SY19 – Machine Learning

Chapter 7: Gaussian mixture models and the EM algorithm

Thierry Denœux

Université de technologie de Compiègne

https://www.hds.utc.fr/~tdenoeux
email: tdenoeux@utc.fr

Automne 2021

Overview

Overview

Back to LDA and QDA

- In LDA and QDA, we assume that the conditional density of input vector X given $Y = k$ is multivariate Gaussian

$$\phi(x; \mu_k, \mathbf{\Sigma}_k) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}_k^{-1}(x-\mu_k)\right)$$

(with $\mathbf{\Sigma}_k = \mathbf{\Sigma}$ in the case of LDA)

- The marginal density of X is then a mixture of c Gaussian densities:

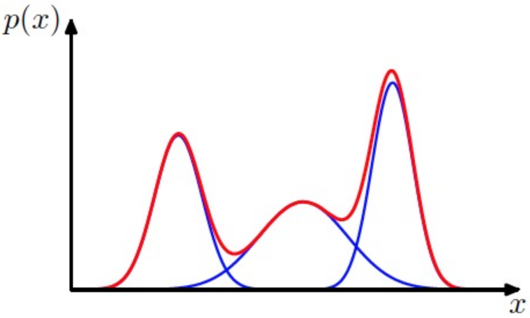
$$p(x) = \sum_{k=1}^{c} p(x \mid Y = k) P(Y = k) = \sum_{k=1}^{c} \pi_k \phi(x; \mu_k, \mathbf{\Sigma}_k)$$

- This is called a Gaussian Mixture Model (GMM).

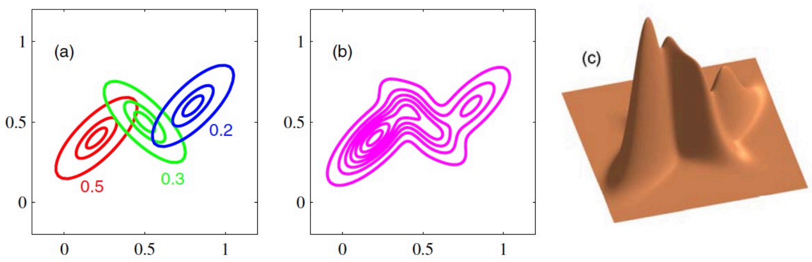
Gaussian Mixture Models

- GMMs are widely used in Machine Learning for
  - Density estimation
  - Clustering (finding groups in data)
  - Classification (modeling complex-shaped class distributions)
  - Regression (accounting for different linear relations within subgroups of a population)
  - etc.

Example with $p = 1$



Example with $p = 2$



How to generate data from a mixture?

- Assume $X \sim \sum_{k=1}^{c} \pi_k \mathcal{N}(\mu_k, \mathbf{\Sigma}_k)$
- It is the marginal distribution of X in the pair (X, Y), where Y takes values in $\{1, \ldots, c\}$ with probabilities $\pi_1, \ldots, \pi_c$, and the conditional distribution of X given $Y = k$ is the normal distribution $\mathcal{N}(\mu_k, \mathbf{\Sigma}_k)$
- How to generate X?
  1. Generate $Y \in \{1, \ldots, c\}$ with probabilities $\pi_1, \ldots, \pi_c$.
  2. If $Y = k$, generate X from $p(x \mid Y = k) = \phi(x; \mu_k, \mathbf{\Sigma}_k)$.
- Remark: we can define mixtures of other distributions. In this chapter, we will focus (without loss of generality) on mixtures of normal distributions, called Gaussian mixtures.

Overview

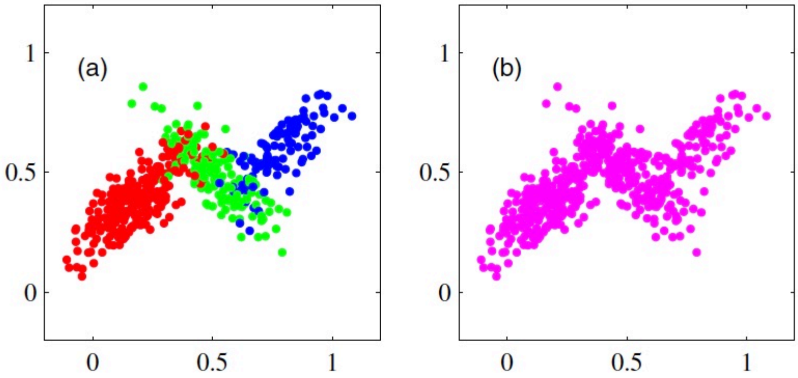
Supervised learning

- In discriminant analysis, we observe both the input vector X and the response (class label) Y for n individuals taken randomly from the population.
- The learning set has the form $\mathcal{L}_s = \{(x_i, y_i)\}_{i=1}^n$. We say that the data are labeled.
- Learning a classifier from such data is called supervised learning.

Unsupervised learning

- In some situations, we observe X, but Y is not observed. We say that Y is a latent variable.
- The learning set is composed of unlabeled data of the form $\mathcal{L}_{ns} = \{x_i\}_{i=1}^n$.
- Estimating the model parameters from such data is called unsupervised learning.
- Applications: density estimation, clustering, feature extraction.
- Unsupervised learning is usually more difficult than supervised learning, because we have less information to estimate the parameters.

Labeled vs. unlabeled data



Semi-supervised learning

- Sometimes, we collect of lot of data, but we can label only a part of them.
- Examples: image data from the web, or from sensors on a robot.
- The data then have the form

$$\mathcal{L}_{ss} = \underbrace{\{(x_i, y_i)\}_{i=1}^{n_s}}_{\text{labeled part}} \cup \underbrace{\{x_i\}_{i=n_s+1}^{n}}_{\text{unlabeled part}}$$

- This is called a semi-supervised learning problem.
- Semi-supervised learning is intermediate between supervised and unsupervised learning.

Overview

Maximum likelihood: supervised case I

- In the case of supervised learning of GMMs, the MLEs of μ_k, $\boldsymbol{\Sigma}_k$ and π_k have simple closed-form expressions.
- Assuming the sample $(X_1, Y_1) \ldots, (X_n, Y_n)$ to be i.i.d., the likelihood function is

$$L(\theta; \mathcal{L}_s) = \prod_{i=1}^{n} p(x_i, y_i) = \prod_{i=1}^{n} \underbrace{p(x_i \mid Y_i = y_i)}_{\prod_{k=1}^{c} \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k)^{y_{ik}}} \underbrace{p(Y_i = y_i)}_{\prod_{k=1}^{c} \pi_k^{y_{ik}}}$$

$$= \prod_{i=1}^{n} \prod_{k=1}^{c} \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k)^{y_{ik}} \pi_k^{y_{ik}}$$

with $y_{ik} = I(y_i = k)$.

Maximum likelihood: supervised case II

- The log-likelihood function is

$$\ell(\theta;\mathcal{L}_s) = \underbrace{\sum_{k=1}^{c}\left\{\sum_{i=1}^{n} y_{ik}\log\phi(x_i;\mu_k,\boldsymbol{\Sigma}_k)\right\}}_{\text{term } \ell_k \text{ depending on } \mu_k \text{ and } \boldsymbol{\Sigma}_k} + \underbrace{\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}\log\pi_k}_{\text{term depending on } \pi_1,\ldots,\pi_c}$$

- The parameters $\mu_k, \boldsymbol{\Sigma}_k$ can be estimated separately using the data from class k.

MLE in the supervised case I

- We have

$$\ell_k = -\frac{1}{2}\sum_{i=1}^{n} y_{ik}(x_i-\mu_k)^T \boldsymbol{\Sigma}_k^{-1}(x_i-\mu_k) - \frac{n_k}{2}\log|\boldsymbol{\Sigma}_k| - \frac{n_k p}{2}\log(2\pi)$$

  with $n_k = \sum_{i=1}^n y_{ik}$.

