Analyse de Données

Slides 1ère année Sciences du Numérique

Vincent Charvillat et Jean-Yves Tourneret[1]

(1) Université de Toulouse, ENSEEIHT-IRIT
vincent.charvillat@enseeiht.fr,jyt@n7.fr

Janvier 2020

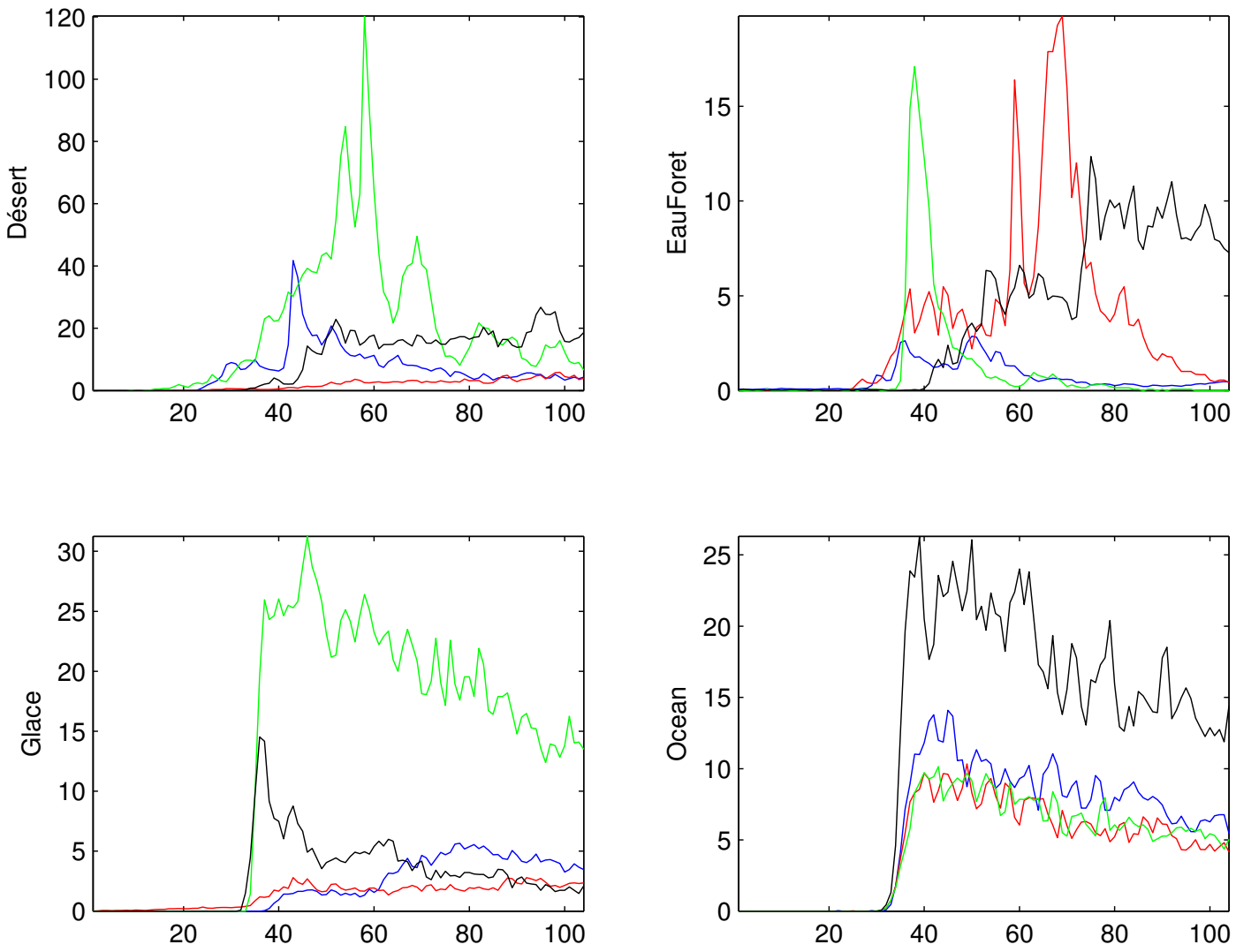
Summary

Summary

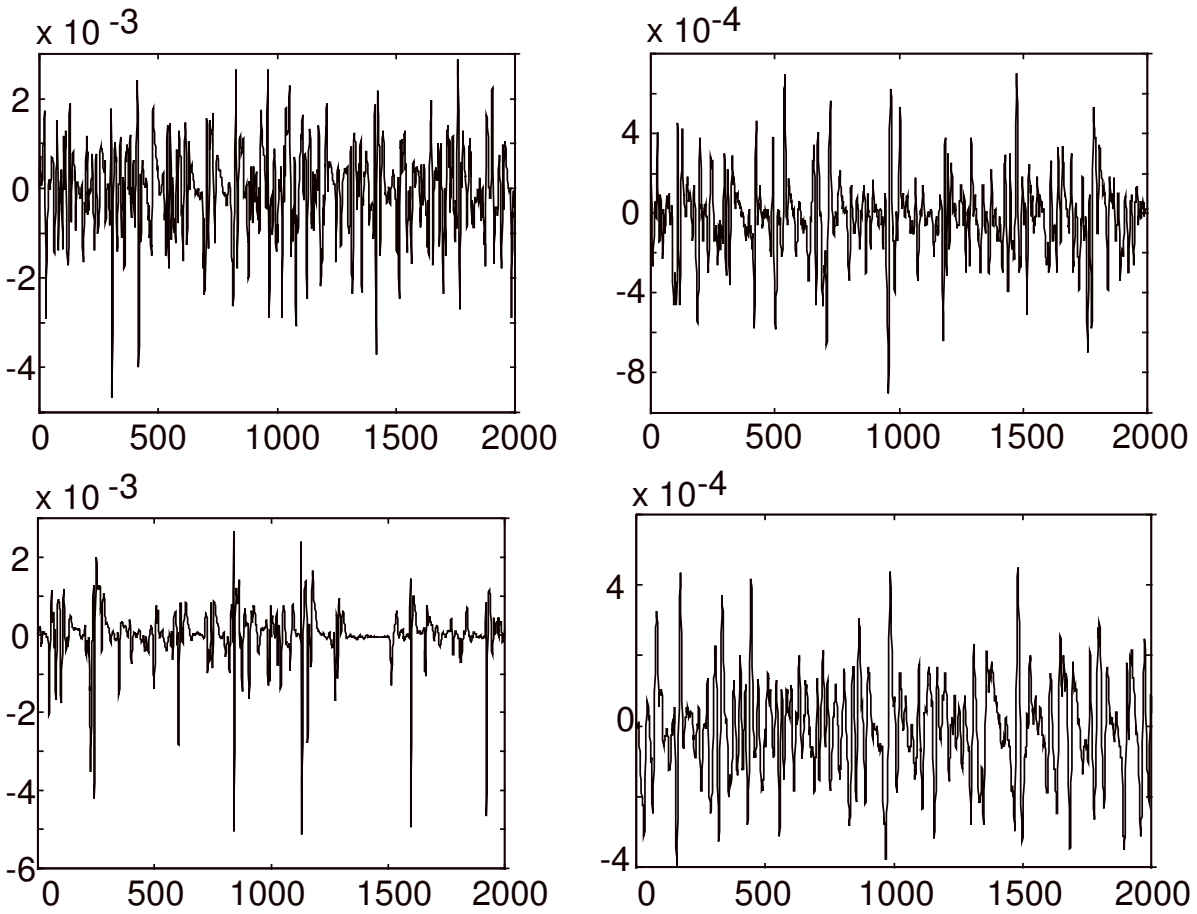
References

- R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, 2nd edition, Wiley, 2000.
- S. Theodoridis and K. Koutroumbas, Pattern Recognition, 4th edition, Academic Press, 2008.
- A. Jain, R. Duin and J. Mao, Statistical Pattern Recognition: A Review, IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4-37, Jan. 2000.

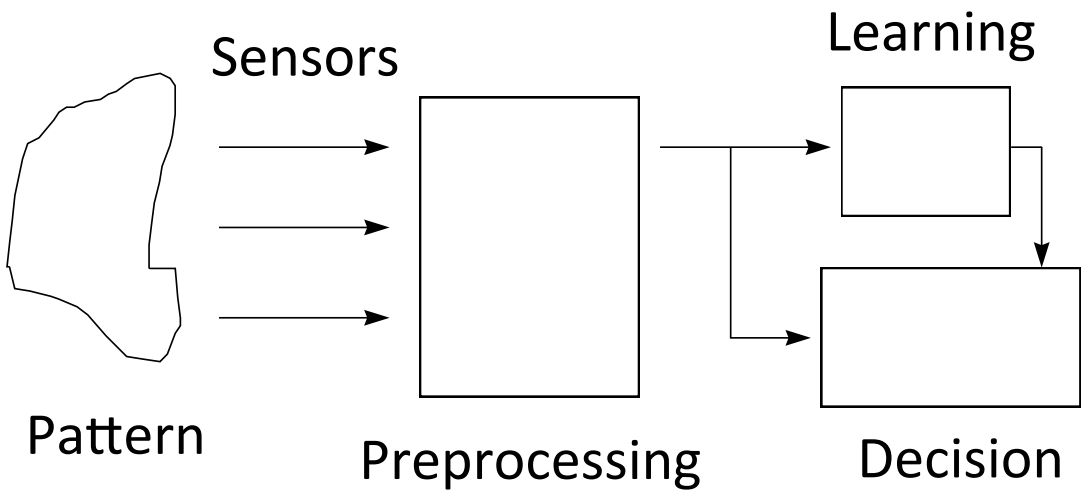
Example 1 : Classification of Altimetric Signals



Example 2 : Classification of EMG Signals



Classification Model



Summary

Classification

Notations

- K classes $\omega_1, ..., \omega_K$
- $\boldsymbol{x} = [x(1), ..., x(p)]^T$ measurements $\in X = \mathbb{R}^p$
- A: set of possible actions $a_1, ..., a_q$ where $a_i =$ "assign the vector $\boldsymbol{x}$ to the class ω_i", $\forall i = 1, ..., K$

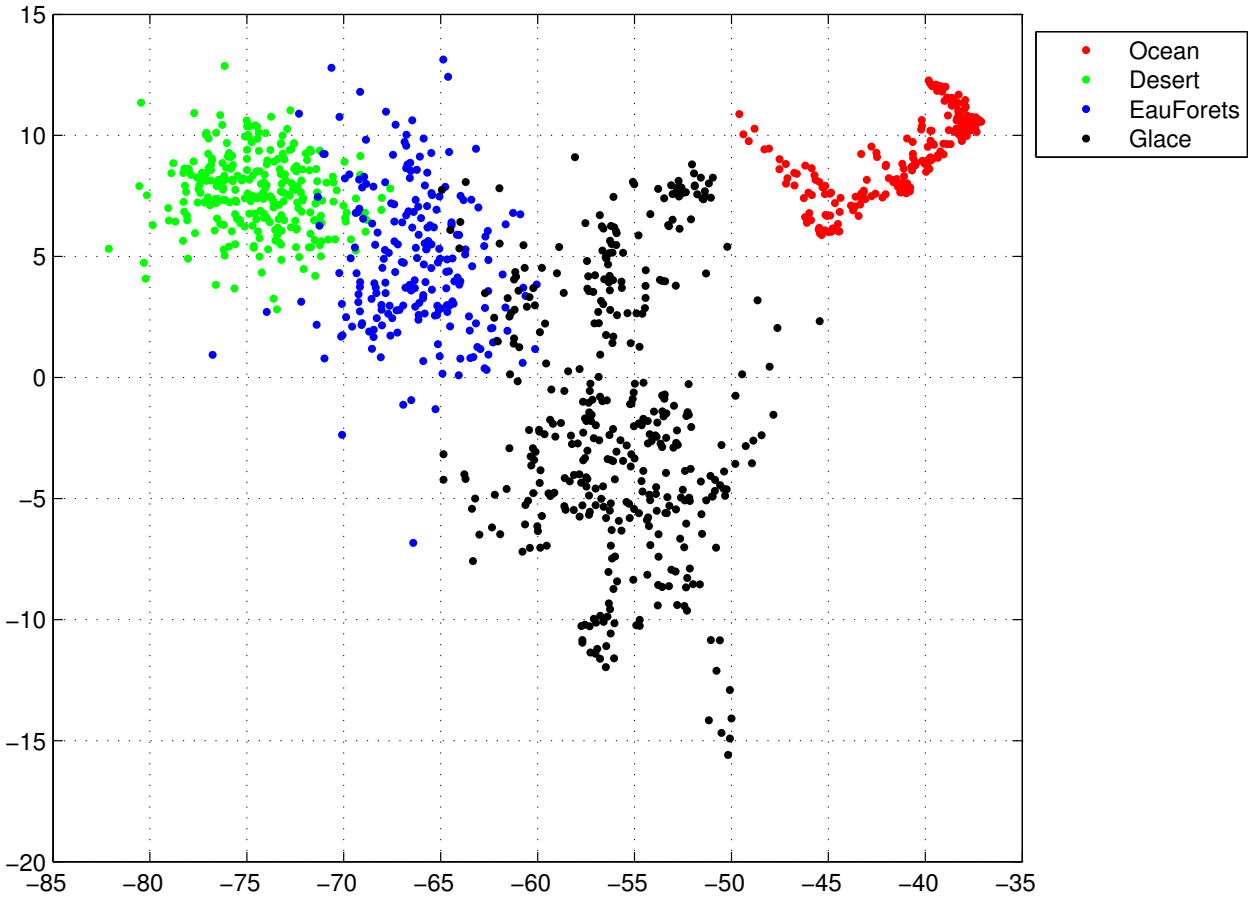
Definition

$$d : \begin{array}{l} X \to A \\ \boldsymbol{x} \mapsto d(\boldsymbol{x}) \end{array}$$

Remark

Classification with reject option: $A = \{a_0, a_1, ..., a_K\}$ where $a_0 =$ "do not classify the vector $\boldsymbol{x}$"

Example


15
10
5
0
−5
−10
−15
−20
−85
−80
−75
−70
−65
−60
−55
−50
−45
−40
−35
Ocean
Desert
EauForets
Glace

Bayesian Rule

Hypothesis: Probabilistic Model

- *A priori* probability of class ω_i

$$P(\omega_i)$$

- Probability density function of the observation vector $\boldsymbol{x}$ conditionally to class ω_i

$$f(\boldsymbol{x}|\omega_i)$$

Conclusion

- *A posteriori* probability that $\boldsymbol{x}$ belongs to class ω_i

$$P(\omega_i|\boldsymbol{x}) = \frac{f(\boldsymbol{x}|\omega_i)\,P(\omega_i)}{f(\boldsymbol{x})}$$

with $f(\boldsymbol{x}) = \sum_{i=1}^{K} f(\boldsymbol{x}|\omega_i)\,P(\omega_i)$.

MAP Classifier

Definition

$$d^{*}\left(\boldsymbol{x}\right)=a_{j}\Leftrightarrow P\left(\omega_{j}\left|\boldsymbol{x}\right.\right)\geq P\left(\omega_{k}\left|\boldsymbol{x}\right.\right),\forall k\in\left\{1,...,K\right\}$$

Equiprobable Classes: Maximum Likelihood Classifier

$$d^{*}\left(\boldsymbol{x}\right)=a_{j}\Leftrightarrow f\left(\boldsymbol{x}\left|\omega_{j}\right.\right)\geq f\left(\boldsymbol{x}\left|\omega_{k}\right.\right),\forall k\in\left\{1,...,K\right\}$$

Property

The MAP classifier minimizes the probability of error

Proof (2 classes)

$$\begin{aligned} P_e &= P\left[d\left(\boldsymbol{x}\right)=a_1 \cap \boldsymbol{x} \in \omega_2\right] + P\left[d\left(\boldsymbol{x}\right)=a_2 \cap \boldsymbol{x} \in \omega_1\right] \\ &= P\left[d\left(\boldsymbol{x}\right)=a_1 \mid \boldsymbol{x} \in \omega_2\right] P\left(\omega_2\right) + P\left[d\left(\boldsymbol{x}\right)=a_2 \mid \boldsymbol{x} \in \omega_1\right] P\left(\omega_1\right) \end{aligned}$$

Let $R_i = \{\boldsymbol{x} \in \mathbb{R}^p / d(\boldsymbol{x}) = a_i\}$ be the acceptance region for class ω_i

$$\begin{aligned} P_e &= \int_{R_1} P\left(\omega_2\right) f\left(\boldsymbol{x} \left|\omega_2\right.\right) d\boldsymbol{x} + \int_{R_2} P\left(\omega_1\right) f\left(\boldsymbol{x} \left|\omega_1\right.\right) d\boldsymbol{x} \\ &= P\left(\omega_2\right)\left[1 - \int_{R_2} f\left(\boldsymbol{x} \left|\omega_2\right.\right) d\boldsymbol{x}\right] + \int_{R_2} P\left(\omega_1\right) f\left(\boldsymbol{x} \left|\omega_1\right.\right) d\boldsymbol{x} \\ &= P\left(\omega_2\right) + \int_{R_2} \left[P\left(\omega_1\right) f\left(\boldsymbol{x} \left|\omega_1\right.\right) - P\left(\omega_2\right) f\left(\boldsymbol{x} \left|\omega_2\right.\right)\right] d\boldsymbol{x} \\ &= P\left(\omega_2\right) - \int_{R_2} \left[P\left(\omega_2 \left|\boldsymbol{x}\right.\right) - P\left(\omega_1 \left|\boldsymbol{x}\right.\right)\right] f\left(\boldsymbol{x}\right) d\boldsymbol{x} \end{aligned}$$

P_e is minimum when $R_2 = \{\boldsymbol{x} / P\left(\omega_2 \left|\boldsymbol{x}\right.\right) > P\left(\omega_1 \left|\boldsymbol{x}\right.\right)\}$

Probability of Error

- Definition (K classes)

$$P_e = \sum_{i=1}^{K} P\left[d\left(x\right) = a_i \cap x \notin \omega_i\right]$$

- Proof
  admitted

Gaussian Case

Densities

$$f\left(\boldsymbol{x}\left|\omega_i\right.\right) = \frac{1}{(2\pi)^{p/2}\sqrt{\det \Sigma_i}} \exp\left[-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{m}_i\right)^t \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{x}-\boldsymbol{m}_i\right)\right]$$

General Case

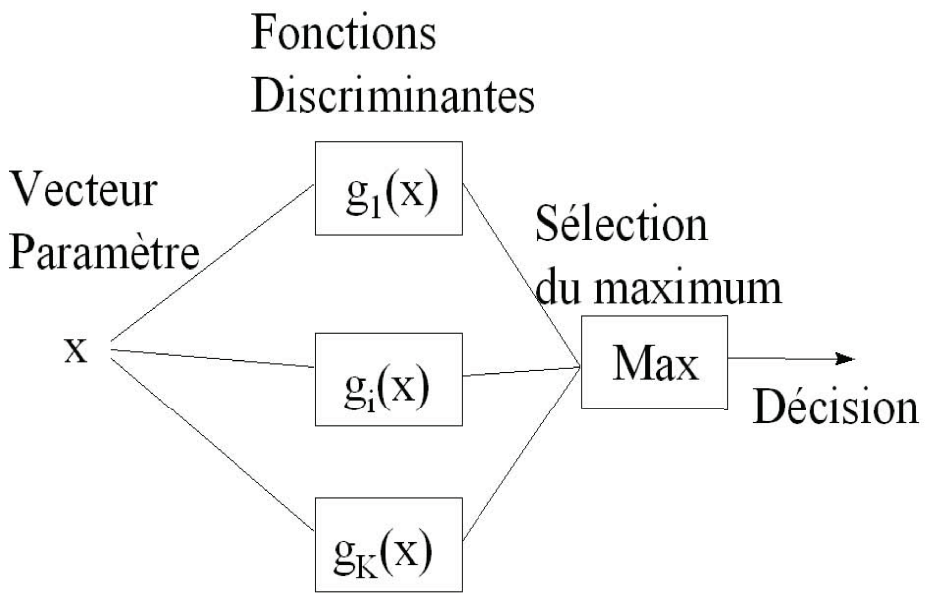
$$d^*\left(\boldsymbol{x}\right) = a_i \Leftrightarrow g_i\left(\boldsymbol{x}\right) \geq g_k\left(k\right) \qquad \forall k$$

with

$$g_i\left(\boldsymbol{x}\right) = -\left(\boldsymbol{x}-\boldsymbol{m}_i\right)^t \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{x}-\boldsymbol{m}_i\right) - \ln\det \boldsymbol{\Sigma}_i + 2\ln P\left(\omega_i\right)$$

Gaussian Case

Classifier



Discriminant functions

Quadratic term + linear term + constant

Identical covariance matrices ($\Sigma_i = \Sigma$)

Centroid Distance Rule

$$d^*(\boldsymbol{x}) = a_i \Leftrightarrow d_M(\boldsymbol{x}, \boldsymbol{m}_i) \leq d_M(\boldsymbol{x}, \boldsymbol{m}_k) \qquad \forall k$$

where d_M is the Mahalanobis distance

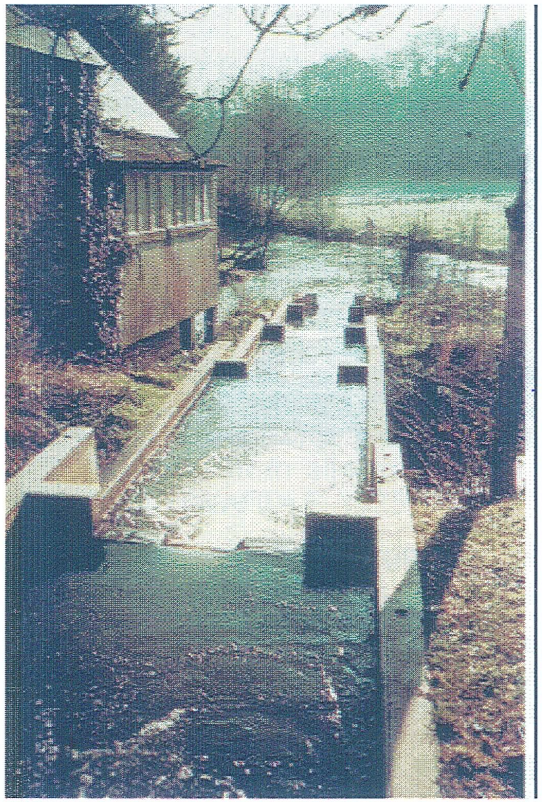
$$d_M(\boldsymbol{x}, \boldsymbol{m}_k) = \sqrt{(\boldsymbol{x} - \boldsymbol{m}_k)^t \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{m}_k)}$$

Affine discriminant functions

$$d^*(\boldsymbol{x}) = a_i \Leftrightarrow \left(\boldsymbol{x} - \frac{1}{2}(\boldsymbol{m}_i + \boldsymbol{m}_k)\right)^t \boldsymbol{\Sigma}^{-1} (\boldsymbol{m}_i - \boldsymbol{m}_k) \geq \ln \frac{P(\omega_k)}{P(\omega_i)}$$

