

# Analyse de Données

## Slides 1ère année Sciences du Numérique

Jean-Yves Tourneret<sup>(1)</sup> et Axel Carlier<sup>(2)</sup>

(1) Université de Toulouse, ENSEEIHT-IRIT  
[jyt@n7.fr](mailto:jyt@n7.fr), <http://perso.tesa.prd.fr/jyt/>, [axel.carlier@toulouse-inp.fr](mailto:axel.carlier@toulouse-inp.fr)

Janvier 2023

## Summary

- ▶ Chapter 1 : Introduction
  - ▶ Examples
  - ▶ Classification Model
- ▶ Chapter 2 : Statistical Classification Methods
  - ▶ Bayesian Rule
  - ▶ Supervised Learning
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
  - ▶ Support vector machines (SVMs)
  - ▶ Neural networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods

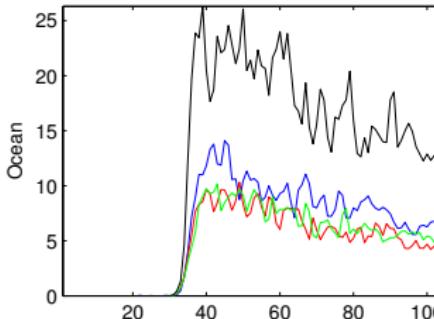
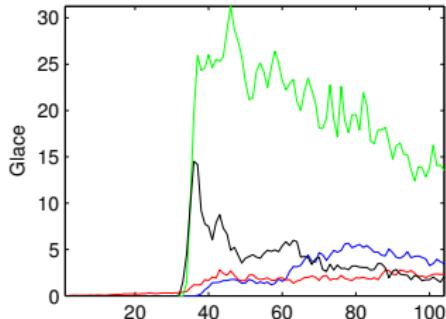
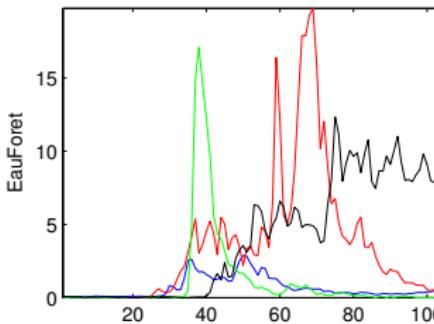
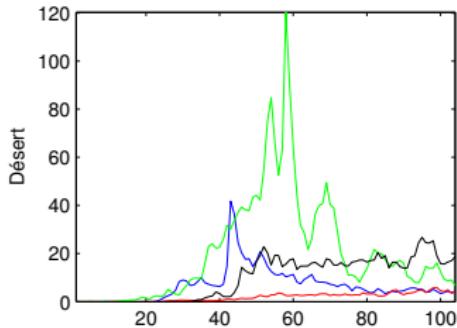
## Summary

- ▶ Chapter 1 : Introduction
  - ▶ Examples
    - ▶ Altimetry
    - ▶ Electromyography (EMG)
  - ▶ Classification Model
- ▶ Chapter 2 : Statistical Classification Methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees

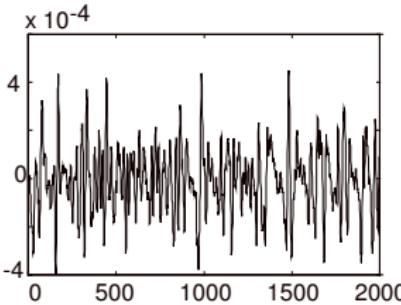
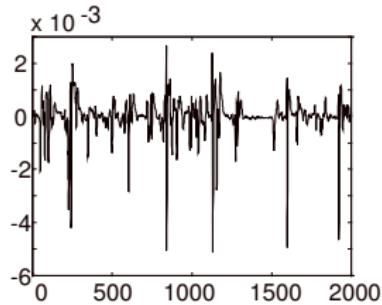
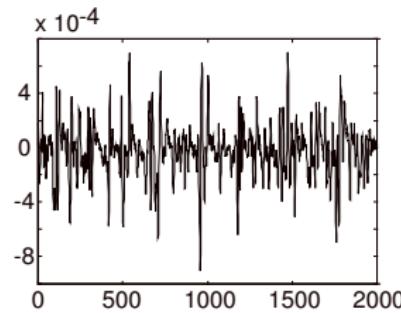
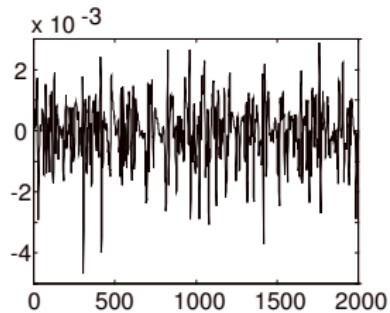
## References

- ▶ 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.

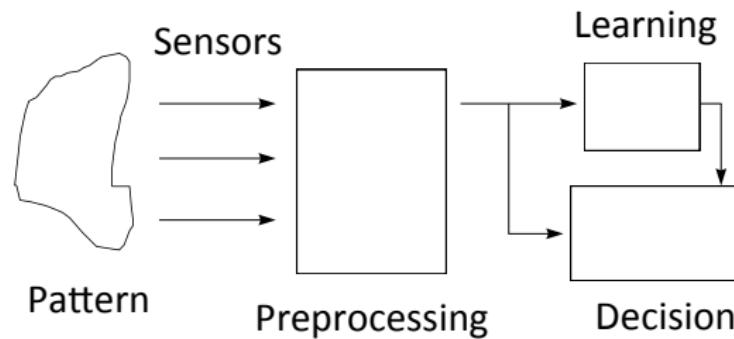
## Example 1 : Classification of Altimetric Signals



## Example 2 : Classification of EMG Signals



## Classification Model



## Summary

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
  - ▶ Bayesian rule
    - ▶ Theory
    - ▶ Two classes
    - ▶ Uninformative cost function
    - ▶ Gaussian densities
  - ▶ Supervised learning
    - ▶ Parametric methods
    - ▶ Non-parametric methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods

# Classification

## Notations

- ▶  $K$  classes  $\omega_1, \dots, \omega_K$
- ▶  $\mathbf{x} = [x(1), \dots, x(p)]^T$  **measurements**  $\in X = \mathbb{R}^p$
- ▶  $A$ : set of possible **actions**  $a_1, \dots, a_q$  where  $a_k$  = "assign the vector  $\mathbf{x}$  to the class  $\omega_k$ ",  $\forall k = 1, \dots, K$

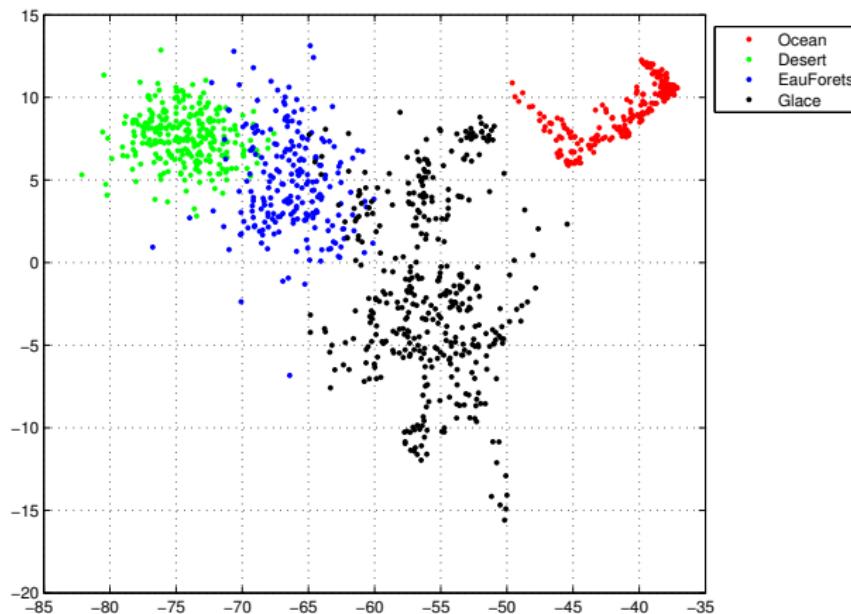
## Definition

$$\begin{aligned} d : & \quad X \rightarrow A \\ & \mathbf{x} \mapsto d(\mathbf{x}) \end{aligned}$$

## Remark

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

## Example



## Bayesian Rule

### Hypothesis: Probabilistic Model

- *A priori* probability of class  $\omega_k$

$$P(\omega_k)$$

- Probability density function of the observation vector  $\mathbf{x}$  *conditionally to class  $\omega_k$*

$$f(\mathbf{x} | \omega_k)$$

### Conclusion

- *A posteriori* probability that  $\mathbf{x}$  belongs to class  $\omega_k$

$$P(\omega_k | \mathbf{x}) = \frac{f(\mathbf{x} | \omega_k) P(\omega_k)}{f(\mathbf{x})}$$

with  $f(\mathbf{x}) = \sum_{k=1}^K f(\mathbf{x} | \omega_k) P(\omega_k)$ .

## MAP Classifier

### Definition

$$d^*(\mathbf{x}) = a_j \Leftrightarrow P(\omega_j | \mathbf{x}) \geq P(\omega_k | \mathbf{x}), \forall k \in \{1, \dots, K\}$$

### Equiprobable Classes: Maximum Likelihood Classifier

$$d^*(\mathbf{x}) = a_j \Leftrightarrow f(\mathbf{x} | \omega_j) \geq f(\mathbf{x} | \omega_k), \forall k \in \{1, \dots, K\}$$

### Property

The MAP classifier minimizes the probability of error

## Proof (2 classes)

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

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

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

$P_e$  is minimum when  $R_2 = \{\mathbf{x} / P(\omega_2 | \mathbf{x}) > P(\omega_1 | \mathbf{x})\}$

## Probability of Error

- ▶ Definition ( $K$  classes)

$$P_e = \sum_{k=1}^K P [d(x) = a_k \cap x \notin \omega_k]$$

- ▶ Proof  
admitted

## Gaussian Case

### Densities

$$f(\mathbf{x} | \omega_k) = \frac{1}{(2\pi)^{p/2} \sqrt{\det \Sigma_k}} \exp \left[ -\frac{1}{2} (\mathbf{x} - \mathbf{m}_k)^T \Sigma_k^{-1} (\mathbf{x} - \mathbf{m}_k) \right]$$

### General Case

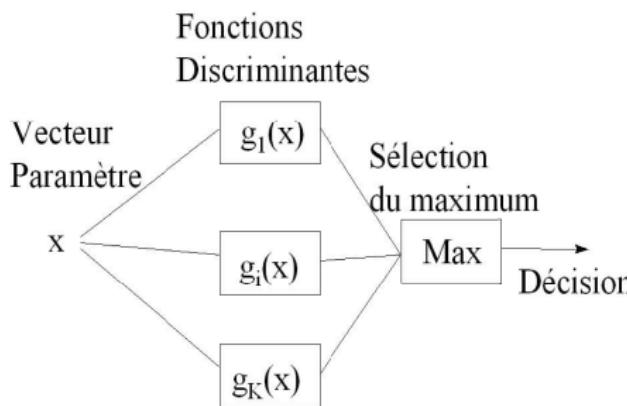
$$d^*(\mathbf{x}) = a_i \Leftrightarrow g_i(\mathbf{x}) \geq g_k(k) \quad \forall k = 1, \dots, K$$

with

$$g_i(\mathbf{x}) = -(\mathbf{x} - \mathbf{m}_i)^T \Sigma_i^{-1} (\mathbf{x} - \mathbf{m}_i) - \ln \det \Sigma_i + 2 \ln P(\omega_i)$$

