Comparison: Contingency table

Definition

Consider two clustering U and V of Ω elements, in respectively $|U|$ and $|V|$ classes. The $|U| \times |V|$ contingency matrix stores at position i, j the number of elements that are simultaneously in cluster i of U and in cluster j of V.

$\mathbf{U \backslash V}$	V_1	V_2	$\dots$	$V_{\lvert V\rvert}$	Sums
U_1	n_{11}	n_{12}	$\dots$	$n_{1\lvert V\rvert}$	$n_{1.}$
U_2	n_{21}	n_{22}	$\dots$	$n_{2\lvert V\rvert}$	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$U_{\lvert U\rvert}$	$n_{\lvert U\rvert 1}$	$n_{\lvert U\rvert 2}$	$\dots$	$n_{\lvert U\rvert\lvert V\rvert}$	$n_{\lvert U\rvert .}$
Sums	$n_{.1}$	$n_{.2}$	$\dots$	$n_{.\lvert V\rvert}$	$n_{..} = n$

Clusterings Comparison: Measures

The ARI (most popular) is a measure of accuracy between two clustering, adjusted for chance grouping of element.

Definition (Adjusted Rand-index)

$$ARI(U,V) = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \left[\sum_i \binom{n_{i.}}{2} \sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}$$

Other measures include [**?**]

- NVI, the normalized variation information
- NID, the normalized information distance
- NMI, the normalized mutual information

Outline

Outline

K-means heuristic

Idea

1. Clustering is defined by a partition in K classes
2. Minimize a criteria of clustering quality
3. Use Euclidean distances to measure dissimilarity

Criteria: intra-class variance/inertia

Intra-class variance measures inner homogeneity

$$I_w = \sum_{k=1}^{K} \sum_{i=1}^{n} c_{ik} \left\| \mathbf{x}_i - \boldsymbol{\mu}_k \right\|_2^2 ,$$

where

- $\boldsymbol{\mu}_k$ are the centers (prototypes) of classes
- $c_{ik} = \mathbf{1}_{i \in \mathcal{P}_k}$ is a partition matrix

K-means algorithm

Ideally, one would solve

$$(\hat{\mathbf{c}}, \hat{\boldsymbol{\mu}}) = \arg\min_{(\mathbf{c},\boldsymbol{\mu})} I_w((\mathbf{c}, \boldsymbol{\mu})), \qquad \text{s.t } \mathbf{c} \text{ is a partion matrix.}$$

This problem is hard to solve but can be optimized locally as follows:

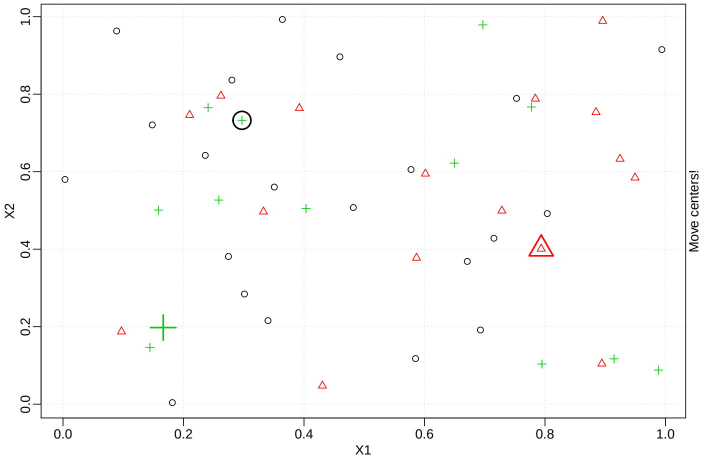
K-means algorithm (Loyds)

Initialization start by a (pseudo) random choice for the centers $\boldsymbol{\mu}_k$

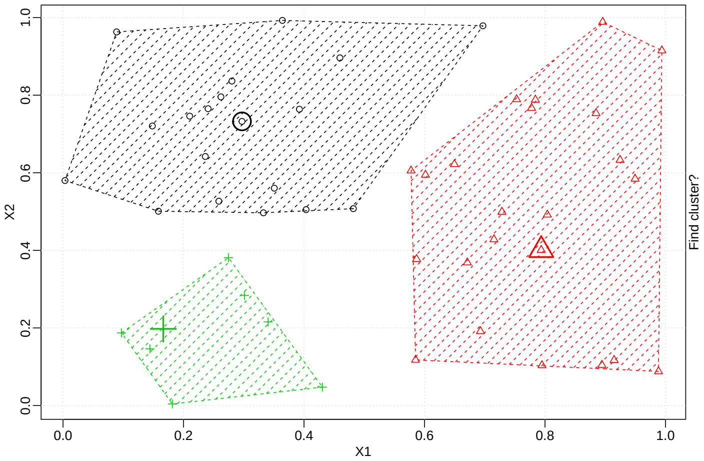
Alternate until convergence

- step 1 given $\boldsymbol{\mu}$, chose $\mathbf{c}$ minimizing $I_w \equiv$ assign $\mathbf{x}_i$ to nearest prototype
- step 2 given $\mathbf{c}$, chose $\boldsymbol{\mu}$ minimizing $I_w \equiv$ update $\boldsymbol{\mu}$ by the new means of classes

