Advanced Data Analytics & Machine Learning

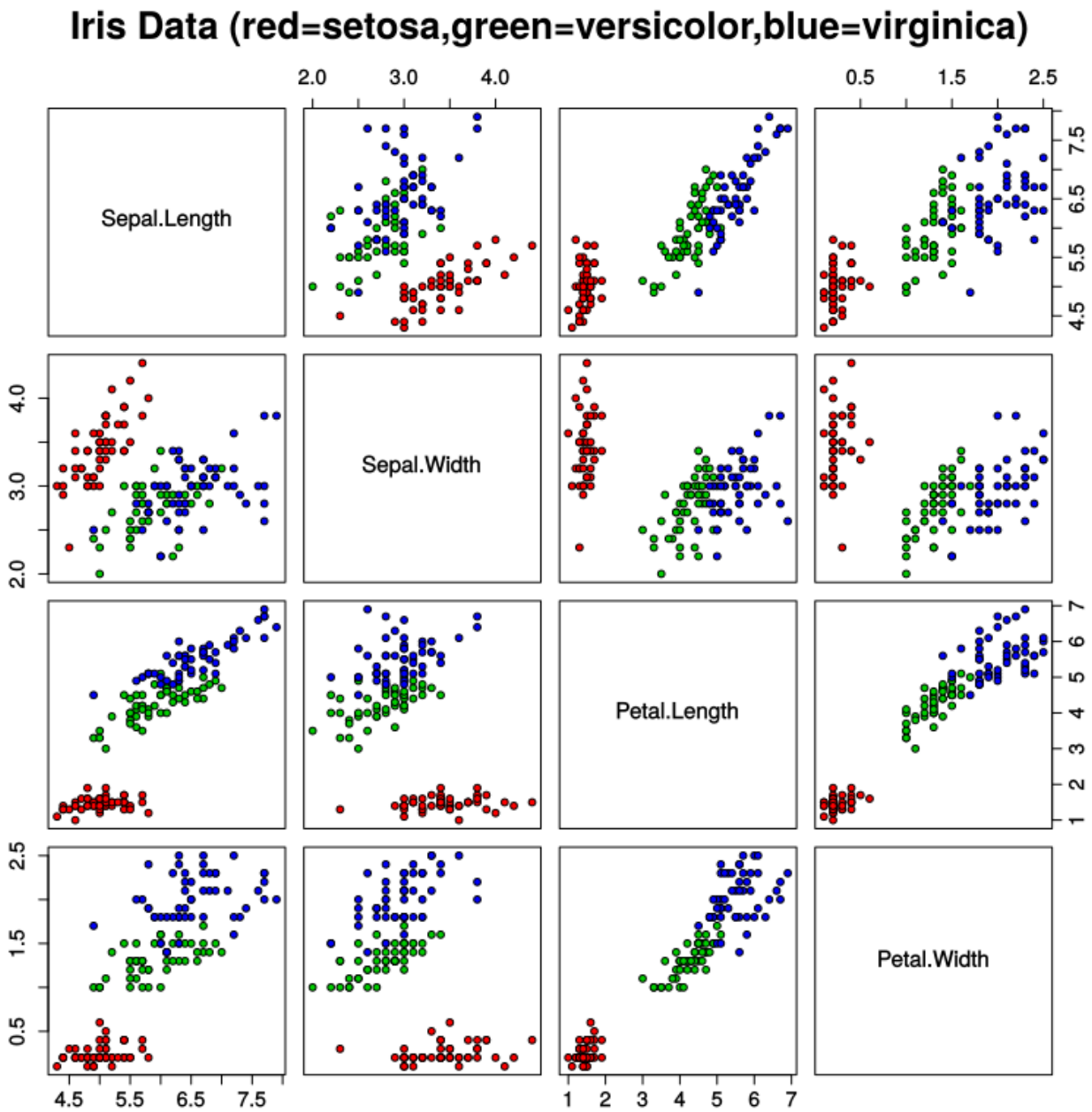
M2 Statistics & Data Science / Informatique

Faïcel Chamroukhi

https://chamroukhi.com/

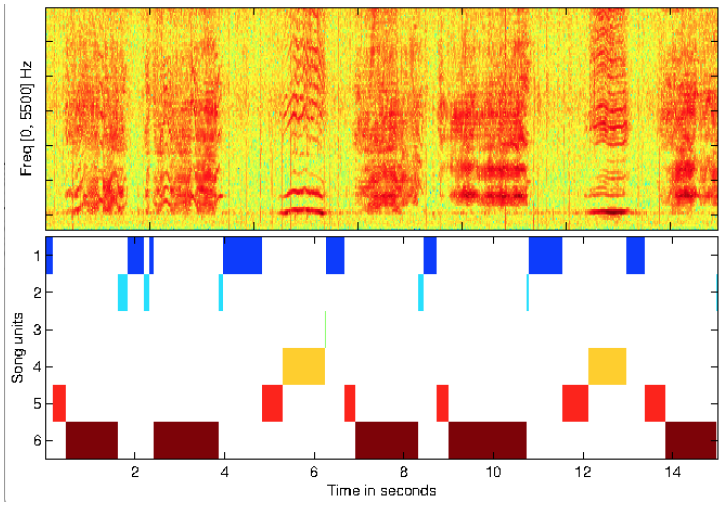
⚠ Remarque : Cette version qui est disponible sur ecampus évoluera continuellement, penser à toujours charger une version après chaque séance

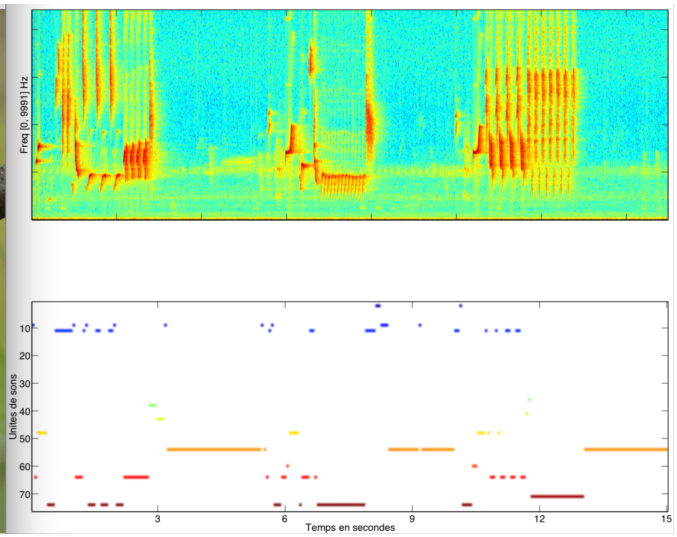
Données Iris : Les Iris est un jeu de données présenté par Ronald Fisher en 1936 et qui consiste en des données multivariées qui ont été collectées afin de quantifier les variations de morphologie des fleurs d'iris de trois espèces : Iris setosa, Iris virginica et Iris versicolor. Le jeu de données comprend 50 observations de chacune des trois espèces d'iris. Quatre caractéristiques ont été mesurées pour chaque observation : la longueur et la largeur des sépales et des pétales, en centimètres. Sur la base de la combinaison de ces quatre variables, Fisher a élaboré un modèle d'analyse permettant de distinguer les espèces les unes des autres.


Iris Data (red=setosa,green=versicolor,blue=virginica)
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width

Comprendre la communication chez des espèces animales

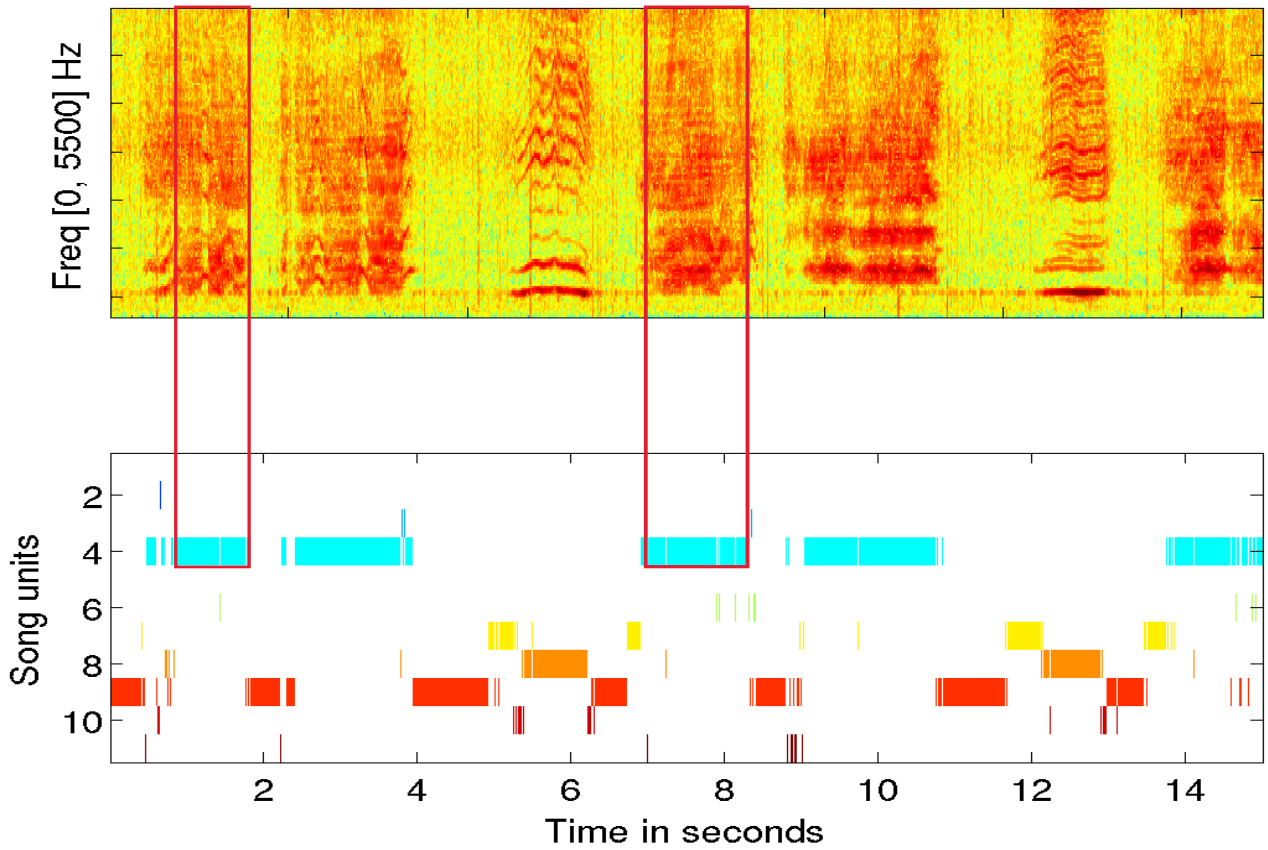
Apporter une aide au biologistes pour comprendre la communications chez certaines espèces (eg. des mammifères marins, s'ils ont un éventuel alphabet, comment ils migrent, etc)



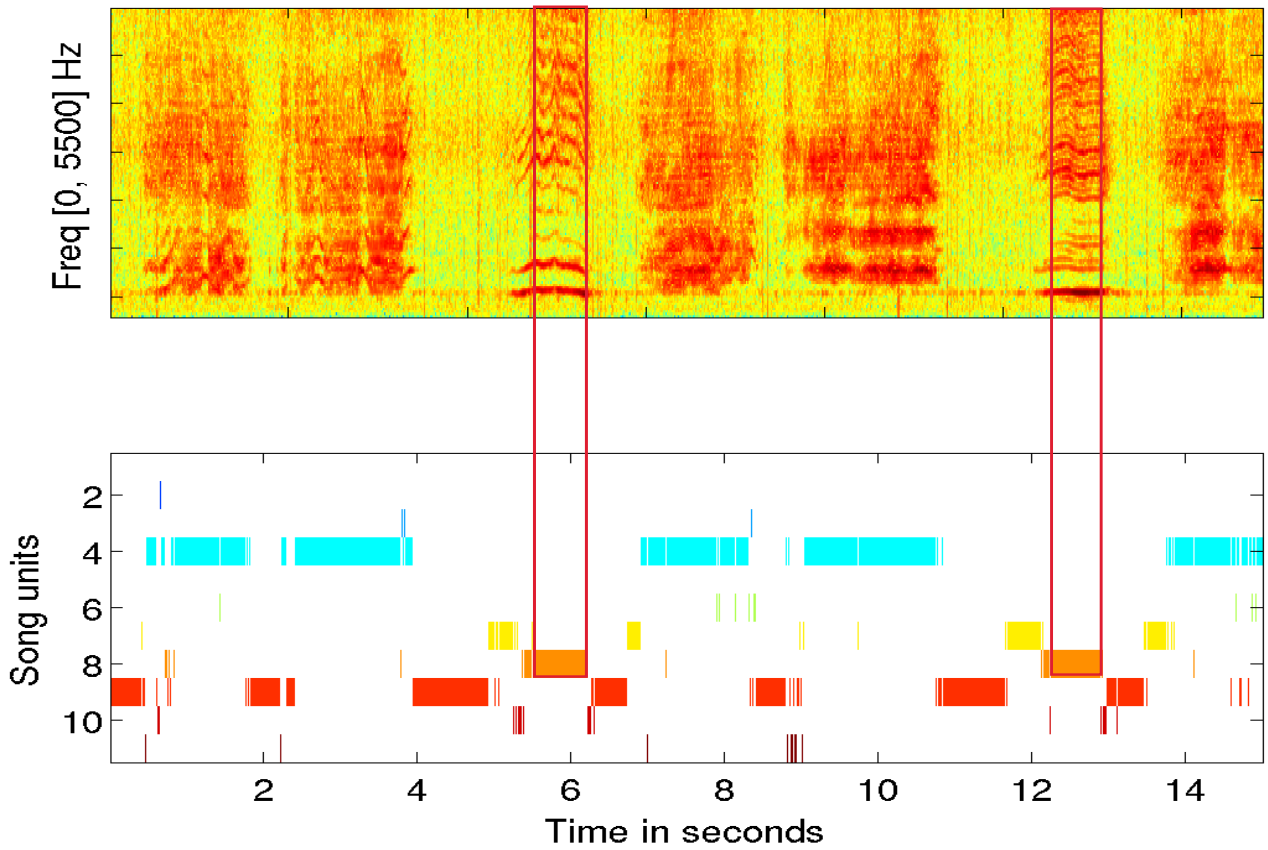


chants de baleine ; Chants de oiseaux

Analyse automatique du chant de baleine

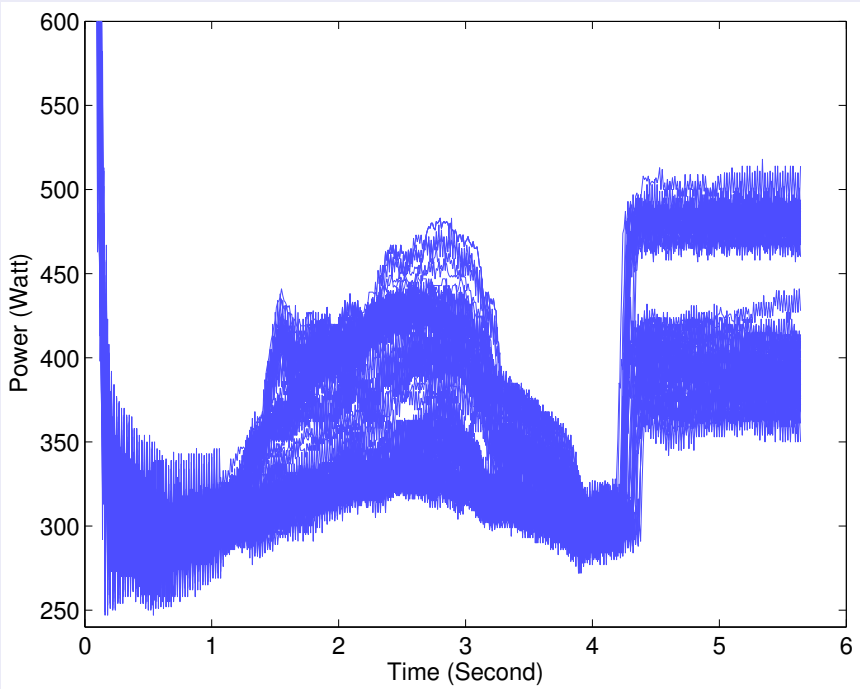


Analyse automatique du chant de baleine

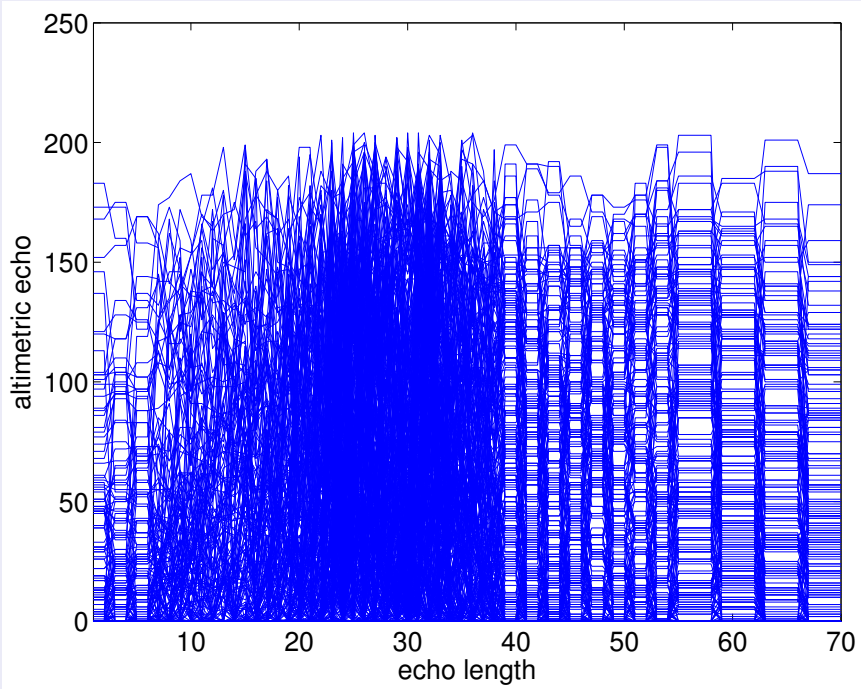


High-dimensional FDA by clustering/segmentation

Non-stationary time series/functions



Railway curves

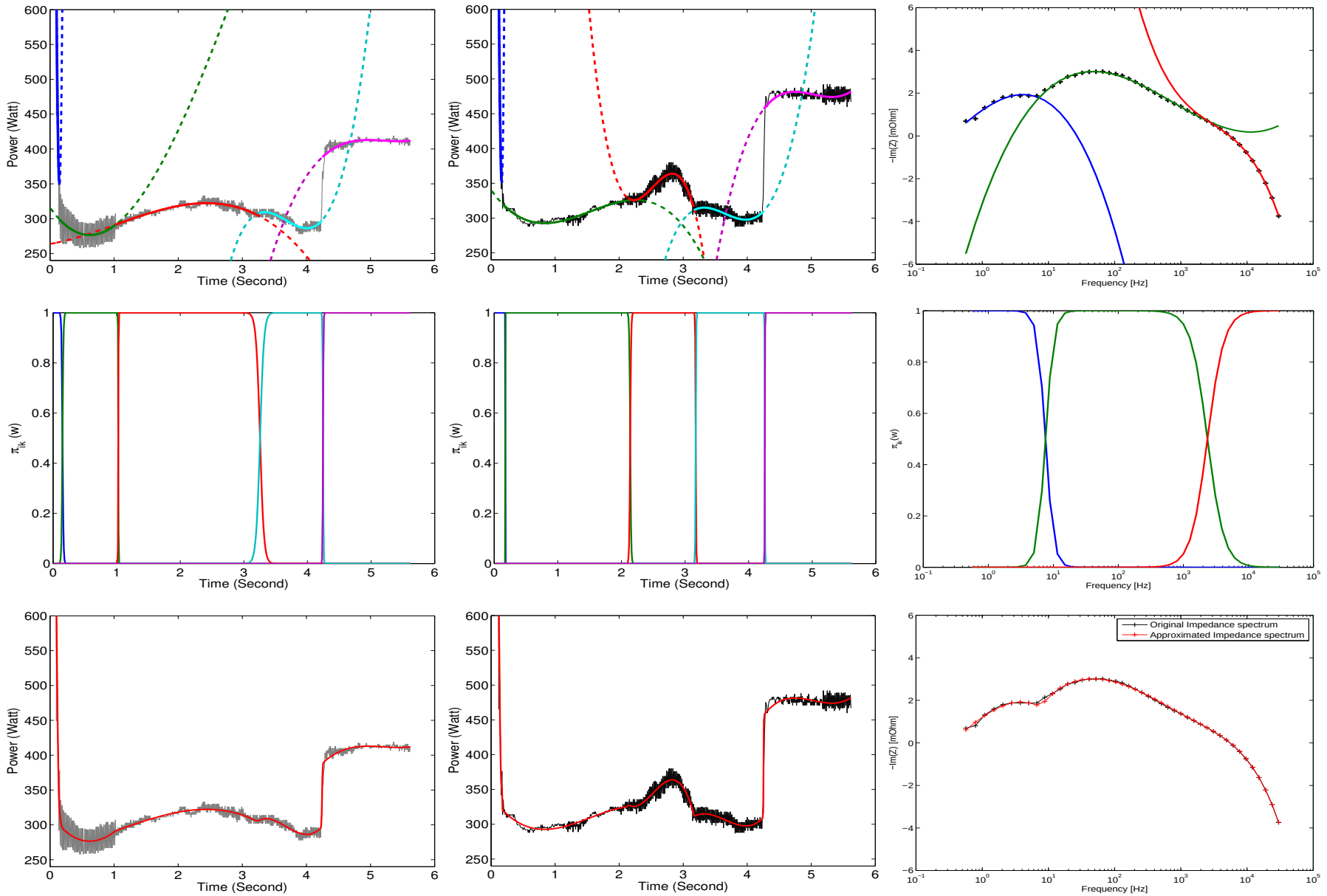


Satellite waveforms

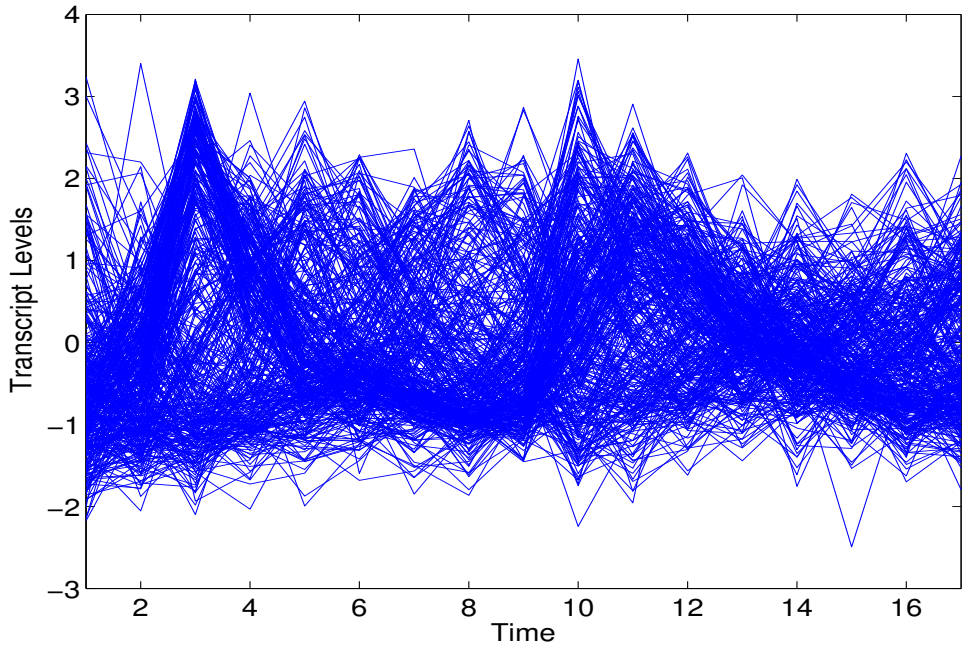
Objectives

- Curve clustering/classification (functional data analysis framework)
- Deal with the problem of regime changes $\hookrightarrow$ Curve segmentation

Application to temporal data modeling and segmentation

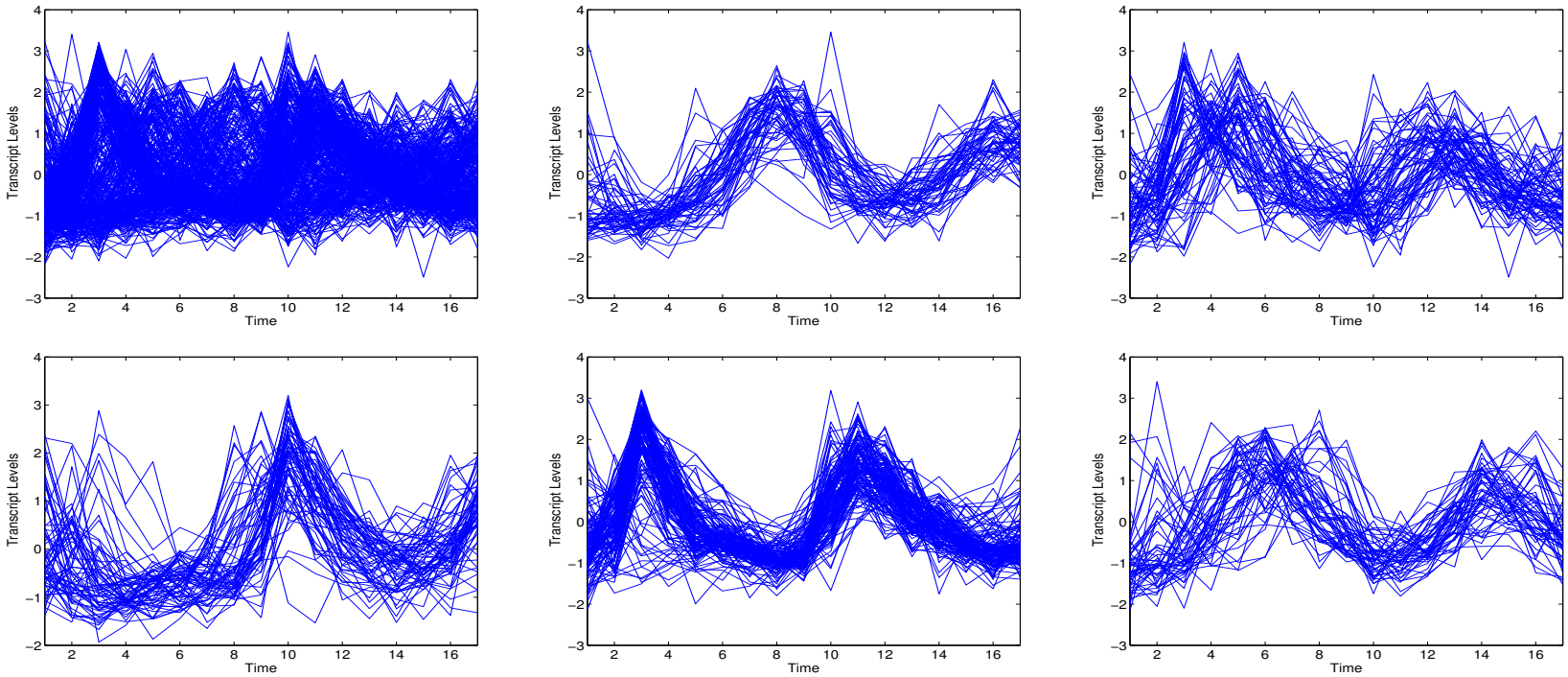


- Données temporelles d'expression génomique (YeungMBC2001 [1], Chamroukhi 2016)
- 384 niveaux d'expression des gènes sur 17 pas de temps.



1. http://faculty.washington.edu/kayee/model/

- Données temporelles d'expression génomique (YeungMBC2001 [2], Chamroukhi 2016)
- 384 niveaux d'expression des gènes sur 17 pas de temps.



2. http://faculty.washington.edu/kayee/model/

EM-like clustering results for yeast cell cycle data

- Time course Gene expression data as in ? [3]
- 384 genes expression levels over 17 time points.

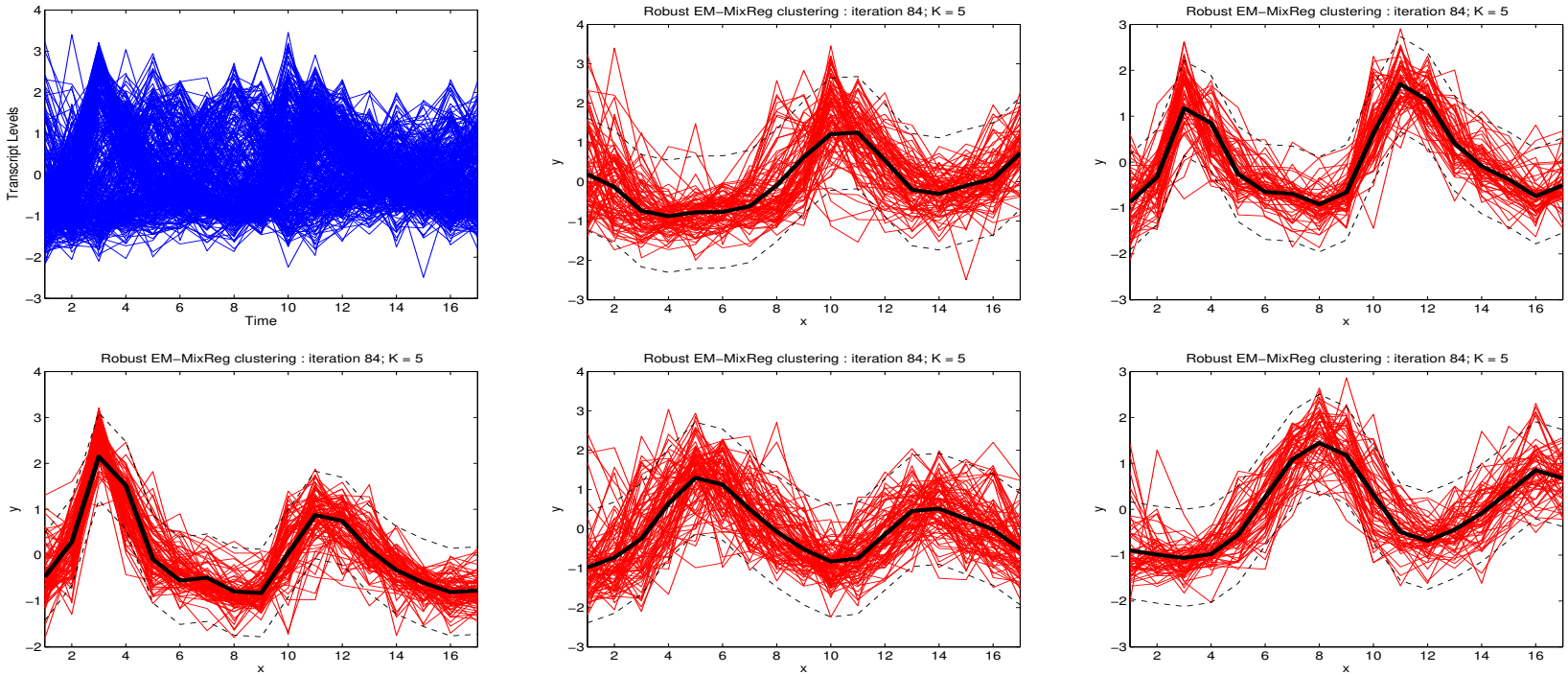
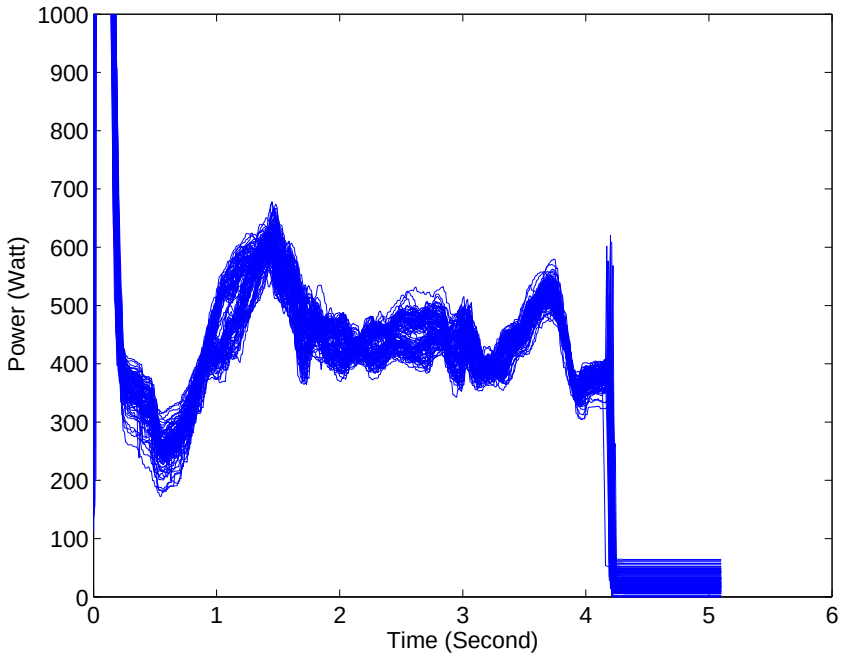


Figure – EM-like clustering results with the bSRM model.

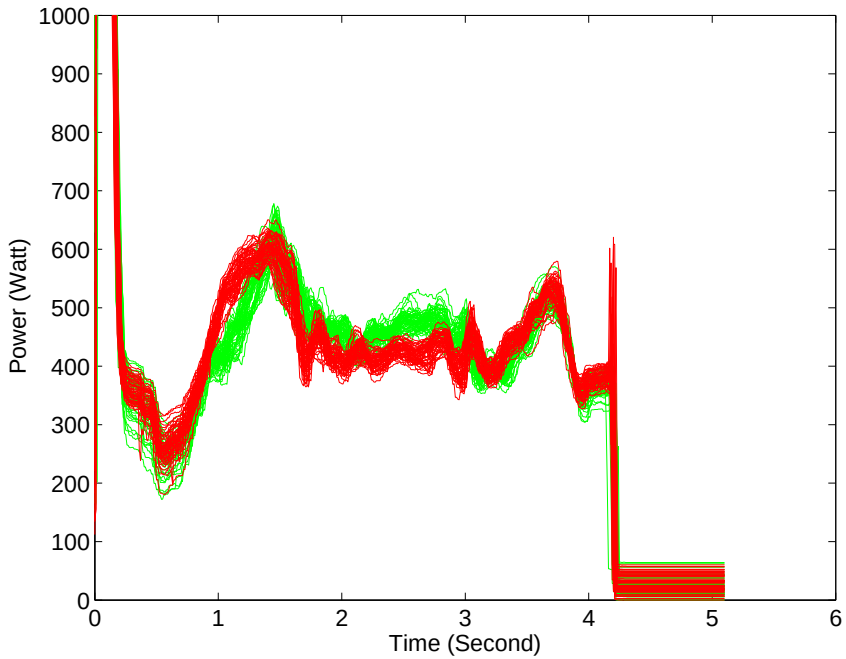
Rand index : 0.7914 which indicates that the partition is quite well defined.

3. http://faculty.washington.edu/kayee/model/

Time-Series clustering



Time-Series clustering

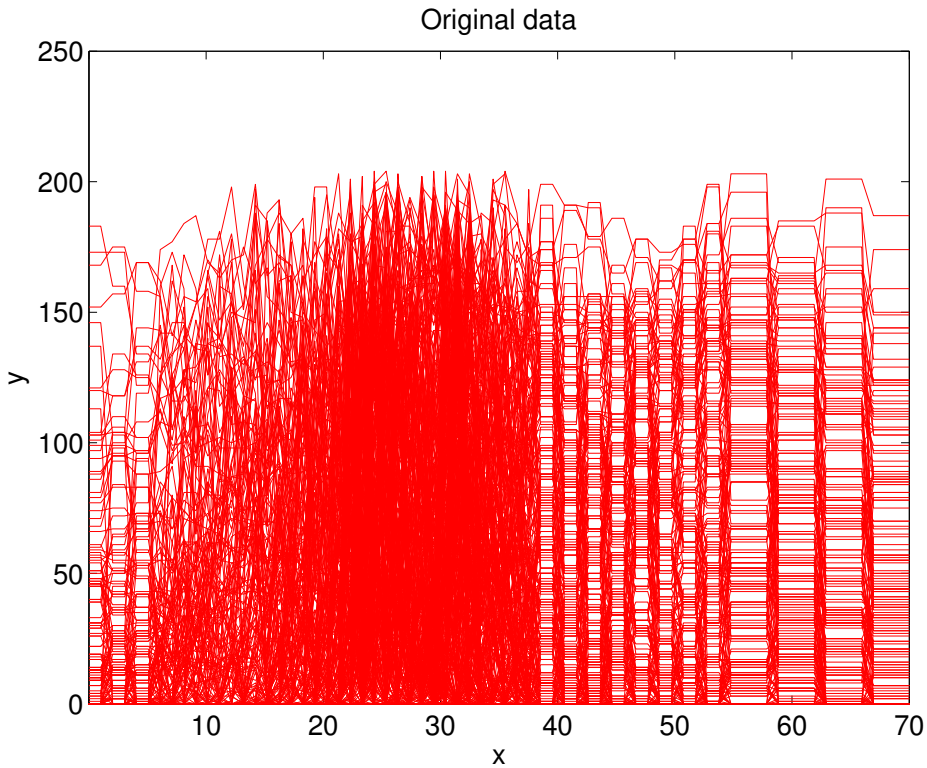


Time series clustering and segmentation

Application to Topex/Poseidon satellite data

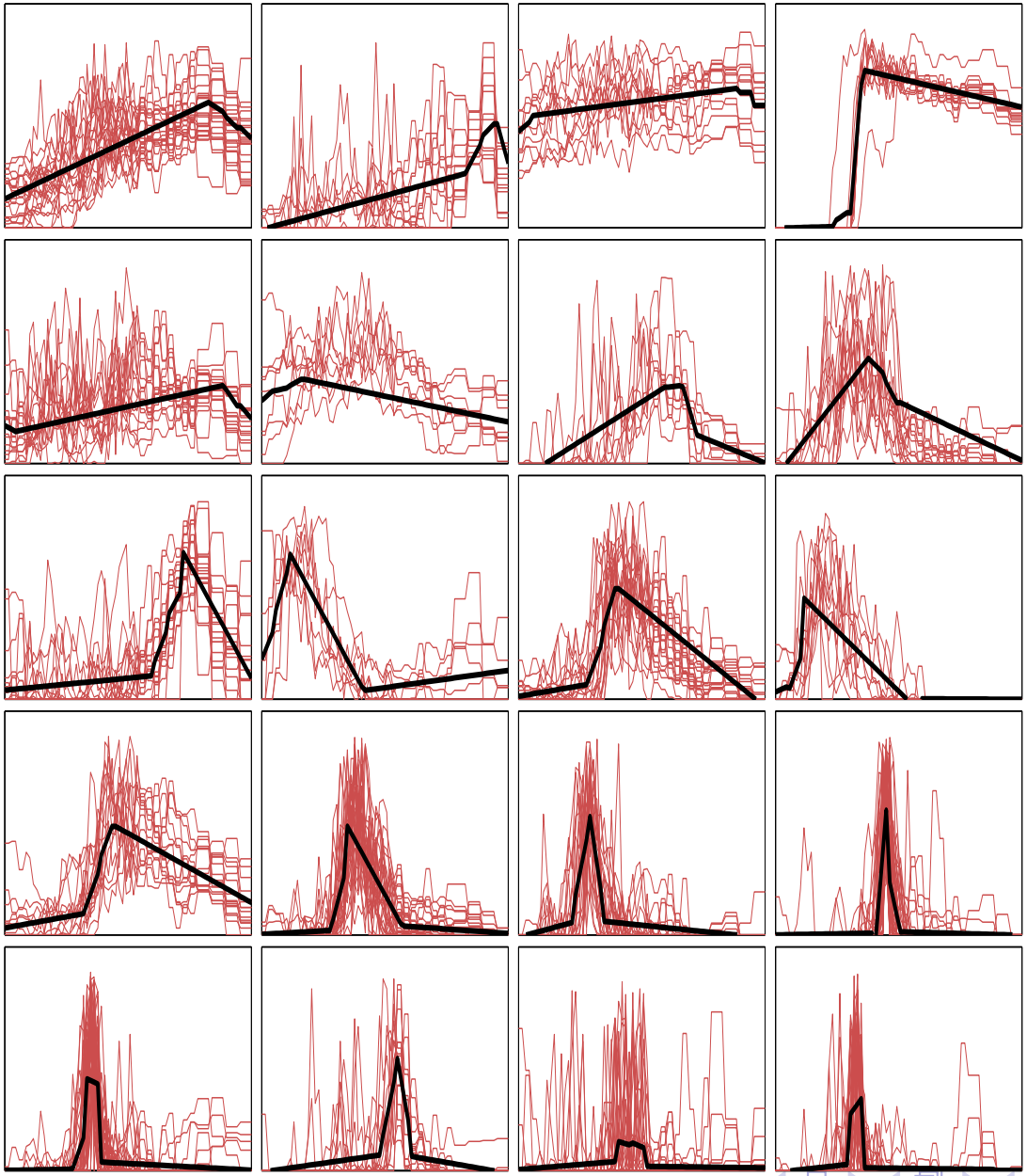
The Topex/Poseidon radar satellite data [4] contains $n = 472$ waveforms of the measured echoes, sampled at $m = 70$ (number of echoes)

We considered the same number of clusters (twenty) and a piecewise linear approximation of four segments per cluster as in **?**.



4. Satellite data are available at `http://www.lsp.ups-tlse.fr/staph/npfda/npfda-datasets.html`.

CEM-PWRM clustering of the satellite data



Cadre général de l'analyse statistique de données

Les statistiques jouent un rôle central en analyse de données

- Permet de quantifier la composante aléatoire dans les données
- Un cadre bien établi pour tenir compte de l'incertitude (cadre probabiliste) et pour établir des méthodes généralisables de prédiction et d'estimation
- Permet une décision souple : par ex. intervalle de confiance en régression et probabilités a posteriori en classification
- Généralement se prête facilement à l'interprétabilité, aide à la compréhension du processus génératif sous-jacent

Cadre général

Analyse statistique de données

- L'analyse statistique des données est la branche scientifique qui permet de convertir des données brutes observées pour un échantillon dans des scénarios réels ou en laboratoire, à travers des modélisations statistiques en les représentant sous forme de variables aléatoires, en des informations et connaissances généralisables à l'échelle de la population étudiée.
- Les scénarios du monde réel sont en effet très souvent perçus avec incertitude, entachés d'incomplétude, corrompus par du bruit, peuvent provenir de problèmes de grande dimension, etc, et une modélisation probabiliste dans laquelle les données sont représentées sous forme de variables aléatoires et de lois de probabilités est ainsi une façon naturelle et de choix de s'y prendre.
- L'approche statistique pour décrire, représenter, et inférer des connaissances interprétables et des règles de décision à partir des données, repose ainsi essentiellement sur la construction et l'inférence de modèles aléatoires pouvant accommoder la nature et la richesse des données brutes observées pour représenter et comprendre au mieux les informations et connaissances qui les composent.

- Ces informations peuvent consister par exemple en des statistiques permettant de synthétiser et résumer les données étudiées, en les décrivant au mieux, de les visualiser, etc, on parle dans ce cas d'approche **descriptive**, ou des modèles ou attributs pertinents construits à représenter au mieux les données originales pour permettre une meilleure prédiction dans des scénarios futurs, on parle dans ce cas d'approche **prédictive**, où des modèles et règles de décision sont inférés à partir des données observées.
- L'approche statistique est aussi généralement celle qui se prête le mieux à l'interprétabilité et l'explicabilité, souvent demandées aussi bien pour des besoins méthodologiques, qu'appliqués, et repose sur un formalisme théorique solide.
- Dans ce cours on se place dans le cadre dans lequel on suppose que les données du problème peuvent être représentées par une variable ou un vecteur aléatoire $\boldsymbol{X}$, ou un couple aléatoire $(\boldsymbol{X}, Y) \in \mathcal{X} \times \mathcal{Y}$ où $\boldsymbol{X}$ est un vecteur de descripteurs potentiellement grand décrivant une certaine variable d'intérêt Y.

  On considère un échantillon aléatoire $(\boldsymbol{X}_i, Y_i)_{i=1,\dots,n}$ et on suppose que l'on dispose d'un échantillon d'observations (ensemble d'apprentissage) $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1,\dots,n}$.

