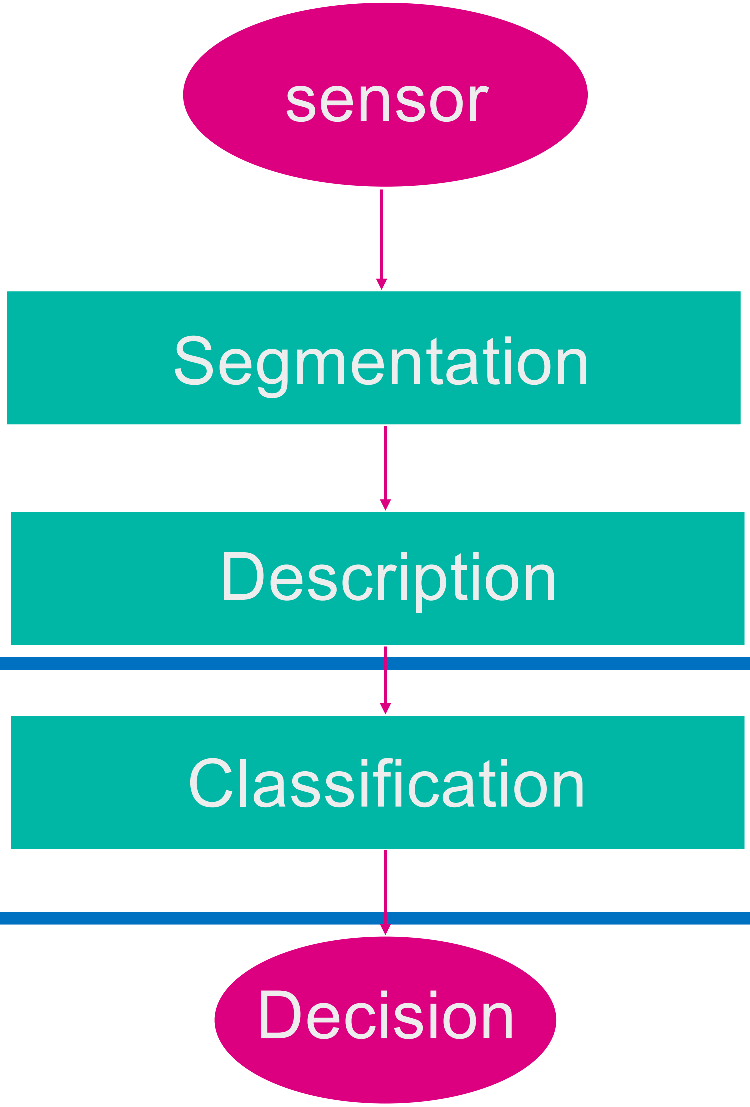
Image analysis and Pattern Recognition

Lecture 5 : Statistical shape classification

Prof. Jean-Philippe THIRAN
JP.Thiran@epfl.ch

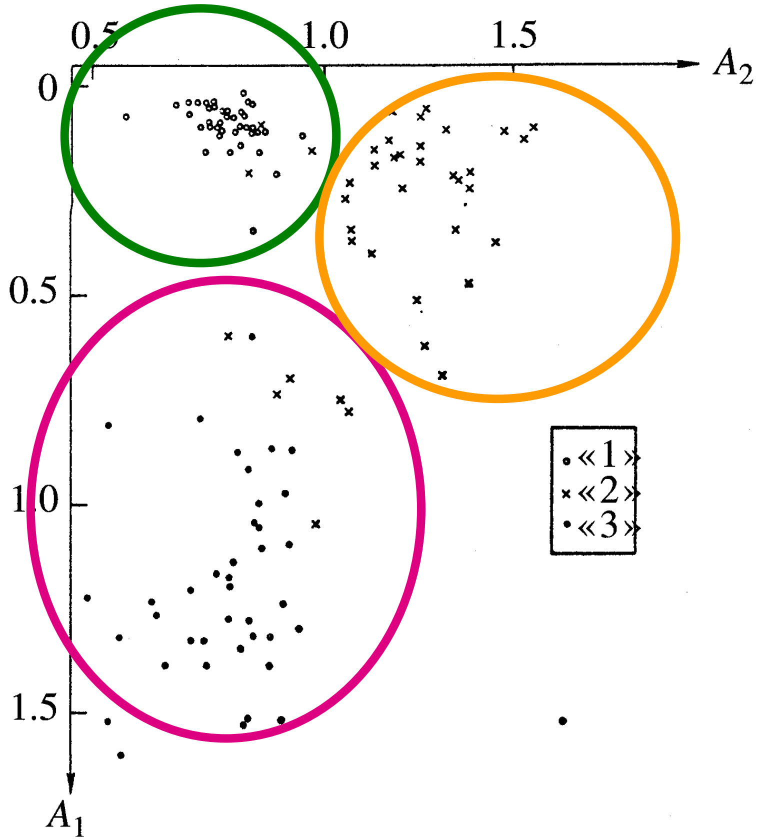
The Pattern Recognition System



- Electricity, electronics, metrology
- Signal Processing
- Image Processing
- Geometry, topology, . . .
- Statistics
  - AI
- Robotics, …

