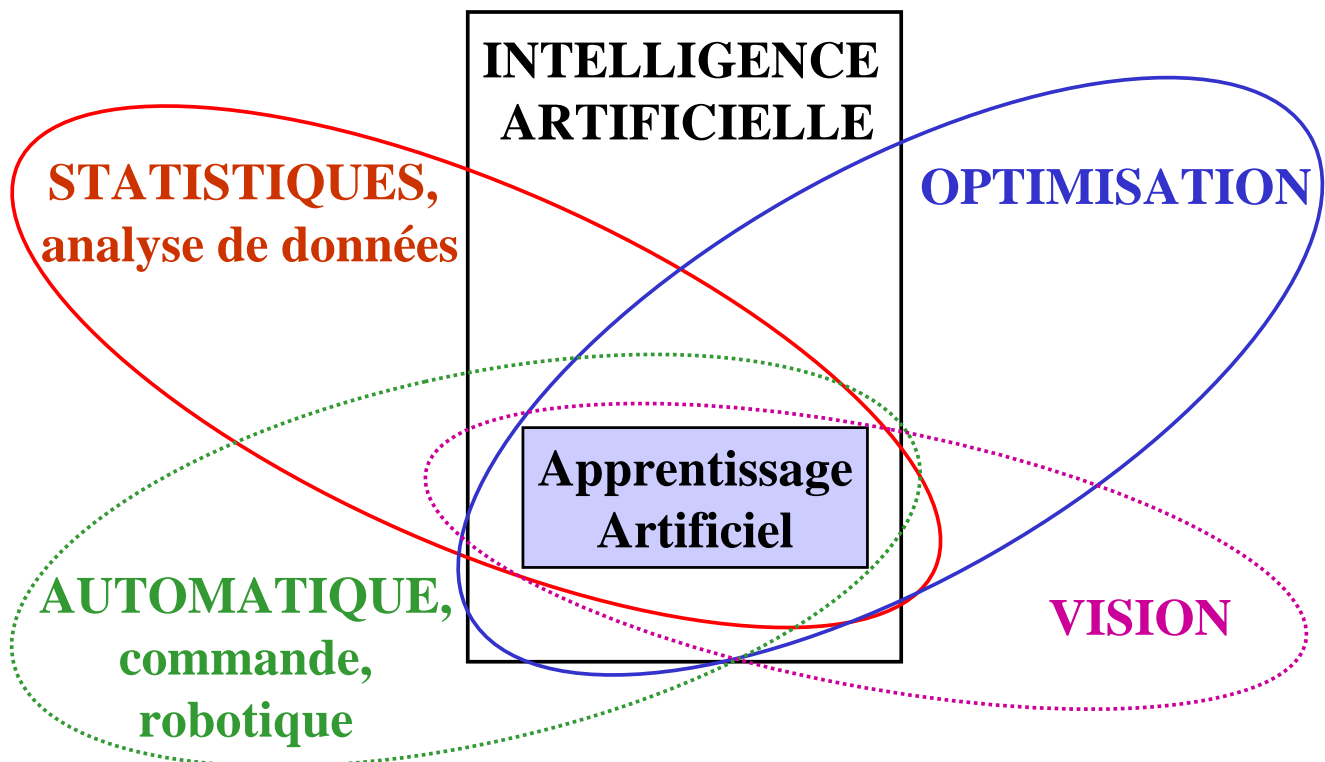


# APPRENTISSAGE ARTIFICIEL (« Machine-Learning »)

Fabien Moutarde  
Centre de Robotique (CAOR)  
MINES ParisTech (Ecole des Mines de Paris)

Fabien.Moutarde@mines-paristech.fr  
<http://www.mines-paristech.fr/~moutarde>

## Un domaine interdisciplinaire

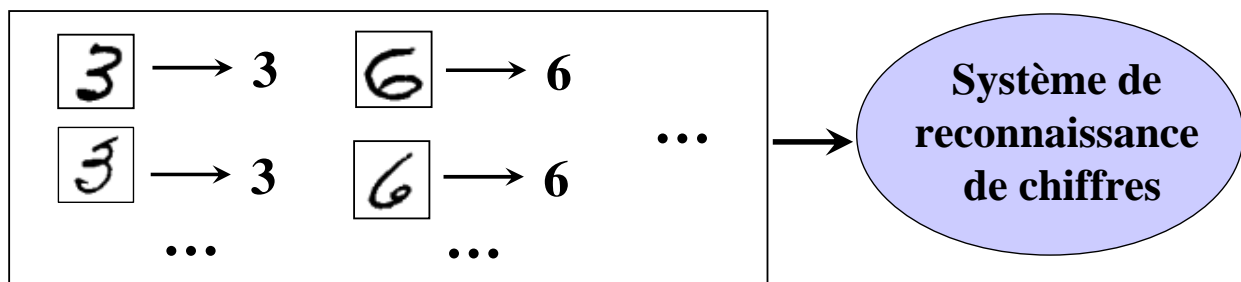


« Capacité d'un système à  
améliorer ses performances via  
des interactions avec son environnement »

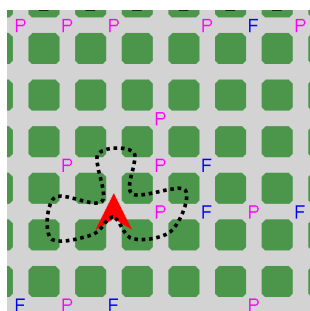
Une des familles essentielles de techniques pour  
l'Intelligence Artificielle (IA) : permet  
conception et/ou adaptation automatisée du modèle  
et/ou du comportement d'agents « intelligents »