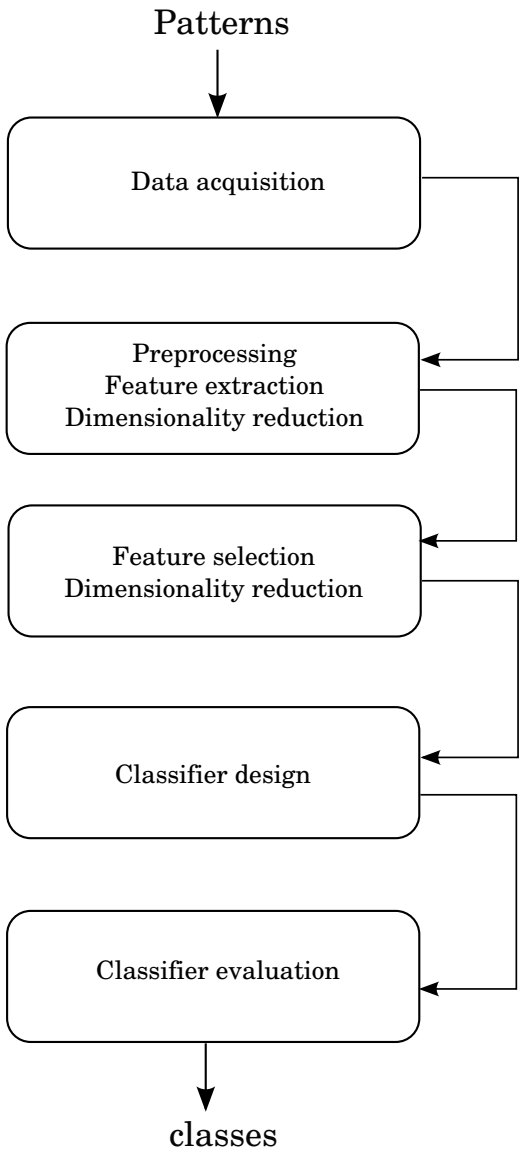
Apprentissage statistique (statistical machine learning)

Ce cours portera sur la formalisation du problème du point de vue de l'**apprentissage statistique**. On y présentera quelques fondamentaux de l'apprentissage avec un focus sur l'estimation statistique paramétrique, en particulier en prédiction, en clustering et en traitement de séquences.
Dans le cas de la **prédiction** l'objectif est de voir comment apprendre un modèle de prédiction $\widehat{\phi} : \mathcal{X} \to \mathcal{Y}$ sur les données d'apprentissage $\mathcal{D}$ pour lequel $\widehat{y} = \widehat{\phi}(\boldsymbol{x})$ est une bonne approximation de la vraie sortie y (au sens d'un certain critère d'optimalité, i.e en minimisant la version empirique d'un risque théorique $R(\phi) = \mathbb{E}\ell\left(Y, \phi(\boldsymbol{X})\right)$ pour une fonction de perte ℓ). C'est le cas des problèmes de **régression** et de **classification supervisée (discrimination)**.
On considèrera des modèles paramétriques $\phi(\boldsymbol{X}; \boldsymbol{\theta})$ en régression, i.e., $\widehat{y} = \mathbb{E}_{\widehat{\boldsymbol{\theta}}}(Y|\boldsymbol{X} = \boldsymbol{x})$ et en classification, i.e., $\widehat{y} = \arg\max_{y \in \mathcal{Y}} \mathbb{P}_{\widehat{\boldsymbol{\theta}}}(Y = y|\boldsymbol{X} = \boldsymbol{x})$.
Une partie portera sur l'**apprentissage non-supervisé** où l'on dispose uniquement d'observations marginales $(\boldsymbol{X}_i)_{i=1,\ldots,n}$ du couple $(\boldsymbol{X}, Y)$ et où l'objectif est de reconstruire les données manquantes $(Y_i)_{i=1,\ldots,n}$ (e.g typiquement une structure latente). On s'intéressera à la fois au cas où les données sont séquentielles (i.e $(Y_i)_i$ est un processus caché, typiquement Markovien) ou non (e.g. modélisation par mélange de densités). La dernière partie sera dédiée a la **segmentation** et au **clustering** de **données temporelles hétérogènes et non-stationnaires.**

Overview

Introduction on pattern recognition

Pattern recognition system



- Sensors
-> *Rough data*

- Denoising
- Data standardization
- Feature extraction
-> *Features*

- Feature selection
- Optimize the reprezentation space:
linear/nonlinear projection
-> *"Optimal features"*

- Chose the learning approach
- Learn the classes' parameters (if generative)
learn the decision boundaries (eg, discriminative)
- Model selection
-> *Decision rule*

- *"Score (e.g, classification error rate)"*

Data analysis process

- **Data acquisition :** the sensors part in which the rough data (e.g, speech signals, images ...) are acquired (e.g., measured) through dedicated sensors.

Data analysis process

- **Data acquisition :** the sensors part in which the rough data (e.g, speech signals, images ...) are acquired (e.g., measured) through dedicated sensors.
- **Data Preprocessing :** *denoising*, *standardization*,... **feature extraction** for data *representation*.

Data analysis process

- **Data acquisition :** the sensors part in which the rough data (e.g, speech signals, images ...) are acquired (e.g., measured) through dedicated sensors.
- **Data Preprocessing :** *denoising*, *standardization*,... **feature extraction** for data *representation*.
- **Feature selection :** optimization of the representation space by applying for example, *linear* or *nonlinear dimensionality reduction* techniques, in a *supervised* or *unsupervised* context.

Data analysis process

- **Data acquisition :** the sensors part in which the rough data (e.g, speech signals, images ...) are acquired (e.g., measured) through dedicated sensors.
- **Data Preprocessing :** *denoising*, *standardization*,... **feature extraction** for data *representation*.
- **Feature selection :** optimization of the representation space by applying for example, *linear* or *nonlinear dimensionality reduction* techniques, in a *supervised* or *unsupervised* context.
- **Classifier design :** Given a training set of (selected) features (observations) $\rightarrow$ design of a *decision rule* with respect to a chosen *optimality criterion*

Data analysis process

- **Classifier evaluation :** Once the classifier is designed, its performance has to be assesses for example by computing the *classification error rate* on new data examples, in order to evaluate its generalization capabilities.

Data analysis process

- **Classifier evaluation** : Once the classifier is designed, its performance has to be assesses for example by computing the *classification error rate* on new data examples, in order to evaluate its generalization capabilities.

- these stages are not independent. They are highly interrelated and, depending on the results, one may go back to redesign earlier stages in order to improve the overall performance.

- There are some methods that combine stages, for example, a statistical learning at two stages (learning for feature extraction and learning for classification).

- Some stages can also be removed, for example when the classifier is directly built without feature extraction nor feature selection.

Data representation

Speech signal representation :

- Linear Predictive Coding (LPC)
- Mel Frequency Cepstrum Coding (MFCC)
- ...

Image representation :

- Histogram
- Local Binary Patterns (LBP)
- ...

Data Classification

- We talk about "direct" classification, e.g., K-NN

Data Classification

- We talk about "direct" classification, e.g., K-NN
- Very often, the classification task involves a learning problem (e.g., Neural Networks, Support Vector Machines, Gaussian Discriminant Analysis, Gaussian Mixtures, Hidden Markov Models,...)

  $\Rightarrow$ we learn a decision rule, or a functional mapping between possibly labeled features and the considered classes.

Data Classification

- We talk about "direct" classification, e.g., K-NN
- Very often, the classification task involves a learning problem (e.g., Neural Networks, Support Vector Machines, Gaussian Discriminant Analysis, Gaussian Mixtures, Hidden Markov Models,...)

  $\Rightarrow$ we learn a decision rule, or a functional mapping between possibly labeled features and the considered classes.
- The main questions concerning the classifier design : the type of the approach (generative, discriminative,..), linear/non-linear separation, and the type of the criterion.

Data Classification

- We talk about "direct" classification, e.g., K-NN
- Very often, the classification task involves a learning problem (e.g., Neural Networks, Support Vector Machines, Gaussian Discriminant Analysis, Gaussian Mixtures, Hidden Markov Models,...)

  $\Rightarrow$ we learn a decision rule, or a functional mapping between possibly labeled features and the considered classes.
- The main questions concerning the classifier design : the type of the approach (generative, discriminative,..), linear/non-linear separation, and the type of the criterion.

  $\Rightarrow$ In this course, we focus on probabilistic classifiers in a *maximum likelihood estimation (MLE)* framework.
- Probabilistic approaches can easily address problems related to missing information and allows for the integration of prior knowledge (Bayesian approaches), such as experts information.

Machine learning context

- The paradigm for automatically (without human intervention) learning from raw data is known as *machine learning*
- acquisition of knowledge from rough data for analysis, interpretation, prediction.
- *automatically* extracting useful information, possibly unknown, from rough data :
  - features
  - simplified models
  - classes,...

Machine learning context

- Statistical learning = machine learning + Statistics
- distinguished by the fact that the data are assumed to be realizations of random variables $\Rightarrow$ define probability densities over the data $\Rightarrow$ statistical (probabilistic) models.
- take benefit from the asymptotic properties of the estimators, e.g., consistency (e.g., Maximum likelihood)
- To make accurate decisions and predictions for future data, there is an important need to understand the *processes generating the data*.

Machine learning context

- Statistical learning = machine learning + Statistics
- distinguished by the fact that the data are assumed to be realizations of random variables $\Rightarrow$ define probability densities over the data $\Rightarrow$ statistical (probabilistic) models.
- take benefit from the asymptotic properties of the estimators, e.g., consistency (e.g., Maximum likelihood)
- To make accurate decisions and predictions for future data, there is an important need to understand the *processes generating the data*.
- $\Rightarrow$ This therefore leads us to *generative* learning

Machine learning context

Supervised/Unsupervised :

- *Supervised learning* : both the input (observation) and the output (target : class in classification) are available

- The objective is to predict the class of new data given predefined learned classes : *classification (discrimination)* problem

Machine learning context

Supervised/Unsupervised :

- *Supervised learning* : both the input (observation) and the output (target : class in classification) are available

- The objective is to predict the class of new data given predefined learned classes : *classification (discrimination)* problem

- In several application domains, we are confronted with the problem of missing information (class label missing, unknown, hidden).
- $\Rightarrow$ an *unsupervised* learning problem
- The objective is to discover possible classes (exploratory analysis)
- main models : latent data models : e.g., mixture models, HMMs
  $\Rightarrow$ a *clustering/segmentation* problem.

Machine learning context

Static/Dynamic :
Two contexts for the classification problem :

- Static context : The classification rules are taken from *static* modeling techniques because the data are assumed to be independent
- this hypothesis may be restrictive regarding some real phenomena
- *dynamical* framework : building decision rules from sequential data (or time series).

Objectives

- machine learning concepts for data analysis
- overview of some statistical learning approaches from the literature, with a particular focus on generative learning (see how they work).
- How can we define an accurate discrimination rule by considering both homogeneous and dispersed data ? (classification (discrimination)
- When expert information is missing, how can we automatically search for possible classes ? unsupervised learning for segmentation, clustering..
- How can we model the underlying (dynamical) behavior from sequential data ? (Sequential modeling)

Discriminative learning

- Two main approaches are generally used in the statistical learning literature : the *discriminative* approach and the *generative* approach
- Discriminative approaches (especially used in supervised learning (classification, regression)) learn a direct map from the inputs $\mathbf{x}$ to the output y, or they directly learn a model of the conditional distribution $p(y|\mathbf{x})$.
- From the conditional distribution $p(y|\mathbf{x})$, we can make predictions of y for any new value of $\mathbf{x}$ by using the Maximum A Posteriori (MAP) classification rule :

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}).$$

Generative learning

- Generative classifiers learn a model of the joint distribution $p(\mathbf{x}, y)$

  $\Rightarrow$ model the class conditional density $p(\mathbf{x}|y)$ together with the prior probability $p(y)$.
- The required posterior class probability is then computed using Bayes' theorem

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{\sum_{y'} p(y')p(\mathbf{x}|y')}.$$

- the outputs y are not always available (i.e., they may be missing or hidden)

  $\Rightarrow$ generative approaches are more suitable for unsupervised learning.

$\Rightarrow$ In this course we focus on probabilistic models for data modeling, discrimination and clustering.

Classification (discrimination)

Data Classification

- Given a training data set comprising n labeled observations $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$ where $\mathbf{x}$ denotes the observation (or the input) which is assumed to be continuous-valued in $\mathcal{X} = \mathbb{R}^d$

- y denotes the target variable (or the output) representing the class label which is a discrete-valued variable in $\mathcal{Y} = \{1, \ldots, K\}$

- K being the number of classes.

- In classification, the aim is to predict the value of the class label y for a new observation $\mathbf{x}$.

K-NN

- a direct supervised classification approach

K-NN

- a direct supervised classification approach
- Does not need "learning" but only storing the data

K-NN

- a direct supervised classification approach
- Does not need "learning" but only storing the data
- It's Son very simple : its principle is as follows : the class of a new data point is the one of its nearest neighbors (the majority among the K nearest neighbors) in the sense of a chosen distance (e.g, Euclidean distance)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \tag{1}$$

K-NN

- a direct supervised classification approach
- Does not need "learning" but only storing the data
- It's Son very simple : its principle is as follows : the class of a new data point is the one of its nearest neighbors (the majority among the K nearest neighbors) in the sense of a chosen distance (e.g, Euclidean distance)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \tag{1}$$

- $\Rightarrow$ As it needs computing, for each test data point, the distances with all the data points from the labeled training set, it may be computationally expensive for large data sets.

K-NN

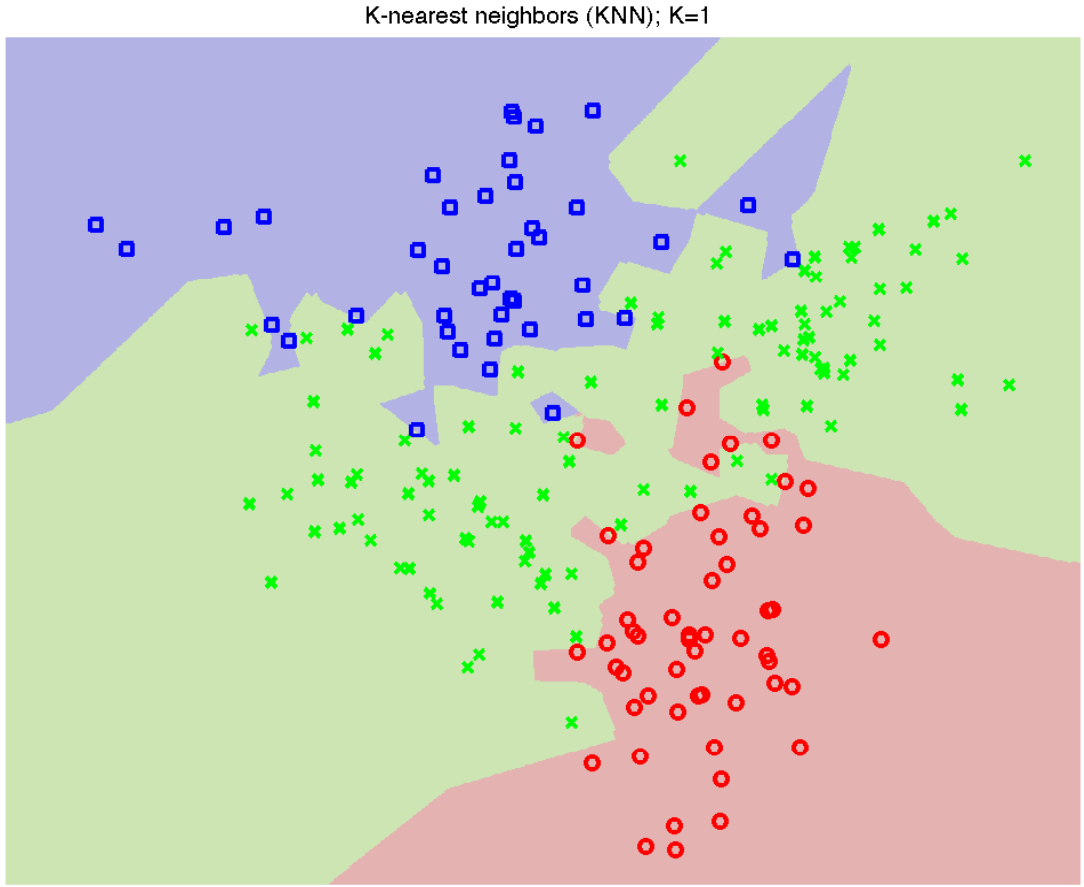
Algorithm 1 K-NN algorithm.

Inputs : Labeled data set : $\mathbf{X}^{\text{train}} = (\mathbf{x}_1^{\text{train}}, ..., \mathbf{x}_n^{\text{train}})$ and $\mathbf{y}^{\text{train}} = (y_1^{\text{train}}, ..., y_n^{\text{train}})$; Test data set $\mathbf{X}^{\text{test}} = (\mathbf{x}_1^{\text{test}}, ..., \mathbf{x}_m^{\text{test}})$; number of NN : K

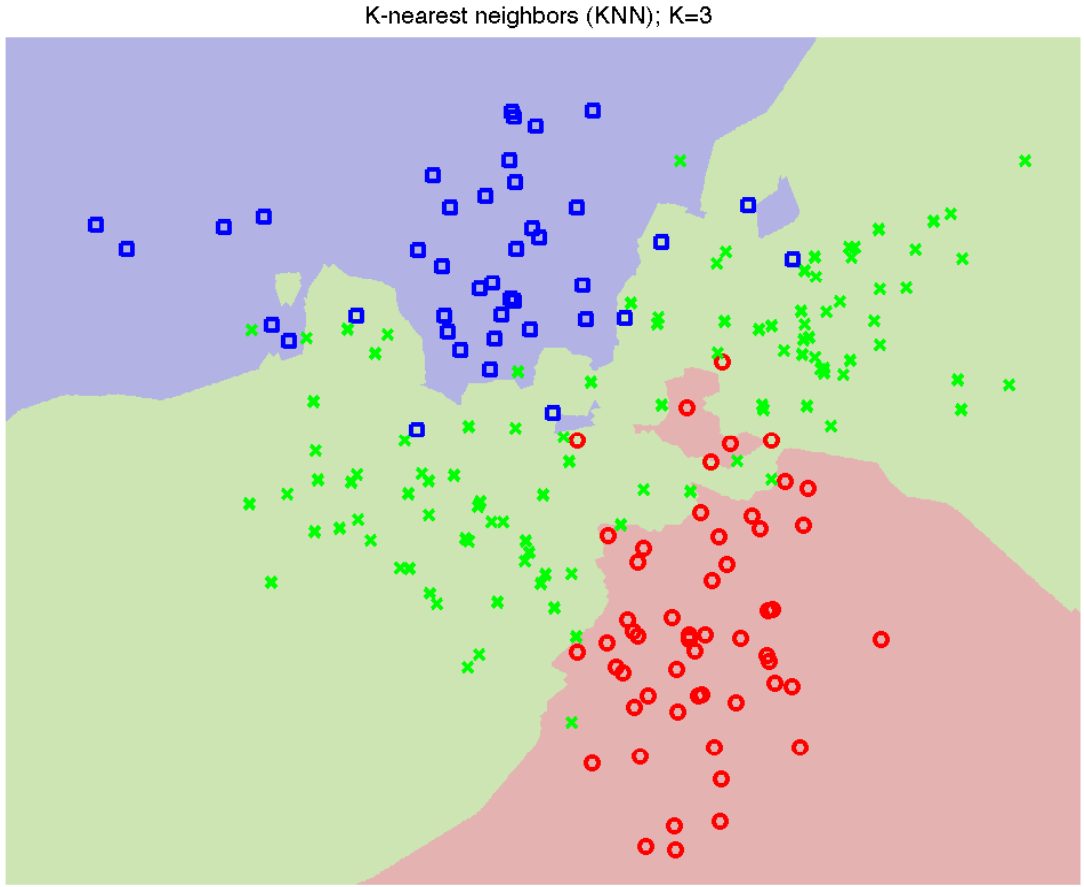
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, n$ **do**
    compute the Euclidean distances d_{ij} between $\mathbf{x}_i^{\text{test}}$and $\mathbf{x}_j^{\text{train}}$
    $\mathbf{d}_j \leftarrow \|\mathbf{x}_i - \mathbf{x}_j\|^2$
  end for
  The class y_i^{test} for the ith example is the one of its nearest neighbors :
  Sort the distance vector $\mathbf{d}_j$ in an increasing order for $j = 1, \ldots, n$
  Get at the same time the indexes of the elements in the new order
  Get the classes of the first K elements
  $\Rightarrow$ the class y_i^{test} is the majority class
end for

Output : Classes of the test data $\mathbf{y}^{\text{test}} = (y_1^{\text{test}}, ..., y_m^{\text{test}})$

K-NN



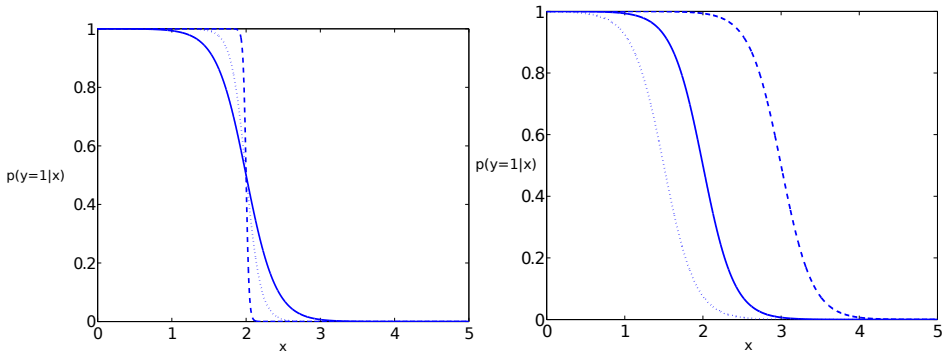
K-NN



Multi-class logistic regression

- a probabilistic supervised discriminative approach
- directly models the classes' posterior probabilities via :

$$p(y = k|\mathbf{x}) = \pi_k(\mathbf{x}; \mathbf{w}) = \frac{\exp(\boldsymbol{w}_k^T \mathbf{x})}{\sum_{h=1}^{K} \exp(\boldsymbol{w}_h^T \mathbf{x})}$$



- a logistic transformation of a linear function in $\mathbf{x}$
- ensures that the posterior probabilities are constrained to sum to one and remain in $[0, 1]$.
- The model parameter : $\mathbf{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K)^T$

Parameter estimation for Multi-class logistic regression

- The maximum likelihood is used to fit the model.
- The conditional log-likelihood of $\mathbf{w}$ for the given class labels $\mathbf{y} = (y_1, \ldots, y_n)$ conditionally on the inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}) = \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) &= \log \prod_{i=1}^{n} p(y_i|\mathbf{x}_i; \mathbf{w}) \\
&= \log \prod_{i=1}^{n} \prod_{k=1}^{K} p(y_i = k|\mathbf{x}_i; \mathbf{w})^{y_{ik}} \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log \pi_k(\mathbf{x}_i; \mathbf{w})
\end{aligned}$$

  where y_{ik} is an indicator binary variable such that $y_{ik} = 1$ if and only $y_i = k$ (i.e, $\mathbf{x}_i$ belongs to the class k).
- This log-likelihood is convex but can not be maximized in a closed form.
- The Newton-Raphson (NR) algorithm is generally used

Newton-Raphson for Multi-class logistic regression

- The Newton-Raphson algorithm is an iterative numerical optimization algorithm
- starts from an initial arbitrary solution $\mathbf{w}^{(0)}$, and updates the estimation of $\mathbf{w}$
- A single NR update is given by :

$$\mathbf{w}^{(l+1)} = \mathbf{w}^{(l)} - \left[\frac{\partial^2 \mathcal{L}(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T}\right]^{-1} \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \tag{2}$$

  where the Hessian and the gradient of $\mathcal{L}(\mathbf{w})$ (which are respectively the second and first derivative of $\mathcal{L}(\mathbf{w})$) are evaluated at $\mathbf{w} = \mathbf{w}^{(l)}$.
- NR can be stopped when the relative variation of $\mathcal{L}(\mathbf{w})$ is below a prefixed threshold.

IRLS I

The gradient component $\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}_h}$ $(h = 1, \ldots, K-1)$ is given by

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}_h} = \sum_{i=1}^{n} \big(y_{ih} - \pi_h(\mathbf{x}_i; \mathbf{w})\big)\mathbf{x}_i$$

which can be formulated in a matrix form as

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}_h} = \mathbf{X}^T(\mathbf{y}_h - \mathbf{p}_h)$$

where $\mathbf{X}$ is the $n \times (d+1)$ matrix whose rows are the input vectors $\mathbf{x}_i$, $\mathbf{y}_h$ is the $n \times 1$ column vector whose elements are the indicator variables y_{ih} for the hth logistic component :

$$\mathbf{y}_h = (y_{1h}, \ldots, y_{nh})^T$$

IRLS II

and $\mathbf{p}_h$ is the $n \times 1$ column vector of logistic probabilities corresponding to the ith input

$$\mathbf{p}_h = (\pi_h(\mathbf{x}_1; \mathbf{w}), \ldots, \pi_h(\mathbf{x}_n; \mathbf{w}))^T.$$

Thus, the matrix formulation of the gradient of $\mathcal{L}(\mathbf{w})$ for all the logistic components is

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{X}^{*T}(\mathbf{Y} - \mathbf{P}) \tag{3}$$

where $\mathbf{Y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_{K-1}^T)^T$ and $\mathbf{P} = (\mathbf{p}_1^T, \ldots, \mathbf{p}_{K-1}^T)^T$ are $n \times (K-1)$ column vectors and $\mathbf{X}^*$ is the $(n \times (K-1))$ by $(d+1)$ matrix of $K-1$ copies of $\mathbf{X}$ such that $\mathbf{X}^* = (\mathbf{X}^T, \ldots, \mathbf{X}^T)^T$.
The Hessian matrix is composed of $(K-1) \times (K-1)$ block matrices where each block matrix is of dimension $(d+1) \times (d+1)$ and is given by :

$$\frac{\partial^2 \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}_h \partial \boldsymbol{w}_k^T} = -\sum_{i=1}^{n} \pi_h(\mathbf{x}_i; \mathbf{w}) \left(\delta_{hk} - \pi_k(\mathbf{x}_i; \mathbf{w})\right) \mathbf{x}_i \mathbf{x}_i^T$$

which can be formulated in a matrix form as

$$\frac{\partial^2 \mathcal{L}(\mathbf{w})}{\partial \boldsymbol{w}_h \partial \boldsymbol{w}_k^T} = -\mathbf{X}^T \mathbf{W}_{hk} \mathbf{X}$$

where $\mathbf{W}_{hk}$ is the $n \times n$ diagonal matrix whose diagonal elements are $\pi_h(\mathbf{x}_i; \mathbf{w}) \left(\delta_{hk} - \pi_k(\mathbf{x}_i; \mathbf{w})\right)$ for $i = 1, \ldots, n$. For all the logistic components $(h, k = 1, \ldots, K-1)$, the Hessian takes the following form :

$$\frac{\partial^2 \mathcal{L}(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = -\mathbf{X}^{*T} \mathbf{W} \mathbf{X}^* \quad (4)$$

where $\mathbf{W}$ is the $(n \times (K-1))$ by $(n \times (K-1))$ matrix composed of $(K-1)) \times (K-1))$ block matrices, each block is $\mathbf{W}_{hk}$ $(h, k = 1, \ldots, K-1)$. It can be shown that the Hessian matrix for the multi-class logistic regression model is positive semi definite and therefore the optimized log-likelihood is concave.
The NR algorithm (2) in this case can therefore be reformulated from the Equations (3) and (4) as

$$\begin{aligned}
\mathbf{w}^{(l+1)} &= \mathbf{w}^{(l)} + (\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X}^{*})^{-1}\mathbf{X}^{*T}(\mathbf{Y} - \mathbf{P}^{(l)}) \\
&= (\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X}^{*})^{-1}\left[\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X}^{*}\mathbf{w}^{(l)} + \mathbf{X}^{*T}(\mathbf{Y} - \mathbf{P}^{(l)})\right] \\
&= (\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X})^{-1}\mathbf{X}^{*T}\left[\mathbf{W}^{(l)}\mathbf{X}^{*}\mathbf{w}^{(l)} + (\mathbf{Y} - \mathbf{P}^{(l)})\right] \\
&= (\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X}^{*})^{-1}\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{Y}^{*}
\end{aligned}$$

where $\mathbf{Y}^{*} = \mathbf{X}^{*}\mathbf{w}^{(l)} + (\mathbf{W}^{(l)})^{-1}(\mathbf{Y} - \mathbf{P}^{(l)})$ which yields in the Iteratively Reweighted Least Squares (IRLS) algorithm.

Neural Network

notes vues en cours

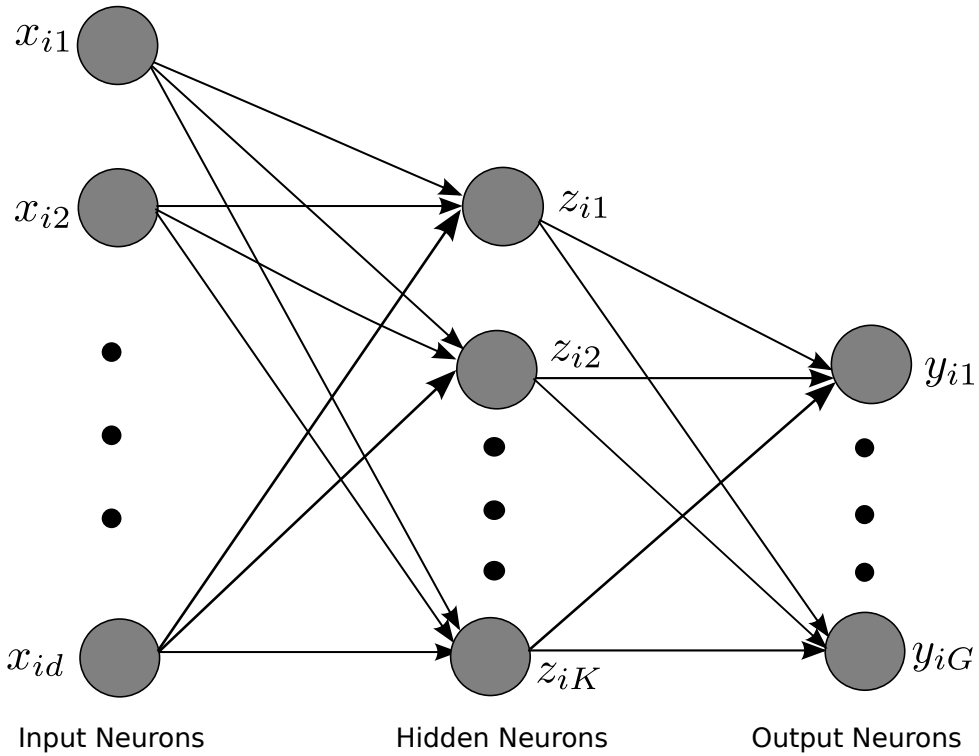


Figure – Graphical representation of Multi-Layer Perceptron (MLP).

Linear Discriminant Analysis

- generative approach that consists in modeling each conditional-class density by a multivariate Gaussian :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big)$$

Linear Discriminant Analysis

- generative approach that consists in modeling each conditional-class density by a multivariate Gaussian :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big)$$

- $\boldsymbol{\mu}_k \in \mathbb{R}^d$ is the mean vector
- $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d\times d}$ is the covariance matrix
- $\boldsymbol{\Psi}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ for $k = 1, \ldots, K$.

Linear Discriminant Analysis

- generative approach that consists in modeling each conditional-class density by a multivariate Gaussian :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big)$$

- $\boldsymbol{\mu}_k \in \mathbb{R}^d$ is the mean vector
- $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d\times d}$ is the covariance matrix
- $\boldsymbol{\Psi}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ for $k = 1, \ldots, K$.

- Linear Discriminant Analysis (LDA) arises when we assume that all the classes have a common covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}\ \forall k = 1, \ldots, K$.

Linear Discriminant Analysis

- The term “linear" in LDA is due to the fact that the decision boundaries between each pair of classes k and h are linear.

Linear Discriminant Analysis

- The term “linear" in LDA is due to the fact that the decision boundaries between each pair of classes k and h are linear.

- The decision boundary between classes k and h, which is the set of inputs $\mathbf{x}$ verifying $p(y = k|\mathbf{x}) = p(y = h|\mathbf{x})$, or by equivalence :

$$\log \frac{p(y = g|\mathbf{x}; \boldsymbol{\Psi}_k)}{p(y = h|\mathbf{x}; \boldsymbol{\Psi}_h)} = 0 \Leftrightarrow \log \frac{\pi_k}{\pi_h} + \log \frac{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma})}{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_h, \boldsymbol{\Sigma})} =$$

Linear Discriminant Analysis

- The term "linear" in LDA is due to the fact that the decision boundaries between each pair of classes k and h are linear.

- The decision boundary between classes k and h, which is the set of inputs $\mathbf{x}$ verifying $p(y = k|\mathbf{x}) = p(y = h|\mathbf{x})$, or by equivalence :

$$\log \frac{p(y = g|\mathbf{x}; \boldsymbol{\Psi}_k)}{p(y = h|\mathbf{x}; \boldsymbol{\Psi}_h)} = 0 \Leftrightarrow \log \frac{\pi_k}{\pi_h} + \log \frac{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma})}{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_h, \boldsymbol{\Sigma})} =$$

$$\Leftrightarrow \log \tfrac{\pi_k}{\pi_h} - \tfrac{1}{2}(\boldsymbol{\mu}_k + \boldsymbol{\mu}_h)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_h) + \mathbf{x}^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_h) = 0,$$

$\Rightarrow$ a linear function in $\mathbf{x}$ and therefore the classes will be separated by hyperplanes in the input space.

Linear Discriminant Analysis : Parameter Estimation

- Each of the class prior probabilities π_k is calculated with the proportion of the class g in the training data set :

$$\pi_k = \frac{\sum_{i|y_i=k}}{n} = \frac{n_k}{n}.$$

Linear Discriminant Analysis : Parameter Estimation

- Each of the class prior probabilities π_k is calculated with the proportion of the class g in the training data set :

$$\pi_k = \frac{\sum_{i|y_i=k}}{n} = \frac{n_k}{n}.$$

- The parameters $\boldsymbol{\Psi}_k$ are estimated by maximum likelihood

Linear Discriminant Analysis : Parameter Estimation

- Each of the class prior probabilities π_k is calculated with the proportion of the class g in the training data set :

$$\pi_k = \frac{\sum_{i|y_i=k}}{n} = \frac{n_k}{n}.$$

- The parameters $\boldsymbol{\Psi}_k$ are estimated by maximum likelihood
- the log-likelihood of $\boldsymbol{\Psi}_k$ given an i.i.d sample :

$$\mathcal{L}(\boldsymbol{\Psi}_k) = \log \prod_{i|y_i=k} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) = \sum_{i|y_i=k} \log \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}).$$

- $\Rightarrow$ The problem is solved in a closed form

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i|y_i=k} \mathbf{x}_i,$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i|y_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T,$$

Illustration

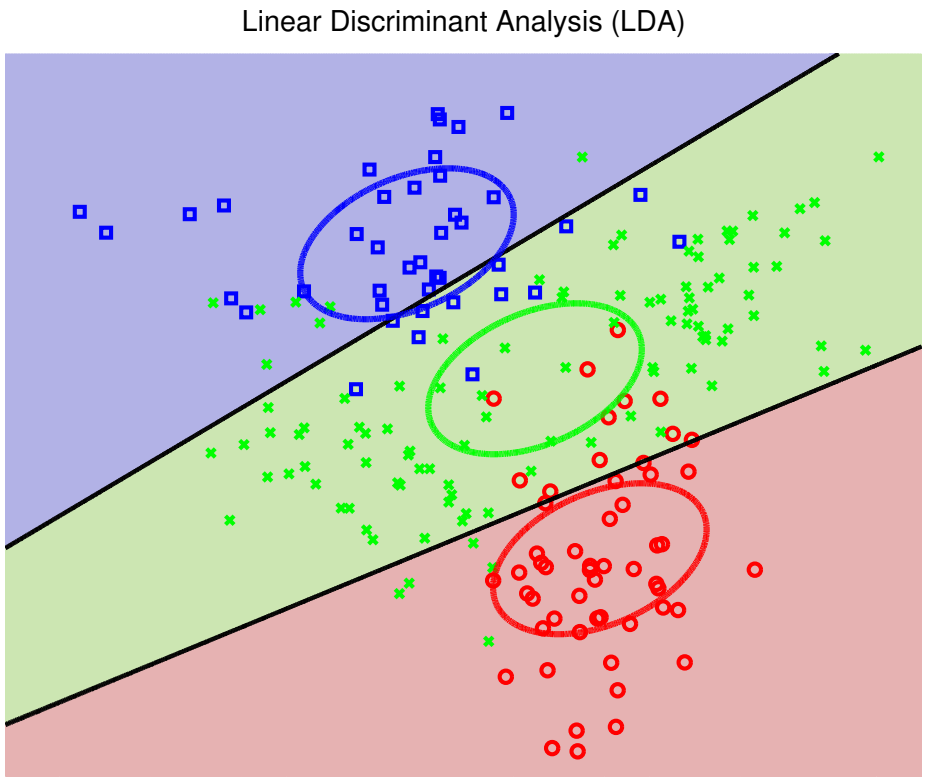


Figure – A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that LDA provides linear separation.

Quadratic Discriminant Analysis

- Quadratic Discriminant Analysis (QDA) is an extension of LDA that considers a different covariance matrix for each class.

Quadratic Discriminant Analysis

- Quadratic Discriminant Analysis (QDA) is an extension of LDA that considers a different covariance matrix for each class.
- The decision functions are quadratic :

$$\log \frac{p(y=k|\mathbf{x})}{p(y=h|\mathbf{x})} = \log \frac{\pi_k}{\pi_h} - \frac{1}{2} \log \frac{|\mathbf{\Sigma}_k|}{|\mathbf{\Sigma}_h|} - \frac{1}{2}\{(\mathbf{x}-\boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_h)^T \mathbf{\Sigma}_h^{-1}(\mathbf{x}-\boldsymbol{\mu}_h)\} = 0.$$

$\Rightarrow$ This function is quadratic in $\mathbf{x}$, we then get quadratic discriminant functions in the input space.

Quadratic Discriminant Analysis

- Quadratic Discriminant Analysis (QDA) is an extension of LDA that considers a different covariance matrix for each class.
- The decision functions are quadratic :

$$\log \frac{p(y=k|\mathbf{x})}{p(y=h|\mathbf{x})} = \log \frac{\pi_k}{\pi_h} - \frac{1}{2} \log \frac{|\mathbf{\Sigma}_k|}{|\mathbf{\Sigma}_h|} - \frac{1}{2}\{(\mathbf{x}-\boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_h)^T \mathbf{\Sigma}_h^{-1}(\mathbf{x}-\boldsymbol{\mu}_h)\} = 0.$$

  $\Rightarrow$ This function is quadratic in $\mathbf{x}$, we then get quadratic discriminant functions in the input space.
- The parameters $\mathbf{\Psi}_k$ for QDA are estimated similarly as for LDA, except that separate covariance matrix must be estimated for each class :

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i|y_i=k} \mathbf{x}_i$$

$$\hat{\mathbf{\Sigma}}_k = \frac{1}{n_k} \sum_{i|y_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T.$$

Illustration

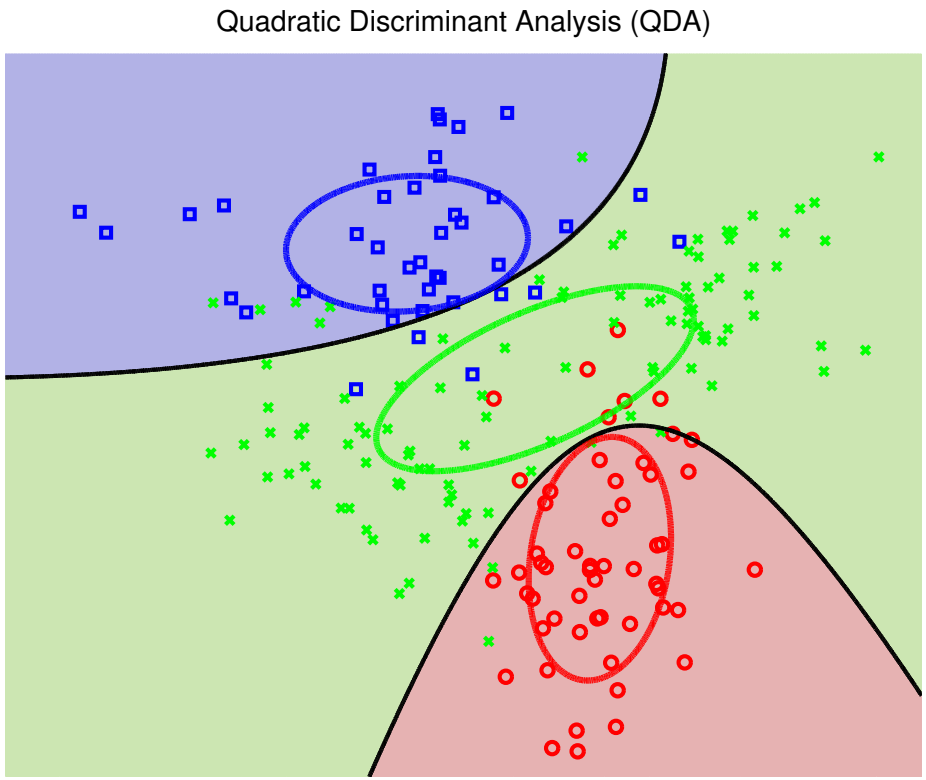


Figure – A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that QDA provides quadratic boundaries in the plan.

Mixture Discriminant Analysis

- for Gaussian discriminant analysis, in both LDA and QDA, each class density is modeled by a single Gaussian.

- This may be limited for modeling non homogeneous classes where the classes are dispersed.

  $\Rightarrow$ In Mixture Discriminant Analysis (MDA) each class density is modeled by a Gaussian mixture density

- with MDA, we can therefore capture many specific properties of real data such as multimodality, unobserved heterogeneity, heteroskedasticity, etc.

Mixture Discriminant Analysis (MDA)

- Each class g is modeled by a Gaussian mixture density :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \sum_{r=1}^{R_k} \pi_{kr}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{kr},\boldsymbol{\Sigma}_{kr})$$

where R_k is the number of mixture components for class k

Mixture Discriminant Analysis (MDA)

- Each class g is modeled by a Gaussian mixture density :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \sum_{r=1}^{R_k} \pi_{kr}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{kr},\boldsymbol{\Sigma}_{kr})$$

  where R_k is the number of mixture components for class k

- $$\boldsymbol{\Psi}_k = (\pi_{k1},\ldots,\pi_{kR_k},\boldsymbol{\mu}_{k1},\ldots,\boldsymbol{\mu}_{kR_k},\ldots,\boldsymbol{\Sigma}_{k1},\ldots,\boldsymbol{\Sigma}_{kR_k})$$

  is the parameter vector of the mixture density of class k

Mixture Discriminant Analysis (MDA)

- Each class g is modeled by a Gaussian mixture density :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \sum_{r=1}^{R_k} \pi_{kr}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{kr},\boldsymbol{\Sigma}_{kr})$$

  where R_k is the number of mixture components for class k

- $$\boldsymbol{\Psi}_k = (\pi_{k1},\ldots,\pi_{kR_k},\boldsymbol{\mu}_{k1},\ldots,\boldsymbol{\mu}_{kR_k},\ldots,\boldsymbol{\Sigma}_{k1},\ldots,\boldsymbol{\Sigma}_{kR_k})$$

  is the parameter vector of the mixture density of class k
- the π_{kr}'s $(r=1,\ldots,R_k)$ are the non-negative mixing proportions satisfying $\sum_{r=1}^{R_k}\pi_{kr}=1\ \forall k$.

Mixture Discriminant Analysis (MDA)

- Each class g is modeled by a Gaussian mixture density :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \sum_{r=1}^{R_k} \pi_{kr}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{kr},\boldsymbol{\Sigma}_{kr})$$

  where R_k is the number of mixture components for class k

- $$\boldsymbol{\Psi}_k = (\pi_{k1},\ldots,\pi_{kR_k},\boldsymbol{\mu}_{k1},\ldots,\boldsymbol{\mu}_{kR_k},\ldots,\boldsymbol{\Sigma}_{k1},\ldots,\boldsymbol{\Sigma}_{kR_k})$$

  is the parameter vector of the mixture density of class k
- the π_{kr}'s $(r=1,\ldots,R_k)$ are the non-negative mixing proportions satisfying $\sum_{r=1}^{R_k}\pi_{kr}=1\ \forall k$.
- we can allow a different covariance matrix for each mixture component as well as a common covariance matrix

Illustration

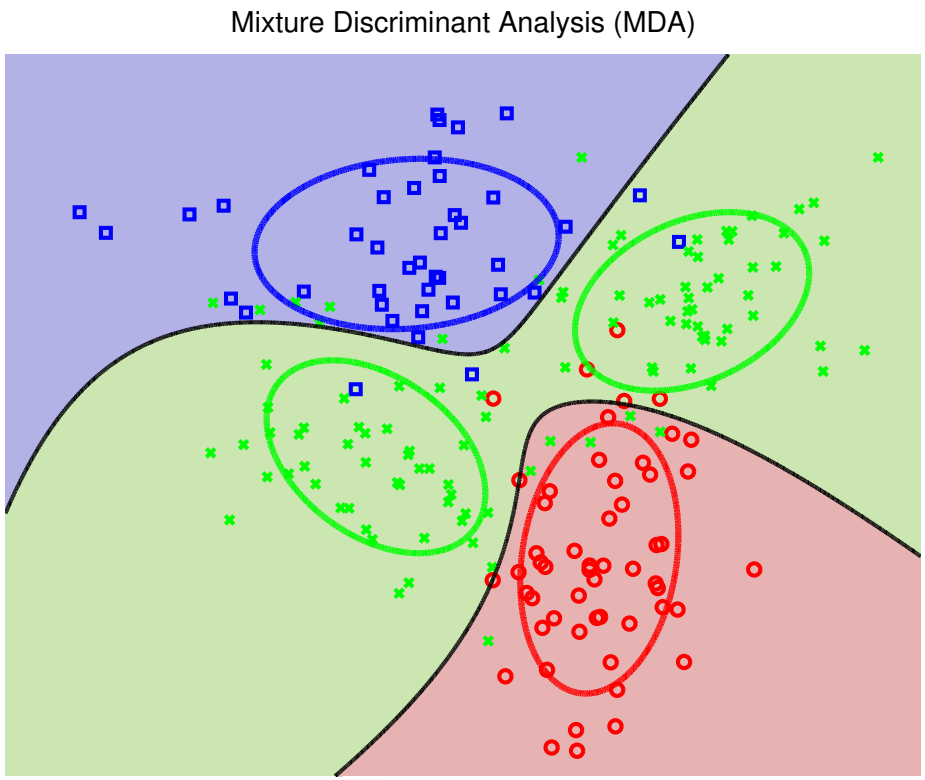


Figure – A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that MDA provides more non-linear decision boundaries.