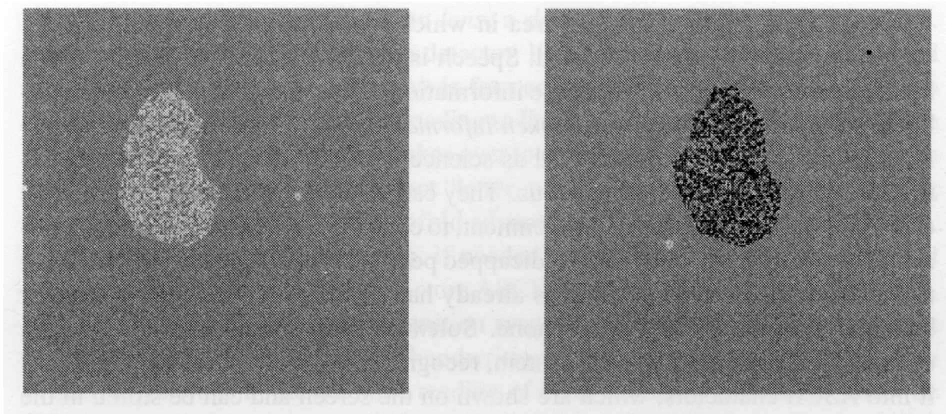
Shape classification

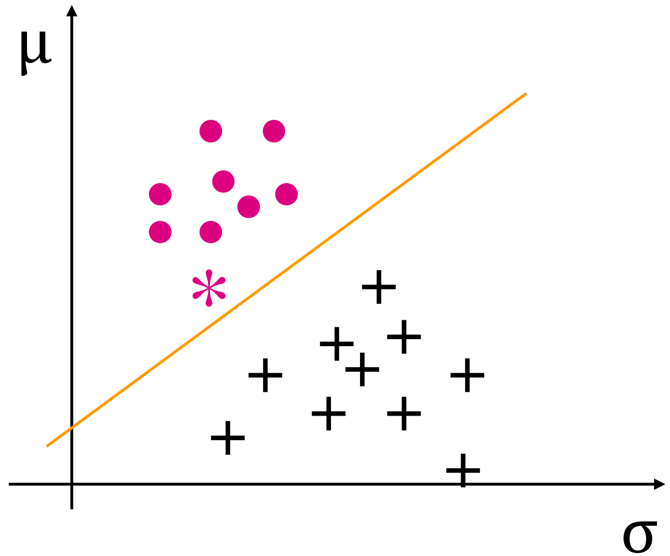
- Pattern recognition means:
  - Choose good descriptors for your application => feature vector
  - Use a classification rule to classify the feature vectors



Shape classification

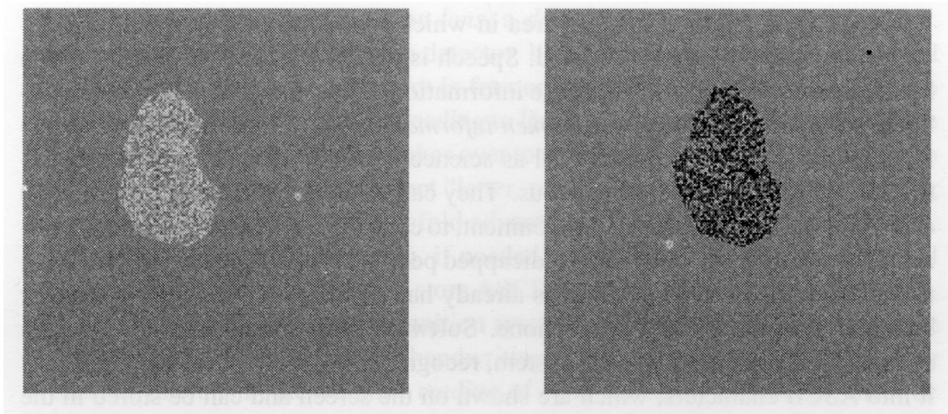
- Another example: cancerous cells
  - Assume that we have a set of data already classified
  - Let us chose 2 features : the mean μ and the standard deviation σ of the gray levels
  - A new data point (*) has to be classified
- Generalisation:
  - Feature vecotr $x=[x_1, x_2, \ldots x_n]^T$
  - Role of the classifier: assign a class label to a feature vector
  - Separation of the classes in the feature space: decision line

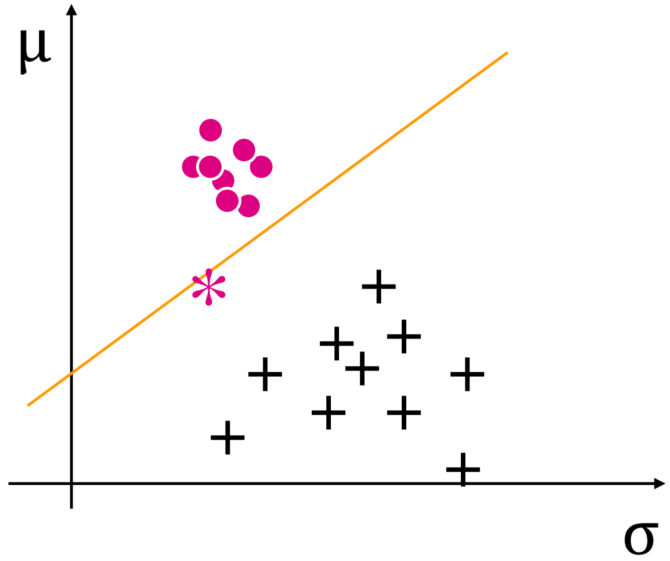




Shape classification

- Another example: cancerous cells
  - Assume that we have a set of data already classified
  - Let us chose 2 features : the mean μ and the standard deviation σ of the gray levels
  - A new data point (*) has to be classified
- Generalisation:
  - Feature vecotr $x=[x_1, x_2, \ldots x_n]^T$
  - Role of the classifier: assign a class label to a feature vector
  - Separation of the classes in the feature space: decision line





Supervised vs unsupervised classifiers

- Supervised classifiers :
  - We have a training set, i.e. a set of feature vectors with their correct class label
  - We have to build a classifiers that exploits this prior information
  - *Example : Optical Character Recognition*
  - *Bayesian Classifiers*
  - *Linear Classifiers*
  - *Non-linear classifiers – neural networks*
- Unsupervised classifiers
  - We just have a set of feature vectors, without their class label
  - We have to group similar vectors to create clusters, and identify those clusters
  - Clustering algorithms

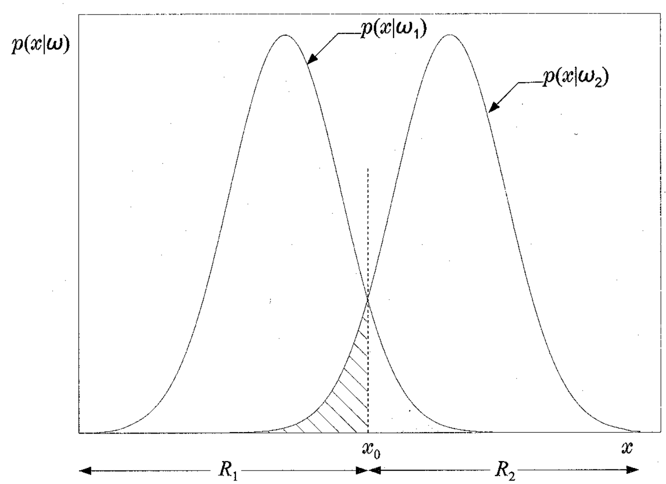
- Probabilistic approach:
  - Feature vectors are assumed to come from a probability distribution function (*pdf*)
  - We will design a classifier which will assign a feature vector to the « most probable » class:
    - *M classes* $w_1, w_2, \dots w_M$
    - *A feature vector* x
    - *We classify* x *in class* i *if* $P(w_i \mid x) > P(w_j \mid x)$