Contexte de l'étude

Les passes à poissons



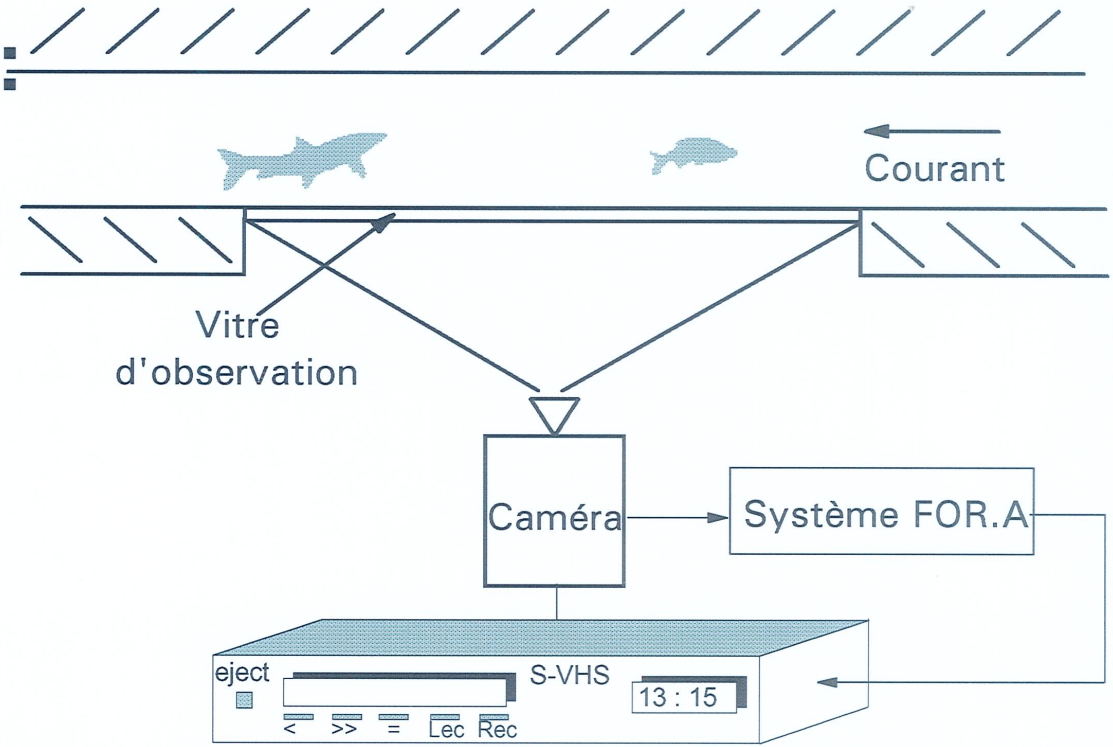
à bassins successifs

à ralentisseurs

ascenseur

Suivi des passes à poissons

- Dispositif vidéo actuel de comptage par espèces :

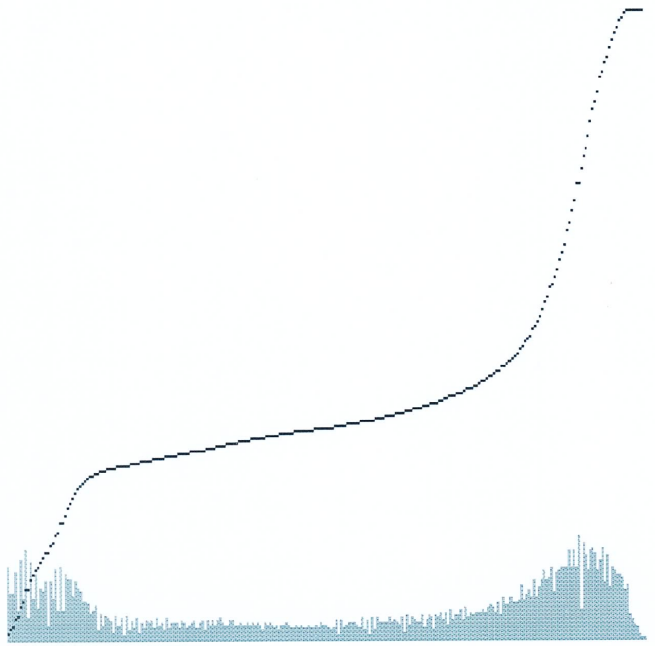


- Dépouillement des bandes vidéo enregistrées par un opérateur humain

Acquisition des images

Nouvelles conditions d'acquisition

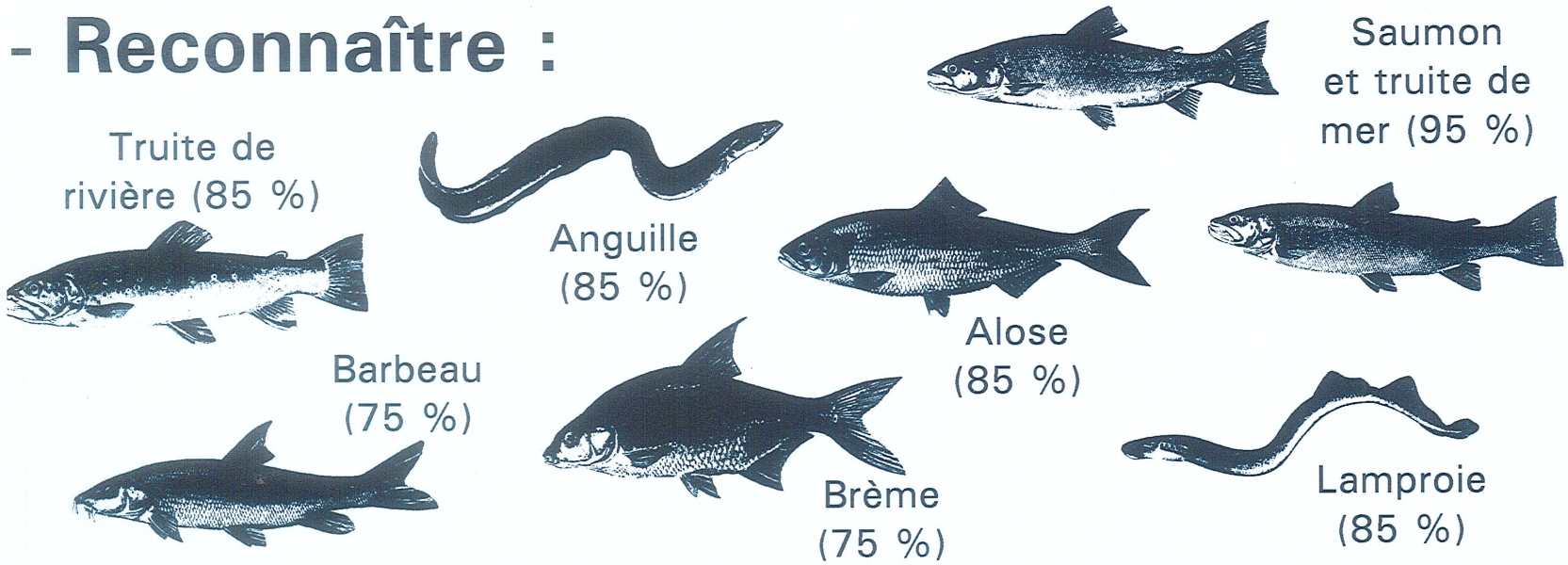
Image



Histogramme : bimodal

Cahier des charges du système de vision automatique

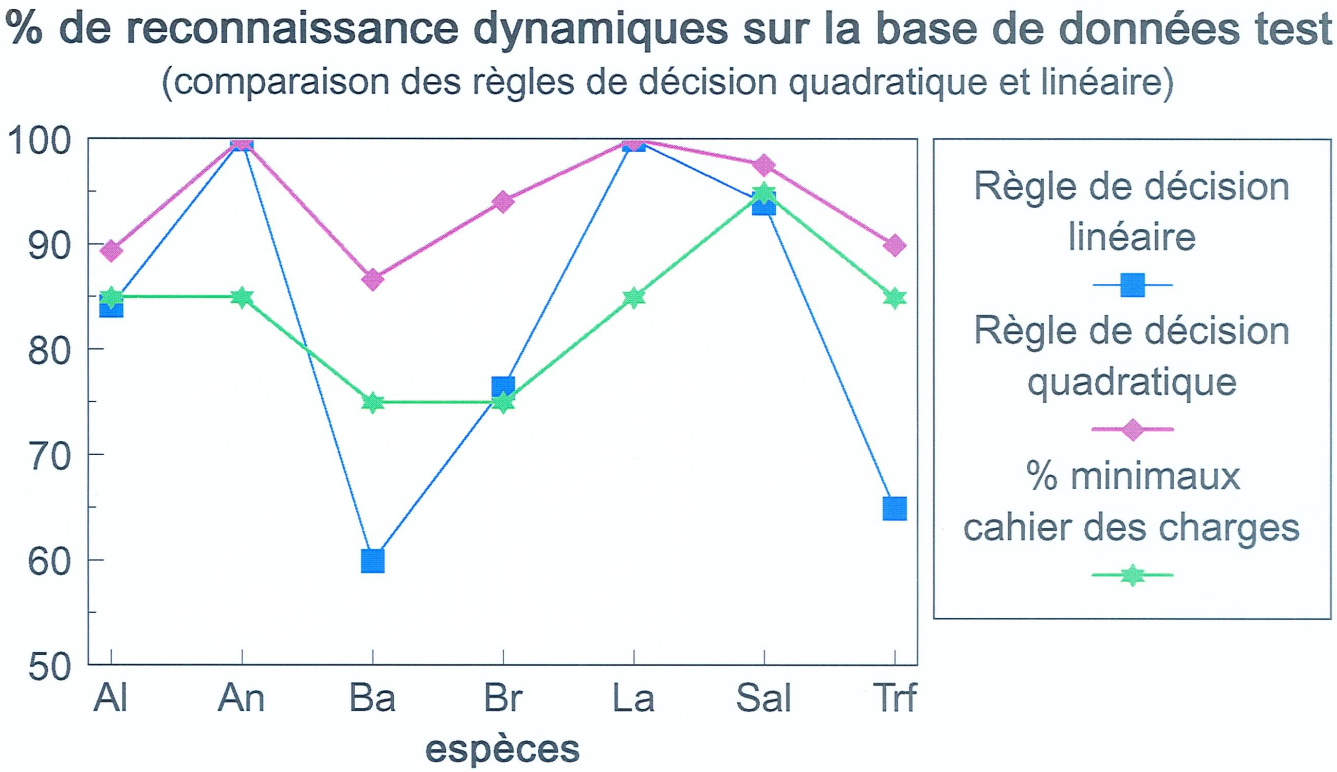
- Compter les poissons, toutes espèces confondues, au delà d'une certaine taille
- Reconnaître :



- Fournir : La comptabilisation des différentes espèces
  Les dates et heures de passage

Mesure d'efficacité de la règle de classement

Résultats obtenus sur la BDT



Summary

Parametric methods

Principle

Assume that the distribution of $\boldsymbol{x} = (x_1, ..., x_n)^T$ is characterized by a parametric probability density function, which depends on an unknown parameter vector $\theta \in \mathbb{R}^p$ and estimate this parameter vector using a method considered in statistics

- Maximum likelihood method
- Method of moments
- MMSE or MAP estimators

Summary

Histogram

Density estimation

- Probability: P_R
  Let $\boldsymbol{x}$ be a random vector of $\mathbb{R}^p$

$$P_R = P\left[\boldsymbol{x} \in R\right] = \int_R f\left(\boldsymbol{u}\right) d\boldsymbol{u}$$

- Number of vectors $\boldsymbol{x}_i$ belonging to the region R: N_R

$$P\left[N_R = k\right] = C_n^k P_R^k \left(1 - P_R\right)^{n-k} \qquad k \in \{0, ..., n\}$$

$$E\left(\frac{N_R}{n}\right) = P_R, \quad \text{Var}\left(\frac{N_R}{n}\right) = \frac{P_R\left(1 - P_R\right)}{n}$$

N_R/n is an unbiased and convergent estimator of P_R.

- Estimator of the probability density function: $f\left(\boldsymbol{x}\right)$
  If f is approximately constant on R

$$\int_R f\left(\boldsymbol{u}\right) d\boldsymbol{u} \simeq f\left(\boldsymbol{x}\right) V_R \Rightarrow \boxed{\widehat{f}\left(\boldsymbol{x}\right) = \frac{N_R}{nV_R}}$$

Parzen windows

Definition of a kernel ϕ

Let R_k be an hypercube of side h_k and dimension p. Thus $V_k = h_k^p$ and

$$N_k = \sum_{i=1}^{k} \phi\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h_k}\right)$$

with

$$\begin{aligned} &\phi(\boldsymbol{u}) = 1 \text{ if } |u(j)| \leq 1/2 \qquad \forall j \in \{1, ..., p\} \\ &\phi(\boldsymbol{u}) = 0 \text{ else} \end{aligned}$$

Indeed, $\phi\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_k}\right) = 1$ if $\boldsymbol{x}_i$ is in the region R_k.

Density estimator

$$\widehat{f}_k(x) = \frac{1}{k h_k^p} \sum_{i=1}^{k} \phi\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h_k}\right)$$

k-nearest neighbor method

Assumptions

N_k is fixed (e.g., N_k is the integer part of $\sqrt{k}$)
V_k is estimated

Density estimator

$$\widehat{f}_k(\boldsymbol{x}) = \frac{N_k}{kV_k}$$

Convergence

The estimator $\widehat{f}_k(\boldsymbol{x})$ converges to $f(\boldsymbol{x})$ if and only if

$$\begin{aligned} \lim_{k\to\infty} N_k &= \infty \\ \lim_{k\to\infty} \frac{N_k}{k} &= 0 \end{aligned}$$

The k-nearest neighbor rule

The nearest neighbor rule

$$\boxed{d(x) = a_j \text{ if the nearest neighbor of } x \text{ belongs to } \omega_j}$$

The observed vector $\boldsymbol{x}$ is affected to the class of its nearest neighbor.

Inequality of Cover and Hart

$$P^* \leq P_1 \leq P^* \left(2 - \frac{K}{K-1} P^*\right)$$

The k-nearest neighbor rule

$\boldsymbol{x}$ is assigned to the class most common amongst its k-nearest neighbors (with a given distance measure)

Probability of error

Inequalities

$$P^* \leq P_k \leq P^* + \frac{1}{\sqrt{ke}} \text{ or } P^* \leq P_k \leq P^* + \sqrt{\frac{2P_1}{k}}$$

Approximations

When P^* is small, the following results can be obtained

$$P_1 \approx 2P^* \text{ and } P_3 \approx P^* + 3\left(P^*\right)^2$$

Comparing and selecting classifiers.

- We just saw how a "k-nearest-neighbors" (knn) classifier works and the rationale behind such a non-parametric method.
- How should we select k? is a remaining question.

- Let's consider a collection of training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$ where $y_i \in \{\omega_1, \dots, \omega_K\}$ are the annotated classes corresponding to the feature vectors $\mathbf{x}_i$ from $\mathbb{R}^p$.
- We now denote $f^D_{\text{knn}}(x)$ the knn classifier that assigns to the input vector $x \in \mathbb{R}^p$ the class most common among its "k-nearest-neighbors" in D thanks to a distance δ in $\mathbb{R}^p$.
- Example: assuming a classification task with $(K = 2)$ classes such that $\delta(\mathbf{x}_{i_1}, x)$, $\delta(\mathbf{x}_{i_2}, x)$ and $\delta(\mathbf{x}_{i_3}, x)$ are the 3 smallest distances among $\{(\delta(\mathbf{x}_i, x))\}_{i=1,\dots,n}$ then $f^D_{\text{3nn}}(x)$ returns ω_1 (or ω_2) the majority class of $\{y_{i_1}, y_{i_2}, y_{i_3}\}$.

Comparing $f^D_{k\text{nn}}(x)$ and $f^D_{k'\text{nn}}(x)$

- Intuitively, $f^D_{5\text{nn}}(x)$ is "smoother" than $f^D_{1\text{nn}}(x)$
- However, $f^D_{5\text{nn}}(x)$ could be more "biased" than $f^D_{1\text{nn}}(x)$
- We aim at finding the "best" k and the "best" $f^D_{k\text{nn}}(x)$.

- Let's start with a couple of new (supervised) data: $(\mathbf{x}_{\text{new}}, y_{\text{new}}) \notin D$
- Let's consider the cost of making the decision $f^D_{k\text{nn}}(\mathbf{x}_{\text{new}})$ (resp. $f^D_{k'\text{nn}}(\mathbf{x}_{\text{new}})$) knowing that $\mathbf{x}_{\text{new}}$ belongs to class y_{new}
- We compare the costs $c(f^D_{k\text{nn}}(\mathbf{x}_{\text{new}}), y_{\text{new}})$ and $c(f^D_{k'\text{nn}}(\mathbf{x}_{\text{new}}), y_{\text{new}})$ (e.g., $c(f(x), y) = 1$ if $f(x) \neq y$ and $c(f(x), y) = 0$ else).

- More generally, let's take m extra (new) data for testing $T = \left\{(\mathbf{x}'_j, y'_j)\right\}_{j=1,\ldots,m}$
- Compare predictive performance score
- $\frac{1}{m}\sum_{j=1}^{m} c(f^D_{k\text{nn}}(\mathbf{x}'_j), y'_j)$
- $\frac{1}{m}\sum_{j=1}^{m} c(f^D_{k'\text{nn}}(\mathbf{x}'_j), y'_j)$

S-fold Cross-Validation

Dilemma

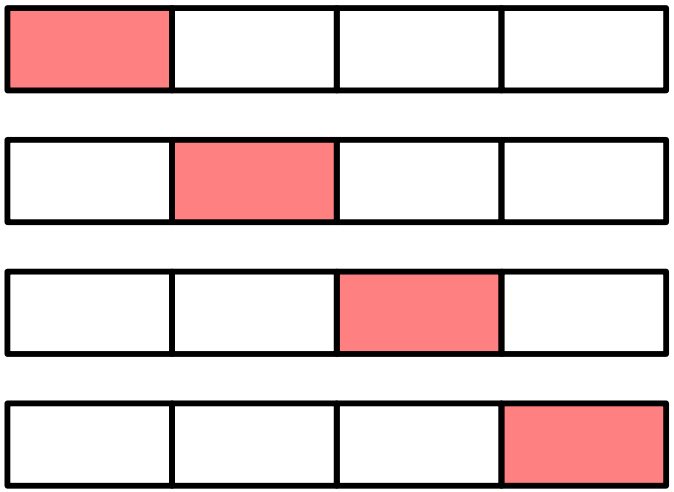
- In many applications, supplying data for both training and testing is difficult.
- We also wish to use as much of the available data as possible for training.
- However, if the test/validation set is small, it will give a relatively noisy estimate of predictive performance!

S-fold cross-validation

- S-fold cross-validation involves taking the available data and partitioning it into S groups. Then $S-1$ of the groups are used to train a set of models that are then evaluated/tested on the remaining group (termed ζ in the sequel).
- This procedure is then repeated for all S possible choices for the held-out group, indicated here by the red blocks, and the performance scores from the S runs are then averaged.

S-fold Cross-Validation

4-fold cross-validation



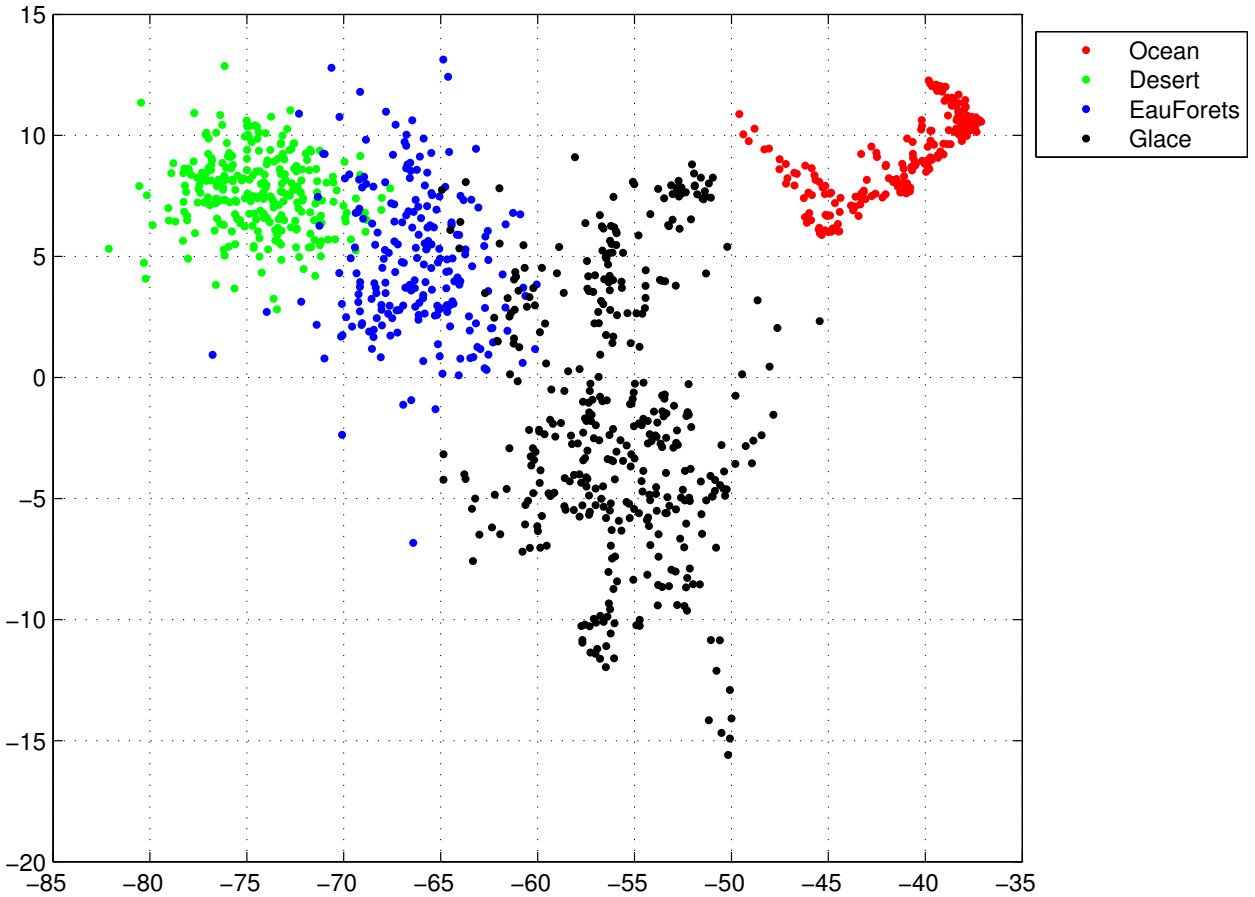
CV score for each k and $\zeta(j)$ is the jth group used as testing data

$$CV(k) = \frac{1}{S} \sum_{\textbf{run } j=1}^{S} \frac{1}{|\zeta(j)|} \sum_{(\mathbf{x}',y') \in \zeta(j)} c(f_{k\text{nn}}^{D-\zeta(j)}(\mathbf{x}'), y')$$

Leave-One-Out (LOO) score for each k

$$\text{CV}(k) = \frac{1}{n} \sum_{i=1}^{n} c(f_{k\text{nn}}^{D-\{(\boldsymbol{x_i},y_i)\}}(\boldsymbol{x_i}), y_i)$$

Example


15
10
5
0
−5
−10
−15
−20
−85
−80
−75
−70
−65
−60
−55
−50
−45
−40
−35
Ocean
Desert
EauForets
Glace

Some classification results

Bayesian Classification (Gaussian densities)

Classes	*Ocean*	*Desert*	*Forest*	*Ice*
Ocean	**99.6**	0.0	0.0	0.0
Desert	0.0	**95.3**	1.8	0.0
Forest	0.0	4.4	**97.7**	0.8
Ice	0.4	0.4	0.5	**99.2**

1-PPV Method

Classes	*Ocean*	*Desert*	*Forest*	*Ice*
Ocean	**100**	0.0	0.0	0.0
Desert	0.0	**96.0**	5.4	0.0
Forest	0.0	4.0	**93.2**	0.0
Ice	0.0	0.0	1.4	**100**

Some classification results

5-PPV Method

Classes	*Ocean*	*Desert*	*Forest*	*Ice*
Ocean	**100**	0.0	0.0	0.0
Desert	0.0	**93.8**	5.9	0.3
Forest	0.0	6.2	**93.2**	0.0
Ice	0.0	0.0	0.9	**99.7**

Neural Networks

Classes	*Ocean*	*Desert*	*Forest*	*Ice*
Ocean	**100**	0.0	0.0	0.0
Desert	0.0	**96.0**	5.4	0.0
Forest	0.0	4.0	**92.8**	0.8
Ice	0.0	0.0	1.8	**99.2**

Summary

Unsupervised learning

N unlabelled data vectors of $\mathbb{R}^p$ denoted as $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ which should be split into K classes $\omega_1, ..., \omega_K$.