- The derivative wrt to μ_k is

$$\sum_i y_{ik}\boldsymbol{\Sigma}_k^{-1}(x_i-\mu_k) = \boldsymbol{\Sigma}_k^{-1}\sum_i y_{ik}(x_i-\mu_k).$$

  Hence,

$$\boxed{\widehat{\mu}_k = \frac{1}{n_k}\sum_{i=1}^{n} y_{ik}x_i}$$

MLE in the supervised case II

- It can be shown that

$$\boxed{\widehat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k}\sum_{i=1}^n y_{ik}(x_i - \widehat{\mu}_k)(x_i - \widehat{\mu}_k)^T}$$

- To find the MLE of the π_k, we maximize the last term

$$\sum_{i=1}^n \sum_{k=1}^c y_{ik} \log \pi_k$$

wrt to π_k, subject to the constraint $\sum_{k=1}^c \pi_k = 1$.

- The solution is

$$\boxed{\widehat{\pi}_k = \frac{n_k}{n}, \quad k = 1, \ldots, c}$$

Maximum likelihood: unsupervised case

- In the case of unsupervised learning, assuming the sample $X_1, \ldots, X_n$ to be i.i.d., the likelihood is

$$L(\theta; \mathcal{L}_{ns}) = \prod_{i=1}^{n} p(x_i)$$

and the log-likelihood function is

$$\ell(\theta; \mathcal{L}_{ns}) = \sum_{i=1}^{n} \log p(x_i) = \sum_{i=1}^{n} \left(\log \sum_{k=1}^{c} \pi_k \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k) \right)$$

- We can no longer separate the terms corresponding to each class.
- Maximizing the log-likelihood becomes a difficult nonlinear optimization problem, for which no closed-form solution exists.
- A powerful method: the Expectation-Maximization (EM) algorithm.

Overview

EM Algorithm

- An iterative optimization strategy useful when the maximizing the likelihood is difficult, but:
  - There are missing (non-observed) data
  - If the missing data were observed, maximizing the likelihood would be easy.
- Many applications in statistics and ML
- Can be very simple to implement. Can reliably find an optimum through stable, uphill steps.

Overview

Notation

$$
\begin{aligned}
\mathbf{X} &: \text{Observed variables}\\
\mathbf{Y} &: \text{Missing or latent variables}\\
\mathbf{Z} &: \text{Complete data } \mathbf{Z} = (\mathbf{X}, \mathbf{Y})\\
\theta &: \text{Unknown parameter}\\
L(\theta) &: \text{observed-data likelihood, short for } L(\theta; \mathbf{x}) = p(\mathbf{x}; \theta)\\
L_c(\theta) &: \text{complete-data likelihood, short for } L(\theta; \mathbf{z}) = p(\mathbf{z}; \theta)\\
\ell(\theta), \ell_c(\theta) &: \text{observed and complete-data log-likelihoods}
\end{aligned}
$$

Q function

- Suppose we seek to maximize $L(\theta)$ with respect to θ.
- Define $Q(\theta; \theta^{(t)})$ to be the expectation of the complete-data log-likelihood (assuming $\theta = \theta^{(t)}$), conditional on the observed data $\mathbf{X} = \mathbf{x}$. Namely

$$\begin{aligned} Q(\theta, \theta^{(t)}) =& \mathbb{E}_{\theta^{(t)}} \left\{ \ell_c(\theta) \mid \mathbf{x} \right\} \\ =& \mathbb{E}_{\theta^{(t)}} \left\{ \log p(\mathbf{Z}; \theta) \mid \mathbf{x} \right\} \\ =& \int \left[\log p(\mathbf{z}; \theta) \right] \underbrace{p(\mathbf{z} \mid \mathbf{x}; \theta^{(t)})}_{p(\mathbf{y} \mid \mathbf{x}; \theta^{(t)})} \, d\mathbf{y} \end{aligned}$$

($p(\mathbf{z} \mid \mathbf{x}; \theta^{(t)}) = p(\mathbf{y} \mid \mathbf{x}; \theta^{(t)})$ because $\mathbf{Y}$ is the only random part of $\mathbf{Z}$ once we are given $\mathbf{X} = \mathbf{x}$)

The EM Algorithm

Start with $\theta^{(0)}$ and set $t = 0$. Then

1. **E step**: Compute $Q(\theta, \theta^{(t)})$.
2. **M step**: Maximize $Q(\theta, \theta^{(t)})$ with respect to θ. Set $\theta^{(t+1)}$ equal to the maximizer of Q.
3. Return to the E step and increment t unless a stopping criterion has been met, e.g.,
$$|\ell(\theta^{(t+1)}) - \ell(\theta^{(t)})| \leq \epsilon$$

Convergence of the EM Algorithm

- It can be proved that $L(\theta)$ increases after each EM iteration, i.e., $L(\theta^{(t+1)}) \geq L(\theta^{(t)})$ for $t = 0, 1, \ldots$ (see below)
- Consequently, the algorithm converges to a local maximum of $L(\theta)$ if the likelihood function is bounded above.
- Typically, we run the algorithm several times with random initial conditions, and we keep the results of the best run.

Overview

Mixture of two univariate normal distributions

- Let $\mathbf{X} = (X_1, \ldots, X_n)$ be an i.i.d. sample from a mixture of two univariate normal distributions $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$, with pdf

  $$p(x_i; \theta) = \pi\phi(x_i; \mu_1, \sigma_1) + (1-\pi)\phi(x_i; \mu_2, \sigma_2),$$

  where $\phi(\cdot; \mu, \sigma)$ is the univariate normal pdf and

  $$\theta = (\mu_1, \sigma_1, \mu_2, \sigma_2, \pi)^T$$

  is the vector of parameters.
- We introduce latent variables $\mathbf{Y} = (Y_1, \ldots, Y_n)$, such that
  - $Y_i \sim \mathcal{B}(\pi)$,
  - $p(x_i \mid Y_i = 1) = \phi(x_i; \mu_1, \sigma_1)$ and
  - $p(x_i \mid Y_i = 0) = \phi(x_i; \mu_2, \sigma_2)$.

Observed and complete-data likelihoods

- Observed-data likelihood:

$$L(\theta) = \prod_{i=1}^{n} p(x_i;\theta) = \prod_{i=1}^{n} \left[\pi\phi(x_i;\mu_1,\sigma_1) + (1-\pi)\phi(x_i;\mu_2,\sigma_2)\right]$$

- Complete-data likelihood:

$$L_c(\theta) = \prod_{i=1}^{n} p(x_i, y_i;\theta) = \prod_{i=1}^{n} p(x_i \mid y_i;\theta)p(y_i;\pi)$$

$$= \prod_{i=1}^{n} \left\{\phi(x_i;\mu_1,\sigma_1)^{y_i}\phi(x_i;\mu_2,\sigma_2)^{1-y_i}\pi^{y_i}(1-\pi)^{1-y_i}\right\}$$

Derivation of function Q

- Complete-data log-likelihood:

$$\ell_c(\theta) = \sum_{i=1}^{n} \{y_i \log \phi(x_i; \mu_1, \sigma_1) + (1 - y_i) \log \phi(x_i; \mu_2, \sigma_2)\} + \sum_{i=1}^{n} \{y_i \log \pi + (1 - y_i) \log(1 - \pi)\}$$

- It is linear in the y_i. Consequently, the Q function is simply

$$Q(\theta, \theta^{(t)}) = \sum_{i=1}^{n} \left\{y_i^{(t)} \log \phi(x_i; \mu_1, \sigma_1) + (1 - y_i^{(t)}) \log \phi(x_i; \mu_2, \sigma_2)\right\} + \sum_{i=1}^{n} \left\{y_i^{(t)} \log \pi + (1 - y_i^{(t)}) \log(1 - \pi)\right\}$$

with $y_i^{(t)} = \mathbb{E}_{\theta^{(t)}}[Y_i \mid x_i]$

EM algorithm: E-step

Compute

$$
\begin{aligned}
y_i^{(t)} &= \mathbb{E}_{\theta^{(t)}}[Y_i \mid x_i] \\
&= \mathbb{P}_{\theta^{(t)}}[Y_i = 1 \mid x_i] \\
&= \frac{\phi(x_i; \mu_1^{(t)}, \sigma_1^{(t)})\pi^{(t)}}{\phi(x_i; \mu_1^{(t)}, \sigma_1^{(t)})\pi^{(t)} + \phi(x_i; \mu_2^{(t)}, \sigma_2^{(t)})(1 - \pi^{(t)})}
\end{aligned}
$$