## Gaussian Case

### Classifier



**Discriminant functions**

Quadratic term + linear term  
+ constant

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

### Equiprobable classes - Centroid Distance Rule

$$d^*(\mathbf{x}) = a_i \Leftrightarrow d_M(\mathbf{x}, \mathbf{m}_i) \leq d_M(\mathbf{x}, \mathbf{m}_k) \quad \forall k = 1, \dots, K$$

where  $d_M$  is the Mahalanobis distance

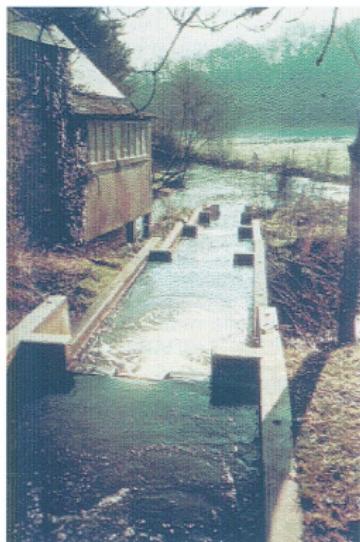
$$d_M(\mathbf{x}, \mathbf{m}_k) = \sqrt{(\mathbf{x} - \mathbf{m}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \mathbf{m}_k)}$$

### Non equiprobable classes: affine discriminant functions

$$d^*(\mathbf{x}) = a_i \Leftrightarrow \left[ \mathbf{x} - \frac{1}{2} (\mathbf{m}_i + \mathbf{m}_k) \right]^T \boldsymbol{\Sigma}^{-1} (\mathbf{m}_i - \mathbf{m}_k) \geq \ln \frac{P(\omega_k)}{P(\omega_i)}, \quad \forall k = 1, \dots, K$$

# Contexte de l'étude

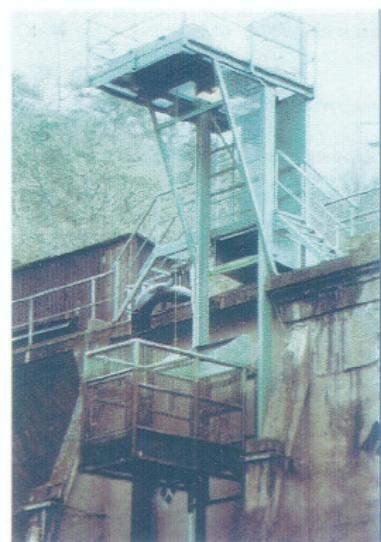
## *Les passes à poissons*



à bassins successifs



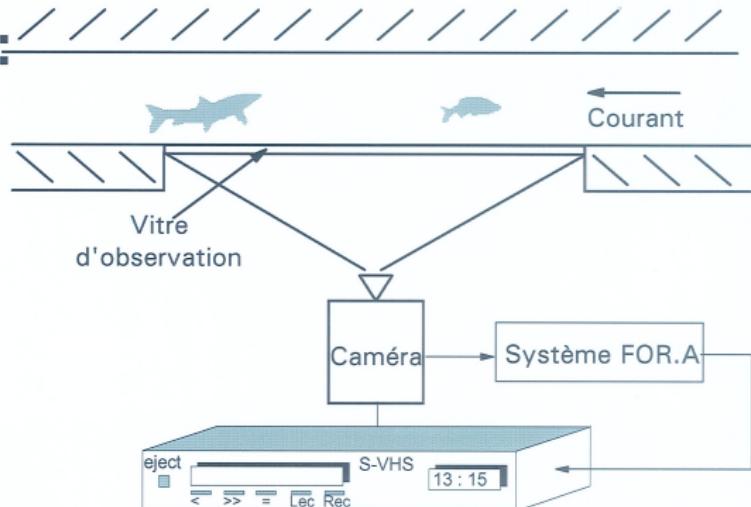
à ralentisseurs



ascenseur

## *Suivi des passes à poissons*

- Dispositif vidéo actuel de comptage par espèces :



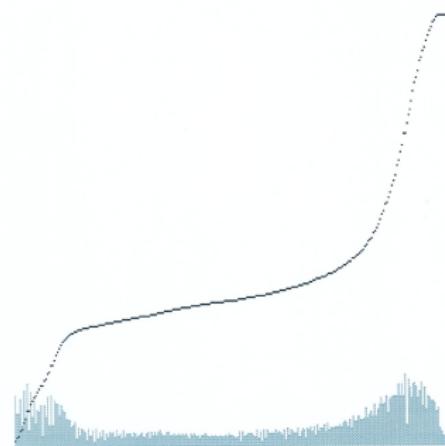
- Dépouillement des bandes vidéo enregistrées par un opérateur humain

# 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 :

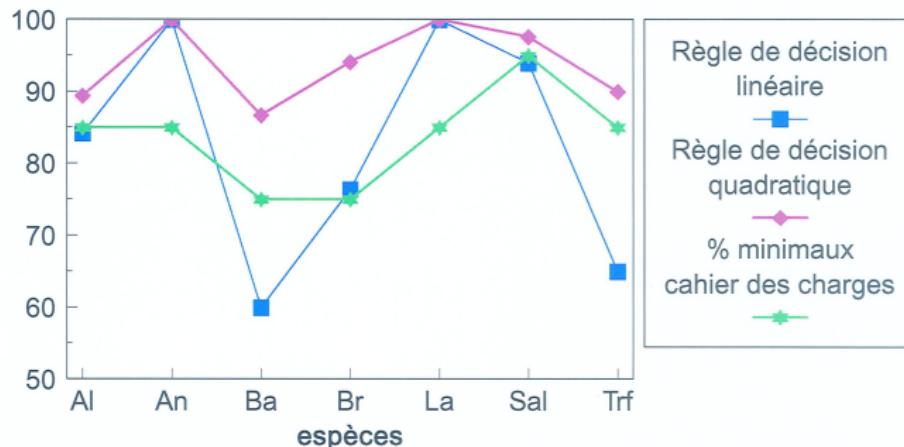


- Fournir :
  - La comptabilisation des différentes espèces
  - Les dates et heures de passage

# Mesure d'efficacité de la règle de classement

## Résultats obtenus sur la BDT

% de reconnaissance dynamiques sur la base de données test  
(comparaison des règles de décision quadratique et linéaire)



- Règle de décision linéaire
- Règle de décision quadratique
- % minimaux cahier des charges

## Summary

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
  - ▶ Bayesian rule
    - ▶ Theory
    - ▶ Two classes
    - ▶ Uninformative cost function
    - ▶ Gaussian densities
  - ▶ Supervised learning
    - ▶ Parametric methods
    - ▶ Non-parametric methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods

## Parametric methods

### Principle

Assume that the distribution of  $x = (x_1, \dots, x_n)^T$  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

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
  - ▶ Bayesian rule
    - ▶ Theory
    - ▶ Two classes
    - ▶ Uninformative cost function
    - ▶ Gaussian densities
  - ▶ Supervised learning
    - ▶ Parametric methods
    - ▶ Non-parametric methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods

## The $k$ -nearest neighbor rule

### The nearest neighbor rule

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

The observed vector  $x$  is assigned 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

$x$  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

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

## Comparing and selecting classifiers.

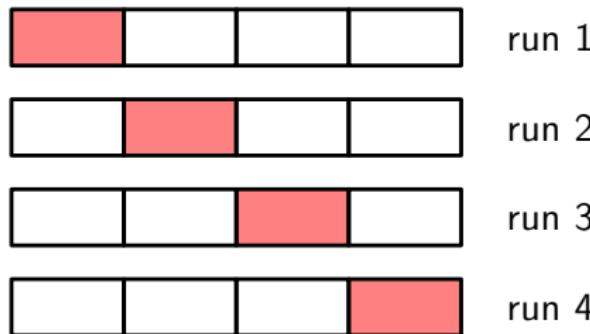
- ▶ We just saw how a “k-nearest-neighbors” (knn) classifier works and the rationale behind such a non-parametric method.
- ▶ How should we select  $k$ ? is a remaining question.

### 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  $\zeta$  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

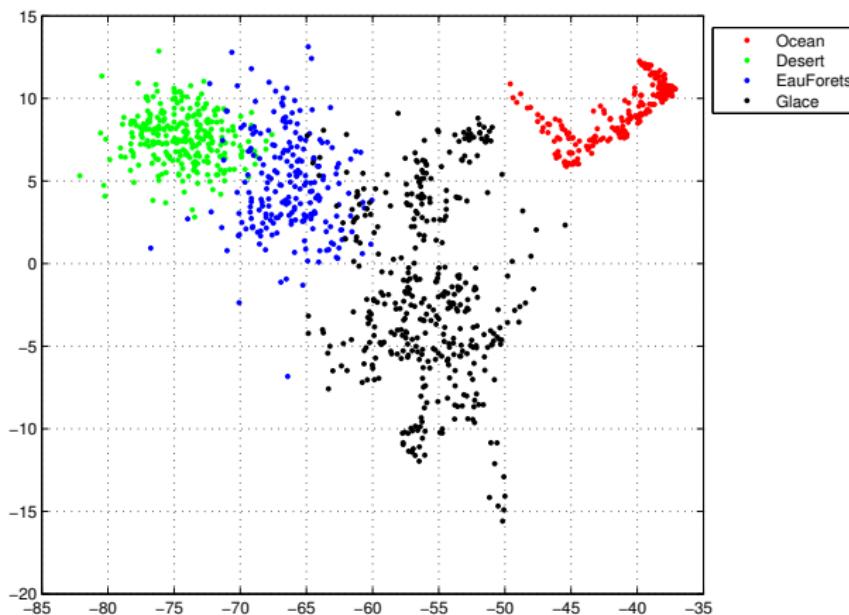
CV score for each  $k$  and  $\zeta(j)$  is the  $j$ th group used as testing data

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

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

$$CV(k) = \frac{1}{n} \sum_{i=1}^n c(f_{k_{\text{nn}}}^{D-\{(\mathbf{x}_i, y_i)\}}(\mathbf{x}_i), y_i)$$

## Example



## Some classification results

Bayesian Classification (Gaussian densities)

<i>Classes</i>	<i>Ocean</i>	<i>Desert</i>	<i>Forest</i>	<i>Ice</i>
<i>Ocean</i>	<b>99.6</b>	0.0	0.0	0.0
<i>Desert</i>	0.0	<b>95.3</b>	1.8	0.0
<i>Forest</i>	0.0	4.4	<b>97.7</b>	0.8
<i>Ice</i>	0.4	0.4	0.5	<b>99.2</b>

1-PPV Method

<i>Classes</i>	<i>Ocean</i>	<i>Desert</i>	<i>Forest</i>	<i>Ice</i>
<i>Ocean</i>	<b>100</b>	0.0	0.0	0.0
<i>Desert</i>	0.0	<b>96.0</b>	5.4	0.0
<i>Forest</i>	0.0	4.0	<b>93.2</b>	0.0
<i>Ice</i>	0.0	0.0	1.4	<b>100</b>

## Some classification results

### 5-PPV Method

<i>Classes</i>	<i>Ocean</i>	<i>Desert</i>	<i>Forest</i>	<i>Ice</i>
<i>Ocean</i>	<b>100</b>	0.0	0.0	0.0
<i>Desert</i>	0.0	<b>93.8</b>	5.9	0.3
<i>Forest</i>	0.0	6.2	<b>93.2</b>	0.0
<i>Ice</i>	0.0	0.0	0.9	<b>99.7</b>

### Neural Networks

<i>Classes</i>	<i>Ocean</i>	<i>Desert</i>	<i>Forest</i>	<i>Ice</i>
<i>Ocean</i>	<b>100</b>	0.0	0.0	0.0
<i>Desert</i>	0.0	<b>96.0</b>	5.4	0.0
<i>Forest</i>	0.0	4.0	<b>92.8</b>	0.8
<i>Ice</i>	0.0	0.0	1.8	<b>99.2</b>

## Summary

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
  - ▶ Support vector machines (SVMs)
    - ▶ Introduction
    - ▶ Optimal separating hyperplane
    - ▶ Optimization problem
    - ▶ The "soft-margin SVM" classifier
    - ▶ Non-separable case: the " $\nu$ -SVM" classifier
    - ▶ Non-linear preprocessing - kernels
  - ▶ Neural networks
    - ▶ The multi-layer perceptron
    - ▶ The backpropagation algorithm
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods

## Support vector machines (SVMs)

### Learning set

$$\mathcal{B} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$$

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

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

### Hyperplane definition

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

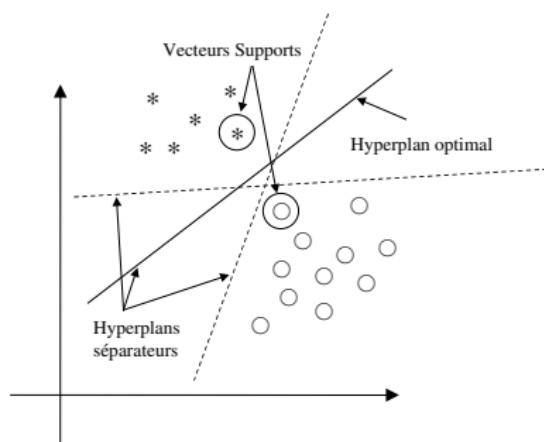
with

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

### Classification rule

$$f(\mathbf{x}) = \text{sign}[g_{\mathbf{w}, b}(\mathbf{x})] \tag{1}$$

## Illustration



## Problem formulation

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

$$\gamma_i(\tilde{\mathbf{w}}) = \frac{y_i (\mathbf{w}^T \mathbf{x}_i - b)}{\|\mathbf{w}\|}$$

with  $\tilde{\mathbf{w}} = (\mathbf{w}, b)$  ( $\mathbf{x}_i$  is correctly classified by (1) if  $\gamma_i(\tilde{\mathbf{w}}) > 0$ )

Margin of the learning set

$$\gamma_{\mathcal{B}}(\tilde{\mathbf{w}}) = \min_{i \in \{1, \dots, n\}} \frac{y_i (\mathbf{w}^T \mathbf{x}_i - b)}{\|\mathbf{w}\|}$$

Since  $\gamma_{\mathcal{B}}(a\tilde{\mathbf{w}}) = \gamma_{\mathcal{B}}(\tilde{\mathbf{w}})$ ,  $\forall a > 0$ ,  $\tilde{\mathbf{w}}$  is not unique!

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

$$y_i (\mathbf{w}^T \mathbf{x}_i - b) = 1 \Rightarrow \min_{i \in \{1, \dots, n\}} y_i (\mathbf{w}^T \mathbf{x}_i - b) = 1$$

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

## Problem formulation

### Canonical hyperplane

$$y_i (\mathbf{w}^T \mathbf{x}_i - b) \geq 1, \quad \forall i = 1, \dots, n$$

Classifier margin for a canonical hyperplane

$$\gamma_{\mathcal{B}}(\tilde{\mathbf{w}}) = \frac{1}{\|\mathbf{w}\|}$$

New formulation

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

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

# Optimization

## Lagrangian

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

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

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

## Kuhn and Tucker multipliers

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

$$L'(\mathbf{x}) = f'(\mathbf{x}) + \sum_{i=1}^n \alpha_i G'_i(\mathbf{x}) = 0$$

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

## Dual problem

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

$$\mathbf{w} = \sum_{\text{Support vectors}} \alpha_i y_i \mathbf{x}_i = \mathbf{x}^T \mathbf{Y} \boldsymbol{\alpha} \quad (2)$$

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

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

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  $\mathbf{w}_0$

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

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

Classifier margin

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

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

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

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

## Extensions

### The “soft-margin” SVM classifier

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

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

### The $\nu$ -SVM classifier

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

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

## Non-linear preprocessing: the kernel trick

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

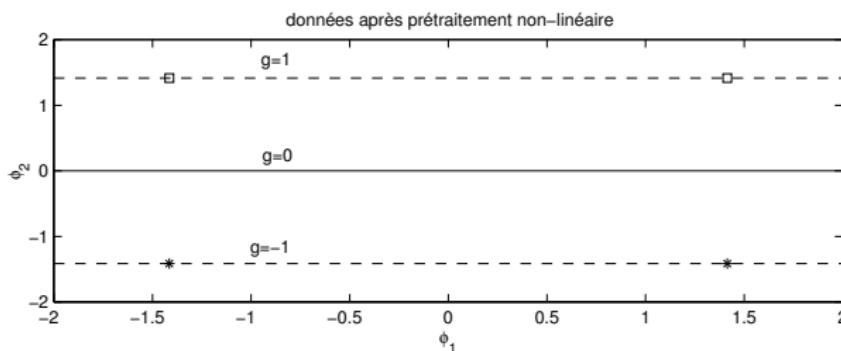
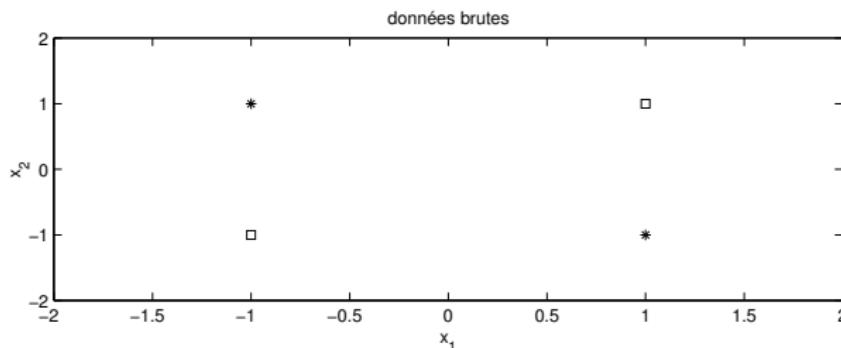
### Classical example

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

$$\begin{aligned}\phi : \quad & \mathbb{R}^2 \rightarrow \mathbb{R}^6 \\ & (x_1, x_2)^T \mapsto (\sqrt{2}x_1, \sqrt{2}x_1x_2, 1, \sqrt{2}x_2, x_1^2, x_2^2)^T\end{aligned}$$

The data are separable in the plane  $(\phi_1, \phi_2)$

## Example of separability



## Non-linear SVM classifier

Decision rule after preprocessing

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

Cost function to optimize

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

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

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

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

## Kernels

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

**Example:**  $\mathbf{x} = (x_1, x_2)^T$  and  $\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$

$$\begin{aligned}\phi(\mathbf{x})^T \phi(\mathbf{y}) &= \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \cdot \begin{pmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}y_1y_2 \end{pmatrix} \\ &= x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \\ &= (\mathbf{x}^T \mathbf{y})^2\end{aligned}$$

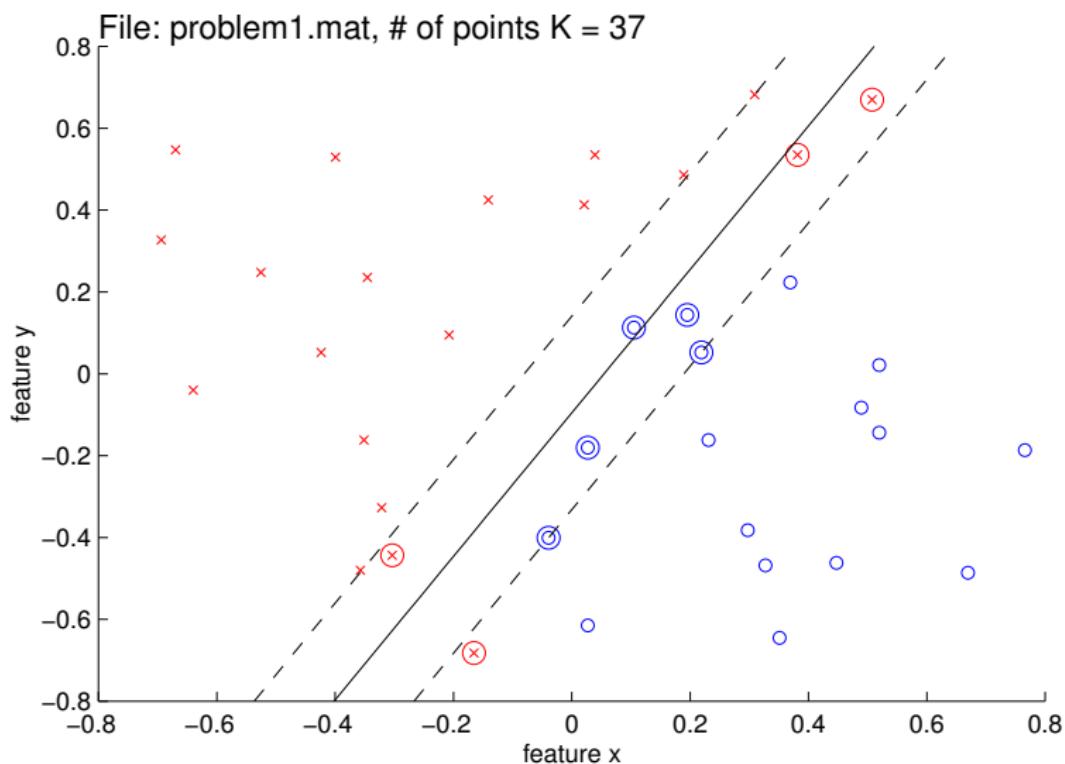
**Mercer kernels** can be expressed as inner products

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

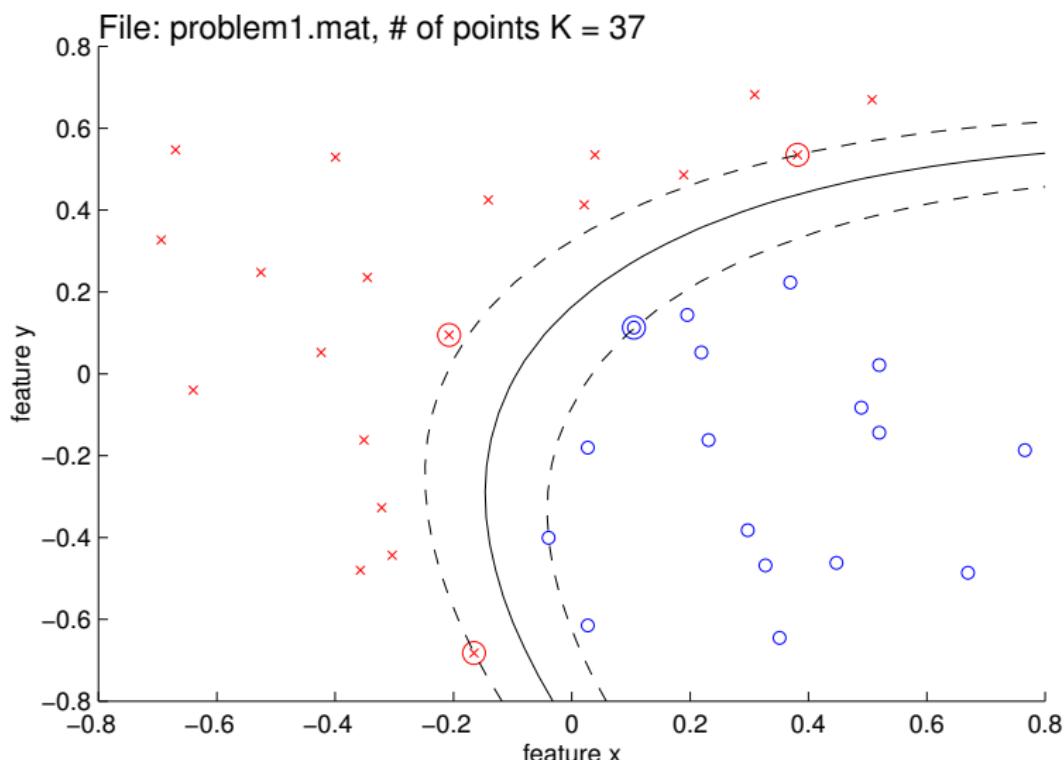
## Classical kernels

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

## Example of linear SVMs



## Example of non-linear SVMs



## Equalization of non-linear communication channels

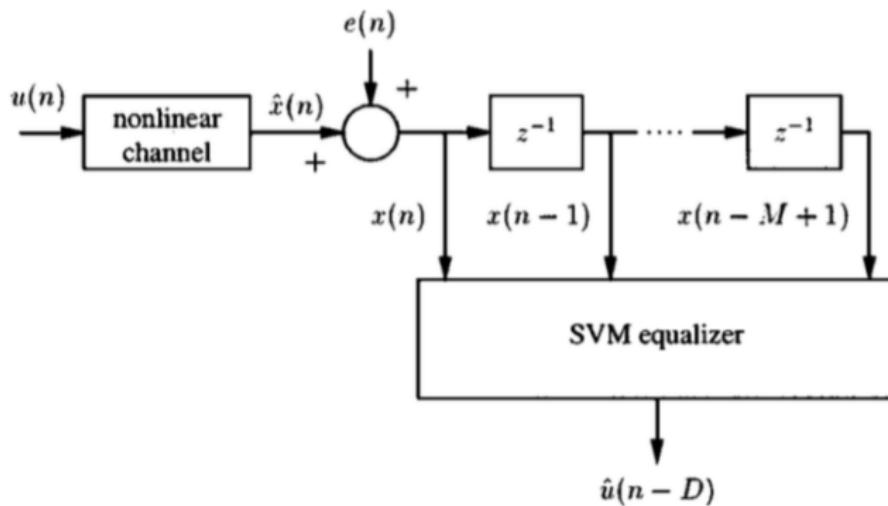


Fig. 2. Model of the nonlinear transmission system originating from Chen *et al.* [12].