Motivations

- Supervised learning is costly
- The classes can change with time
- Provide some information about the data structure

Optimal solution

Number of partitions of $\boldsymbol{X}$ in K subsets

$$P(N, K) = \frac{1}{K!} \sum_{i=0}^{K} i^N (-1)^{K-i} C_K^i \qquad K < N$$

Example: $P(100, 5) \approx 10^{68}$!

Partition with minimum mean square error

Mean square error of a partition

Mean square error (MSE) of the class ω_i and of the partition $\boldsymbol{X}$

$$E_i^2 = \sum_{k=1}^{N_i} d^2\left(\boldsymbol{x}_k, \boldsymbol{g}_i\right), \qquad E^2 = \sum_{i=1}^{K} E_i^2$$

where $\boldsymbol{g}_i = \frac{1}{N_i}\sum_{k=1}^{N_i} \boldsymbol{x}_k$ is the centroid of the class ω_i

Properties

$$\sum_{k=1}^{N_i} d^2\left(\boldsymbol{x}_k, \boldsymbol{y}\right) = \sum_{k=1}^{N_i} d^2\left(\boldsymbol{x}_k, \boldsymbol{g}_i\right) + N_i d^2\left(\boldsymbol{g}_i, \boldsymbol{y}\right)$$

In particular, for $\boldsymbol{y} = \boldsymbol{g}$ (data centroid), we obtain

$$\sum_{i=1}^{K}\sum_{k=1}^{N_i} d^2\left(\boldsymbol{x}_k, \boldsymbol{g}\right) = \underbrace{\sum_{i=1}^{K}\sum_{k=1}^{N_i} d^2\left(\boldsymbol{x}_k, \boldsymbol{g}_i\right)}_{E^2} + \sum_{i=1}^{K} N_i d^2\left(\boldsymbol{g}_i, \boldsymbol{g}\right)$$

MSE of $\boldsymbol{X}$ = within-class MSE + between-class MSE

K-means Algorithm (ISODATA)

Search a partition of $\boldsymbol{X}$ ensuring a local minimum of E^2

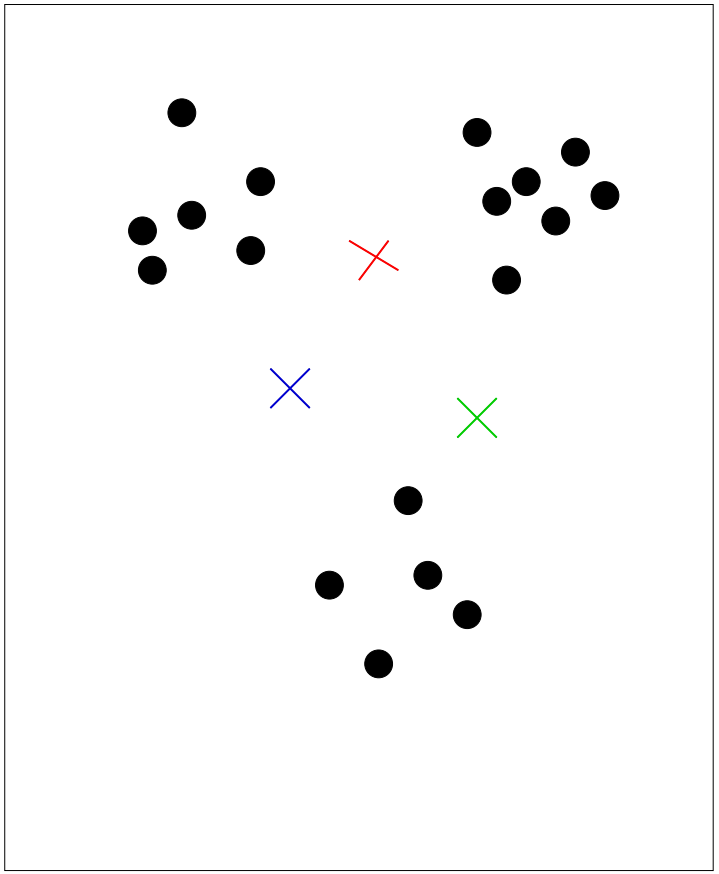
1. Initial choice of the number of classes and the class centroids
2. Assign each vector $\boldsymbol{x}_i$ to ω_j (using the centroid distance rule) such that

$$d\left(\boldsymbol{x}_i, \boldsymbol{g}_j\right) = \inf_k d\left(\boldsymbol{x}_i, \boldsymbol{g}_k\right)$$

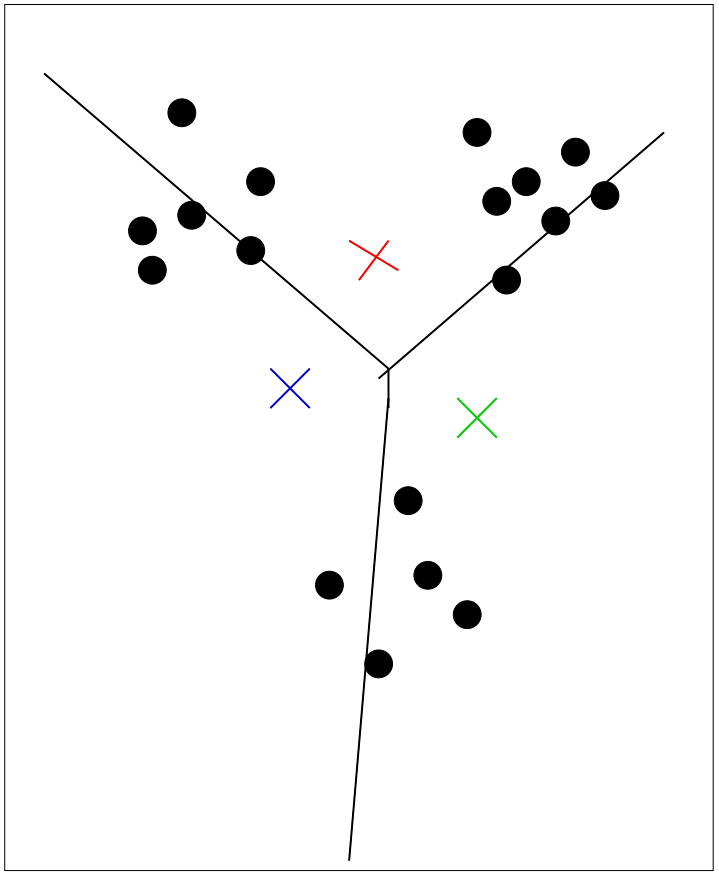
3. Compute the centroids $\boldsymbol{g}_k^*$ of the new classes ω_k^*
4. Repeat steps 2) and 3) until convergence

- *Improved version of ISODATA*
  - Two classes are merged if their centroids are close
  - A class is split if it contains too many vectors x_i or if its mean square error is too large
- *Convergence*: see notes or textbooks