EM algorithm: M-step

Maximize $Q(\theta, \theta^{(t)})$. We get

$$\pi^{(t+1)} = \frac{n_1^{(t)}}{n},$$

$$\mu_1^{(t+1)} = \frac{\sum_{i=1}^n y_i^{(t)} x_i}{n_1^{(t)}}, \quad \sigma_1^{(t+1)} = \sqrt{\frac{\sum_{i=1}^n y_i^{(t)} (x_i - \mu_1^{(t+1)})^2}{n_1^{(t)}}}$$

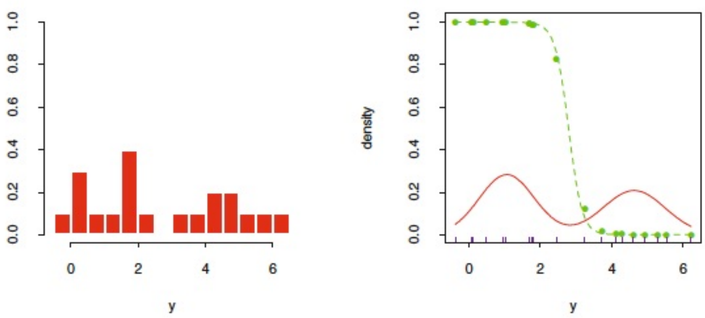
$$\mu_2^{(t+1)} = \frac{\sum_{i=1}^n (1 - y_i^{(t)}) x_i}{n_2^{(t)}}, \quad \sigma_2^{(t+1)} = \sqrt{\frac{\sum_{i=1}^n (1 - y_i^{(t)}) (x_i - \mu_2^{(t+1)})^2}{n_2^{(t)}}}$$

with

$$n_1^{(t)} = \sum_{i=1}^n y_i^{(t)} \quad \text{and} \quad n_2^{(t)} = n - n_1^{(t)}$$

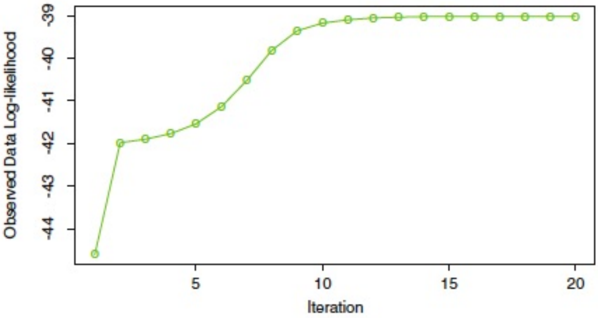
Example

-0.39	0.12	0.94	1.67	1.76	2.44	3.72	4.28	4.92	5.53
0.06	0.48	1.01	1.68	1.80	3.25	4.12	4.60	5.28	6.22



(green curve: $\mathbb{P}_{\widehat{\theta}}[Y = 1 \mid x]$ as a function of x, assuming $Y = 1$ corresponds to the left component)

Example (continued)



Solution: $\widehat{\mu}_1 = 4.66$, $\widehat{\sigma}_1 = 0.91$, $\widehat{\mu}_2 = 1.08$, $\widehat{\sigma}_2 = 0.90$, $\widehat{\pi} = 0.45$.

Overview

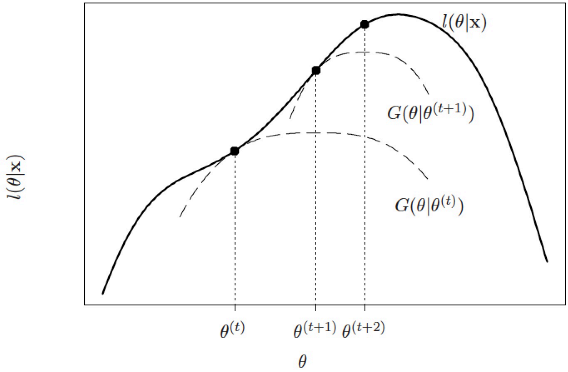
Why does it work?

- Ascent: Each M-step increases the log-likelihood.
- Optimization transfer:

$$\ell(\theta) \geq \underbrace{Q(\theta, \theta^{(t)}) + \ell(\theta^{(t)}) - Q(\theta^{(t)}, \theta^{(t)})}_{G(\theta,\theta^{(t)})}.$$

- The last two terms in $G(\theta, \theta^{(t)})$ do not depend on θ, so Q and G are maximized at the same θ.
- Further, G is tangent to ℓ at $\theta^{(t)}$, and lies everywhere below ℓ. We say that G is a minorizing function for ℓ (see next slide).
- EM transfers optimization from ℓ to the surrogate function G, which is more convenient to maximize.

The nature of EM (continued)



One-dimensional illustration of EM algorithm as a minorization or optimization transfer strategy. Each E step forms a minorizing function G, and each M step maximizes it to provide an uphill step.

Proof

- We have

$$p(y \mid x;\theta) = \frac{p(x,y;\theta)}{p(x;\theta)} = \frac{p(z;\theta)}{p(x;\theta)} \Rightarrow p(x;\theta) = \frac{p(z;\theta)}{p(y|x;\theta)}$$

- Consequently,

$$\ell(\theta) = \log p(x;\theta) = \underbrace{\log p(z;\theta)}_{\ell_c(\theta)} - \log p(y \mid x;\theta)$$

- Taking expectations on both sides wrt the conditional distribution of Z given $X = x$ and using $\theta^{(t)}$ for θ:

$$\ell(\theta) = Q(\theta,\theta^{(t)}) - \underbrace{\mathbb{E}_{\theta^{(t)}}[\log p(Y \mid x;\theta) \mid x]}_{H(\theta,\theta^{(t)})} \tag{1}$$

Proof - the minorizing function

- Now, for all $\theta \in \Theta$,

$$H(\theta, \theta^{(t)}) - H(\theta^{(t)}, \theta^{(t)}) = \mathbb{E}_{\theta^{(t)}} \left[\log \frac{p(Y \mid x; \theta)}{p(Y \mid x; \theta^{(t)})} \mid x \right] \tag{2a}$$

$$\leq \log \underbrace{\mathbb{E}_{\theta^{(t)}} \left[\frac{p(Y \mid x; \theta)}{p(Y \mid x; \theta^{(t)})} \mid x \right]}_{\int \frac{p(y|x;\theta)}{p(y|x;\theta^{(t)})} p(y|x;\theta^{(t)}) dy} (*) \tag{2b}$$

$$\leq \log \underbrace{\int p(y \mid x; \theta) dy}_{1} = 0 \tag{2c}$$

(*): from the concavity of the log and Jensen's inequality.

- Hence, $\theta^{(t)}$ is a maximizer of $H(\theta, \theta^{(t)})$

Proof - the minorizing function (continued)

Hence, for all $\theta \in \Theta$,

$$H(\theta^{(t)}, \theta^{(t)}) \geq H(\theta, \theta^{(t)})$$

$$Q(\theta^{(t)}, \theta^{(t)}) - \ell(\theta^{(t)}) \geq Q(\theta, \theta^{(t)}) - \ell(\theta)$$

$$\ell(\theta) \geq \underbrace{Q(\theta, \theta^{(t)}) + \ell(\theta^{(t)}) - Q(\theta^{(t)}, \theta^{(t)})}_{G(\theta,\theta^{(t)})}$$

Proof - G is tangent to ℓ at $\theta^{(t)}$

- As $\theta^{(t)}$ maximizes $H(\theta, \theta^{(t)}) = Q(\theta, \theta^{(t)}) - \ell(\theta)$, we have

$$H'(\theta, \theta^{(t)})|_{\theta=\theta^{(t)}} = Q'(\theta, \theta^{(t)})|_{\theta=\theta^{(t)}} - \ell'(\theta)|_{\theta=\theta^{(t)}} = 0,$$

so

$$Q'(\theta, \theta^{(t)})|_{\theta=\theta^{(t)}} = \ell'(\theta)|_{\theta=\theta^{(t)}}.$$

- Consequently, as $G(\theta, \theta^{(t)}) = Q(\theta, \theta^{(t)}) + \text{cst}$,

$$G'(\theta, \theta^{(t)})|_{\theta=\theta^{(t)}} = Q'(\theta, \theta^{(t)})|_{\theta=\theta^{(t)}} = \ell'(\theta)|_{\theta=\theta^{(t)}}.$$

Proof - monotonicity

- From (1),

$$\ell(\theta^{(t+1)}) - \ell(\theta^{(t)}) = \underbrace{Q(\theta^{(t+1)}, \theta^{(t)}) - Q(\theta^{(t)}, \theta^{(t)})}_{A} - \underbrace{\left(H(\theta^{(t+1)}, \theta^{(t)}) - H(\theta^{(t)}, \theta^{(t)}) \right)}_{B}$$