Illustrations of Logistic Regression, LDA, QDA and MDA

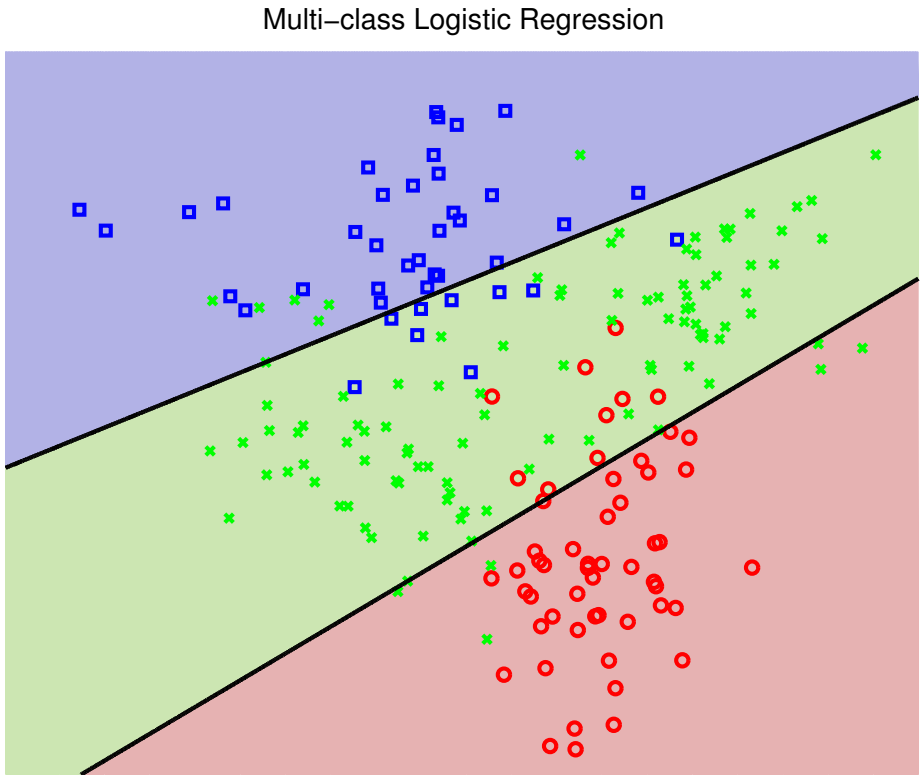


Figure – A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that both LDA and Logistic regression provide linear separation, while QDA and MDA provide non linear separation. MDA can further deal the problem of heterogeneous classes.

Illustrations of Logistic Regression, LDA, QDA and MDA

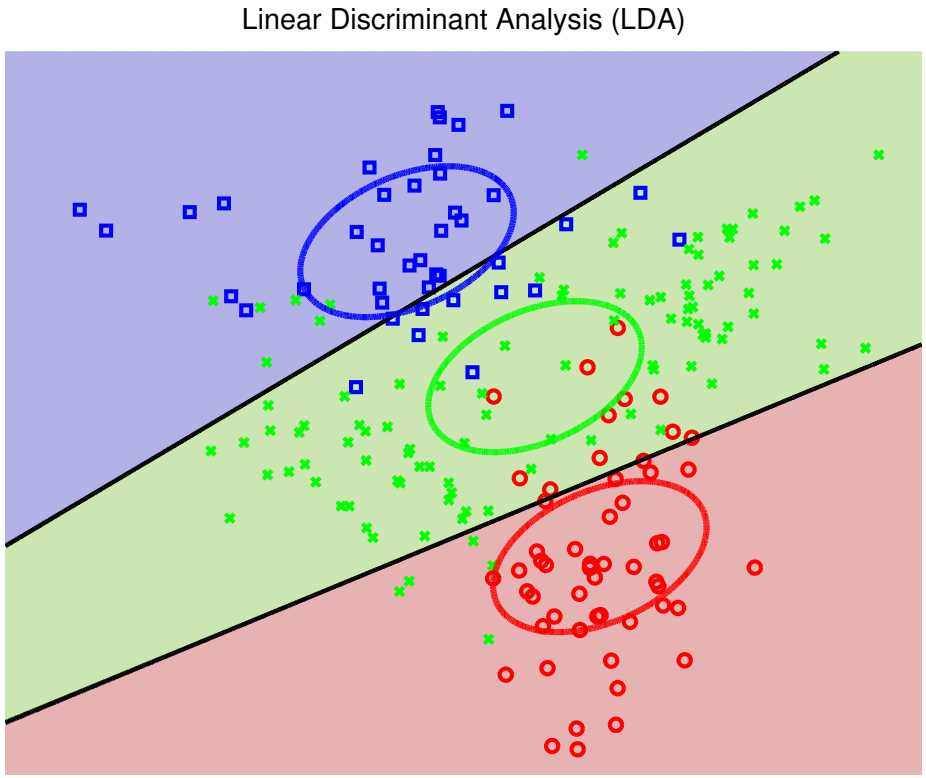


Figure – A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that both LDA and Logistic regression provide linear separation, while QDA and MDA provide non linear separation. MDA can further deal the problem of heterogeneous classes.

Illustrations of Logistic Regression, LDA, QDA and MDA

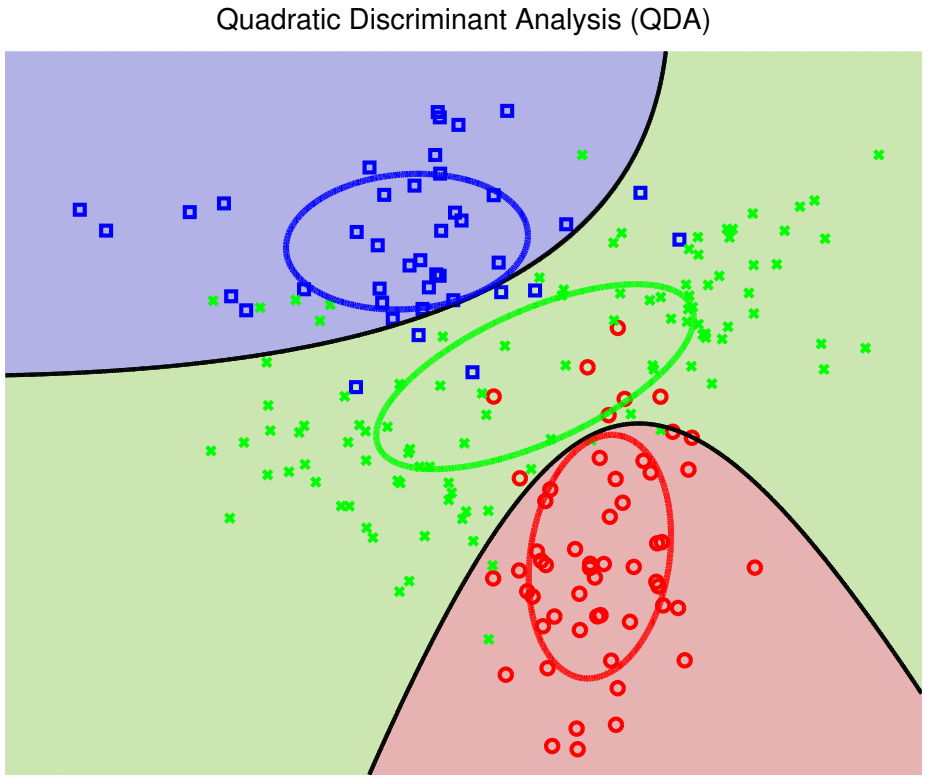


Figure – A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that both LDA and Logistic regression provide linear separation, while QDA and MDA provide non linear separation. MDA can further deal the problem of heterogeneous classes.

Illustrations of Logistic Regression, LDA, QDA and MDA

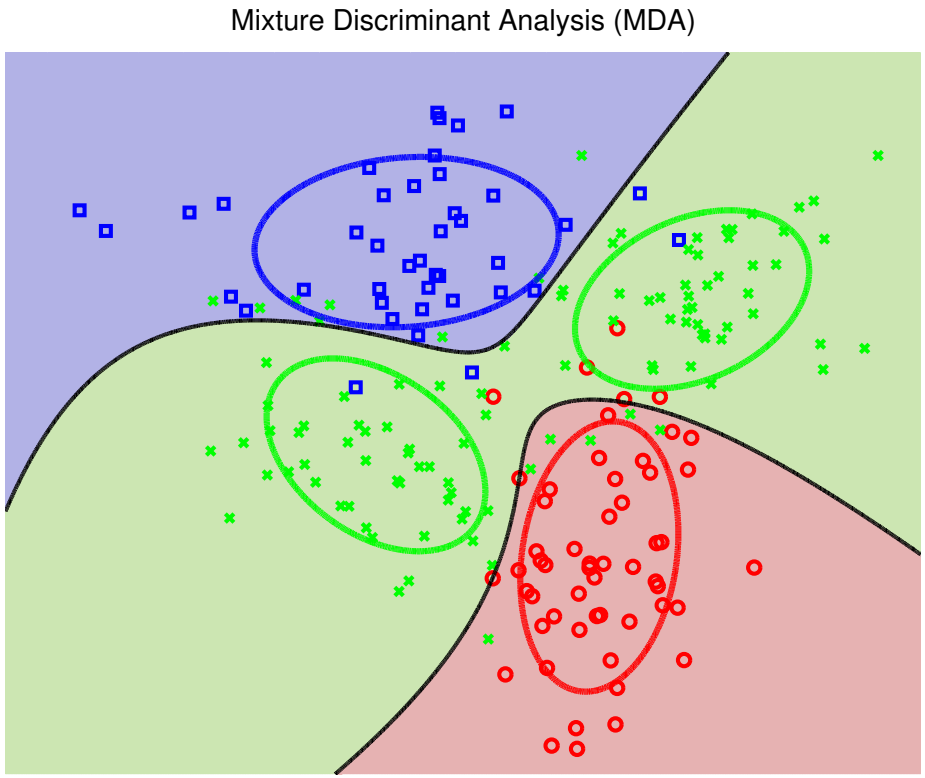


Figure – A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that both LDA and Logistic regression provide linear separation, while QDA and MDA provide non linear separation. MDA can further deal the problem of heterogeneous classes.

Mixture models

Mixture models

- Finite mixture models are an example of latent variable models
- widely used in probabilistic machine learning and pattern recognition.
- very useful to model heterogeneous classes since they assume that each class is composed of sub-classes.
- The finite mixture model decomposes the density of $\mathbf{x}$ into a weighted linear combination of K component densities.
- The mixture model allows for placing K component densities in the input space to approximate the true density.

  $\Rightarrow$ Mixtures provide a natural generalization of the simple parametric density model which is global, to a weighted sum of these models, allowing local adaptation to the density of the data in the input space.

Model definition

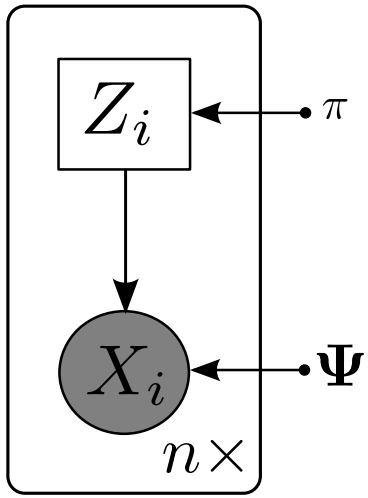
- Let z represent a discrete random variable (binomial or multinomial) which takes its values in the finite set $\mathcal{Z} = \{1, \ldots, K\}$.
- In a general setting, the mixture density of $\mathbf{x}$ is

$$\begin{aligned} f(\mathbf{x}; \boldsymbol{\Psi}) &= \sum_{k=1}^{K} p(z = k) f(\mathbf{x} | z = k; \boldsymbol{\Psi}_k) \\ &= \sum_{k=1}^{K} \pi_k f_k(\mathbf{x}; \boldsymbol{\Psi}_k), \end{aligned}$$

  - $\pi_k = p(z = k)$: the probability that a randomly chosen data point was generated by component k. Referred to as *mixing proportions* $\pi_k \geq 0 \ \forall k$, and $\sum_{k=1}^{K} \pi_k = 1$.
  - $f_1, \ldots, f_K$ are the *component densities*.
  - Each f_k typically consists of a relatively simple parametric model $p(\mathbf{x}|z = k; \boldsymbol{\Psi}_k)$ (such as a Gaussian distribution with parameters $\boldsymbol{\Psi}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$).

Model definition

Figure – Graphical representation of a mixture model.



Parameter estimation for the mixture model

The common parameter estimation methods for mixture models :

- the *maximum likelihood*
- the *Bayesian* methods (Maximum A Posteriori (MAP)) where a prior distribution is assumed for the model parameters

$\Rightarrow$ In this course, we focus on the maximum likelihood framework.

- maximize the observed-data likelihood as a function of the parameters $\boldsymbol{\Psi} = (\pi_1, \ldots, \pi_K, \boldsymbol{\Psi}_k, \ldots, \boldsymbol{\Psi}_K)$, over the parameter space $\boldsymbol{\Omega}$
- The optimization algorithm is the Expectation-Maximization (EM) algorithm

Parameter estimation for the mixture model

- Assume we have an i.i.d sample $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$.
- The observed-data log-likelihood of $\boldsymbol{\Psi}$ given $\mathbf{X}$ is given by :

$$\begin{aligned} \mathcal{L}(\boldsymbol{\Psi}; \mathbf{X}) &= \log \prod_{i=1}^{n} p(\mathbf{x}_i; \boldsymbol{\Psi}) \\ &= \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k f_k(\mathbf{x}_i; \boldsymbol{\Psi}_k). \end{aligned}$$

- the log-likelihood to be maximized results in a nonlinear function due to the logarithm of the sum
- very difficult to maximize it in a closed form

  $\Rightarrow$ maximize it (locally) using iterative procedures such as gradient ascent, a Newton Raphson procedure or the Expectation-Maximization (EM) algorithm

  $\Rightarrow$ We will focus on the EM algorithm which is widely used and particularly adapted for mixture models.

EM algorithm

- a broadly applicable approach to the iterative computation of maximum likelihood estimates in the framework of latent data models.
- In particular, the EM algorithm simplifies considerably the problem of fitting finite mixture models by maximum likelihood.

- an iterative algorithm where each iteration consists of two steps :
  1. the Expectation step (E-step) : computes the expectation of the complete-data log-likelihood, given the observations $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and a current value $\boldsymbol{\Psi}^{(q)}$ of the model parameter
  2. the Maximization step (M-step) : Maximize the expected complete-data log-likelihood over the parameter space

EM algorithm

- let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a set of n i.i.d observations with $\mathbf{x}_i \in \mathbb{R}^d$
- $\mathbf{z} = (z_1, \ldots, z_n)$ denote the corresponding unobserved (missing) labels with $z_i \in \mathcal{Z} = \{1, \ldots, K\}$.
- The complete-data : $(\mathbf{X}, \mathbf{z}) = ((\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_n, z_n))$
- The complete-data log-likelihood :

$$\begin{aligned}
\mathcal{L}_c(\boldsymbol{\Psi}; \mathbf{X}, \mathbf{z}) &= \log p((\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_n, z_n); \boldsymbol{\Psi}) = \log \prod_{i=1}^{n} p(\mathbf{x}_i, z_i; \boldsymbol{\Psi}) \\
&= \sum_{i=1}^{n} \log \prod_{k=1}^{K} \Big[p(z_i = k) p(\mathbf{x} | z_i = k; \boldsymbol{\Psi}_k) \Big]^{z_{ik}} \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \pi_k f_k \left(\mathbf{x}_i; \boldsymbol{\Psi}_k \right),
\end{aligned}$$

  where $z_{ik} = 1$ if $z_i = k$ (i.e, when $\mathbf{x}_i$ is generated by the kth component density) and $z_{ik} = 0$ otherwise.
- this log-likelihood depends on the unobservable data $\mathbf{z}$!.

EM algorithm

- The EM algorithm starts with an initial parameter $\boldsymbol{\Psi}^{(0)}$ and iteratively alternates between the two following steps until convergence :
- **E-step (Expectation)** : computes the expectation of the complete-data log-likelihood given the observations $\mathbf{X}$ and the current value $\boldsymbol{\Psi}^{(q)}$ of the parameter $\boldsymbol{\Psi}$ (q being the current iteration).

$$
\begin{aligned}
Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)}) &= \mathbb{E}\left[\mathcal{L}_c(\boldsymbol{\Psi}; \mathbf{X}, \mathbf{z}) | \mathbf{X}; \boldsymbol{\Psi}^{(q)}\right] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{E}[z_{ik}|\mathbf{x}_i, \boldsymbol{\Psi}^{(q)}] \log \pi_k f_k\left(\mathbf{x}_i; \boldsymbol{\Psi}_k\right) \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} p(z_{ik} = 1|\mathbf{x}_i; \boldsymbol{\Psi}^{(q)}) \log \pi_k f_k\left(\mathbf{x}_i; \boldsymbol{\Psi}_k\right) \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \tau_{ik}^{(q)} \log \pi_k f_k\left(\mathbf{x}_i; \boldsymbol{\Psi}_k\right)
\end{aligned}
$$

EM algorithm

- where

$$\tau_{ik}^{(q)} = p(z_i = k|\mathbf{x}_i; \boldsymbol{\Psi}^{(q)}) = \frac{\pi_k f_k\left(\mathbf{x}_i; \boldsymbol{\Psi}_k^{(q)}\right)}{\sum_{\ell=1}^{K} \pi_\ell f_\ell\left(\mathbf{x}_i; \boldsymbol{\Psi}_\ell^{(q)}\right)}$$

  is the posterior probability that $\mathbf{x}_i$ originates from the kth component density.

- In $\mathbb{E}[z_{ik}|\mathbf{x}_i, \boldsymbol{\Psi}^{(q)}]$, we used the fact that conditional expectations and conditional probabilities are the same for the indicator binary-valued variables z_{ik} : $\mathbb{E}[z_{ik}|\mathbf{x}_i, \boldsymbol{\Psi}^{(q)}] = p(z_{ik} = 1|\mathbf{x}_i, \boldsymbol{\Psi}^{(q)})$.

  $\Rightarrow$ From the expression of $Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)})$, we can see that this step simply requires the computation of the posterior probabilities $\tau_{ik}^{(q)}$.

EM algorithm

M-step (Maximization) : updates the estimate of $\boldsymbol{\Psi}$ by the value $\boldsymbol{\Psi}^{(q+1)}$ of $\boldsymbol{\Psi}$ that maximizes the Q-function $Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)})$ with respect to $\boldsymbol{\Psi}$ over the parameter space $\boldsymbol{\Omega}$:

$$\boldsymbol{\Psi}^{(q+1)} = \arg\max_{\boldsymbol{\Psi}\in\boldsymbol{\Omega}} Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)}).$$

We can write

$$Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)}) = Q_{\pi}(\pi_1, \ldots, \pi_K, \boldsymbol{\Psi}^{(q)}) + \sum_{k=1}^{K} Q_{\boldsymbol{\Psi}_k}(\boldsymbol{\Psi}_k, \boldsymbol{\Psi}^{(q)})$$

where

$$Q_{\pi}(\pi_1, \ldots, \pi_K, \boldsymbol{\Psi}^{(q)}) = \sum_{i=1}^{n}\sum_{k=1}^{K} \tau_{ik}^{(q)} \log \pi_k$$

$$Q_{\boldsymbol{\Psi}_k}(\boldsymbol{\Psi}_k, \boldsymbol{\Psi}^{(q)}) = \sum_{i=1}^{n} \tau_{ik}^{(q)} \log f_k(\mathbf{x}_i; \boldsymbol{\Psi}_k)$$

M-Step

$\Rightarrow$ the maximization of the function $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(q)})$ w.r.t $\boldsymbol{\Psi}$ can be performed by separately maximizing Q_π with respect to the mixing proportions $(\pi_1, \ldots, \pi_K)$ and $Q_{\boldsymbol{\Psi}_k}$ with respect to parameters $\boldsymbol{\Psi}_k$ for each of the K components densities.

- The function Q_π is maximized with respect to $(\pi_1, \ldots, \pi_K) \in [0,1]^K$ subject to the constraint $\sum_k \pi_k = 1$. This maximization is done in a closed using Lagrange multipliers form and leads to

$$\pi_k^{(q+1)} = \frac{\sum_{i=1}^n \tau_{ik}^{(q)}}{n} = \frac{n_k^{(q)}}{n},$$

- $n_k^{(q)}$ can be viewed as the expected cardinal number of the subpopulation k estimated at iteration q.
- The update of $\boldsymbol{\Psi}_k$ depends on the form of the density f_k (e.g., Gaussian)

EM for Gaussian mixture models (GMMs)

The Gaussian mixture model (GMM) :

$$f(\mathbf{x}_i; \boldsymbol{\Psi}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

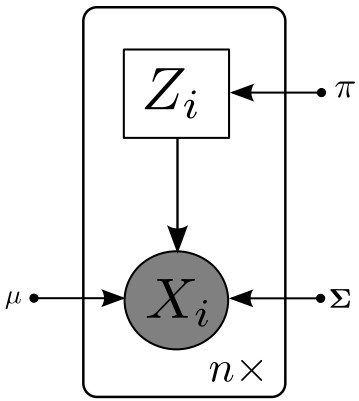


Figure – Graphical representation of a Gaussian mixture model.

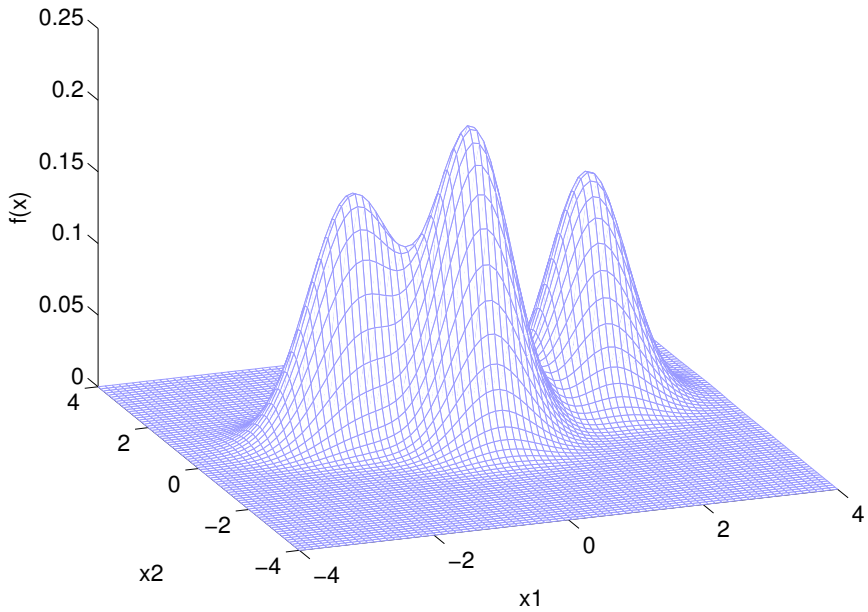


Figure – An example of a three-component Gaussian mixture density in $\mathbb{R}^2$.

EM for GMMs

- The observed-data log-likelihood of $\boldsymbol{\Psi}$ for the Gaussian mixture model :

$$\mathcal{L}(\boldsymbol{\Psi}; \mathbf{X}) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right).$$

- The complete-data log-likelihood of $\boldsymbol{\Psi}$ for the Gaussian mixture model :

$$\mathcal{L}_c(\boldsymbol{\Psi}; \mathbf{X}, \mathbf{z}) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \pi_k \mathcal{N}\left(\mathbf{x}_i; \boldsymbol{\mu}_k \boldsymbol{\Sigma}_k\right).$$

EM :

- Starts with an initial parameter $\boldsymbol{\Psi}^{(0)} = (\pi_1^{(0)}, \ldots, \pi_K^{(0)}, \boldsymbol{\Psi}_1^{(0)}, \ldots, \boldsymbol{\Psi}_K^{(0)})$ where $\boldsymbol{\Psi}_k^{(0)} = (\boldsymbol{\mu}_k^{(0)}, \boldsymbol{\Sigma}_k^{(0)})$

E-Step for GMMs

- the expected complete-data log-likelihood :

$$
\begin{aligned}
Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)}) &= \mathbb{E}\left[\mathcal{L}_c(\boldsymbol{\Psi}; \mathbf{X}, \mathbf{z}) | \mathbf{X}; \boldsymbol{\Psi}^{(q)}\right] \\
&= \sum_{i=1}^{n}\sum_{k=1}^{K} \tau_{ik}^{(q)} \log \pi_k + \sum_{i=1}^{n}\sum_{k=1}^{K} \tau_{ik}^{(q)} \log \mathcal{N}\left(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right).
\end{aligned}
$$

$\Rightarrow$ This step therefore computes the posterior probabilities

$$
\tau_{ik}^{(q)} = p(z_i = k | \mathbf{x}_i, \boldsymbol{\Psi}^{(q)}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k^{(q)}, \boldsymbol{\Sigma}_k^{(q)})}{\sum_{\ell=1}^{K} \pi_\ell \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_\ell^{(q)}, \boldsymbol{\Sigma}_\ell^{(q)})}
$$

that $\mathbf{x}_i$ originates from the kth component density.

M-Step for GMMs

- update the parameter $\boldsymbol{\Psi}$ by the value $\boldsymbol{\Psi}^{(q+1)}$ of $\boldsymbol{\Psi}$ that maximizes the function $Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)})$ w.r.t $\boldsymbol{\Psi}$ over the parameter space $\boldsymbol{\Omega}$.

$$\begin{aligned}
\boldsymbol{\mu}_k^{(q+1)} &= \frac{1}{n_k^{(q)}} \sum_{i=1}^{n} \tau_{ik}^{(q)} \mathbf{x}_i, \\
\boldsymbol{\Sigma}_k^{(q+1)} &= \frac{1}{n_k^{(q)}} \sum_{i=1}^{n} \tau_{ik}^{(q)} (\mathbf{x}_i - \boldsymbol{\mu}^{(q+1)})(\mathbf{x}_i - \boldsymbol{\mu}^{(q+1)})^T.
\end{aligned}$$

- The E- and M-steps are alternated iteratively until the change in the log likelihood value are less than some specified threshold.

Algorithm 2 Pseudo code of the EM algorithm for GMMs.

Inputs : a data set $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and the number of clusters K

fix a threshold $\epsilon > 0$; set $q \leftarrow 0$ (iteration)

Initialize : $\boldsymbol{\Psi}^{(0)} = (\pi_1^{(0)}, \ldots, \pi_K^{(0)}, \boldsymbol{\Psi}_1^{(0)}, \ldots, \boldsymbol{\Psi}_K^{(0)})$ with $\boldsymbol{\Psi}_k^{(0)} = (\boldsymbol{\mu}_K^{(0)}, \boldsymbol{\Sigma}_K^{(0)})$

while increment in log-likelihood $> \epsilon$ **do**

E-step :

for $k = 1, \ldots, K$ **do**

Compute $\tau_{ik}^{(q)} = \frac{\pi_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k^{(q)}, \boldsymbol{\Sigma}_k^{(q)})}{\sum_{\ell=\mathbf{1}}^{K} \pi_\ell \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_\ell^{(q)}, \boldsymbol{\Sigma}_\ell^{(q)})}$ for $i = 1, \ldots, n$

end for

M-step :

for $k = 1, \ldots, K$ **do**

Compute $\pi_k^{(q+1)} = \frac{\sum_{i=\mathbf{1}}^{n} \tau_{ik}^{(q)}}{n}$

Compute $\boldsymbol{\mu}_k^{(q+1)} = \frac{1}{n_k^{(q)}} \sum_{i=1}^{n} \tau_{ik}^{(q)} \mathbf{x}_i$

Compute $\boldsymbol{\Sigma}_k^{(q+1)} = \frac{1}{n_k^{(q)}} \sum_{i=1}^{n} \tau_{ik}^{(q)} (\mathbf{x}_i - \boldsymbol{\mu}^{(q+1)})(\mathbf{x}_i - \boldsymbol{\mu}^{(q+1)})^T$

end for

$q \leftarrow q + 1$

end while

Outputs : $\hat{\boldsymbol{\Psi}} = \boldsymbol{\Psi}^{(q)}$; $\hat{\tau}_{ik} = \tau_{ik}^{(q)}$ (a fuzzy partition of the data)

Initialization Strategies and stopping rules for EM

- The initialization of EM is a crucial point since it maximizes locally the log-likelihood.
- if the initial value is inappropriately selected, the EM algorithm may lead to an unsatisfactory estimation.
- The most used strategy : use several EM tries and select the solution maximizing the log-likelihood among those runs.
- For each run of EM, one can initialize it
  - randomly
  - by Computing a parameter estimate from another clustering algorithm such as K-means, Classification EM, Stochastic EM ...
  - with a few number of steps of EM itself.
- Stop EM when the relative increase of the log-likelihood between two iterations is below a fixed threshold $|\frac{\mathcal{L}^{(q+1)}-\mathcal{L}^{(q)}}{\mathcal{L}^{(q)}}| \leq \epsilon$ or when a predefined number of iterations is reached.

EM properties

- The EM algorithm always monotonically increases the observed-data log-likelihood.
- The sequence of parameter estimates generated by the EM algorithm converges toward at least a local maximum or a stationary value of the incomplete-data likelihood function.
- numerical stability
- simplicity of implementation
- reliable convergence
- In general, both the E- and M-steps will have particularly simple forms when the complete-data probability density function is from the exponential family ;
- Some drawbacks : EM is sometimes very slow to converge especially for high dimensional data ;

  in some problems, the E- or M-step may be analytically intractable (but this can be tackled by using EM extensions)

EM extensions

- The EM variants mainly aim at :
  1. increasing the convergence speed of EM and addressing the optimization problem in the M-step
  2. computing the E-step when it is intractable.
- In the first case, one can speak about deterministic algorithms :
  - e.g., Incremental EM (IEM)
  - Gradient EM
  - Generalized EM (GEM) algorithm
  - Expectation Conditional Maximization (ECM)
  - Expectation Conditional Maximization Either (ECME)
- In the second case, one can speak about stochastic algorithms :
  - e.g., Monte Carlo EM (MCEM)
  - Stochastic EM (SEM)
  - Simulated Annealing EM (SAEM)

Clustering

Clustering

- Clustering is often referred to as unsupervised learning in the sense that the class labels of the data are unknown (missing, hidden). Only the observations $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ are given,
- suitable for many applications where labeled data is difficult to obtain.
- also used to explore and characterize a dataset before running a supervised learning task.
- In clustering, the data are grouped by some notion of dissimilarity.

  $\Rightarrow$ a dissimilarity measure must be defined based on the data.
- the aim of clustering is to find a partition of the data by dividing them into clusters (groups) such that the data within a group tend to be more similar to one another as compared to the data belonging to different groups.
- There is, distance-based, model-based, hierarchical, topographical clustering approaches, etc

Example : Image segmentation

image originale

image etiquetee par les classes estimees

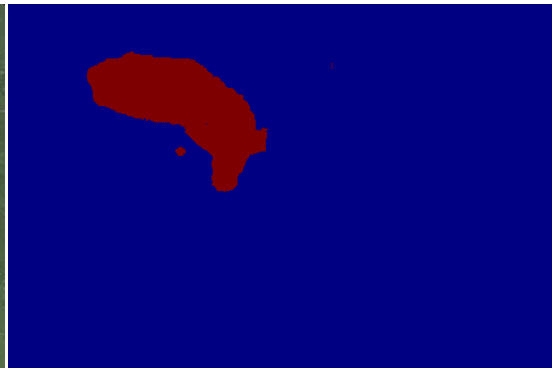


image originale

image segmentee par GMM–EM (K=3)

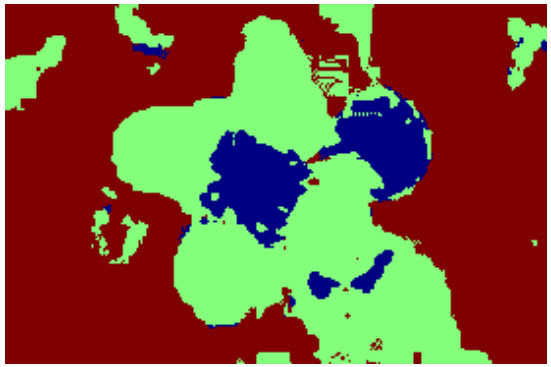
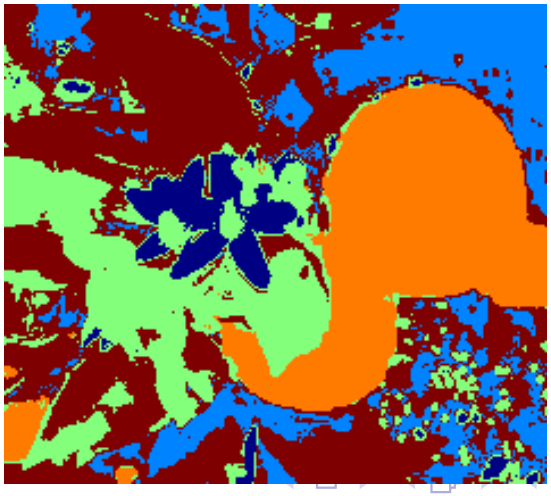


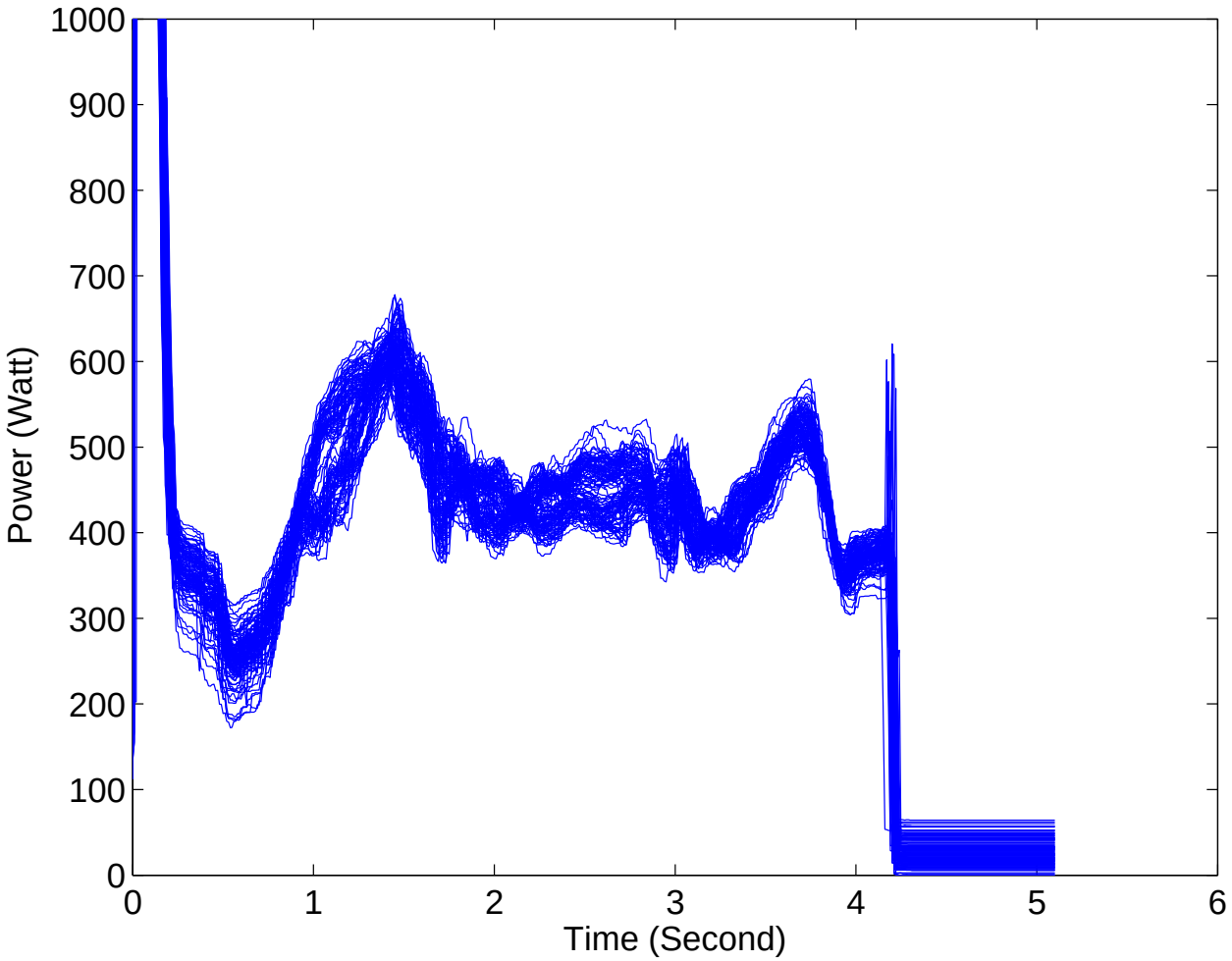
image originale

image segmentee par GMM–CEM (K=5)



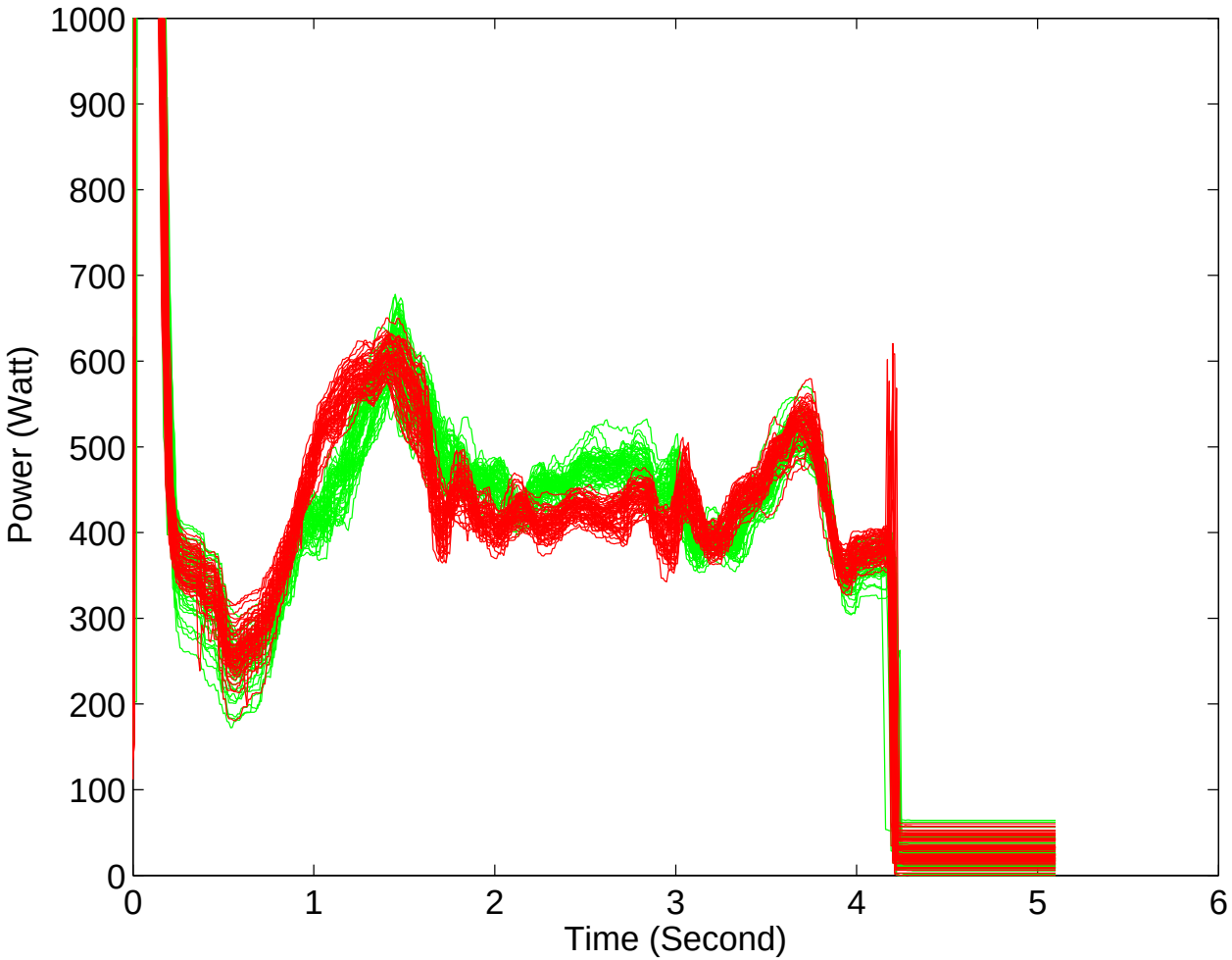
example 2 : Fault detection

Real data (curves)

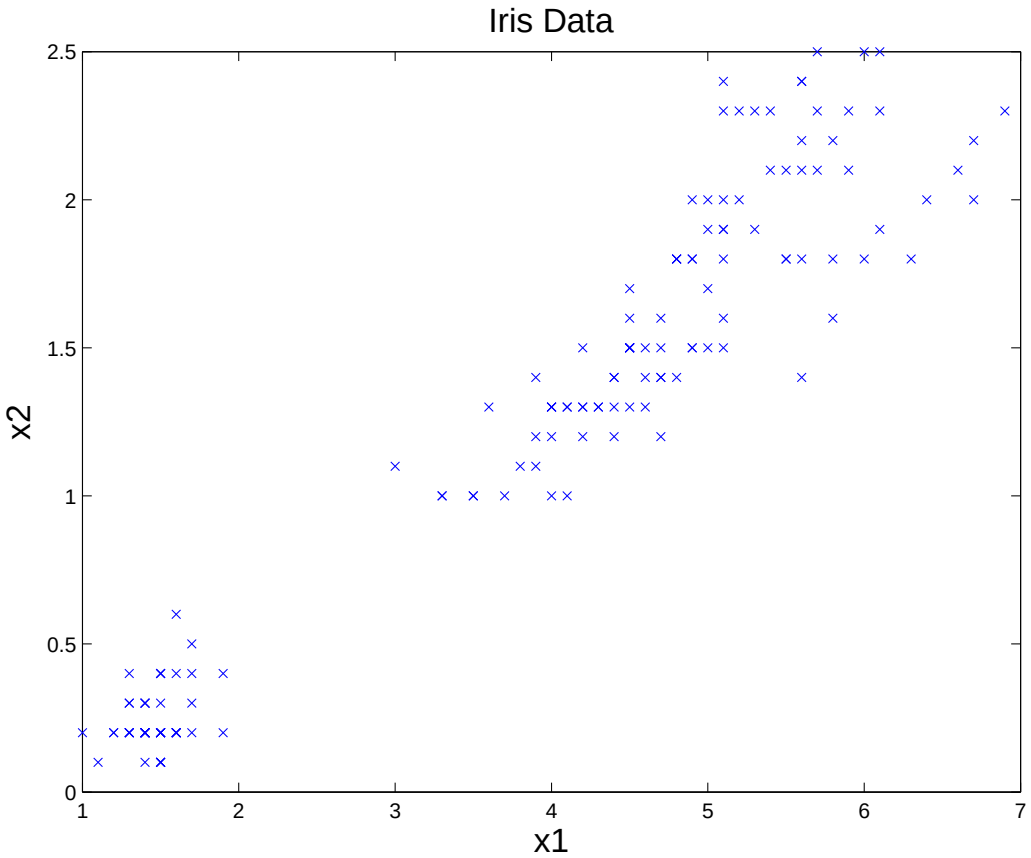


example 2 : Fault detection

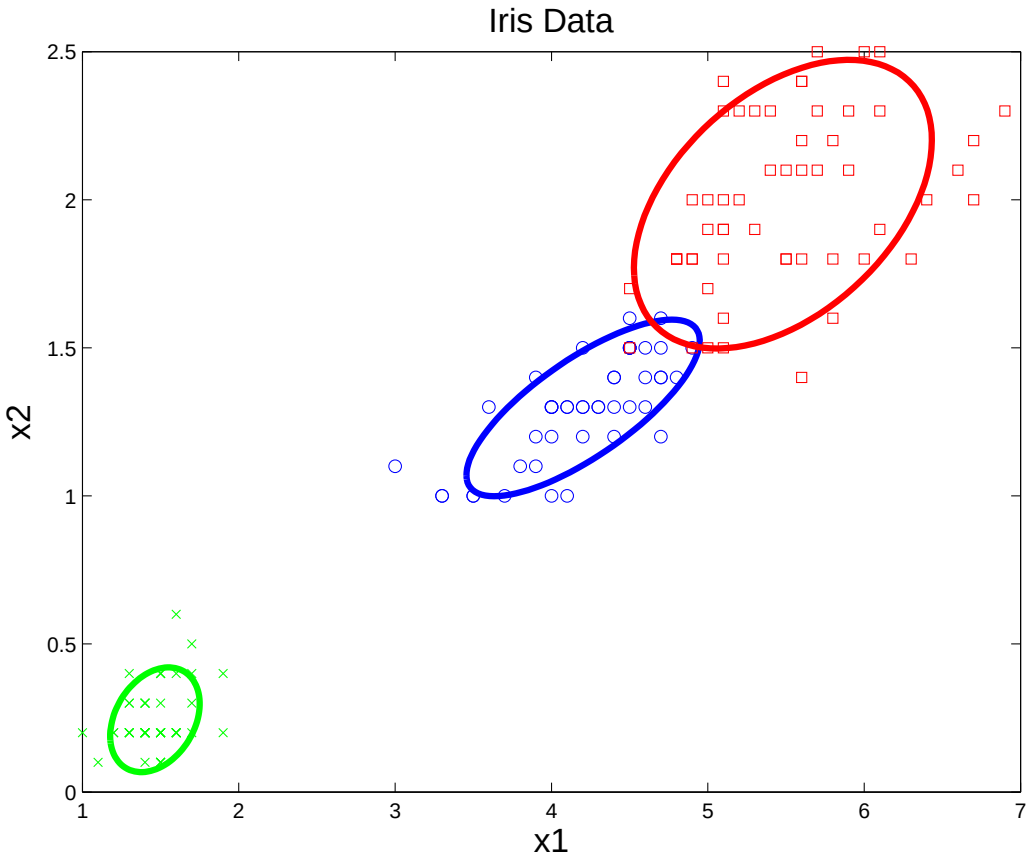
Clustering results



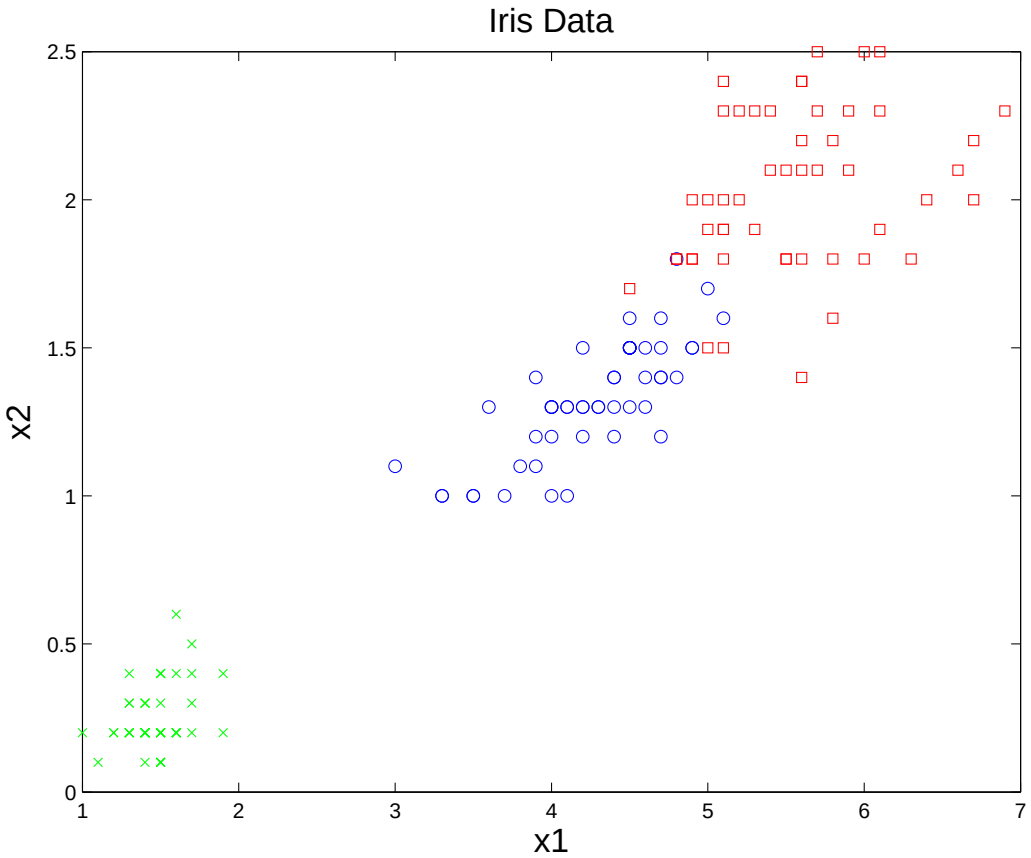
Example 3 : Iris data



Example 3 : Iris data



Example 3 : Iris data



K-means

- a straightforward and widely used clustering algorithm, is one of the most important algorithms in unsupervised learning.
- an iterative clustering algorithm that partitions a given dataset into a predefined number of clusters K.
- the value K is chosen by prior knowledge ; how many clusters are desired ; ..
- In K-means, each cluster is represented by its mean (cluster centroid) $\boldsymbol{\mu}_k$ in $\mathbb{R}^d$.
- The default measure of dissimilarity for K-means is the Euclidean distance $\|.\|^2$.
- K-means attempts to minimize the following nonnegative objective function referred to as *distortion measure* :

$$J(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \mathbf{z}) = \sum_{k=1}^{K} \sum_{i=1}^{n} z_{ik} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

which corresponds to the total squared Euclidean distance between each data point $\mathbf{x}_i$ and its closest cluster representative $\boldsymbol{\mu}_{z_i}$.