- Let us consider the two class problem, with known prior probabilities $P(w_1)$ and $P(w_2)$
  - Easy to evaluate of not known
- The conditional *pdf*s $p(x \mid w_i)$ are also assumed known
  - Can be identified from the training set
- Bayes Rule: $$P(w_i \mid x) = \frac{p(x \mid w_i)\, P(w_i)}{p(x)}$$
- Bayesian classification (maximum a posteriori)

$$P(w_1 \mid x) \overset{?}{<>} P(w_2 \mid x)$$

$$p(x \mid w_1) P(w_1) \overset{?}{<>} p(x \mid w_2) P(w_2)$$

- Example : 1 feature, 2 classes
  - x_0 indicated the separation between the classes
  - There is obviously a classification error
  - But it can be shown that the Bayesian classifier minimises the classification errors



- Generalization to *n* classes
  - Assignation to the most probable class
  - The decision surface between classes i and j has the equation $P(w_i \mid x) - P(w_j \mid x) = 0$
  - We can also write it as follows: $g_i(x) \equiv f(P(w_i \mid x))$

    where $f(.)$ is a monotonally increasing function, called discriminant function.
  - Decision will thus be taken to assign the feature vector to class w_i if $g_i(x) > g_j(x)$ *for all* $j \neq i$
  - The decision surface is given by

$$g_{ij}(x) \equiv g_i(x) - g_j(x) = 0$$

- Normal law : the pdf follows a Gaussian law:
  - 1D: $$p(x \mid w_i) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$
  - *l*D : $$p(x \mid w_i) = \frac{1}{(2\pi)^{l/2} \left|\Sigma_i\right|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i)}$$
    - *μ_i is the mean of class w_i*
    - *Σ_i is the covariance matrix of size l x l, defined by*

$$\Sigma_i = E\left[(x-\mu_i)(x-\mu_i)^T\right]$$

- Discriminant function:

$$
\begin{aligned}
g_i(x) &= \ln(p(x \mid w_i)P(w_i)) = \ln p(x \mid w_i) + \ln P(w_i) \\
&= -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) + \ln P(w_i) + c_i \\
&= -\frac{1}{2}x^T\Sigma_i^{-1}x + \frac{1}{2}x^T\Sigma_i^{-1}\mu_i - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i + \frac{1}{2}\mu_i^T\Sigma_i^{-1}x + \ln P(w_i) + c_i
\end{aligned}
$$

- Example :

si $l = 2$ et $\Sigma_i = \begin{pmatrix} \sigma_i^2 & 0 \\ 0 & \sigma_i^2 \end{pmatrix}$, we have

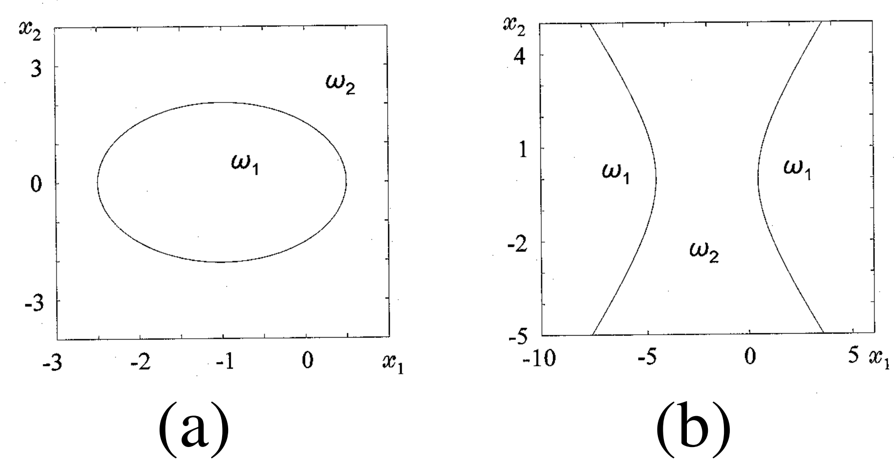
$$g_i(x) = -\frac{1}{2\sigma_i^2}(x_1^2 + x_2^2) + \frac{1}{\sigma_i^2}(\mu_{i1}x_1 + \mu_{i2}x_2) - \frac{1}{2\sigma_i^2}(\mu_1^2 + \mu_2^2) + \ln P(w_i) + c_i$$

and the decision curves $g_i(x) - g_j(x) = 0$ are (hyper)quadrics

Example : $\mu_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

$(a) \quad \Sigma_1 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.15 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.25 \end{pmatrix}$

$(b) \quad \Sigma_1 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.15 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.15 & 0 \\ 0 & 0.1 \end{pmatrix}$



- Special case, very frequent : Σ_i identical for all classes: $\Sigma_i = \Sigma$
  - The quadratic terms will disappear in the equation of the decision curves, as well as the constant c_i
  - Thus the discriminant function can be written as :

$$g_i(x) = w_i^T x + w_{i0}$$

$$\text{with} \quad w_i = \Sigma^{-1}\mu_i \quad \text{and} \quad w_{i0} = \ln P(w_i) - \frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i$$

  - Therefore the discriminant functions are linear and the decision curves (surfaces) hyperplanes