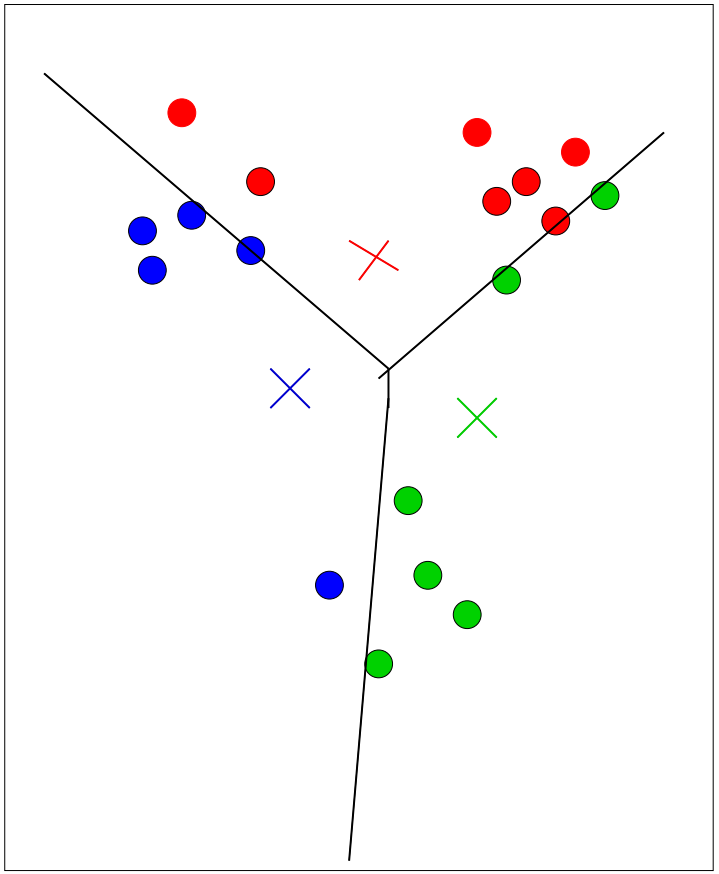
K-means



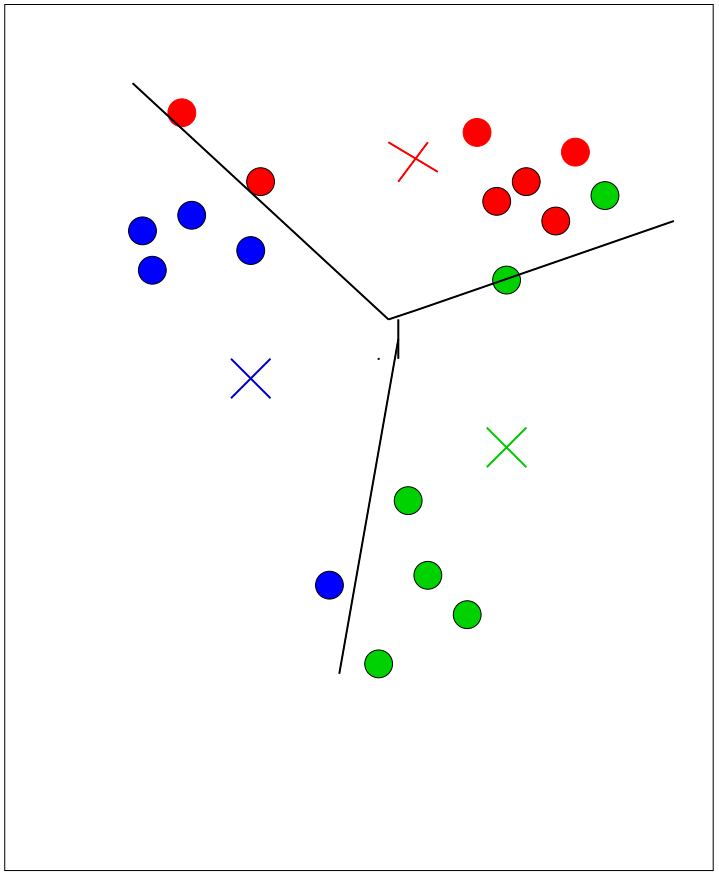
K-means



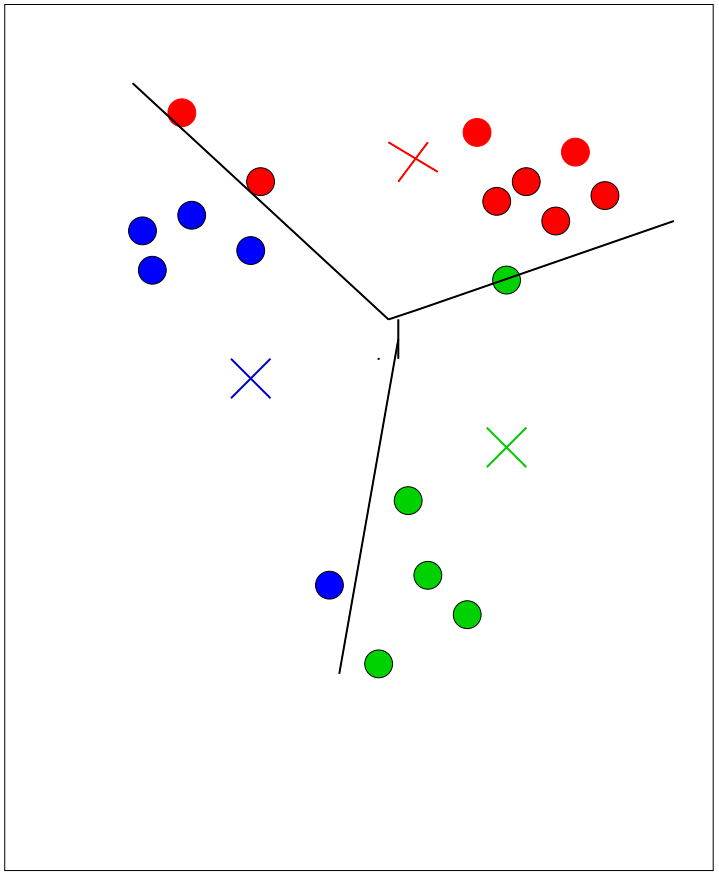
K-means



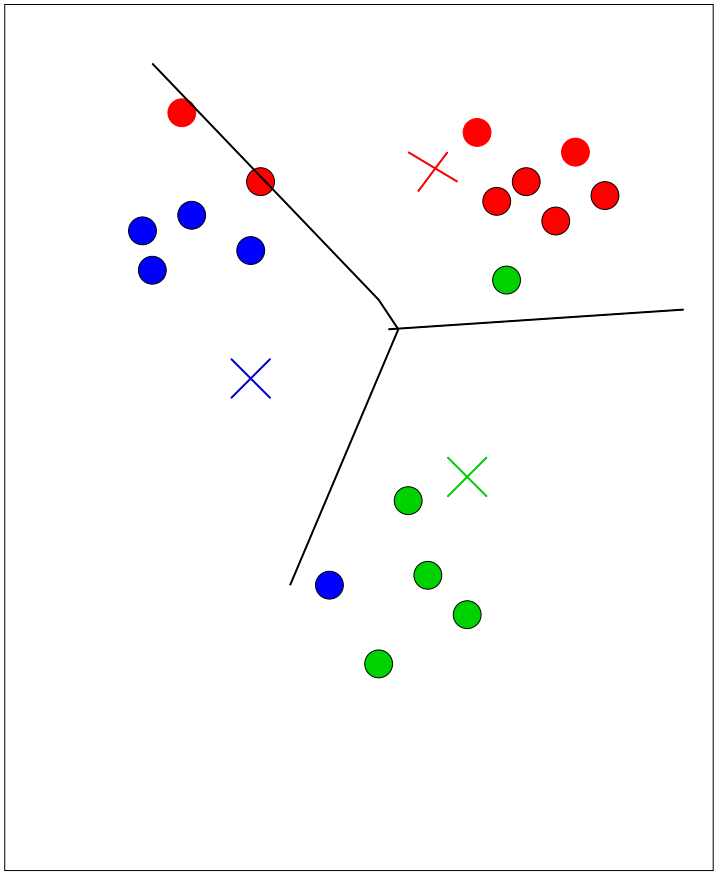
K-means



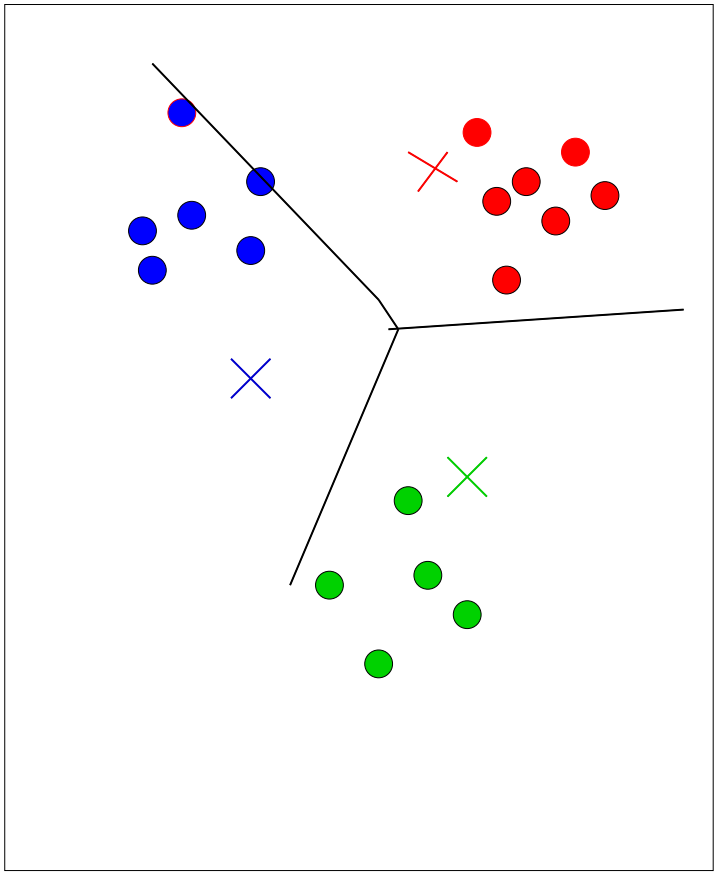
K-means



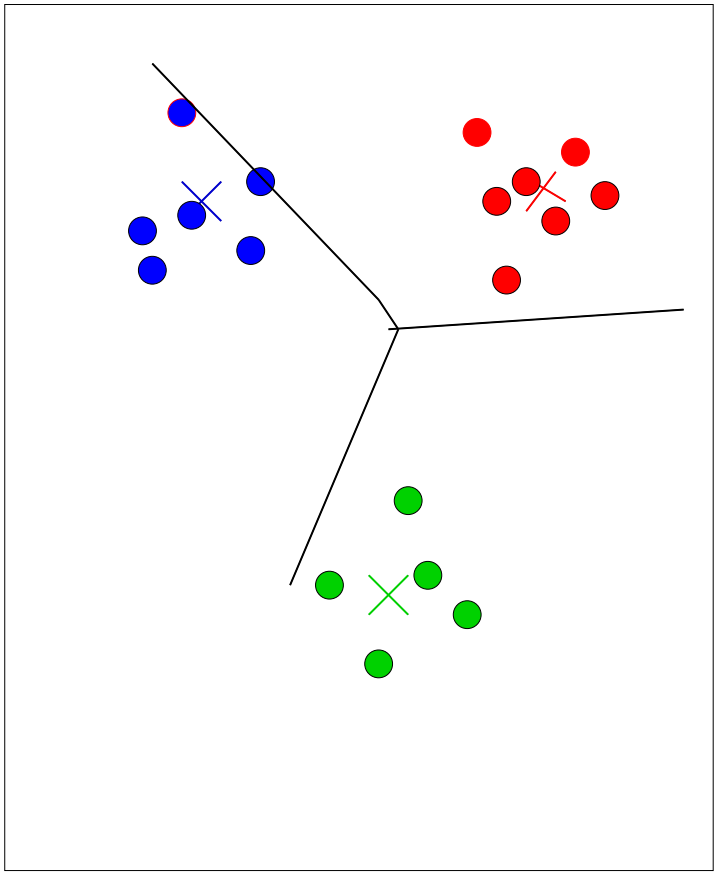
K-means



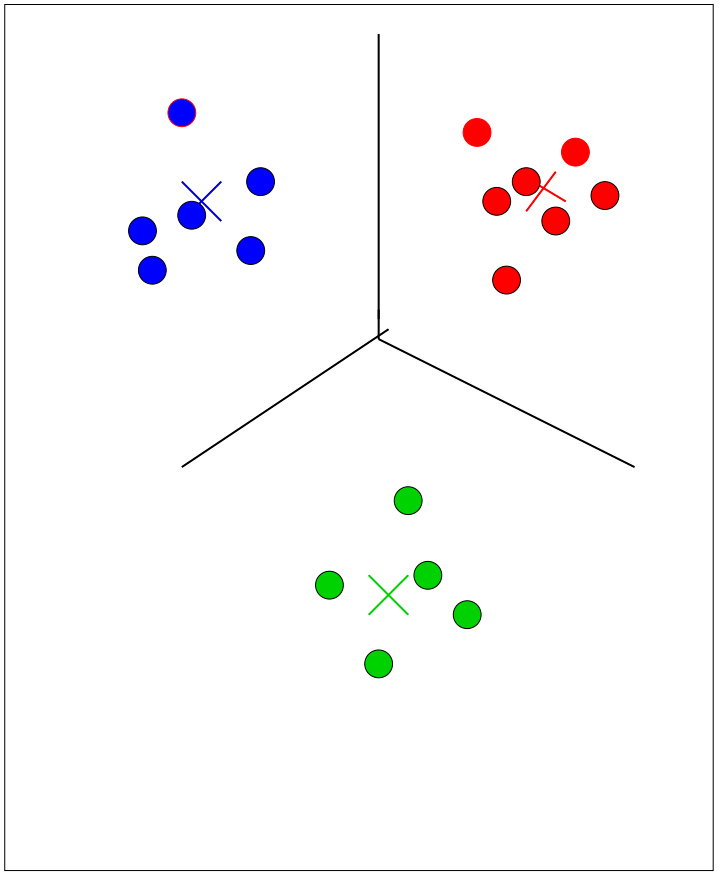
K-means



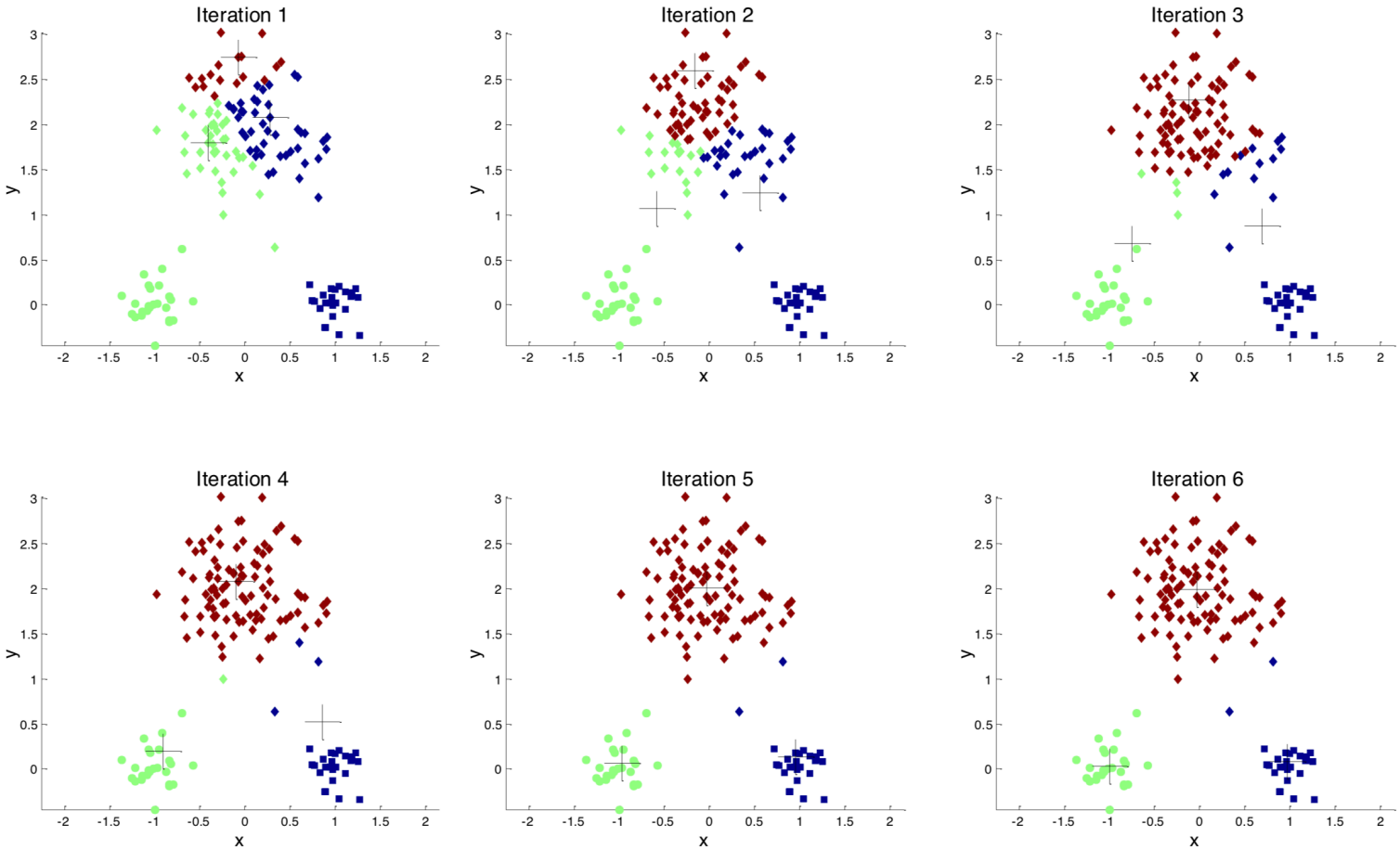
K-means



K-means

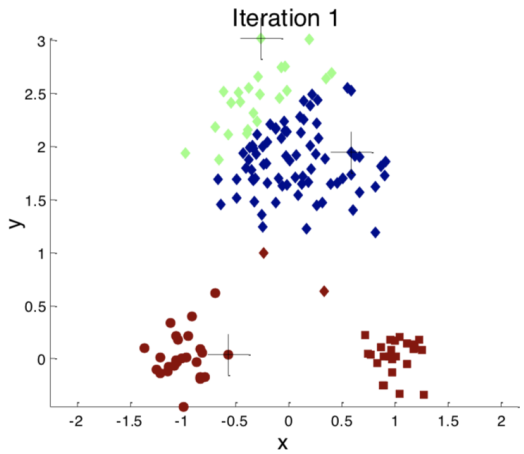


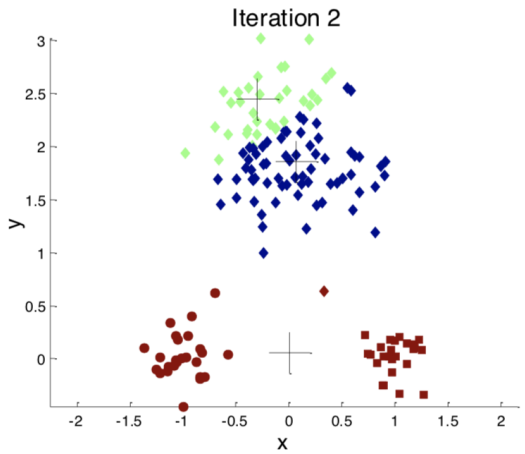
Initialization Problems

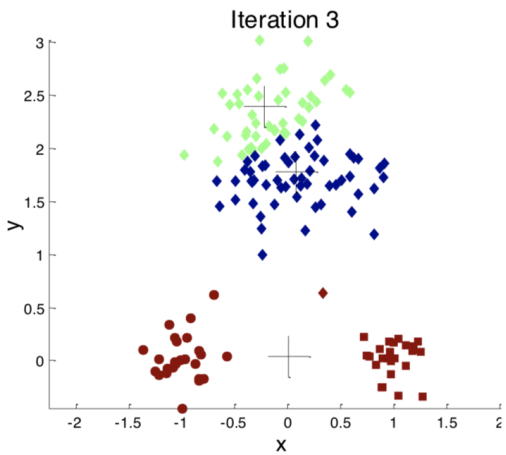


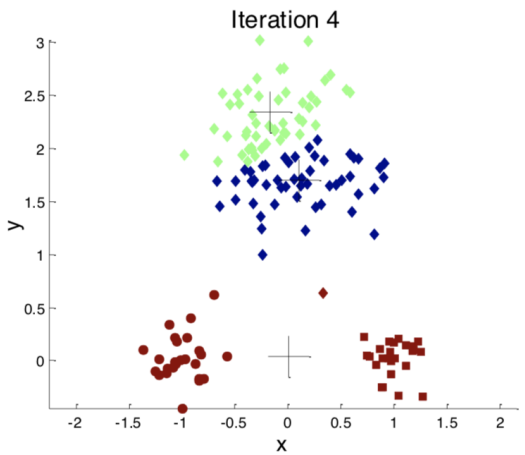
Thanks to Jing Gao from SUNY Buffalo university for her slides!!

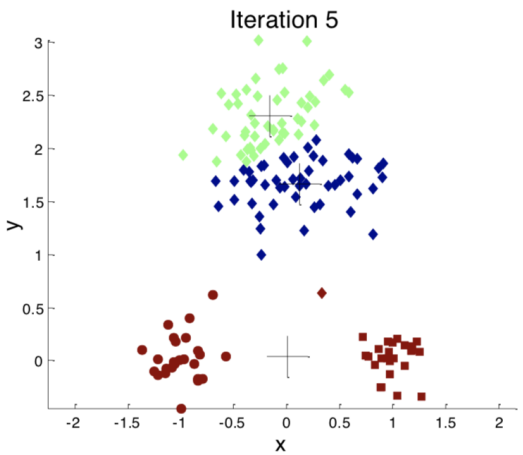
Initialization Problems



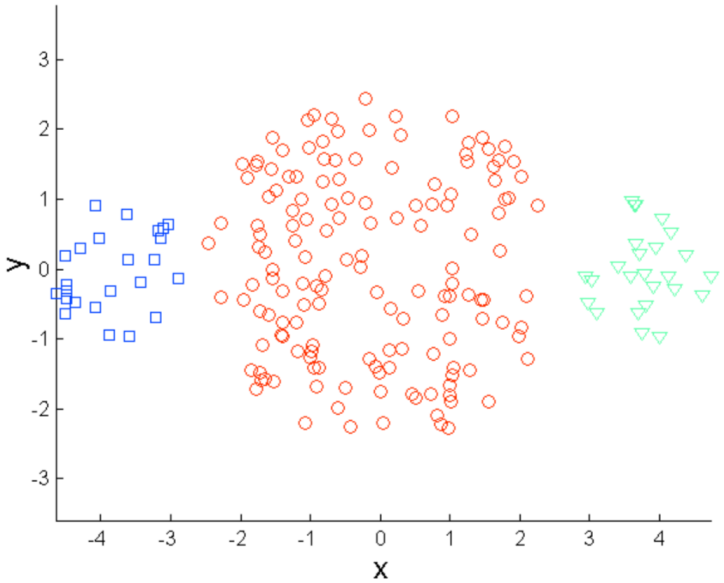




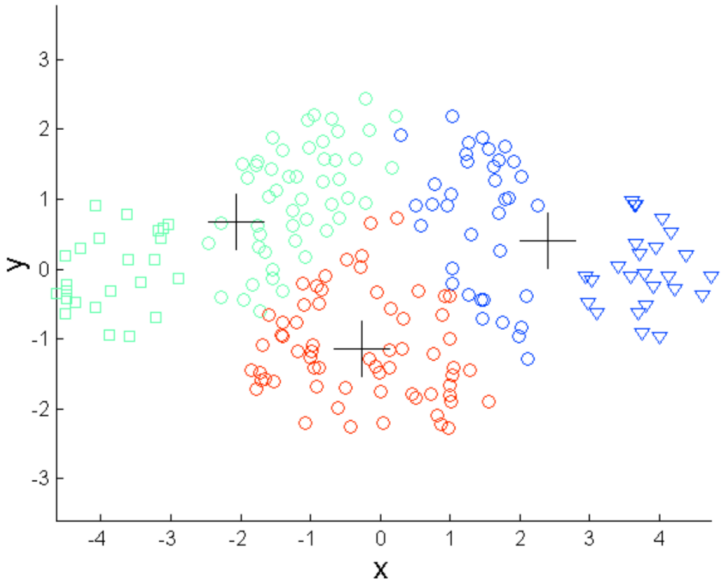




Limitations of K-means: Differing Sizes

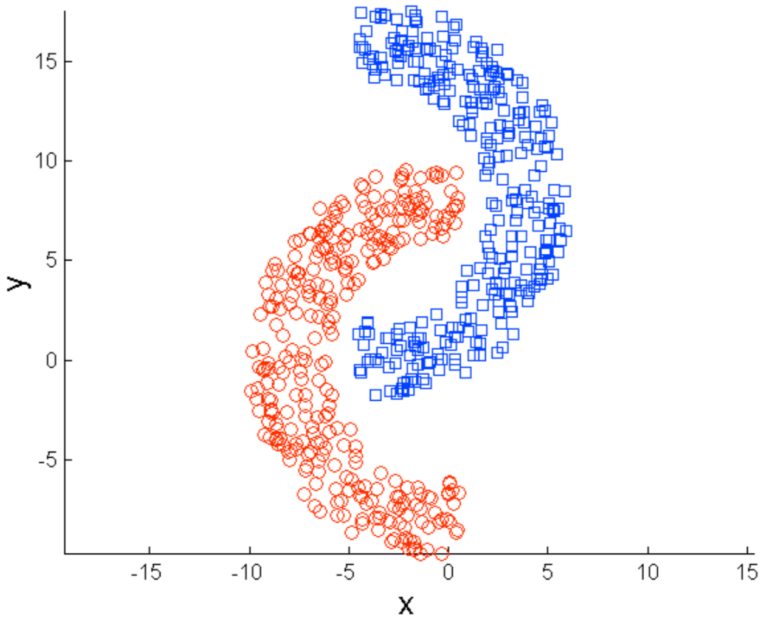


Original Points

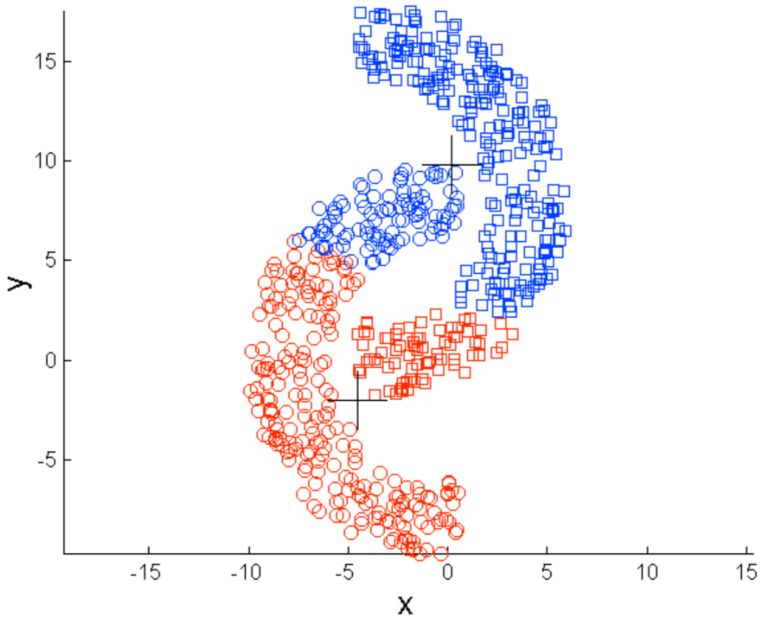


K-means (3 Clusters)

Limitations of K-means: Irregular Shapes

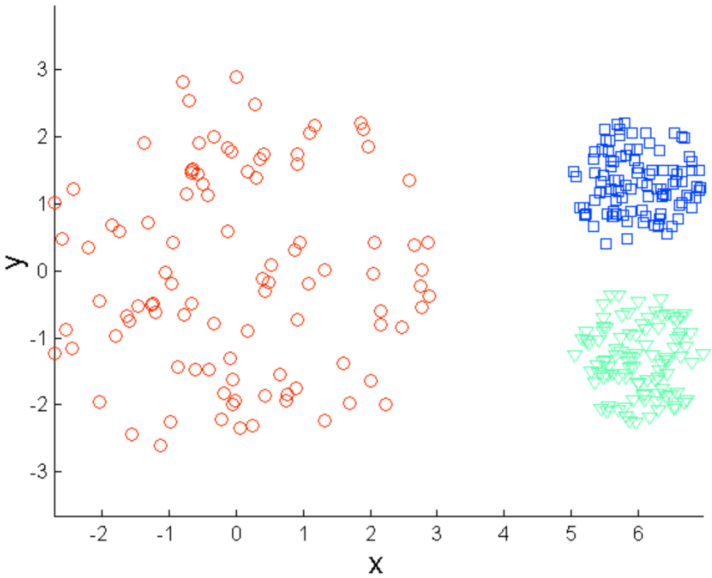


Original Points

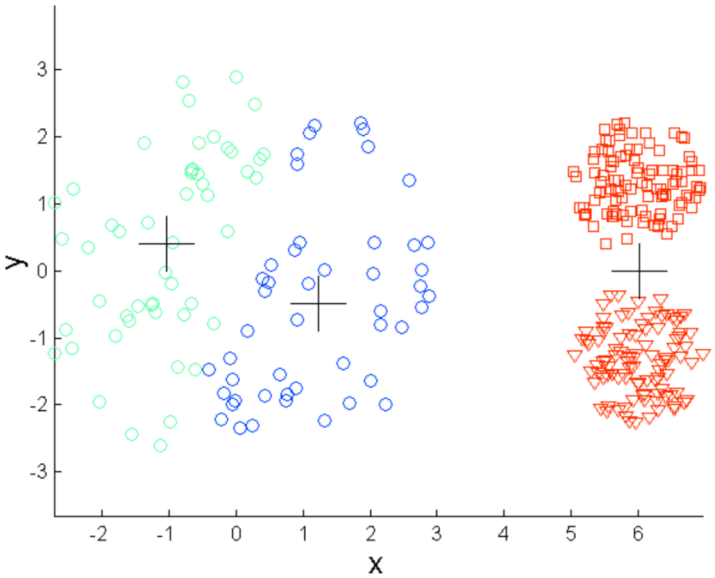


K-means (2 Clusters)

Limitations of K-means: Differing Density



Original Points



K-means (3 Clusters)

Summary

Hierarchical classification

Ascending hierarchy: method of distances

- Distance *Min* (single linkage algorithm)

$$d(X_i, X_j) = \min d(\boldsymbol{x}, \boldsymbol{y}) \qquad \boldsymbol{x} \in X_i, \boldsymbol{y} \in X_j$$

  This distance favors elongated classes

- Distance *Max* (complete linkage algorithm)

$$d(X_i, X_j) = \max d(\boldsymbol{x}, \boldsymbol{y}) \qquad \boldsymbol{x} \in X_i, \boldsymbol{y} \in X_j$$

- Average linkage algorithm

$$d(X_i, X_j) = \frac{1}{N_i N_j} \sum_{\boldsymbol{x} \in X_i, \boldsymbol{y} \in X_j} d(\boldsymbol{x}, \boldsymbol{y})$$

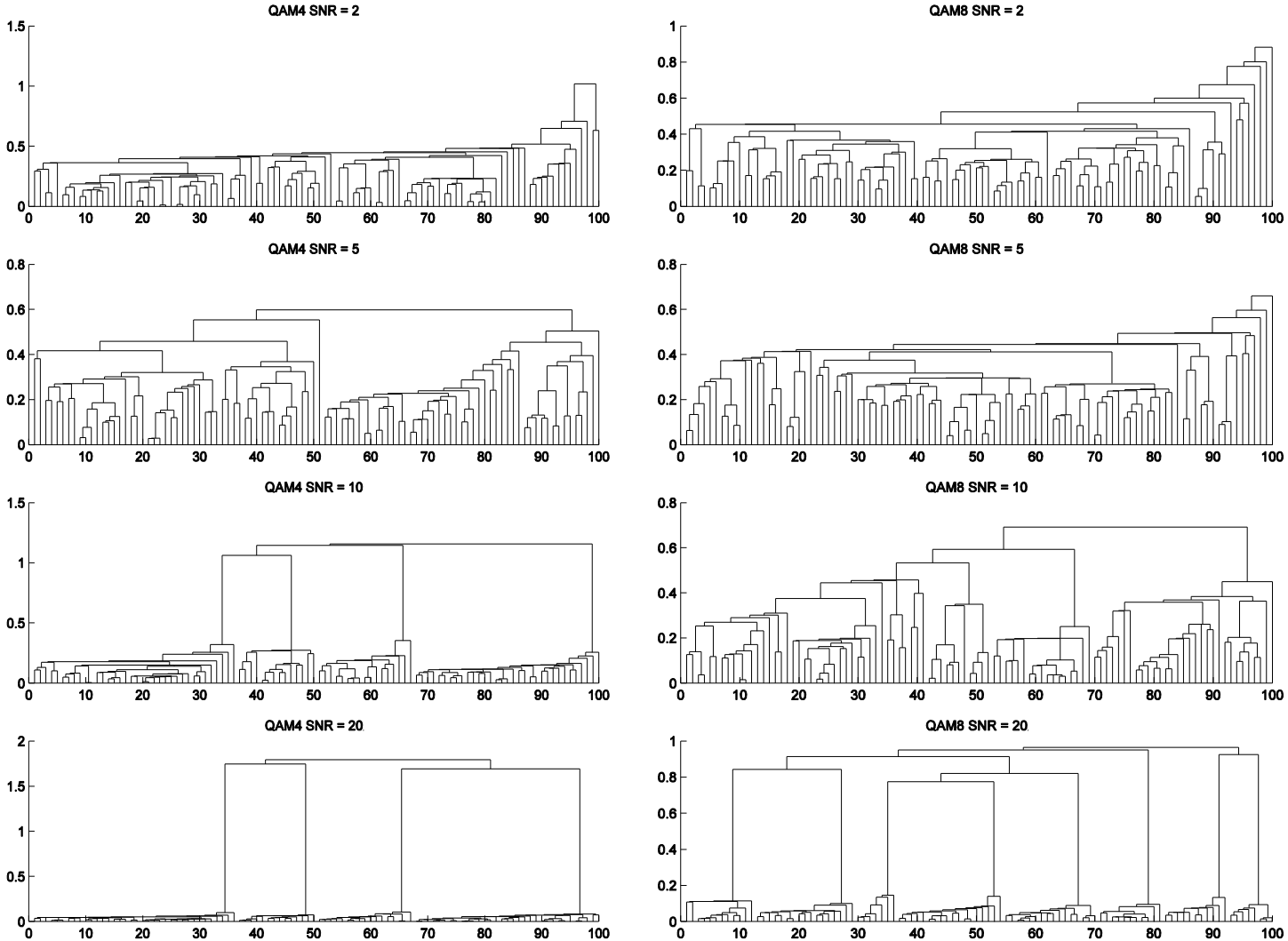
- Distance between the means

$$d(X_i, X_j) = d(\boldsymbol{g}_i, \boldsymbol{g}_j)$$

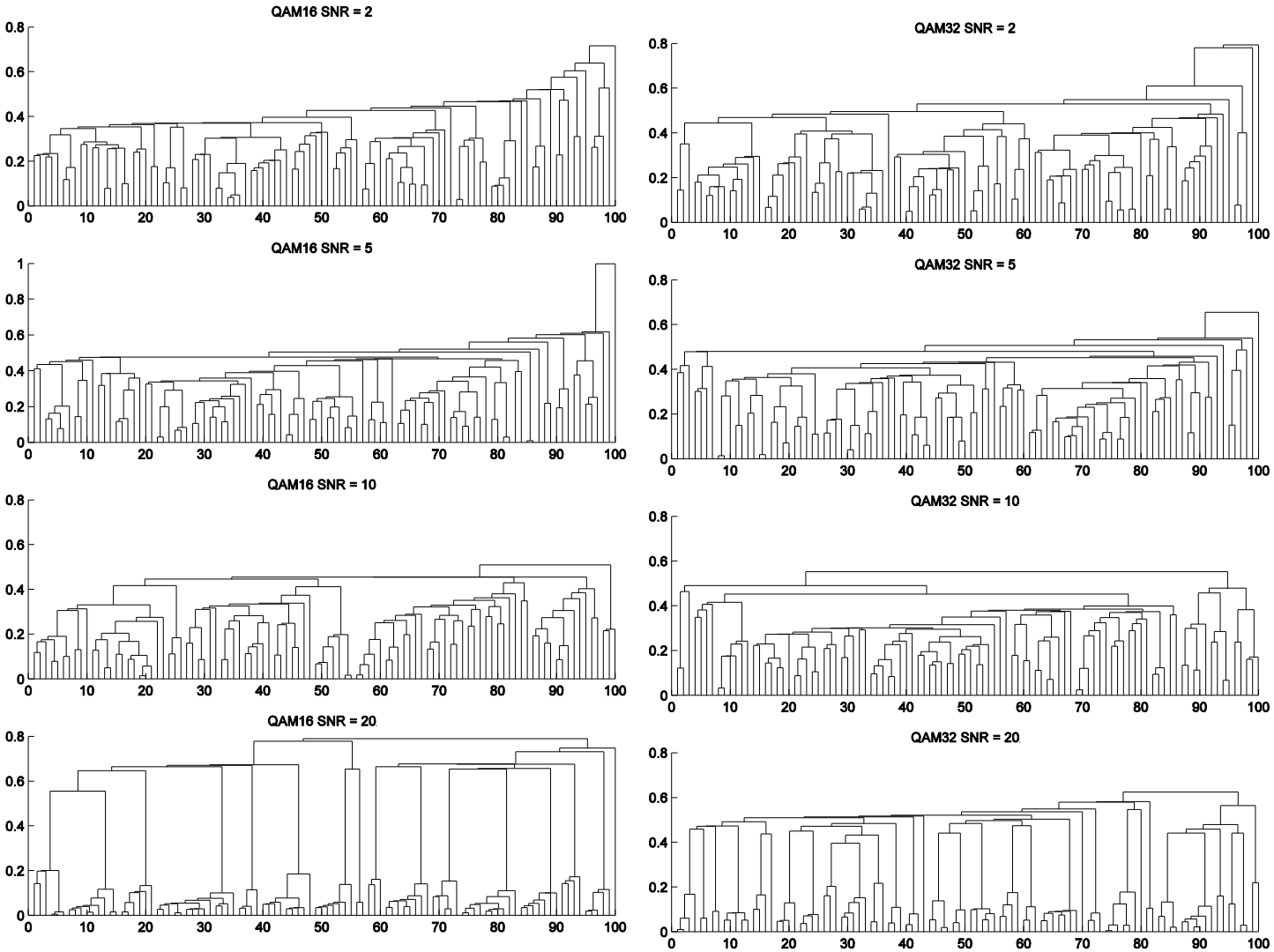
  where X_i and X_j have cardinals N_i and N_j and centroids $\boldsymbol{g}_i$ and $\boldsymbol{g}_j$.