K-means

- start with an initial solution $(\boldsymbol{\mu}_1^{(0)}, \ldots, \boldsymbol{\mu}_K^{(0)})$ (eg, by randomly choosing K points in $\mathbb{R}^d$ or some data points)

1. **Assignment step** : Each data point is assigned to its closest centroid using the Euclidian distance : $\forall i = 1, \ldots, n$

$$z_{ik}^{(q)} = \begin{cases} 1 \text{ if } k = \arg\min_{z \in \mathcal{Z}} \|\mathbf{x}_i - \boldsymbol{\mu}_z\|^2 \\ 0 \text{ otherwise.} \end{cases}$$

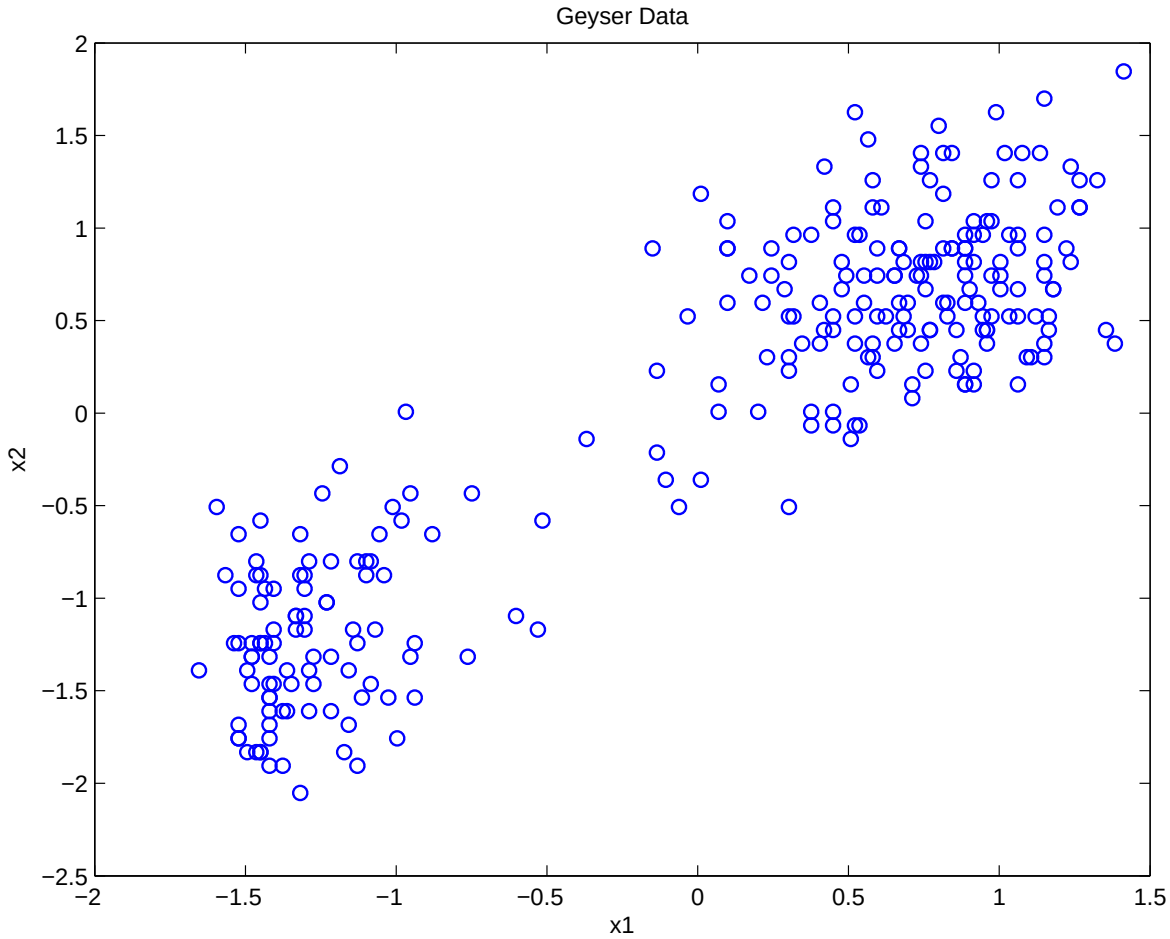
2. **Relocation step** : Each cluster representative is relocated to the center (i.e., arithmetic mean) of all data points assigned to it :

$$\boldsymbol{\mu}_k^{(q+1)} = \frac{\sum_{i=1}^n z_{ik}^{(q)} \mathbf{x}_i}{\sum_{i=1}^n z_{ik}^{(q)}},$$

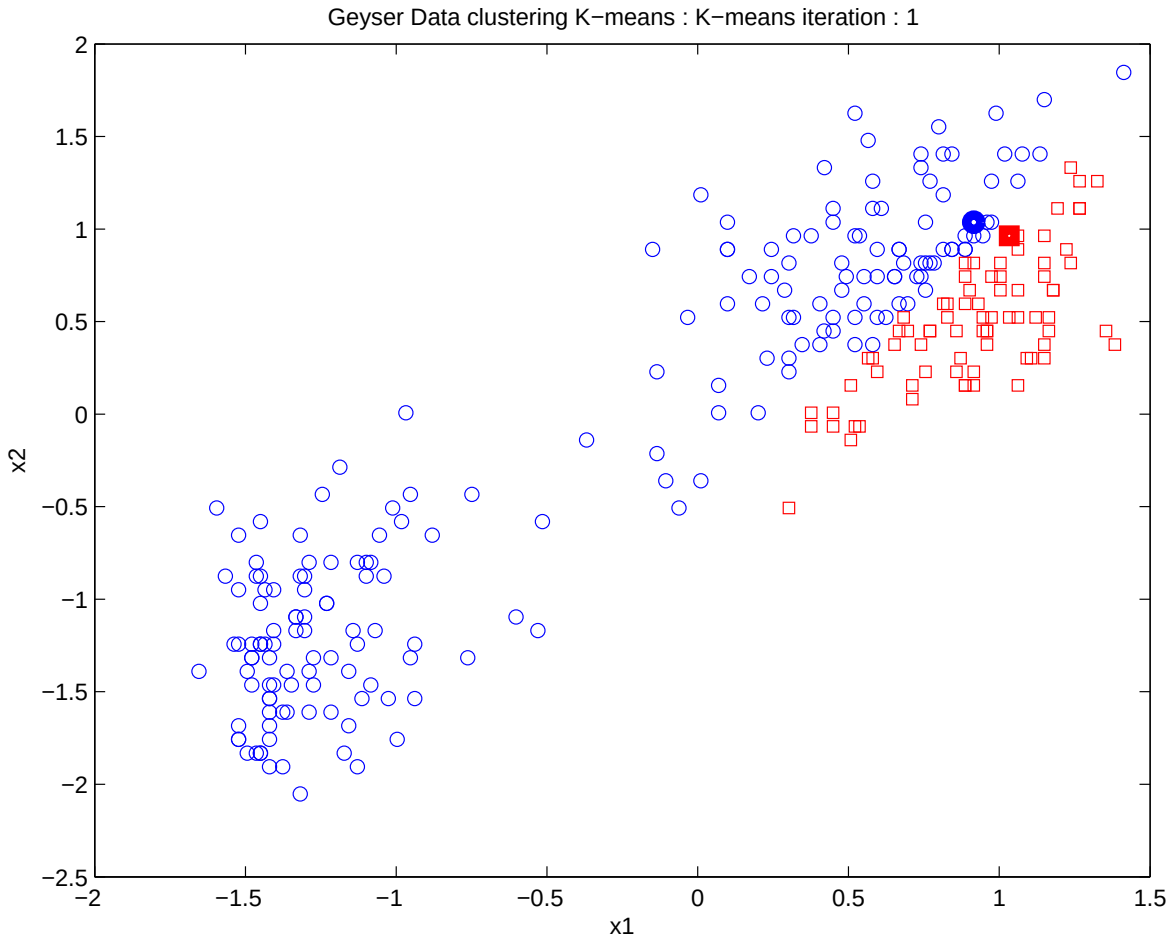
q being the current iteration.

$\Rightarrow$ The K-means algorithm is simple to implement and relatively fast.

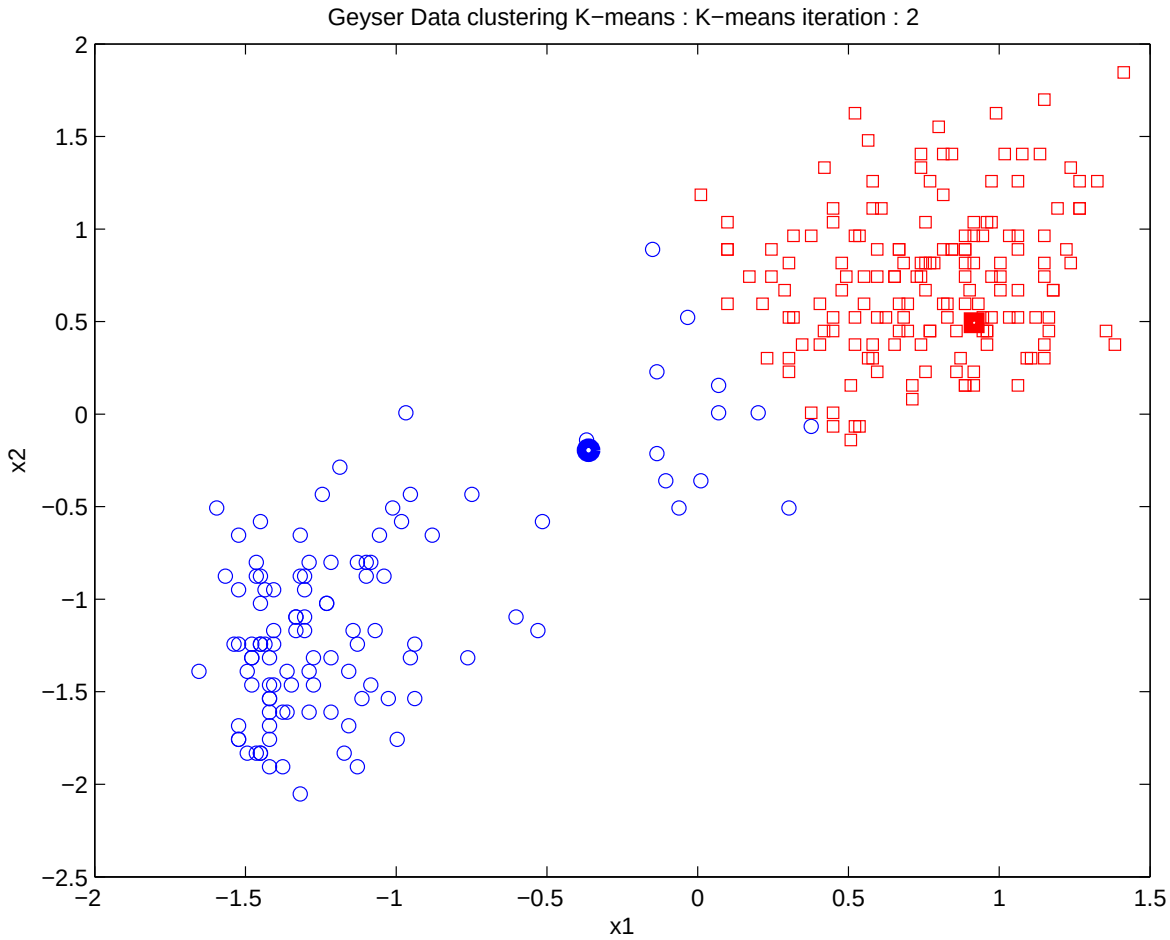
Illustration



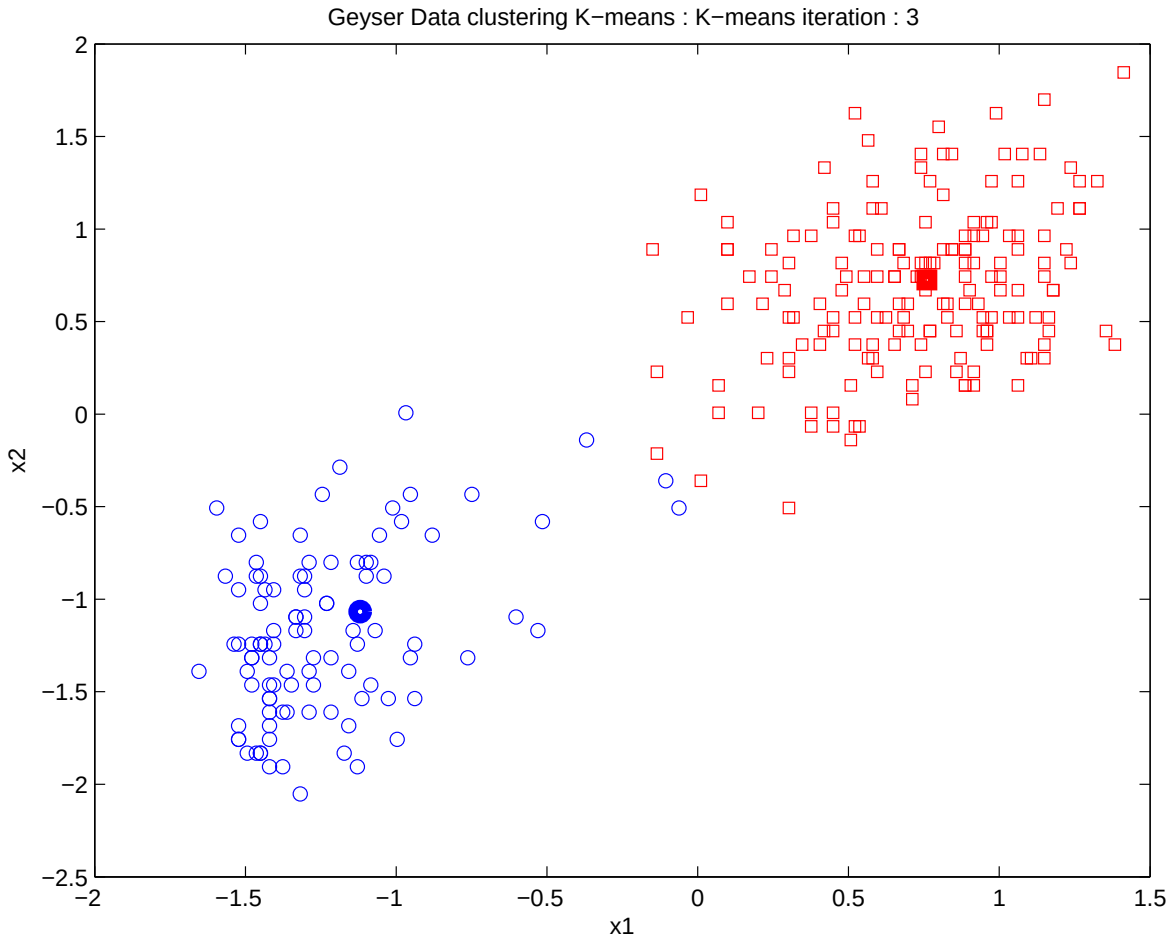
Illustration



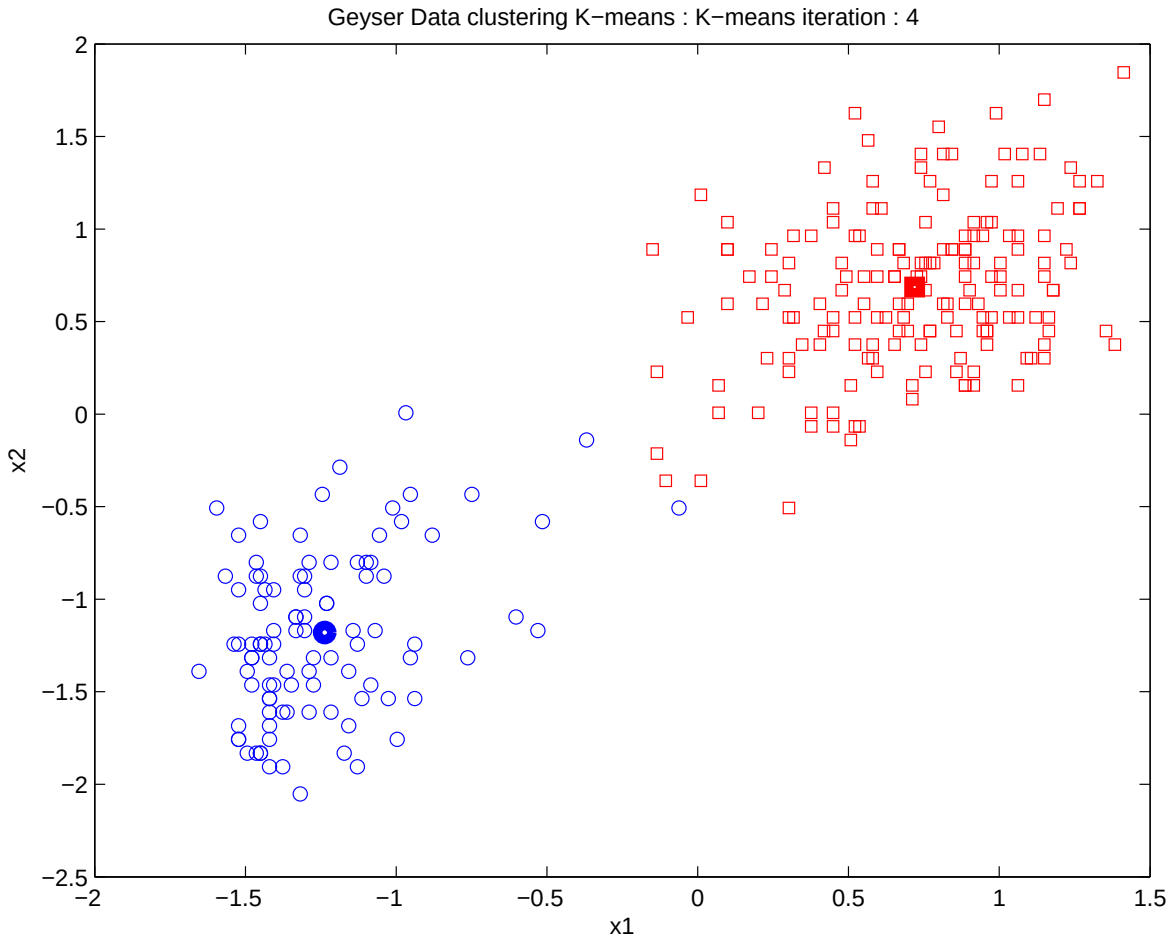
Illustration



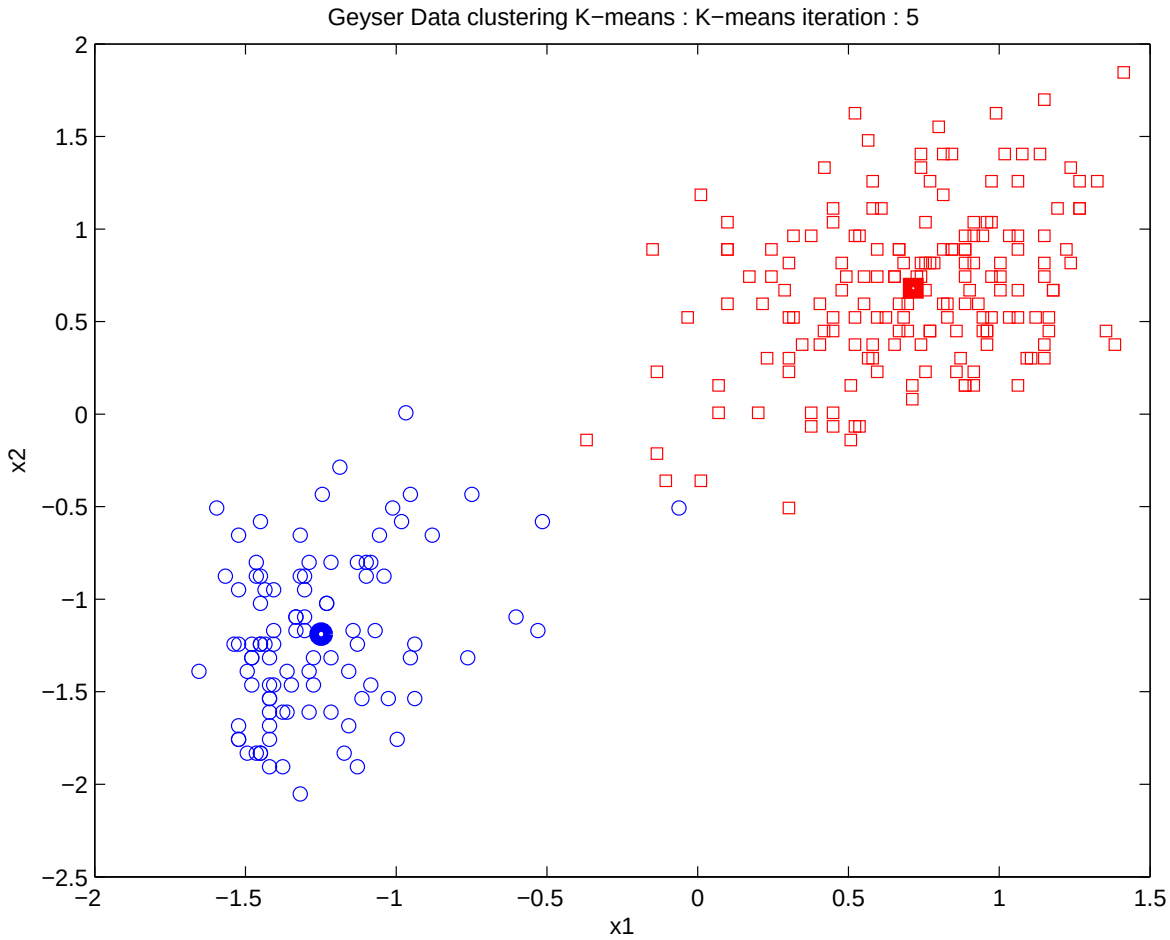
Illustration



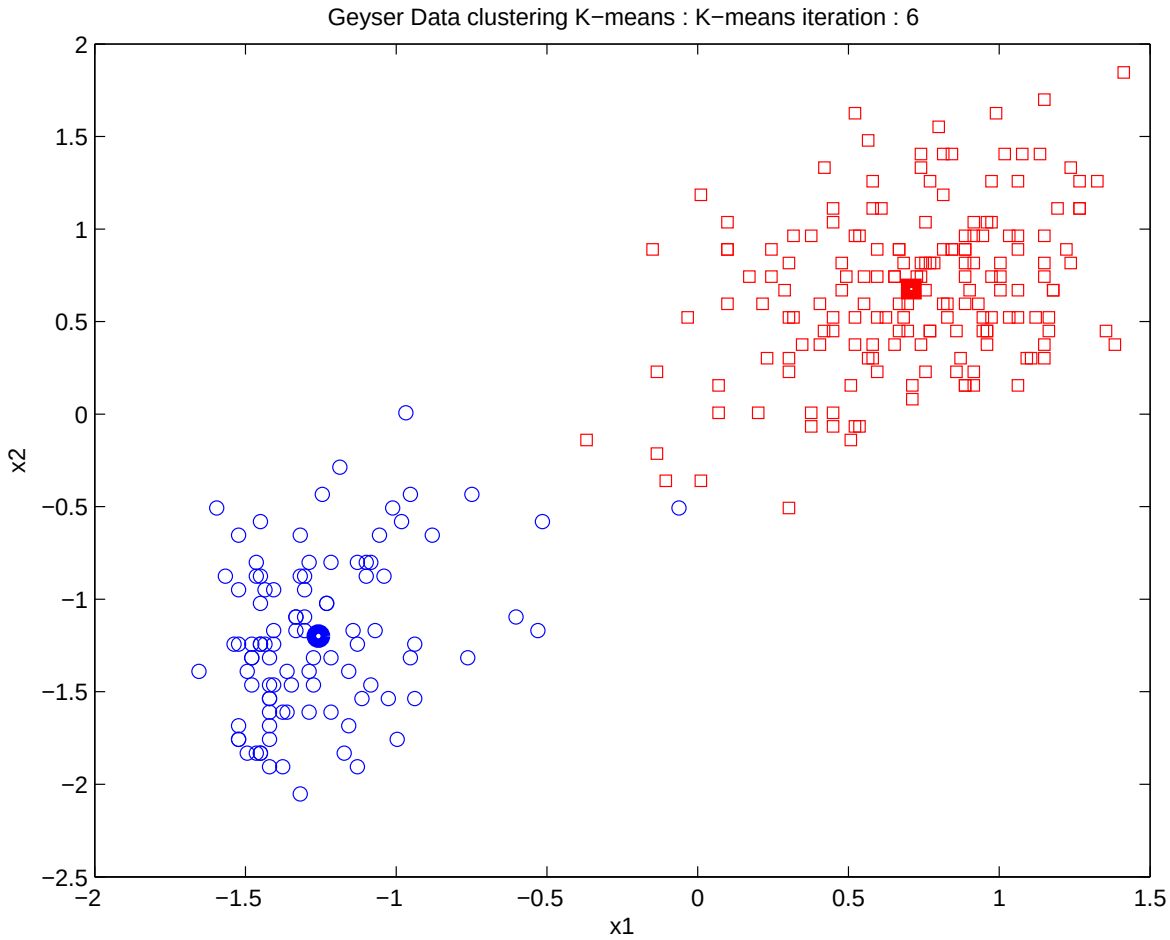
Illustration



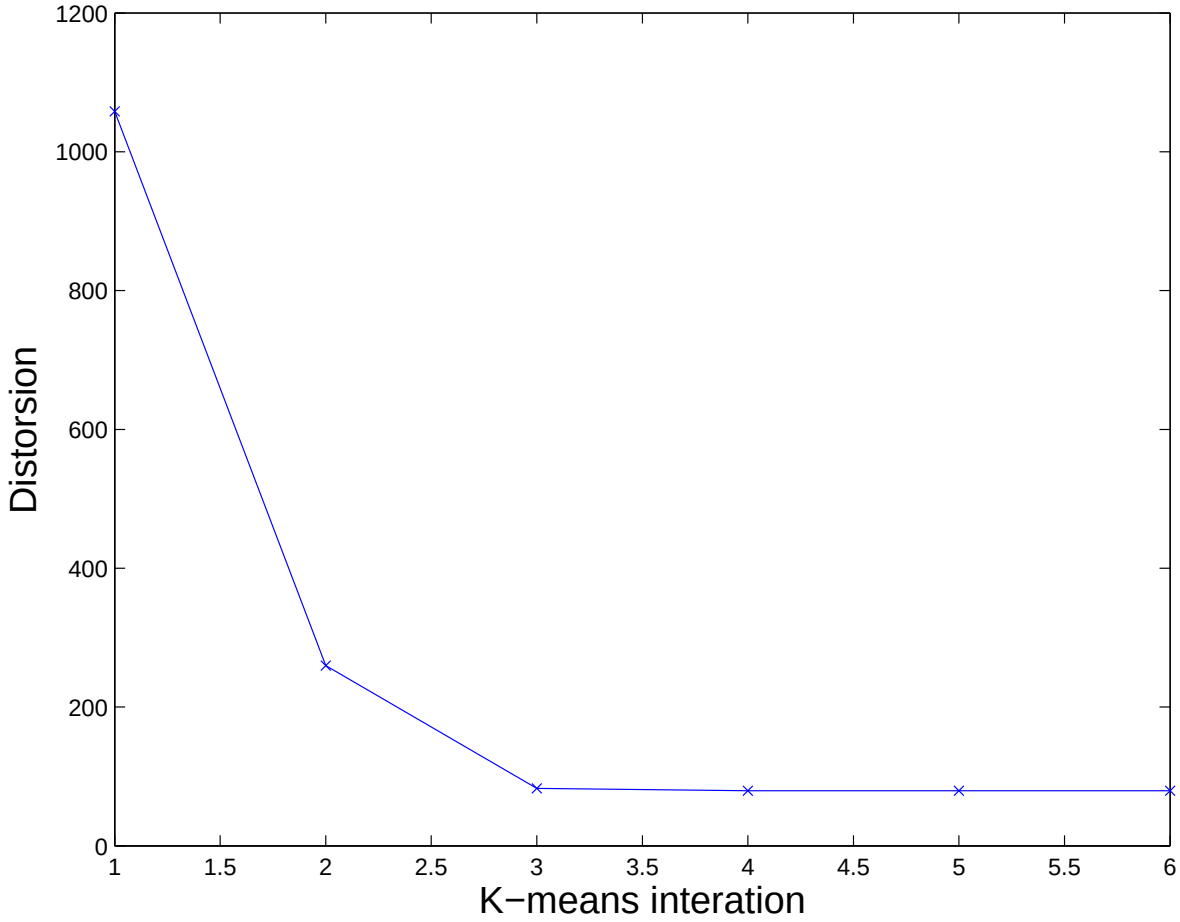
Illustration



Illustration



Illustration



Clustering via finite mixture models

- In the previous section we saw the main common partition-based clustering algorithm, that is K-means.
- Now we describe general clustering methods based on finite mixture models.
- $\Rightarrow$ This approach is known as the *model-based clustering*
- The clustering problem is reformulated as a density estimation problem
- the data probability density function is assumed to be a mixture density, each component density being associated with a cluster.
- $\Rightarrow$ The problem of clustering becomes the one of estimating the parameters of the assumed mixture model (e.g, estimating the means and the covariances for Gaussian mixtures).

Mixture approach/Classification approach

Two main approaches are possible. The former is refereed to as *the mixture approach* or *the estimation approach* and the latter is known as *the classification approach.*

1. **The mixture approach** consists of two steps :
   1. The parameters of the mixture density are estimated by maximizing the *observed-data likelihood* generally via the EM algorithm
   2. After performing the probability density estimation, the posterior probabilities τ_{ik} are then used to determine the cluster memberships through the MAP principle.
2. **The classification approach**
   - consists in optimizing a classification likelihood function which is (can be) the *complete-data likelihood* by using the CEM algorithm Celeux and Govaert (1992).
   - The cluster memberships and the model parameters are estimated simultaneously as the learning proceeds.

Classification EM (CEM) algorithm

- we saw that EM computes the maximum likelihood (ML) estimate of a mixture model.
- The Classification EM (CEM) algorithm Celeux and Govaert (1992) estimates both the mixture model parameters and the classes' labels by maximizing the completed-data log-likelihood $\mathcal{L}_c(\boldsymbol{\Psi}; \mathbf{X}, \mathbf{z}) = \log p(\mathbf{X}, \mathbf{z}; \boldsymbol{\Psi})$
- start with an initial parameter $\boldsymbol{\Psi}^{(0)}$

1. **Step 1 :** Compute the missing data $\mathbf{z}^{(q+1)}$ given the observations and the current estimated model parameters $\boldsymbol{\Psi}^{(q)}$:

$$\mathbf{z}^{(q+1)} = \arg\max_{\mathbf{z} \in \mathcal{Z}^n} \mathcal{L}_c(\boldsymbol{\Psi}^{(q)}; \mathbf{X}, \mathbf{z})$$

2. **Step 2 :** Compute the model parameters update $\boldsymbol{\Psi}^{(q+1)}$ by maximizing the complete-data log-likelihood given the current estimation of the missing data $\mathbf{z}^{(q+1)}$:

$$\boldsymbol{\Psi}^{(q+1)} = \arg\max_{\boldsymbol{\Psi} \in \boldsymbol{\Omega}} \mathcal{L}_c(\boldsymbol{\Psi}; \mathbf{X}, \mathbf{z}^{(q+1)}).$$

CEM for GMMs

- the CEM algorithm, for the case of mixture models, is equivalent to integrating a classification step (C-step) between the E- and the M-steps of the EM algorithm.
- The C-step assigns the observations to the component densities by using the MAP rule :

1. **E-step :** Compute the conditional posterior probabilities $\tau_{ik}^{(q)}$ that the observation $\mathbf{x}_i$ arises from the kth component density.
2. **C-step :** Assign each observation $\mathbf{x}_i$ to the component maximizing the conditional posterior probability τ_{ik} :

$$z_i^{(q+1)} = \arg\max_{k \in \mathcal{Z}} \tau_{ik}^{(q)} \quad (i = 1, \ldots, n).$$

   $\Rightarrow$ this step provides a hard partition of the data
3. **M-step :** Update the mixture model parameters by maximizing the completed-data log-likelihood for the partition provided by the C-step.

Algorithm 3 Pseudo code of the CEM algorithm for GMMs.

Inputs : a data set $\mathbf{X}$ and the number of clusters K

fix a threshold $\epsilon > 0$; set $q \leftarrow 0$ (iteration)

Initialize : $\boldsymbol{\Psi}^{(\mathbf{0})} = (\pi_{\mathbf{1}}^{(\mathbf{0})}, \ldots, \pi_K^{(\mathbf{0})}, \boldsymbol{\Psi}_{\mathbf{1}}^{(\mathbf{0})}, \ldots, \boldsymbol{\Psi}_K^{(\mathbf{0})})$ with $\boldsymbol{\Psi}_k^{(\mathbf{0})} = (\boldsymbol{\mu}_K^{(\mathbf{0})}, \boldsymbol{\Sigma}_K^{(\mathbf{0})})$

while increment in the complete-data log-likelihood $> \epsilon$ **do**

E-step :

for $k = 1, \ldots, K$ **do**

Compute $\tau_{ik}^{(q)} == \dfrac{\pi_k \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k^{(q)}, \boldsymbol{\Sigma}_k^{(q)})}{\sum_{\ell=\mathbf{1}}^{K} \pi_\ell \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_\ell^{(q)}, \boldsymbol{\Sigma}_\ell^{(q)})}$

end for

C-step :

for $k = 1, \ldots, K$ **do**

Compute $z_i^{(q)} = \arg\max_{k \in \mathcal{Z}} \tau_{ik}^{(q)}$ for $i = 1, \ldots, n$

Set $z_{ik}^{(q)} = 1$ if $z_i^{(q)} = k$ and $z_{ik}^{(q)} = 0$ otherwise, for $i = 1, \ldots, n$

end for

M-step :

for $k = 1, \ldots, K$ **do**

Compute $\pi_k^{(q+\mathbf{1})} = \dfrac{\sum_{i=\mathbf{1}}^{n} z_{ik}^{(q)}}{n}$

Compute $\boldsymbol{\mu}_k^{(q+\mathbf{1})} = \dfrac{\mathbf{1}}{n_k^{(q)}} \sum_{i=\mathbf{1}}^{n} z_{ik}^{(q)} \mathbf{x}_i$

Compute $\boldsymbol{\Sigma}_k^{(q+\mathbf{1})} = \dfrac{\mathbf{1}}{n_k^{(q)}} \sum_{i=\mathbf{1}}^{n} z_{ik}^{(q)} (\mathbf{x}_i - \boldsymbol{\mu}^{(q+\mathbf{1})})(\mathbf{x}_i - \boldsymbol{\mu}^{(q+\mathbf{1})})^T$

end for

$q \leftarrow q + 1$

end while

Output : $\hat{\boldsymbol{\Psi}} = \boldsymbol{\Psi}^{(q)}$; $\hat{z}_i = z_i^{(q)}$ $(i = 1, \ldots, n)$

CEM algorithm

- CEM is easy to implement, typically faster to converge than EM and monotonically improves the complete-data log-likelihood as the learning proceeds.
- converges toward a local maximum of the complete-data log-likelihood
- ! CEM provides biased estimates of the mixture model parameters. Indeed, CEM updates the model parameters from a truncated sample contrary to EM for which the model parameters are updated from the whole data through the fuzzy posterior probabilities and therefore the parameter estimations provided by EM are more accurate.
- **link with K-means** :
  - It can be shown that CEM which is formulated in a probabilistic framework, generalizes K-means
  - From a probabilistic point of view, K-means is equivalent to a particular case of the CEM algorithm for a mixture of K Gaussian densities with the same proportions $\pi_k = \frac{1}{K}\ \forall k$ and identical isotropic covariance matrices $\mathbf{\Sigma}_k = \sigma^2\mathbf{I}\ \forall k$.

Parsimonious Gaussian mixtures

- Parsimonious Gaussian mixture models are statistical models that allow for capturing a specific cluster shapes (e.g., clusters having the same shape or different shapes, spherical or elliptical clusters, etc).
- decompositions of the covariance matrices for the Gaussian mixture model :

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

where

  - λ_k represents the volume of the kth cluster (the amount of space of the cluster).
  - $\mathbf{D}_k$ is a matrix with columns corresponding to the eigenvectors of $\mathbf{\Sigma}_k$ that determines the orientation of the cluster.
  - $\mathbf{A}_k$ is a diagonal matrix, whose diagonal entries are the normalized eigenvalues of $\mathbf{\Sigma}_k$ arranged in a decreasing order and its determinant is 1. This matrix is associated with the shape of the cluster.

Parsimonious Gaussian mixtures

- This eigenvalue decomposition provides three main families of models : the spherical family, the diagonal family, and the general family and produces 14 different models, according to the choice of the configuration for the parameters λ_k, $\mathbf{A}_k$, and $\mathbf{D}_k$
- In addition to providing flexible statistical models for the clusters, parsimonious Gaussian mixture can be viewed as techniques for reducing the number of parameters in the model.
- imposing constraints on the covariance matrices reduces the dimension of the optimization problem.
- The EM algorithms therefore provide more accurate estimations compared to the full mixture model.

Model selection

- The problem of choosing the number of clusters can be seen as a model selection problem.
- The model selection task consists of choosing a suitable compromise between flexibility so that a reasonable fit to the available data is obtained, and over-fitting.
- A common way is to use a criterion (score function) that ensure the compromise.
- In general, we choose an overall score function that is explicitly composed of two components : a component that measures the goodness of fit of the model to the data, and a penalty component that governs the model complexity :

$$\text{score(model)} = \text{error(model)} + \text{penalty(model)}$$

which will be minimized.

Model selection

- The complexity of a model $\mathcal{M}$ is related to the number of its (free) parameters ν, the penalty function then involves the number of model parameters.
- Let $\mathcal{M}$ denote a model, $\mathcal{L}(\hat{\boldsymbol{\Psi}})$ its log-likelihood and ν the number of its free parameters. Consider that we fitted M different model structures $(\mathcal{M}_1, \ldots, \mathcal{M}_M)$, from which we wish to choose the "best" one (ideally the one providing the best prediction on future data).
- Assume we have estimated the model parameters $\hat{\boldsymbol{\Psi}}_m$ for each model structure $\mathcal{M}_m$ $(m = 1, \ldots, M)$ from a sample of n observations $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and now we wish to choose among these fitted models.