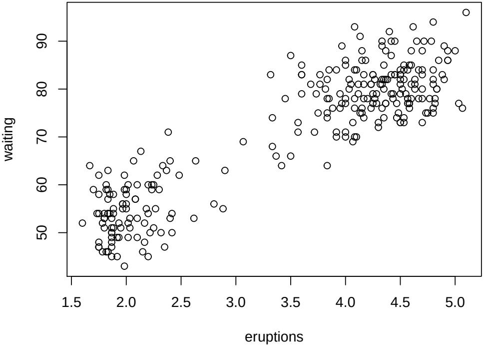
- $A \geq 0$ because $\theta^{(t+1)}$ is a maximizer of $Q(\theta, \theta^{(t)})$, and $B \leq 0$ because from (2) $\theta^{(t)}$ is a maximizer of $H(\theta, \theta^{(t)})$.
- Hence,

$$\ell(\theta^{(t+1)}) \geq \ell(\theta^{(t)})$$

Overview

Overview

Old Faithful geyser data



Waiting time between eruptions and duration of the eruption (in min) for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA (272 observations).

Old Faithful geyser data (continued)

- Questions:
  1. How can we best partition these data into c groups/clusters (for instance, $c = 2$)?
  2. What is the most plausible number of groups?
- Approach:
  1. Fit GMMs to these data
  2. Select the best model according to some criterion

General GMM

- Let $\mathbf{X} = (X_1, \ldots, X_n)$ be an i.i.d. sample from a mixture of c multivariate normal distributions $\mathcal{N}(\mu_k, \mathbf{\Sigma}_k)$ with proportions π_k. The pdf of X_i is

$$p(x_i; \theta) = \sum_{k=1}^{c} \pi_k \phi(x_i; \mu_k, \mathbf{\Sigma}_k),$$

  where θ is the vector of parameters.
- We introduce latent variables $\mathbf{Y} = (Y_1, \ldots, Y_n)$, such that
  - $Y_i \sim \mathcal{M}(1, \pi_1, \ldots, \pi_c)$
  - $p(x_i \mid Y_i = k) = \phi(x_i; \mu_k, \mathbf{\Sigma}_k)$, $k = 1 \ldots, c$

Observed and complete-data likelihoods

- Observed-data likelihood:

$$L(\theta) = \prod_{i=1}^{n} p(x_i; \theta) = \prod_{i=1}^{n} \sum_{k=1}^{c} \pi_k \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k)$$

- Complete-data likelihood:

$$L_c(\theta) = \prod_{i=1}^{n} p(x_i, y_i; \theta) = \prod_{i=1}^{n} p(x_i \mid y_i; \theta) p(y_i; \pi)$$

$$= \prod_{i=1}^{n} \prod_{k=1}^{c} \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k)^{y_{ik}} \pi_k^{y_{ik}}.$$

Derivation of function Q

- Complete-data log-likelihood:

$$\ell_c(\theta) = \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik} \log \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k) + \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik} \log \pi_k$$

- It is linear in the y_{ik}. Consequently, the Q function is simply

$$Q(\theta, \theta^{(t)}) = \sum_{k=1}^{c} \underbrace{\sum_{i=1}^{n} y_{ik}^{(t)} \log \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k)}_{\text{term depending only on } \mu_k \text{ and } \boldsymbol{\Sigma}_k} + \underbrace{\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}^{(t)} \log \pi_k}_{\text{term depending only on } \{\pi_k\}}$$

with $y_{ik}^{(t)} = \mathbb{E}_{\theta^{(t)}}[Y_{ik} \mid x_i] = \mathbb{P}_{\theta^{(t)}}[Y_i = k \mid x_i]$.

EM algorithm

- E-step: compute

$$y_{ik}^{(t)} = \mathbb{P}_{\theta^{(t)}}[Y_i = k \mid x_i] = \frac{\phi(x_i; \mu_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)})\pi_k^{(t)}}{\sum_{\ell=1}^{c} \phi(x_i; \mu_\ell^{(t)}, \boldsymbol{\Sigma}_\ell^{(t)})\pi_\ell^{(t)}}$$

- M-step: Maximize $Q(\theta, \theta^{(t)})$. We get

$$\pi_k^{(t+1)} = \frac{n_k^{(t)}}{n}, \quad \mu_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{n} y_{ik}^{(t)} x_i$$

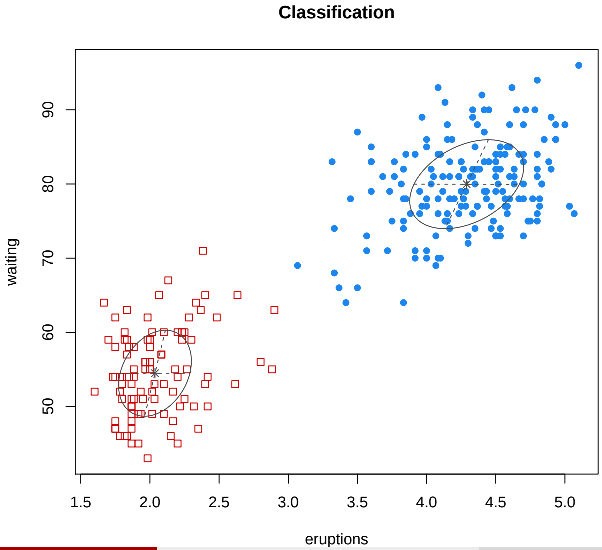
$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{n} y_{ik}^{(t)} (x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T$$

with $n_k^{(t)} = \sum_{i=1}^{n} y_{ik}^{(t)}$.

GMM with the package mclust

```
library(mclust)
data(faithful)

faithfulMclust <- Mclust(faithful,G=2,modelNames="VVV")
plot(faithfulMclust)
```

Result


Classification
waiting
eruptions
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
50
60
70
80
90

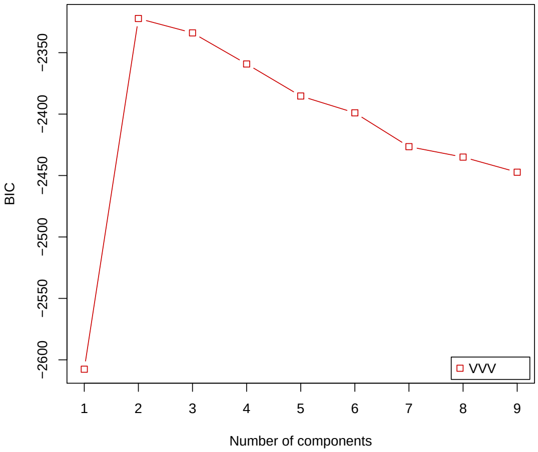
Choosing the number of clusters

- In clustering, selecting the number of clusters is often a difficult problem.
- This is a model selection problem. We can use the BIC criterion. (Reminder: $BIC = -2\ell(\widehat{\theta}) + d \log(n)$; actually, `Mclust` computes $-BIC$).

```
> faithfulMclust <- Mclust(faithful,modelNames="VVV")
> summary(faithfulMclust)
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model with 2 components:

 log.likelihood   n df       BIC       ICL
      -1130.264 272 11 -2322.192 -2322.695

Clustering table:
  1   2
175  97
```

Choosing the number of clusters

```
plot(faithfulMclust)
```



Reducing the number of parameters

- The general model has $c[p + p(p+1)/2 + 1] - 1$ parameters.
- When n is small and/or p is large: we need more parsimonious models (i.e., models with fewer parameters).
- Simple approaches:
  - Assume equal covariance matrix (homoscedasticity)
  - Assume the covariance matrices to be diagonal, or scalar
- More flexible approach: reparameterize matrix $\boldsymbol{\Sigma}_k$ using its eigendecomposition.