- Sub-particular case 1: Σ diagonal with equal values on the diagonal: $\Sigma = \sigma^2 I$
  - The discriminant functions become

$$g_i(x) = \frac{1}{\sigma^2} \mu_i^T x + w_{i0}$$

  - And the decision hyperplanes are

$$g_{ij}(x) \equiv g_i(x) - g_j(x) = w^T (x - x_0) = 0$$

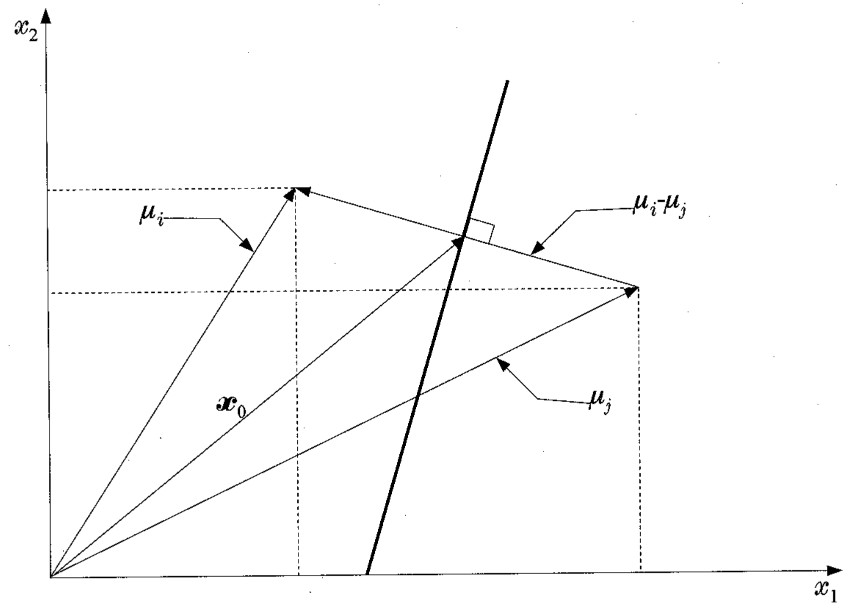
with $w = \mu_i - \mu_j$ and $x_0 = \frac{1}{2}(\mu_i + \mu_j) - \sigma^2 \ln\left(\frac{P(w_i)}{P(w_j)}\right) \frac{\mu_i - \mu_j}{\|\mu_i - \mu_j\|^2}$

Bayesian classification for normal laws

$$g_{ij}(x) \equiv g_i(x) - g_j(x) = w^T(x - x_0) = 0$$

$$\text{with } w = \mu_i - \mu_j \quad \text{and} \quad x_0 = \frac{1}{2}(\mu_i + \mu_j) - \sigma^2 \ln\left(\frac{P(w_i)}{P(w_j)}\right)\frac{\mu_i - \mu_j}{\|\mu_i - \mu_j\|^2}$$

- Thus
  - The decision hyper plane passes by x_0
  - if $P(w_1)=P(w_2)$, $x0=(\mu_1+\mu_2)/2$
  - Since moreover, for every x on the decision hyperplane, $x-x_0$ is also on the hyperplane, and since $(\mu_1-\mu_2)^T(x-x_0)=0$, *the decision hyperplane is orthogonal to* $\mu_1-\mu_2$



Bayesian classification for normal laws

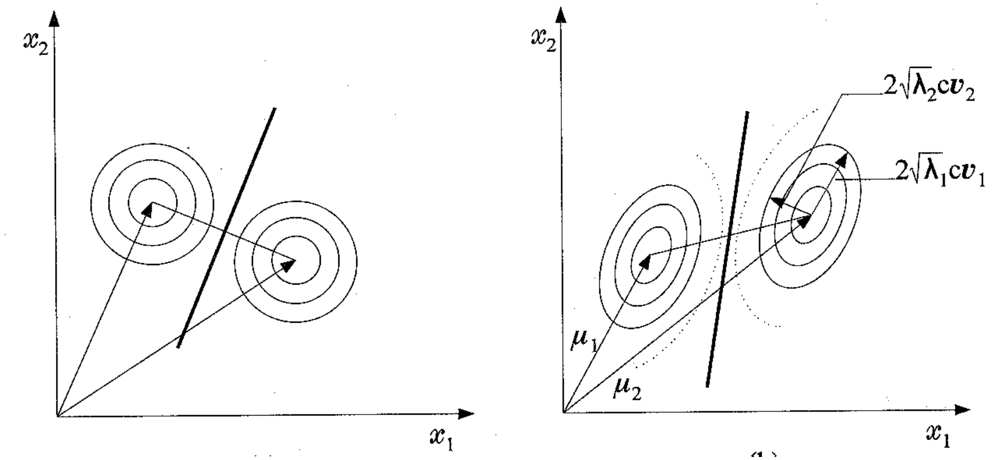
- If Σ is different from $\sigma^2 I$:

$$g_{ij}(x) \equiv g_i(x) - g_j(x) = w^T(x - x_0) = 0$$

$$\text{with } w = \Sigma^{-1}(\mu_i - \mu_j) \quad \text{and} \quad x_0 = \frac{1}{2}(\mu_i + \mu_j) - \ln\left(\frac{P(w_i)}{P(w_j)}\right)\frac{\mu_i - \mu_j}{\left\|\mu_i - \mu_j\right\|^2_{\Sigma^{-1}}}$$

$$\left\|\mu_i - \mu_j\right\|^2_{\Sigma^{-1}} = (x^T \Sigma^{-1} x)^{1/2}$$

- Thus:
  - The decision hyperplane passes by x_0
  - if $P(w_1)=P(w_2)$, $x0=(\mu_1+\mu_2)/2$
  - *The decision hyperplane is not orthogonal to μ_1-μ_2, but to a linear transformation of it:$\Sigma^{-1}(\mu_1$-$\mu_2)$*



- Minimal distance classifier :
  - If we neglect the constants, we have

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)$$

- if $\Sigma = \sigma^2 I$: *the most probable class is the one that maximizes $g_i(x)$, i.e. which minimizes the Euclidean distance*

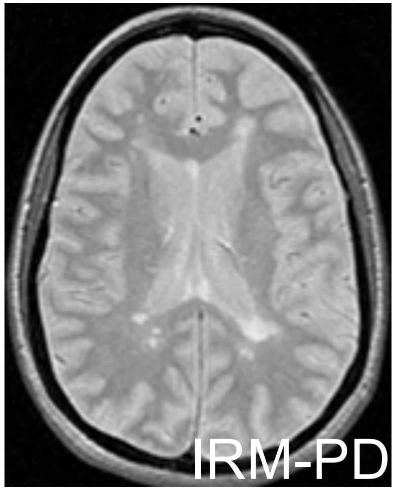
$$d_e = \|x-\mu_i\|$$

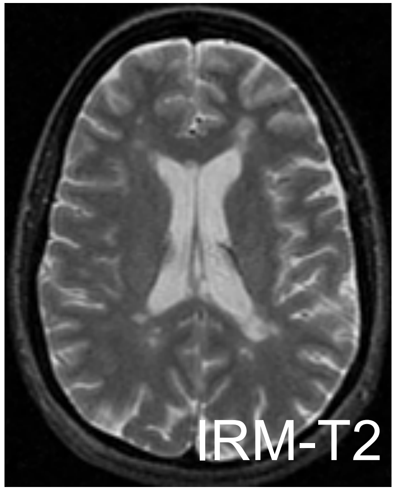
- If Σ is not diagonal: *the most probable class is the one that maximizes $g_i(x)$, i.e. which minimizes the Mahalanobis distance*