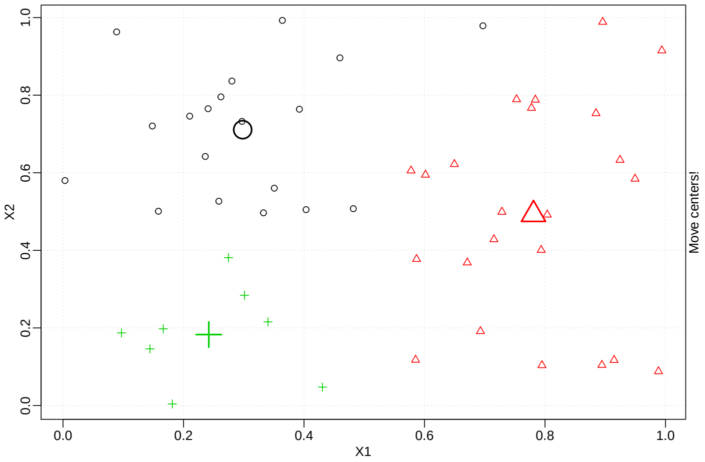
K-means in action I



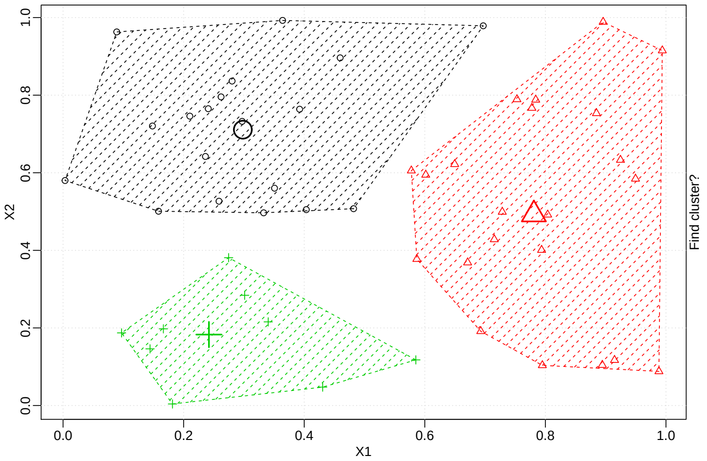
K-means in action II



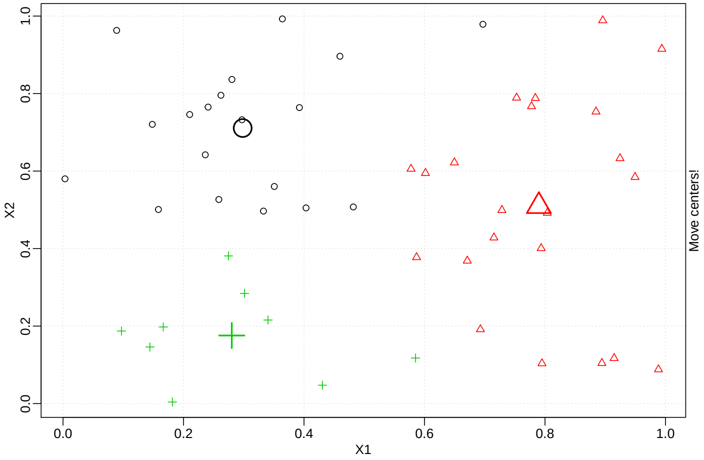
K-means in action III



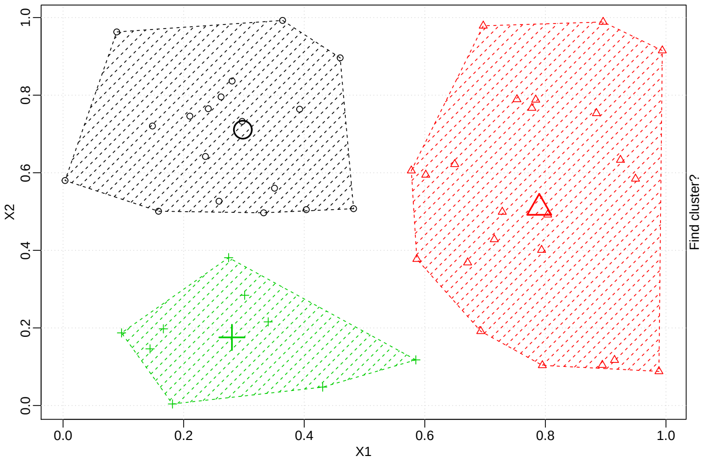
K-means in action IV



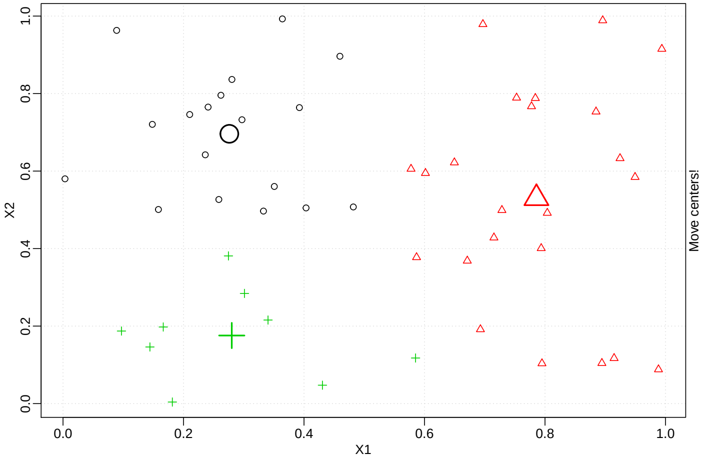
K-means in action V



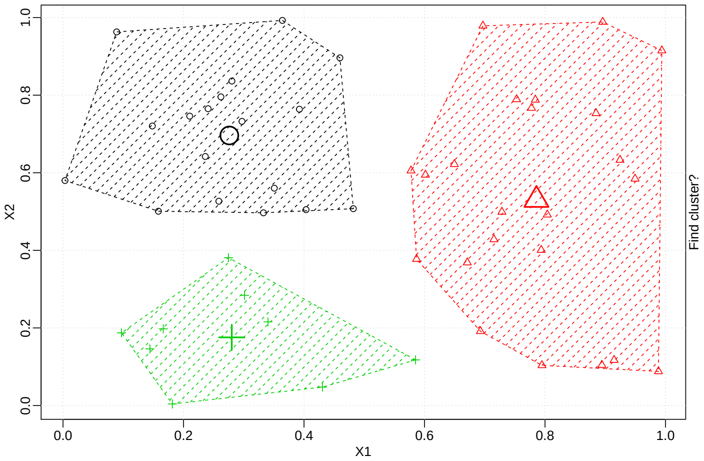
K-means in action VI



K-means in action VII



K-means in action VIII



K-means: properties

Other schemes

- McQueen: modify the mean each time a sample is assigned to a new cluster.
- Hartigan: modify the mean by removing the considered sample, assign it to the nearby center and recompute the new mean after assignment.

Initialization

No guarantee to converge to a global optimum

- Repeat and keep the best result
- k-Mean++: try to take them as separated as possible.