Representation using a tree whose nodes indicate the different groups

Classification of Modulations



Classification of Modulations



Summary

Density-based clustering methods

Definitions

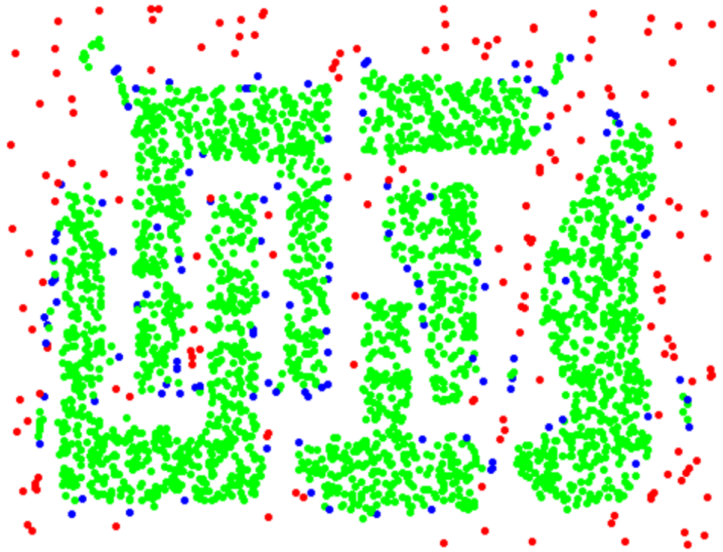
- Neighborhood of a point $p \in \boldsymbol{X}$

$$N_\epsilon(p) = \{q \in \boldsymbol{X} | d(p,q) \leq \epsilon\}$$

- Core point: a point p is a core point if it has more than MinPts neighbors in $N_\epsilon(p)$.
- Border point: a point p is a border point if it has less than MinPts neighbors in $N_\epsilon(p)$ and if it is in the neighborhood of a core point
- Noise point: a noise point is a point that is not a core point nor a border point

Core, border and noise points

Original Points



Point types: core, border and outliers

$\varepsilon = 10$, MinPts = 4

Thanks to Jing Gao from SUNY Buffalo university for her slides!!

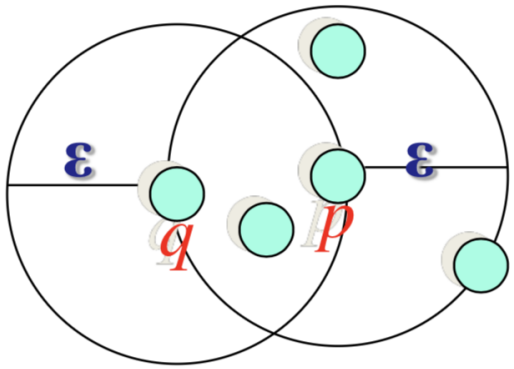
Relationships between different points

Directly density-reachable points

A point q is directly reachable from p for $(\epsilon, \mathrm{MinPts})$ if

- $q \in N_\epsilon(p)$
- p is a core point, i.e., $N_\epsilon(p)$ contains more than MinPts points.

Example



$\mathrm{MinPts} = 4$

- q is directly density-reachable from p
- p is not directly density-reachable from q (q is not a core point)

It is an asymmetric relationship!

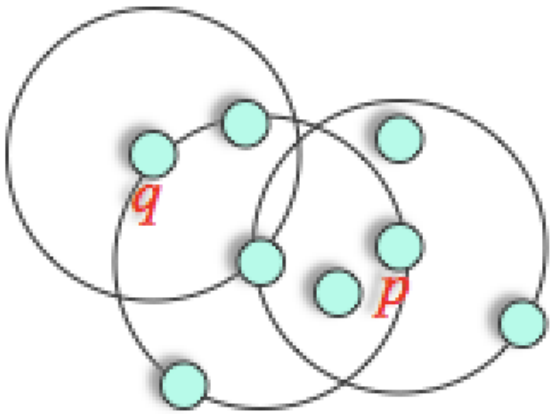
Relationships between different points

Density-reachable points

A point q is density-reachable from p for $(\epsilon, \mathrm{MinPts})$ if there is chain of points p_i such that

- $p_1 = p$ and $p_N = q$
- each point p_{i+1} is directly density-reachable from p_i.

Example



- q is directly density-reachable from p
- p is not directly density-reachable from q (q is not a core point)

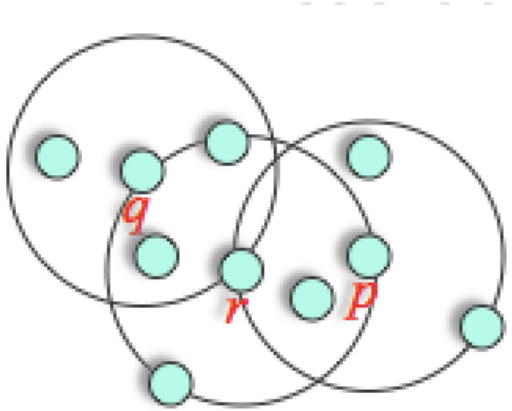
It is an asymmetric relationship!

Relationships between different points

Density-connectivity

Points p and q are density-connected for $(\epsilon, \mathrm{MinPts})$ if there is an object r such that both p and q are density reachable from r.

Example



It is a symmetric relationship!

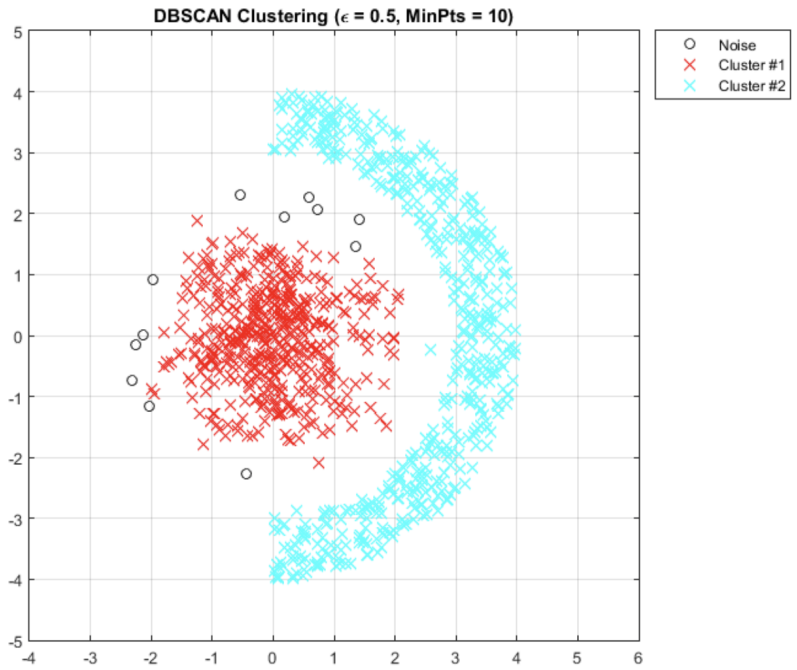
Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Cluster definition

A cluster C is defined as a maximal set of density-connected points

- Maximality: if $p \in C$ and if q is density-reachable from p, then $q \in C$.
- Connectivity: for all $(p, q) \in C$, p and q are density-connected.

Example



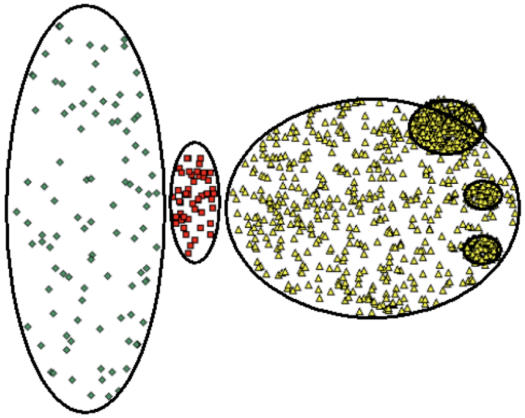
DBSCAN

A simple algorithm

- Select a point p
- Determine all density-reachable points from p for $(\epsilon, \mathrm{MinPts})$.
- If p is a core point, i.e., if the cardinal of $N_\epsilon(p)$ is larger than MinPts, a cluster is formed
- If p is a border point, DBSCAN visits the next point
- Continue the procedure until all points have been visited

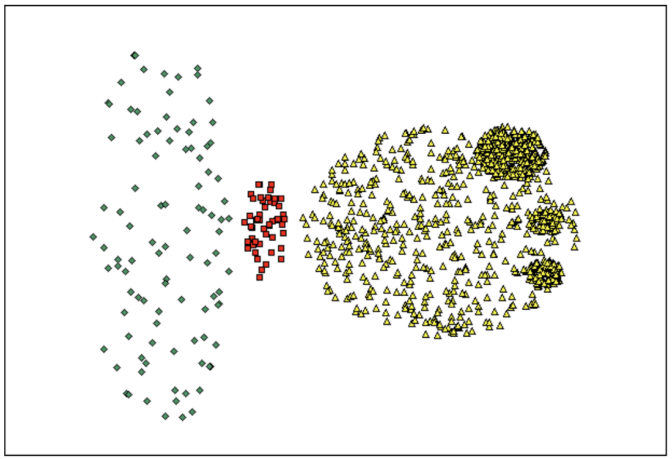
The DBSCAN algorithm generally provides a result independent of the order the points have been processed.

How can we choose ϵ and MinPts?

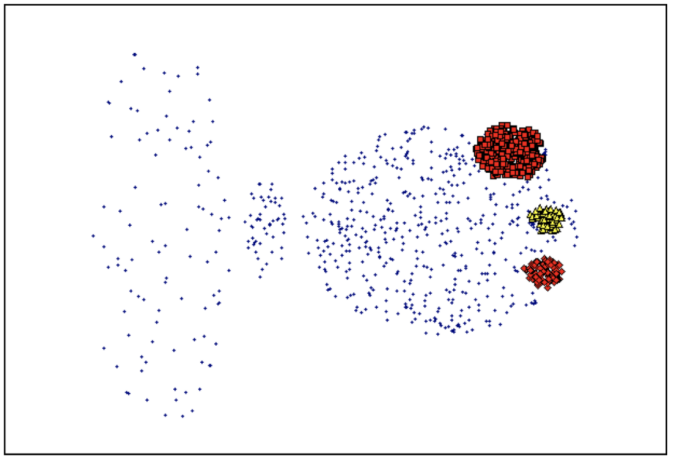


Original Points

- **Cannot handle varying densities**
- **sensitive to parameters—hard to determine the correct set of parameters**



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

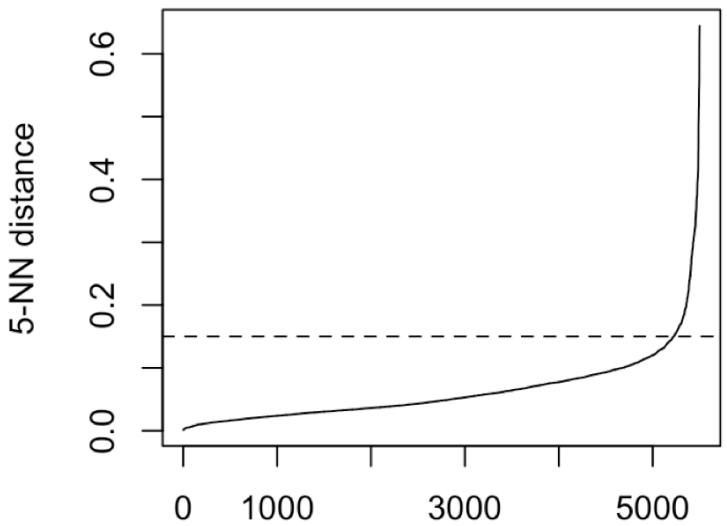
Thanks to Jing Gao from SUNY Buffalo university for her slides!!

How can we choose ϵ and MinPts?

Some ideas

- Points belonging to a cluster have a distance to their k-nearest neighbor (denoted as k-dist) smaller than noise points
- Compute all the k-dist values and sort them in increasing order
- Choose ϵ as the value of k-dist associated with a sharp change in the curve (this value does not vary significantly with the value of k)

Example



Conclusions

Pros

- Clusters of arbitrary shapes
- Robustness to noise

Cons

- Problems with clusters of different densities
- Parameter determination can be difficult

Some references (related to DBSCAN)

- M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'96), Portland, Oregon, Aug. 1996.
- J. Gao, Slides on Data Mining and Bioinformatics, University at Buffalo, https://www.cse.buffalo.edu/ jing/cse601/fa13/.

Summary

Gaussian mixture (unsupervised learning)

Idea: N unlabelled data vectors of $\mathbb{R}^p$ denoted as $\boldsymbol{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ emerge from K Gaussian components/classes denoted as $\omega_1, \ldots, \omega_K$.

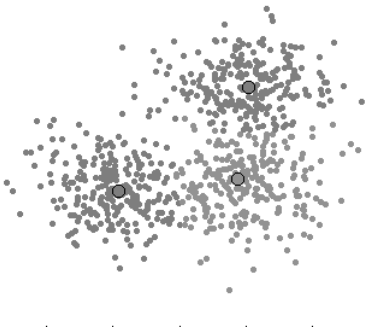
Gaussian mixture model (GMM)

- Definition

$$p(\mathbf{x}|\theta) = \sum_{j=1}^{K} \pi_j p(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{1}$$

- $\boldsymbol{\theta} = (\pi_1, \ldots, \pi_j, \ldots, \pi_K, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_j, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_j, \ldots, \boldsymbol{\Sigma}_K)$ contains all the parameters of the mixture model.

Example: $K = 3$ seems reasonable ($\boldsymbol{\theta}$ is unknown)



Gaussian mixture (supervised learning)

When the N data $\boldsymbol{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ are *complete* (i.e. labelled, assigned to classes), the parameter estimation problem is straightforward (each Gaussian can be estimated separately). Besides the data, we also have their labels: $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ where $y_i \in \{\omega_1, \ldots, \omega_K\}$.

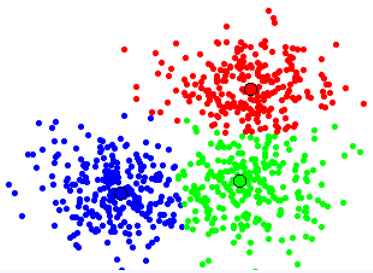
Gaussian mixture model (GMM)

- Definition

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{K} \pi_j p(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{2}$$

- Assignments: binary variable $\delta(j|i)$ assigns data $\mathbf{x}_i$ to the jth Gaussian (class ω_j) if $\delta(j|i) = 1$ ($\delta(j|i) = 0$ otherwise).

Example: $K = 3$ is now given (along with data assignments).



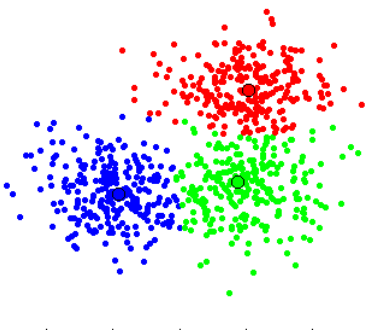
Gaussian mixture (supervised learning)

When the N data $\boldsymbol{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ are *complete* (i.e., labelled or assigned to classes), the parameter estimation problem is straightforward (each Gaussian density can be estimated separately).

Estimation of the Gaussian mixture model

- $\widehat{\pi}_j \leftarrow \frac{\widehat{n}_j}{n}$ with $\widehat{n}_j = \sum_{i=1}^{n} \delta(j|i)$
- $\widehat{\boldsymbol{\mu}}_j \leftarrow \frac{1}{\widehat{n}_j} \sum_{i=1}^{n} \delta(j|i) \boldsymbol{x_i}$
- $\widehat{\boldsymbol{\Sigma}}_j \leftarrow \frac{1}{\widehat{n}_j} \sum_{i=1}^{n} \delta(j|i)(\mathbf{x_i} - \widehat{\boldsymbol{\mu}}_j)(\mathbf{x_i} - \widehat{\boldsymbol{\mu}}_j)^T$
  where $\delta(j|i) = 1$ assigns data $\mathbf{x}_i$ to the jth Gaussian density

Example: $K = 3$ is now given (along with data assignments).



Gaussian mixture (back to unsupervised case)

Without assignment (unsupervised learning), we start from an initial setting of the parameters $\boldsymbol{\theta}$: $\boldsymbol{\theta}^0 = (\pi_1^0, \ldots, \pi_K^0, \boldsymbol{\mu}_1^0, \boldsymbol{\Sigma}_1^0, \ldots, \boldsymbol{\mu}_K^0, \boldsymbol{\Sigma}_K^0)$ and compute

$$P(y_i = \omega_j | \boldsymbol{x_i}, \boldsymbol{\theta}^0) = \frac{p(\boldsymbol{x_i} | y_i = \omega_j, \boldsymbol{\theta}^0) P(\omega_j)}{\sum_j p(\boldsymbol{x_i} | y_i = \omega_j, \boldsymbol{\theta}^0) P(\omega_j)} \tag{3}$$

$$P(y_i = \omega_j | \boldsymbol{x_i}, \boldsymbol{\theta}^0) = \frac{\pi_j^0 p(\boldsymbol{x_i} | \boldsymbol{\mu}_j^0, \boldsymbol{\Sigma}_j^0)}{\sum_j \pi_j^0 p(\boldsymbol{x_i} | \boldsymbol{\mu}_j^0, \boldsymbol{\Sigma}_j^0)} \tag{4}$$

Soft assignment

- $\widehat{\delta}(j|i) = P(y_i = \omega_j | \boldsymbol{x_i}, \boldsymbol{\theta}^0)$
- $\sum_{j=1}^K \widehat{\delta}(j|i) = 1.$

Expectation-Maximization (EM-Algorithm for GMM estimation)

Without assignment (unsupervised case), we start from $\boldsymbol{\theta}^0$ and iterate soft-assignments that "complete the incomplete data" (Expectation-step) and parameter refinement (Maximisation-step). We can show that the new setting of the parameters $\boldsymbol{\theta}^{(k+1)}$ increases the log-likelihood of the "completed" data.

EM-algorithm:

1. **Initialization** Specify $\boldsymbol{\theta}^{(k=0)}$.
2. **Repeat**

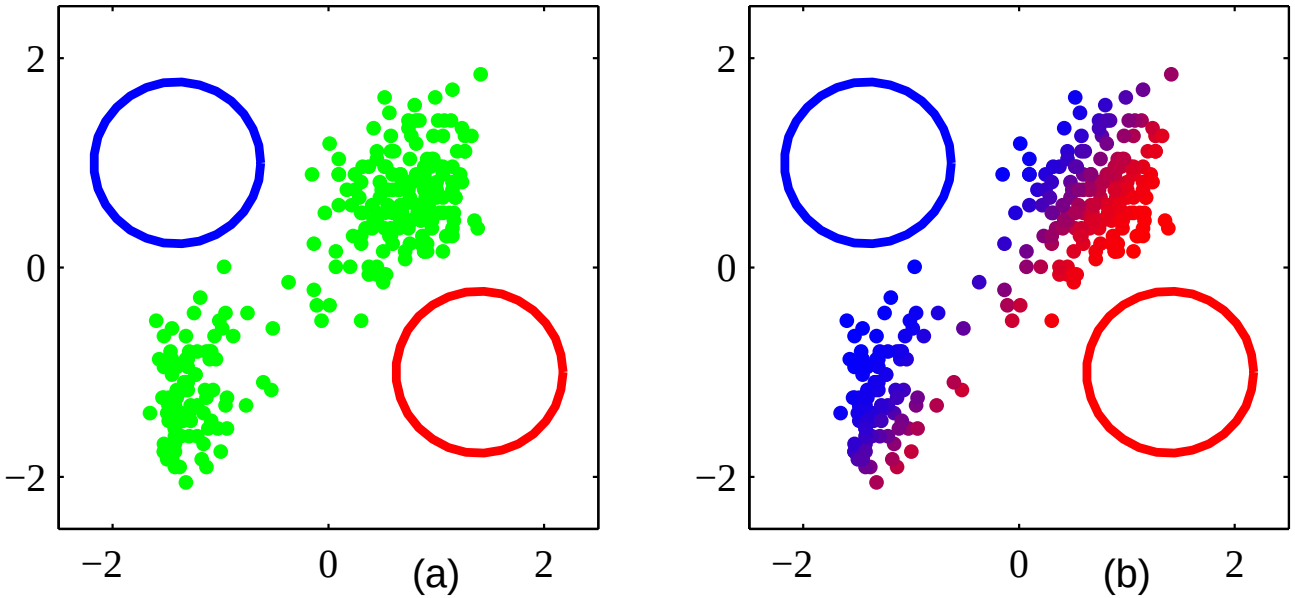
(E-step) soft-assignments of $\mathbf{x_i}$ $\forall i$

$$\widehat{\delta}(j|i) \leftarrow P(y_i = \omega_j | \mathbf{x_i}, \boldsymbol{\theta}^{(k)}) \tag{5}$$

(M-step) Refine $\theta^{(k+1)}$:

- $\widehat{\pi}_j \leftarrow \frac{\widehat{n}_j}{n}$ with $\widehat{n}_j = \sum_{i=1}^{n} \widehat{\delta}(j|i)$
- $\widehat{\boldsymbol{\mu}}_j \leftarrow \frac{1}{\widehat{n}_j} \sum_{i=1}^{n} \widehat{\delta}(j|i) \mathbf{x_i}$
- $\widehat{\boldsymbol{\Sigma}}_j \leftarrow \frac{1}{\widehat{n}_j} \sum_{i=1}^{n} \widehat{\delta}(j|i)(\mathbf{x_i} - \widehat{\boldsymbol{\mu}}_j)(\mathbf{x_i} - \widehat{\boldsymbol{\mu}}_j)^T$
- $k \leftarrow k + 1$

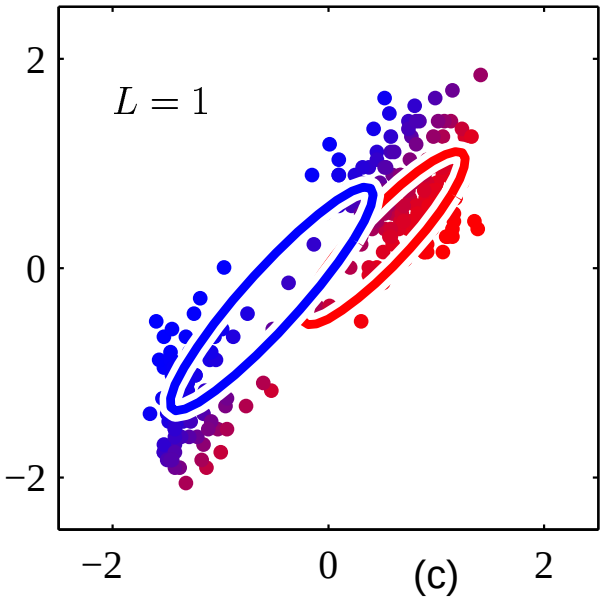
EM-Algorithm for GMM estimation (demo from book by Bishop*)



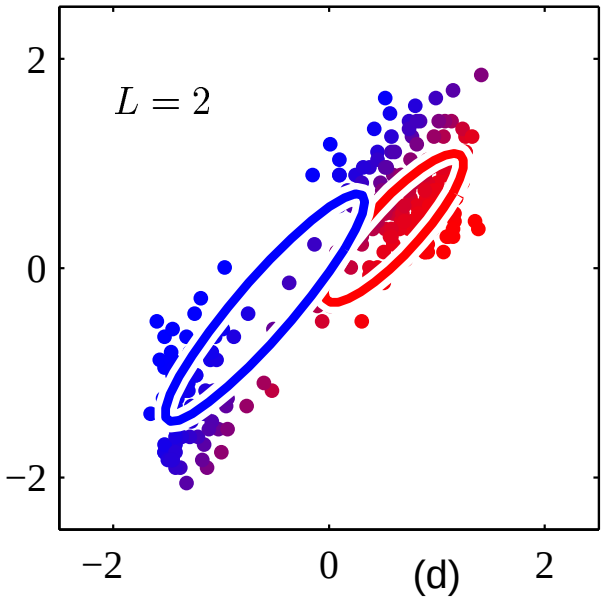
(Initialisation) (E)

* Christopher Bishop, *Pattern Recognition and Machine Learning*. New-York: Springer Verlag, 2006.