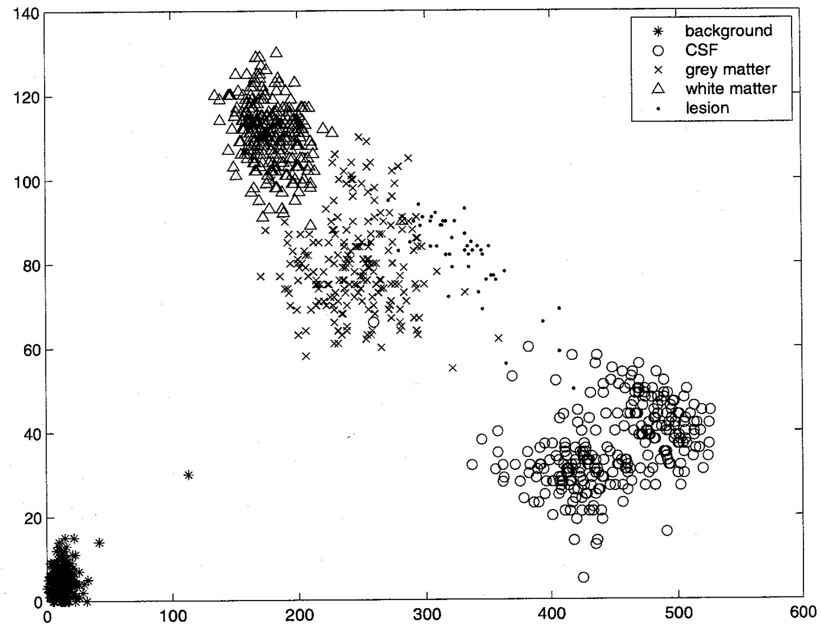
$$d_m = \left((x-\mu_i)\Sigma^{-1}(x-\mu_i)\right)^{1/2}$$

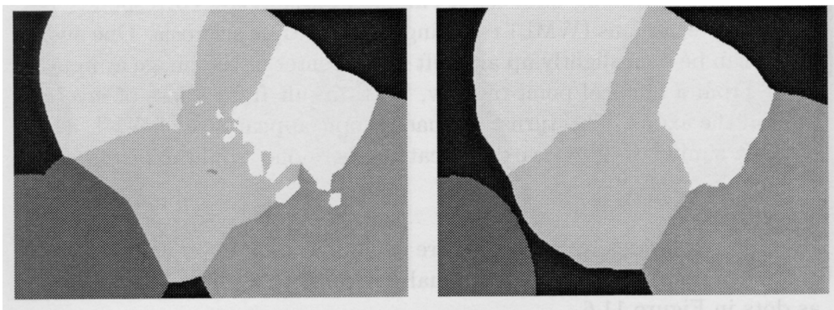
- This algorithm does not make any assumption on the class pdfs
- *k-NN* algorithm:
  - We have a training set of feature vectors with their class label
  - We classify the unknown vector x in the most represented class among the *k* nearest neighbors of x
  - The error probability *R* is at least as large as the Bayesian one (*Pe*)
    - *1-NN : R < 2Pe*
    - *k-NN : R < (1+1/k) Pe*

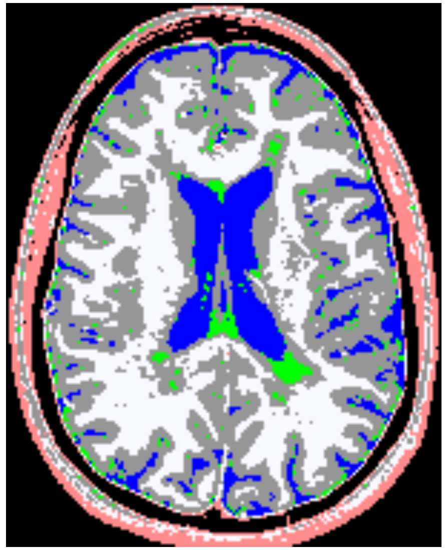
- Example:


IRM-PD


IRM-T2


background
CSF
grey matter
white matter
lesion
0
20
40
60
80
100
120
140
0
100
200
300
400
500
600





- Let us consider again the case of a linear discriminant function. The decision surfaces are hyper planes:

$$g_{ij}(x) \equiv g_i(x) - g_j(x) = w^T x - w_0 = 0$$

  - w is called the weight vector and w_0 the threshold
  - w is orthogonal to the decision surface
  - We can also write $w'^T x' = 0,\ \text{with } w' = [w^T, -w_0]^T \text{ and } x' = [x^T, 1]^T$
  - Without any additional information, we can try to find the best vector $w*$ which best separates the classes, i.e. such that

$$w*^T x > 0 \quad \forall x \in \omega_1$$

$$w*^T x < 0 \quad \forall x \in \omega_2$$

- Optimization process :
  - Search space : space of w
  - Cost function: $$J(w) = \sum_{x\ mal\ classifiés} (\delta_x w^T x)$$

$$\delta_x = -1 \text{ si } x \in \omega_1, \quad \delta_x = +1 \text{ si } x \in \omega_2$$

$$\text{On observe que } J(w) \geq 0$$

  - Optimization algorithm : gradient descent :

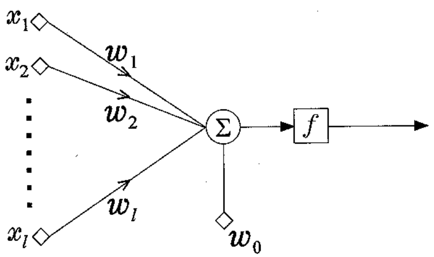
$$w(t+1) = w(t) - \rho_t \left. \frac{\partial J(w)}{\partial w} \right|_{w=w(t)} \quad \text{with} \quad \frac{\partial J(w)}{\partial w} = \sum_{x\ mal\ classifiés} \delta_x x$$