## Equalization of non-linear communication channels

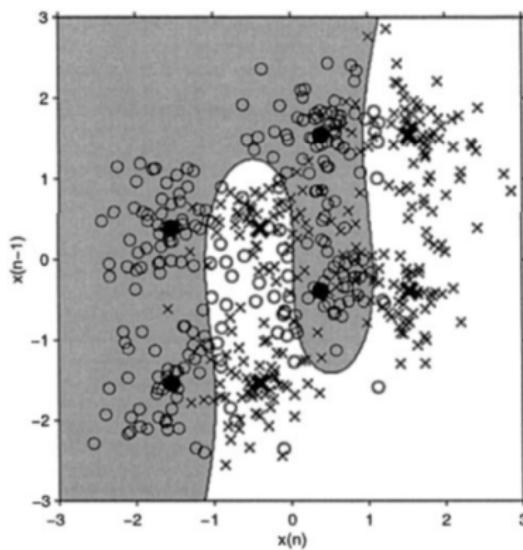


Fig. 3. Example of typical classification regions of an SVM and associated training points with channel  $\hat{x}(n) = \tilde{x}(n) - 0.9 \tilde{x}^3(n)$ ,  $\tilde{x}(n) = u(n) + 0.5u(n-1)$ , and Gaussian white noise of power  $\sigma_e^2 = 0.2$  and equalizer dimension  $M = 2$ , polynomial kernel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

## Equalization of non-linear communication channels

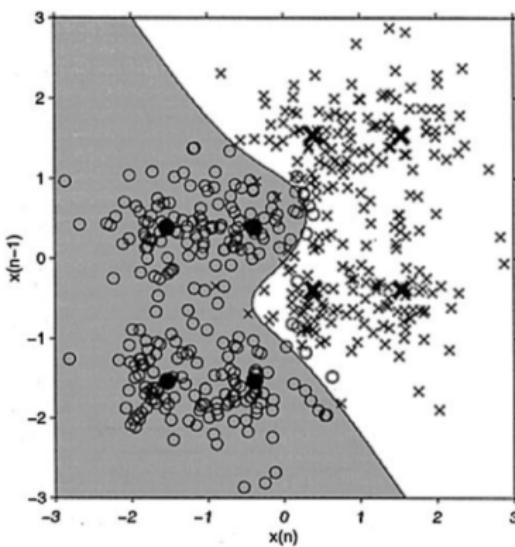


Fig. 4. Example of typical classification regions of an SVM and associated training points with the channel described in Fig. 3, except the equalizer lag is now  $D = 1$ .

## Equalization of non-linear communication channels

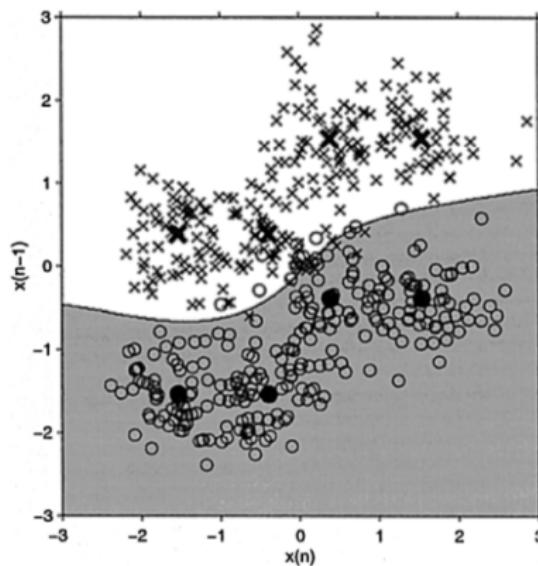
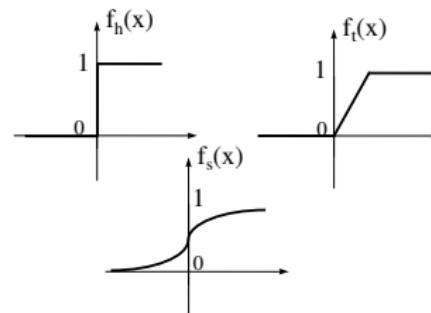
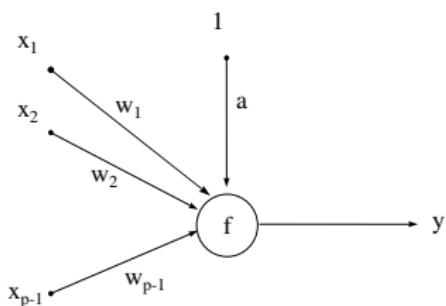


Fig. 5. Example of typical classification regions of an SVM and associated training points with the channel described in Fig. 3, except the equalizer lag is now  $D = 2$ .

## Summary

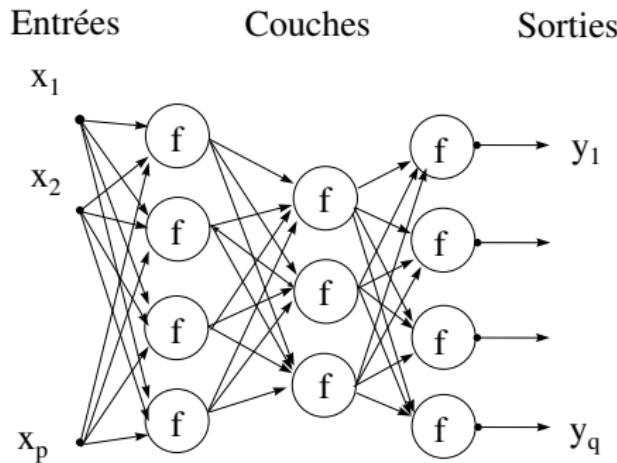
- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
  - ▶ Support vector machines (SVMs)
    - ▶ Introduction
    - ▶ Optimal separating hyperplane
    - ▶ Optimization problem
    - ▶ The "soft-margin SVM" classifier
    - ▶ Non-separable case: the " $\nu$ -SVM" classifier
    - ▶ Non-linear preprocessing - kernels
  - ▶ Neural networks
    - ▶ The multi-layer perceptron
    - ▶ The backpropagation algorithm
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods

## Neural networks



- ▶ **inputs**:  $x_1, \dots, x_{p-1}$
- ▶ **weights**:  $w_1, \dots, w_{p-1}$
- ▶ **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  $j$ th output of the  $l$ th layer (with  $y^{(L)}(n) = y(n)$  et

$y_j^{(0)}(n) = x_j(n)$ ),  $N_l$  is the number of nodes of the  $l$ th layer and  $L$  is the number of layers.

☞ Weight update

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

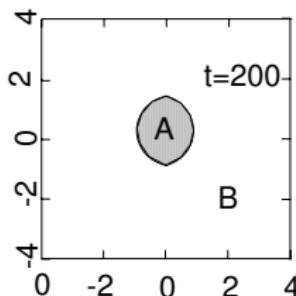
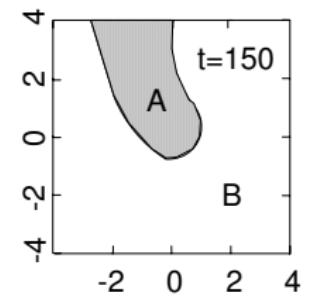
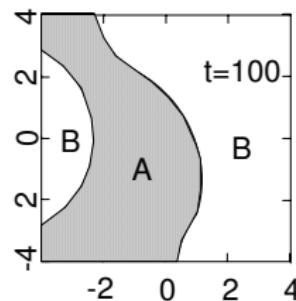
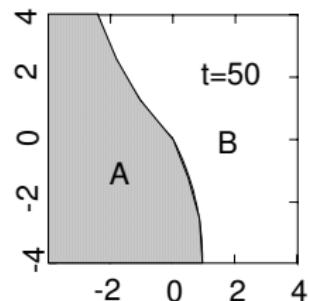
with  $\delta^{(L)}(n) = e(n)y(n)[1 - y(n)]$  and

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

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:  $D(0, 5) \setminus D(0, 1)$



## Remarks

- ▶ Epoch
- ▶ How should we choose the parameter  $\mu$ ?
  - ▶ small  $\mu$ : slow convergence, small residual error
  - ▶ large  $\mu$ : 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  $\beta > 0$ :  $\Delta\widehat{w}_{ij}(n) = \mu \sum_{k=0}^n \beta^{n-k} \delta_j(n) y_i(n)$

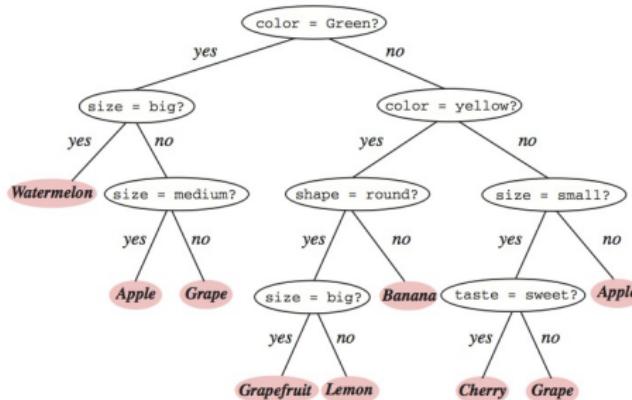
Convergence is ensured if  $0 \leq |\beta| \leq 1$ .

- ▶ 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  $\|\text{input} - \text{output}\| < \varepsilon$
- ▶ Universal approximation theorem of Cybenko.

## Summary

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods

## 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 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)

$$i_n = \sum_j \frac{n_j}{n} \left( 1 - \frac{n_j}{n} \right) = 1 - \sum_j \left( \frac{n_j}{n} \right)^2$$

## 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  $D_L, D_R$ .

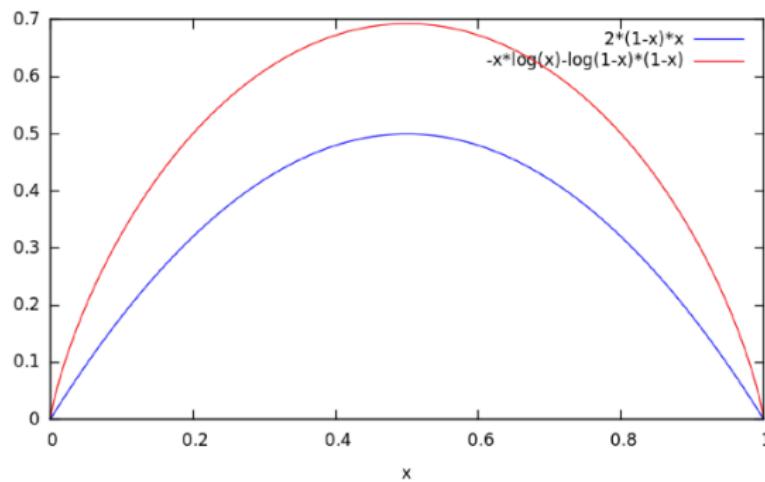
Choose the split associated with the maximum drop of impurity!