EM-Algorithm for GMM estimation (demo 2/3)

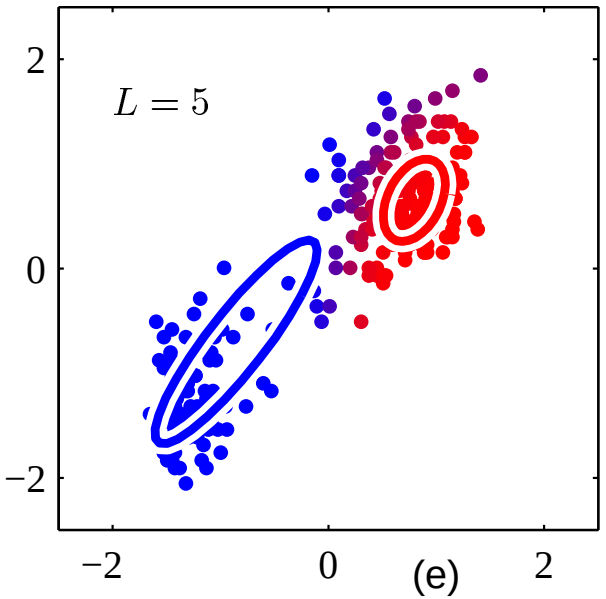


(M)

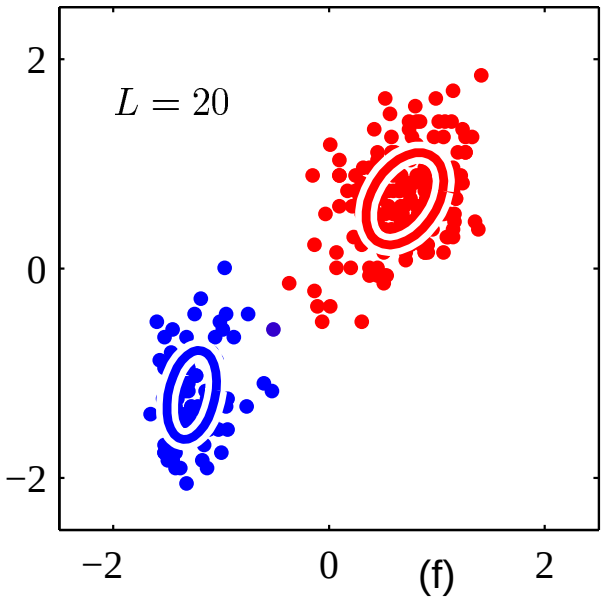


(Iteration $k = 2$)

EM-Algorithm for GMM estimation (demo 3/3)



(Iteration $k = 5$)



(Iteration $k = 20$)

Convergence

- The EM algorithm monotonically increases the log-likelihood of the data.
- We have $l(\boldsymbol{\theta}^0) < l(\boldsymbol{\theta}^1) < ... < l(\boldsymbol{\theta}^k)$ with $l(\boldsymbol{\theta}^k) = \sum_{i=1}^{n} \ln\, p(\boldsymbol{x_i}|\boldsymbol{\theta}^k)$

Mélange de lois AGGD

Vraisemblance du mélange

$$p(x|\theta) = \sum_{j=1}^{M} p_j p(x|\theta_j)$$

avec

$$p(x|\theta_j) = \begin{cases} \frac{\delta_j^{1/\lambda_j}}{\gamma_j^{1/\lambda_j}\Gamma(1+1/\lambda_j)} \exp\left(-\frac{\delta_j}{\gamma_j}\left(\frac{\mu_j - x}{\alpha_j}\right)^{\lambda_j}\right) & \text{si } x < \mu_j \\ \frac{\delta_j^{1/\lambda_j}}{\gamma_j^{1/\lambda_j}\Gamma(1+1/\lambda_j)} \exp\left(-\frac{\delta_j}{\gamma_j}\left(\frac{x - \mu_j}{1-\alpha_j}\right)^{\lambda_j}\right) & \text{si } x \geq \mu_j \end{cases}$$

- α_j : paramètre d'asymétrie
- μ_j : paramètre de décalage
- γ_j : paramètre d'échelle
- λ_j : paramètre de forme
- $\delta_j = \frac{2\alpha_j^{\lambda_j}(1-\alpha_j)^{\lambda_j}}{\alpha_j^{\lambda_j} + (1-\alpha_j)^{\lambda_j}}$
- M : nombre de composantes

Algorithme EM + Binarisation

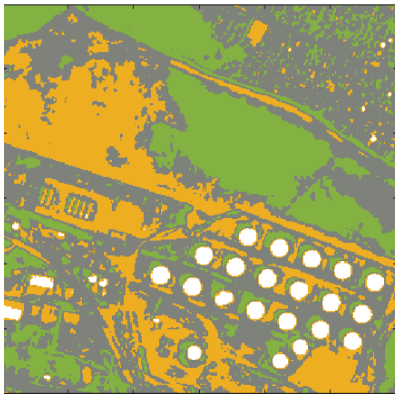
Augmentation des données avec des labels : $\mathbf{x} \rightarrow (\mathbf{x}, \mathbf{Z})$ où $\mathbf{x} = (x^1, \ldots, x^N)$ et $\mathbf{Z} = (\mathbf{z}^1, \ldots, \mathbf{z}^N)$ avec $\mathbf{z}^i$ de dimension M (correspondant aux M classes)
$\Rightarrow$ Log-vraisemblance augmentée

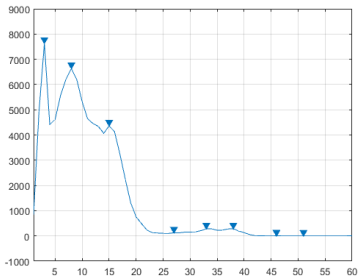
$$L(\mathbf{x}, \mathbf{Z}; \theta) = \sum_{i=1}^{N} \sum_{m=1}^{M} z_m^i \log \left(p_m p(x^i; \theta_m) \right)$$

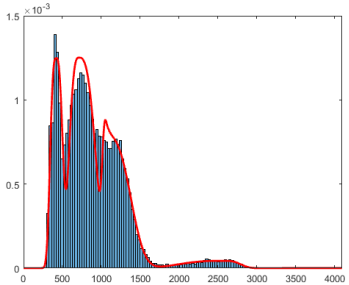
BUT : estimer $\mathbf{p} = (p_1, \ldots, p_M)$ et θ

- **Étape E** *Expectation* : calcul d'une espérance conditionnelle de la log-vraisemblance (fonction Q) afin de mettre à jour les labels $\mathbf{Z}$,
- **Étape B** *Binarization* : seuillage de la matrice des labels $\mathbf{Z}$ en utilisant la loi multinomiale,
- **Étape M** *Maximization* : mise à jour des paramètres par maximisation de la vraisemblance de chaque classe (ML).

Algorithme 2







19 / 78

Summary

Support vector machines (SVMs)

Learning set

$$\mathcal{B} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$$

where $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ are n vectors of $\mathbb{R}^p$ and $y_1, ..., y_n$ are binary variables

$$y_i = 1 \text{ if } \boldsymbol{x}_i \in \omega_1, \quad y_i = -1 \text{ if } \boldsymbol{x}_i \in \omega_2$$

Hyperplane definition

$$g_{\boldsymbol{w},b}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} - b = 0$$

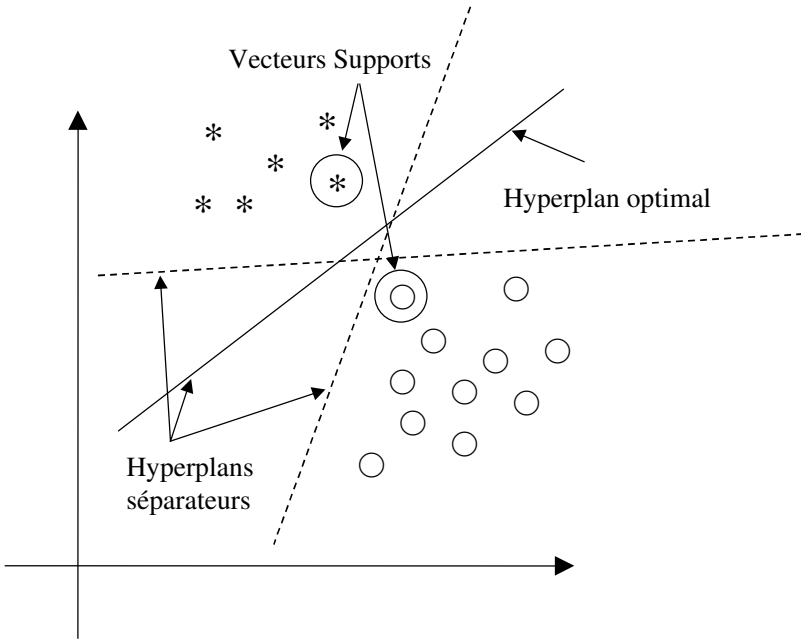
with

$$g_{\boldsymbol{w},b}(\boldsymbol{x}_i) > 0 \text{ if } \boldsymbol{x}_i \in \omega_1, \quad g_{\boldsymbol{w},b}(\boldsymbol{x}_i) < 0 \text{ if } \boldsymbol{x}_i \in \omega_2$$

Classification rule

$$f(\boldsymbol{x}) = \text{sign}\left[g_{\boldsymbol{w},b}(\boldsymbol{x})\right] \tag{6}$$

Illustration



Problem formulation

Margin of $\boldsymbol{x}_i$ with label y_i (algebraic distance to the hyperplane)

$$\gamma_i\left(\widetilde{\boldsymbol{w}}\right) = \frac{y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i - b\right)}{\|\boldsymbol{w}\|}$$

with $\widetilde{\boldsymbol{w}} = (\boldsymbol{w}, b)$ ($\boldsymbol{x}_i$ is correctly classified by (6) if $\gamma_i\left(\widetilde{\boldsymbol{w}}\right) > 0$)

Margin of the learning set

$$\gamma_{\mathcal{B}}\left(\widetilde{\boldsymbol{w}}\right) = \min_{i\in\{1,\ldots,n\}} \frac{y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i - b\right)}{\|\boldsymbol{w}\|}$$

Since $\gamma_{\mathcal{B}}\left(a\widetilde{\boldsymbol{w}}\right) = \gamma_{\mathcal{B}}\left(\widetilde{\boldsymbol{w}}\right)$, $\forall a > 0$, $\widetilde{\boldsymbol{w}}$ is not unique!

Constraints for the hyperplane: one forces the training samples that are the closest to the hyperplane to satisfy

$$y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i - b\right) = 1 \Rightarrow \min_{i\in\{1,\ldots,n\}} y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i - b\right) = 1$$

The vectors $\boldsymbol{x}_i$ satisfying $y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i - b\right) = 1$ are called support vectors.

Problem formulation

Canonical hyperplane

$$y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i - b \right) \geq 1, \quad \forall i = 1, ..., n$$

Classifier margin for a canonical hyperplane

$$\gamma_{\mathcal{B}} \left(\widetilde{\boldsymbol{w}} \right) = \frac{1}{\|\boldsymbol{w}\|}$$

New formulation

Minimize $\frac{1}{2} \|w\|^2$ **with the constraints** $y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i - b \right) \geq 1, \forall i$

Simple problem since the cost function to optimize is quadratic and the constraints are linear!

Optimization

Lagrangian

$$L(\widetilde{\boldsymbol{w}}, \alpha) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \left[y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i - b \right) - 1 \right]$$

Set to zero the partial derivatives of L with respect to b and w

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } \boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$$

Kühn and Tucker multipliers

For a convex optimization problem (convex function $f(x)$ to optimize and convex constraints $G_i(\boldsymbol{x}) \leq 0$), an optimality condition is the existence of parameters $\alpha_i \geq 0$ such that the Lagrangian derivative is zero, i.e.,

$$L'(\boldsymbol{x}) = f'(\boldsymbol{x}) + \sum_{i=1}^{n} \alpha_i G_i'(\boldsymbol{x}) = 0$$

with $\alpha_i = 0$ if $G_i(\boldsymbol{x}) < 0$ (i.e., $\alpha_i G_i(\boldsymbol{x}) = 0$).

Dual problem

Solve $L'(\boldsymbol{x}) = 0$

$$\boldsymbol{w} = \sum_{\text{Support vectors}} \alpha_i y_i \boldsymbol{x}_i = \boldsymbol{x}^T \boldsymbol{Y} \alpha \tag{7}$$

with $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)^T$, $\boldsymbol{x} = (x_1, ..., x_n)^T$, $\boldsymbol{Y} = \text{diag}\,(y_1, ..., y_n)$ and

$$\begin{cases} \alpha_i = 0 \text{ if the constraint is a strict inequality} \\ \alpha_i > 0 \text{ if the constraint is an equality} \end{cases}$$

After replacing the expression of w in the Lagrangian, we obtain

$$U(\alpha) = -\frac{1}{2} \alpha^T \boldsymbol{Y} \left(\boldsymbol{x} \boldsymbol{x}^T \right) \boldsymbol{Y} \alpha + \sum_{i=1}^{n} \alpha_i$$

that has to be maximized in the domain defined by $\alpha_i \geq 0, \forall i$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$.

Remarks

Simple optimization problem

Quadratic (hence convex) function to optimize and linear constraints

Norm of the solution $\boldsymbol{w}_0$

$$\|\boldsymbol{w}_0\|^2 = \sum_{\text{support vectors}} \alpha_i^0 \left(1 + y_i b\right) = \sum_{\text{support vectors}} \alpha_i^0$$

because of the constraint $\sum_{i=1}^n \alpha_i y_i = 0$.

Classifier margin

$$\boxed{\gamma = \frac{1}{\|\boldsymbol{w}_0\|} = \left(\sum \alpha_i^0\right)^{-1/2}}$$

Classification rule for a vector $\boldsymbol{x}$

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{z_i \text{ support vectors}} \alpha_i^0 y_i \boldsymbol{x}_i^T \boldsymbol{x} - b_0\right), \quad b_0 = \frac{1}{2}\left(\boldsymbol{w}_0^T \boldsymbol{z}^+ + \boldsymbol{w}_0^T \boldsymbol{z}^-\right),$$

where $\boldsymbol{z}^+$ (resp. $\boldsymbol{z}^-$) is a support vector belonging to the 1st (resp. 2nd) class.

Extensions

The "soft-margin" SVM classifier

$$\text{minimize } \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{with the constraints } y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i - b\right) \geq 1 - \xi_i, \forall i$$

The ν-SVM classifier

$$\text{Minimize } \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\xi_i - \nu\gamma$$

$$\text{with the constraints } y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i - b\right) \geq \gamma - \xi_i, \forall i$$

Non-linear preprocessing

Search a non-linear transformation $\phi : \mathbb{R}^p \to \mathbb{R}^q$ ensuring linear separability.

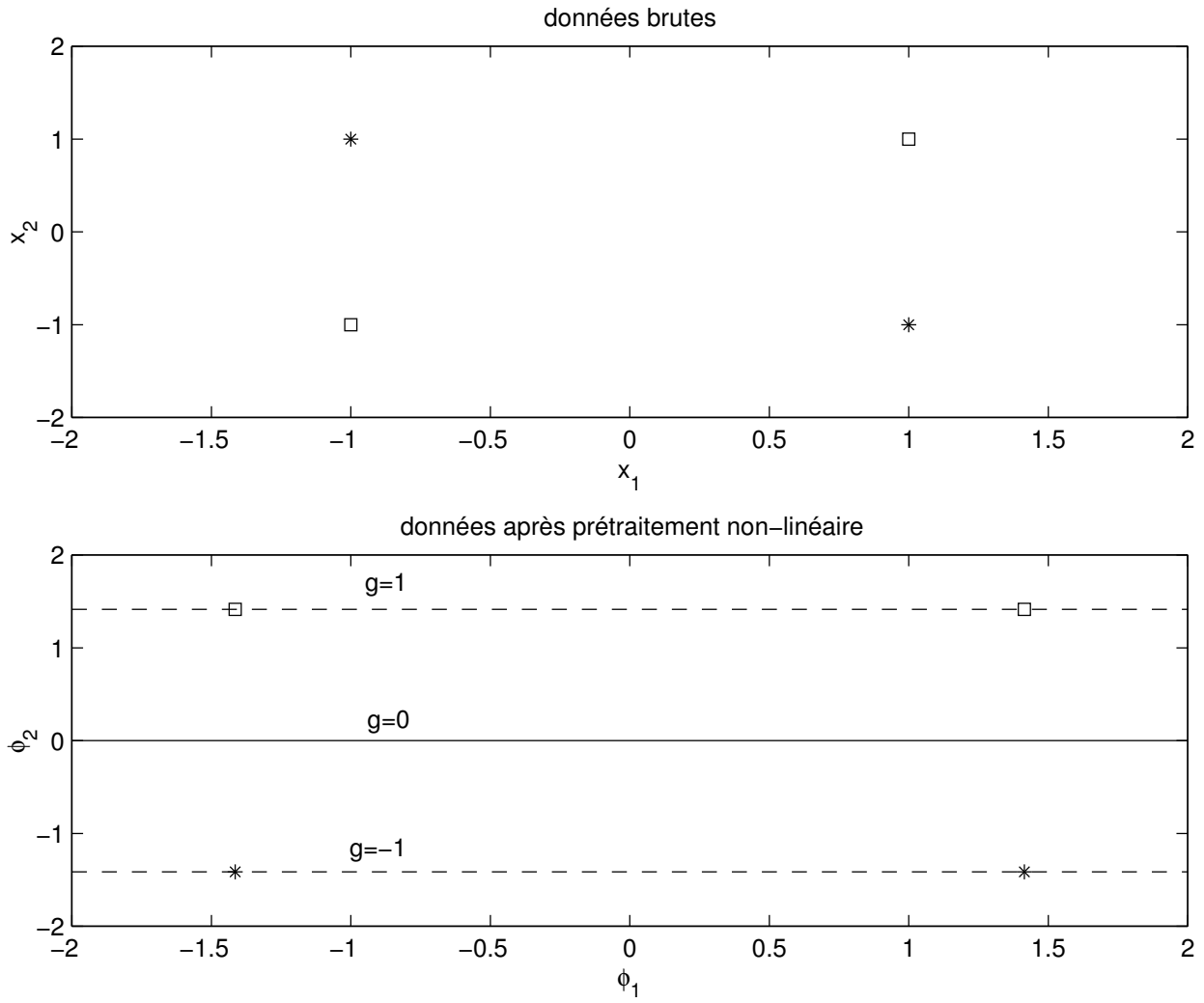
Classical example

The classes $\chi_1 = \{(1,1), (-1,-1)\}$ and $\chi_2 = \{(1,-1), (-1,1)\}$ are not linearly separable. Consider the application ϕ defined by

$$\phi : \begin{array}{l} \mathbb{R}^2 \to \mathbb{R}^6 \\ (x_1, x_2)^T \mapsto \left(\sqrt{2}x_1, \sqrt{2}x_1x_2, 1, \sqrt{2}x_2, x_1^2, x_2^2\right)^T \end{array}$$

The data are separable in the plane (ϕ_1, ϕ_2)

Example of separability



Non-linear SVM classifier

Decision rule after preprocessing

$$\begin{aligned} f(\tilde{\boldsymbol{x}}) &= \text{sign}\left(g_{\boldsymbol{w},b}(\tilde{\boldsymbol{x}})\right) = \text{sign}\left(\boldsymbol{w}^T\phi\left(\boldsymbol{x}\right) - b\right) \\ &= \text{sign}\left(\sum_{\text{support vectors}} \alpha_i^0 y_i \phi\left(\boldsymbol{x}_i\right)^T \phi\left(\boldsymbol{x}\right) - b_0\right) \end{aligned}$$

Cost function to optimize

$$\boxed{U\left(\boldsymbol{\alpha}\right) = -\tfrac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Y}\boldsymbol{G}\boldsymbol{Y}^T\alpha + \sum_{i=1}^n \alpha_i}$$

where $\boldsymbol{G}$ is the Gram matrix defined by $G_{ij} = \phi\left(\boldsymbol{x}_i\right)^T\phi\left(\boldsymbol{x}_j\right)$. The cost function $U\left(\alpha\right)$ has to be maximized under the constraints

$$\boxed{0 \leq \alpha_i \leq \tfrac{1}{n},\ \forall i} \quad \text{and} \quad \boxed{\sum_{i=1}^n \alpha_i \geq \nu}$$

Conclusions: the cost function and the decision rule only depend on the inner products $\phi\left(\boldsymbol{x}_i\right)^T\phi\left(\boldsymbol{x}_j\right)$ and $\phi\left(\boldsymbol{x}_i\right)^T\phi\left(\boldsymbol{x}\right)$.

Kernels

A kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$ allows inner products $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ and $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x})$ to be computed with a reduced computational cost.