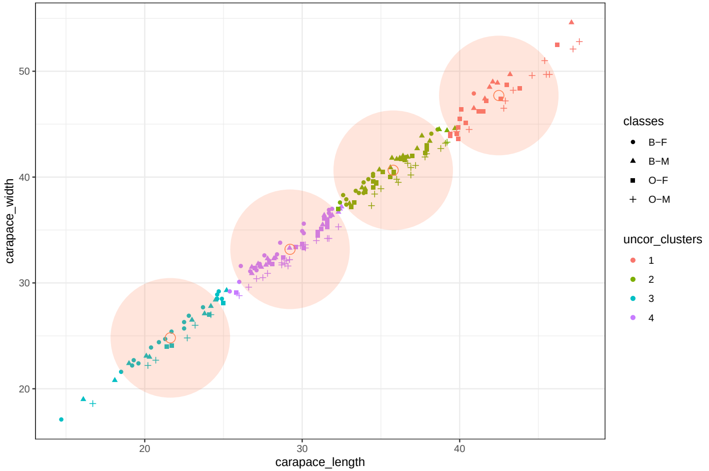
Complexity

$O(n \times K \times T)$ where T is the number of step in the algorithm.

K-means in R on uncorrected data set I

```r
uncor_kmeans_res <- crabs %>%
  select(-species, -sex) %>%
  kmeans(4, nstart = 10)
uncor_clusters <- as.factor(uncor_kmeans_res$cluster)
uncor_centers  <- as.tibble(uncor_kmeans_res$centers)
classes <- paste(crabs_corrected$species, crabs_corrected$sex, sep = "-")

crabs %>%
  ggplot(aes(x = carapace_length, y = carapace_width, color = uncor_clusters)) +
  geom_point(aes(shape = classes)) +
  geom_point(data = uncor_centers, color = 'coral', size = 4 , pch = 21) +
  geom_point(data = uncor_centers, color = 'coral', size = 50, alpha = 0.2)
```

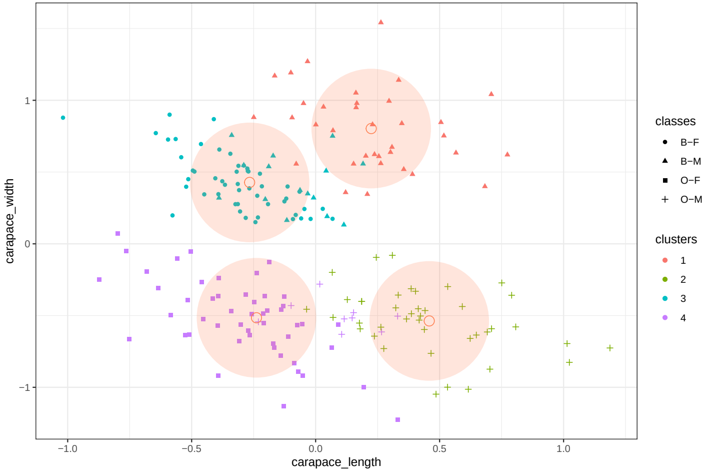
K-means in R on uncorrected data set II



K-means in R on corrected crabs data set I

```r
kmeans_res <- crabs_corrected %>%
  select(-species, -sex) %>%
  kmeans(4, nstart = 10)
clusters <- as.factor(kmeans_res$cluster)
centers  <- as.tibble(kmeans_res$centers)
classes <- paste(crabs_corrected$species, crabs_corrected$sex, sep = "-")

crabs_corrected %>%
  ggplot(aes(x = carapace_length, y = carapace_width, color = clusters)) +
  geom_point(aes(shape = classes)) +
  geom_point(data = centers, color = 'coral', size = 4 , pch = 21) +
  geom_point(data = centers, color = 'coral', size = 50, alpha = 0.2)
```

K-means in R on corrected crabs data set II



Clustering comparison

```
aricode::ARI(clusters, classes)

## [1] 0.7223637

aricode::ARI(uncor_clusters, classes)

## [1] 0.01573617

table(clusters, classes)

##         classes
## clusters B-F B-M O-F O-M
##        1   0  35   0   0
##        2   0   0   0  41
##        3  50  15   0   0
##        4   0   0  50   9
```

Outline

Agglomerative Clustering I

Agglomerative Clustering Heuristic

- Start with very small clusters (a sample by cluster?)
- Sequential merging of the most similar clusters...
- according to some *greedy* criterion Δ.

- Generates a hierarchy of clustering instead of a single one.
- Need to select the number of cluster afterwards.
- Several choice for the merging criterion...
- Examples:
  - Minimum Linkage: merge the closest cluster in term of the usual distance
  - Ward's criterion: merge the two clusters yielding the less inner inertia loss (k-means criterion)

Agglomerative Clustering II

Algorithm

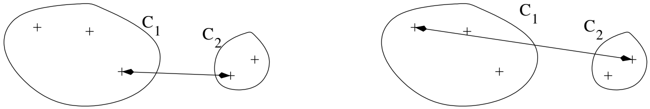
- Start with $(\mathcal{C}_i^{(0)}) = (\{\mathbf{x}_i\})$ the collection of all singletons.
- At step s, we have $n - s$ clusters $(\mathcal{C}_i^{(s)})$:
    - Find the two clusters the most similar according to a criterion Δ:

$$(i, i') = \underset{(j,j')}{\arg\min}\, \Delta(\mathcal{C}_j^{(s)}, \mathcal{C}_{j'}^{(s)})$$

    - Merge $\mathcal{C}_i^{(s)}$ and $\mathcal{C}_{i'}^{(s)}$ into $\mathcal{C}_i^{(s+1)}$
    - Keep the $n - s - 2$ other clusters $\mathcal{C}_{i''}^{(s+1)} = \mathcal{C}_{i''}^{(s)}$
- Repeat until there is only one cluster.

- Complexity: $O(n^3)$ if no restriction on the merging possibilities.
- Can be reduced to $O(n^2)$ if only a bounded number of merging is possible for a given cluster.