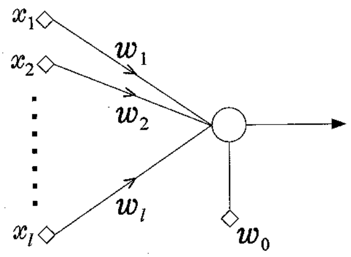
$$\boxed{w(t+1) = w(t) - \rho_t \sum_{misclassified\ x} \delta_x x}$$

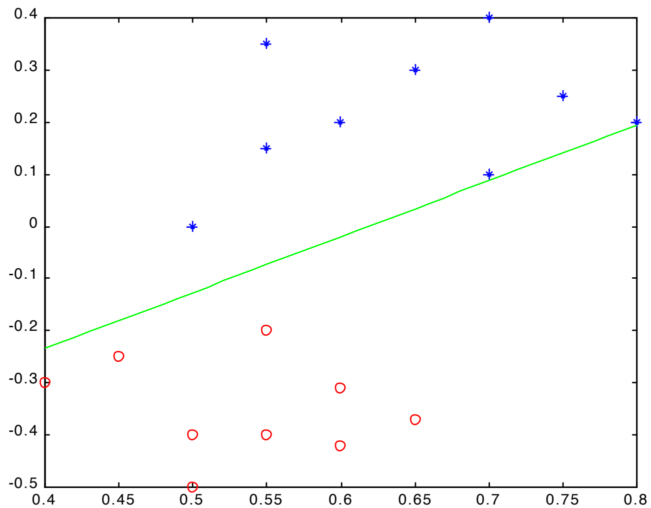
Linear classifier: the perceptron

$$w(t+1) = w(t) - \rho_t \sum_{misclassified\ x} \delta_x x$$

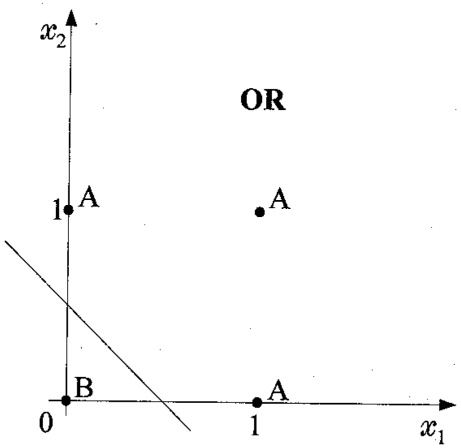
- ρ_t is a critical parameter for the convergence
  - Should be large at the beginning, to correctly drive the convergence
  - Should become small later on, to smoothly converge
  - Example : $\rho_t = cst/t$

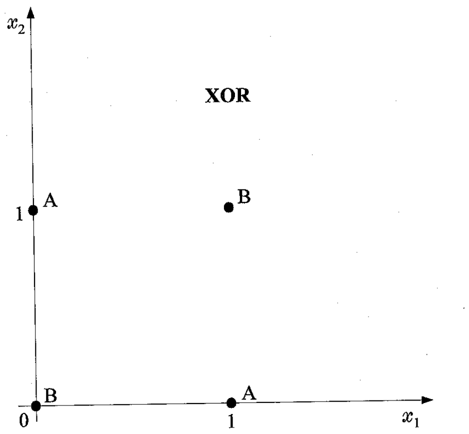




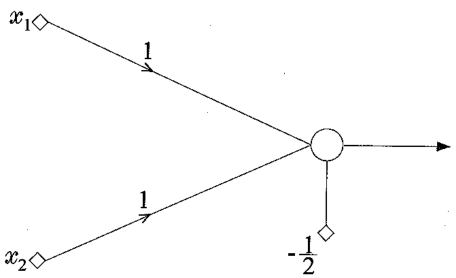


- Let us take a simple example : the XOR function, which is not linearly separable, contrary to AND and to OR



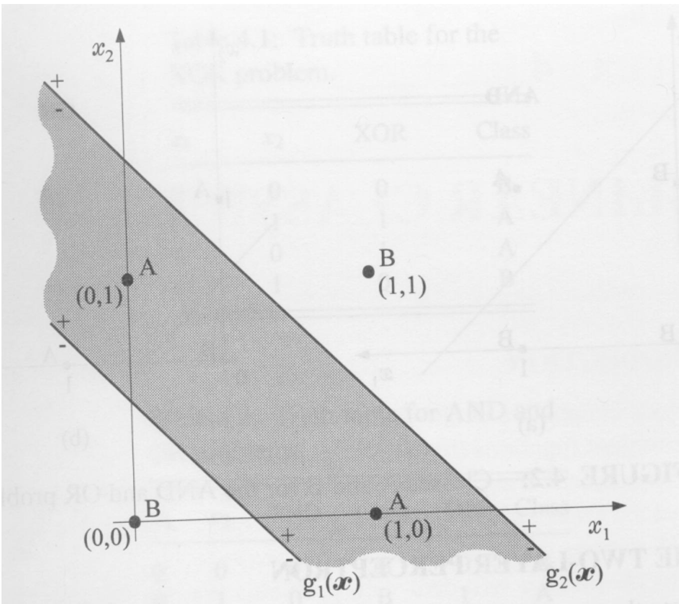


x1	x2	OR	XOR
0	0	0 (B)	0 (B)
0	1	1 (A)	1 (A)
1	0	1 (A)	1 (A)
1	1	1 (A)	0 (B)

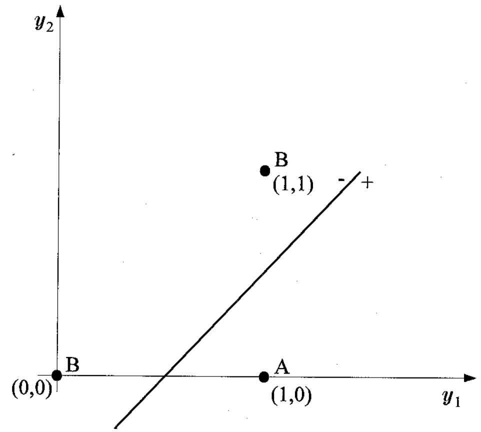


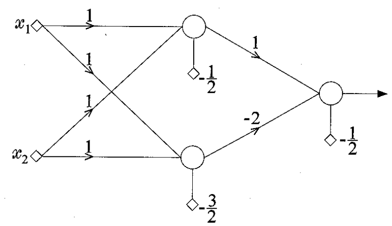
Non-linear classifiers: multi-layer perceptrons