Example: $\boldsymbol{x} = (x_1, x_2)^T$ and $\phi(\boldsymbol{x}) = \left(x_1^2, x_2^2, \sqrt{2}x_1 x_2\right)$

$$\begin{aligned} \phi(\boldsymbol{x})^T \phi(\boldsymbol{y}) &= \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{pmatrix} . \begin{pmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}y_1 y_2 \end{pmatrix} \\ &= x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2 \\ &= \left(\boldsymbol{x}^T \boldsymbol{y}\right)^2 \end{aligned}$$

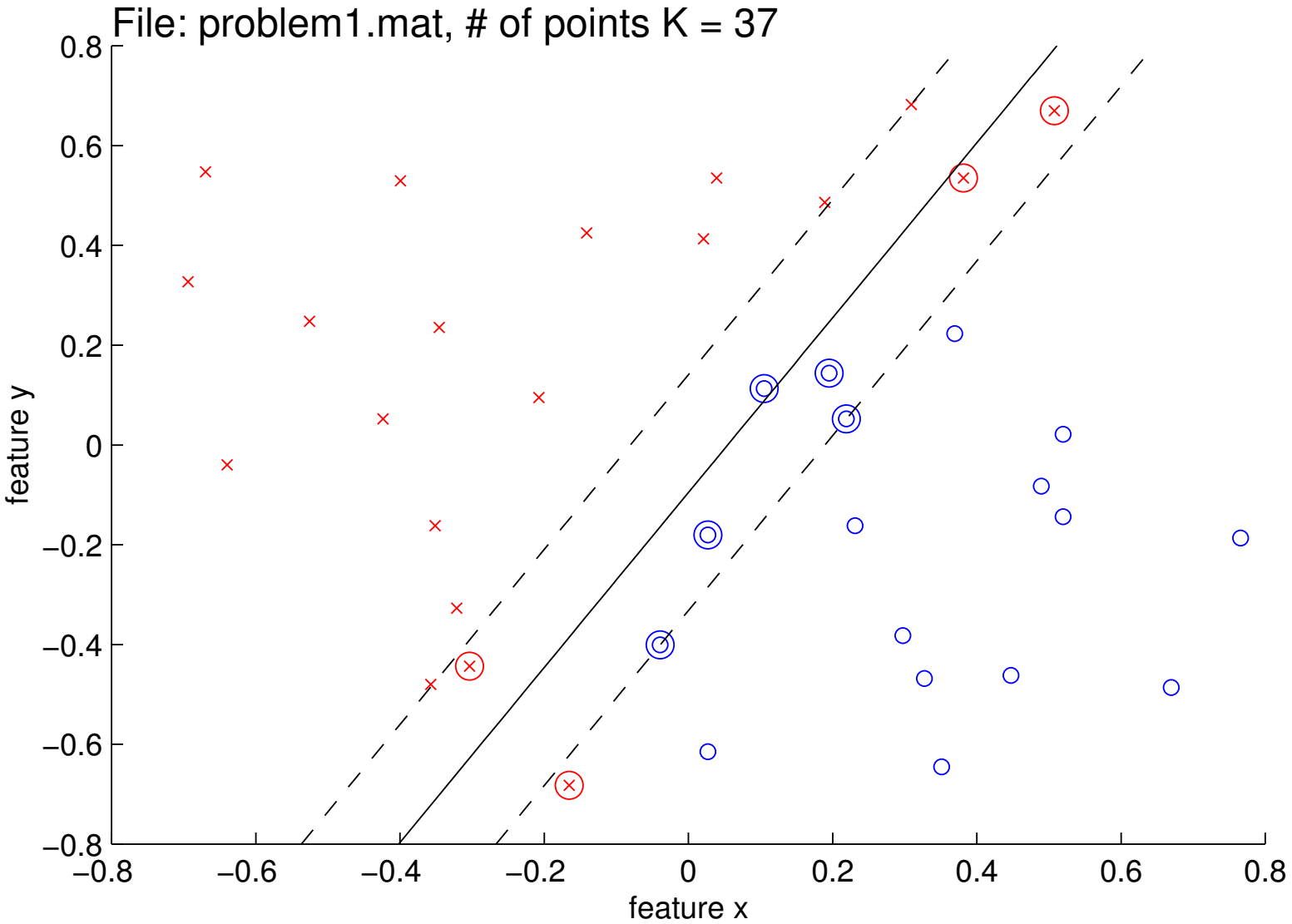
Mercer kernels can be expressed as inner products

$$k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$$

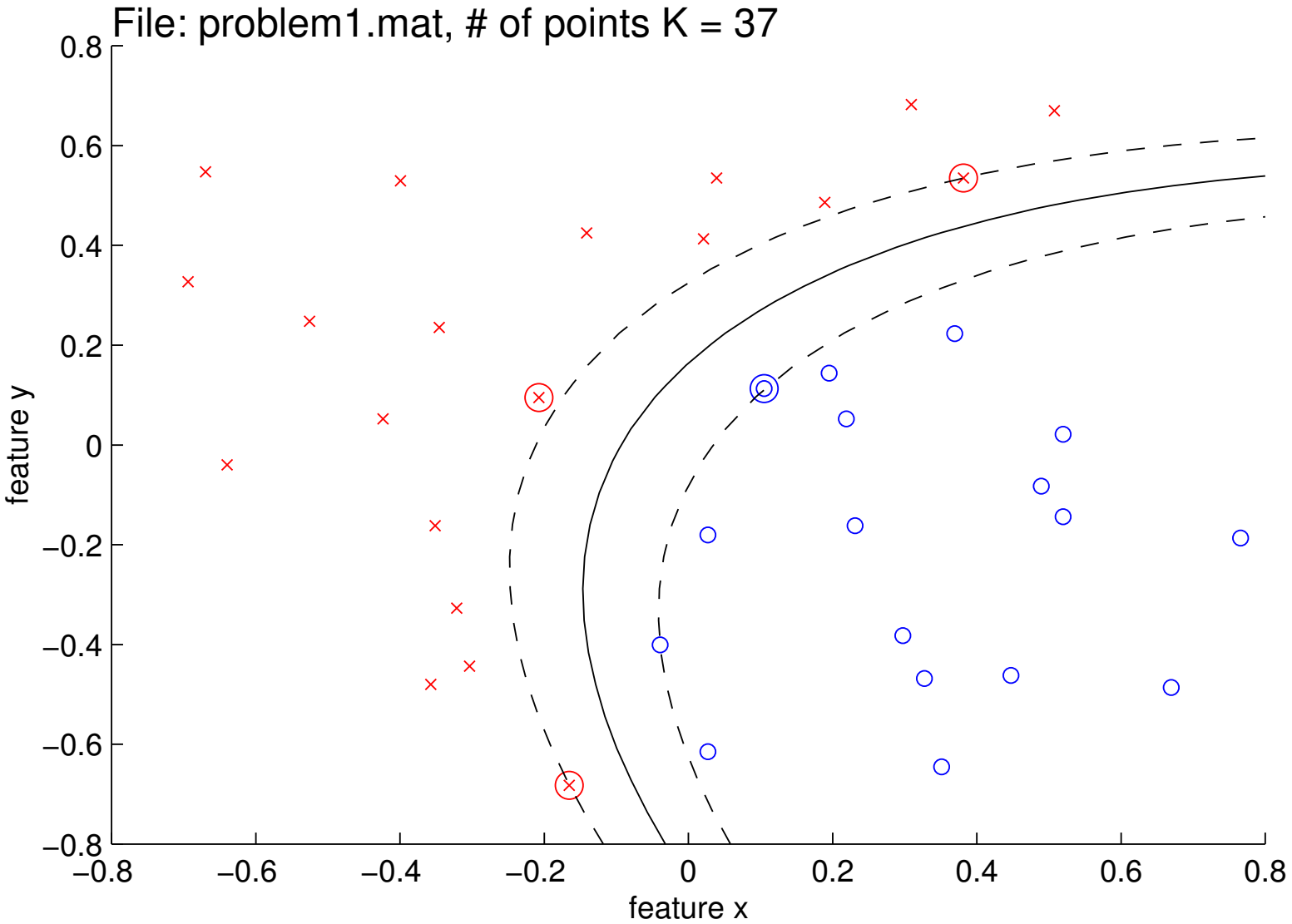
Classical kernels

Kernel	Expression
Polynomial (of degree p)	$k(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y} \rangle)^q$ $q \in \mathbb{N}^+$
Full polynomial	$k(\boldsymbol{x}, \boldsymbol{y}) = (\langle \boldsymbol{x}, \boldsymbol{y} \rangle + c)^q$ $c \in \mathbb{R}^+$, $q \in \mathbb{N}^+$
RBF	$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$ $\sigma \in \mathbb{R}^+$
Mahalanobis	$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left[-(\boldsymbol{x}-\boldsymbol{y})^T \Sigma (\boldsymbol{x}-\boldsymbol{y})\right]$ $\Sigma = \text{diag}\left(\frac{1}{\sigma_1^2}, ..., \frac{1}{\sigma_p^2}\right)$, $\sigma_i \in \mathbb{R}^+$

Example of linear SVMs



Example of non-linear SVMs



Equalization of non-linear communication channels

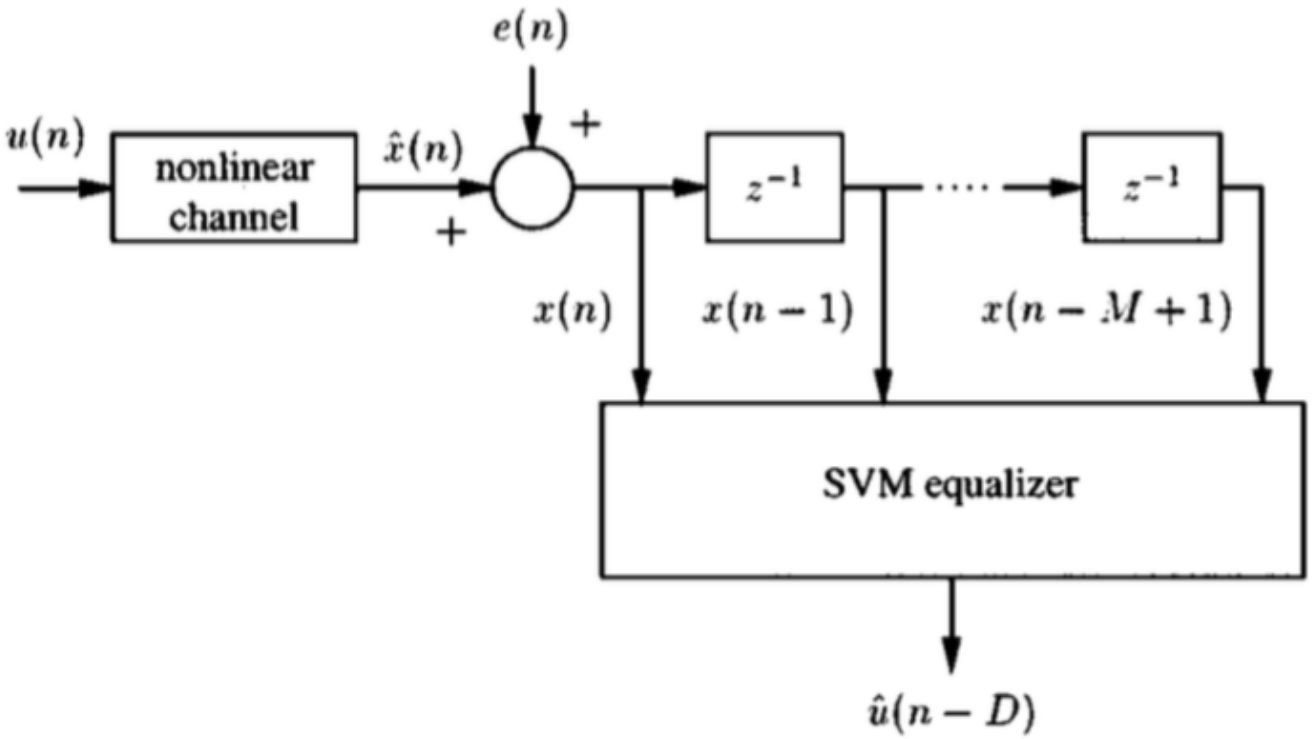


Fig. 2. Model of the nonlinear transmission system originating from Chen *et al.* [12].

Equalization of non-linear communication channels

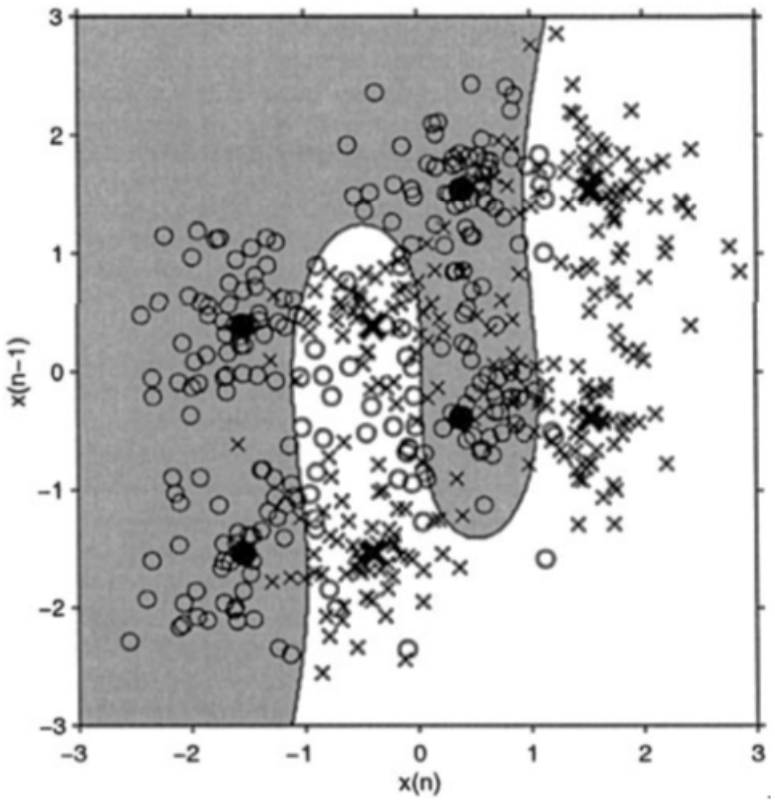


Fig. 3. Example of typical classification regions of an SVM and associated training points with channel $\hat{x}(n) = \tilde{x}(n) - 0.9\ \tilde{x}^3(n)$, $\tilde{x}(n) = u(n) + 0.5u(n-1)$, and Gaussian white noise of power $\sigma_e^2 = 0.2$ and equalizer dimension $M = 2$, polynomial kernel order $d = 3$, constraint $C = 5$, and lag $D = 0$.

Equalization of non-linear communication channels

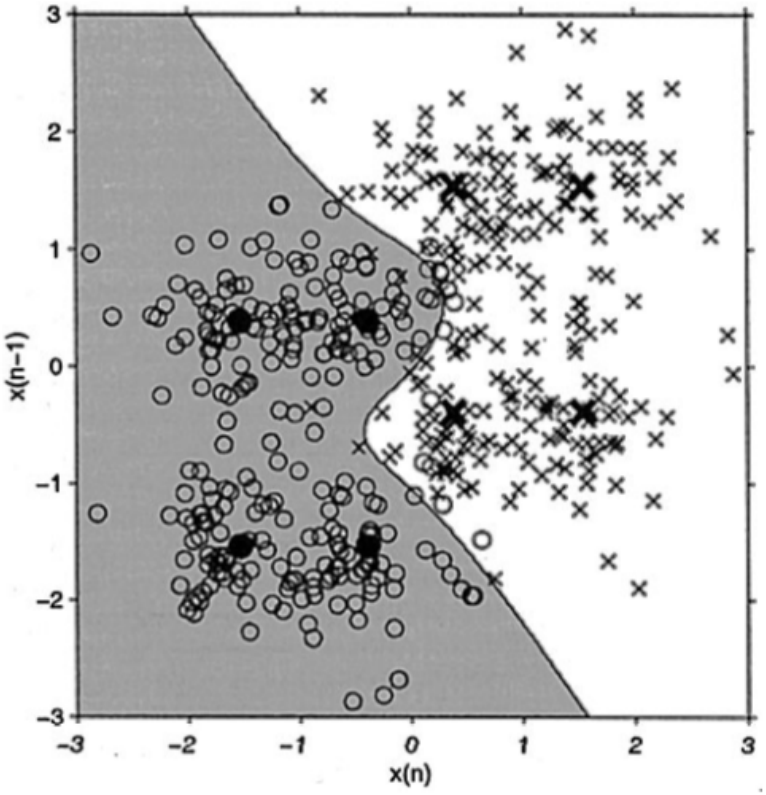


Fig. 4. Example of typical classification regions of an SVM and associated training points with the channel described in Fig. 3, except the equalizer lag is now $D = 1$.

Equalization of non-linear communication channels

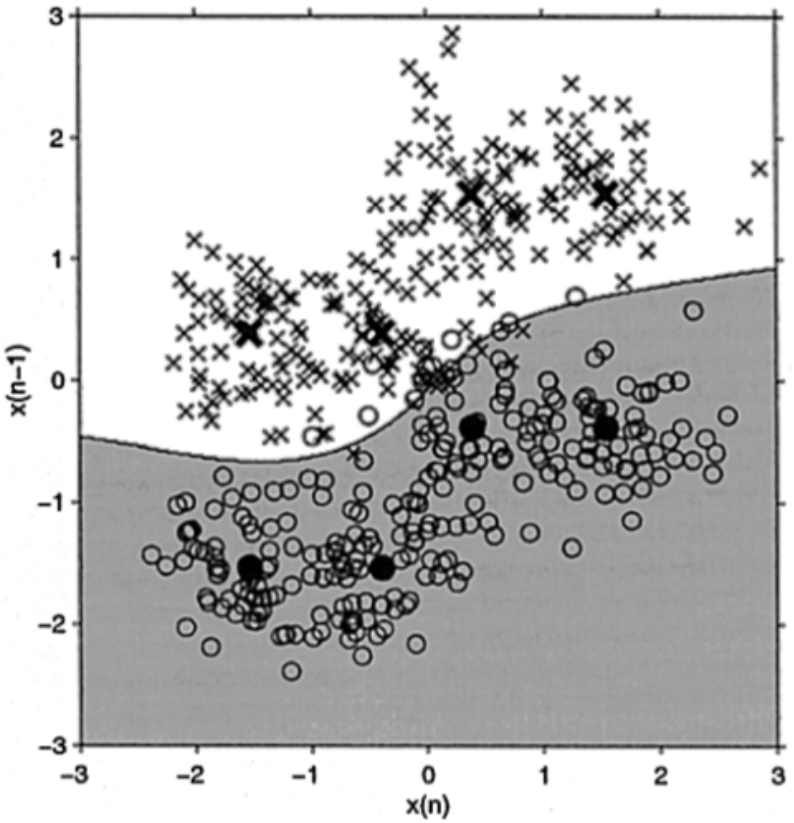
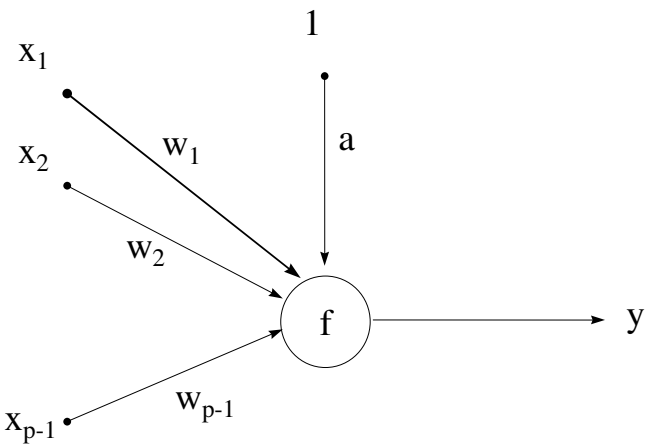
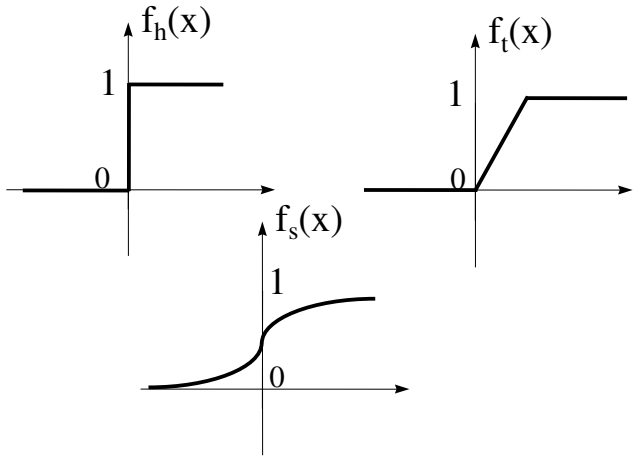


Fig. 5. Example of typical classification regions of an SVM and associated training points with the channel described in Fig. 3, except the equalizer lag is now $D = 2$.

Summary

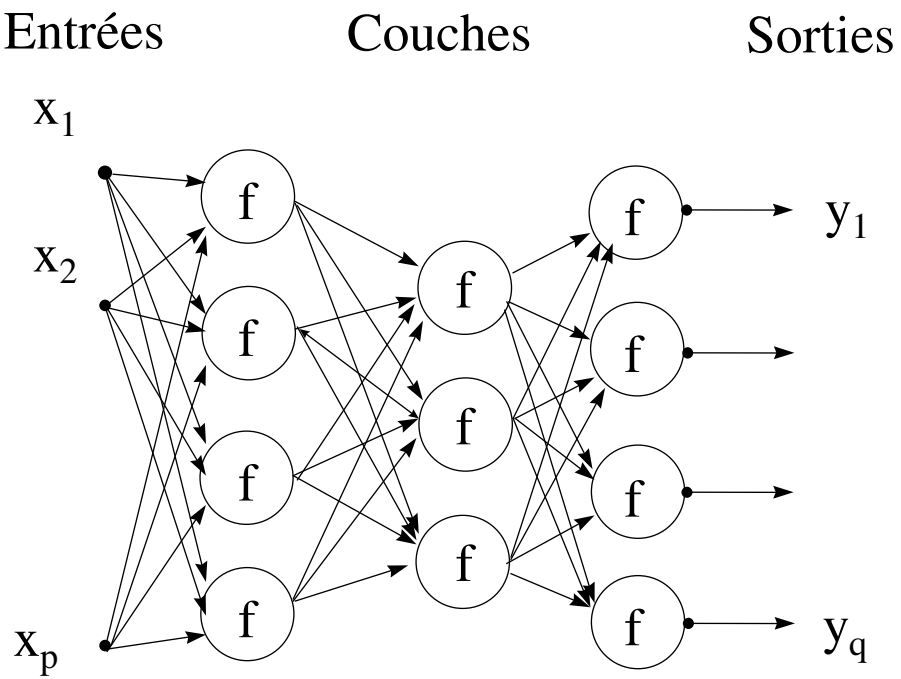
Neural networks





- inputs: $x_1, ..., x_{p-1}$
- weights: $w_1, ..., w_{p-1}$
- bias: a (or offset)
- output : $y = f\left(\sum_{i=1}^{p-1} w_i x_i - a\right)$
- Non-linearity
  - Heaviside: $f_h(x)$
  - Logical threshold: $f_t(x)$
  - Sigmoid: $f_s(x)$
- Equivalent definition: $x_p = -1, w_p = a$ et $y = f\left(\sum_{i=1}^{p} w_i x_i\right)$

Multi-layer perceptron



- Definition of the network structure
  - ☞ Number of outputs?
  - ☞ Number of layers?
  - ☞ Number of nodes per layer?
- Learning rules?