Agglomerative Clustering III



Merging criterion based on the distance between points

Agglomerative Clustering IV

- Minimum linkage:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \min_{\mathbf{x}_i \in \mathcal{C}_i} \min_{\mathbf{x} \in \mathcal{C}_j} d(\mathbf{x}_i, \mathbf{x}_j)$$

- Maximum linkage:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \max_{\mathbf{x}_i \in \mathcal{C}_i} \max_{\mathbf{x} \in \mathcal{C}_j} d(\mathbf{x}_i, \mathbf{x}_j)$$

- Average linkage:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_i||\mathcal{C}_j|} \sum_{\mathbf{x}_i \in \mathcal{C}_i} \sum_{\mathbf{x} \in \mathcal{C}_j} d(\mathbf{x}_i, \mathbf{x}_j)$$

Agglomerative Clustering V

Merging criterion based on the inertia (distance to the mean)

- Ward's criterion:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \sum_{\mathbf{x}_i \in \mathcal{C}_i} \left(d^2(\mathbf{x}_i, \mu_{\mathcal{C}_i \cup \mathcal{C}_j}) - d^2(\mathbf{x}_i, \mu_{\mathcal{C}_i}) \right) + \sum_{\mathbf{x}_j \in \mathcal{C}_j} \left(d^2(\mathbf{x}_j, \mu_{\mathcal{C}_i \cup \mathcal{C}_j}) - d^2(\mathbf{x}_j, \mu_{\mathcal{C}_j}) \right)$$

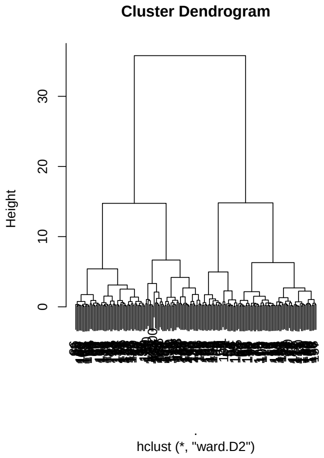
- If d is the euclidean distance:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \frac{2|\mathcal{C}_i||\mathcal{C}_j|}{|\mathcal{C}_i| + |\mathcal{C}_j|} d^2(\mu_{\mathcal{C}_i}, \mu_{\mathcal{C}_j})$$

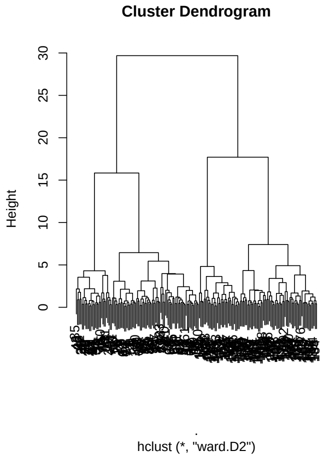
- Same criterion than in the k-means algorithm but greedy optimization.

Agglomerative Clustering VI

```r
Ward <- crabs %>%
  select(-sex, -species) %>%
  scale() %>%
  dist(method = "euclidean") %>%
  hclust(method = "ward.D2")
Ward_corrected <- crabs_corrected %>%
  select(-sex, -species) %>%
  scale() %>%
  dist(method = "euclidean") %>%
  hclust(method = "ward.D2")
par(mfrow=c(1,2))
plot(Ward)
plot(Ward_corrected)
```



Cluster Dendrogram
Height
0
10
20
30
hclust (*, "ward.D2")



Cluster Dendrogram
Height
0
5
10
15
20
25
30
hclust (*, "ward.D2")

```r
ARI_species <- Ward %>%
  cutree(k = 1:10) %>%
  as.data.frame() %>% as.list() %>%
  sapply(aricode::ARI, paste(crabs$species,crabs$sex, sep="-"))
ARI_species_corr <- Ward_corrected %>%
  cutree(k = 1:10) %>%
  as.data.frame() %>% as.list() %>%
  sapply(aricode::ARI, paste(crabs$species,crabs$sex, sep="-"))
```

```r
ARI_species <- Ward %>%
  cutree(k = 1:10) %>%
  as.data.frame() %>% as.list() %>%
  sapply(aricode::ARI, paste(crabs$species,crabs$sex, sep = "-"))
```

Outline

References

Pattern recognition and machine learning,
Christopher Bishop
Chapter 9: Mixture Models and EM
http://users.isr.ist.utl.pt/~wurmd/Livros/school/

Models with Hidden Structure with Applications in Biology and Genomics,
Stéphane Robin
Master MathSV Course
https://www6.inra.fr/mia-paris/content/download/4587/42934/version/1/file/ModelsHiddenStruct-Biology.pdf

Classification non-supervisées,
É. Lebarbier, T. Mary-Huard
Chapitre 3 - méthode probabiliste: le modèle de mélange
https://www.agroparistech.fr/IMG/pdf/ClassificationNonSupervisee-AgroParisTech.pdf

Outline

Latent variables models

Definition

A latent variable model is a statistical model that relates, for $i = 1, \dots, n$ individuals,

- a set of manifest (observed) variables $\mathbf{X} = (X_i, i = 1, \dots, n)$ to
- a set of latent (unobserved) variables $\mathbf{Z} = (Z_i, i = 1, \dots, n)$.

Common assumption: conditional independence

$$\mathbb{P}((X_1, \dots, X_n)|(Z_1, \dots, Z_n)) = \prod_{i=1}^{n} \mathbb{P}(X_i|Z_i).$$

Famous examples

- $(Z_i, i \geq 1)$ is Markov chain: Markov models
- Z_i categorical and independent: mixture models

what if $X_i = X_{ij}$ is a collection of edges in a graph?

Latent variables models

Definition

A latent variable model is a statistical model that relates, for $i = 1, \dots, n$ individuals,

- a set of manifest (observed) variables $\mathbf{X} = (X_i, i = 1, \dots, n)$ to
- a set of latent (unobserved) variables $\mathbf{Z} = (Z_i, i = 1, \dots, n)$.

Common assumption: conditional independence

$$\mathbb{P}((X_1, \dots, X_n)|(Z_1, \dots, Z_n)) = \prod_{i=1}^{n} \mathbb{P}(X_i|Z_i).$$

Famous examples

- $(Z_i, i \geq 1)$ is Markov chain: Markov models
- Z_i categorical and independent: mixture models
- what if $X_i = X_{i'j'}$ is a collection of edges in a graph?