## 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)

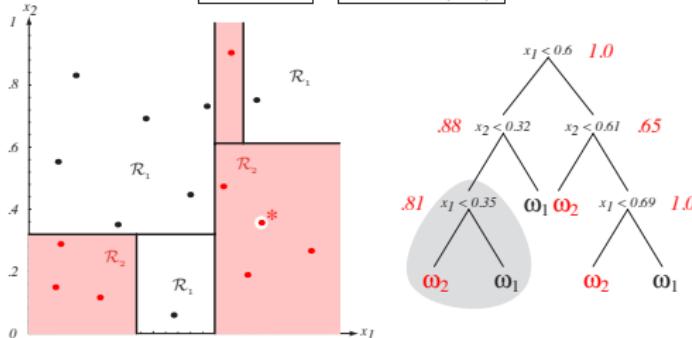
- ▶ All variables are considered for each split
- ▶ All splits are considered
- ▶ Stopping rule: pure node or number of elements less than  $n_{\min}$  (specified by the user)
- ▶ Splitting rule: Gini index

# 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).

$\omega_1$ (black)		$\omega_2$ (red)	
$x_1$	$x_2$	$x_1$	$x_2$
.15	.83	.10	.29
.09	.55	.08	.15
.29	.35	.23	.16
.38	.70	.70	.19
.52	.48	.62	.47
.57	.73	.91	.27
.73	.75	.65	.90
.47	.06	.75	.36* (.32†)

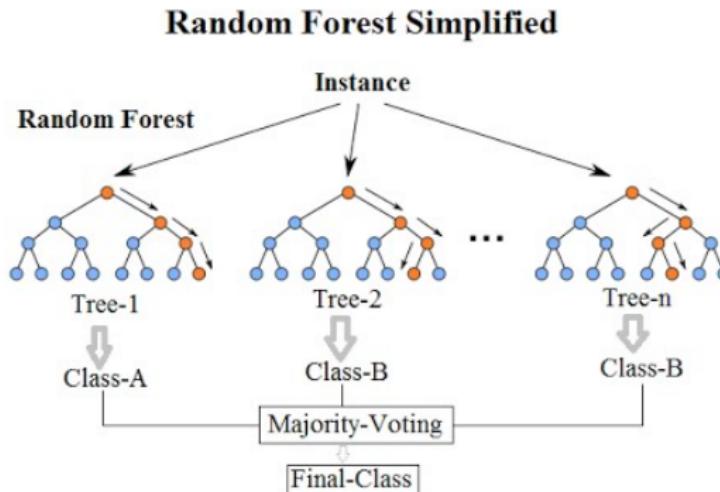


## CART algorithm (example #2 for qualitative data)

	Weight	Size	Age	Result
$x_1$	Light	Small	Young	Pass
$x_2$	Light	Small	Young	Pass
$x_3$	Light	Tall	Young	Pass
$x_4$	Light	Tall	Old	Fail
$x_5$	Light	Tall	Old	Pass
$x_6$	Light	Tall	Old	Fail
$x_7$	Heavy	Small	Old	Fail
$x_8$	Heavy	Small	Young	Fail
$x_9$	Light	Small	Old	Pass
$x_{10}$	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

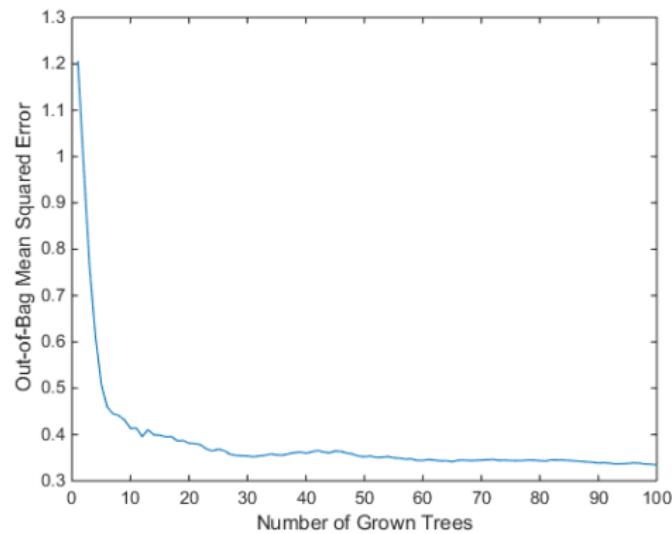


### TreeBagger.m (options by default)

- ▶ **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:  $\sqrt{n_{\text{var}}}$

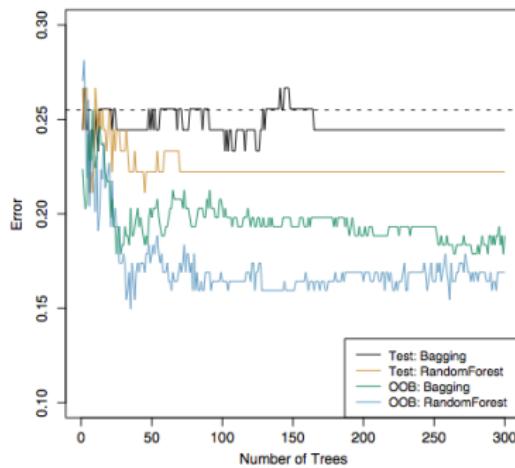
## Out-of-bag error

Matlab example



## Comparison between CART and Random Forests

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

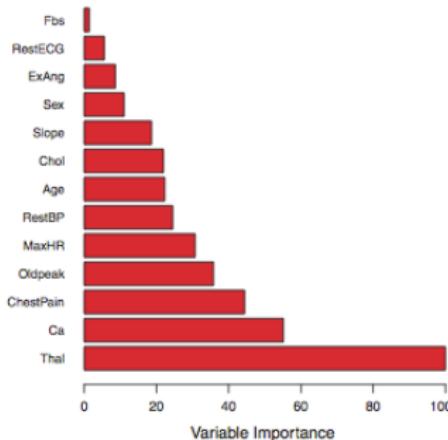


**FIGURE 8.8.** Bagging and random forest results for the Heart data. The test error (black and orange) is shown as a function of  $B$ , the number of bootstrapped training sets used. Random forests were applied with  $m = \sqrt{p}$ . The dashed line indicates the test error resulting from a single classification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

Average of decreases of the Gini index when the attribute has been used for splitting



**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.

## Summary

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods
  - ▶ Optimization methods
  - ▶ Hierarchical classification
  - ▶ Density-based methods
  - ▶ Mixture models
  - ▶ Spectral methods

## Unsupervised learning

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

### Motivations

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

### Optimal solution

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

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

*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  $\omega_i$  and of the partition  $\mathbf{X}$

$$E_i^2 = \sum_{k=1}^{N_i} d^2(\mathbf{x}_k, \mathbf{g}_i), \quad E^2 = \sum_{i=1}^K E_i^2$$

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

### Properties

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

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

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

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

## *K*-means Algorithm (ISODATA)

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

1. Initial **choice** of the number of classes and the class centroids
2. Assign each vector  $x_i$  to  $\omega_j$  (using the **centroid distance rule**) such that

$$d(\mathbf{x}_i, \mathbf{g}_j) = \inf_k d(\mathbf{x}_i, \mathbf{g}_k)$$

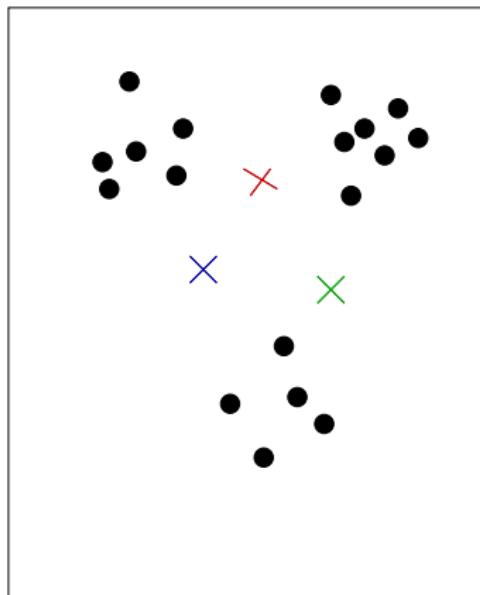
3. Compute the centroids  $\mathbf{g}_k^*$  of the new classes  $\omega_k^*$
4. Repeat steps 2) and 3) until convergence

► *Improved version of ISODATA*

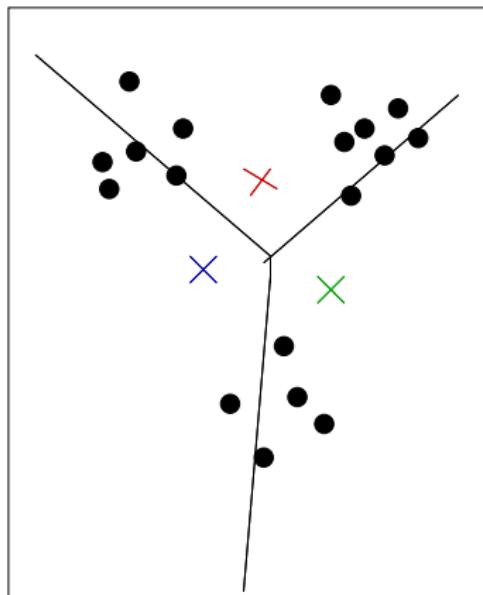
- Two classes are merged if their centroids are close
- A class is split if it contains too many vectors  $x_i$  or if its mean square error is too large