Eigendecomposition of $\boldsymbol{\Sigma}_k$

- As matrix $\boldsymbol{\Sigma}_k$ is symmetric, we can write

$$\boldsymbol{\Sigma}_k = \boldsymbol{D}_k \boldsymbol{\Lambda}_k \boldsymbol{D}_k^T$$

  where
  - $\boldsymbol{\Lambda}_k = \text{diag}(\lambda_{k1}, \ldots, \lambda_{kp})$ is a diagonal matrix whose components are the decreasing eigenvalues of $\boldsymbol{\Sigma}_k$, with $|\boldsymbol{\Lambda}_k| = \prod_{j=1}^p \lambda_{kj} = |\boldsymbol{\Sigma}_k|$
  - $\boldsymbol{D}_k$ is an orthogonal matrix ($\boldsymbol{D}_k \boldsymbol{D}_k^T = \mathbf{I}$) whose columns are the normalized eigenvectors of $\boldsymbol{\Sigma}_k$; it is a rotation matrix
- $\boldsymbol{\Lambda}_k$ can be further decomposed as

$$\boldsymbol{\Lambda}_k = \lambda_k \mathbf{A}_k$$

  where
  - $\lambda_k = \left(\prod_{j=1}^p \lambda_{kj}\right)^{1/p} = |\boldsymbol{\Sigma}_k|^{1/p}$
  - $\mathbf{A}_k = \boldsymbol{\Lambda}_k / \lambda_k$ is a diagonal matrix verifying $|\mathbf{A}_k| = 1$.

Interpretation

- Each term in the decomposition

$$\boxed{\boldsymbol{\Sigma}_k = \lambda_k \boldsymbol{D}_k \mathbf{A}_k \boldsymbol{D}_k^T}$$

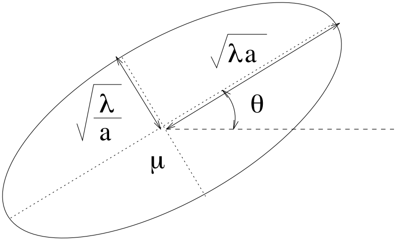
has a simple interpretation:

  - $\mathbf{A}_k$ describes the shape of the cluster (defined by the ratios of the eigenvalues of $\boldsymbol{\Sigma}_k$)
  - $\boldsymbol{D}_k$ (a rotation matrix) describes its orientation
  - λ_k describes its volume

- Number of parameters:

$\boldsymbol{\Sigma}_k$	λ_k	$\mathbf{A}_k$	$\boldsymbol{D}_k$
$\frac{p(p+1)}{2}$	1	$p-1$	$\frac{p(p-1)}{2}$

Example in $\mathbb{R}^2$



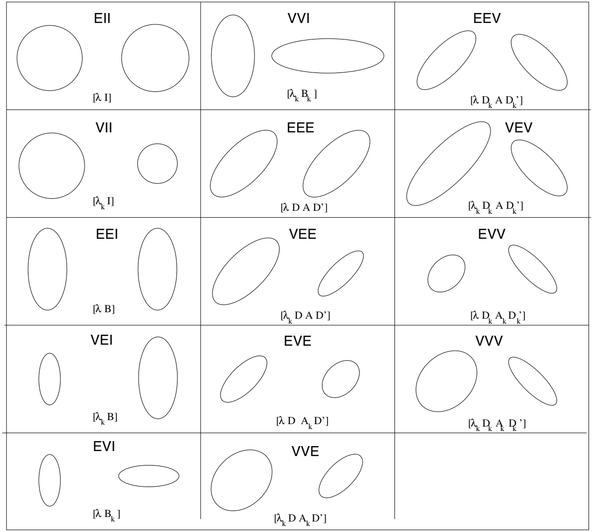
$$A = \begin{bmatrix} a & 0 \\ 0 & 1/a \end{bmatrix} \qquad D = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\,\theta & \cos\theta \end{bmatrix}$$

- $\boldsymbol{D}$: rotation matrix, angle θ
- $\mathbf{A}$: diagonal matrix with diagonal terms a and $1/a$
- The eigenvalues of $\boldsymbol{\Sigma}$ are λa and λ/a.

Parsimonious models

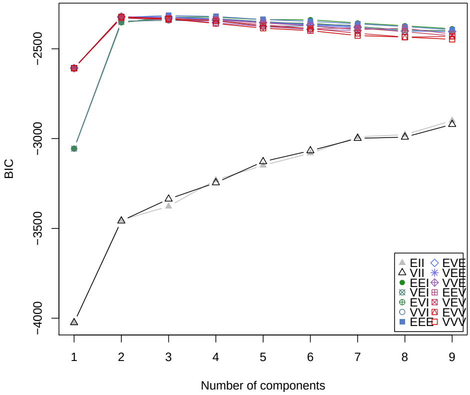
- With this parametrization, the parameters of the GMM are: the centers, volumes, shapes, orientations and proportions.
- 28 different models:
  - Spherical, diagonal, arbitrary
  - Volumes equal or not
  - Shapes equal or not
  - Orientations equal or not
  - Proportions equal or not

The 14 models based on assumptions on variance matrices



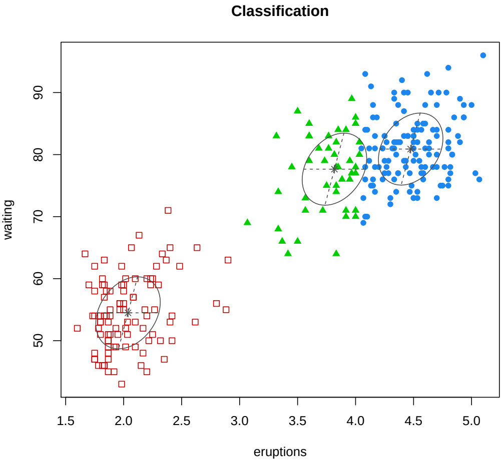
Parsimonious models in mclust

```
faithfulMclust <- Mclust(faithful)
plot(faithfulMclust)
```



Best model

Best model: EEE or λDAD^T (ellipsoidal, equal volume, shape and orientation) model with 3 components



Overview

Semi-supervised learning I

- In semi-supervised learning, the data have the form

$$\mathcal{L}_{ss} = \underbrace{\{(x_i, y_i)\}_{i=1}^{n_s}}_{\text{labeled part}} \cup \underbrace{\{x_i\}_{i=n_s+1}^{n}}_{\text{unlabeled part}}$$

- Observed-data likelihood:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n_s} p(x_i, y_i; \theta) \prod_{i=n_s+1}^{n} p(x_i; \theta) \\ &= \left(\prod_{i=1}^{n_s} \prod_{k=1}^{c} \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k)^{y_{ik}} \pi_k^{y_{ik}} \right) \left(\prod_{i=n_s+1}^{n} \sum_{k=1}^{c} \pi_k \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k) \right) \end{aligned}$$

Semi-supervised learning II

- Complete-data likelihood:

$$L_c(\theta) = \prod_{i=1}^{n}\prod_{k=1}^{c}\phi(x_i;\mu_k,\mathbf{\Sigma}_k)^{y_{ik}}\pi_k^{y_{ik}}$$
$$= \underbrace{\prod_{i=1}^{n_s}\prod_{k=1}^{c}\phi(x_i;\mu_k,\mathbf{\Sigma}_k)^{y_{ik}}\pi_k^{y_{ik}}}_{\text{observed}}\underbrace{\prod_{i=n_s+1}^{n}\prod_{k=1}^{c}\phi(x_i;\mu_k,\mathbf{\Sigma}_k)^{y_{ik}}\pi_k^{y_{ik}}}_{\text{non-observed}}$$

- Complete-data log-likelihood:

$$\ell_c(\theta) = \sum_{i=1}^{n_s}\sum_{k=1}^{c} y_{ik}(\log\phi(x_i;\mu_k,\mathbf{\Sigma}_k)+\log\pi_k)+$$
$$\sum_{i=n_s+1}^{n}\sum_{k=1}^{c} y_{ik}(\phi(x_i;\mu_k,\mathbf{\Sigma}_k)+\log\pi_k)$$

Semi-supervised learning III

- Q function:

$$Q(\theta, \theta^{(t)}) = \sum_{i=1}^{n_s} \sum_{k=1}^{c} y_{ik} (\log \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k) + \log \pi_k) + \sum_{i=n_s+1}^{n} \sum_{k=1}^{c} y_{ik}^{(t)} (\log \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k) + \log \pi_k)$$

$$= \sum_{k=1}^{c} \sum_{i=1}^{n} y_{ik}^{(t)} \log \phi(x_i; \mu_k, \boldsymbol{\Sigma}_k) + \sum_{i=1}^{n} \sum_{k=1}^{c} y_{ik}^{(t)} \log \pi_k$$

with

$$y_{ik}^{(t)} = \begin{cases} y_{ik} & i = 1, \ldots, n_s \\ \mathbb{E}_{\theta^{(t)}}[Y_{ik} \mid x_i] & i = n_s + 1, \ldots, n \end{cases}$$