- For XOR : We can consider two decision lines
  - We consider where *x* is w.r.t *g1* and *g2*
  - We consider the combination of the two decision to take the final decision
- Thus 2 linear steps : 2-layer perceptron

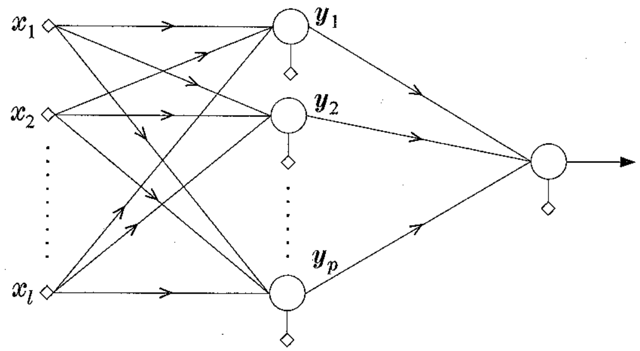


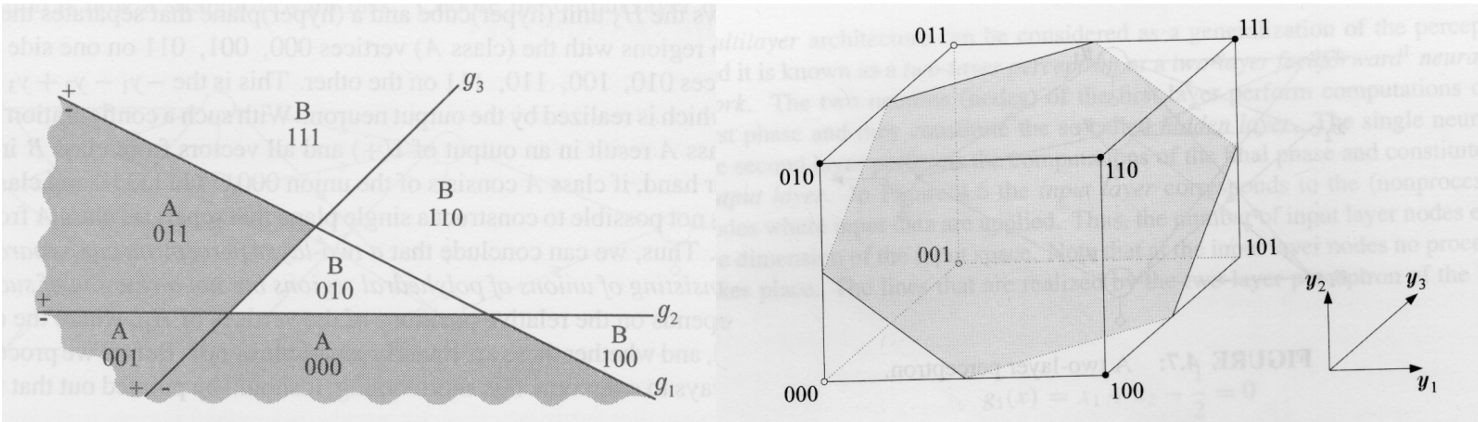
x1	x2	y1	y2	Cla.
0	0	0(-)	0(-)	B(0)
0	1	1(+)	0(-)	A(1)
1	0	1(+)	0(-)	A(1)
1	1	1(+)	1(+)	B(0)





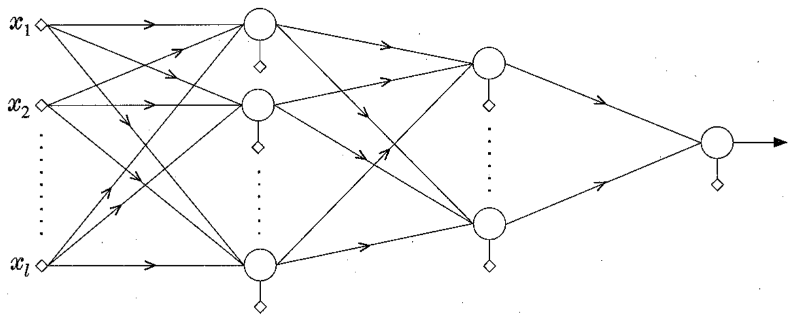
- We can generalize: perceptron with l inputs and p « hidden » neurons realizes two successive classifications
  - One towards the summits of an hypercube in the p-dimensional space
  - One which separates this cube in 2 semi-spaces by an hyperplane

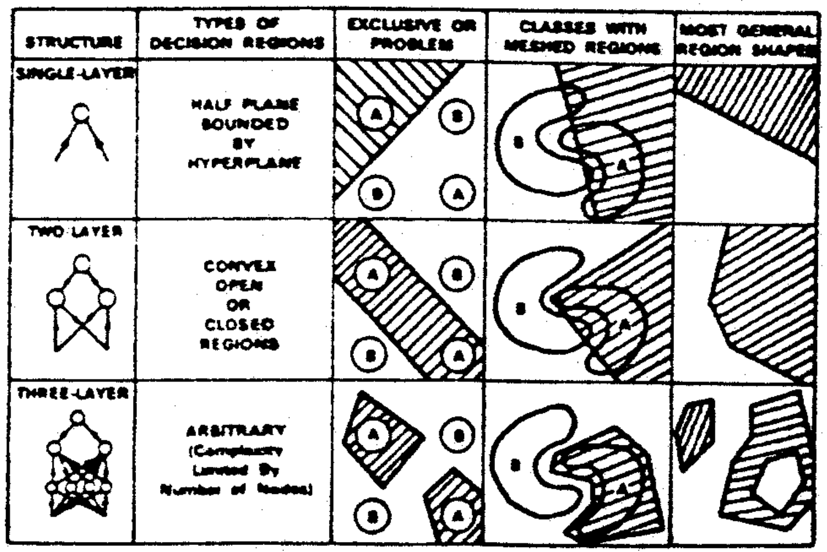




  - A 2-layer perceptron can thus separate classes that are the union of polyhedra (not any union though)