► *Convergence*: see notes or textbooks

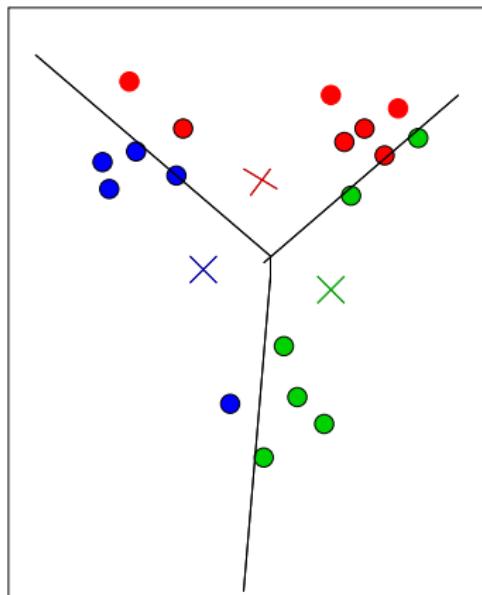
## K-means



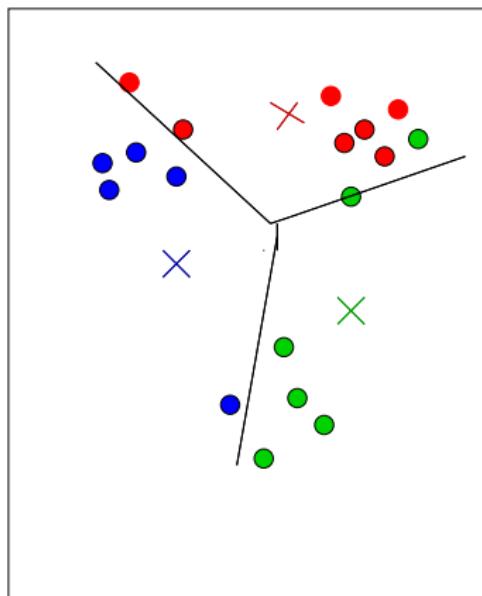
## K-means



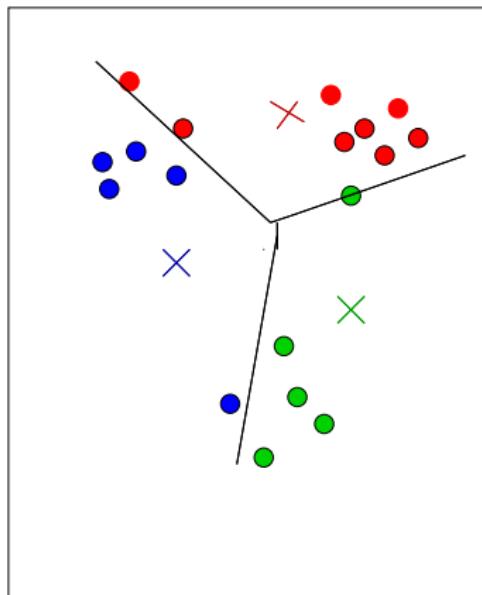
## K-means



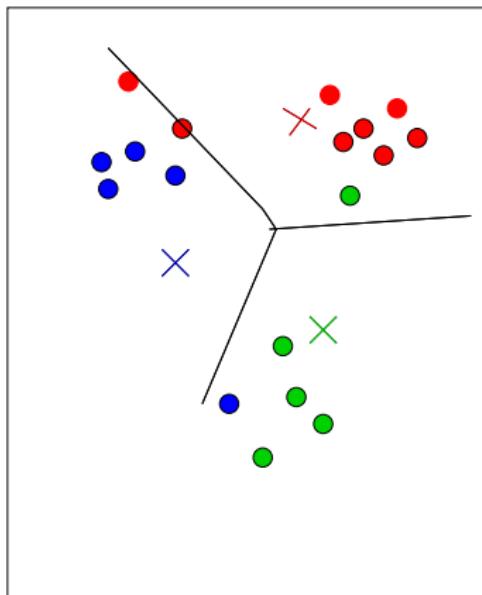
## K-means



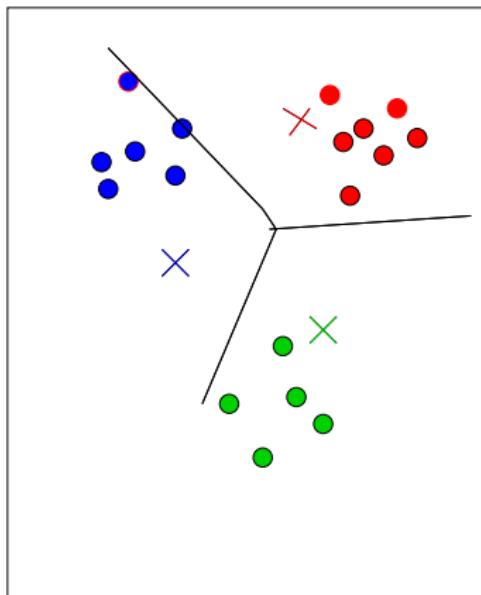
## K-means



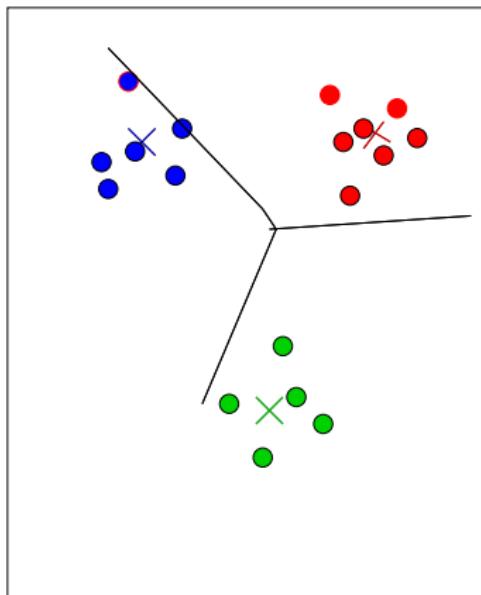
## K-means



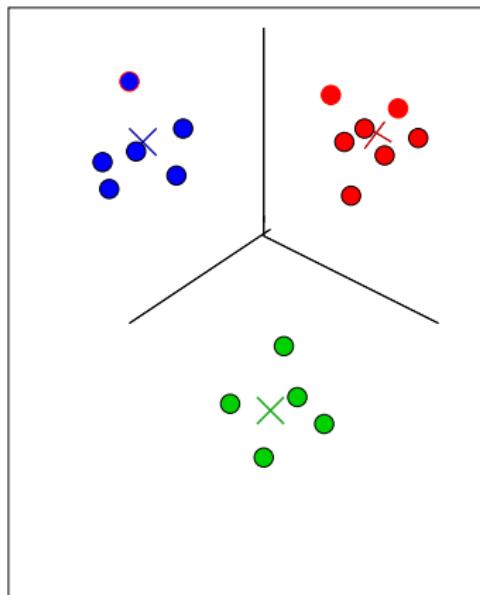
## K-means



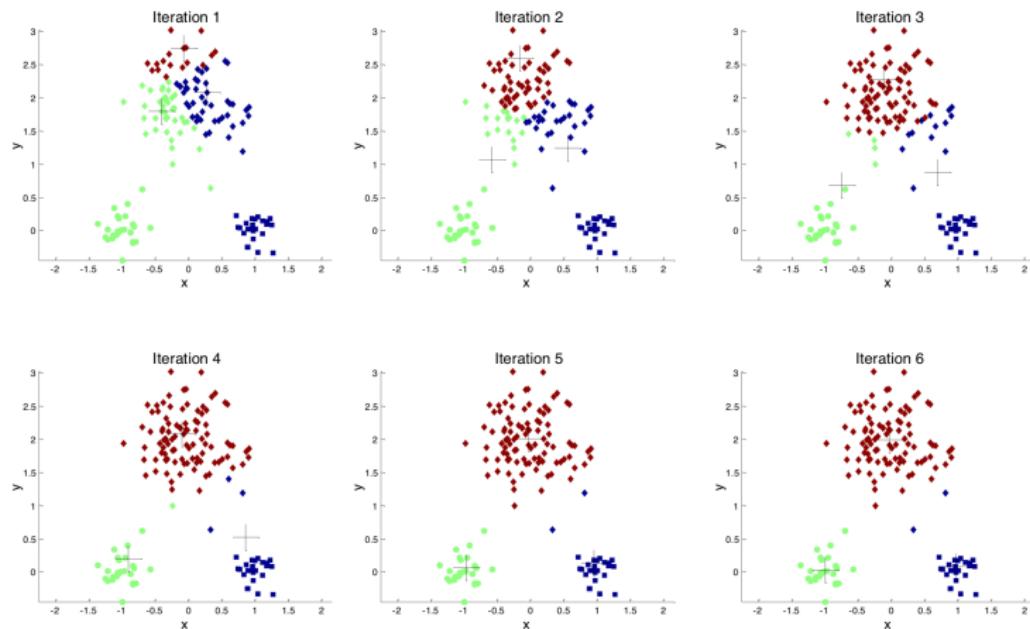
## K-means



## K-means

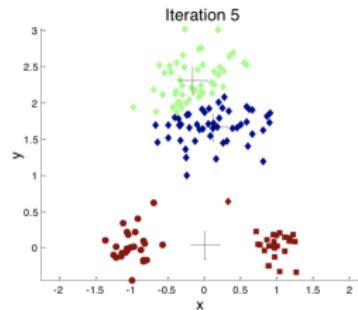
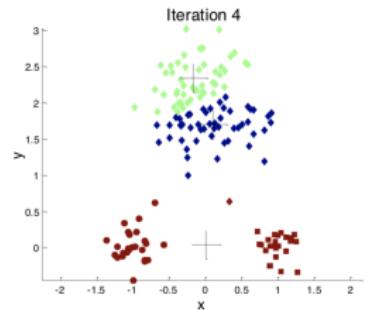
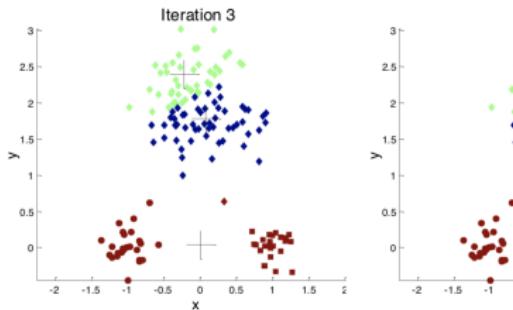
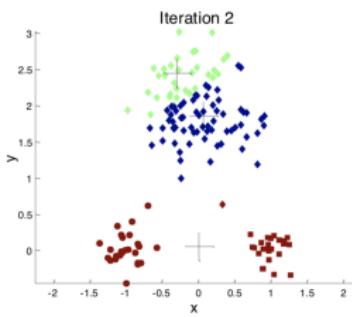
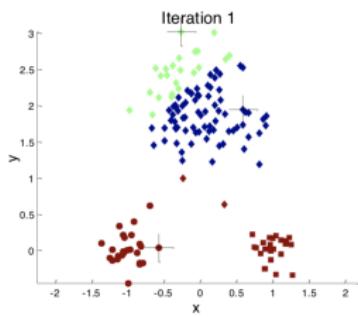


## Initialization Problems

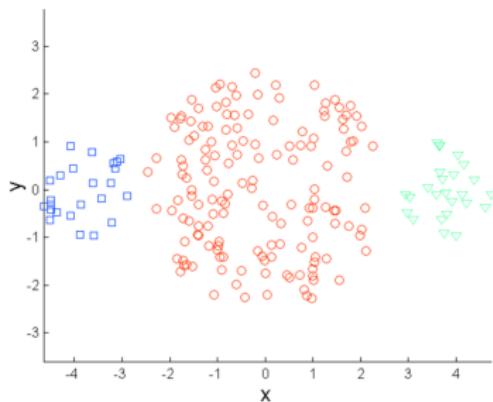


Thanks to Jing Gao from SUNY Buffalo university for her slides!!

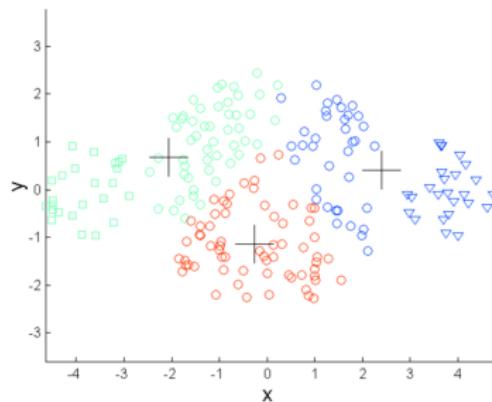
## Initialization Problems



## Limitations of K-means: Differing Sizes

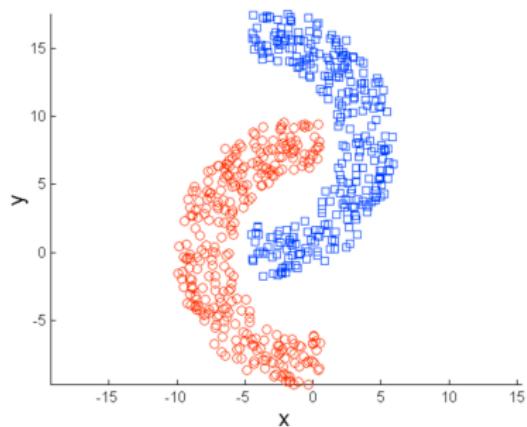


Original Points

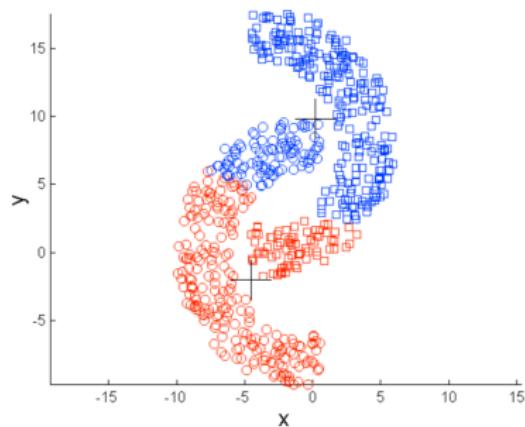


K-means (3 Clusters)