Backpropagation

- Weight Initialization: $w_{ij}^{(l)}(0)$ with small values
- at iteration n
  - ☞ Compute the desired output

$$d(n) = 1 \text{ if } x(n) \in \omega_1, \quad d(n) = 0 \text{ if } x(n) \in \omega_2$$

  - ☞ Compute the network output

$$y_j^{(l)}(n) = f_s\left(\sum_{i=1}^{N_{l-1}} w_{ij}^{(l)}(n)\, y_i^{(l-1)}(n)\right)$$

where $y_j^{(l)}$ is the jth output of the lth layer (with $y^{(L)}(n) = y(n)$ et $y_j^{(0)}(n) = x_j(n)$), N_l is the number of nodes of the lth layer and L is the number of layers.

  - ☞ Weight update

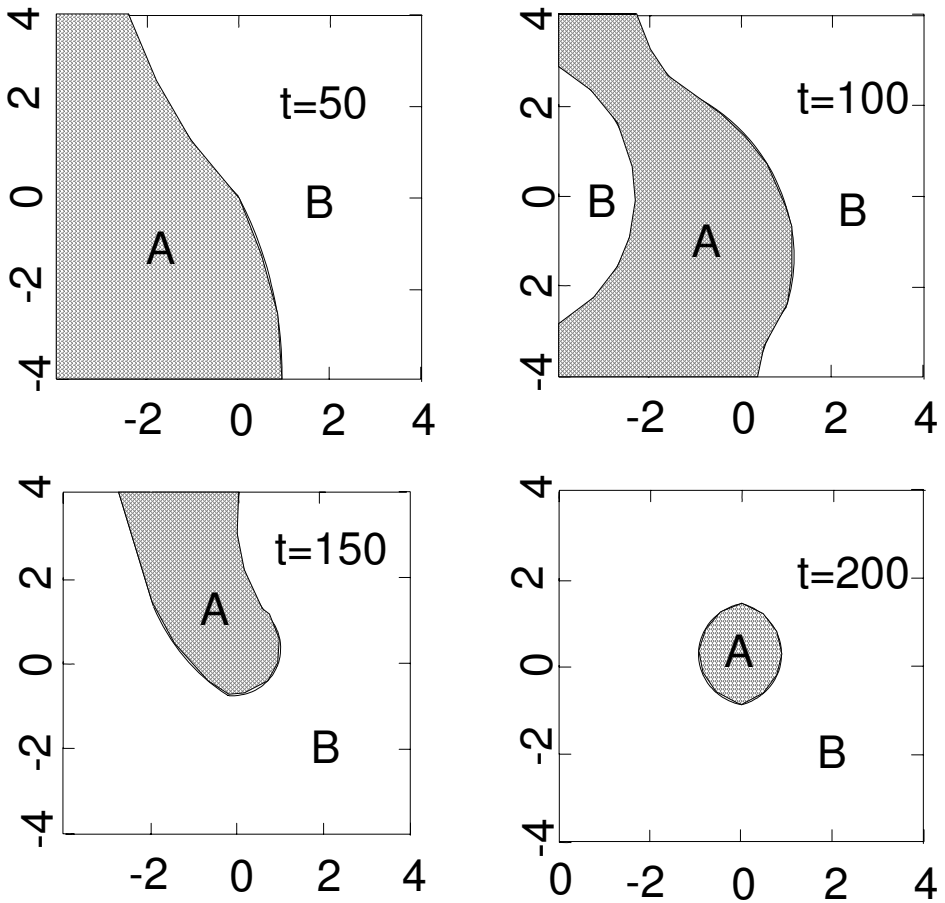
$$\widehat{w}_{ij}^{(l)}(n+1) = \widehat{w}_{ij}^{(l)}(n) + \mu\delta_j^{(l)}(n)\, y_i^{(l-1)}(n)$$

with $\delta^{(L)}(n) = e(n)y(n)[1 - y(n)]$ and

$$\delta_j^{(l)}(n) = y_j^{(l)}(n)\left[1 - y_j^{(l)}(n)\right]\sum_k \delta_k^{(l+1)}(n)\, w_{jk}^{(l+1)}(n)$$

where the summation covers all the errors of the $(l+1)$th layer and $e(n) = d(n) - y(n)$ is the error at the output of the network.

Class 1: $D(0,1)$ Class 2: $D(0,5)\backslash D(0,1)$



Remarks

- Epoch
- How should we choose the parameter μ?
  - small μ: slow convergence, small residual error
  - large μ: fast convergence, large residual error
- Classification rule with constant momentum noté $\beta \in [0, 1[$

$$\widehat{w}_{ij}(n+1) = \widehat{w}_{ij}(n) + \Delta\widehat{w}_{ij}(n)$$

$$\Delta\widehat{w}_{ij}(n) = \beta\Delta\widehat{w}_{ij}(n-1) + \mu\delta_j(n)\, y_i(n)$$

  For $\beta = 0$: classical backpropagation rule
  For $\beta > 0$: $\Delta\widehat{w}_{ij}(n) = \mu\sum_{k=0}^{n}\beta^{n-k}\delta_j(n)\, y_i(n)$
  Convergence is ensured if $0 \leq |\beta| \leq 1$.
- The convergence is accelerated when consecutive errors have the same sign whereas it is reduced when two consecutive errors have different signs.
- Stopping rules are problem dependent. For example, in some equalization problems, we use a sequence of known samples (pilot symbols) and the learning algorithm is stopped when $\|$input - output$\| < \varepsilon$
- Universal approximation theorem of Cybenko.

Summary

Example of decision tree

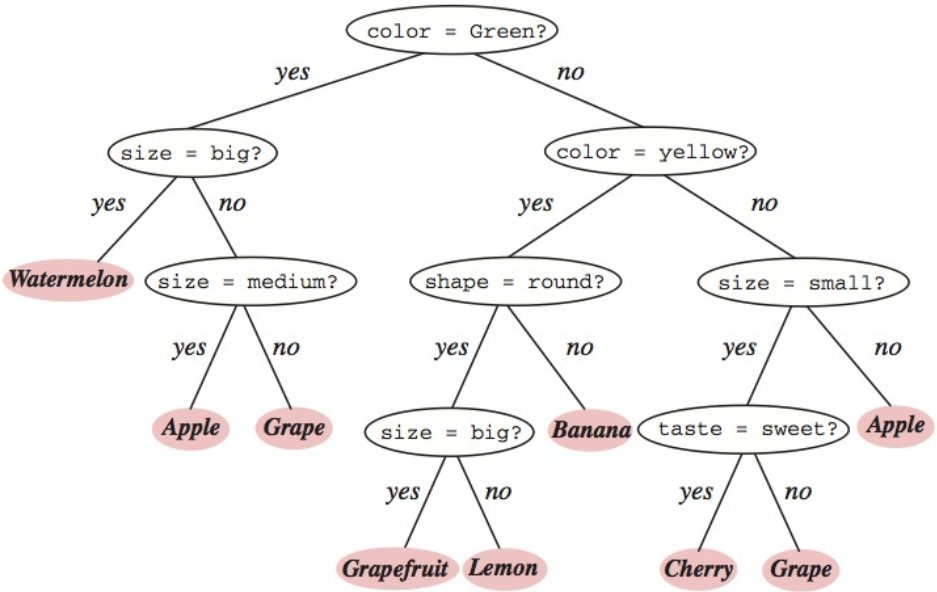


FIGURE 8.2. A tree with arbitrary branching factor at different nodes can always be represented by a functionally equivalent binary tree—that is, one having branching factor $B = 2$ throughout, as shown here. By convention the "yes" branch is on the left, the "no" branch on the right. This binary tree contains the same information and implements the same classification as that in Fig. 8.1. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

- Construction of the tree
- Classification rule

Splitting Rules

Inhomogeneity or impurity of the data

- Entropy (Algorithm C4.5)

$$i_n = -\sum_j \frac{n_j}{n} \log_2 \left(\frac{n_j}{n} \right)$$

- Gini Index (CART)

$$i_n = \sum_j \frac{n_j}{n} \left(1 - \frac{n_j}{n} \right)$$

Drop of Impurity

$$\Delta i_n = i_n - P_L i_L - P_R i_R$$

where $P_L = \frac{n_L}{n}, P_R = \frac{n_R}{n}$ are the proportions of the sets D_L, D_R.

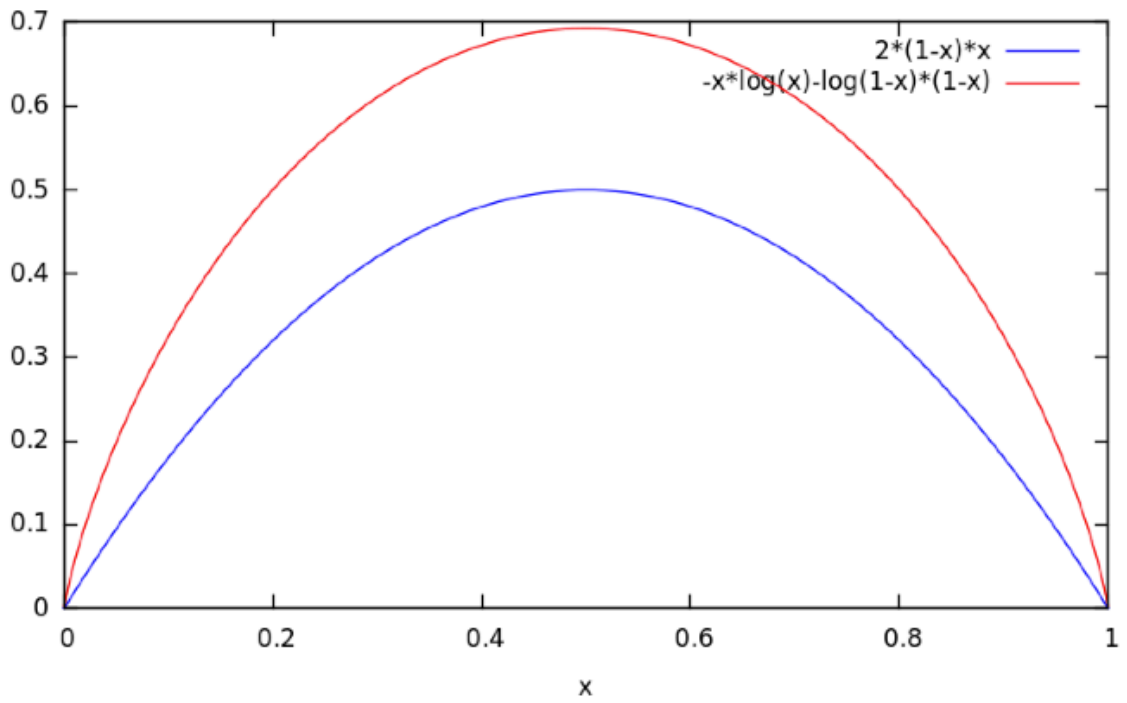
Choose the split associated with the maximum drop of impurity!

Gini Index or Entropy?

Example of 2 classes ($x = n_1/n$)

$$\text{Gini}(S) = 2x(1-x)$$

$$\text{Ent}(S) = -x\log x - (1-x)\log(1-x)$$



Implementation by default

Algorithm CART (classregtree.m)

- All variables are considered for each split
- All splits are considered
- Stopping rule: pure node or number of elements less than $n_{\min}$ (specified by the user)
- Splitting rule: Gini index

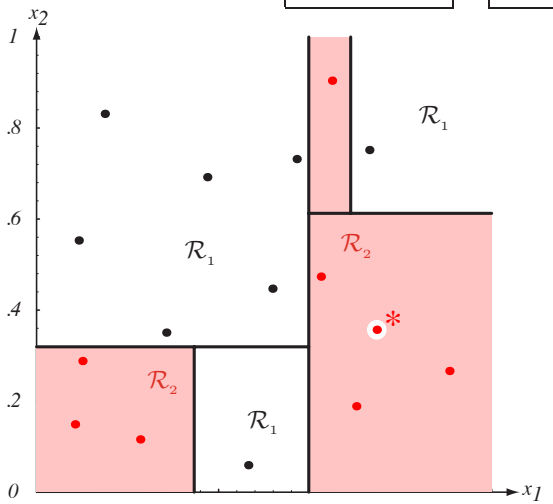
CART algorithm (example #1 for vectors in $\mathbb{R}^2$)

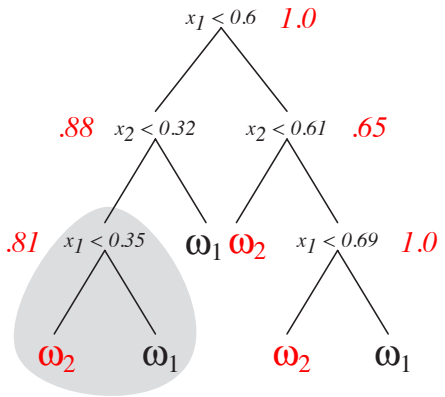
Example 1: A simple tree classifier

Consider the following $n = 16$ points in two dimensions for training a binary CART tree ($B = 2$) using the entropy impurity (Eq. 1).

ω_1 (black)	
x_1	x_2
.15	.83
.09	.55
.29	.35
.38	.70
.52	.48
.57	.73
.73	.75
.47	.06

ω_2 (red)	
x_1	x_2
.10	.29
.08	.15
.23	.16
.70	.19
.62	.47
.91	.27
.65	.90
.75	.36* (.32†)



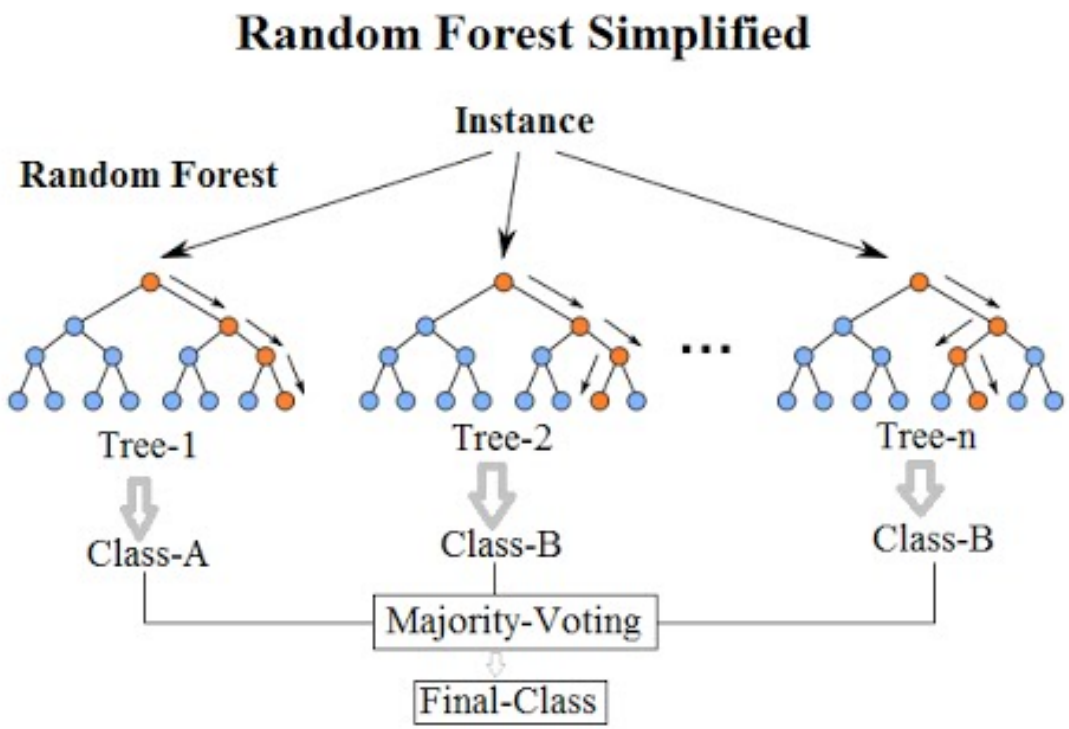


CART algorithm (example #2 for qualitative data)

	Weight	Size	Age	Result
$\boldsymbol{x}_1$	Light	Small	Young	Pass
$\boldsymbol{x}_2$	Light	Small	Young	Pass
$\boldsymbol{x}_3$	Light	Tall	Young	Pass
$\boldsymbol{x}_4$	Light	Tall	Old	Fail
$\boldsymbol{x}_5$	Light	Tall	Old	Pass
$\boldsymbol{x}_6$	Light	Tall	Old	Fail
$\boldsymbol{x}_7$	Heavy	Small	Old	Fail
$\boldsymbol{x}_8$	Heavy	Small	Young	Fail
$\boldsymbol{x}_9$	Light	Small	Old	Pass
$\boldsymbol{x}_{10}$	Heavy	Tall	Old	Pass

- First branch (Gini Index)
  - Weight $\Rightarrow 2(1/7 + 1/15) \sim 0.42$
  - Size $\Rightarrow 12/25 \sim 0.48$
  - Age $\Rightarrow 2(3/40 + 3/20) \sim 0.45$

Random Forests

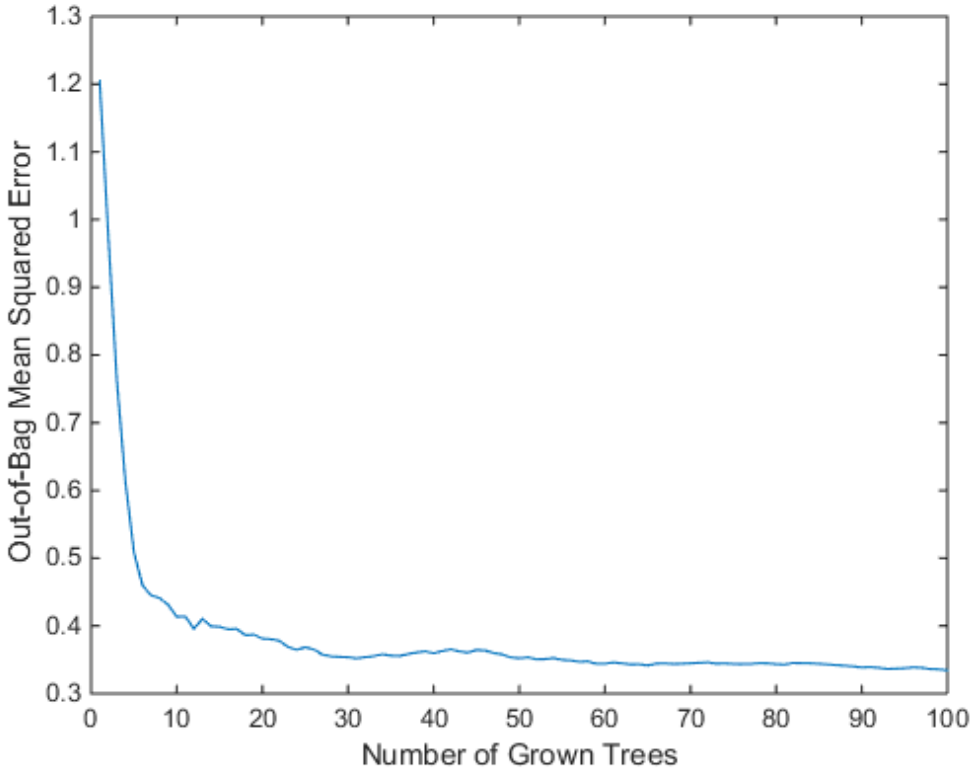


TreeBagger.m (options by default)

- Resampling all the data in the training set by bootstrap (and not a subset)
- Number of variables to select at random for each decision split: $\sqrt{n_{\text{var}}}$

Out-of-bag error

Matlab example



Comparison between CART and Random Forests

Book by G. James, D. Witten, T. Hastie and R. Tibshirani

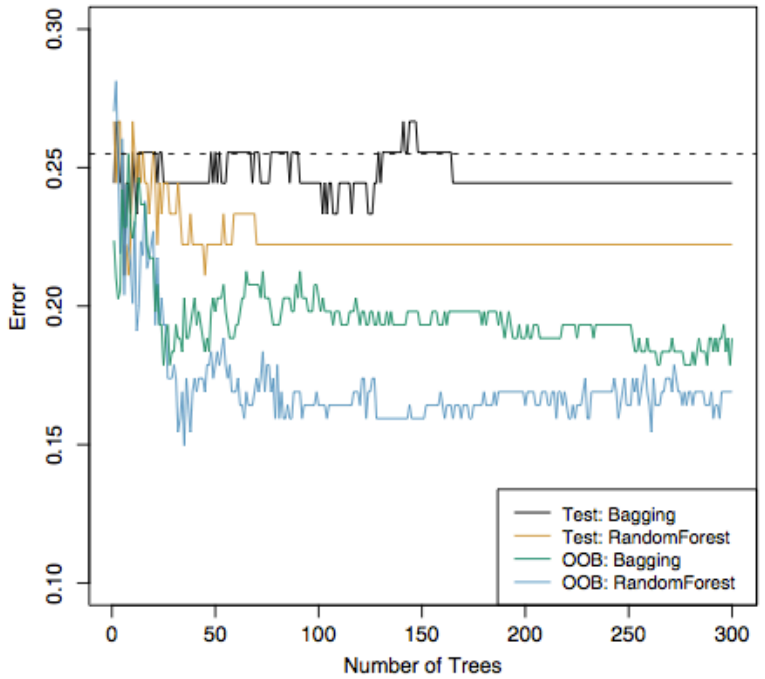


FIGURE 8.8. *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of B, the number of bootstrapped training sets used. Random forests were applied with $m = \sqrt{p}$. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.*

Importance of the different variables

Mean decrease in Gini index

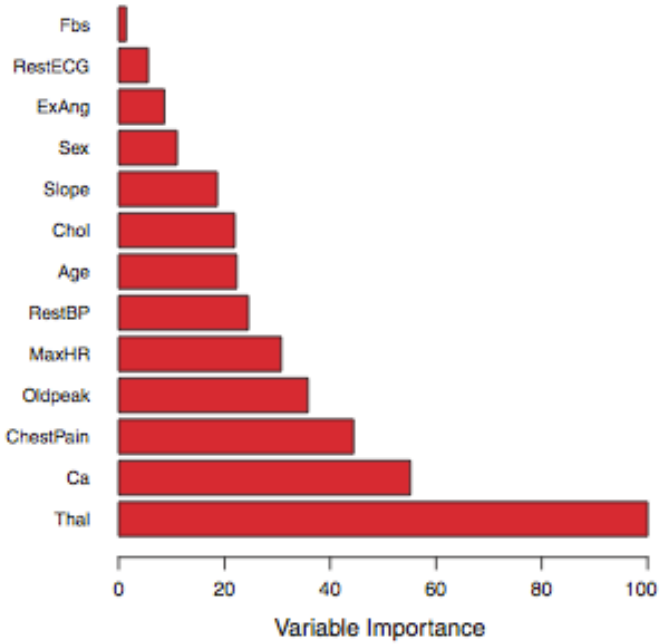


FIGURE 8.9. *A variable importance plot for the* `Heart` *data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.*

References

CART and Random Forests

- L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, Classification and Regression Trees, Chapman & Hall, New-York, 1993.
- L. Breiman, Random Forests, Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.
- R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification, 2nd edition, Wiley, 2000.
- G. James, D. Witten, T. Hastie and R. Tibshirani, An Introduction to Statistical Learning with Applications in R, Springer, New-York, 2013.

Thanks for your attention