## Exemples introductifs

- **Reconnaissance de caractères**

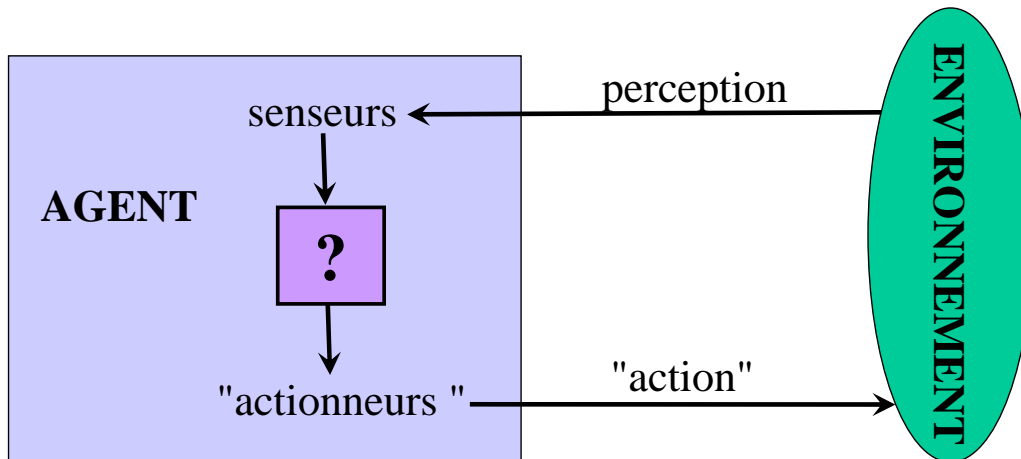


- **Comportement d'un « robot » autonome**



Navigation « à vue » optimisant  
accomplissement d'une tâche  
(e.g., collecter de la « nourriture »)

## • Modèle général pour l'IA : « agent intelligent »



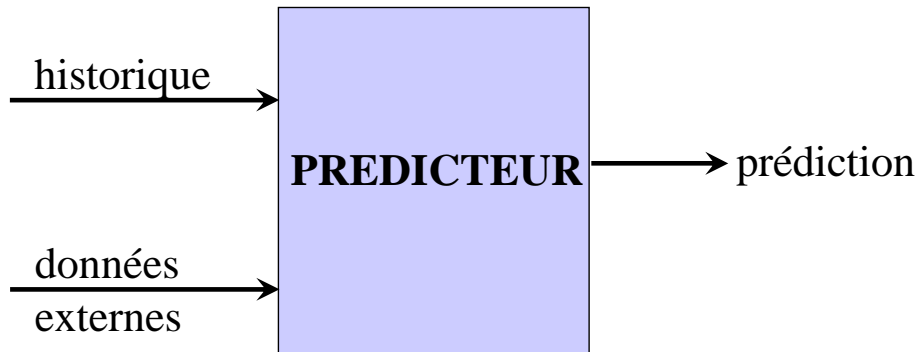
### NOTES :

1. « action » à comprendre AU SENS LARGE (par exemple ça peut être « fournir un diagnostic »)
2. Boucle Agent/Environnement pas nécessairement fermée

# Spécificité de l'apprentissage

**Conception et/ou adaptation de l'agent**  
**par analyse automatisée (généralement statistique)**  
**de son environnement, et/ou du résultat**  
**de son action dans celui-ci.**

- **Agent « prédicteur »**



- **Performance visée : minimiser erreur de prédiction**
- **Moyen mis en œuvre :**  
 utiliser des données expérimentales pour trouver un modèle  $\text{prédiction} = f(\text{historique}, \text{données externes})$  le plus correct possible

**« Capacité d'un système à améliorer ses performances via des interactions avec son environnement »**

- **Quel « système » ?**  
 → types de modèle (Ad hoc ? Issu d'une famille particulière de fonctions mathématiques [tq splines, arbre de décision, réseau de neurones, arbre d'expression, machine à noyau...] ?)
- **Quelles « interactions avec l'environnement » ?**  
 → apprentissage « hors-ligne » v.s. « en-ligne »  
 → apprentissage « supervisé » ou non, « par renforcement »
- **Quelles « performances » ?**  
 → fonction de coût, objectif, critère implicite, ...
- **Comment améliorer ?**  
 → type d'algorithme (gradient, résolution exacte problème quadratique, heuristique, ...)

## Chaque paradigme se caractérise par :

Un modèle, le plus souvent paramétrique

+

Une façon d'interagir avec l'environnement

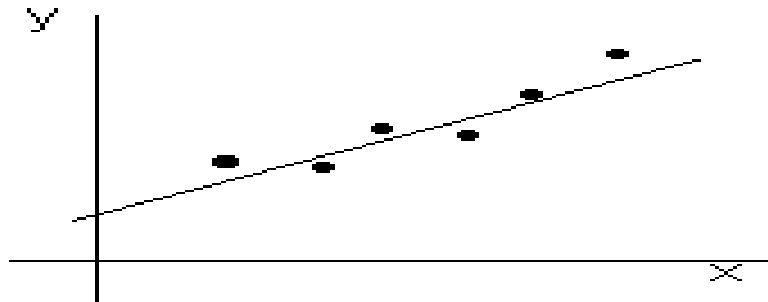
+

Une « fonction de coût » à minimiser (sauf exceptions)

+

Un algorithme pour adapter le modèle,  
en utilisant les données issues de l'environnement,  
de façon à optimiser la fonction de coût