Model selection

- Akaike Information Criterion (AIC) :

$$\text{AIC}(\mathcal{M}_m) = \mathcal{L}(\hat{\boldsymbol{\Psi}}_m) - \nu_m$$

- Bayesian Information Criterion (BIC) :

$$\text{BIC}(\mathcal{M}_m) = \mathcal{L}(\hat{\boldsymbol{\Psi}}_m) - \frac{\nu_m \log(n)}{2}$$

- Integrated Classification Likelihood (ICL) :

$$\text{ICL}(\mathcal{M}_m) = \mathcal{L}_c(\hat{\boldsymbol{\Psi}}_m) - \frac{\nu_m \log(n)}{2}$$

where $\mathcal{L}_c(\hat{\boldsymbol{\Psi}}_m)$ is the complete-data log-likelihood for the model $\mathcal{M}_m$ and ν_m denotes the number of free model parameters. For example, in the case of a d-dimensional Gaussian mixture model we have :

$$\nu = \underbrace{(K-1)}_{\pi_k\text{'s}} + \underbrace{K \times d)}_{\{\boldsymbol{\mu}_k\}} + \underbrace{K \times \frac{d \times (d+1)}{2}}_{\{\boldsymbol{\Sigma}_k\}} = \frac{K \times (d+1) \times (d+2)}{2} - 1.$$

Examples

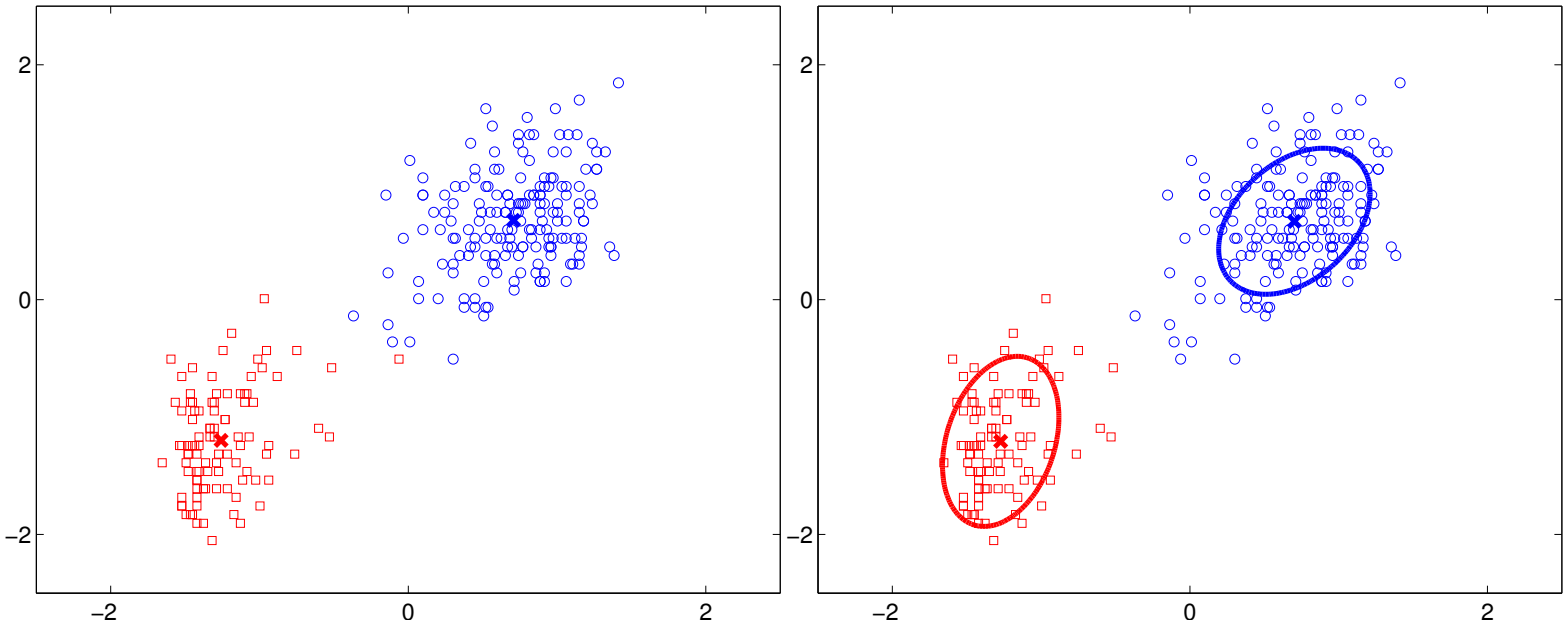


Figure – Clustering results obtained with K-means algorithm (left) with $K = 2$ and the EM algorithm (right). The cluster centers are shown by the red and blue crosses and the ellipses are the contours of the Gaussian component densities at level 0.4 estimated by EM. The number of clusters for EM have been chosen by BIC for $K = 1, \ldots, 4$.

Examples

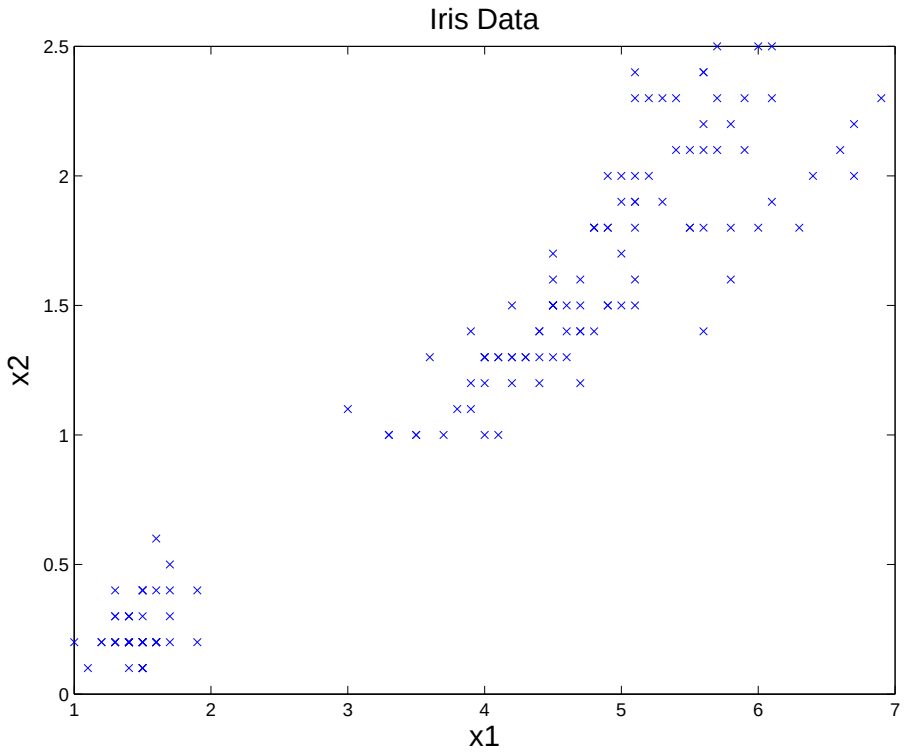


Figure – A three-class example of a real data set : Iris data of Fisher.

Examples

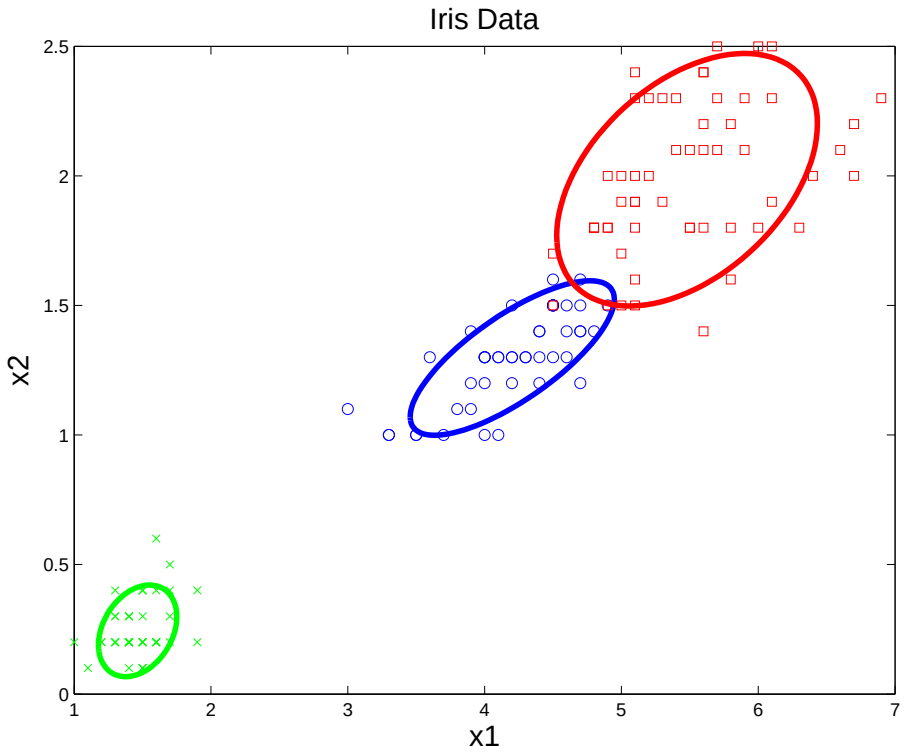


Figure – Iris data : Clustering results with EM for a GMM and AIC.

Examples

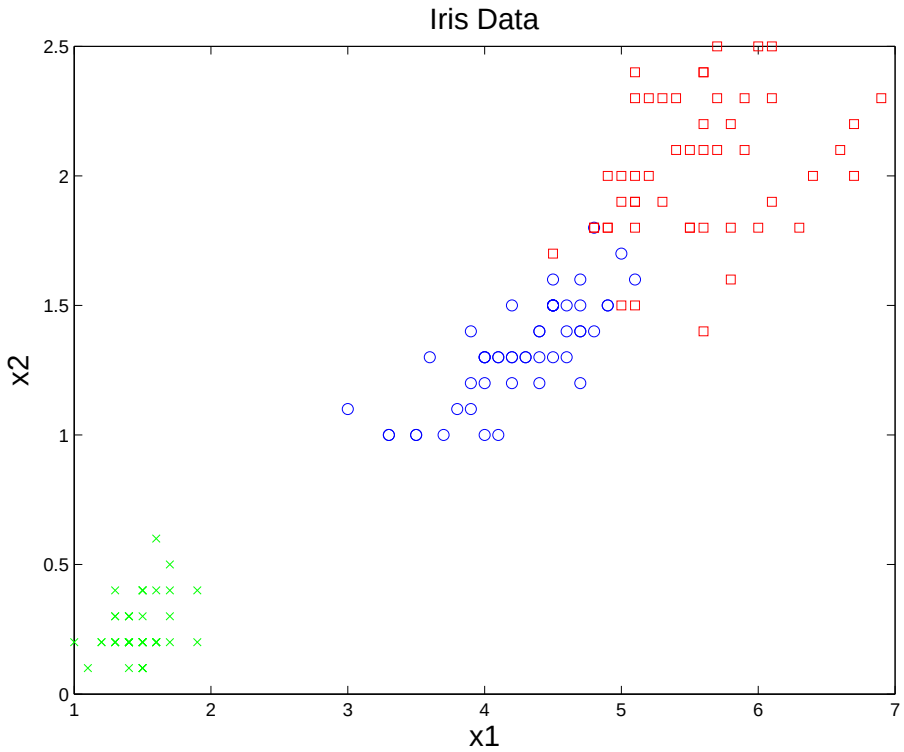
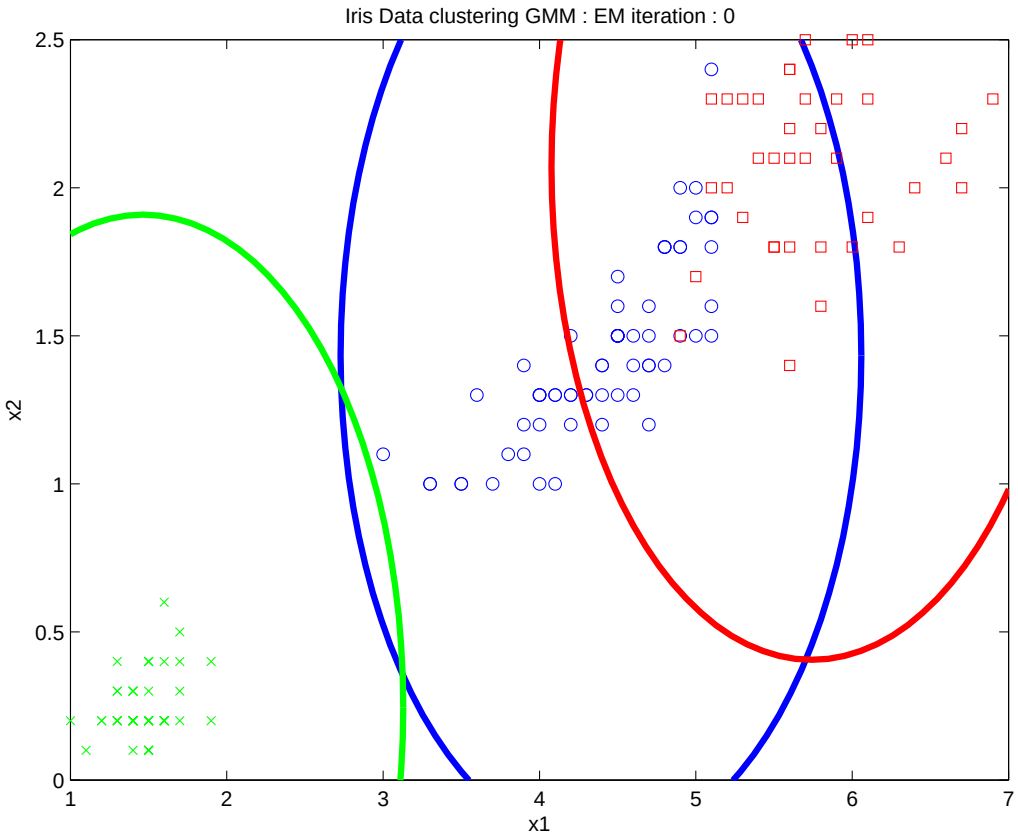
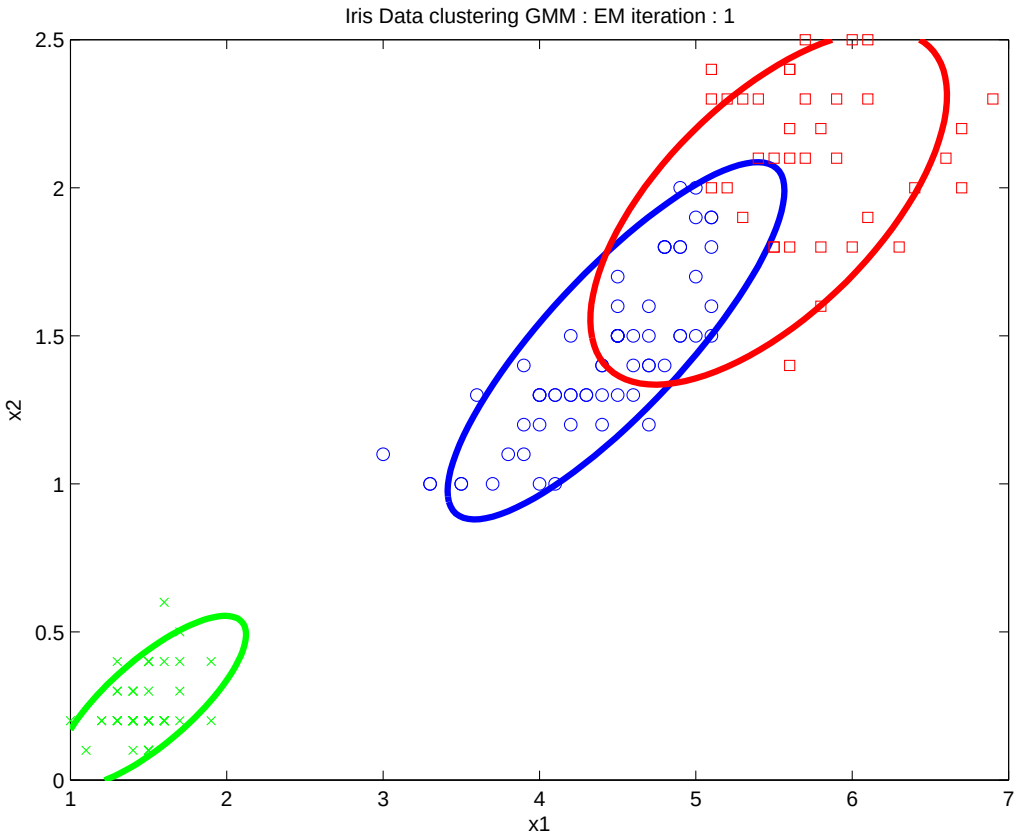


Figure – Iris data of Fisher : The data are colored according to the true partition.

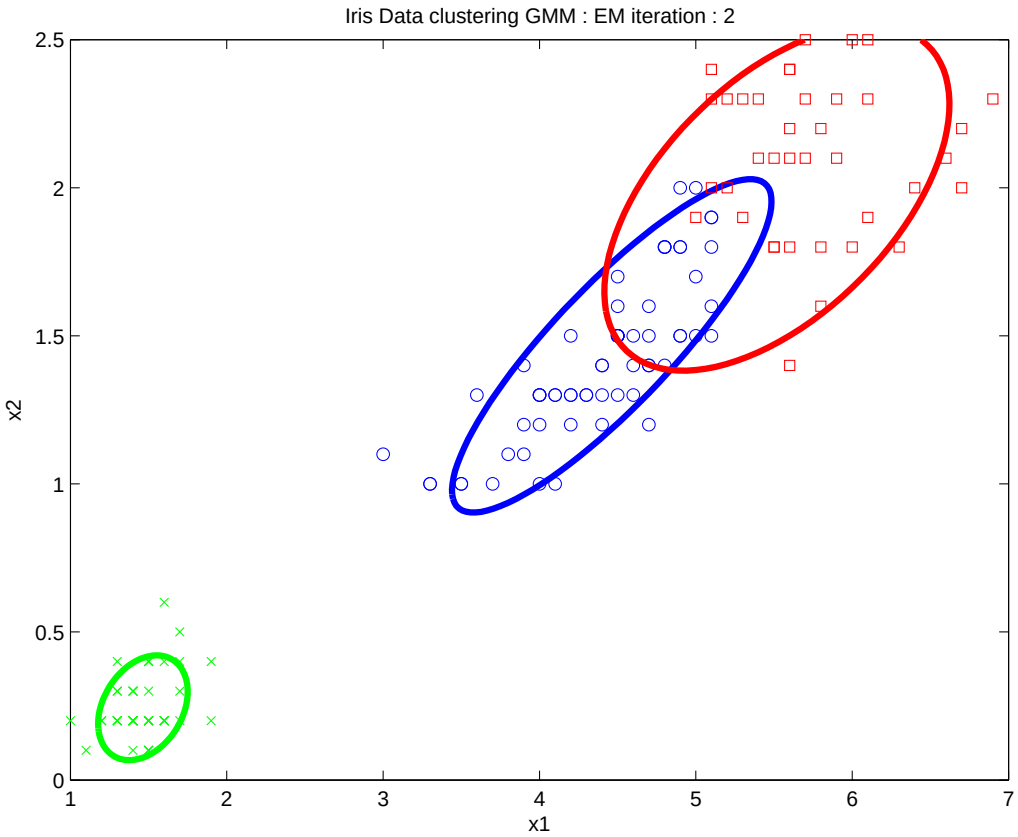
Examples


Iris Data clustering GMM : EM iteration : 0
x1
x2
1
2
3
4
5
6
7
0
0.5
1
1.5
2
2.5

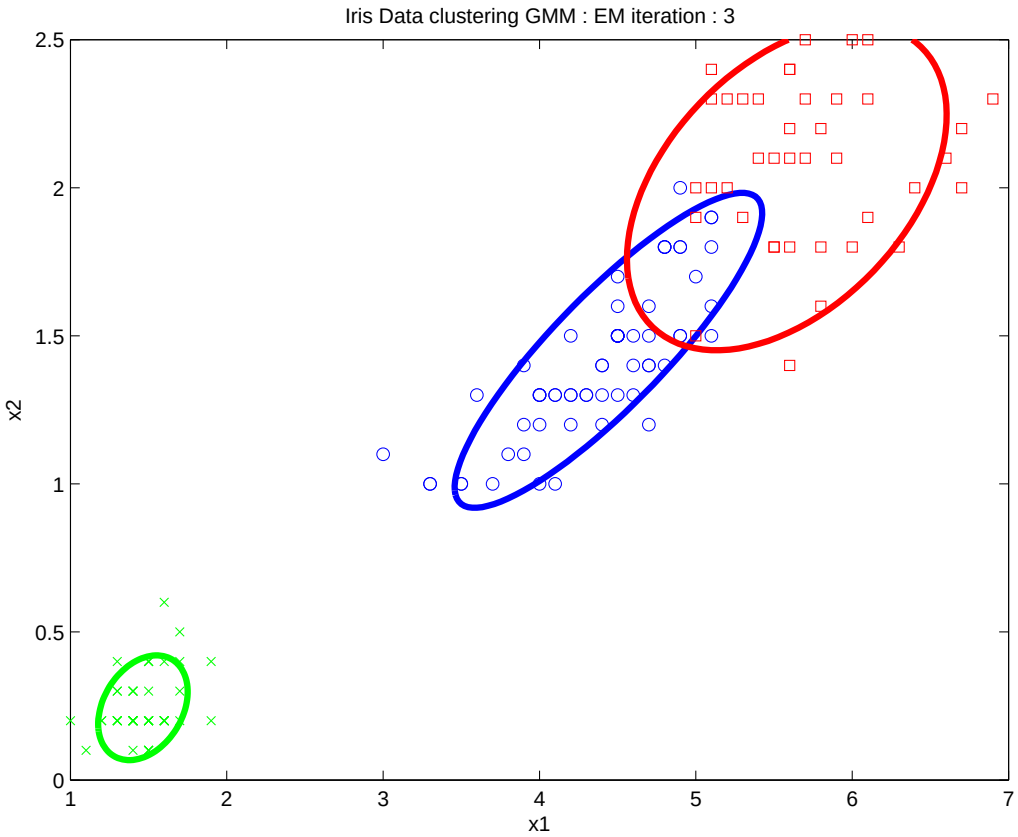
Examples


Iris Data clustering GMM : EM iteration : 1
2.5
2
1.5
1
0.5
0
x2
1
2
3
4
5
6
7
x1

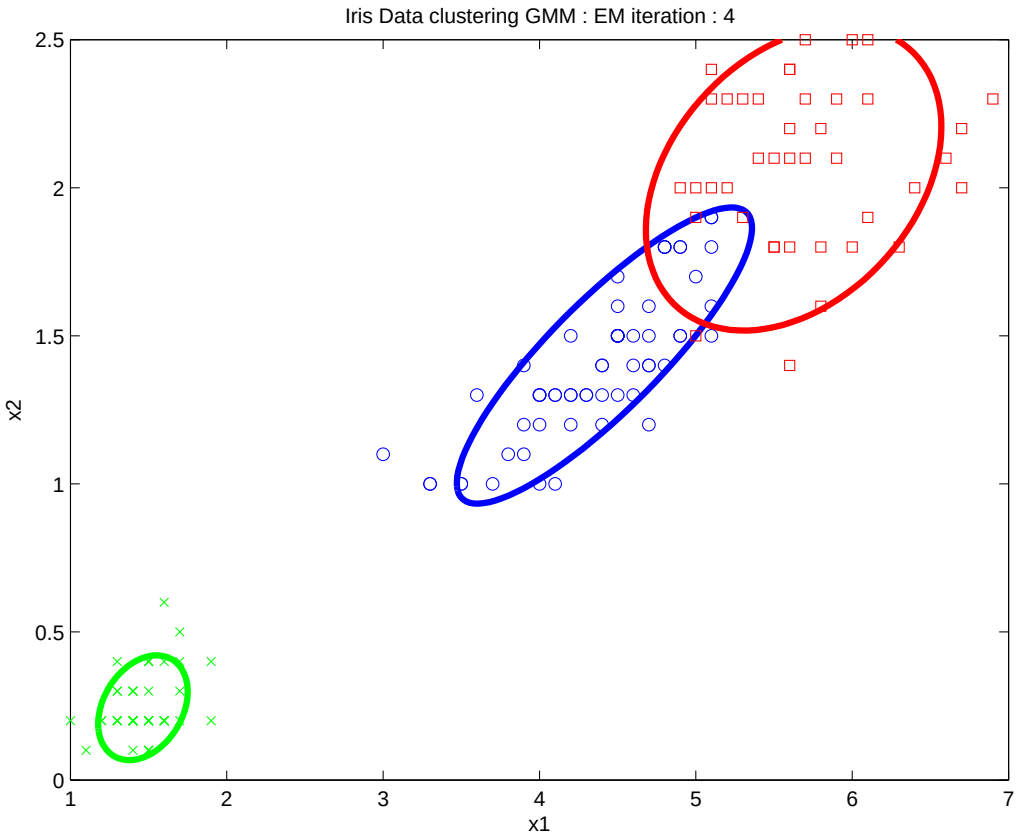
Examples


Iris Data clustering GMM : EM iteration : 2
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

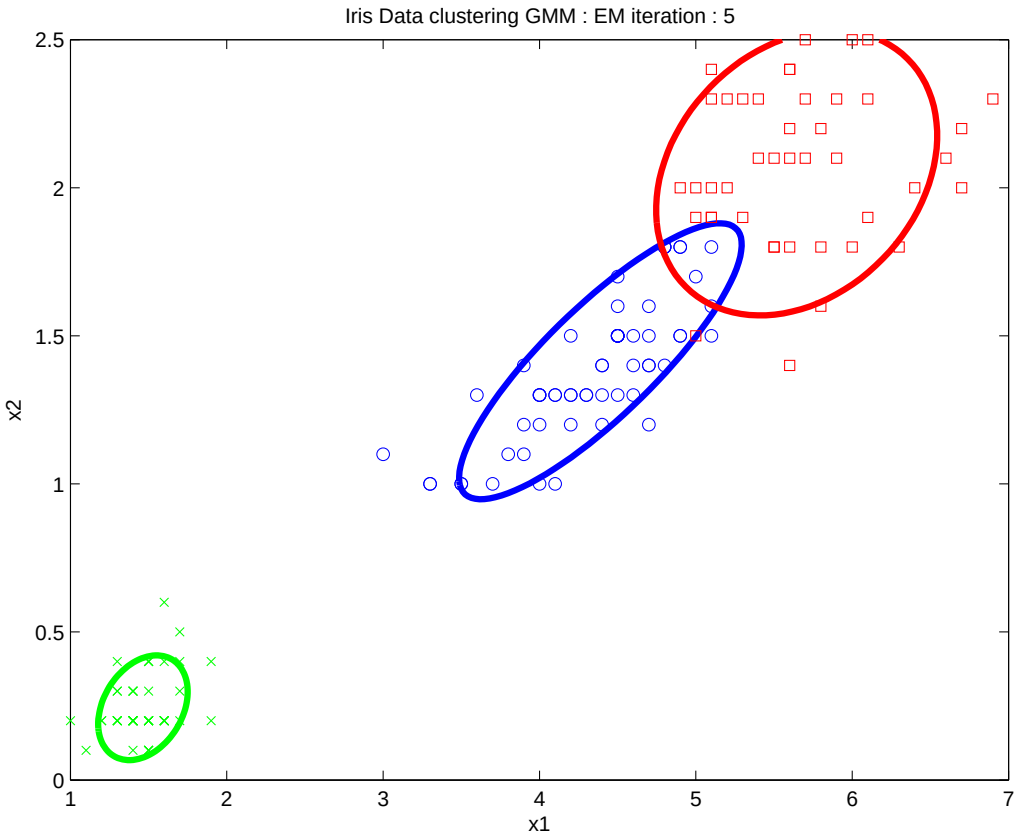
Examples


Iris Data clustering GMM : EM iteration : 3
2.5
2
1.5
1
0.5
0
x2
1
2
3
4
5
6
7
x1

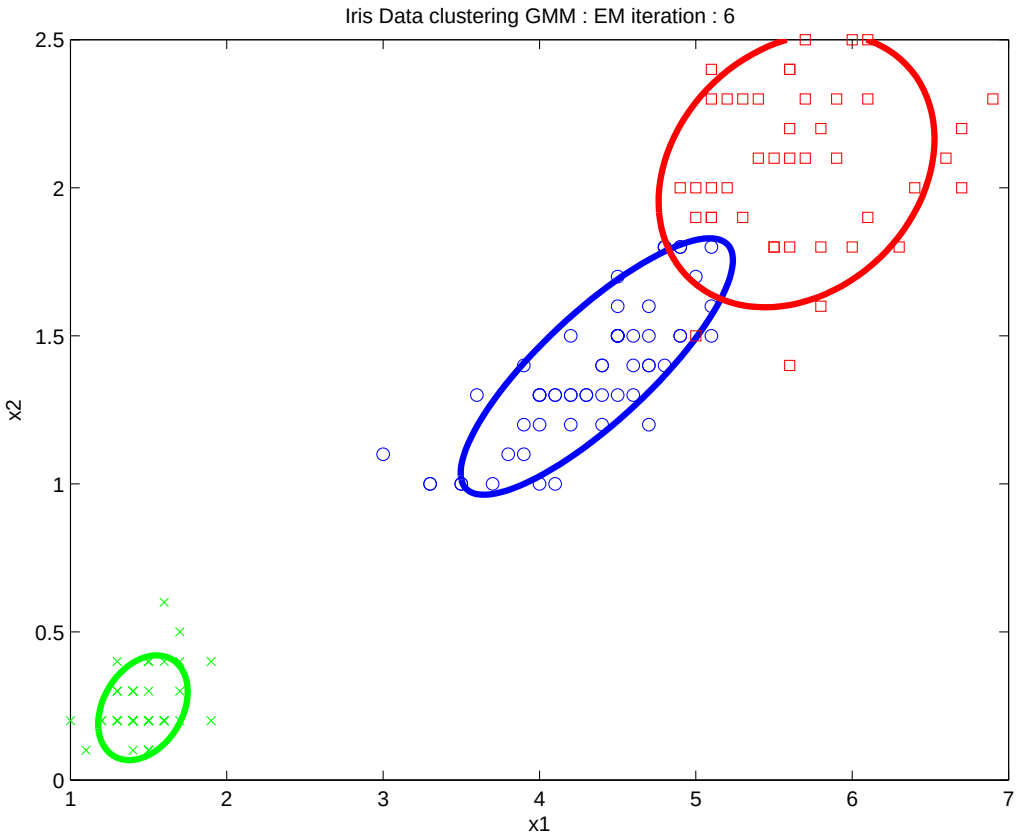
Examples


Iris Data clustering GMM : EM iteration : 4
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

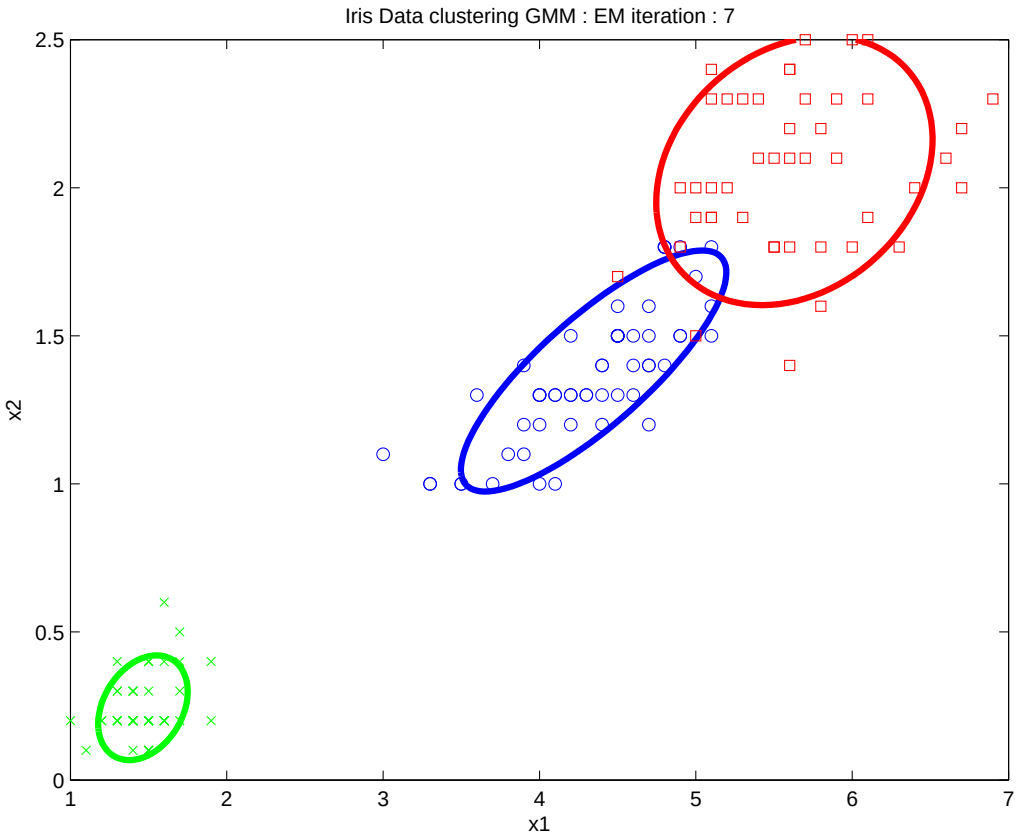
Examples


Iris Data clustering GMM : EM iteration : 5
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

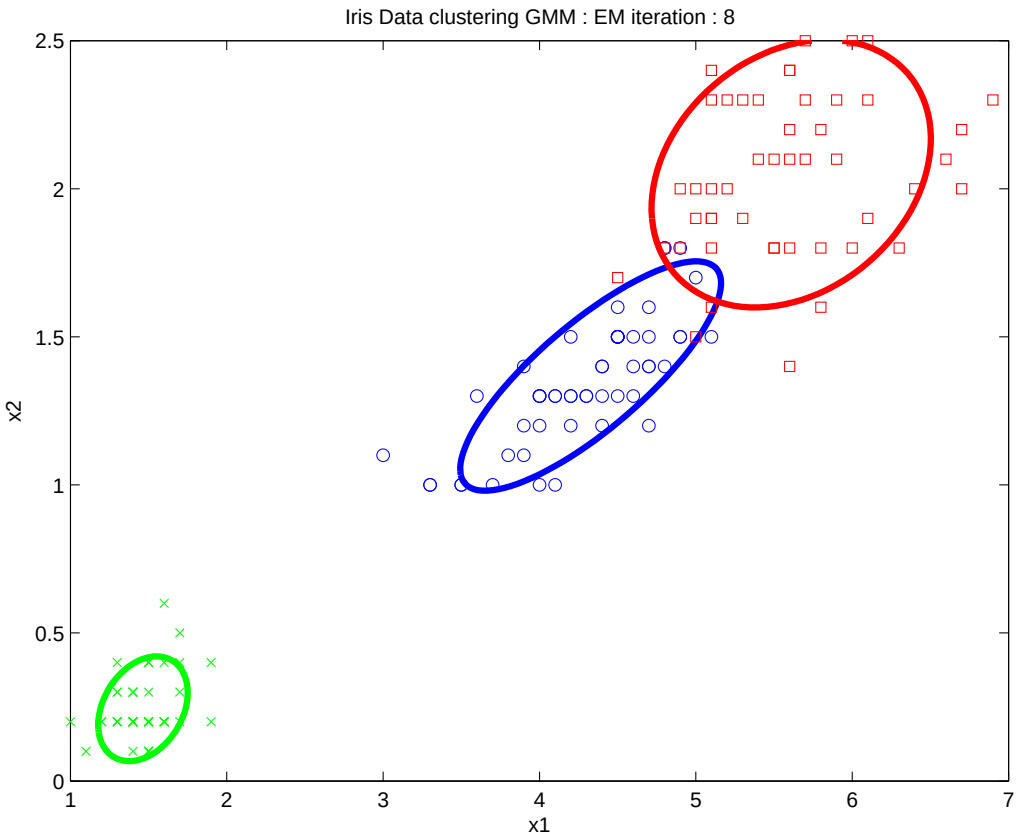
Examples


Iris Data clustering GMM : EM iteration : 6
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

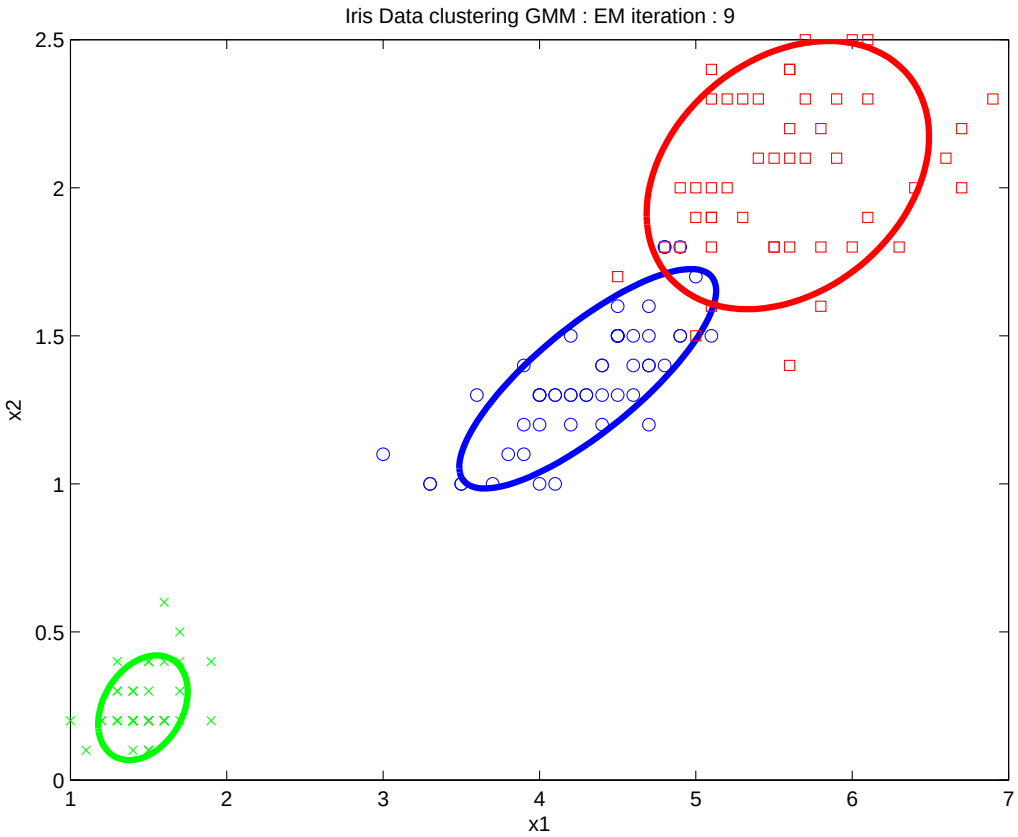
Examples


Iris Data clustering GMM : EM iteration : 7
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

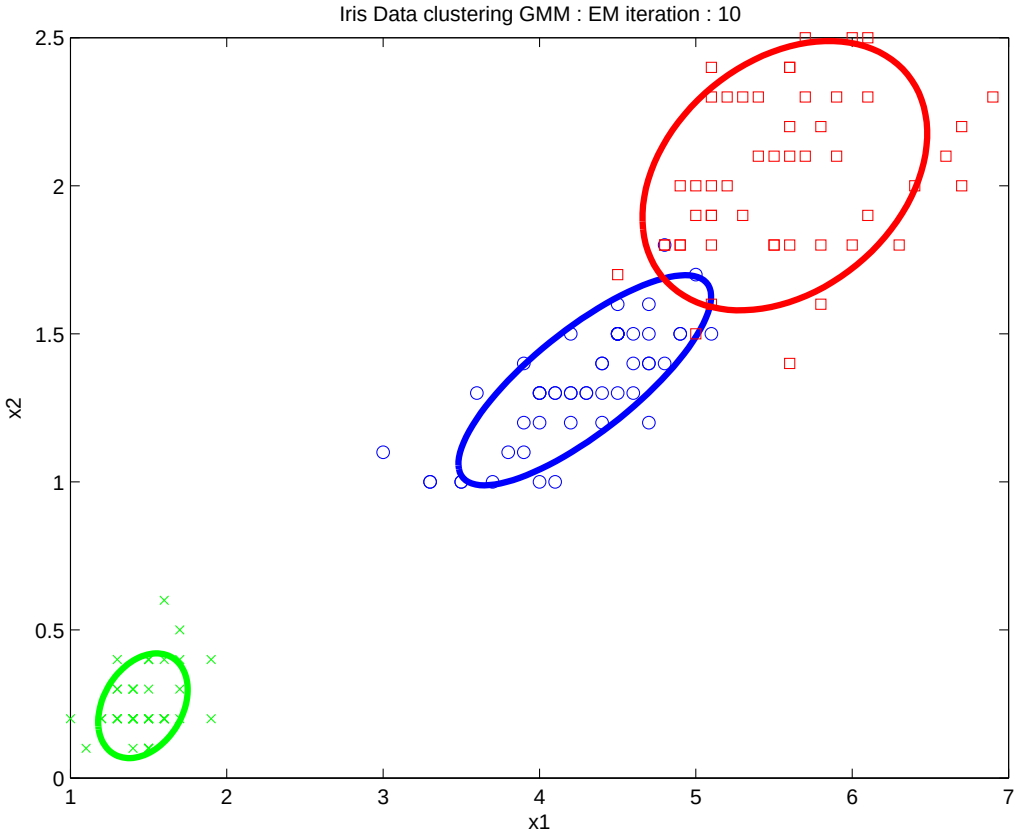
Examples


Iris Data clustering GMM : EM iteration : 8
2.5
2
1.5
1
0.5
0
x2
1
2
3
4
5
6
7
x1

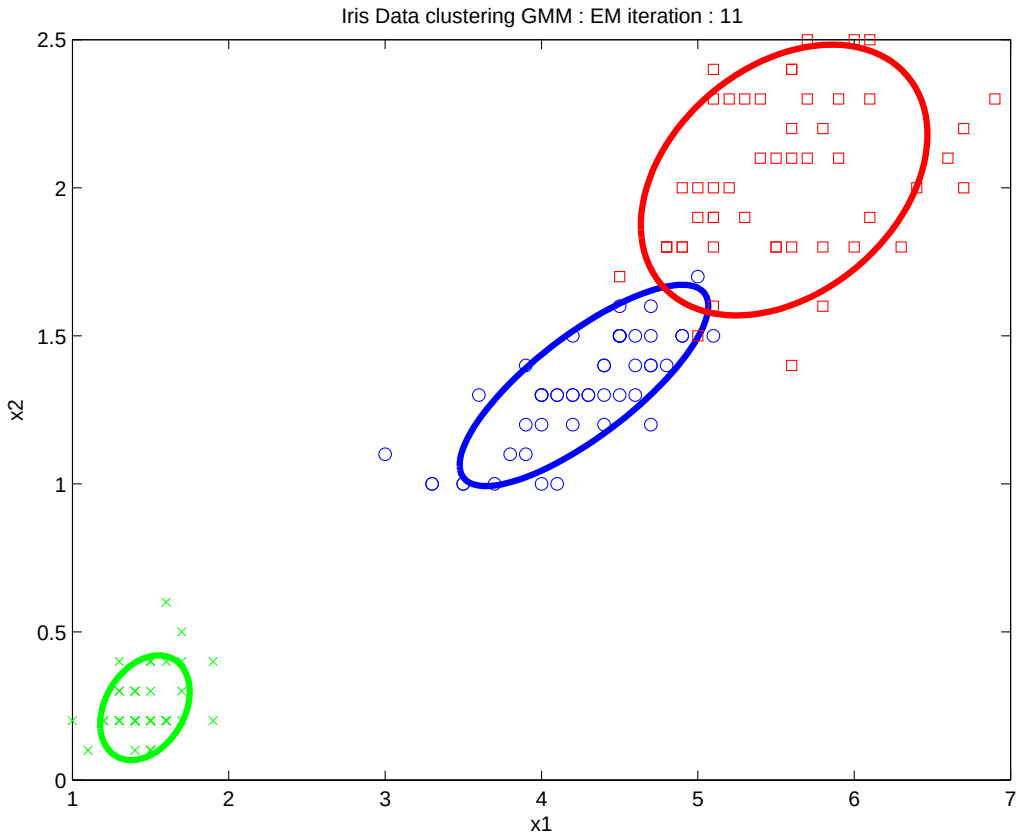
Examples


Iris Data clustering GMM : EM iteration : 9
x1
x2
0
0.5
1
1.5
2
2.5
1
2
3
4
5
6
7

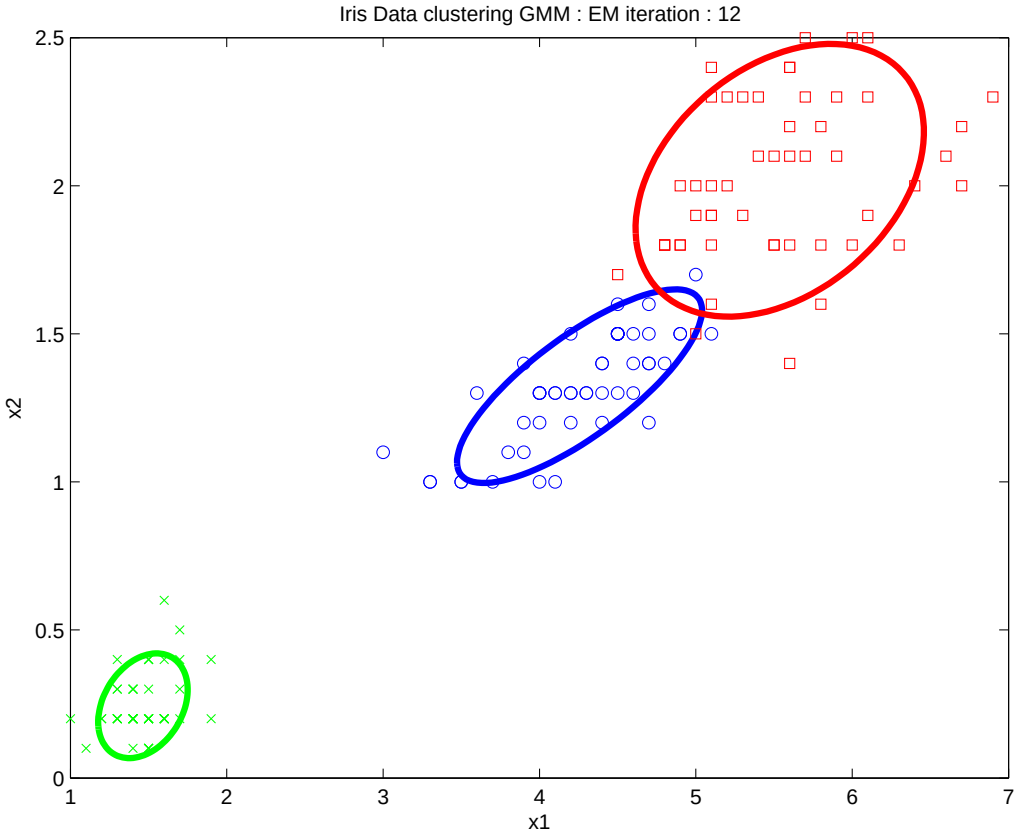
Examples


Iris Data clustering GMM : EM iteration : 10
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