EM algorithm

E-step: Compute

$$y_{ik}^{(t)} = \begin{cases} y_{ik} & i = 1, \ldots, n_s \text{ (fixed)} \\ \frac{\phi(x_i; \mu_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}) \pi_k^{(t)}}{\sum_{\ell=1}^{c} \phi(x_i; \mu_\ell^{(t)}, \boldsymbol{\Sigma}_\ell^{(t)}) \pi_\ell^{(t)}} & i = n_s + 1, \ldots, n \end{cases}$$

M-step: Same as in the unsupervised case.

$$\pi_k^{(t+1)} = \frac{n_k^{(t)}}{n}, \quad \mu_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{n} y_{ik}^{(t)} x_i$$

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{n} y_{ik}^{(t)} (x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T$$

with $n_k^{(t)} = \sum_{i=1}^{n} y_{ik}^{(t)}$

Overview

Mixture Discriminant Analysis

- GMM can also be useful in supervised classification.
- Here, we model the distribution of X in each class by a GMM:

$$p(x \mid Y=k) = \sum_{r=1}^{R_k} \pi_{kr} \phi(x; \mu_{kr}, \mathbf{\Sigma}_{kr})$$

  with $\sum_{r=1}^{R_k} \pi_{kr} = 1$.
- This method is called Mixture Discriminant Analysis (MDA). It extends LDA.
- By varying the number of components in each mixture, we can handle classes of any shape, and obtain arbitrarily complex nonlinear decision boundaries.
- We may impose $\mathbf{\Sigma}_{kr} = \mathbf{\Sigma}$, $\mathbf{\Sigma}_{kr} = \sigma_{kr}\mathbf{I}$, or any other parsimonious model, to control the complexity of the model.

Observed-data likelihood

- Observed-data likelihood:

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{n} p(x_i, y_i; \theta) = \prod_{i=1}^{n} p(x_i \mid y_i; \theta) p(y_i; \theta) \\
&= \prod_{i=1}^{n} \prod_{k=1}^{c} \left(\sum_{r=1}^{R_k} \pi_{kr} \phi(x; \mu_{kr}, \boldsymbol{\Sigma}_{kr}) \right)^{y_{ik}} \pi_k^{y_{ik}}
\end{aligned}
$$

- Observed-data log-likelihood:

$$
\ell(\theta) = \sum_{k=1}^{c} \sum_{i=1}^{n} y_{ik} \log \left(\sum_{r=1}^{R_k} \pi_{kr} \phi(x; \mu_{kr}, \boldsymbol{\Sigma}_{kr}) \right) + \sum_{k=1}^{c} \sum_{i=1}^{n} y_{ik} \log \pi_k
$$

- Again, the EM algorithm can be used to estimate the model parameters (see ESL pp. 399-402 for details).

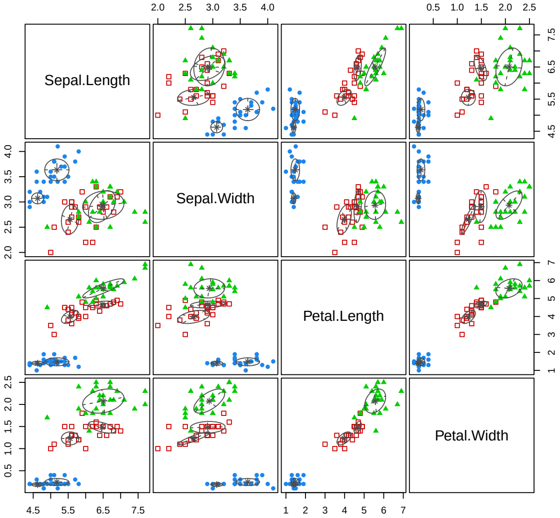
MDA using package mclust: Iris data

```
odd <- seq(from = 1, to = nrow(iris), by = 2)
even <- odd + 1
X.train <- iris[odd,-5]
Class.train <- iris[odd,5]
X.test <- iris[even,-5]
Class.test <- iris[even,5]

# general covariance structure selected by BIC
irisMclustDA <- MclustDA(X.train, Class.train)
summary(irisMclustDA, newdata = X.test, newclass = Class.test)

plot(irisMclustDA)
```

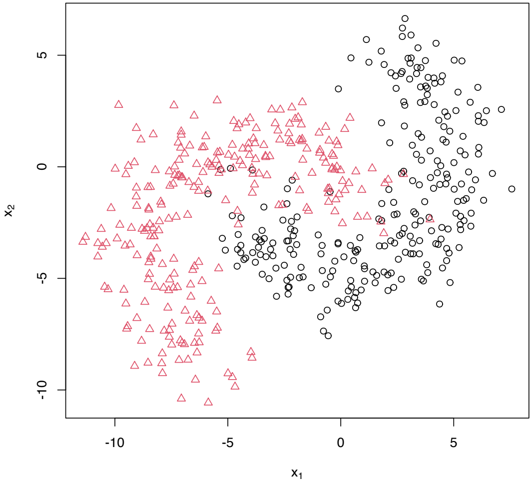
Result

```
> summary(irisMclustDA, newdata = X.test, newclass = Class.test)
------------------------------------------------
Gaussian finite mixture model for classification
------------------------------------------------

MclustDA model summary:

 log.likelihood  n df       BIC
      -63.55015 75 53 -355.9272

Classes        n Model G
  setosa      25   VEI 2
  versicolor 25   EEV 2
  virginica  25   XXX 1

Training classification summary:

            Predicted
Class         setosa versicolor virginica
  setosa          25          0         0
  versicolor       0         25         0
  virginica        0          0        25

Training error = 0

Test classification summary:

            Predicted
Class         setosa versicolor virginica
  setosa          25          0         0
  versicolor       0         24         1
  virginica        0          0        25

Test error = 0.01333333
```

Result


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0 2.5 3.0 3.5 4.0
0.5 1.0 1.5 2.0 2.5
4.5 5.5 6.5 7.5
1 2 3 4 5 6 7

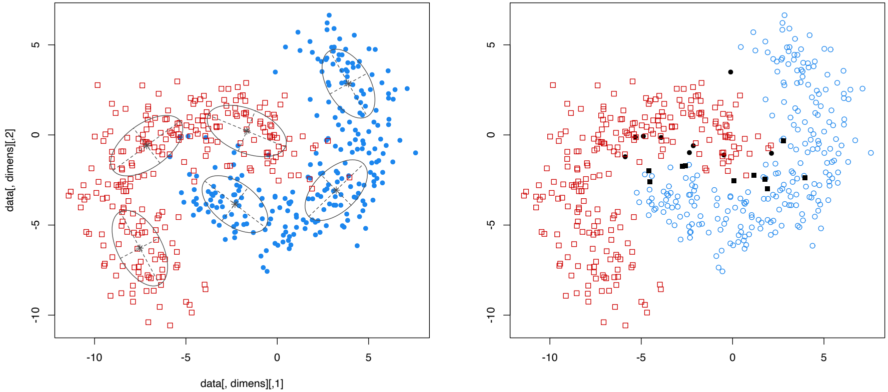
MDA using package `mclust`: Bananas data



Result

```
> summary(res, newdata = data.test$x, newclass = data.test$y)
------------------------------------------------
Gaussian finite mixture model for classification
------------------------------------------------

MclustDA model summary:

 log-likelihood   n df       BIC
      -2633.035 500 26 -5427.649

Classes   n  % Model G
      1 250 50   EEV 3
      2 250 50   EEV 3

Training confusion matrix:
     Predicted
Class   1   2
    1 241   9
    2  10 240
Classification error = 0.038
Brier score          = 0.0306

Test confusion matrix:
     Predicted
Class   1   2
    1 471  29
    2  18 482
Classification error = 0.047
Brier score          = 0.0378
```

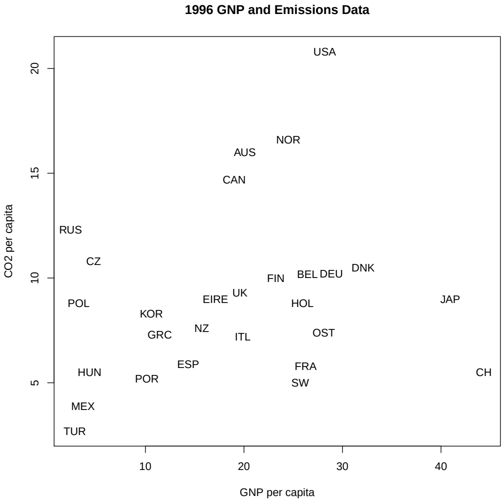
Result



Overview

Overview

Introductory example



Introductory example (continued)

- The data in the previous slide do not show any clear linear trend.
- However, there seem to be several groups for which a linear model would be a reasonable approximation.
- How to identify those groups and the corresponding linear models?