## Exemple trivial : régression linéaire par moindres carrés



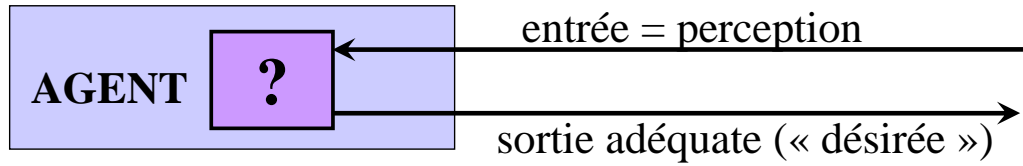
- **Modèle** : droite  $y=ax+b$  (2 paramètres  $a$  et  $b$ )
- **Interaction** : collecte *préalable* de  $n$  points  $(x_i, y_i) \in \mathcal{R}^2$
- **Fonction de coût** : somme des carrés des écarts à la droite  $K = \sum_i (y_i - a \cdot x_i - b)^2$
- **Algorithme** : résolution directe (ou itérative) du système linéaire

$$\begin{pmatrix} \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & n \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n y_i \end{pmatrix}$$

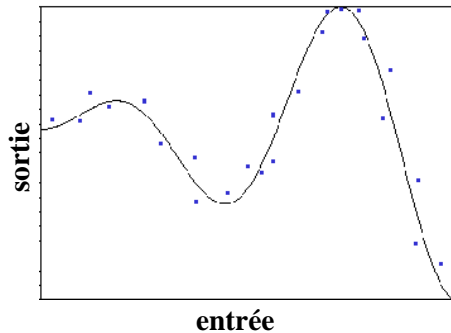
- **Régression linéaire par moindres carrés**
- **Algo ID3 ou CART pour arbres de décision**
- **Méthodes probabilistes**
- ...
  
- **Rétropropagation du gradient sur réseau neuronal à couches**
- **Cartes topologiques de Kohonen**
- **Support Vector Machines**
- **Boosting de classifieurs faibles**
- ...

- **Résolution système linéaire** (régression, Kalman, ...)
- **Algos classiques d'optimisation**
  - Descente de gradient, gradient conjugué, ...
  - Optimisation sous contrainte
  - ...
- **Heuristiques diverses :**
  - Algo d'auto-organisation non supervisée de Kohonen
  - Algorithmes évolutionnistes (GA, GP, ...)
  - « colonies de fourmis » (Ant Colony Optimization)
  - Optimisation par Essaim Particulaire (OEP)
  - Renforcement (Q-learning, ...)

Environnement → « exemples » de type (entrée, sortie)

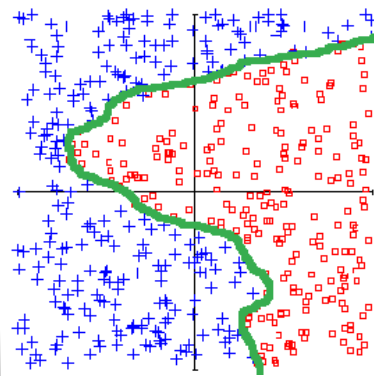


## Régression (approximation)



points = exemples → courbe = régression

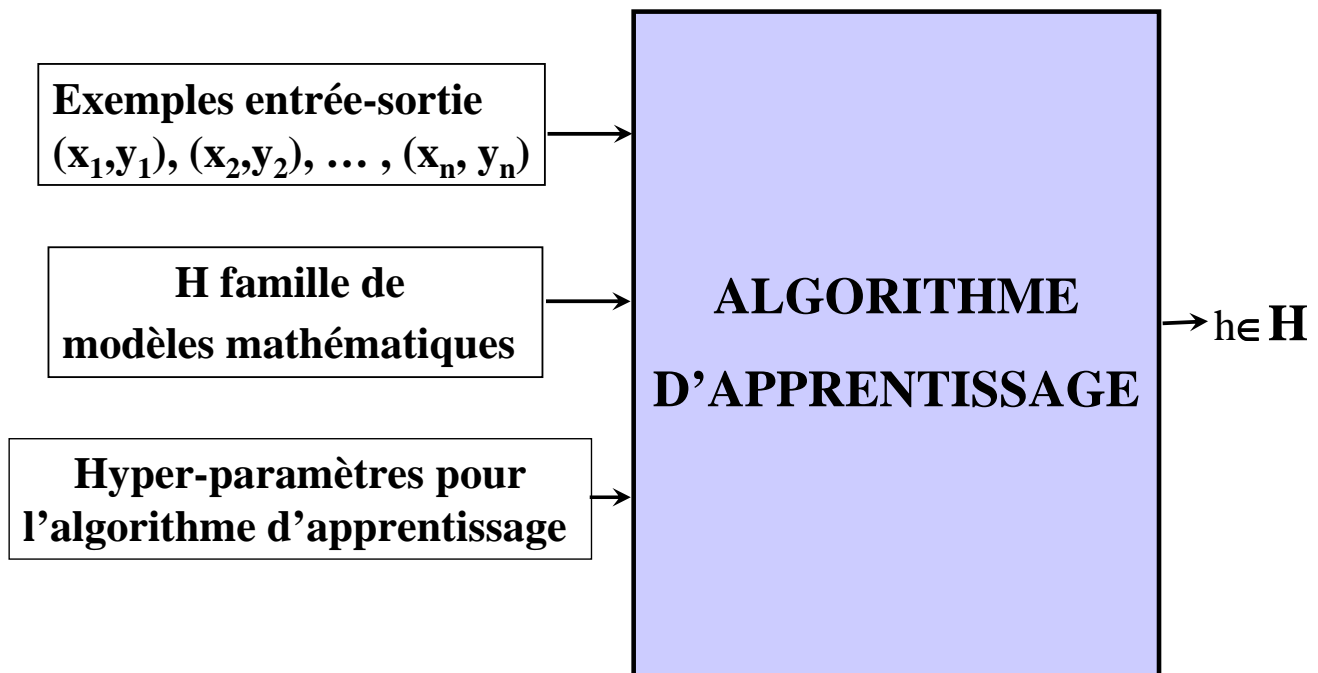
## Classification ( $y_i =$ « étiquettes »)



entrée =  
position point  
sortie désirée =  
classe ( $\square = -1, + = +1$ )