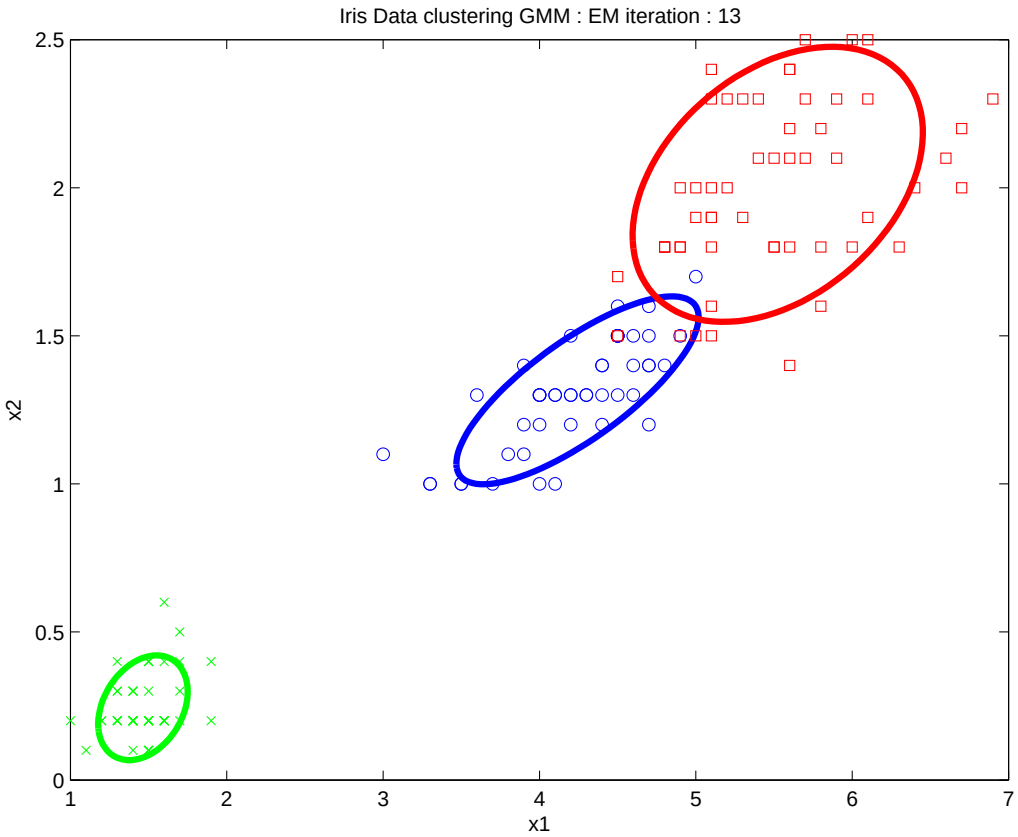
Examples


Iris Data clustering GMM : EM iteration : 11
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

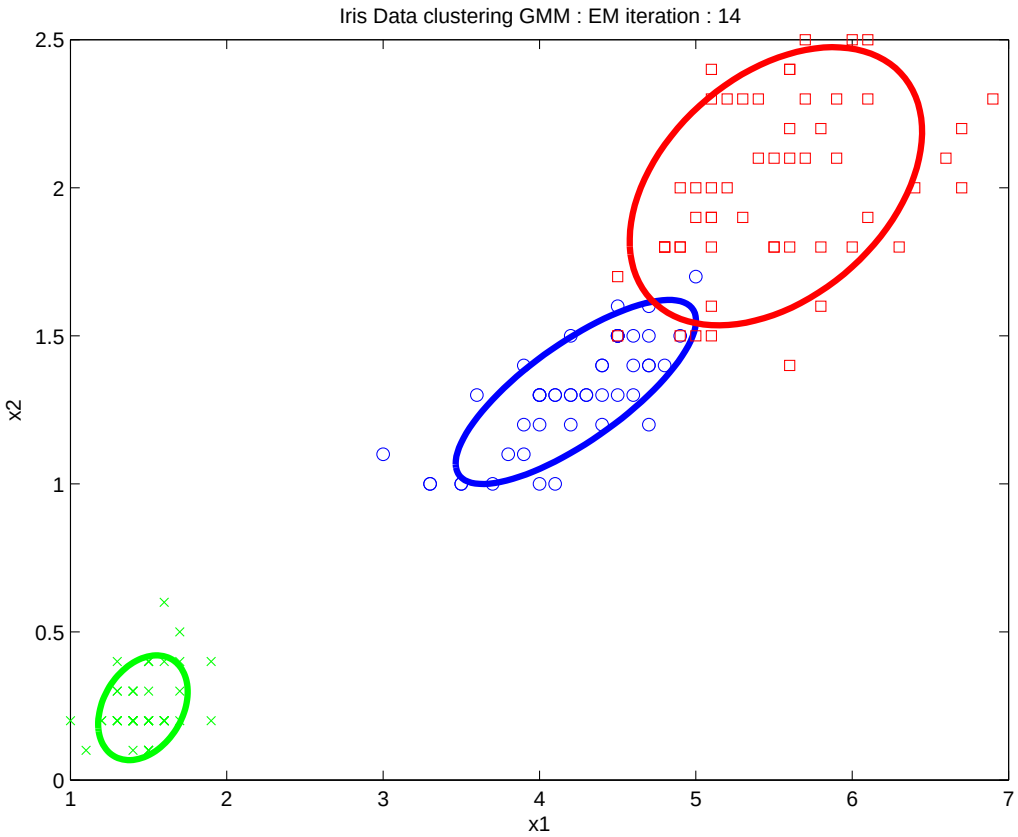
Examples


Iris Data clustering GMM : EM iteration : 12
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

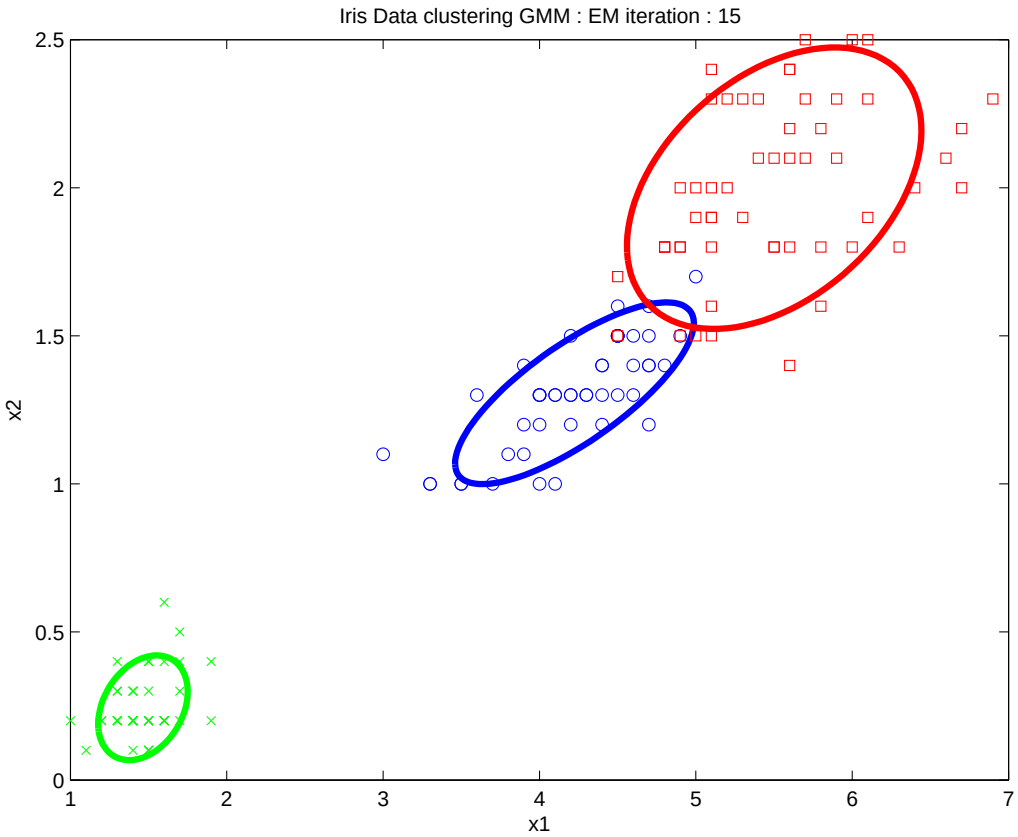
Examples


Iris Data clustering GMM : EM iteration : 13
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

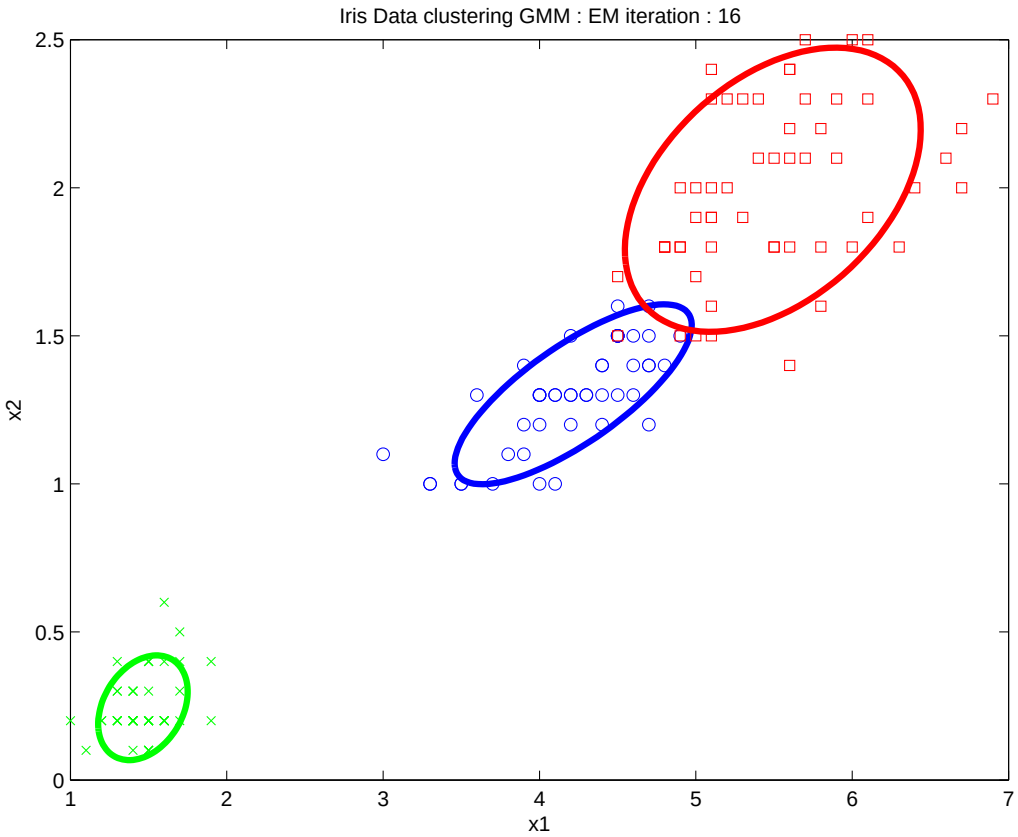
Examples


Iris Data clustering GMM : EM iteration : 14
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

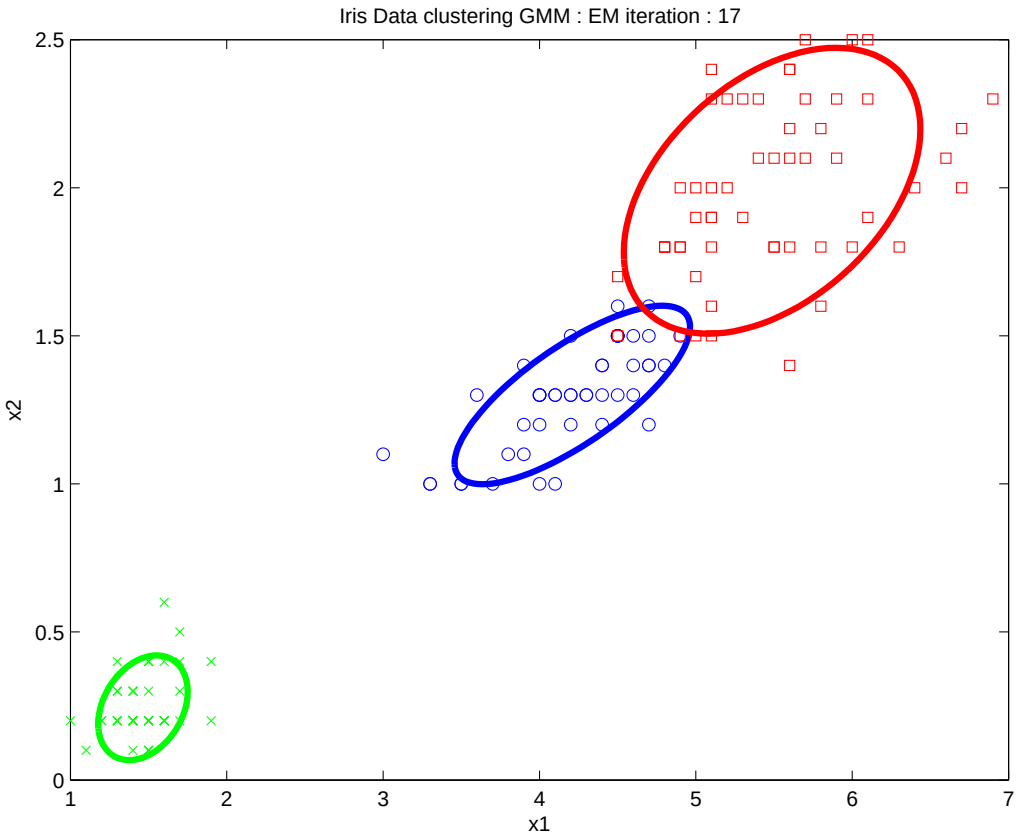
Examples


Iris Data clustering GMM : EM iteration : 15
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

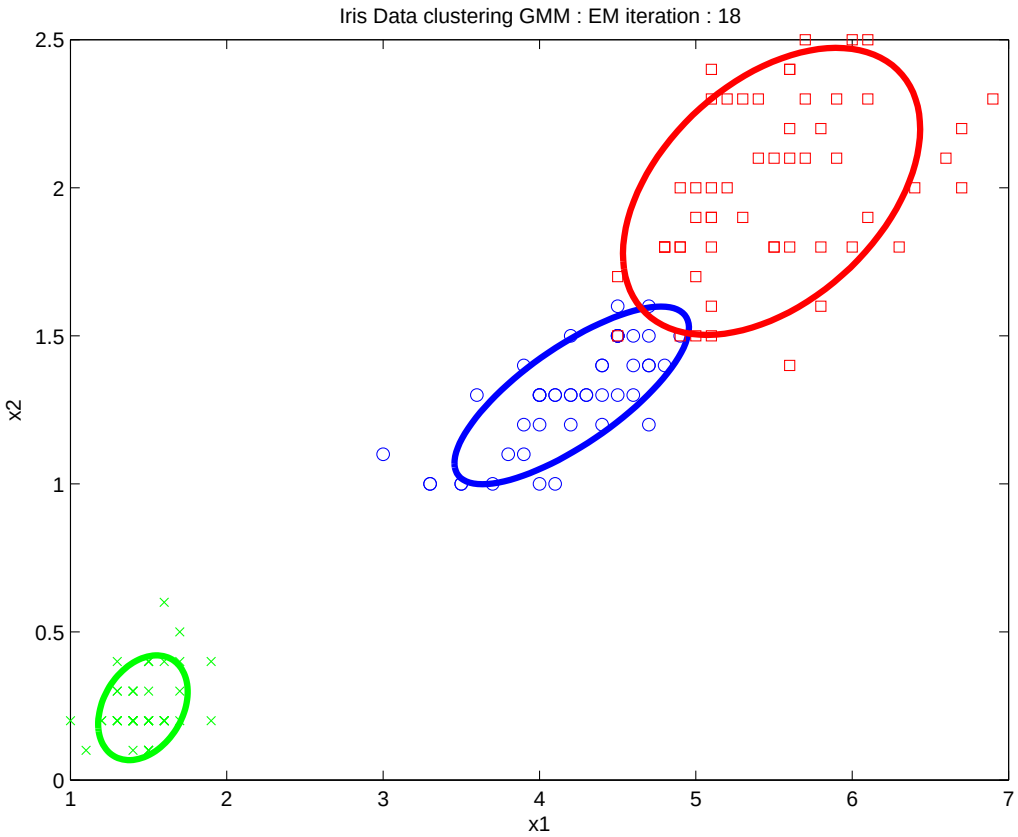
Examples


Iris Data clustering GMM : EM iteration : 16
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

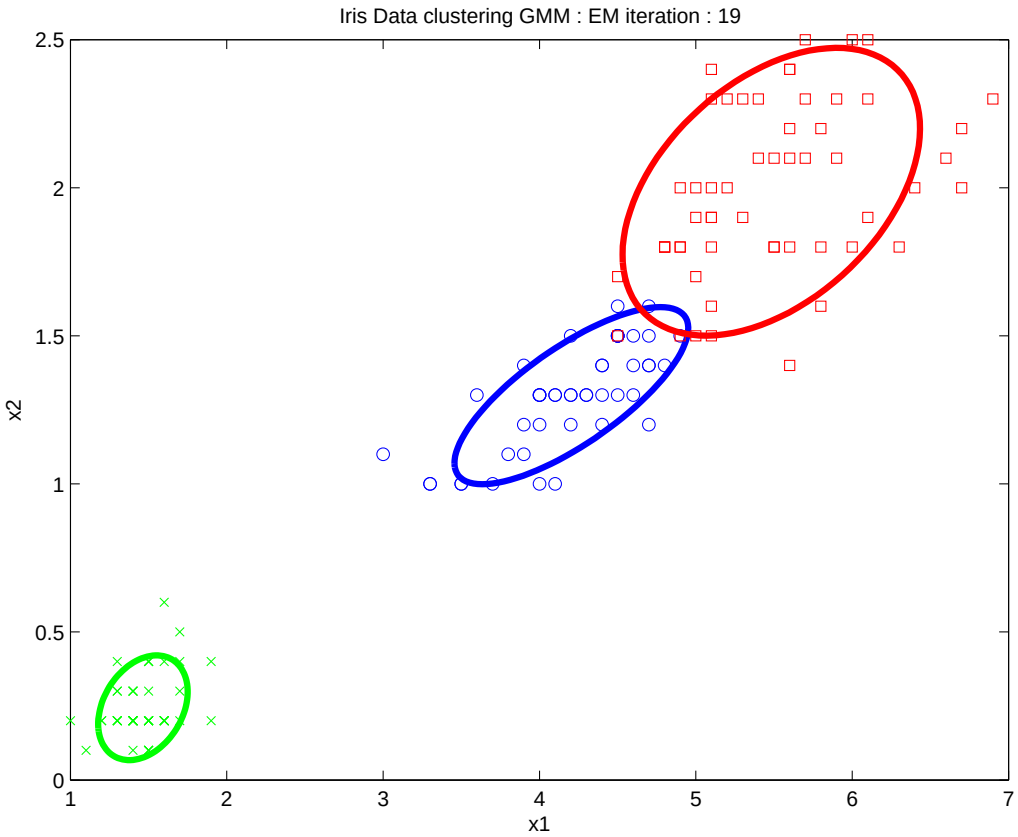
Examples


Iris Data clustering GMM : EM iteration : 17
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

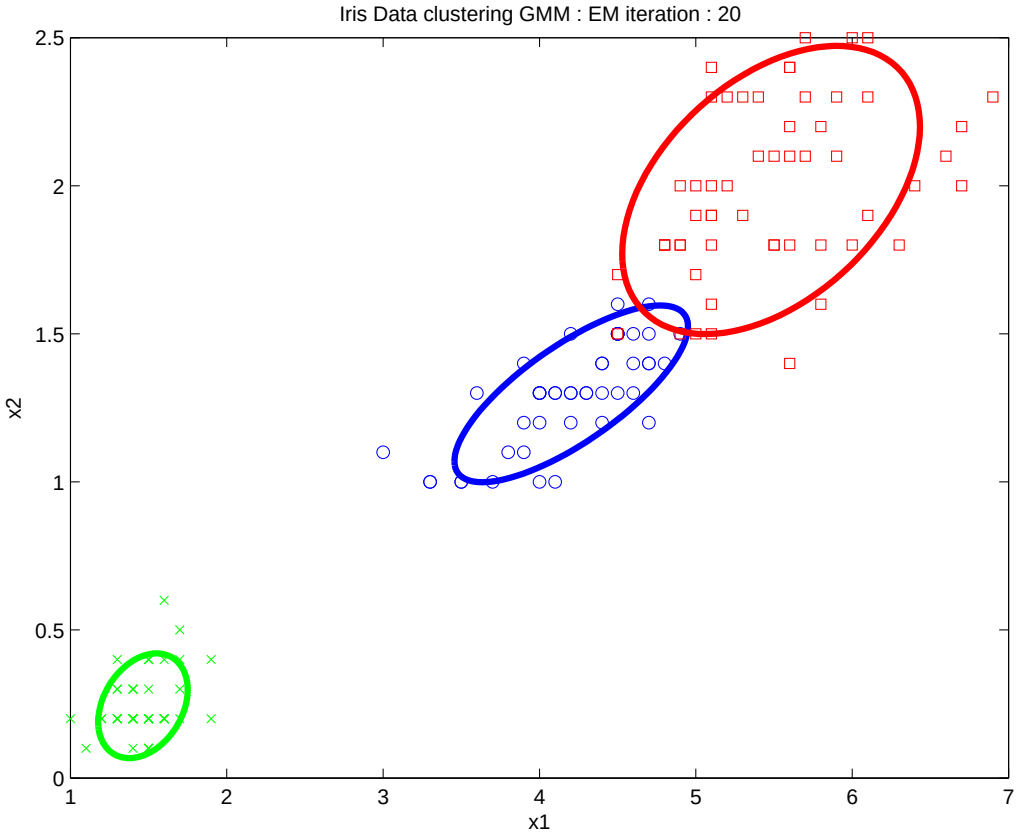
Examples


Iris Data clustering GMM : EM iteration : 18
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

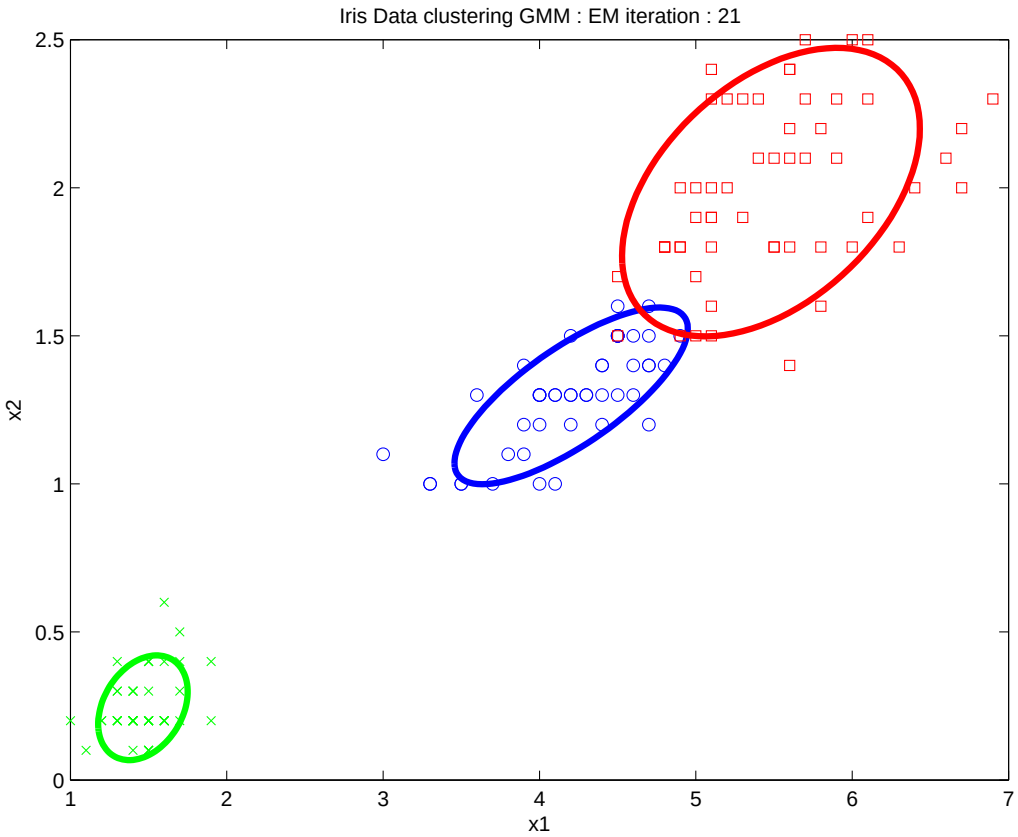
Examples


Iris Data clustering GMM : EM iteration : 19
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

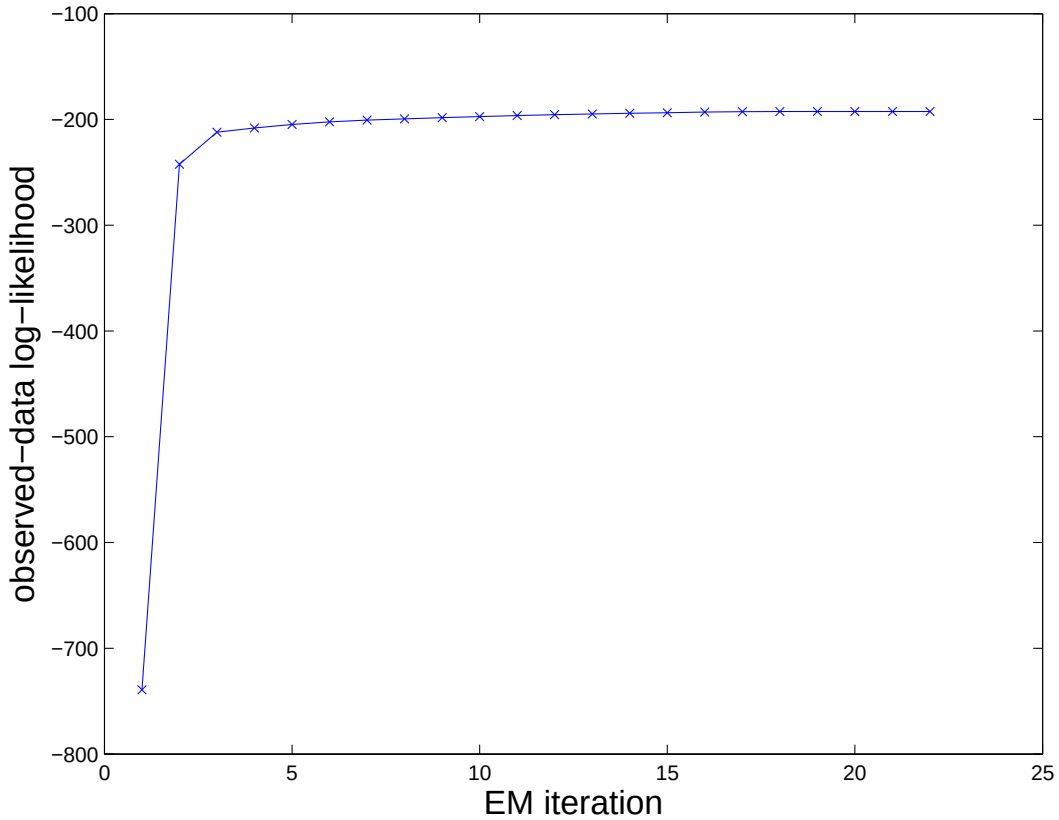
Examples


Iris Data clustering GMM : EM iteration : 20
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

Examples


Iris Data clustering GMM : EM iteration : 21
x2
x1
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7

Examples


−100
−200
−300
−400
−500
−600
−700
−800
0
5
10
15
20
25
observed−data log−likelihood
EM iteration

Topographic Learning for clustering/visualisation

- Self-Organizing Maps (SOMs)
- Generative Topographic Mapping (GTM)
- Formulation of the Generative Topographic Mapping (GTM)
- GTM Through Time
- Formulation of the GTM Through Time (GTM-TT)

Introduction I

Topographic learning has become a widely used approach for the analysis, dimensionality reduction, visualization of high-dimensional data.

One of the most used topographic approaches is the self-organizing map (SOM) (Kohonen, 2001) and its generative version : the Generative Topographic Mapping (GTM) (Bishop and Williams, 1998).

The SOM idea consists in an unsupervised learning approach based on artificial neural networks and was inspired from the competitive learning.

Competitive learning I

Competitive learning (Kong and Kosko, 1991) is an unsupervised adaptive process during which the neurons of a neural network (units) compete for the right to respond to a subset of the input data

Each unit (neuron) of the network gradually becomes specialized of a subset of the data

When an input is presented, the neuron that is best to represent it in the sense of a chosen similarity measure, typically an Euclidean distance, wins the competition and is allowed to "learn" from it, this is the well-known **"winner-take-all"** rule.

During the learning process, the neurons (units) **compete** for the right to respond to a subset of the input data and each unit of the network becomes specialized of a subset of the data.

Competitive learning II

When an input $\mathbf{x}_i$ is presented, the neuron that is best to represent it (in the sense of a chosen similarity measure, typically a distance (Euclidean)) wins the competition and it allowed to learn from it.

Only the winner neuron (also called the best matching unit (BMU)) is updated, that is :

$$\boldsymbol{\mu}_{z_i}^{(q+1)} = \boldsymbol{\mu}_{z_i}^{(q)} + \alpha^{(q)}(\mathbf{x}_i - \boldsymbol{\mu}_{z_i}^{(q)}) \tag{5}$$

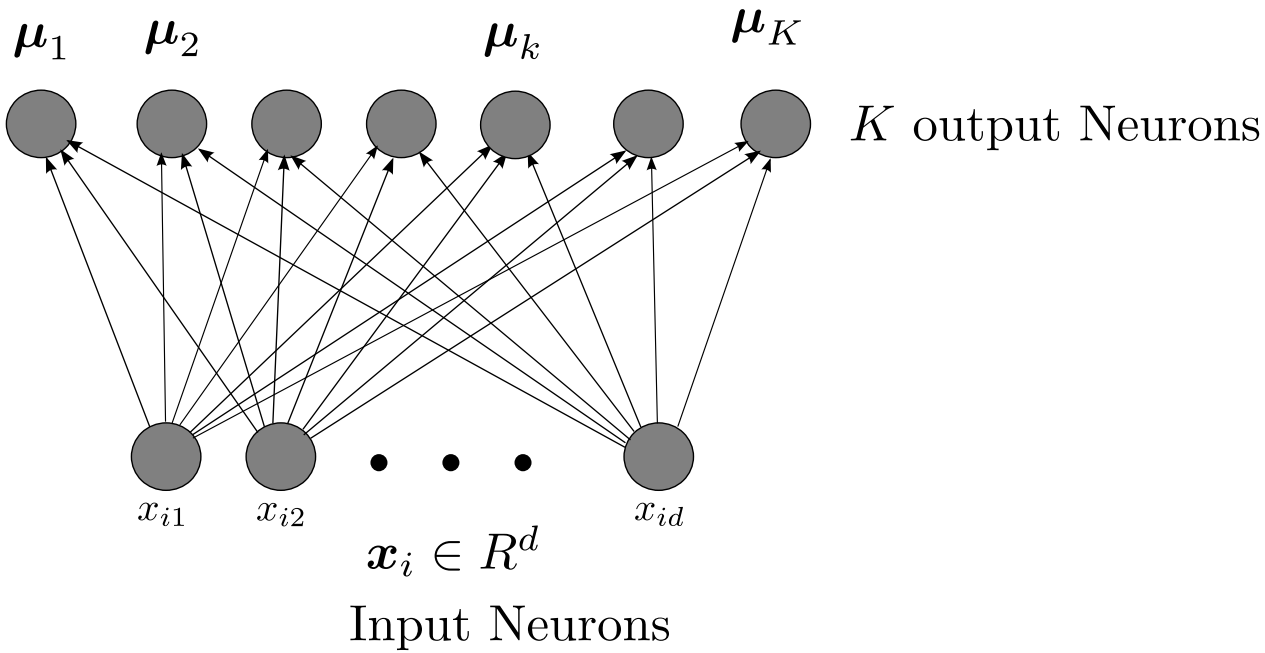
where

$$z_i = \arg\min_{k \in \mathcal{Z}} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 \tag{6}$$

is the index of the winner prototype and $0 < \alpha^{(q)} < 1$ is the learning rate which decreases monotonically as the learning proceeds.

Competitive learning III

$$\boldsymbol{\mu}_k = (\mu_{k1}, \ldots, \mu_{kd})' \in R^d \quad \begin{matrix} \text{prototype (vector of weights)} \\ \text{associated to neuron } k \end{matrix}$$

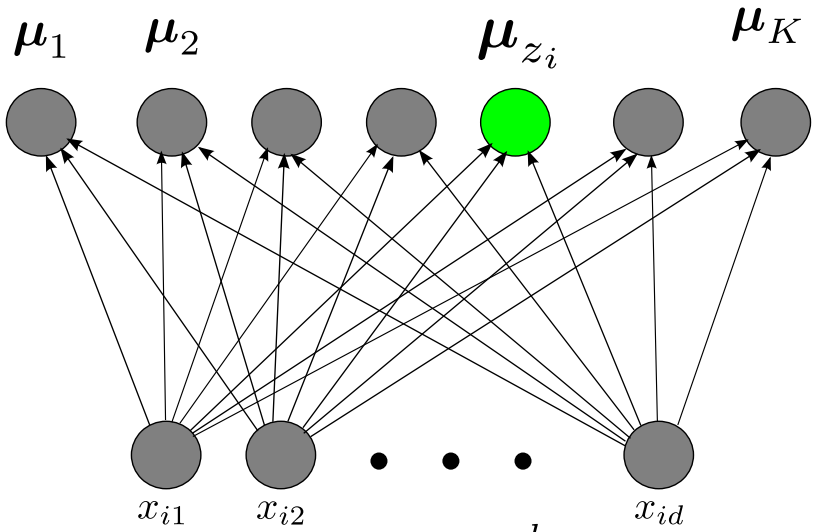


Competitive learning IV

Step1 : Determine the **winner neuron**

$$z_i = \arg \min_{1<k<K} \| \boldsymbol{x}_i - \boldsymbol{\mu}_k \|^2$$

z_i being the index of the **winner neuron** for $\boldsymbol{x}_i$



K output Neurons

$\boldsymbol{x}_i \in R^d$

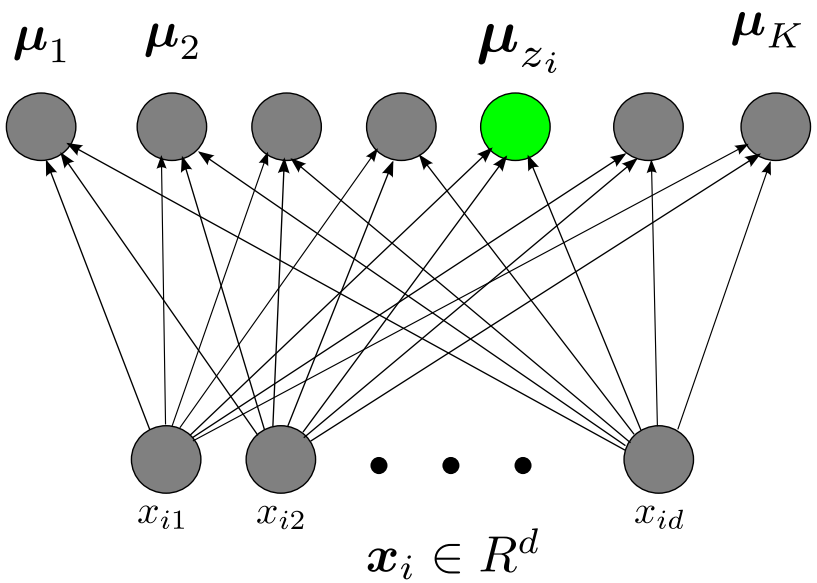
Input Neurons

Competitive learning V

Step 2 : Update **only** the winner neuron (Winner-Take-All):

$\boldsymbol{\mu}_{z_i}^{\text{new}} = \boldsymbol{\mu}_{z_i}^{\text{old}} + \alpha^{\text{old}}(\boldsymbol{x}_i - \boldsymbol{\mu}_{z_i}^{\text{old}})$

z_i being the index of the **winner neuron** for $\boldsymbol{x}_i$



Competitive learning VI

If $\alpha^{(q)} = \frac{1}{(\#\text{class } z_i)^{(q)}+1}$, one obtains the sequential K-means algorithm with $(\#\text{class } z_i)^{(q)}$ being the number observations assigned to the neuron z_i at the previous iteration.

Some limitations :

In the standard formulation of the competitive learning, only the winner neuron is updated, this is the well-known “winner-take-all" rule.

In addition, it does not consider the order between the neurons (i.e, when the neurons are located on a map lattice).

$\Rightarrow$ The self-organizing map (SOM) generalizes the competitive learning by allowing also the neighbors of the winner to be updated and for which the neurons become ordered on a map lattice.

Self-Organizing Maps (SOMs) I

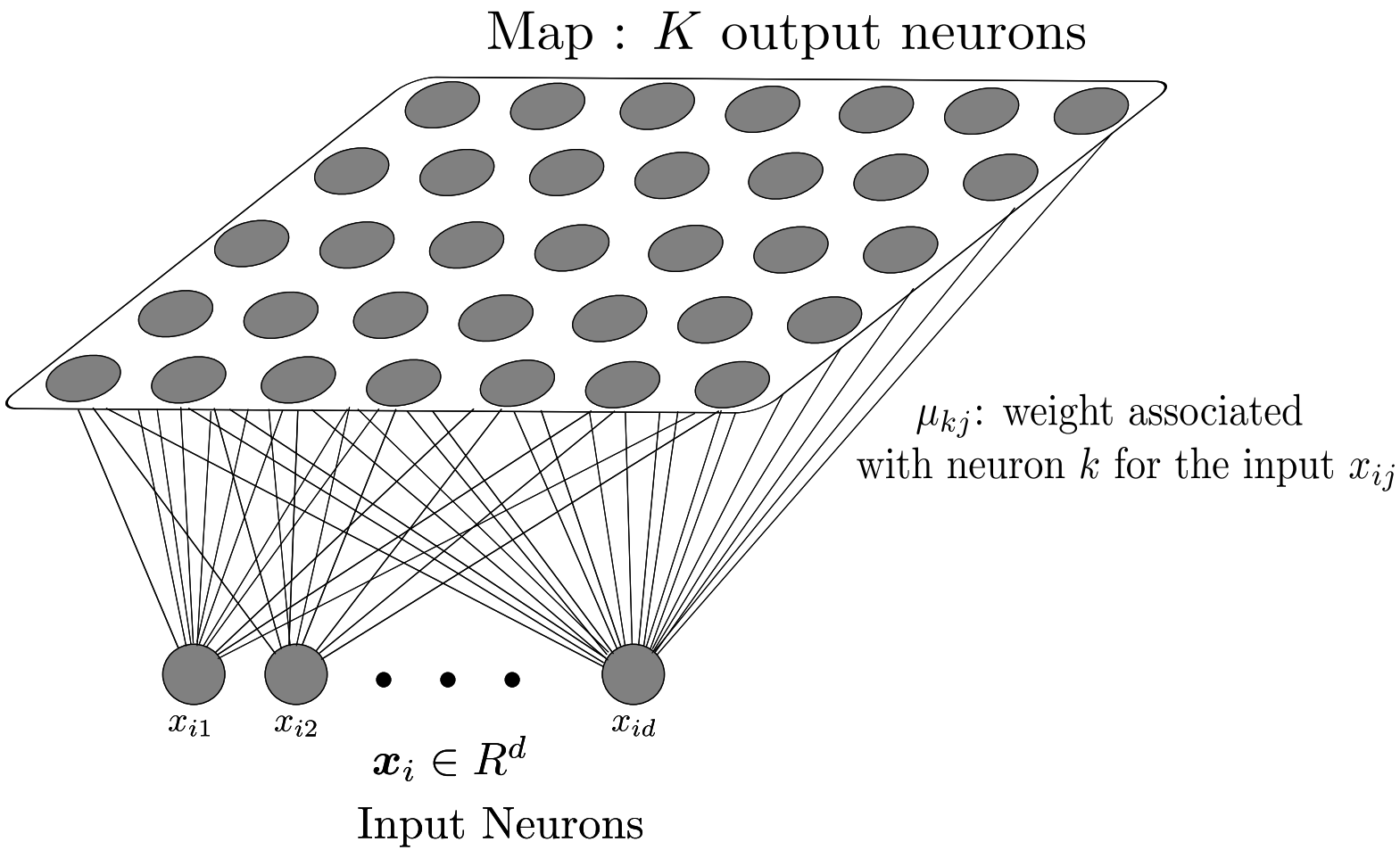
The self-organizing map (SOM) (Kohonen, 2001, 1989; Kohonen et al., 2000) is a neural-based approach for the exploration and visualization of high-dimensional data.

It derives an orderly mapping of multidimensional data onto a regular typically 2-dimensional grid (map).

It is a non linear projection method that converts complex nonlinear relationships in the high-dimensional space into simpler geometric relationships in the plan such that the important topological and metric relationships are conveyed.

The data are organized on the map in such a way that observations that are close together in the high-dimensional space are also closer to each other on the map.

Self-Organizing Maps (SOMs) II

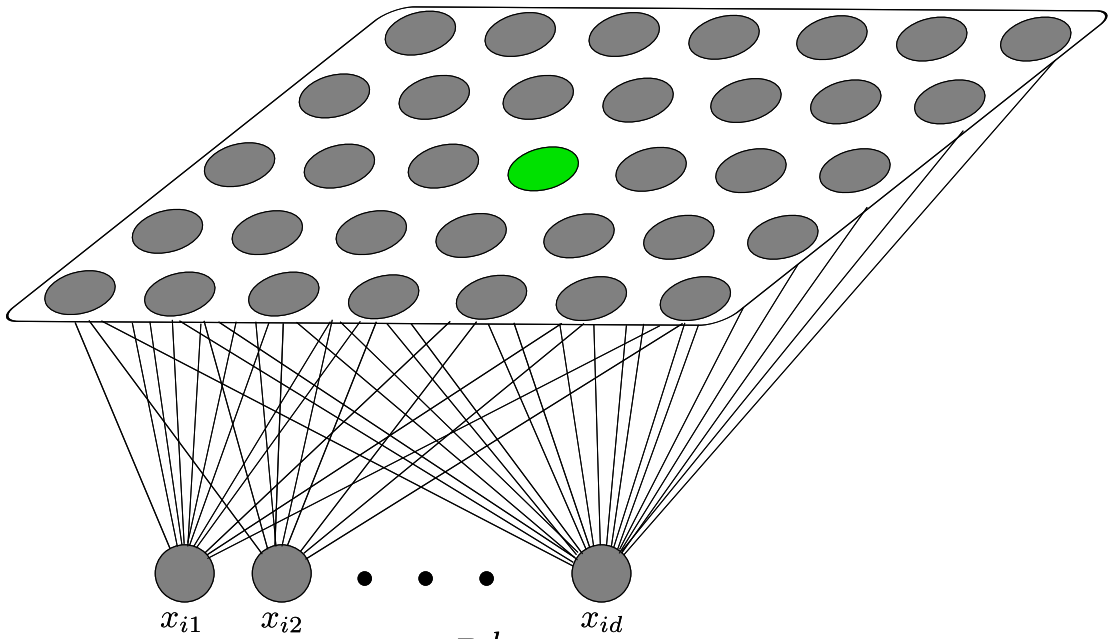


Self-Organizing Maps (SOMs) III

Step 1: **Competetion** : determine the **winner neuron**

$$z_i = \arg \min_{1<k<K} \| \boldsymbol{x}_i - \boldsymbol{\mu}_k \|^2$$

z_i being the index of the **winner neuron** for $\boldsymbol{x}_i$



$\boldsymbol{x}_i \in R^d$

Input Neurons

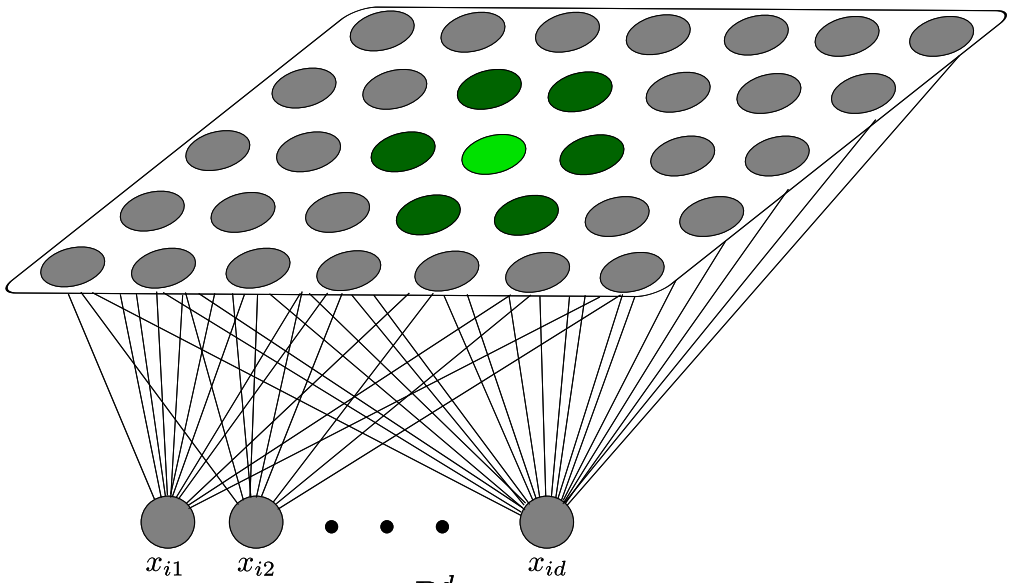
Self-Organizing Maps (SOMs) IV

Step 2: **cooperation** : Update **all** the neurons in a weighted manner

$$\boldsymbol{\mu}_k^{\text{new}} = \boldsymbol{\mu}_k^{\text{old}} + \alpha^{\text{old}} \mathcal{K}^{\text{old}}(\mathbf{r}_{z_i}, \mathbf{r}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{\text{old}})$$

z_i being the index of the **winner neuron** for $\boldsymbol{x}_i$

$\mathbf{r}_k$ denotes the coordinates of neuron k on the map



Self-Organizing Maps (SOMs) V

The Learning process of the SOM

There are two methods used in learning the SOM : incremental (sequential) learning and batch learning.