Latent variables models

Definition

A latent variable model is a statistical model that relates, for $i = 1, \dots, n$ individuals,

- a set of manifest (observed) variables $\mathbf{X} = (X_i, i = 1, \dots, n)$ to
- a set of latent (unobserved) variables $\mathbf{Z} = (Z_i, i = 1, \dots, n)$.

Common assumption: conditional independence

$$\mathbb{P}((X_1, \dots, X_n)|(Z_1, \dots, Z_n)) = \prod_{i=1}^{n} \mathbb{P}(X_i|Z_i).$$

Famous examples

- $(Z_i, i \geq 1)$ is Markov chain: Markov models
- Z_i categorical and independent: mixture models
- what if $X_i = X_{i'j'}$ is a collection of edges in a graph?

Mixture models: the latent variables

When $(Z_1, \dots, Z_n)$ are independent categorical variables, they give a natural (latent) classification of the observations $(X_1, \dots, X_n)$ – or labels.

Notations

Let $(Z_1, \dots, Z_n)$ be *iid* categorical variables with distribution

$$\mathbb{P}(i \in q) = \mathbb{P}(Z_i = q) = \alpha_q, \quad \text{s.t.} \sum_{q=1}^{Q} \alpha_q = 1.$$

Alternative (equivalent) notation

Let $Z_i = (Z_{i1}, \dots, Z_{iq})$ be an indicator vector of label for i:

$$\mathbb{P}(i \in q) = \mathbb{P}(Z_{iq} = 1) = \alpha_q, \quad \text{s.t.} \sum_{q=1}^{Q} \alpha_q = 1.$$

By definition, $Z_i \sim \mathcal{M}(1, \boldsymbol{\alpha})$, with $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_Q)$.

Mixture models: the latent variables

When $(Z_1, \dots, Z_n)$ are independent categorical variables, they give a natural (latent) classification of the observations $(X_1, \dots, X_n)$ – or labels.

Notations

Let $(Z_1, \dots, Z_n)$ be *iid* categorical variables with distribution

$$\mathbb{P}(i \in q) = \mathbb{P}(Z_i = q) = \alpha_q, \quad \text{s.t.} \sum_{q=1}^{Q} \alpha_q = 1.$$

Alternative (equivalent) notation

Let $Z_i = (Z_{i1}, \dots, Z_{iq})$ be an indicator vector of label for i:

$$\mathbb{P}(i \in q) = \mathbb{P}(Z_{iq} = 1) = \alpha_q, \quad \text{s.t.} \sum_{q=1}^{Q} \alpha_q = 1.$$

By definition, $Z_i \sim \mathcal{M}(1, \boldsymbol{\alpha})$, with $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_Q)$.

Mixture models: the latent variables

When $(Z_1, \dots, Z_n)$ are independent categorical variables, they give a natural (latent) classification of the observations $(X_1, \dots, X_n)$ – or labels.

Notations

Let $(Z_1, \dots, Z_n)$ be *iid* categorical variables with distribution

$$\mathbb{P}(i \in q) = \mathbb{P}(Z_i = q) = \alpha_q, \quad \text{s.t.} \sum_{q=1}^{Q} \alpha_q = 1.$$

Alternative (equivalent) notation

Let $Z_i = (Z_{i1}, \dots, Z_{iq})$ be an indicator vector of label for i:

$$\mathbb{P}(i \in q) = \mathbb{P}(Z_{iq} = 1) = \alpha_q, \quad \text{s.t.} \sum_{q=1}^{Q} \alpha_q = 1.$$

By definition, $Z_i \sim \mathcal{M}(1, \boldsymbol{\alpha})$, with $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_Q)$.

Mixture models: the manifest variables

A mixture model represents the presence of subpopulations within an overall population as follows:

$$\mathbb{P}(X_i) = \sum_{z_i \in \mathcal{Z}_i} \mathbb{P}(X_i, Z_i) = \sum_{Z_i \in \mathcal{Z}_i} \mathbb{P}(X_i | Z_i) \mathbb{P}(Z_i).$$

Conditional distribution of the manifest variables

We assume a parametric distribution of X in each subpopulation

$$X_i | \{Z_i = q\} \sim \mathbb{P}_{\theta_q} \qquad \left(\Leftrightarrow X_i | \{Z_{iq}\} = 1 \sim \mathbb{P}_{\theta_q} \right)$$

The specificity of each class is handled by $\{\boldsymbol{\theta}_q\}_{q=1}^{Q}$.

Mixture models: likelihoods

The complete-data likelihood

It is the join distribution of (X_i, Z_i):

$$\mathbb{P}(X_i, Z_i) = \alpha_{Z_i} \mathbb{P}_{\boldsymbol{\theta}_{Z_i}}(X_i)$$

The incomplete-data likelihood

It is the marginal distribution of X_i once Z_i integrated:

$$\mathbb{P}(X_i) = \sum_{q=1}^{Q} \mathbb{P}(X_i, Z_i = q) = \sum_{q=1}^{Q} \alpha_q \mathbb{P}_{\boldsymbol{\theta}_q}(X_i)$$

$\rightsquigarrow$ A mixture model is a sum of distributions weigthed by the proportion of each subpopulation.

Mixture models: likelihoods

The complete-data likelihood

It is the join distribution of (X_i, Z_i):

$$\mathbb{P}(X_i, Z_i) = \alpha_{Z_i} \mathbb{P}_{\boldsymbol{\theta}_{Z_i}}(X_i)$$

The incomplete-data likelihood

It is the marginal distribution of X_i once Z_i integrated:

$$\mathbb{P}(X_i) = \sum_{q=1}^{Q} \mathbb{P}(X_i, Z_i = q) = \sum_{q=1}^{Q} \alpha_q \mathbb{P}_{\boldsymbol{\theta}_q}(X_i)$$

$\rightsquigarrow$ A mixture model is a sum of distributions weigthed by the proportion of each subpopulation.