↓  
Fonction  
étiquette =  $f(x)$   
(et frontière de  
séparation)

## Apprentissage *supervisé*



- Mesure de la qualité du modèle h :

$$E(h) = E(L(h(x), y))$$

où  $L(h(x), y)$  est la « fonction de perte »  
généralement  $= \|h(x) - y\|^2$

- Divers optima possibles

$$h^* \text{ optimum « absolu »} = \operatorname{argMin}_h (E(h))$$

$$h^*_H \text{ optimum dans } H = \operatorname{argMin}_{h \in H} (E(h))$$

$$h^*_{H,n} \text{ optim. ds } H \text{ avec ex.} = \operatorname{argMin}_{h \in H} (E_n(h))$$

$$\text{où } E_n(h) = 1/N \sum_i (L(h(x_i), y_i))$$

$$E(h^*_{H,n}) - E(h^*) = [E(h^*_{H,n}) - E(h^*_H)] + [E(h^*_H) - E(h^*)]$$

## APPRENTISSAGE SUPERVISÉ : définition formelle

« **APPRENDRE = INFERER/INDUIRE + GENERALISER** »

- Etant donné un ensemble fini d'exemples  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , où  $x_i \in \mathbb{R}^d$  vecteurs d'entrée, et  $y_i \in \mathbb{R}^s$  sorties désirées (fournies par le « superviseur »), trouver une fonction  $h$  qui « approxime et généralise au mieux » la fonction sous-jacente  $f$  telle que  $y_i = f(x_i) + \text{bruit}$

$\Rightarrow$  but = minimiser erreur de généralisation

$$E_{\text{gen}} = \int \|h(x) - f(x)\|^2 p(x) dx$$

(où  $p(x)$  = distrib. de proba de  $x$ )

- En pratique, seule est mesurable l'erreur empirique sur les exemples d'apprentissage :

$$E_{\text{emp}} = \left( \sum_i \|h(\mathbf{x}_i) - y_i\|^2 \right) / n$$

- Travaux de Vapnik et théorie de la « régularisation »  
 $\Rightarrow$  minimiser  $E_{\text{emp}}(h)$  sur une famille  $H$  minimisera aussi  $E_{\text{gen}}$  si  $H$  est de VC-dimension finie

VC-dimension : taille *maximum* d'un échantillon  $S$  telle que pour toute dichotomie de  $S$ , il existe  $h \in H$  la réalisant (en gros, la « complexité » de la famille  $H$ )

## Fonction de coût et terme de régularisation

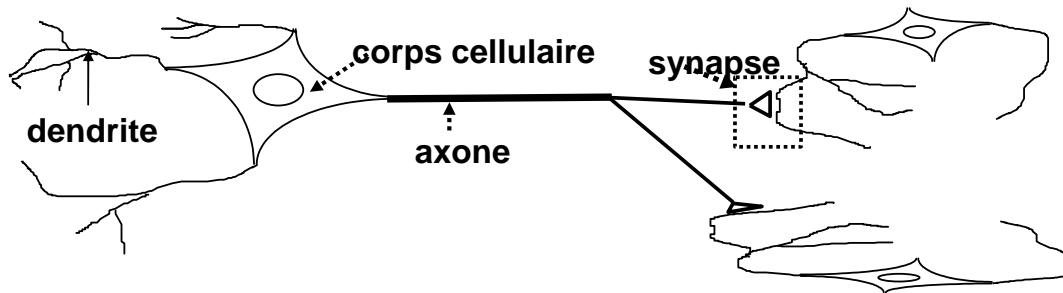
- Plus précisément Vapnik a montré que :  
 $\text{Proba}(\max_{h \in H} |E_{\text{gen}}(h) - E_{\text{emp}}(h)| \geq \varepsilon) < G(n, \delta, \varepsilon)$   
 où  $n = nb$  d'ex. et  $\delta = \text{VC-dim}$ , et  $G$  décroît si  $n/\delta$  augmente  
 $\Rightarrow$  pour être certain de bien minimiser  $E_{\text{gen}}$  en réduisant  $E_{\text{emp}}$ , il faut une VC-dim d'autant plus petite que  $n$  est petit

une approche possible : minimiser  $C = E_{\text{emp}} + \Omega(h)$   
 où  $\Omega(h)$  pénalise les  $h$  « trop complexes »  
 ( $\Rightarrow$  réduction de la VC-dim « effective »)

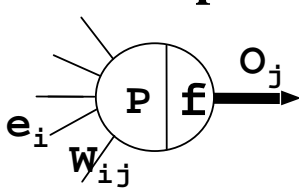
Principe similaire au « rasoir d'Ockham » !!  
 ( $\approx$  « pourquoi faire compliqué si on peut faire simple ? »)

# RESEAUX NEURONAUX

- Inspirés de l'architecture et fonctionnement cerveau



- Modèle mathématique paramétré simple + algos d'adaptation des paramètres



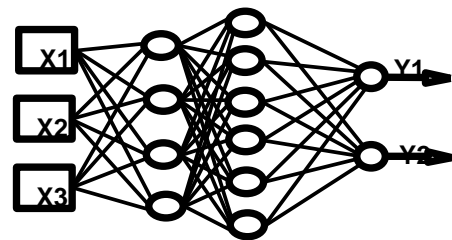
$$O_j = f(P(\vec{e}, \vec{W}_j))$$

avec par exemple

$$P(\vec{e}, \vec{W}_j) = \sum_i e_i W_{ij}$$

$$f(p) = \tanh(p)$$

neurone « formel »