The incremental learning method is an iterative procedure

we start with a data point $\mathbf{x}_i$ and a set of d-dimensional model vectors $\boldsymbol{\mu}_k$ called neurons, units or prototypes.

Each neuron (prototype) is associated with a coordinate vector $\mathbf{r}_k$ on a 2-D map lattice and starts with some initial value $\boldsymbol{\mu}_k^{(q=0)}$).

At each iteration, a vector $\mathbf{x}_i$ is selected and the distance (typically Euclidean, but other choices are possible) between it and all the prototypes is calculated (for the **competition**).

Self-Organizing Maps (SOMs) VI

The best-matching unit (BMU) or prototype (the winner of the competition) is found and is denoted by $\boldsymbol{\mu}_{z_i}$

$$z_i = \arg\min_{k \in \mathcal{Z}} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2. \tag{7}$$

Once the closest prototype $\boldsymbol{\mu}_{z_i}$ is found, the prototypes are updated so that $\boldsymbol{\mu}_{z_i}$ is moved closer to the data vector $\mathbf{x}_i$. The neighbors of the BMU $\boldsymbol{\mu}_{z_i}$ are also updated, in a weighted manner (the **cooperation**) as follows :

$$\boldsymbol{\mu}_k^{(q+1)} = \boldsymbol{\mu}_k^{(q)} + \alpha^{(q)} \mathcal{K}^{(q)}(z_i, k)(\mathbf{x}_i - \boldsymbol{\mu}_k^{(q)}) \tag{8}$$

q denotes iteration number, $0 < \alpha^{(q)} < 1$ is the learning rate which decreases monotonically as the learning proceeds

$\mathcal{K}(z_i, k)$ is a chosen neighborhood function around the winner unit z_i (in general we have $\mathcal{K}^{(q)}(z_i, z_i) = 1$).

Self-Organizing Maps (SOMs) VII

The neighborhood function can for example be a Gaussian centered at the best-matching unit :

$$\mathcal{K}^{(q)}(z_i, k) = \exp\left(-\frac{\|\mathbf{r}_k - \mathbf{r}_{z_i}\|^2}{2\sigma^{2(q)}}\right)$$

$\mathbf{r}$ denotes the coordinates of the prototypes on the map, $\sigma^{(q)}$ the width of the neighborhood which decreases monotonically as the learning proceeds.

$\Rightarrow$ Due to the neighborhood function, the units which are closer to the BMU will be more affected than the others.

The previous steps are repeated until all the patterns $\mathbf{x}_i$ $(i = 1, \ldots, n)$ in the training set have been processed (at iteration q).

To achieve a better convergence towards the desired mapping it is usually required to repeat the previous loop until some convergence criteria are met (loop on iteration q $(q = 1, \ldots, q_{\max})$).

Self-Organizing Maps (SOMs) VIII

After the training step, we have the set of prototypes over the 2-D coordinates on the map.

In a clustering context, to find a partition of the data one can run a stand clustering algorithm on the prototypes, such as K-means or hierarchical clustering

Thus, from this point of view of clustering, SOM can be seen as a particular initialization for a later clustering step

Self-Organizing Maps (SOMs) IX

The batch training

The batch training is also iterative, but, at each iteration, it uses the whole data set before adjustments are made rather than a single data vector.

At each step of the algorithm, the data set is partitioned such that each observation is associated with its nearest model vector (prototype) in the sense of the Euclidean distance : $z_i = \arg\min_{k \in \mathcal{Z}} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$

The updated prototypes (units) $\boldsymbol{\mu}_k^{(q+1)}$ are found as a weighted average of the data, where the weight of each observation is the value of the neighborhood function at its BMU z_i, that is $\mathcal{K}^{(q)}(z_i, k)$:

$$\boldsymbol{\mu}_k^{(q+1)} = \frac{\sum_{i=1}^{n} \mathcal{K}^{(q)}(z_i, k)\mathbf{x}_i}{\sum_{i=1} \mathcal{K}^{(q)}(z_i, k)}. \tag{9}$$

Self-Organizing Maps (SOMs) X

SOM as optimizing a cost function

We note that while the SOM can be seen as an unsupervised learning algorithm (stochastically) minimizing the following cost function (Kaski, 1997; Kohonen, 2001)

$$J^{\text{SOM}}(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \mathbf{z}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \mathcal{K}(z_i, k) \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2, \tag{10}$$

The previous learning rules for the SOM correspond to a gradient descent in minimizing this SOM cost function

Several methods exist for visualizing the resulting map and prototypes, among them one can cite the U-matrix (Ultsch and Siemon, 1990) that is often used to locate clusters in the data.

Self-Organizing Maps (SOMs) XI

Limitations

Note that, at the origin, the SOM algorithm is based on heuristics and is not derived from the optimization of an objective function.

In addition, the preservation of the neighborhood structure is not guaranteed by the SOM method, and there could be problems with convergence of the prototype vectors.

The SOM does not define a density model, the choice of how the neighborhood function should shrink during training is also sensitive (the parameter σ).

$\Rightarrow$ The generative topographic mapping (GTM) (Bishop and Williams, 1998) was inspired by the SOM and attempts to overcome its limitations.

Self-Organizing Maps (SOMs) XII

The GTM is described in terms of a latent variable model (or space) with dimensionality d (Bishop and Williams, 1998).

The goal is to find a representation for the distribution $p(\mathbf{x})$ of d-dimensional data, in terms of a smaller number of p latent variables where $p < d$ and often take d $= 2$ for ease of visualization.

Additionally, the model parameters learning is performed the Expectation-Maximization (EM) algorithm in a maximum likelihood framework.

Both convergence and topographic ordering are guaranteed with the GTM.

Generative Topographic Mapping (GTM) I

The Generative Topographic Mapping (GTM) (Bishop and Williams, 1998), is a non-linear probabilistic projection method for data visualization, dimensionality reduction, etc

It was inspired by the SOM and attempts to overcome its limitations through a **probabilistic formulation**.

The GTM is described in terms of a **latent variable (or space) model** with dimensionality L

$\Rightarrow$ the goal is to find a representation for the distribution $p(\mathbf{y})$ of d-dimensional data, in terms of a smaller number of L latent variables where $L < d$ (often take $L = 2$ for visualization in the plan)

$\Rightarrow$ the model parameters learning is performed by the well-established **EM algorithm**

Generative Topographic Mapping (GTM) II

GTM has a well established statistical background and relies on the well-known stability and convergence properties of the Expectation-Maximization (EM) algorithm (Dempster et al., 1977).

Both convergence and topographic ordering are guaranteed with the GTM.

In addition, GTM performs soft clustering in contrary to SOM which assigns the neurons to the clusters in a hard way.

Further comparisons with the SOM algorithm can be found in (Bishop and Williams, 1998).

Generative Topographic Mapping (GTM) III

Let $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ be a set of n iid multidimensional data vectors $\mathbf{y}_i = (y_{i1}, \ldots, y_{id})^T \in \mathbb{R}^d$

Let $\mathbf{z} = (z_1, \ldots, z_n)$ be the associated unknown (hidden) states with $z_i \in \{1, \ldots, K\}$.

Now consider a two-dimensional latent space (the map) $\mathbf{x} = (x_1, x_2)^T$ on which we aim to visualize the data.

Generative Topographic Mapping (GTM) IV

the GTM model is a latent data (space) model asd is based on the finite mixture model formulation (McLachlan and Peel., 2000)

The aim is to represent the distribution of the observed data $p(\mathbf{y}_i)$ in the data space $\mathbb{R}^d$ in terms of a number of $L < d$-dimensional latent variables $\mathbf{x}$ with prior distribution $p(\mathbf{x})$.

The distribution $p(\mathbf{y})$ is then obtained by integration over the distribution of $\mathbf{x}$ by considering a specified conditional density $p(\mathbf{y}|\mathbf{x})$, that is :

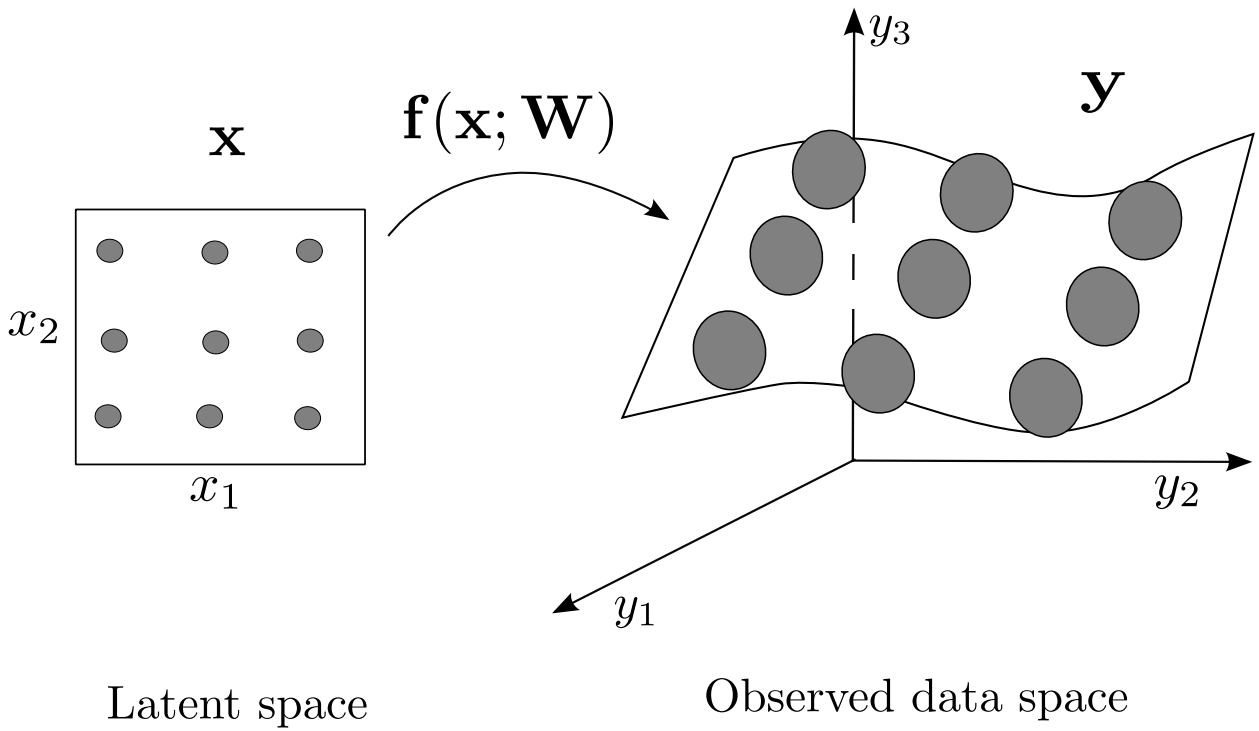
$$p(\mathbf{y}_i) = \int_{\mathcal{X}} p(\mathbf{y}_i|\mathbf{x})p(\mathbf{x})d\mathbf{x}. \tag{11}$$

For computational tractability of this integral, the GTM model assumes that the latent variables $\mathbf{x}$ have a prior Dirac mixture density given by

$$p(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K}\delta(\mathbf{x} - \mathbf{x}_k) \tag{12}$$

$\mathbf{x}_k$ represents the coordinates of the Dirac placement on the latent space.

Generative Topographic Mapping (GTM) V



Generative Topographic Mapping (GTM) VI

To specify the conditional density of the observations $\mathbf{y}$ on the latent variables $\mathbf{x}$, For the GTM, this is achieved by considering a parametric non-linear mapping function $\mathbf{f}(\mathbf{x};\mathbf{W})$ that maps the latent data $\mathbf{x}$ from the latent space to corresponding points in the data space.

$\Rightarrow$ the conditional density of the observations is then given as a Gaussian density centered at the projected points $\mathbf{f}(\mathbf{x};\mathbf{W})$ with variance β^{-1} :

$$p(\mathbf{y}_i|\mathbf{x}_k) = \left(\frac{\beta}{2\pi}\right)^{d/2} \exp\left\{-\frac{\beta}{2}\|\mathbf{y}_i - \mathbf{f}(\mathbf{x}_k;\mathbf{W})\|^2\right\} = \mathcal{N}(\mathbf{y}_i;\mathbf{f}(\mathbf{x}_k;\mathbf{W}),\beta^{-1}\mathbf{I}_d) \quad (13)$$

$\mathbf{f}(\mathbf{x}_k;\mathbf{W}) = \mathbf{W}\tilde{}(\mathbf{x}_k)$ is a d-dimensional point in the manifold embedded in data space

$\mathbf{W}$ is a $d \times M$ matrix of parameters that govern the mapping,

$\tilde{}(\mathbf{x}_k) = (\Phi_1(\mathbf{x}_k),\ldots,\Phi_M(\mathbf{x}_k))$ consists of M non-linear basis functions (in the standard model $\phi_m(\mathbf{x}_k)$ is a Gaussian : $\Phi_m(\mathbf{x}_k) = \exp\left\{-\frac{\|\mathbf{x}_k-\mu_m\|^2}{2\sigma^2}\right\}$).

Generative Topographic Mapping (GTM) VII

The GTM density (11) finally results in the following mixture density :

$$p(\mathbf{y}_i; \mathbf{W}, \beta) = \int_{\mathcal{X}} p(\mathbf{y}_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \frac{1}{K} \sum_{k=1}^{K} p(\mathbf{y}_i|\mathbf{x}_k) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\mathbf{y}_i; \mathbf{f}(\mathbf{x}_k; \mathbf{W}), \beta^{-1} \mathbf{I}_d). \tag{14}$$

The estimation of the GTM models parameters $(\mathbf{W}, \beta)$ from and i.i.d data sample is performed by maximizing the observed-data likelihood

$$p(\mathbf{Y}; \bar{}) = \prod_{i=1}^{n} \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\mathbf{y}_i; \mathbf{f}(\mathbf{x}_k; \mathbf{W}), \beta^{-1} \mathbf{I}_d). \tag{15}$$

via the EM algorithm (see the previous chapters or (Bishop and Williams, 1997)(Bishop et al., 1998)).

Generative Topographic Mapping (GTM) VIII

Some limitations

The standard GTM model (Bishop and Williams, 1997; Bishop et al., 1998) is dedicated to i.i.d data

The independence assumption becomes however very restricting for analyzing sequences.

$\Rightarrow$ the GTM Through Time (GTM-TT) (Bishop et al., 1997) overcomes these two limitations by relying on a hidden Markov model (HMM) formulation

GTM Through Time I

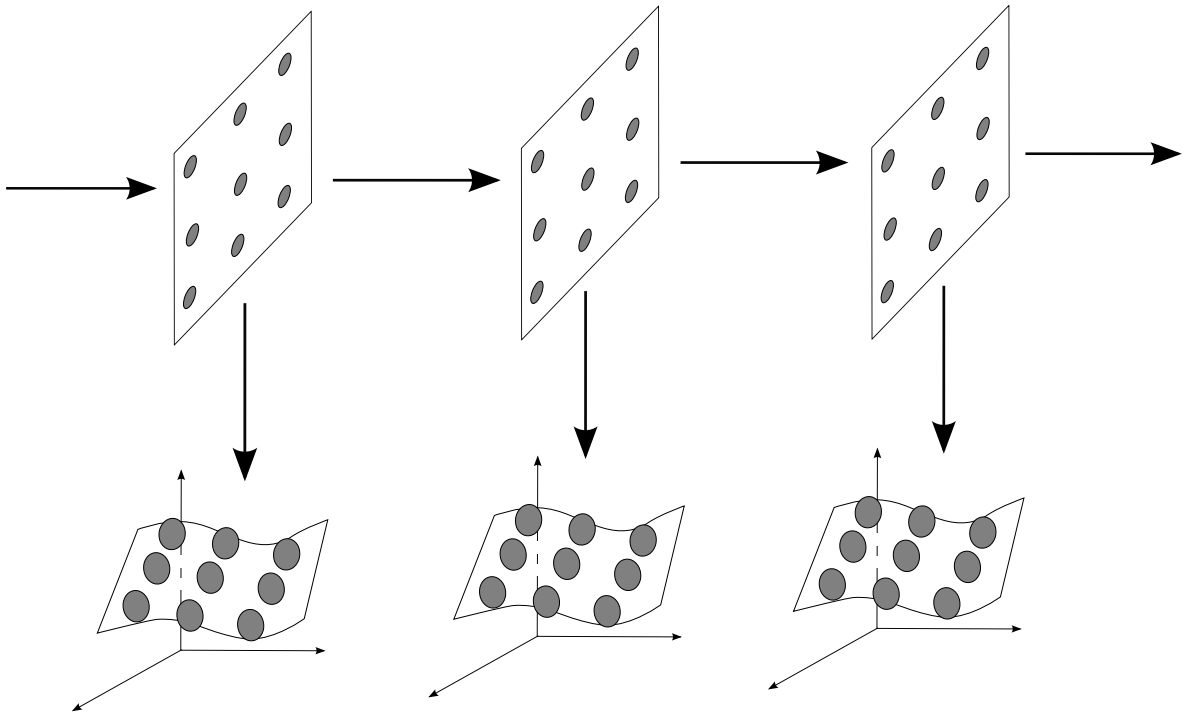
The GTM through time (GTM-TT) model extends the standard GTM model to learn from sequences by relaxing the independence assumption.

More specifically, the GTM-TT model incorporates the standard GTM model as the emission density in a Hidden Markov Model (HMM) (HMM will be studied later) as follows :

The hidden sequence $(z_1, \ldots, z_n)$ indicating the location on the latent space $(\mathbf{x}_{z_t})$ at each time step is a Markov chain with initial distribution π and transition matrix $\mathbf{A}$: $\pi_k = p(z_1 = k)$ and $\mathbf{A}_{\ell k} = p(z_t = k | z_{t-1} = \ell)$

The conditional emission density function is the one of the GTM model, that is $p(\mathbf{y}_t|\mathbf{x}_{z_t}) = \left(\frac{\beta}{2\pi}\right)^{d/2} \exp\left\{-\frac{\beta}{2}\|\mathbf{y}_t - \mathbf{f}(\mathbf{x}_{z_t}; \mathbf{W})\|^2\right\}$ where z_t denotes the state at time t.

GTM Through Time II



GTM Through Time III

Learning the GTM-TT

The model parameters $(\boldsymbol{\pi}, \mathbf{A}, \beta, \mathbf{W})$ are estimated by maximizing the observed data likelihood, which is expressed as the one of a standard HMM as follows

$$p(\mathbf{Y}; \bar{}) = \sum_{z_1} \cdots \sum_{z_n} p(z_1) p(\mathbf{y}_1 | \mathbf{x}_{z_1}) \prod_{t=2}^{n} p(z_t | z_{t-1}) p(\mathbf{y}_t | \mathbf{x}_{z_t}). \quad (16)$$

The maximization is performed by the EM (Baum-Welch) algorithm (Bishop et al., 1997)(Dempster et al., 1977)(Baum et al., 1970) where the E-step includes a forward-backward recursion to evaluate the posterior state distribution and to compute the likelihood.

Models for sequential data

- Markov chains
- Hidden Markov Models (HMMs)
- Types of HMMs
- Parameter estimation for HMMs
- Inference in HMMs
- Viterbi algorithm

Sequential data modeling

- Until now we have considered independence assumption for the observations which were assumed to be independent and identically distributed (i.i.d).
- Now we will relax this assumption by allowing a dependence between the data : the data are supposed to be an observation sequence and therefore ordered in the time.

Markov Chains

- Markov chains are a statistical modeling approach for sequences
- A Markov chain is a sequence of n random variables $(z_1, \ldots, z_n)$, generally referred to as the *states* of the chain, verifying the Markov property that is, the current state given the previous state sequence depends only on the previous state :

$$p(z_t|z_{t-1}, z_{t-2}, \ldots, z_1) = p(z_t|z_{t-1}) \ \forall t > 1.$$

- The probabilities $p(.|.)$ computed from the distribution p are called the *transition probabilities*.
- When the transition probabilities do not depend on t, the chain is called a *homogeneous* or *a stationary* Markov chain.

Markov Chains

- The standard Markov chain can be extended by assuming that the current state depends on a history of the state sequence, in this cas one can speak about high order Markov chains (see for example the thesis of (Muri, 1997)).
- A Markov chain of order p, p being a finite integer, can be defined as

$$p(z_t|z_{t-1}, z_{t-2}, \ldots, z_1) = p(z_t|z_{t-1}, \ldots, z_{t-p}) \ \forall t > p.$$

Hidden Markov Model (HMM)

- Markov chains are often integrated in a statistical latent data model for sequential data where the hidden sequence is assumed to be a Markov chain.
- The resulting model is the so-called hidden Markov model (HMM)
- Hidden Markov Models (HMMs) are a class of latent data models widely used in many application domains, including speech recognition, image analysis, time series prediction, etc Rabiner (1989); Derrode and Pieczynski (2006), etc.
- data are no longer assumed to be independent.
- It can be seen as a generalization of the mixture model by relaxing the independence assumption.
- Let us denote by $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ the observation sequence where the multidimensional data example $\mathbf{y}_t$ is observed data at time t, and let us denote by $\mathbf{z} = (z_1, \ldots, z_n)$ the hidden state sequence where the discrete random variable z_t which takes its values in the finite set $\mathcal{Z} = \{1, \ldots, K\}$ represents the unobserved state associated with $\mathbf{y}_t$.

Hidden Markov Model (HMM)

- An HMM is fully determined by :
  - the initial distribution $\pi = (\pi_1, \ldots, \pi_K)$ where $\pi_k = p(z_1 = k); k \in \{1, \ldots, K\}$,
  - the matrix of transition probabilities $\mathbf{A}$ where $\mathbf{A}_{\ell k} = p(z_t = k | z_{t-1} = \ell)$ for $t = 2, \ldots, n$, satisfying $\sum_k \mathbf{A}_{\ell k} = 1$,
  - the set of parameters $(\mathbf{\Psi}_1, \ldots, \mathbf{\Psi}_K)$ of the parametric conditional probability densities of the observed data $p(\mathbf{y}_t | z_t = k; \mathbf{\Psi}_k)$ for $t = 1, \ldots, n$ and $k = 1, \ldots, K$. These probabilities are also called the *emission probabilities*.
- e.g., a Gaussian HMM :

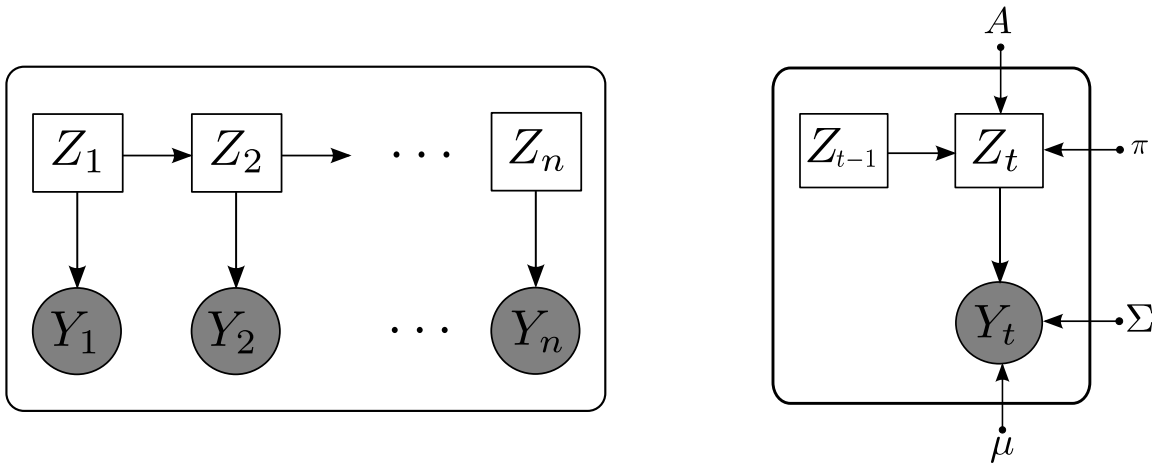


Figure – Graphical model structure for a Gaussian HMM.

Types of Hidden Markov Models

- HMMs can be classified according to the properties of their hidden Markov chain and the type of the emission state distribution.

- Homogeneous HMMs : models for which the hidden Markov chain has a stationary transition matrix.

- Non-homogeneous HMMs arise in the case when a temporal dependency is assumed for the HMM transition probabilities. (Diebold et al., 1994; Hughes et al., 1999; Meila and Jordan, 1996)
- Left-right HMMs : the states proceed from left to right according to the state indexes in a successive manner, for example such as in speech signals (Rabiner and Juang, 1993; Rabiner, 1989)
  $\Rightarrow$ imposing some restriction for the model through imposing particular constraints on the transition matrix : e.g.,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & 0 \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix}.$$

Types of Hidden Markov Models

- high order HMMs : when the current state depends on a finite history of the HMM states rather than only on the previous one
- Input Output HMMs (IOHMMs) (Bengio and Frasconi, 1995, 1996)
- Autoregressive HMM further generalize the standard HMMs by allowing the observations to be Autoregressive Markov chains (Muri, 1997; Rabiner, 1989; Juang and Rabiner, 1985; Celeux et al., 2004; Frühwirth-Schnatter, 2006).
- Another HMM extension lies in the Semi-Markov HMM Murphy (2002) which is like an HMM except each state can emit a sequence of observations.

Parameter estimation for a HMM

- $\boldsymbol{\Psi} = (\pi, \mathbf{A}, \boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_K)$: the model parameter vector to be estimated.

- The parameter estimation is performed by maximum likelihood.
- The observed-data log-likelihood to be maximized is given by :

$$\begin{aligned} \mathcal{L}(\boldsymbol{\Psi}) &= \log p(\mathbf{Y}; \boldsymbol{\Psi}) = \log \sum_{\mathbf{z}} p(\mathbf{Y}, \mathbf{z}; \boldsymbol{\Psi}) \\ &= \log \sum_{z_1, \ldots, z_n} p(z_1; \pi) \prod_{t=2}^{n} p(z_t | z_{t-1}; \mathbf{A}) \prod_{t=1}^{n} p(\mathbf{y}_t | z_t; \boldsymbol{\Psi}). \end{aligned}$$

- this log-likelihood is difficult to maximize directly
- $\Rightarrow$ use the EM algorithm, known as Baum Welch algorithm in the context of HMMs

Hidden Markov Model (HMM)

- the distribution of a particular configuration $\mathbf{z} = (z_1, \ldots, z_n)$ of the latent state sequence is written as

$$p(\mathbf{z}; \pi, \mathbf{A}) = p(z_1; \pi) \prod_{t=2}^{n} p(z_t | z_{t-1}; \mathbf{A}),$$

- conditional independence of the HMM : that is the observation sequence is independent given a particular configuration of the hidden state sequence
- $\Rightarrow$ the conditional distribution of the observed sequence :

$$p(\mathbf{Y} | \mathbf{z}; \mathbf{\Psi}) = \prod_{t=1}^{n} p(\mathbf{y}_t | z_t; \mathbf{\Psi}).$$

$\Rightarrow$ We then get the joint distribution (the complete-data likelihood) :

$$\begin{aligned} p(\mathbf{Y}, \mathbf{z}; \mathbf{\Psi}) &= p(\mathbf{z}; \mathbf{A}, \pi) p(\mathbf{Y} | \mathbf{z}; \theta) \\ &= p(z_1; \pi) \prod_{t=2}^{n} p(z_t | z_{t-1}; \mathbf{A}) \prod_{t=1}^{n} p(\mathbf{y}_t | z_t; \mathbf{\Psi}). \end{aligned}$$

Deriving EM for HMMs

- complete-data likelihood of $\boldsymbol{\Psi}$:

$$
\begin{aligned}
p(\mathbf{Y}, \mathbf{z}; \boldsymbol{\Psi}) &= p(z_1; \pi) \prod_{t=2}^{n} p(z_t | z_{t-1}; \mathbf{A}) \prod_{t=1}^{n} p(\mathbf{y}_t | z_t; \boldsymbol{\Psi}) \\
&= \prod_{k=1}^{K} p(z_1 = k; \pi)^{z_{1k}} \prod_{t=2}^{n} \prod_{k=1}^{K} \prod_{\ell=1}^{K} p(z_t = k | z_{t-1} = \ell; \mathbf{A})^{z_{t-1,\ell} z_{tk}} \prod_{t=1}^{n} \prod_{k=1}^{K} p(\mathbf{y}_t | z_t = k; \boldsymbol{\Psi}_k)^{z_{tk}} \\
&= \prod_{k=1}^{K} \pi_k^{z_{1k}} \prod_{t=2}^{n} \prod_{k=1}^{K} \prod_{\ell=1}^{K} \mathbf{A}_{\ell k}^{z_{t-1,\ell} z_{tk}} \prod_{t=1}^{n} \prod_{k=1}^{K} p(\mathbf{y}_t | z_t = k; \boldsymbol{\Psi}_k)^{z_{tk}}
\end{aligned}
$$

- $z_{tk} = 1$ if $z_t = k$ (i.e $\mathbf{y}_t$ originates from the kth state at time t) and $z_{tk} = 0$ otherwise.
- complete-data log-likelihood of $\boldsymbol{\Psi}$:

$$
\begin{aligned}
\mathcal{L}_c(\boldsymbol{\Psi}) &= \log p(\boldsymbol{Y}, \mathbf{z}; \boldsymbol{\Psi}) \\
&= \sum_{k=1}^{K} z_{1k} \log \pi_k + \sum_{t=2}^{n} \sum_{k=1}^{K} \sum_{\ell=1}^{K} z_{tk} z_{t-1,\ell} \log \mathbf{A}_{\ell k} + \sum_{t=1}^{n} \sum_{k=1}^{K} z_{tk} \log p(\mathbf{y}_t | z_t = k; \boldsymbol{\Psi}_k).
\end{aligned}
$$

The EM (Baum-Welch) algorithm

Start with an initial parameter $\boldsymbol{\Psi}^{(0)}$ and repeat the E- and M- steps until convergence :

E-step : compute the expectation of the complete-data log-likelihood :

$$
\begin{aligned}
Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)}) &= \mathbb{E}\Big[\mathcal{L}_c(\boldsymbol{\Psi})|\mathbf{Y}; \boldsymbol{\Psi}^{(q)}\Big] = \sum_{k=1}^{K} \mathbb{E}\Big[z_{1k}|\mathbf{Y}; \boldsymbol{\Psi}^{(q)}\Big] \log \pi_k + \\
&\sum_{t=2}^{n}\sum_{k=1}^{K}\sum_{\ell=1}^{K} \mathbb{E}\Big[z_{tk} z_{t-1,\ell}|\mathbf{Y}; \boldsymbol{\Psi}^{(q)}\Big] \log \mathbf{A}_{\ell k} + \sum_{t=1}^{n}\sum_{k=1}^{K} \mathbb{E}\Big[z_{tk}|\mathbf{Y}; \boldsymbol{\Psi}^{(q)}\Big] \log p(\mathbf{y}_t|z_t = k; \\
&= \sum_{k=1}^{K} p(z_1 = k|\mathbf{Y}; \boldsymbol{\Psi}^{(q)}) \log \pi_k + \sum_{t=2}^{n}\sum_{k=1}^{K}\sum_{\ell=1}^{K} p(z_t = k, z_{t-1} = \ell|\mathbf{Y}; \boldsymbol{\Psi}^{(q)}) \log \mathbf{A}_{\ell k} \\
&+ \sum_{t=1}^{n}\sum_{k=1}^{K} p(z_t = k|\mathbf{Y}; \boldsymbol{\Psi}^{(q)}) \log p(\mathbf{y}_t|z_t = k; \boldsymbol{\Psi}_k) \\
&= \sum_{k=1}^{K} \tau_{1k}^{(q)} \log \pi_k + \sum_{t=2}^{n}\sum_{k=1}^{K}\sum_{\ell=1}^{K} \xi_{t\ell k}^{(q)} \log \mathbf{A}_{\ell k} + \sum_{t=1}^{n}\sum_{k=1}^{K} \tau_{tk}^{(q)} \log p(\mathbf{y}_t|z_t = k; \boldsymbol{\Psi}_k),
\end{aligned}
$$

The EM (Baum-Welch) algorithm

where

- $\tau_{tk}^{(q)} = p(z_t = k|\mathbf{Y}; \boldsymbol{\Psi}^{(q)})$ $\forall t = 1, \ldots, n$ and $k = 1, \ldots, K$ is the posterior probability of the state k at time t given the whole observation sequence and the current parameter estimation $\boldsymbol{\Psi}^{(q)}$. The τ_{tk}'s are also referred to as the *smoothing probabilities*,

- $\xi_{t\ell k}^{(q)} = p(z_t = k, z_{t-1} = \ell|\mathbf{Y}; \boldsymbol{\Psi}^{(q)})$ $\forall t = 2, \ldots, n$ and $k, \ell = 1, \ldots, K$ is the joint posterior probability of the state k at time t and the state ℓ at time $t-1$ given the whole observation sequence and the current parameter estimation $\boldsymbol{\Psi}^{(q)}$.

- As shown in the expression of the Q-function, this step requires the computation of the posterior probabilities $\tau_{tk}^{(q)}$ and $\xi_{t\ell k}^{(q)}$.
- $\Rightarrow$ These posterior probabilities are computed by the forward-backward recursions.

Forward-Backward

- The forward procedure computes recursively the probabilities

$$\alpha_{tk} = p(\mathbf{y}_1, \ldots, \mathbf{y}_t, z_t = k; \boldsymbol{\Psi}),$$

$\Rightarrow$ the probability of observing the partial sequence $(\mathbf{y}_1, \ldots, \mathbf{y}_t)$ and ending with the state k at time t.

- $\Rightarrow$ the log-likelihood $\mathcal{L}$ can be computed after the forward pass as :

$$\log p(\mathbf{Y}; \boldsymbol{\Psi}) = \log \sum_{k=1}^{K} \alpha_{nk}.$$

Forward-Backward

- The backward procedure computes the probabilities

$$\beta_{tk} = p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_n | z_t = k; \boldsymbol{\Psi})$$

  $\Rightarrow$ the probability of observing the rest of the sequence $(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_1)$ knowing that we start with the k at time t.

- The forward and backward probabilities are computed recursively by the so-called Forward-Backward algorithm

- Notice that in practice, since the recursive computation of the α's and the β's involve repeated multiplications of small numbers which causes underflow problems, their computation is performed using a scaling technique in order to avoid underflow problems.

Posterior probabilities for an HMM

The posterior probability of the state k at time t given the whole sequence of observations $\mathbf{Y}$ and a model parameters $\boldsymbol{\Psi}$ is computed from the Forward-Backward and is given by

$$\begin{aligned}
\tau_{tk} &= p(z_t = k|\mathbf{Y}) \\
&= \frac{p(\mathbf{Y}, z_t = k)}{p(\mathbf{Y})} \\
&= \frac{p(\mathbf{Y}|z_t = k)p(z_t = k)}{\sum_{l=1}^{K} p(\mathbf{Y}|z_t = l)p(z_t = l)} \\
&= \frac{p(\mathbf{y}_1, \ldots, \mathbf{y}_t|z_t = k)p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_n|z_t = k)p(z_t = k)}{\sum_{l=1}^{K} p(\mathbf{y}_1, \ldots, \mathbf{y}_t|z_t = l)p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_n|z_t = l)p(z_t = l)} \\
&= \frac{p(\mathbf{y}_1, \ldots, \mathbf{y}_t, z_t = k)p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_n|z_t = k)}{\sum_{l=1}^{K} p(\mathbf{y}_1, \ldots, \mathbf{y}_t, z_t = l)p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_n|z_t = l)} \\
&= \frac{\alpha_{tk}\beta_{tk}}{\sum_{l=1}^{K} \alpha_{tl}\beta_{tl}} . \qquad (17)
\end{aligned}$$

Joint posterior probabilities for an HMM

The joint posterior probabilities of the state k at time t and the state ℓ at time $t-1$ given the whole sequence of observations are therefore given by

$$
\begin{aligned}
\xi_{t\ell k} &= p(z_t = k, z_{t-1} = \ell | \mathbf{Y}) \\
&= \frac{p(z_t = k, z_{t-1} = \ell, \mathbf{Y})}{p(\mathbf{Y})} \\
&= \frac{p(z_t = k, z_{t-1} = \ell, \mathbf{Y})}{\sum_{\ell=1}^{K} \sum_{k=1}^{K} p(z_t = k, z_{t-1} = \ell, \mathbf{Y})} \\
&= \frac{p(\mathbf{Y} | z_t = k, z_{t-1} = \ell) p(z_t = k, z_{t-1} = \ell)}{\sum_{\ell=1}^{K} \sum_{k=1}^{K} p(\mathbf{Y} | z_t = k, z_{t-1} = \ell) p(z_t = k, z_{t-1} = \ell)} \\
&= \frac{p(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1}, \ldots, \mathbf{y}_1 | z_t = k, z_{t-1} = \ell) p(z_t = k, z_{t-1} = \ell)}{\sum_{\ell=1}^{K} \sum_{k=1}^{K} p(\mathbf{Y} | z_t = k, z_{t-1} = \ell) p(z_t = k, z_{t-1} = \ell)} \\
&= \frac{\alpha_{(t-1)\ell} p(\mathbf{y}_t | z_t = k) \beta t k A_{\ell k}}{\sum_{\ell=1}^{K} \sum_{k=1}^{K} \alpha_{(t-1)\ell} p(\mathbf{y}_t | z_t = k) \beta t k A_{\ell k}} . \qquad (18)
\end{aligned}
$$

Forward-Backward

- The posterior probabilities are then expressed in function of the forward backward probabilities as follows :

$$\tau_{tk}^{(q)} = \frac{\alpha_{tk}^{(q)} \beta_{tk}^{(q)}}{\sum_{k=1}^{K} \alpha_{tk}^{(q)} \beta_{tk}^{(q)}}$$

and

$$\xi_{t\ell k}^{(q)} = \frac{\alpha_{t-1,\ell}^{(q)} p(\mathbf{y}_t | z_t = k; \boldsymbol{\theta}^{(q)}) \beta_{tk}^{(q)} \mathbf{A}_{\ell k}^{(q)}}{\sum_{\ell=1}^{K} \sum_{k=1}^{K} \alpha_{t-1,\ell}^{(q)} p(\mathbf{y}_t^{(q)} | z_t = k; \boldsymbol{\Psi}) \beta_{tk}^{(q)} \mathbf{A}_{\ell k}^{(q)}}.$$

Forward Recursion

$$
\begin{aligned}
\alpha_{tk} &= p(\mathbf{y}_1, \ldots, \mathbf{y}_t, z_t = k) \\
&= p(\mathbf{y}_1, \ldots, \mathbf{y}_t | z_t = k) p(z_t = k) \\
&= p(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1} | z_t = k) p(\mathbf{y}_t | z_t = k) p(z_t = k) \\
&= p(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}, z_t = k) p(\mathbf{y}_t | z_t = k) \\
&= \sum_{\ell=1}^{K} p(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}, z_{t-1} = \ell, z_t = k) p(\mathbf{y}_t | z_t = k) \\
&= \sum_{\ell=1}^{K} p(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1} | z_{t-1} = \ell) p(z_t = k, z_{t-1} = \ell) p(\mathbf{y}_t | z_t = k) \\
&= \sum_{\ell=1}^{K} p(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}, z_t = k | z_{t-1} = \ell) p(z_t = k | z_{t-1} = \ell) p(z_{t-1} = \ell) p(\mathbf{y}_t | z_t = \\
&= \sum_{\ell=1}^{K} p(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}, z_{t-1} = \ell) p(z_t = k | z_{t-1} = \ell) p(\mathbf{y}_t | z_t = k) \\
&= \left[\sum_{\ell=1}^{K} \alpha_{(t-1)\ell} A_{\ell k} \right] p(\mathbf{y}_t | z_t = k) \qquad (1
\end{aligned}
$$

Backward Recursion

$$
\begin{aligned}
\beta_{t\ell} &= p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_n | z_t = \ell) \\
&= \sum_{k=1}^{K} p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_n, z_{t+1} = k | z_t = \ell) \\
&= \sum_{k=1}^{K} p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_n | z_{t+1} = k, z_t = \ell) p(z_{t+1} = k | z_t = \ell) \\
&= \sum_{k=1}^{K} p(\mathbf{y}_{t+2}, \ldots, \mathbf{y}_n | z_{t+1} = k, z_t = \ell) p(z_{t+1} = k | z_t = \ell) p(\mathbf{y}_{t+1} | z_{t+1} = k) \\
&= \sum_{k=1}^{K} p(\mathbf{y}_{t+2}, \ldots, \mathbf{y}_n | z_{t+1} = k) p(z_{t+1} = k | z_t = \ell) p(\mathbf{y}_{t+1} | z_{t+1} = k) \\
&= \sum_{k=1}^{K} \beta_{(t+1)k} A_{\ell k} p(\mathbf{y}_{t+1} | z_{t+1} = k). \qquad (20)
\end{aligned}
$$

Forward-Backward

The computation of these quantities is therefore performed by the Forward Backward procedure. For all $\ell, k = 1, \ldots, K$:

For all $\ell, k = 1, \ldots, K$:

1. **Forward procedure**
   - $\alpha_{1k} = p(\mathbf{y}_1, z_1 = 1; \boldsymbol{\Psi}) = p(z_1 = 1)p(\mathbf{y}_1 | z_1 = 1; \boldsymbol{\theta}) = \pi_k p(\mathbf{y}_1 | z_1 = k; \boldsymbol{\theta})$ **for** $t = 1$**,**
   - $\alpha_{tk} = [\sum_{\ell=1}^{K} \alpha_{(t-1)\ell} A_{\ell k}] p(\mathbf{y}_t | z_t = k; \boldsymbol{\Psi}) \quad \forall\ t = 2, \ldots, n.$
2. **Backward procedure**
   - $\beta_{nk} = 1$ **for** $t = n$**,**
   - $\beta_{t\ell} = \sum_{k=1}^{K} \beta_{(t+1)k} A_{\ell k} p(\mathbf{y}_{t+1} | z_{t+1} = k; \boldsymbol{\Psi}) \quad \forall\ t = n-1, \ldots, 1.$

The EM (Baum-Welch) algorithm

M-step : update the value of $\boldsymbol{\Psi}$ by computing the parameter $\boldsymbol{\Psi}^{(q+1)}$ maximizing the expectation Q-function with respect to $\boldsymbol{\Psi}$. The Q-function can be decomposed as

$$Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)}) = Q_{\pi}(\pi, \boldsymbol{\Psi}^{(q)}) + Q_{\mathbf{A}}(\mathbf{A}, \boldsymbol{\Psi}^{(q)}) + \sum_{k=1}^{K} Q(\boldsymbol{\Psi}_k, \boldsymbol{\Psi}^{(q)})$$

with

$$\begin{aligned}
Q_{\pi}(\pi, \boldsymbol{\Psi}^{(q)}) &= \sum_{k=1}^{K} \tau_{1k}^{(q)} \log \pi_k, \\
Q_{\mathbf{A}}(\mathbf{A}, \boldsymbol{\Psi}^{(q)}) &= \sum_{t=2}^{n} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \xi_{t\ell k}^{(q)} \log \mathbf{A}_{\ell k}, \\
Q_{\boldsymbol{\Psi}_k}(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)}) &= \sum_{t=1}^{n} \tau_{tk}^{(q)} \log p(\mathbf{y}_t | z_t = k; \bar{}_k).
\end{aligned}$$

- The maximization of $Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)})$ with respect to $\boldsymbol{\Psi}$ is then performed by separately maximizing $Q_\pi(\pi, \boldsymbol{\Psi}^{(q)})$, $Q_{\mathbf{A}}(\mathbf{A}, \boldsymbol{\Psi}^{(q)})$ and $Q_{\boldsymbol{\Psi}_k}(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)})$ $(k = 1, \ldots, K)$.