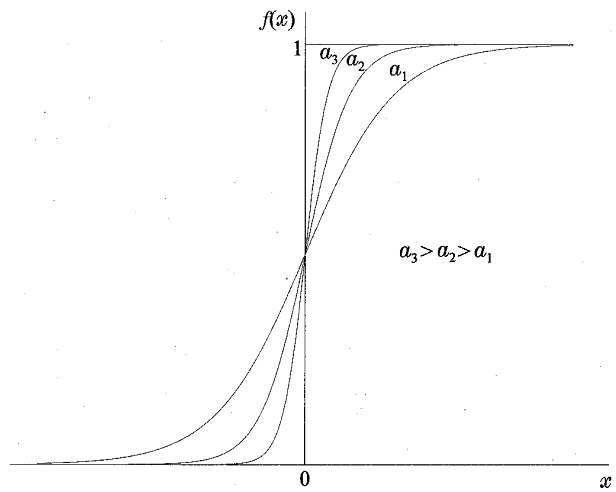
- Solution : 3-layer perceptron

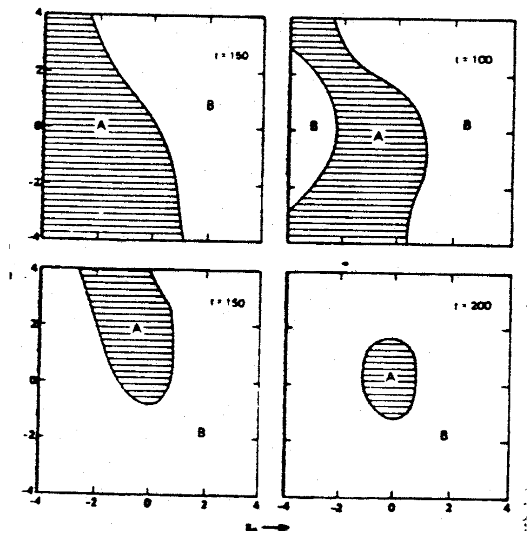




- Until now, the decision function used was a step function (0 or 1)
  - Decision surfaces are hyper planes
- But this is a problem for training, which is an optimization of a cost function
  - Which implies a derivation of the cost function.
  - But the step function is not derivable

- We can thus consider a sigmoïd and not a step function, and the decision function will become curves. The decision will consider the output neuron with the maximal answer
  - There are efficient training algorithms, based on the error backpropagation
  - Cfr article R. Lippmann, IEEE Acoustics, Speech and Signal Processing Magasine, Avril 1987.



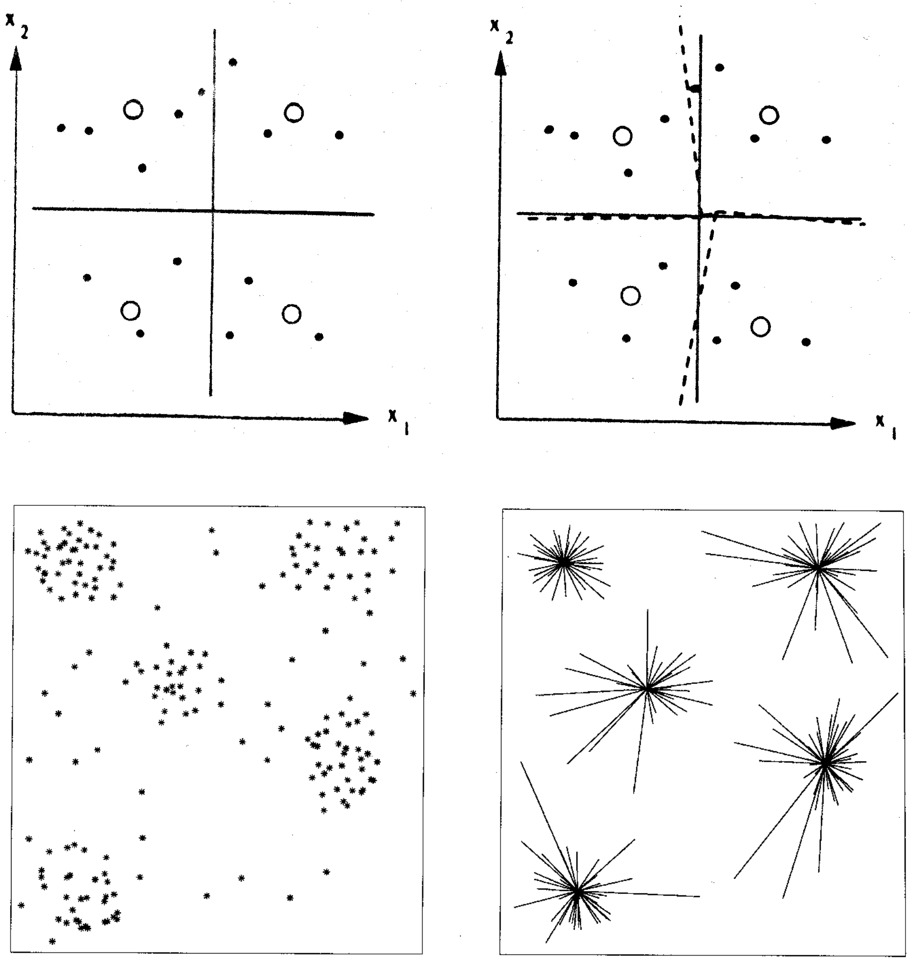


- We have non-classified training samples, we can do an unsupervised training of a classifier
  - The samples live in their feature space
  - We try to identify regions of high sample density, which model the sample probability distribution function: approximation by Gaussian laws
  - This is called *clustering*

Unsupervised training: clustering (k-means)

- We try to identify *m* classes by means of their centers, called *centroids*
- Objective : minimize the intra-class variance
- Algorithm : ISODATA (k-means)
  - Choose *m* centroids randomly
  - iterate
    - *Attach each vector x to the class of the closes centroid*
    - *Recalculate the position of the centroids as the means of the vector of each class*
  - Until convergence

- Examples



General conclusion

- Pattern recognition involved several steps
  - Object segmentation
    - *Image analysis*
  - Feature extraction
    - *Geometry, invariants, etc.*
  - Classification
    - *AI, supervised or unsupervised classification, neural networks*
- Very various applications
- Many different methods, but the basic principles are very stable