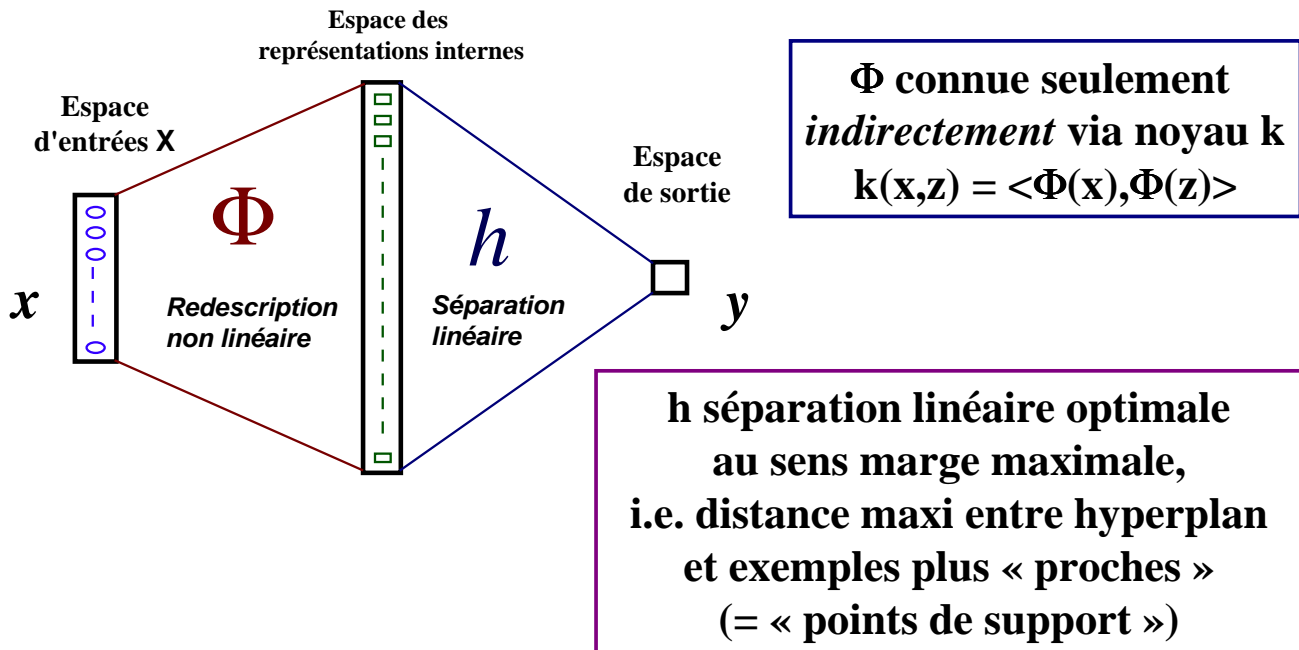
Réseau =  
assemblage de neurones

## RESEAUX NEURONAUX (2)

- Apprentissage = à partir d'exemples de couples (entrée, sortie), le réseau modifie :
  - les paramètres  $W$  (poids des connexions)
  - éventuellement son architecture  $A$   
(en créant/éliminant neurones ou connexions)

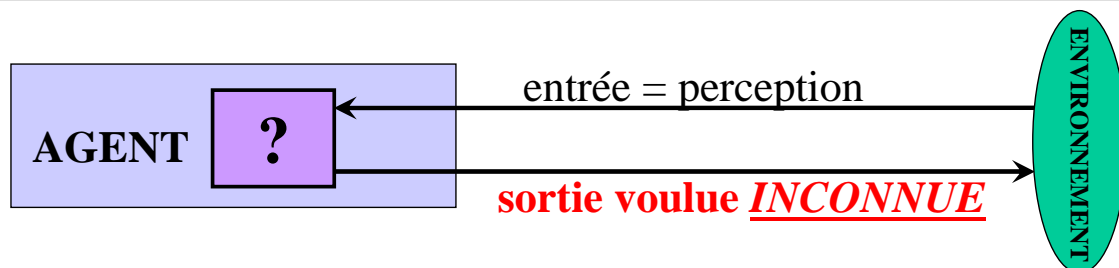
**Plus de détails sur divers types de neurones, de réseaux et les algorithmes d'apprentissage dans le cours dédié aux réseaux neuronaux...**