## Limitations of K-means: Irregular Shapes

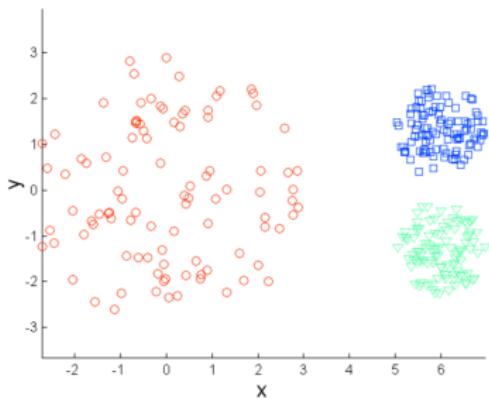


Original Points

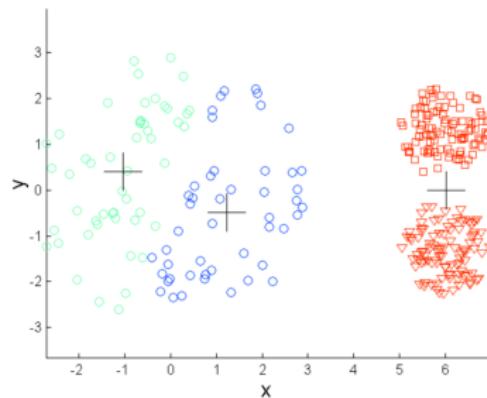


K-means (2 Clusters)

## Limitations of K-means: Differing Density



Original Points



K-means (3 Clusters)

## Summary

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods
  - ▶ Optimization methods
  - ▶ Hierarchical classification
  - ▶ Density-based methods
  - ▶ Mixture models
  - ▶ Spectral methods

## Hierarchical classification

Ascending hierarchy: method of distances

- Distance *Min* (single linkage algorithm)

$$d(X_i, X_j) = \min d(\mathbf{x}, \mathbf{y}) \quad \mathbf{x} \in X_i, \mathbf{y} \in X_j$$

This distance favors elongated classes

- Distance *Max* (complete linkage algorithm)

$$d(X_i, X_j) = \max d(\mathbf{x}, \mathbf{y}) \quad \mathbf{x} \in X_i, \mathbf{y} \in X_j$$

- Average linkage algorithm

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

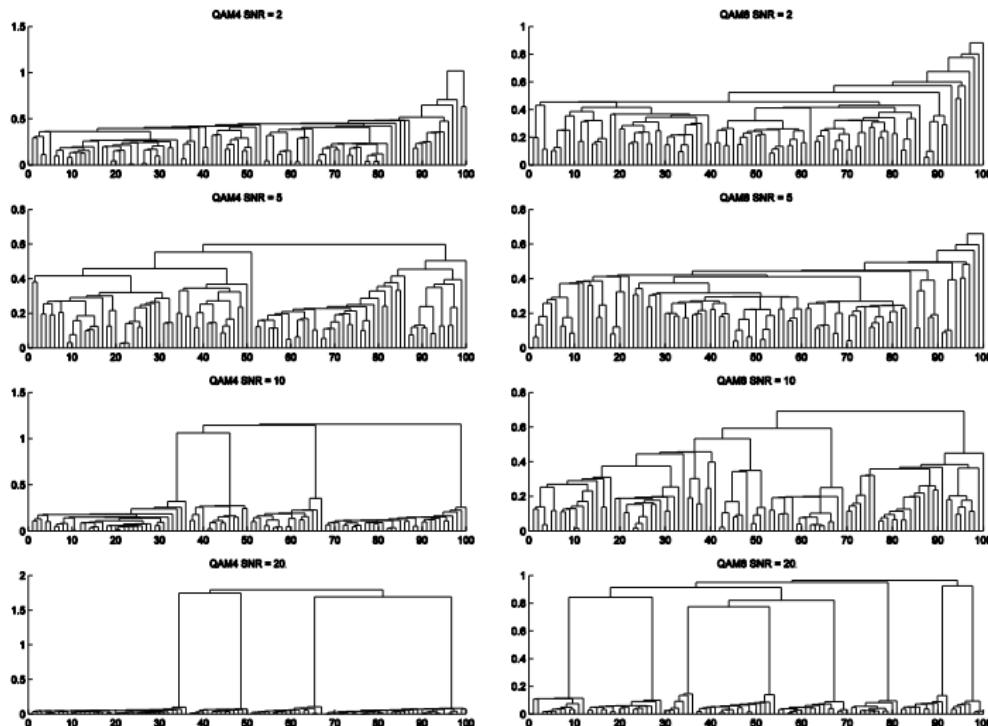
- Distance between the means

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

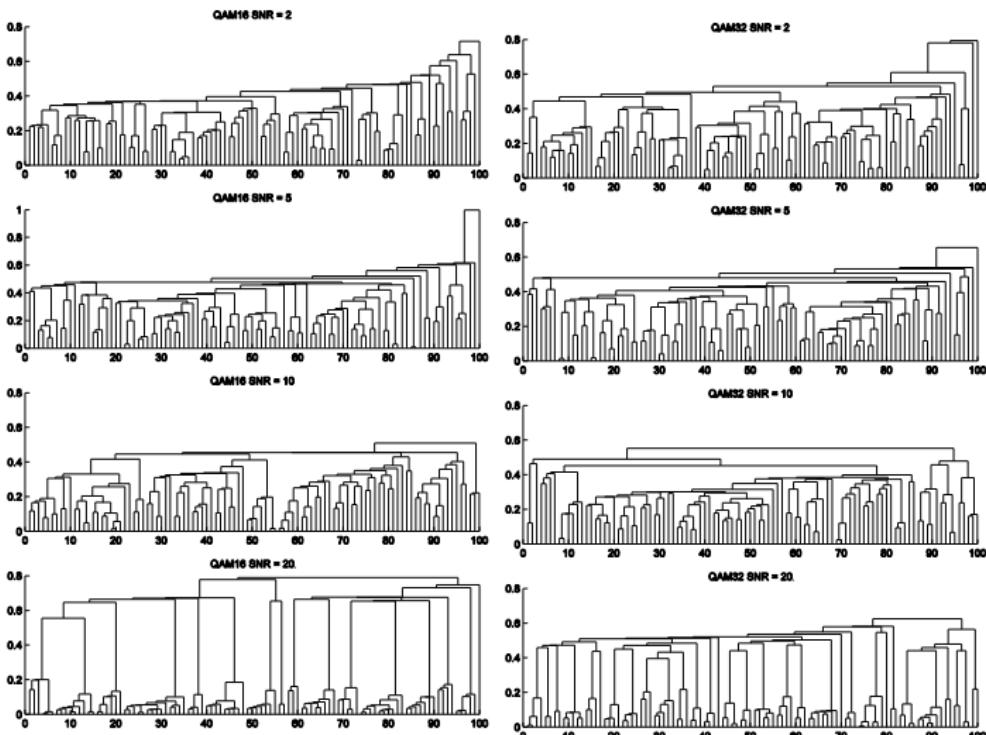
where  $X_i$  and  $X_j$  have cardinals  $N_i$  and  $N_j$  and centroids  $\mathbf{g}_i$  and  $\mathbf{g}_j$ .

Representation using a tree whose nodes indicate the different groups

# Classification of Modulations



# Classification of Modulations



## Summary

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods
  - ▶ Optimization methods
  - ▶ Hierarchical classification
  - ▶ Density-based methods
  - ▶ Mixture models
  - ▶ Spectral methods

## Density-based clustering methods

### Definitions

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

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

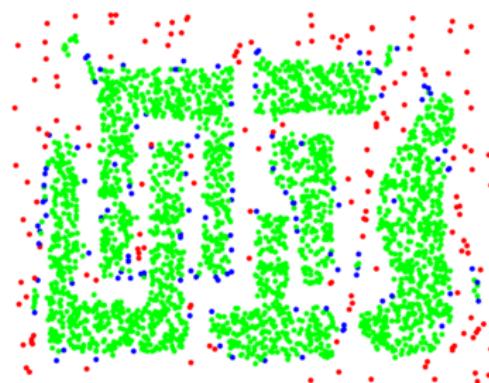
- Core point: a point  $p$  is a core point if it has more than MinPts neighbors in  $N_\epsilon(p)$ .
- Border point: a point  $p$  is a border point if it has less than MinPts neighbors in  $N_\epsilon(p)$  and if it is in the neighborhood of a core point
- Noise point: a noise point is a point that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

## Core, border and noise points



Original Points

$\varepsilon = 10$ , MinPts = 4



Point types: **core**,  
**border** and **outliers**

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

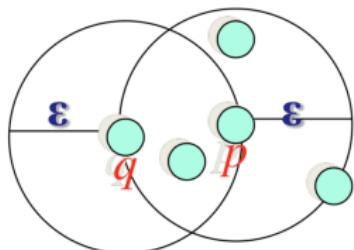
## Relationships between different points

### Directly density-reachable points

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

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

### 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!

$$\text{MinPts} = 4$$

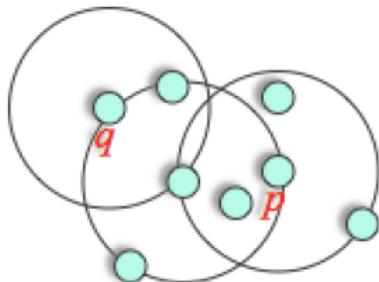
## Relationships between different points

### Density-reachable points

A point  $q$  is density-reachable from  $p$  for  $(\epsilon, \text{MinPts})$  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)

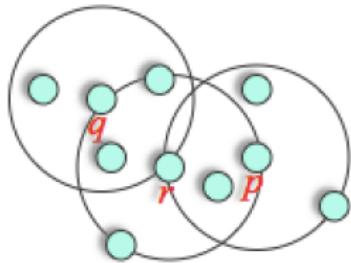
It is an asymmetric relationship!

## Relationships between different points

### Density-connectivity

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

### Example



It is a symmetric relationship!

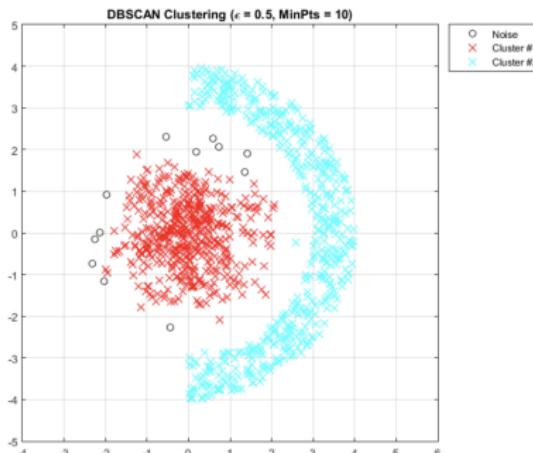
# Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

## 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



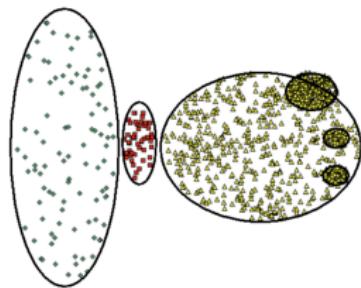
## DBSCAN

### A simple algorithm

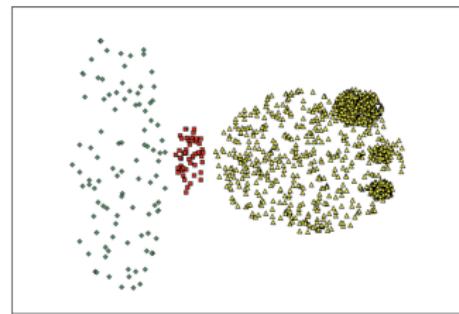
- ▶ Select a point  $p$
- ▶ Determine all density-reachable points from  $p$  for  $(\epsilon, \text{MinPts})$ .
- ▶ If  $p$  is a core point, i.e., if the cardinal of  $N_\epsilon(p)$  is larger than  $\text{MinPts}$ , a cluster is formed
- ▶ If  $p$  is a border point, DBSCAN visits the next point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

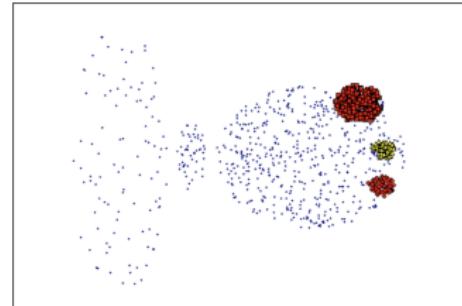
## How can we choose $\epsilon$ and MinPts?