Outline

Intractability of the Likelihood

Maximum Likelihood Estimator

The MLE aims to maximize the (marginal) likehood of the observations:

$$L(\boldsymbol{\theta}; \mathbf{X}) = \mathbb{P}_{\boldsymbol{\theta}}((X_1, \ldots, X_n)) = \int_{\mathbf{Z} \in \mathcal{Z}} \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z}) \mathrm{d}\mathbf{Z}$$

Integrations are summation over $\{1, \ldots, Q\}$: we have Q^n terms !

Intractable summation

With mixture models, for $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_Q)$ we have

$$\log L(\boldsymbol{\theta}; \mathbf{X}) = \sum_{i=1}^{n} \log \left\{ \sum_{q=1}^{Q} \alpha_q \mathbb{P}_{\boldsymbol{\theta}_q}(X_i) \right\}.$$

$\rightsquigarrow$ Direct maximization of the likelihood is impossible in practice

Intractability of the Likelihood

Maximum Likelihood Estimator

The MLE aims to maximize the (marginal) likehood of the observations:

$$L(\boldsymbol{\theta}; \mathbf{X}) = \mathbb{P}_{\boldsymbol{\theta}}((X_1, \dots, X_n)) = \int_{\mathbf{Z} \in \mathcal{Z}} \mathbb{P}_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{Z}) \mathrm{d}\mathbf{Z}$$

Integrations are summation over $\{1, \dots, Q\}$: we have Q^n terms !

Intractable summation

With mixture models, for $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_Q)$ we have

$$\log L(\boldsymbol{\theta}; \mathbf{X}) = \sum_{i=1}^{n} \log \left\{ \sum_{q=1}^{Q} \alpha_q \mathbb{P}_{\boldsymbol{\theta}_q}(X_i) \right\}.$$

$\rightsquigarrow$ Direct maximization of the likelihood is impossible in practice

Bayes decision rule / Maximum *a posteriori*

Principle

Affect an individual i to the subpopulation which is the most likely according to the data:

$$\tau_{iq} = \mathbb{P}(Z_{iq} = 1 | X_i = x_i)$$

This is the posterior probability for $i \in q$.

Application of the Bayes Theorem

It is straightforward to show that

$$\tau_{iq} = \frac{\alpha_q \mathbb{P}_{\theta_q}(x_i)}{\sum_{q=1}^{Q} \alpha_q \mathbb{P}_{\theta_q}(x_i)}$$

Principle of the EM algorithm

If $\boldsymbol{\theta}$ were known

...estimating the posterior probability $\mathbb{P}(Z_i|\mathbf{X})$ of $\mathbf{Z}$ should be easy
By means of the Bayes decision rule

If $\mathbf{Z}$ were known...

...estimating the best set of parameter $\boldsymbol{\theta}$ should be easy
This is close to usual maximum likelihood estimation

EM principle

Maximize the marginal likelihood iteratively:

1. Initialize θ
2. Compute the probability of $\mathbf{Z}$ given θ
3. Get a better θ with the new $\mathbf{Z}$
4. Iterate until convergence

Principle of the EM algorithm

If $\boldsymbol{\theta}$ were known

. . . estimating the posterior probability $\mathbb{P}(Z_i|\mathbf{X})$ of $\mathbf{Z}$ should be easy
By means of the Bayes decision rule

If $\mathbf{Z}$ were known. . .

. . . estimating the best set of parameter $\boldsymbol{\theta}$ should be easy
This is close to usual maximum likelihood estimation

EM principle

Maximize the marginal likelihood iteratively:

1. Initialize $\boldsymbol{\theta}$
2. Compute the probability of $\mathbf{Z}$ given $\boldsymbol{\theta}$
3. Get a better $\boldsymbol{\theta}$ with the new $\mathbf{Z}$
4. Iterate until convergence

Formal algorithm

Initialization: start from a good guess either of $\mathbf{Z}$ or $\boldsymbol{\theta}$, then iterate 1-2

1. Expectation step

Calculate the expected value of the loglikelihood under the current $\boldsymbol{\theta}$

$$Q\left(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}\right) = \mathbb{E}_{\mathbf{Z}|\mathbf{X};\boldsymbol{\theta}^{(t)}}\left[\log L(\boldsymbol{\theta};\mathbf{X},\mathbf{Z})\right] \qquad (\textit{needs } \mathbb{P}_{\boldsymbol{\theta}^{(t)}}(\mathbf{Z}|\mathbf{X}))$$

2. Maximization step

Find the parameters that maximize this quantity

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q\left(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}\right)$$

Stop when $\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\| < \varepsilon$ or $\|Q^{(t+1)} - Q^{(t)}\| < \varepsilon$

(Basic) Convergence analysis

Theorem

At each step of the EM algorithm, the loglikelihood increases. EM thus reaches a local optimum.

Proof.

On board. □

Choosing the number of component

Reminder: Bayesian Information Criterion

The BIC is a model selection criterion which penalizes the adjustement to the data by the number of parameter in model $\mathcal{M}$ as follows:

$$\mathrm{BIC}(\mathcal{M}) = \log L(\hat{\boldsymbol{\theta}}; \mathbf{X}) - \frac{1}{2}\log(n)\mathrm{df}(\mathcal{M}).$$

Integrated Classification Criterion

It is an adaptation working with the complete-data likelihood:

$$\begin{aligned}\mathrm{ICL}(\mathcal{M}) &= \log L(\hat{\boldsymbol{\theta}}; \mathbf{X}, \hat{\mathbf{Z}}) + \frac{1}{2}\log(n)\mathrm{df}(\mathcal{M}) \\ &= \mathrm{BIC} - \mathcal{H}(\mathbb{P}(\hat{\mathbf{Z}}|\mathbf{X}),\end{aligned}$$

where the entropy $\mathcal{H}$ measures the separability of the subpopulations.

$\rightsquigarrow$ We choose $\mathcal{M}(Q)$ that maximizes either BIC or ICL

Choosing the number of component

Reminder: Bayesian Information Criterion

The BIC is a model selection criterion which penalizes the adjustement to the data by the number of parameter in model $\mathcal{M}$ as follows:

$$\mathrm{BIC}(\mathcal{M}) = \log L(\hat{\boldsymbol{\theta}}; \mathbf{X}) - \frac{1}{2}\log(n)\mathrm{df}(\mathcal{M}).$$

Integrated Classification Criterion

It is an adaptation working with the complete-data likelihood:

$$\begin{aligned}\mathrm{ICL}(\mathcal{M}) &= \log L(\hat{\boldsymbol{\theta}}; \mathbf{X}, \hat{\mathbf{Z}}) + \frac{1}{2}\log(n)\mathrm{df}(\mathcal{M}) \\ &= \mathrm{BIC} - \mathcal{H}(\mathbb{P}(\hat{\mathbf{Z}}|\mathbf{X}),\end{aligned}$$

where the entropy $\mathcal{H}$ measures the separability of the subpopulations.