# SVM = Support Vector Machines (= Séparateur à Vastes Marges)

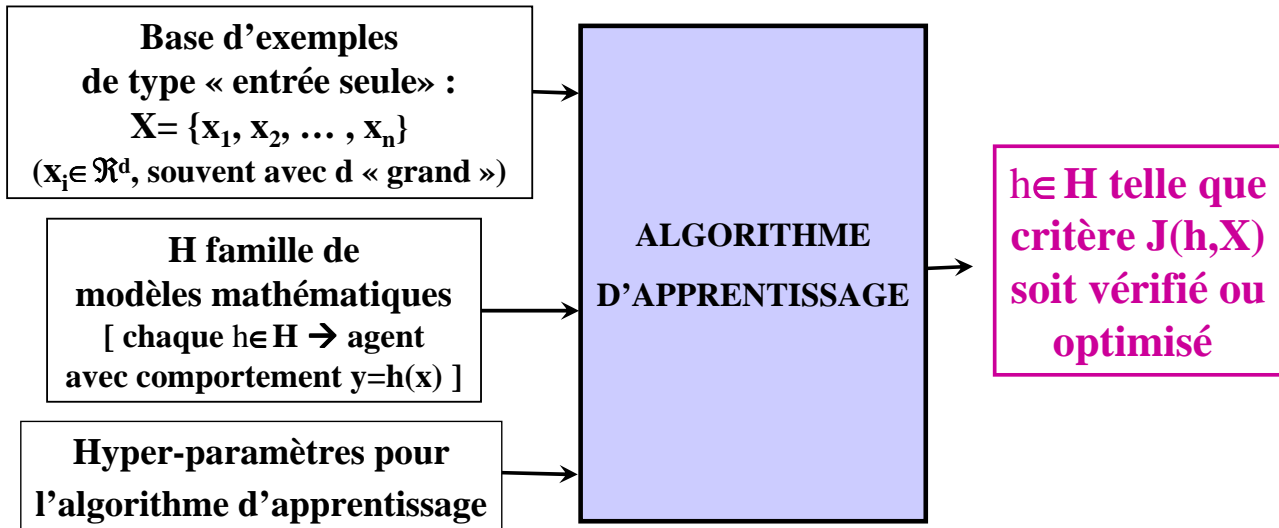


Plus de détails dans partie du cours consacrée à cette technique

## APPRENTISSAGE NON SUPERVISÉ



- Soit on n'a des exemples que de type « entrée », et on cherche à obtenir un agent dont la « sortie » vérifie une certaine propriété (par exemple, sortie obtenue identique ou « proche » pour des entrées « voisines »)
- Soit on dispose juste d'un environnement (réel ou simulé) dans lequel on peut placer l'agent pour « évaluer » son comportement de façon à l'améliorer



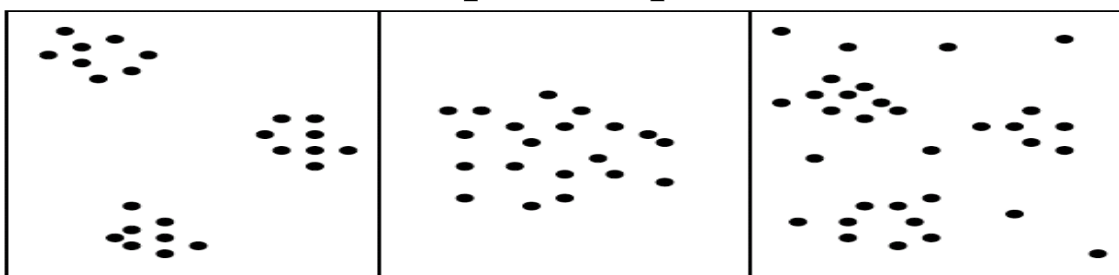
## Exemple typique : le « clustering »

- $h(\mathbf{x}) \in C = \{1, 2, \dots, \kappa\}$  [chaque  $i \leftrightarrow$  « cluster »]
- $J(h, X) : \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \ll \text{plus faible pour } \mathbf{x}_i, \mathbf{x}_j \text{ tq } h(\mathbf{x}_i) = h(\mathbf{x}_j) \text{ que pour des } \mathbf{x}_i, \mathbf{x}_j \text{ tq } h(\mathbf{x}_i) \neq h(\mathbf{x}_j) \gg$

## Le clustering (Regroupement, en français)

Objectif = structuration des données

- On cherche à regrouper les points proches/similaires en « paquets »
- Pb : les groupes peuvent être assez bien définis et séparés, ou au contraire imbriqués/sans frontières claires, et de formes quelconques



## Notion de proximité

- Mesure de dissimilarité DM : plus la mesure est faible, plus les points sont similaires ( distance)
- Mesure de similarité SM : plus la mesure est grande, plus les points sont similaires

## Comment mesurer la distance entre 2 points $d(x_1; x_2)$ ?

- distance euclidienne :  
 $d_2(x_1; x_2) = \sum_i (x_{1_i} - x_{2_i})^2 = (x_1 - x_2) \cdot (x_1 - x_2)$  (norme  $L_2$ )
- distance de Manhattan :  
 $d(x_1; x_2) = \sum_i |x_{1_i} - x_{2_i}|$  (norme  $L_1$ )
- distance de Sebestyen :  
 $d_2(x_1; x_2) = (x_1 - x_2)W^t(x_1 - x_2)$  avec  $W$ = matrice diag.
- distance de Mahalanobis :  
 $d_2(x_1; x_2) = (x_1 - x_2)C^t(x_1 - x_2)$ , avec  $C$ =covariance

- Clustering par agglomération
  - Regroupement Hiérarchique Ascendant (Agglomerative Hierarchical Clustering)
- Clustering par partitionnement
  - Partitionnement Hiérarchique Descendant
  - Partitionnement spectral (cf ACP)
  - K-means
- Clustering par modélisation
  - Mélange de gaussiennes (GMM)
  - Cartes de Kohonen (Self-Organizing Maps, SOM)
- Clustering basé sur la densité