- 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)

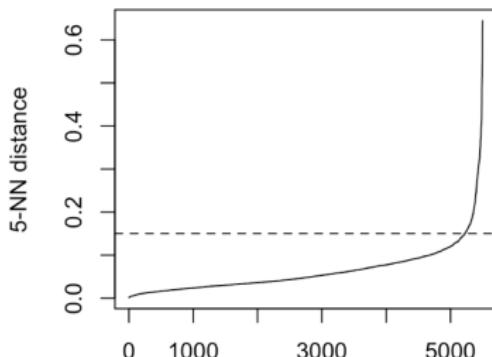
Thanks to Jing Gao from SUNY Buffalo university for her slides!!

## How can we choose $\epsilon$ 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  $\epsilon$  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

- ▶ Chapter 1 : Introduction
- ▶ Chapter 2 : Statistical Classification Methods
- ▶ Chapter 3 : Support Vector Machines and Neural Networks
- ▶ Chapter 4 : Decision Trees
- ▶ Chapter 5 : Unsupervised Classification Methods
  - ▶ Optimization methods
  - ▶ Hierarchical classification
  - ▶ Density-based methods
  - ▶ Mixture models
  - ▶ Spectral methods

## Gaussian mixture (unsupervised learning)

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

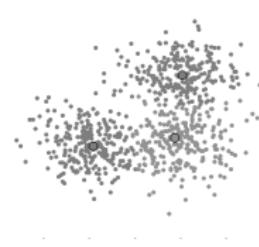
### Gaussian mixture model (GMM)

- ▶ Definition

$$p(\mathbf{x}|\theta) = \sum_{k=1}^K \pi_k p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (3)$$

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

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



## Gaussian mixture (supervised learning)

When the  $N$  data  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  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:  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$  where  $y_i \in \{\omega_1, \dots, \omega_K\}$ .

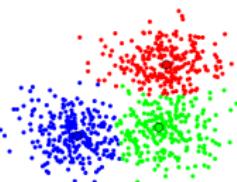
### Gaussian mixture model (GMM)

- ▶ Definition

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^K \pi_k p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (4)$$

- ▶ **Assignments:** binary variable  $\delta(k|i)$  assigns data  $\mathbf{x}_i$  to the  $k$ th Gaussian (class  $\omega_k$ ) if  $\delta(k|i) = 1$  ( $\delta(k|i) = 0$  otherwise).

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



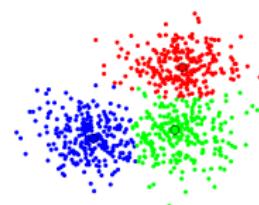
## Gaussian mixture (supervised learning)

When the  $N$  data  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  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

- ▶  $\hat{\pi}_j \leftarrow \frac{\hat{n}_j}{n}$       with       $\hat{n}_j = \sum_{i=1}^n \delta(j|i)$
- ▶  $\hat{\mu}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^n \delta(j|i) \mathbf{x}_i$
- ▶  $\hat{\Sigma}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^n \delta(j|i) (\mathbf{x}_i - \hat{\mu}_j)(\mathbf{x}_i - \hat{\mu}_j)^T$   
where  $\delta(j|i) = 1$  assigns data  $\mathbf{x}_i$  to the  $j$ th 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  $\boldsymbol{\theta}$  :  $\boldsymbol{\theta}^0 = (\pi_1^0, \dots, \pi_K^0, \boldsymbol{\mu}_1^0, \boldsymbol{\Sigma}_1^0, \dots, \boldsymbol{\mu}_K^0, \boldsymbol{\Sigma}_K^0)$  and compute

$$P(y_i = \omega_j | \mathbf{x}_i, \boldsymbol{\theta}^0) = \frac{p(\mathbf{x}_i | y_i = \omega_j, \boldsymbol{\theta}^0) P(\omega_j)}{\sum_{k=1}^K p(\mathbf{x}_i | y_i = \omega_k, \boldsymbol{\theta}^0) P(\omega_k)} \quad (5)$$

$$P(y_i = \omega_j | \mathbf{x}_i, \boldsymbol{\theta}^0) = \frac{\pi_j^0 p(\mathbf{x}_i | \boldsymbol{\mu}_j^0, \boldsymbol{\Sigma}_j^0)}{\sum_{k=1}^K \pi_k^0 p(\mathbf{x}_i | \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0)} \quad (6)$$

### Soft assignment

- ▶  $\hat{\delta}(j|i) = P(y_i = \omega_j | \mathbf{x}_i, \boldsymbol{\theta}^0)$
- ▶  $\sum_{j=1}^K \hat{\delta}(j|i) = 1.$

## Expectation-Maximization (EM-Algorithm for GMM estimation)

Without assignment (unsupervised case), we start from  $\theta^0$  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  $\theta^{(k+1)}$  increases the log-likelihood of the “completed” data.

### EM-algorithm:

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

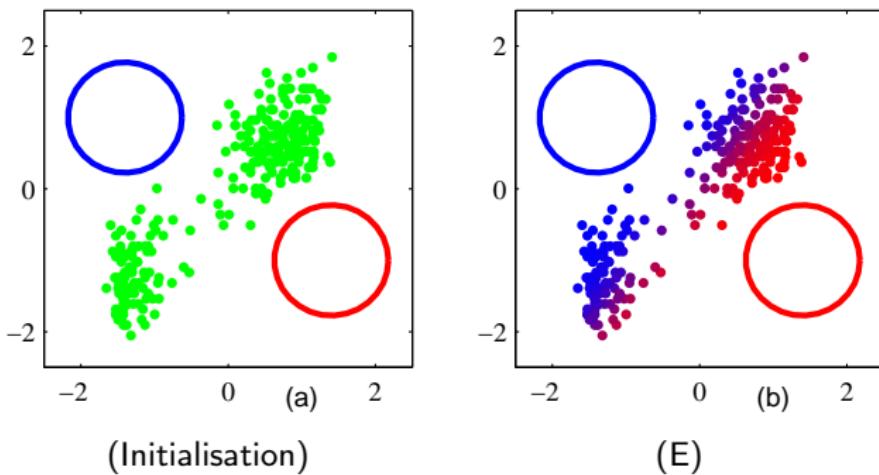
(E-step) soft-assignments of  $\mathbf{x}_i \forall i$

$$\widehat{\delta}(j|i) \leftarrow P(y_i = \omega_j | \mathbf{x}_i, \boldsymbol{\theta}^{(k)}) \quad (7)$$

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

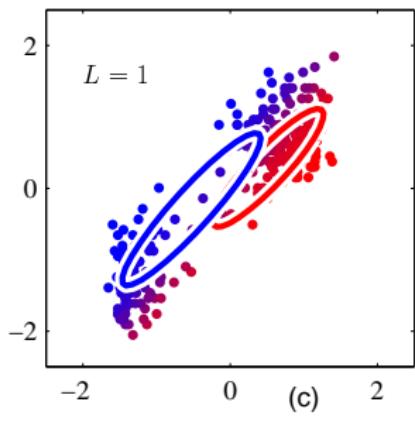
- ▶  $\widehat{\pi}_j \leftarrow \frac{\widehat{n}_j}{n}$  with  $\widehat{n}_j = \sum_{i=1}^n \widehat{\delta}(j|i)$
- ▶  $\widehat{\mu}_j \leftarrow \frac{1}{\widehat{n}_j} \sum_{i=1}^n \widehat{\delta}(j|i) \mathbf{x}_i$
- ▶  $\widehat{\Sigma}_j \leftarrow \frac{1}{\widehat{n}_j} \sum_{i=1}^n \widehat{\delta}(j|i)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

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

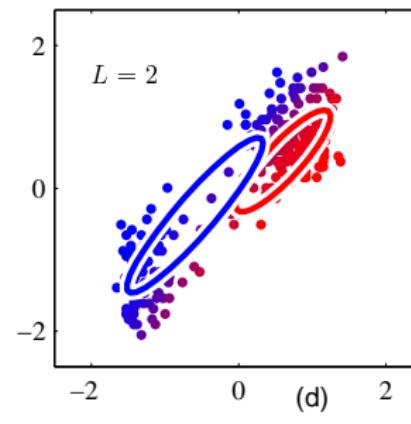


\* Christopher Bishop, *Pattern Recognition and Machine Learning*. New-York: Springer Verlag, 2006.

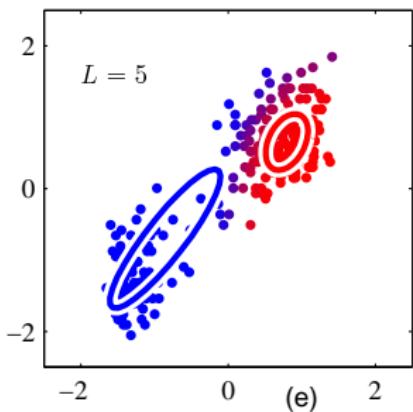
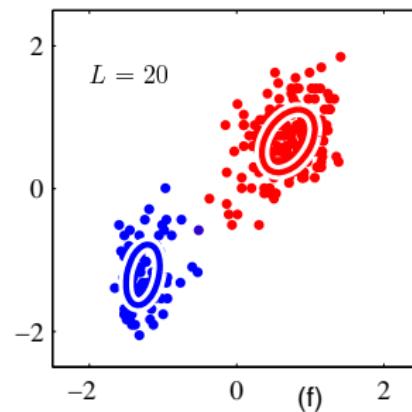
## EM-Algorithm for GMM estimation (demo 2/3)



(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  $l(\boldsymbol{\theta}^0) < l(\boldsymbol{\theta}^1) < \dots < l(\boldsymbol{\theta}^k)$  with  $l(\boldsymbol{\theta}^k) = \sum_{i=1}^n \ln p(\mathbf{x}_i | \boldsymbol{\theta}^k)$

## Mélange de lois gaussiennes généralisées asymétriques (AGGD)

Estimation des paramètres du mélange

Sélection de modèle

Conclusions et perspectives

## Mélange de lois AGGD

## Vraisemblance du mélange

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

avec

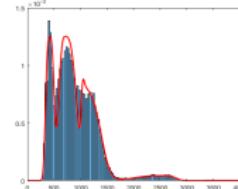
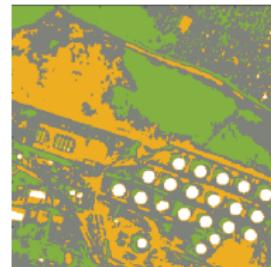
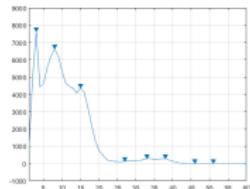
$$p(x|\theta_j) = \begin{cases} \frac{\delta_j^{1/\lambda_j}}{\gamma_j^{1/\lambda_j} \Gamma(1+1/\lambda_j)} \exp\left(-\frac{\delta_j}{\gamma_j} \left(\frac{\mu_j - x}{\alpha_j}\right)^{\lambda_j}\right) & \text{si } x < \mu_j \\ \frac{\delta_j^{1/\lambda_j}}{\gamma_j^{1/\lambda_j} \Gamma(1+1/\lambda_j)}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

- $\alpha_j$  : paramètre d'**asymétrie**
- $\mu_j$  : paramètre de **décalage**
- $\gamma_j$  : paramètre d'**échelle**
- $\lambda_j$  : 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

Mélange de lois gaussiennes généralisées asymétriques (AGGD)  
Estimation des paramètres du mélange  
**Sélection de modèle**  
Conclusions et perspectives

Algorithme EMB + Minimum message length  
Fusion de modes proches

## Algorithme 2



19 / 78

## Examples of Applications

### Bayes classifier

- ▶ Document classification
- ▶ Detection of SPAMS in emails

### SVM

- ▶ Face detection
- ▶ Object detection and recognition

### Neural networks

- ▶ Image recognition
- ▶ Natural language processing
- ▶ Object detection and recognition

### CART and Random Forests

- ▶ Medical applications