Formalization

- We assume that the response variable Y depends on the input variable X in different ways, depending on a latent variable Z. (Beware: we have switched back to the classical notation for regression models!)
- This model is called mixture of regressions or switching regressions. It has been widely studied in the econometrics literature.

Model

- Model:
$$Y = \begin{cases} \beta_1^T X + \epsilon_1,\ \epsilon_1 \sim \mathcal{N}(0, \sigma_1) & \text{if } Z = 1, \\ \vdots & \vdots \\ \beta_c^T X + \epsilon_c,\ \epsilon_c \sim \mathcal{N}(0, \sigma_c) & \text{if } Z = c, \end{cases}$$
with $X = (1, X_1, \ldots, X_p)$, and
$$\mathbb{P}(Z = k) = \pi_k, \quad k = 1, \ldots, c.$$
- So, the marginal pdf of Y is
$$p(y \mid X = x) = \sum_{k=1}^{c} \pi_k \phi(y; \beta_k^T x, \sigma_k)$$

Observed and complete-data likelihoods

- Observed-data likelihood:

$$L(\theta) = \prod_{i=1}^{n} p(y_i; \theta) = \prod_{i=1}^{n} \sum_{k=1}^{c} \pi_k \phi(y_i; \beta_k^T x_i, \sigma_k)$$

- Complete-data likelihood:

$$\begin{aligned} L_c(\theta) &= \prod_{i=1}^{n} p(y_i, z_i; \theta) = \prod_{i=1}^{n} p(y_i \mid z_i; \theta) p(z_i \mid \pi) \\ &= \prod_{i=1}^{n} \prod_{k=1}^{c} \phi(y_i; \beta_k^T x_i, \sigma_k)^{z_{ik}} \pi_k^{z_{ik}}, \end{aligned}$$

with $z_{ik} = I(z_i = k)$.

Derivation of function Q

- Complete-data log-likelihood:

$$\ell_c(\theta) = \sum_{i=1}^{n}\sum_{k=1}^{c} z_{ik} \log \phi(y_i; \beta_k^T x_i, \sigma_k) + \sum_{i=1}^{n}\sum_{k=1}^{c} z_{ik} \log \pi_k$$

- It is linear in the z_{ik}. Consequently, the Q function is simply

$$Q(\theta, \theta^{(t)}) = \sum_{k=1}^{c} \underbrace{\sum_{i=1}^{n} z_{ik}^{(t)} \log \phi(y_i; \beta_k^T x_i, \sigma_k)}_{\text{term depending on } \beta_k \text{ and } \sigma_k} + \underbrace{\sum_{i=1}^{n}\sum_{k=1}^{c} z_{ik}^{(t)} \log \pi_k}_{\text{term depending on } \{\pi_k\}}$$

with $z_{ik}^{(t)} = \mathbb{E}_{\theta^{(t)}}[Z_{ik} \mid y_i] = \mathbb{P}_{\theta^{(t)}}[Z_i = k \mid y_i]$.

EM algorithm

E-step: Compute

$$
\begin{aligned}
z_{ik}^{(t)} &= \mathbb{P}_{\theta^{(t)}}[Z_i = k \mid y_i] \\
&= \frac{\phi(y_i; \beta_k^{(t)T} x_i, \sigma_k^{(t)})\pi_k^{(t)}}{\sum_{\ell=1}^{c} \phi(y_i; \beta_\ell^{(t)T} x_i, \sigma_\ell^{(t)})\pi_\ell^{(t)}}
\end{aligned}
$$

M-step: Maximize $Q(\theta, \theta^{(t)})$. As before, we get

$$
\pi_k^{(t+1)} = \frac{n_k^{(t)}}{n},
$$

with $n_k^{(t)} = \sum_{i=1}^{n} z_{ik}^{(t)}$.

M-step: update of the β_k and σ_k I

- In $Q(\theta,\theta^{(t)})$, the term depending on β_k is

$$\sum_{i=1}^n z_{ik}^{(t)} \log\phi(y_i;\beta_k^T x_i,\sigma_k) = \sum_{i=1}^n z_{ik}^{(t)}\left[-\frac{\log(2\pi\sigma_k^2)}{2} - \frac{1}{2\sigma_k^2}(y_i-\beta_k^T x_i)^2\right]$$

$$= -\frac{1}{2\sigma_k^2}\underbrace{\sum_{i=1}^n z_{ik}^{(t)}(y_i-\beta_k^T x_i)^2}_{SS_k} - \frac{n_k^{(t)}\log(2\pi\sigma_k^2)}{2}$$

with $n_k^{(t)} = \sum_{i=1}^n z_{ik}^{(t)}$.

M-step: update of the β_k and σ_k II

- Minimizing SS_k w.r.t. β_k is a weighted least-squares (WLS) problem. In matrix form,

$$SS_k = (\mathbf{y} - \mathbf{X}\beta_k)^T \mathbf{W}_k^{(t)} (\mathbf{y} - \mathbf{X}\beta_k),$$

  where $\mathbf{W}_k^{(t)} = \text{diag}(z_{1k}^{(t)}, \ldots, z_{nk}^{(t)})$ is a diagonal matrix of size n.
- The solution is the WLS estimate of β_k:

$$\boxed{\beta_k^{(t+1)} = (\mathbf{X}^T \mathbf{W}_k^{(t)} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}_k^{(t)} \mathbf{y}}$$

M-step: update of the β_k and σ_k III

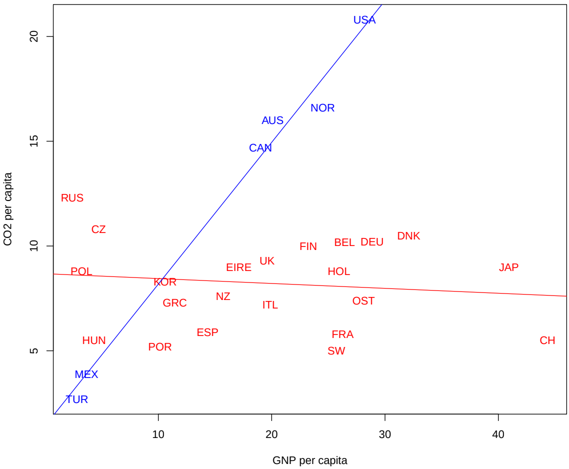
- Plugging in the estimate $\beta_k^{(t+1)}$ in the expression of the Q function and differentiating with respect to σ_k, we obtain the value of σ_k minimizing $Q(\theta, \theta^{(t)})$ as the average of the residuals weighted by the $z_{ik}^{(t)}$:

$$\begin{aligned}\sigma_k^{2(t+1)} &= \frac{1}{n_k^{(t)}} \sum_{i=1}^{n} z_{ik}^{(t)} (y_i - \beta_k^{(t+1)T} x_i)^2 \\ &= \frac{1}{n_k^{(t)}} (\mathbf{y} - \mathbf{X}\beta_k^{(t+1)})^T \mathbf{W}_k^{(t)} (\mathbf{y} - \mathbf{X}\beta_k^{(t+1)})\end{aligned}$$