**Principe :** chaque point ou cluster est progressivement "absorbé« par le cluster le plus proche.

## Algorithme

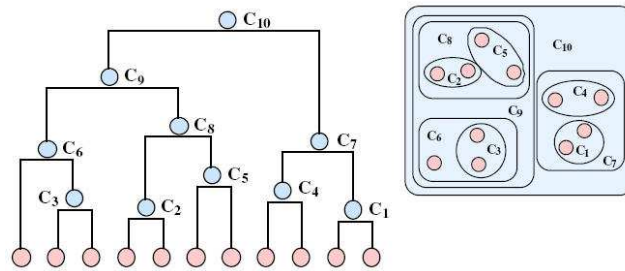
- Initialisation :
  - Chaque individu est placé dans son propre cluster
  - Calcul de la matrice de ressemblance M entre chaque couple de clusters (ici les points)
- Répéter
  - Sélection dans M des deux clusters les plus proches  $C_i$  et  $C_j$
  - Fusion de  $C_i$  et  $C_j$  par un cluster  $C_g$  plus général
  - Mise à jour de M en calculant la ressemblance entre  $C_g$  et les clusters existants

Jusqu'à la fusion des 2 derniers clusters

- plus proche voisin :  $\min(d(i;j); i \in C_1; j \in C_2)$
- distance maximum :  $\max(d(i;j); i \in C_1; j \in C_2)$
- distance moyenne :  $(\sum_{i;j} d(i;j))/(n_1 * n_2)$
- distance des centres de gravité :  $d(b_1;b_2)$
- distance de Ward :  $\sqrt{(n_1 n_2 / (n_1 + n_2)) * d(b_1;b_2)}$

**Chaque mesure → variante ≠ de RHA**

- ppV → single-linkage
- distMax → complete-linkage



- dendrogramme = représentation des fusions successives
- hauteur d'un cluster dans le dendrogramme = similarité entre les 2 clusters avant fusion (sauf exception avec certaines mesures de similarité...)

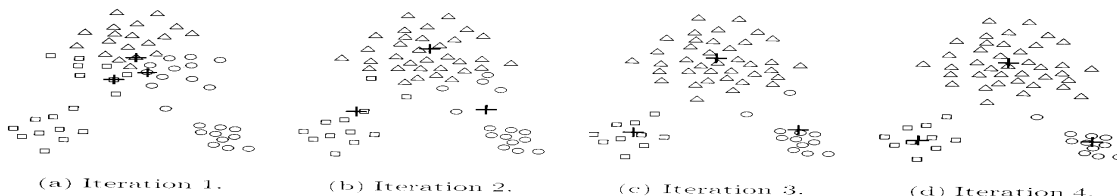
## Clustering par partitionnement

### Cas de l'algo nommé « k-means »

- Chaque cluster  $C_k$  est défini par son « centroïde »  $c_k$ , qui est un « prototype » (un vecteur de l'espace d'entrée) ;
- Tout  $x$  est « assigné » au cluster  $C_{k(x)}$  dont le prototype est le plus proche de  $x$  :  $k(x) = \text{ArgMin}_k (\text{dist}(x, c_k))$
- ALGO :

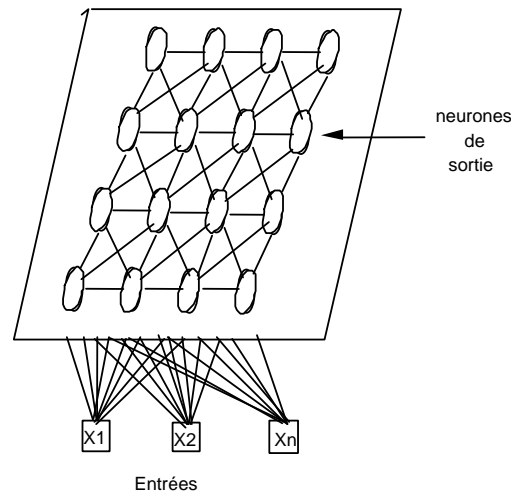
- On choisit  $K$  points distincts  $c_1, \dots, c_K$  au hasard parmi  $\{x_1, \dots, x_n\}$
- On répète jusqu'à « stabilisation » des  $c_k$  :
  - Assigner chaque  $x_i$  au cluster  $C_{k(i)}$  tq  $\text{dist}(x_i, c_{k(i)})$  est minimum
  - Recalculer les centroïdes  $c_k$  des clusters :  $c_k = \sum_{x \in C_k} x / \text{card}(C_k)$

[Ceci revient à minimiser  $D = \sum_{k=1}^K \sum_{x \in C_k} \text{dist}(c_k, x)^2$ ]



# Apprentissage NON supervisé : cartes auto-organisatrices de Kohonen

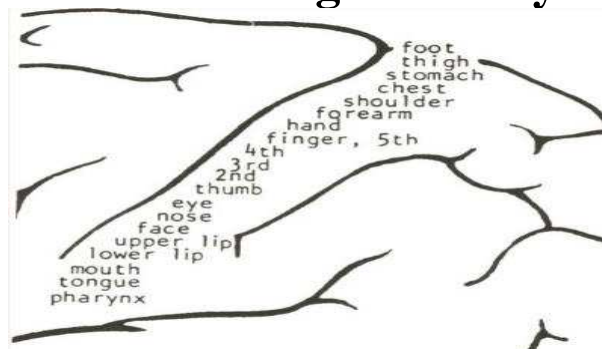
## Réseau de neurones particulier



...avec algorithme d'auto-organisation permettant d'obtenir  
au final un mapping de l'espace d'entrée vers la carte  
qui « respecte la topologie des données »

# Apprentissage NON supervisé : cartes auto-organisatrices de Kohonen

L'inspiration initiale est biologique :  
auto-organisation des régions du système nerveux.