$\rightsquigarrow$ We choose $\mathcal{M}(Q)$ that maximizes either BIC or ICL

Choosing the number of component

Reminder: Bayesian Information Criterion

The BIC is a model selection criterion which penalizes the adjustement to the data by the number of parameter in model $\mathcal{M}$ as follows:

$$\mathrm{BIC}(\mathcal{M}) = \log L(\hat{\boldsymbol{\theta}}; \mathbf{X}) - \frac{1}{2}\log(n)\mathrm{df}(\mathcal{M}).$$

Integrated Classification Criterion

It is an adaptation working with the complete-data likelihood:

$$\begin{aligned}\mathrm{ICL}(\mathcal{M}) &= \log L(\hat{\boldsymbol{\theta}}; \mathbf{X}, \hat{\mathbf{Z}}) + \frac{1}{2}\log(n)\mathrm{df}(\mathcal{M}) \\ &= \mathrm{BIC} - \mathcal{H}(\mathbb{P}(\hat{\mathbf{Z}}|\mathbf{X}),\end{aligned}$$

where the entropy $\mathcal{H}$ measures the separability of the subpopulations.

$\rightsquigarrow$ We choose $\mathcal{M}(Q)$ that maximizes either BIC or ICL

Outline

Mixture of Gaussians

Calculs in the univariate case: complete likelihood

The distribution of X_i conditional on the label of i is assumed to be a univariate Gaussian distribution with unknown parameters:

$$X_i | Z_{iq} = 1 \sim \mathcal{N}(\mu_q, \sigma_q^2)$$

complete Likelihood $(\mathbf{X}, \mathbf{Z})$

The model complete loglikelihood is

$$\log L(\boldsymbol{\mu}, \boldsymbol{\sigma}^2; \mathbf{X}, \mathbf{Z}) = \sum_{i=1}^{n} \sum_{q=1}^{Q} Z_{iq} \left(\log \alpha_q - \log \sigma_q - \log(\sqrt{2\pi}) - \frac{1}{2\sigma_q^2} (x_i - \mu_q)^2 \right)$$

Mixture of Gaussians

Calculs in the univariate case: E-step

E-step

For fixed values of μ_q, σ_q^2 and α_q, the estimates of the posterior probabilities $\hat{\tau}_{iq} = \mathbb{P}(Z_{iq} = 1 | X_i)$ are

$$\hat{\tau}_{iq} = \frac{\alpha_q \mathcal{N}(x_i; \mu_q, \sigma_q^2)}{\sum_{q=1}^{Q} \alpha_q \mathcal{N}(x_i; \mu_q, \sigma_q^2)},$$

where $\mathcal{N}$ is the density of the normal distribution.

Mixture of Gaussians

Calculs in the univariate case: M-step

M-step

For fixed values of τ_{iq}, the estimates of the model parameters are

$$\hat{\alpha}_q = \frac{\sum_{i=1}^n \tau_{iq}}{\sum_{i=1}^n \sum_{q=1}^Q \tau_{iq}} \quad \hat{\mu}_q = \frac{\sum_i \tau_{iq} x_i}{\sum_i \tau_{iq}} \quad \hat{\sigma}_q^2 = \frac{\sum_{i=1}^n \tau_{iq} (x_i - \mu_q)^2}{\sum_{i=1}^n \tau_{iq}}$$

R code: auxiliary functions

We start by defining functions to compute the complete model loglikelihood, perform the E step and the M step.

```r
get.cloglik <- function(X, Z, theta) {
  alpha <- theta$alpha; mu <- theta$mu; sigma <- theta$sigma
  xs <- scale(matrix(X,length(x),length(alpha)),mu,sigma)
  return(sum(Z*(log(alpha)-log(sigma)-.5*(log(2*pi)+xs^2))))
}

M.step <- function(X, tau) {
  n <- length(X); Q <- ncol(tau)
  alpha  <- colMeans(tau)
  mu     <- colMeans(tau * matrix(X,n,Q)) / alpha
  sigma  <- sqrt(colMeans(tau*sweep(matrix(X,n,Q),2,mu,"-")^2)/alpha)
  return(list(alpha=alpha, mu=mu, sigma=sigma))
}

E.step <- function(X, theta) {
  tau <- mapply(function(alpha, mu, sigma) {
      alpha*dnorm(X,mu,sigma)
    }, theta$alpha, theta$mu, theta$sigma)
  return(tau / rowSums(tau))
}
```

R code: EM for univariate mixture

```r
EM.mixture <- function(X, Q,
                       init.cl=sample(1:Q,n,rep=TRUE), max.iter=100, eps=1e-5) {
    n <- length(X); tau <- matrix(0,n,Q); tau[cbind(1:n,init.cl)] <- 1
    Eloglik <- vector("numeric", max.iter)
    iter <- 0; cond <- FALSE

    while (!cond) {
        iter <- iter + 1
        ## M step
        theta <- M.step(X, tau)
        ## E step
        tau <- E.step(X, theta)
        ## check consistency
        Eloglik[iter] <- get.cloglik(X, tau, theta)
        if (iter > 1)
            cond <- (iter>=max.iter) | Eloglik[iter]-Eloglik[iter-1] < eps
    }

    return(list(alpha = theta$alpha,  mu = theta$mu,  sigma = theta$sigma,
                tau   = tau, cl = apply(tau, 1, which.max),
                Eloglik = Eloglik[1:iter]))
}
```

Example: data generation

We first generate data with 4 components:

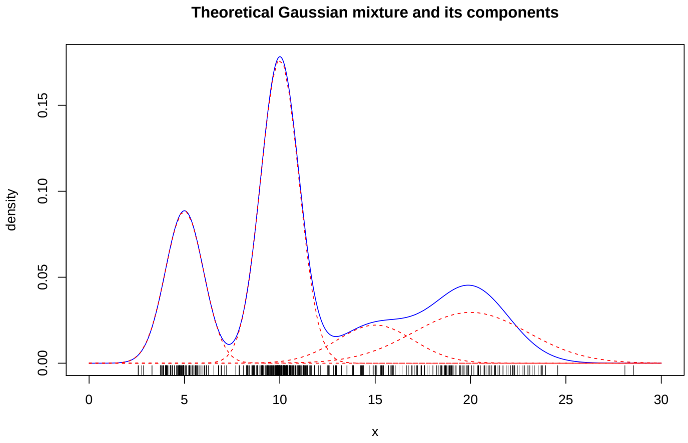
```
mu1 <- 5   ; sigma1 <- 1; n1 <- 100
mu2 <- 10  ; sigma2 <- 1; n2 <- 200
mu3 <- 15  ; sigma3 <- 2; n3 <- 50
mu4 <- 20  ; sigma4 <- 3; n4 <- 100
cl <- rep(1:4,c(n1,n2,n3,n4))
x <- c(rnorm(n1,mu1,sigma1),rnorm(n2,mu2,sigma2),
       rnorm(n3,mu3,sigma3),rnorm(n4,mu4,sigma4))
n <- length(x)

## we randomize the class ordering
rnd <- sample(1:n)
cl <- cl[rnd]
x  <- x[rnd]

alpha <- c(n1,n2,n3,n4)/n
```

Example: data generation - plot I

Let us plot the data and the theoretical mixture.

```
curve(alpha[1]*dnorm(x,mu1,sigma1) +
      alpha[2]*dnorm(x,mu2,sigma2) +
      alpha[3]*dnorm(x,mu3,sigma3) +
      alpha[4]*dnorm(x,mu4,sigma3),
      col="blue", lty=1, from=0,to=30, n=1000,
      main="Theoretical Gaussian mixture and its components",
      xlab="x", ylab="density")
curve(alpha[1]*dnorm(x,mu1,sigma1), col="red", add=TRUE, lty=2)
curve(alpha[2]*dnorm(x,mu2,sigma2), col="red", add=TRUE, lty=2)
curve(alpha[3]*dnorm(x,mu3,sigma3), col="red", add=TRUE, lty=2)
curve(alpha[4]*dnorm(x,mu4,sigma4), col="red", add=TRUE, lty=2)
rug(x)
```

Example: data generation - plot II

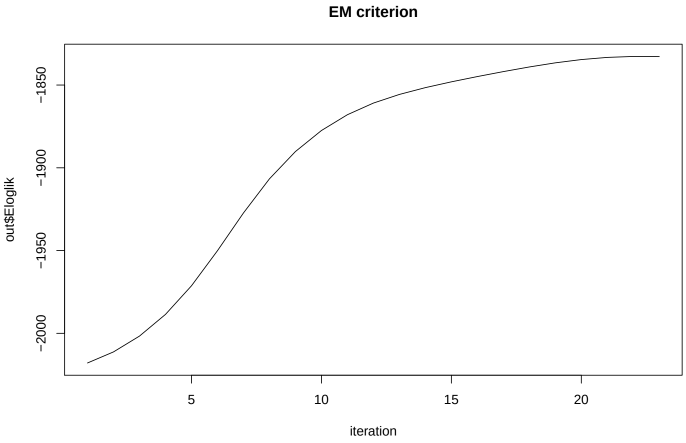


Implementation

See practical 2.

Example: adjustment

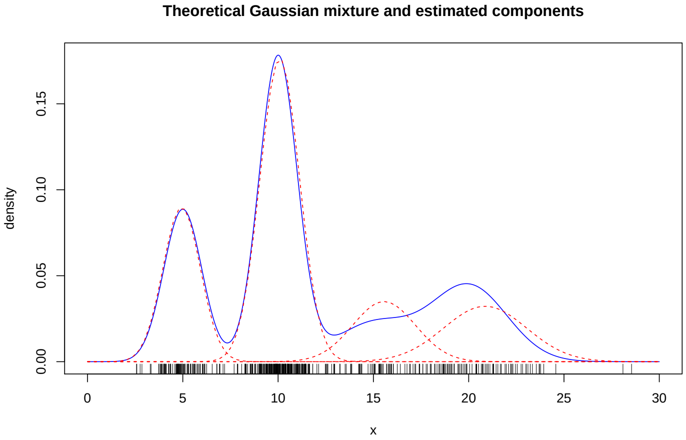
```r
out <- EM.mixture(x, Q=4, init.cl=sample(1:4,n,rep=TRUE))
plot(out$Eloglik, main="EM criterion", type="l", xlab="iteration")
```



Example: adjustment - plot I

```r
out <- EM.mixture(x, Q=4, init.cl=kmeans(x,4)$cl)
curve(alpha[1]*dnorm(x,mu1,sigma1) +
      alpha[2]*dnorm(x,mu2,sigma2) +
      alpha[3]*dnorm(x,mu3,sigma3) +
      alpha[4]*dnorm(x,mu4,sigma3), col="blue",
      lty=1, from=0,to=30, n=1000,
      main="Theoretical Gaussian mixture and estimated components",
      xlab="x", ylab="density")
curve(out$alpha[1]*dnorm(x,out$mu[1],out$sigma[1]), col="red", add=TRUE, lty=2)
curve(out$alpha[2]*dnorm(x,out$mu[2],out$sigma[2]), col="red", add=TRUE, lty=2)
curve(out$alpha[3]*dnorm(x,out$mu[3],out$sigma[3]), col="red", add=TRUE, lty=2)
curve(out$alpha[4]*dnorm(x,out$mu[4],out$sigma[4]), col="red", add=TRUE, lty=2)
rug(x)
```

Example: adjustment - plot II



Example: adjustment - classification I

```
table(cl, out$cl)

##
## cl    1   2   3   4
##   1  99   0   1   0
##   2   1   0 199   0
##   3   0   1   6  43
##   4   0  81   0  19

aricode::ARI(cl, out$cl)

## [1] 0.8946429
```