- The updating formulas for the Markov chain parameters are given by :

$$\begin{aligned}
\pi_k^{(q+1)} &= \arg\max_{\pi_k} Q_\pi(\pi, \boldsymbol{\Psi}^{(q)}) \text{ subject to } \sum_k \pi_k = 1 \\
&= \tau_{1k}^{(q)} \\
\mathbf{A}_{\ell k}^{(q+1)} &= \arg\max_{\boldsymbol{A}_{\ell k}} Q_{\boldsymbol{A}}(\mathbf{s}1, \boldsymbol{\Psi}^{(q)}) \text{ subject to } \sum_k \boldsymbol{A}_{\ell k} = 1 \\
&= \frac{\sum_{t=2}^n \xi_{tk\ell}^{(q)}}{\sum_{t=2}^n \sum_k \xi_{t\ell k}^{(q)}} = \frac{\sum_{t=2}^n \xi_{tk\ell}^{(q)}}{\sum_{t=2}^n \tau_{t\ell}^{(q)}}
\end{aligned}$$

These constrained maximizations are solved using Lagrange multipliers.

- The maximization of $Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)})$ with respect to $Q_{\boldsymbol{\Psi}_k}(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(q)})$ $(k = 1, \ldots, K)$ depends on the form of emission probability function. Foa example, for the Gaussian case where $p(\boldsymbol{y}_t | z_t = k; \boldsymbol{\Psi}_k = \mathcal{N}(\mathbf{y}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, we have the following updating formulas :

- The updating formulas are given by :

$$\begin{aligned}
\boldsymbol{\mu}_k^{(q+1)} &= \frac{1}{\sum_{t=1}^{n} \tau_{tk}^{(q)}} \sum_{t=1}^{n} \tau_{tk}^{(q)} \mathbf{y}_t \\
\boldsymbol{\Sigma}_k^{(q+1)} &= \frac{1}{\sum_{t=1}^{n} \tau_{tk}^{(q)}} \sum_{t=1}^{n} \tau_{tk}^{(q)} (\mathbf{y}_t - \boldsymbol{\mu}_k^{(q+1)})(\mathbf{y}_t - \boldsymbol{\mu}_k^{(q+1)})^T.
\end{aligned}$$

Gaussian HMM

- an HMM with Gaussian emission probabilities :

$$\mathbf{y}_t = \boldsymbol{\mu}_{z_t} + \boldsymbol{\epsilon}_t \quad ; \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{z_t}),$$

- the latent sequence $\mathbf{z} = (z_1, \ldots, z_n)$ is drawn from a first-order homogeneous Markov chain
- the ϵ_t are independent random variables distributed according to a Gaussian distribution with zero mean and covariance matrix $\boldsymbol{\Sigma}_{z_t}$.
- the state conditional density $p(\mathbf{y}_t | z_t = k; \boldsymbol{\Psi}_k)$ is Gaussian :

$$p(\mathbf{y}_t | z_t = k; \boldsymbol{\Psi}_k) = \mathcal{N}(\mathbf{y}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where $\boldsymbol{\Psi}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

Gaussian HMM

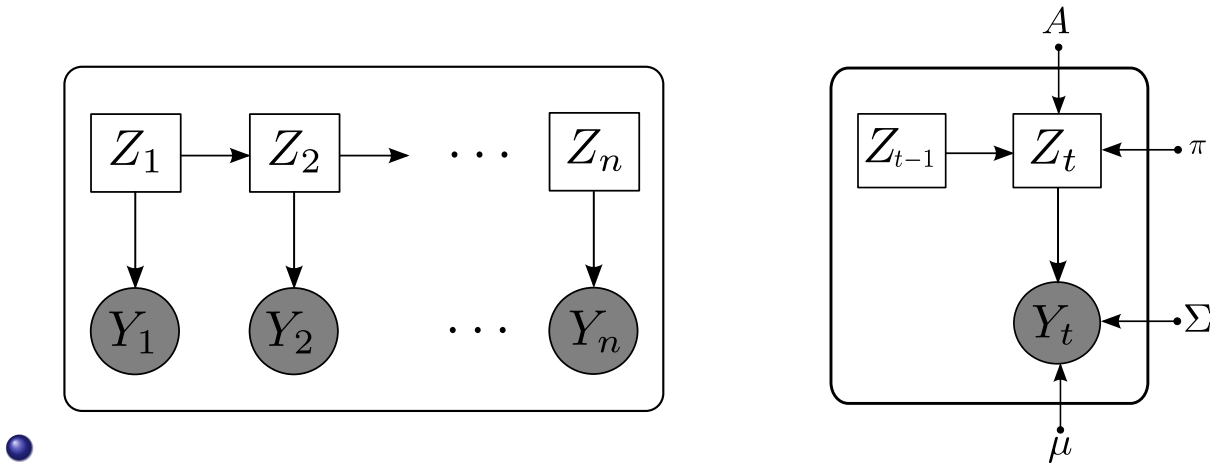


Figure – Graphical model structure for a Gaussian HMM.

- The model parameters are learned in a maximum likelihood framework by the EM algorithm.
- EM (Baum-Welch in this context of HMMs) includes forward-backward recursions to compute the E-Step
- the M-step is performed in a similar way as for a Gaussian mixture

Viterbi decoding algorithm I

Recall that we have three basic problems associated with HMMs :

1. Find $p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n; \boldsymbol{\Psi})$, that is the likelhiood for an observation sequence $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ given an HMM ($\boldsymbol{\Psi}$) : **an evaluation problem**.
   $\Rightarrow$ As seen previously, we use the forward (or the backward) procedure for this since it is much more efficient than direct evaluation.
2. Find an HMM ($\boldsymbol{\Psi}$) given an observation sequence $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$: **a Learning problem**
   $\Rightarrow$ As seen before, the Baum-Welch (EM) algorithm solves this problem,
3. Given an observation sequence $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ and a HMM ($\boldsymbol{\Psi}$), find the most likely state sequence $\boldsymbol{z} = (z_1, \ldots, z_n)$ that have generated $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ under $\boldsymbol{\Psi}$: **an Inference problem**.
   $\Rightarrow$ As we can see it now, the Viterbi algorithm solves this problem

Viterbi decoding algorithm II

The Viterbi algorithm (Viterbi, 1967; Forney, 1973) provides an efficient dynamic programming approach to computing the most likely state sequence $(\hat{z}_1, \ldots, \hat{z}_n)$ that have generated an observation sequence $(\mathbf{y}_1, \ldots, \mathbf{y}_n)$, given a set of HMM parameters $(\boldsymbol{\Psi})$.

It estimates the following MAP state sequence :

$$\begin{aligned}
\hat{\mathbf{z}} &= \arg \max_{z_1, \ldots, z_n} p(\mathbf{y}_1, \ldots, \mathbf{y}_n, z_1, \ldots, z_n; \boldsymbol{\Psi}) \\
&= \arg \max_{z_1, \ldots, z_n} p(z_1) p(\mathbf{y}_1 | z_1) \prod_{t=2}^{n} p(z_t | z_{t-1}) p(\mathbf{y}_t | z_t) \\
&= \arg \min_{z_1, \ldots, z_n} \Big[-\log \pi - \log p(\mathbf{y}_1 | z_1) + \sum_{t=2}^{n} -\log p(z_t | z_{t-1}) - \log p(\mathbf{y}_t | z_t) \Big].
\end{aligned}$$

The Viterbi algorithm works on the dynamic programming principle that is :

Viterbi decoding algorithm III

The minimum cost path to $z_t = k$ is equivalent to the minimum cost path to node z_{t-1} plus the cost of a transition from z_{t-1} to $z_t = k$ (and the cost incurred by observation $\mathbf{y}_t$ given $z_t = k$).

The MAP state sequence is then determined by starting at node z_t and reconstructing the optimal path backwards based on the stored calculations.

Viterbi decoding reduces the computation cost to $\mathcal{O}(K^2 n)$ operations instead of the brute force $O(K^n)$ operations. The Viterbi algorithm steps are outlined in Algorithm 4.

Viterbi decoding algorithm IV

Algorithm 4 Pseudo code of the Viterbi algorithm for an HMM.

Inputs : Observations $(\mathbf{y}_1, \ldots, \mathbf{y}_n)$ and HMM params $\boldsymbol{\Psi}$

1: Initialization : initialize minimum path sum to state $z_1 = k$ for $k = 1, \ldots, K$:

$$S_1(z_1 = k) = -\log \pi_k - \log p(\mathbf{y}_1 | z_1 = k)$$

2: Recursion : for $t = 2, \ldots, n$ and for $k = 1, \ldots, K$, calculate the minimum path sum to state $z_t = k$:

$$S_t(z_t = k) = -\log p(\mathbf{y}_t | z_t = k) + \min_{z_{t-1}} \left[S_{t-1}(z_{t-1}) - \log p(z_t = k | z_{t-1}) \right]$$

and let

$$z^*_{t-1}(z_t) = \arg\min_{z_{t-1}} \left[S_{t-1}(z_{t-1}) - \log p(z_t = k | z_{t-1}) \right]$$

3: Termination : compute $\min_{z_n} S_n(z_n)$ and set $\hat{z}_n = \arg\min_{z_n} S_n(z_n)$

4: State sequence backtracking : iteratively set, for $t = n - 1, \ldots, 1$

$$\hat{z}_t = z^*_t(\hat{z}_{t+1})$$

Outputs : The most likely state sequence $(\hat{z}_1, \ldots, \hat{z}_n)$.

Latent data models for dimensionality reduction

- Introduction
- Principal Component Analysis (PCA)
- Probabilistic Principal Component Analysis (PPCA)
- Factor Analysis (FA)
- tSNE

Introduction I

Until now we have considered discrete latent data models (GMM, HMM) and continuous ones (GTM, etc) ; now we will see other continuous latent data models

The aim here is the dimensionality reduction for preprocessing, compression, feature extraction, visualization etc of high dimensional data

Principal Component Analysis (PCA) (Pearson, 1901; Hotelling, 1933),

Factor Analysis (FA)(Spearman, 1904; Thurstone, 1947),

Independent Component Analysis (ICA)(Comon, 1994; Hyvärinen, 2001), etc

are examples of well-known techniques which can be used to achieve this task.

Introduction II

PCA is a well-established technique for dimensionality reduction,

A linear projection technique that **maximizes the variance** in the projected space

Equivalently, it **minimizes the reconstruction error** (after the dimensionality reduction)

Principal Component Analysis (PCA) I

Two views of PCA :

1. First view : PCA maximizing the projected variance (Hotelling, 1933)
2. Second view : minimizing the reconstruction error (after the dimensionality reduction)

The most common derivation of PCA is in terms of a standardized linear projection which maximizes the variance in the projected space (Hotelling, 1933).

Consider a set of observed d-dimensional data vector $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$

$\Rightarrow$ The aim is to project the data onto a space having dimensionality $M < d$ while maximizing the variance of the projected data.

Principal Component Analysis (PCA) II

Consider the sample mean vector and the sample covariance matrix :

$$\begin{aligned} \bar{\mathbf{x}} &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \\ \mathbf{S} &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^T. \end{aligned}$$

Let us first consider the projection onto a one-dimensional space ($M=1$). The direction of this space can be defined using a d-dimensional direction unit vector $\mathbf{u}_1$ (with $\mathbf{u}_1^T\mathbf{u}_1=1$)

The linear projection of a data vector $\mathbf{x}_i$ on the projected space is given by the scalar :

$$\mathbf{u}_1^T\mathbf{x}_i$$

Principal Component Analysis (PCA) III

$\Rightarrow$ The variance of the projected data is therefore given by the scalar :

$$v(\mathbf{u}_1) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{u}_1^T\mathbf{x}_i - \mathbf{u}_1^T\bar{\mathbf{x}})(\mathbf{u}_1^T\mathbf{x}_i - \mathbf{u}_1^T\bar{\mathbf{x}})^T = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 \cdot \quad (21)$$

The **principal axe** (the direction vector) is then given by :

$$\mathbf{u}_1 = \arg\max_{\mathbf{u}_1 \in \mathbb{R}^d} \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 \text{ subject to } \mathbf{u}_1^T\mathbf{u}_1 = 1 \cdot \quad (22)$$

The normalisation condition is namely to prevent $\|\mathbf{u}_1\| \to \infty$

This is a constrained maximization problem $\Rightarrow$ Use a Lagrange multiplier to solve it

Principal Component Analysis (PCA) IV

The unconstrained maximization is therefore given by

$$\mathbf{u} = \arg\max_{\mathbf{u}_1 \in \mathbb{R}^d} \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^T \mathbf{u}_1) \cdot \tag{23}$$

$$\frac{\partial \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda (1 - \mathbf{u}_1^T \mathbf{u}_1)}{\partial \mathbf{u}_1} = 0 \quad \overset{\frac{\partial \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1}{\partial \mathbf{u}_1} = (\mathbf{S} + \mathbf{S}^T)\mathbf{u}}{\Leftrightarrow} \quad 2\mathbf{S}\mathbf{u}_1 - 2\lambda_1 \mathbf{u}_1 = 0$$

$$\Leftrightarrow \quad \mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1 \tag{24}$$

$\Rightarrow$ $\mathbf{u}_1$ must be an eigenvector of the data covariance matrix S, with eigenvalue λ_1

The variance (21) in the projected space is then given by

$$\begin{aligned} \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 &= \mathbf{u}^T \lambda_1 \mathbf{u}_1 \\ &= \lambda_1 \quad (\text{ since } \mathbf{u}_1^T \mathbf{u}_1 = 1) \end{aligned} \tag{25}$$

Principal Component Analysis (PCA) V

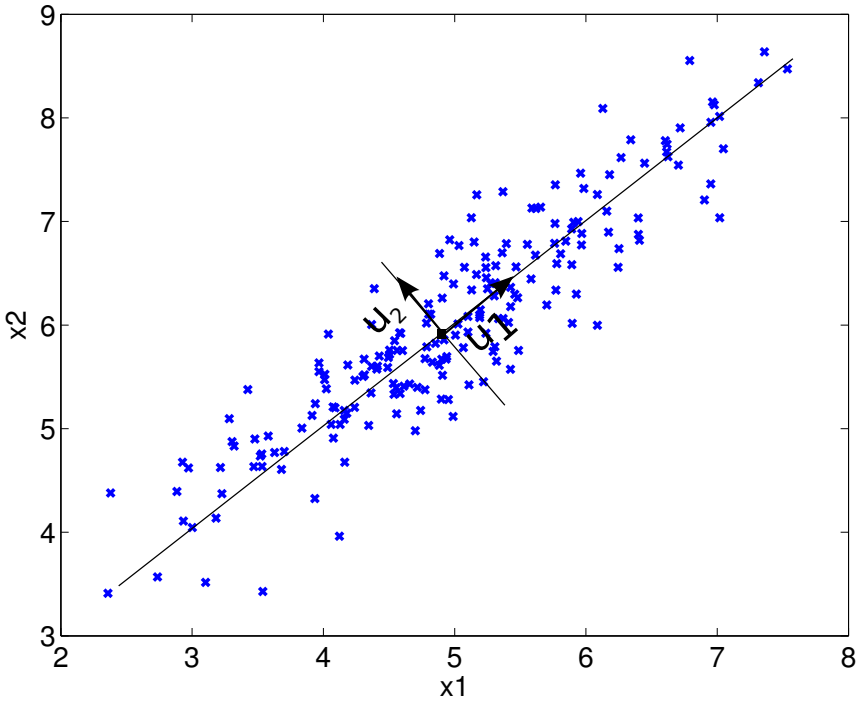
$\Rightarrow$ The variance will be maximum if we set $\mathbf{u}_1$ equal to the eigenvector having the **largest** eigenvalue λ_1.

$\Rightarrow$ The eigenvector $\mathbf{u}_1$ is known as the first **principal component** or the first principal axe.

We can define additional principal components in an incremental way :

The new direction to be chosen is that which maximizes the projected variance amongst all possible directions orthogonal to those already considered.

Principal Component Analysis (PCA) VI



Principal Component Analysis (PCA) VII

General case :

Consider the general case of a projection space of dimension M (M principal components).

The optimal linear projection for which the projected variance is maximized is defined by M eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_M$ of the data covariance matrix $\mathbf{S}$ corresponding to the largest eigenvalues $\lambda_1, \ldots, \lambda_M$

The result was shown for one principal components ($M = 1$).

Now suppose that the result holds for M principal components and we aim to show that it holds for $M + 1$ (by induction)

For $\mathbf{u}_{M+1}$, the projected variance in the direction $\mathbf{u}_{M+1}$ is given by

$$\mathbf{u}_{M+1}^T \mathbf{S} \mathbf{u}_{M+1}$$

Principal Component Analysis (PCA) VIII

We therefore maximize the variance $\mathbf{u}_{M+1}^T \mathbf{S}\mathbf{u}_{M+1}$ by taking into account the normalization constraint and the orthogonality constraints :

- normalization constraint : $\mathbf{u}_{M+1}^T$ is normalized to unit length, that is : $\mathbf{u}_{M+1}^T \mathbf{u}_{M+1} = 1$
- orthogonal constraint : $\mathbf{u}_{M+1}$ orthogonal to the existing vectors $\mathbf{u}_1, \ldots, \mathbf{u}_M$, that is : $\mathbf{u}_{M+1}^T \mathbf{u}_k = 0$ for $k \neq M+1$

$\Rightarrow$ Use a Lagrange multiplier λ_{M+1} and Lagrange multipliers $\eta_k, k = 1, \ldots, M$ to enforce these constraints.

$\Rightarrow$ Thus we solve the following unconstrained maximization problem

$$\begin{aligned} \mathbf{u}_{M+1} &= \arg\max_{\mathbf{u}_{M+1}} \mathbf{u}_{M+1}^T \mathbf{S}\mathbf{u}_{M+1} + \lambda_{M+1}(1 - \mathbf{u}_{M+1}^T \mathbf{u}_{M+1}) + \sum_{k=1}^{m} \eta_k \mathbf{u}_{M+1}^T \mathbf{u}_k \\ &= \arg\max_{\mathbf{u}_{M+1}} v\left(\{\mathbf{u}_k, \eta_k\}_{k=1}^m, \lambda_{M+1}, \mathbf{u}_{M+1}\right) \end{aligned} \tag{26}$$

Principal Component Analysis (PCA) IX

$$\frac{\partial v\left(\{\mathbf{u}_k, \eta_k\}_{k=1}^m, \lambda_{M+1}, \mathbf{u}_{M+1}\right)}{\partial \mathbf{u}_{M+1}} = 0 \Leftrightarrow 2\mathbf{S}\mathbf{u}_{M+1} - 2\lambda_{M+1}\mathbf{u}_{M+1} + \sum_{k=1}^{m} \eta_k \mathbf{u}_k = 0$$

$$\Leftrightarrow \underbrace{\mathbf{u}_j^T \mathbf{S}\mathbf{u}_{M+1}}_{0} - \lambda_{M+1} \underbrace{\mathbf{u}_j^T \mathbf{u}_{M+1}}_{0} + \underbrace{\sum_{k=1}^{m} \eta_k \mathbf{u}_j^T \mathbf{u}_k}_{\eta_j \mathbf{u}_j^T \mathbf{u}_j = \eta_j} = 0$$

$$\Leftrightarrow \eta_j = 0 \quad \text{for} \quad j = 1, \ldots, m \tag{27}$$

We therefore obtain : $\mathbf{S}\mathbf{u}_{M+1} = \lambda_{M+1}\mathbf{u}_{M+1}$

$\Rightarrow \mathbf{u}_{M+1}$ must be an eigenvector of $\mathbf{S}$ with eigenvalue λ_{M+1} .

The projected variance in direction $\mathbf{u}_{M+1}$ is therefore given by

$$\begin{aligned} \mathbf{u}_{M+1}^T \mathbf{S}\mathbf{u}_{M+1} &= \mathbf{u}_{M+1}^T \lambda_{M+1}\mathbf{u}_{M+1} \\ &= \lambda_{M+1} \end{aligned} \tag{28}$$

Principal Component Analysis (PCA) X

$\Rightarrow$ The variance will be maximum if we set $\mathbf{u}_{M+1}$ equal to the eigenvector having the **largest** eigenvalue λ_{M+1} amongst those not previously selected.

Thus the result holds also for projection spaces of dimensionality $M + 1$, which completes the inductive step.

Since we have already shown this result explicitly for $M = 1$, if follows that the result must hold for any $M \leq d$.

Principal Component Analysis (PCA) XI

Second view : minimization of the reconstruction error PCA minimizes the reconstruction error, that is the squared error between a data point $\mathbf{x}_i$ and its approximation $\tilde{\mathbf{x}}_i$, averaged over all the data points : $J = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2$

Consider one direction $\mathbf{u}$ in the projection space

Here we will show minimizing this error w.r.t $\mathbf{u}$ is equivalent to maximizing the projected variance (21) on the direction $\mathbf{u}$

Let us assume that all the original vectors $\mathbf{x}_i$ have been centered :

$$\mathbf{x}_i^c = \mathbf{x}_i - \bar{\mathbf{x}}_i$$

By using the fact that the projection of a data vector $\mathbf{x}$ onto the direction $\mathbf{u}$ is given by the scalar $\mathbf{u}^T\mathbf{x}$; $\mathbf{x}$ is then represented by $(\mathbf{u}^T\mathbf{x})\mathbf{u}$ in the projected space,

Principal Component Analysis (PCA) XII

The reconstruction error for the centred data is then given by :

$$
\begin{aligned}
J(\mathbf{u}) &= \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i^c - \tilde{\mathbf{x}}_i^c\|^2 \frac{1}{n}\sum_{i=1}^{n}\|(\mathbf{x}_i - \bar{\mathbf{x}}_i) - [\mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}}_i)\mathbf{u}]\|^2 \\
&= \frac{1}{n}\Big(\sum_{i=1}^{n}\|\mathbf{x}_i - \bar{\mathbf{x}}_i\|^2 - 2\sum_{i=1}^{n}[\mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}}_i)][\mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}}_i)] + \sum_{i=1}^{n}[\mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}}_i)]^2\|\mathbf{u}\|^2\Big) \\
&= \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \bar{\mathbf{x}}_i\|^2 - \frac{2}{n}\sum_{i=1}^{n}[\mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}}_i)]^2 + \frac{1}{n}\sum_{i=1}^{n}[\mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}}_i)]^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \bar{\mathbf{x}}_i\|^2 - \frac{1}{n}\sum_{i=1}^{n}[\mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}}_i)]^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \bar{\mathbf{x}}_i\|^2 - \frac{1}{n}\sum_{i=1}^{n}\mathbf{u}^T(\mathbf{x}_i - \bar{\mathbf{x}}_i)(\mathbf{x}_i - \bar{\mathbf{x}}_i)\mathbf{u} \\
&= \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \bar{\mathbf{x}}_i\|^2 - \mathbf{u}^T\mathbf{S}\mathbf{u}
\end{aligned}
\tag{29}
$$

The first term does not depend on $\mathbf{u}$. Thus the vector $\mathbf{u}$ that minimizes $J(\mathbf{u})$ is the same one that maximizes the projected variance $\mathbf{u}^T\mathbf{S}\mathbf{u}$ (21).

Principal Component Analysis (PCA) XIII

Summary :

PCA reduces the dimensionality of the data while retaining as much as possible of the variation present in the original dataset

$$\mathbf{X} : [n \times d\ ] \xrightarrow{\text{Linear Proj. onto } \mathbf{U}=[\mathbf{u}_1,\ldots,\mathbf{u}_M]} \tilde{\mathbf{X}} = \mathbf{X}_c\mathbf{U} : [n \times M]; M \leq d$$

To perform PCA on a data set $\mathbf{X}$

1. calculate the mean data vector $\bar{\mathbf{x}}$
2. calculate the data covariance matrix $\mathbf{S}$
3. calculate the eigenvectors and the corresponding eigenvalues of $\mathbf{S}$ (e.g., by using the `eig` function in Matlab)
4. the eigenvalues $\lambda_1, \ldots, \lambda_d$ are sorted in decreasing order ; the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_d$ are placed according to the resulting order

Principal Component Analysis (PCA) XIV

5. the projection space (the space of principal axes) is then obtained by taking the M first eigenvectors $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M]; M \leq d$
6. the projected data are given by $\tilde{\mathbf{X}} = \mathbf{X}_c\mathbf{U}$ where $\mathbf{X}_c$ is the centered data matrix

How to choose M ? for example one way is to choose the first M components that capture a specified percentage e.g., 90%, 95%, or 99%, of the cumulative percentage of variance. $cpv(M) = 100\left(\frac{\sum_{m=1}^{M}\lambda_m}{\sum_{m=1}^{d}\lambda_m}\right)\%$

Disadvantage : One disadvantage of both these definitions of PCA is the absence of a probability density model and associated likelihood measure.

Principal Component Analysis (PCA) XV

Deriving PCA from the perspective of density estimation would offer a number of important advantages, including the following :

- The likelihood measure would permit comparison with other density models
- We can derive EM for PCA and hence deal with missing values in the data set
- Possibility to perform Bayesian inference (e.g. for model selection)
- Possibility of computing the the posterior class probabilities if PCA is used to model the class-conditional densities in a classification problem,
- The value of the probability density function would give a measure of the novelty of a new data point.
- PCA model could be extended to a mixture framework.

$\Rightarrow$ Use Probabilistic Principal Component Analysis (PPCA)

Probabilistic Principal Component Analysis (PPCA) I

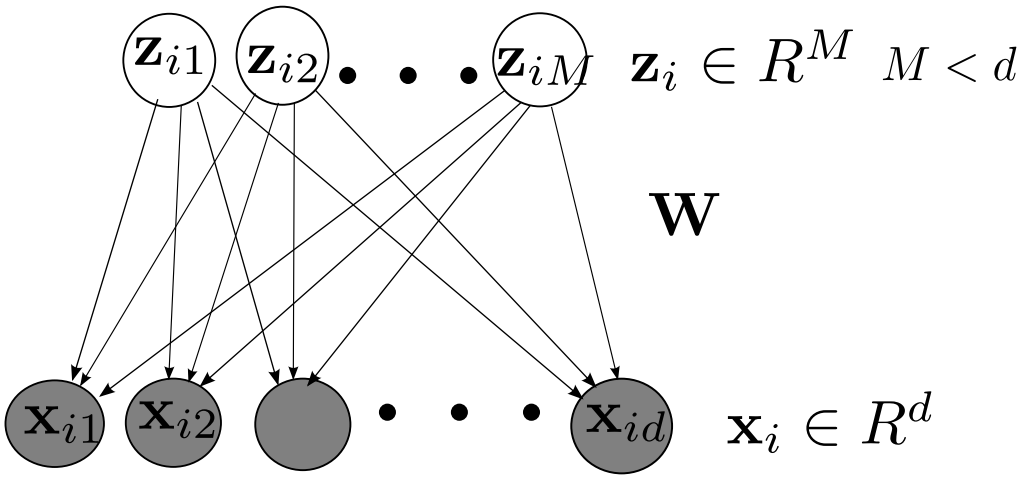
PCA can be formulated into a probabilistic framework : the Probabilistic Principal Component Analysis (PPCA) (Tipping and Bishop, 1997, 1999; Roweis, 1998)

The PC can be expressed as the maximum likelihood solution of a latent **continuous** variable model and the model parameter are optimized using EM (Tipping and Bishop, 1997, 1999; Roweis, 1998)

$\Rightarrow$ Generative formulation : **the latent variable model** for PPCA :

$$
\begin{aligned}
\mathbf{x}_i &= \mathbf{W}\mathbf{z}_i + \boldsymbol{\mu} + \boldsymbol{\epsilon}_i \text{ Observed data = linear transf. of } \mathbf{z} + \text{additive Gaussian noise} \\
\mathbf{z}_i &\sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \text{ latent variables of the principal component subspace} \\
\boldsymbol{\epsilon} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ zero-mean Gaussian noise} \\
\mathbf{x}_i|\mathbf{z}_i &\sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \text{ conditional density for the observed data} \\
\mathbf{x}_i &\sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}) \text{ marginal density for the observed data}
\end{aligned}
\tag{30}
$$

Probabilistic Principal Component Analysis (PPCA) II



Maximum Likelihood for PPCA I

Model parameters : $(\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$ where $\mathbf{W}$ a $[d \times M]$ matrix whose columns represent the principal subspace, $\boldsymbol{\mu}$ a d-dimensional vector

- Assume we have an i.i.d sample $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$.
- The observed-data log-likelihood is given by :

$$\begin{aligned}
\mathcal{L}(\mathbf{W}, \boldsymbol{\mu}, \sigma^2) &= \log \prod_{i=1}^{n} p(\mathbf{x}_i; \mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \log \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{\mu}, \underbrace{\mathbf{W}\mathbf{W}^T + \sigma^2 \mathbf{I}}_{\mathbf{C}}) \\
&= -\frac{nd}{2} \log 2\pi - \frac{n}{2} \log |\mathbf{C}| - \frac{1}{2} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) \\
&= -\frac{n}{2} \left(d \log 2\pi + \frac{1}{2} \log |\mathbf{C}| + \text{trace} \left\{ \mathbf{C}^{-1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})^T (\mathbf{x}_i - \boldsymbol{\mu}) \right\} \right) \\
&= -\frac{n}{2} \left(d \log 2\pi + \frac{1}{2} \log |\mathbf{C}| + \text{trace} \left\{ \mathbf{C}^{-1} \mathbf{S} \right\} \right)
\end{aligned} \tag{31}$$

$\Rightarrow$ Analytical solutions

Maximum Likelihood for PPCA II

ML estimates (Tipping and Bishop, 1999) :

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i = \bar{\mathbf{x}} \tag{32}$$

$$\hat{\sigma}^2 = \frac{1}{d-M}\sum_{m=M+1}^{d}\lambda_m \tag{33}$$

$$\hat{\mathbf{W}} = \mathbf{U}_M(\mathbf{L}_M - \hat{\sigma^2}\mathbf{I})^{1/2}\mathbf{R}, \tag{34}$$

where $\mathbf{U}_M$ is a $[d \times M]$ matrix whose columns are the first M eigenvectors $[\mathbf{u}_1, \ldots, \mathbf{u}_M]$ of the data covariance-matrix $\mathbf{S}$ corresponding to the the first M largest eigenvalues $[\lambda_1, \ldots, \lambda_M]$

$\mathbf{L}_M$ is an $[M \times M]$ matrix whose diagonal elements are the corresponding eigenvalues $[\lambda_1, \ldots, \lambda_M]$

$\mathbf{R}$ is an $[M \times M]$ arbitrary orthogonal matrix ($\mathbf{R}\mathbf{R}^T = \mathbf{I}$)

EM for PPCA I

The PPCA is expressed as a latent data model : so we can use EM to find the ML estimates for PPCA

While we have exact solutions ; using EM, as it is iterative, may have an advantage in spaces of high dimensionality compared to when working with the sample data covariance matrix $\mathbf{S}$ (for the eignvalues and eigenvalues)

The EM procedure can also be extended to Factor Analysis for which there is no analytical solutions

The log-likelihood of the PPCA model parameters $(\mathbf{W}, \boldsymbol{\mu}, \sigma^2)$ for the complete-data $(\mathbf{X}, \mathbf{Z}) = (\mathbf{x}_1, \mathbf{z}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n)$:

$$\mathcal{L}_c(\mathbf{W}, \boldsymbol{\mu}, \sigma^2; \mathbf{X}, \mathbf{Z}) = \log \prod_{i=1}^{n} p(\mathbf{x}_i, \mathbf{z}_i; \mathbf{W}, \boldsymbol{\mu}, \sigma^2) = \sum_{i=1}^{n} [\log p(\mathbf{x}_i|\mathbf{z}_i) + \log p(\mathbf{z}_i)]$$

EM for PPCA II

Complete-data log-likelihood :

$$\begin{aligned}
\mathcal{L}_c(\mathbf{W}, \boldsymbol{\mu}, \sigma^2) &= \sum_{i=1}^{n} [\log p(\mathbf{x}_i|\mathbf{z}_i) + \log p(\mathbf{z}_i)] \\
&= \sum_{i=1}^{n} [\log \mathcal{N}(\mathbf{x}_i; \mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I}) + \log \mathcal{N}(\mathbf{z}_i; \mathbf{0}, \sigma^2\mathbf{I})] \\
&= -\sum_{i=1}^{n} \left\{ \frac{d}{2}\log(2\pi\sigma^2) + \frac{1}{2} trace(\mathbf{z}_i\mathbf{z}_i^T) + \frac{1}{2\sigma^2}\|\mathbf{x}_i - \boldsymbol{\mu}\|^2 \right. \\
&\quad \left. -\frac{1}{\sigma^2}\mathbf{z}_i^T\mathbf{W}^T(\mathbf{x}_i - \boldsymbol{\mu}) + \frac{1}{2\sigma^2} trace(\mathbf{z}_i\mathbf{z}_i^T\mathbf{W}\mathbf{W}^T \right\} \qquad (35)
\end{aligned}$$

Expected complete-data log-likelihood (the Q-function) :

$$\begin{aligned}
\mathbb{E}[\mathcal{L}_c(\mathbf{W}, \boldsymbol{\mu}, \sigma^2)|\mathbf{X}, \{\mathbf{W}, \boldsymbol{\mu}, \sigma^2\}_{\text{old}}] = -\sum_{i=1}^{n} \left\{ \frac{d}{2}\log(2\pi\sigma^2) + \frac{1}{2} trace(\mathbb{E}[\mathbf{z}_i\mathbf{z}_i^T]) + \frac{1}{2\sigma^2}\|\mathbf{x}_i - \boldsymbol{\mu}\|^2 \right. \\
\left. -\frac{1}{\sigma^2}\mathbb{E}[\mathbf{z}_i]^T\mathbf{W}^T(\mathbf{x}_i - \boldsymbol{\mu}) + \frac{1}{2\sigma^2} trace(\mathbb{E}[\mathbf{z}_i\mathbf{z}_i^T]\mathbf{W}\mathbf{W}^T \right\} \qquad (36)
\end{aligned}$$

EM for PPCA III

NB : for $\boldsymbol{\mu}$, we get its closed form solution : $\hat{\boldsymbol{\mu}} = \bar{\mathbf{x}}$

Only $\mathbf{W}$ and σ^2 are computed in an iterative way by EM

1. **E-step** : By using the old parameters values, compute

$$\mathbb{E}[\mathbf{z}_i] = (\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})^{-1}\mathbf{W}^T(\mathbf{x}_i - \bar{\mathbf{x}}) \quad (37)$$

$$\mathbb{E}[\mathbf{z}_i\mathbf{z}_i^T] = \sigma^2(\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})^{-1} + \mathbb{E}[\mathbf{z}_i]\mathbb{E}[\mathbf{z}_i]^T \quad (38)$$

2. **M-step**

$$\mathbf{W}_{\text{new}} = \left[\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})\mathbb{E}[\mathbf{z}_i]^T\right]\left[\sum_{i=1}^{n}\mathbb{E}[\mathbf{z}_i\mathbf{z}_i^T]\right]^{-1} \quad (39)$$

$$\sigma^2_{\text{new}} = \frac{1}{nd}\sum_{i=1}^{n}\left\{\|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 - 2\mathbb{E}[\mathbf{z}_i]^T\mathbf{W}_{\text{new}}^T(\mathbf{x}_i - \bar{\mathbf{x}}) + trace(\mathbb{E}[\mathbf{z}_i\mathbf{z}_i^T]\mathbf{W}_{\text{new}}\mathbf{W}_{\text{new}}^T\right\} \quad (40)$$

NB. Here $\mathbb{E}[.]$ is actually $\mathbb{E}[.|\mathbf{X}, \{\mathbf{W}, \boldsymbol{\mu}, \sigma^2\}_{\text{old}}]$

Factor Analysis (FA) I

Factor Analysis (FA) (Spearman, 1904; Thurstone, 1947)

FA is closely related to PPCA

The only difference is

$$\mathbf{x}_i|\mathbf{z}_i \sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi}) \text{ conditional density for the observed data}$$

$\boldsymbol{\Psi}$ is a $d \times d$ digonal matrix ; rather than

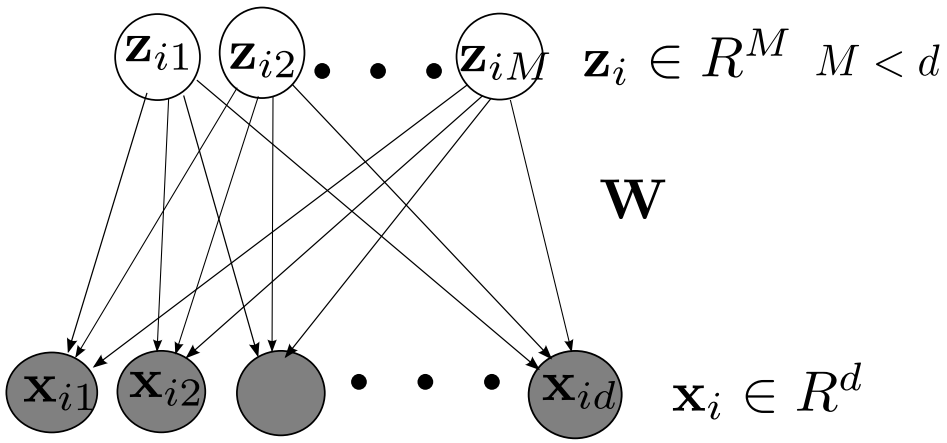
$$\mathbf{x}_i|\mathbf{z}_i \sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I}) \text{ conditional density for the observed data}$$

(isotropic covariance matrix).

Factor Analysis (FA) II

Generative model

$$
\begin{aligned}
\mathbf{x}_i &= \mathbf{W}\mathbf{z}_i + \boldsymbol{\mu} + \boldsymbol{\epsilon}_i \text{ Observed data = linear transf. of } \mathbf{z} + \text{additive Gaussian noise} \\
\mathbf{z}_i &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}) \text{ latent variables of the principal component subspace} \\
\boldsymbol{\epsilon} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ zero-mean Gaussian noise} \\
\mathbf{x}_i|\mathbf{z}_i &\sim \mathcal{N}(\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi}) \text{ conditional density for the observed data} \\
\mathbf{x}_i &\sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi}) \text{ marginal density for the observed data}
\end{aligned}
$$



Factor Analysis (FA) III

EM

1. **E-step**

$$\mathbb{E}[\mathbf{z}_i] = (\mathbf{I} + \mathbf{W}^T\boldsymbol{\Psi}^{-1}\mathbf{W})^{-1}\mathbf{W}^T(\boldsymbol{\Psi}^{-1}\mathbf{x}_i - \bar{\mathbf{x}}) \quad (41)$$

$$\mathbb{E}[\mathbf{z}_i\mathbf{z}_i^T] = (\mathbf{I} + \mathbf{W}^T\boldsymbol{\Psi}^{-1}\mathbf{W})^{-1} + \mathbb{E}[\mathbf{z}_i]\mathbb{E}[\mathbf{z}_i]^T \quad (42)$$

2. **M-step**

$$\mathbf{W}_{\text{new}} = \left[\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})\mathbb{E}[\mathbf{z}_i]^T\right]\left[\sum_{i=1}^{n}\mathbb{E}[\mathbf{z}_i\mathbf{z}_i^T]\right]^{-1} \quad (43)$$

$$\boldsymbol{\Psi}_{\text{new}} = \text{diag}\left\{\mathbf{S} - \mathbf{W}_{\text{new}}\frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\mathbf{z}_i](\mathbf{x}_i - \bar{\mathbf{x}})^T\right\} \quad (44)$$

NB. Here $\mathbb{E}[.]$ is actually $\mathbb{E}[.|\mathbf{X}, \{\mathbf{W}, \boldsymbol{\mu}, \boldsymbol{\Psi}\}_{\text{old}}]$

Illustration on PCA (Face Recognition) seen in classroom

tSNE : supports vus en cours et pdf disponible sur

`https://chamroukhi.com/Teaching/ApprentissageM2/tSNE.pdf`

Bibliography I

Baum, L., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. Annals of Mathematical Statistics, 41 :164–171.

Bengio, Y. and Frasconi, P. (1995). An input output hmm architecture. In Advances in Neural Information Processing Systems, volume 7, pages 427–434.

Bengio, Y. and Frasconi, P. (1996). Input Output HMM's for sequences processing. IEEE Transactions on Neural Networks, 7(5).

Bishop, C. M., Hinton, G. E., and Strachan, I. G. D. (1997). Gtm through time. In IEE Fifth International Conference on Artificial Neural Networks, pages 111–116.

Bishop, C. M., Svensén, M., and Williams, C. K. I. (1998). Gtm : The generative topographic mapping. Neural Computation, 10 :215–234.

Bishop, C. M. and Williams, C. K. I. (1997). Gtm : A principled alternative to the self-organizing map. In Advances in Neural Information Processing Systems, pages 354–360. Springer-Verlag.

Bishop, C. M. and Williams, C. K. I. (1998). Gtm : The generative topographic mapping. Neural Computation, 10 :215–234.

Celeux, G. and Govaert, G. (1992). A classification em algorithm for clustering and two stochastic versions. Computational Statistics and Data Analysis, 14 :315–332.

Celeux, G., Nascimento, J., and Marques, J. (2004). Learning switching dynamic models for objects tracking. PR, 37(9) :1841–1853.

Comon, P. (1994). Independent Component Analysis, a new concept ? Signal Processing, 36(3) :287–314. Special issue on Higher-Order Statistics.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of The Royal Statistical Society, B, 39(1) :1–38.

Derrode, S. Benyoussef, L. and Pieczynski, W. (2006). Contextual estimation of hidden markov chains with application to image segmentation. In ICASSP, pages 15–19, Toulouse.

Diebold, F., Lee, J.-H., and Weinbach, G. (1994). Regime switching with time-varying transition probabilities. Nonstationary Time Series Analysis and Cointegration. (Advanced Texts in Econometrics), pages 283–302.

Bibliography II

Forney, G. D. (1973). The viterbi algorithm. Proceedings of the IEEE, 61(3) :268–278.

Frühwirth-Schnatter, S. (2006). Finite Mixture and Markov Switching Models (Springer Series in Statistics). Springer Verlag, New York.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 24 :417–441.

Hughes, J. P., Guttorp, P., and Charles, S. P. (1999). A non-homogeneous hidden markov model for precipitation occurrence. Applied Statistics, 48 :15–30.

Hyvärinen, A. (2001). Independant Component Analysis. Wiley.

Juang, B.-H. and Rabiner, L. R. (1985). Mixture autoregressive hidden markov models for speech signals. IEEE Transactions on Acoustics, Speech and Signal Processing, 33(6) :1404–1413.

Kaski, S. (1997). Data Exploration Using Self-Organizing Maps. PhD thesis, Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering.

Kohonen, T. (1989). Self-organization and associative memory. Springer-Verlag New York, Inc. New York, NY, USA.

Kohonen, T. (2001). Self-Organizing Maps. Information Sciences. Springer, third edition edition.

Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., and Saarela, A. (2000). Self organization of a massive document collection. IEEE Transactions on Neural Networks, 11(3) :574–585.

Kong, S. and Kosko, B. (1991). Differential competitive learning for centroid estimation and phoneme recognition. IEEE Transactions on Neural Networks, 2(1) :118–124.

McLachlan, G. J. and Peel., D. (2000). Finite mixture models. New York : Wiley.

Meila, M. and Jordan, M. I. (1996). Learning fine motion by markov mixtures of experts. In Advances in Neural Information Processing Systems 8, pages 1003–1009. MIT Press.

Muri, F. (1997). Comparaison d'algorithmes d'identification de chaînes de Markov cachées et application à la détection de régions homogènes dans les séquences ADN. PhD thesis, Université Paris Descartes, Paris V.

Murphy, K. P. (2002). Dynamic Bayesian Networks : Representation, Inference and Learning. PhD thesis, UC Berkeley, Computer Science Division.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. Philosophical Magazine, 2(6) :559–572.

Bibliography III

Rabiner, L. and Juang, B.-H. (1993). Fundamentals of speech recognition. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2) :257–286.

Roweis, S. (1998). EM algorithms for PCA and SPCA. In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, Proceedings of the 11th Conference on Advances in Neural Information Processing Systems (NIPS), volume 10. MIT Press.

Spearman, C. (1904). General intelligence, objectively determined and measured. American Journal of psychology, 15 :201–293.

Thurstone, L. L. (1947). Multiple Factor Analysis. University of Chicago Press.

Tipping, M. E. and Bishop, C. (1997). Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Comuting Research Group, Aston University.

Tipping, M. E. and Bishop, C. (1999). Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B, 61 :611–622.

Ultsch, A. and Siemon, H. P. (1990). Kohonen's self organizing feature maps for exploratory data analysis. In Proceedings of International Neural Networks Conference (INNC), pages 305–308. Kluwer Academic Press.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory, 13(2) :260–269.