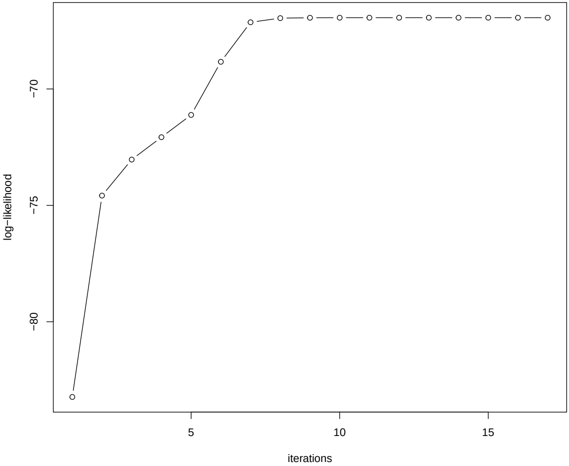
Mixture of regressions using mixtools

```
library(mixtools)
data(CO2data)
attach(CO2data)

CO2reg <- regmixEM(CO2, GNP)
summary(CO2reg)

ii1<-CO2reg$posterior>0.5
ii2<-CO2reg$posterior<=0.5
text(GNP[ii1],CO2[ii1],country[ii1],col='red')
text(GNP[Cii2],CO2[ii2],country[ii2],col='blue')
abline(CO2reg$beta[,1],col='red')
abline(CO2reg$beta[,2],col='blue')
```

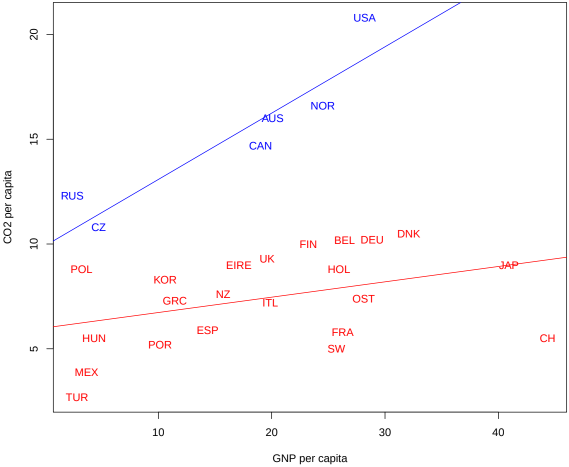
Best solution in 10 runs



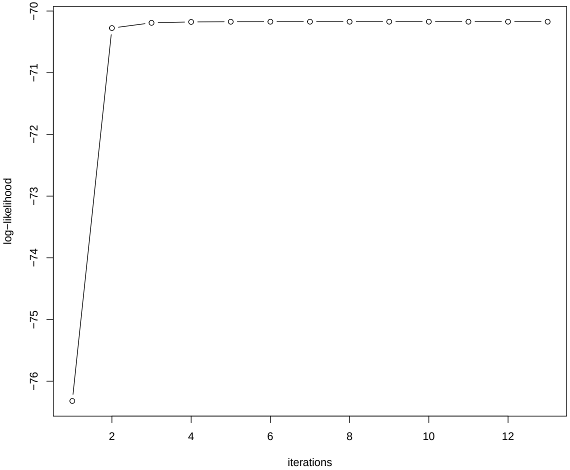
Increase of log-likelihood



Another solution (with lower log-likelihood)



Increase of log-likelihood



Overview

Making the mixing proportions predictor-dependent

- An interesting extension of the previous model is to assume the proportions π_k to be partially explained by a vector of concomitant variables W.
- If $W = X$, we can approximate the regression function by different linear functions in different regions of the predictor space.
- In ML, this method is referred to as the mixture of experts method.
- A useful parametric form for π_k that ensures $\pi_k \geq 0$ and $\sum_{k=1}^{c} \pi_k = 1$ is the multinomial logit (softmax) model:

$$\pi_k(w, \alpha) = \frac{\exp(\alpha_k^T w)}{\sum_{l=1}^{c} \exp(\alpha_l^T w)}$$

with $\alpha = (\alpha_1, \ldots, \alpha_c)$ and $\alpha_1 = 0$.

EM algorithm

- The Q function is the same as before, except that the π_k now depend on the w_i and parameter α:

$$Q(\theta, \theta^{(t)}) = \sum_{i=1}^{n} \sum_{k=1}^{c} z_{ik}^{(t)} \log \phi(y_i; \beta_k^T x_i, \sigma_k) + \sum_{i=1}^{n} \sum_{k=1}^{c} z_{ik}^{(t)} \log \pi_k(w_i, \alpha)$$

- In the M-step, the update formula for β_k and σ_k are unchanged.
- The last term of $Q(\theta, \theta^{(t)})$ can be maximized w.r.t. α using an iterative algorithm, such as the Newton-Raphson procedure. (See remark on next slide)

Generalized EM algorithm

- To ensure the convergence of EM, we only need, at the M step of each iteration t, to find an estimate $\theta^{(t+1)}$ such that

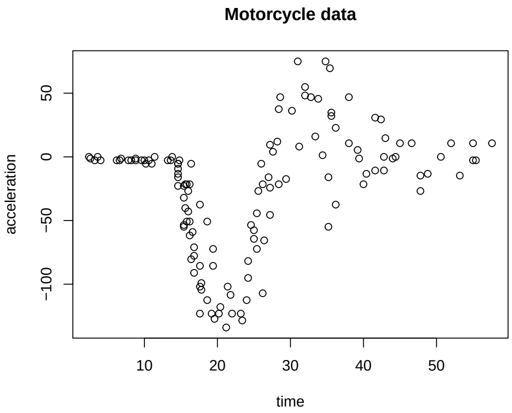
$$\boxed{Q(\theta^{(t+1)}, \theta^{(t)}) \geq Q(\theta^{(t)}, \theta^{(t)})}$$

- Any algorithm that chooses $\theta^{(t+1)}$ at each iteration to guarantee the above condition (without maximizing $Q(\theta, \theta^{(t)})$) is called a Generalized EM (GEM) algorithm.
- Here, we can perform a single step of the Newton-Raphson algorithm to maximize

$$\sum_{i=1}^{n} \sum_{k=1}^{c} z_{ik}^{(t)} \log \pi_k(w_i, \alpha)$$

with respect to α.
- Backtracking can be used to ensure ascent.

Example: motorcycle data




```
library('MASS')
x<-mcycle$times
y<-mcycle$accel
plot(x,y)
```

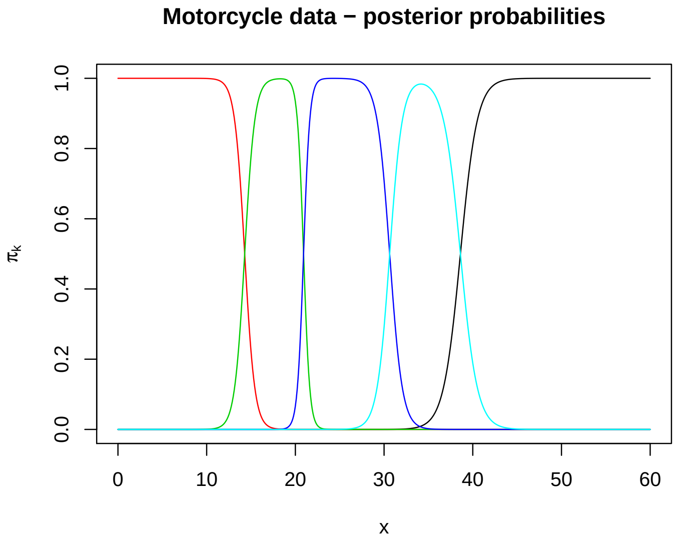
Mixture of experts using `flexmix`

```
library(flexmix)

K<-5
res<-flexmix(y ~ x,k=K,model=FLXMRglm(family="gaussian"),
concomitant=FLXPmultinom(formula=~x))

beta<- parameters(res)[1:2,]
alpha<-res@concomitant@coef
```

Plotting the posterior probabilities

```
xt<-seq(0,60,0.1)
Nt<-length(xt)
plot(x,y)
pit=matrix(0,Nt,K)
for(k in 1:K) pit[,k]<-exp(alpha[1,k]+alpha[2,k]*xt)
pit<-pit/rowSums(pit)

plot(xt,pit[,1],type="l",col=1)
for(k in 2:K) lines(xt,pit[,k],col=k)
```

Posterior probabilities



Plotting the predictions

```
yhat<-rep(0,Nt)
for(k in 1:K) yhat<-yhat+pit[,k]*(beta[1,k]+beta[2,k]*xt)

plot(x,y,main="Motorcycle data",xlab="time",ylab="acceleration")
for(k in 1:K) abline(beta[1:2,k],lty=2)
lines(xt,yhat,col='red',lwd=2)
```

Regression lines and predictions