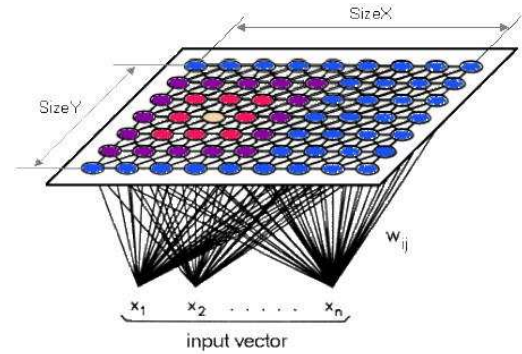


## MOTIVATIONS EN CLASSIFICATION / ANALYSE DE DONNEES

- Organiser/analyser/catégoriser un grand volume de données inexploitable tel quel (en particulier faire du clustering, i.e. regrouper les exemples en paquets "similaires" pour définir des classes)
- Construire une représentation visualisable (1D ou 2D en général) des entrées par une sorte de "projection non-linéaire" de l'espace des entrées (en général de grande dimension) qui respecte la topologie initiale (les "projections" de points proches restent proches).

# Caractéristiques du réseau de Kohonen

- une seule couche de neurones
- neurones de type *distance*
- notion de “voisinage” sur la couche (souvent appelée “carte”)
- chaque neurone peut-être vu comme un vecteur de l'espace d'entrée (cf son vecteur de poids)
- utilisation : pour une entrée  $X$  (de  $\mathcal{R}^d$ ), chaque neurone  $k$  de la carte calcule sa sortie =  $d(W_k, X)$ , et on associe alors  $X$  au neurone « gagnant » qui est celui de sortie la plus faible

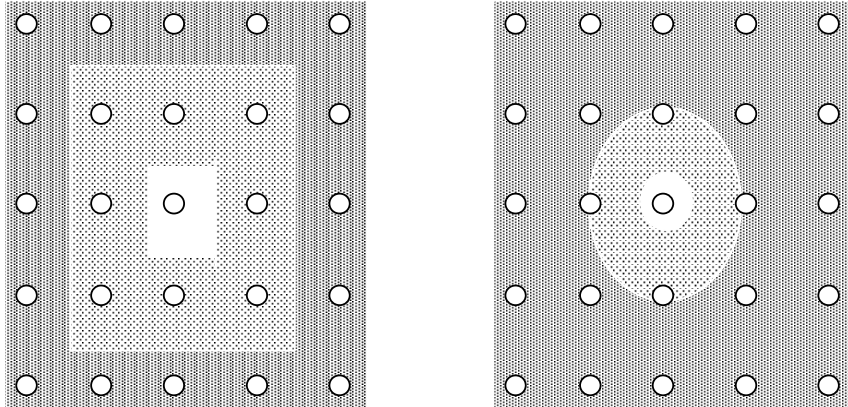


⇒ utilisable pour clustering et/ou comme une sorte de projection non linéaire“ de  $\mathcal{R}^d \rightarrow$  carte

## Principe de l'algorithme de Kohonen

- La réponse d'une cellule  $i$  de poids  $W_i = (w_{i1}, \dots, w_{in})$  à une forme  $X = (x_1, \dots, x_n)$  est la distance euclidienne de  $X$  à  $W_i$ .
- l'apprentissage :
  - repérer la cellule la plus active (la plus proche)
  - essayer de la rendre encore plus active  
EN MEME TEMPS QUE SON VOISINAGE.
- 2 paramètres : la taille du voisinage (rayon)  
le pas  $\alpha(t)$  de la modification des poids  
qui diminuent avec les itérations

## définition de voisinages sur la carte :



$V_i(t)$  voisinage décroissant avec itération  $t$

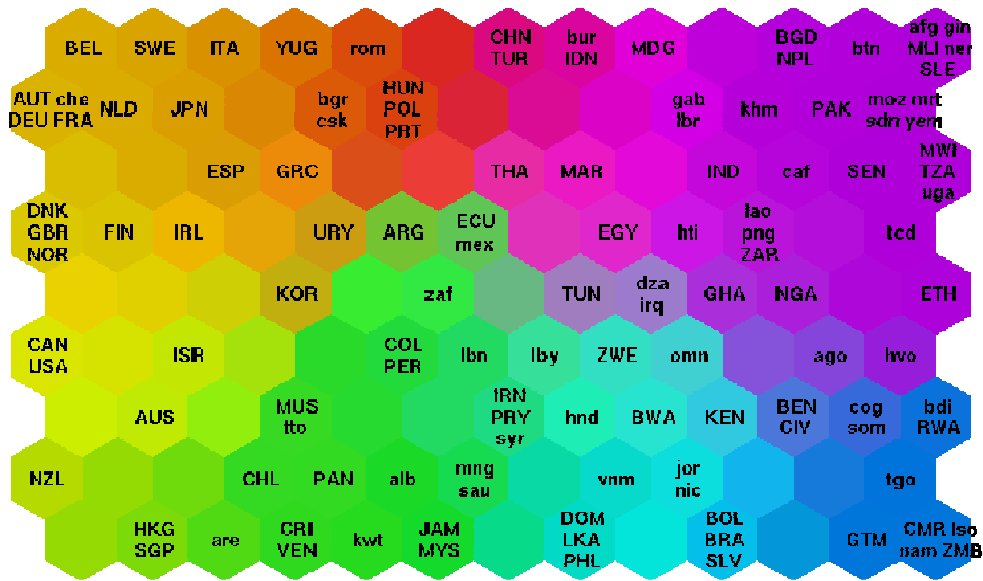
Variante couramment utilisée : « voisinage » gaussien de « largeur » décroissant avec le temps

# L'ALGORITHME DE KOHONEN

- $t=0$ , initialiser les poids (hasard ?)
- date  $t$ , présenter l'exemple  $X$ 
  - déterminer le neurone "gagnant"  $g$  de poids le plus proche
  - déterminer le "pas"  $\alpha(t)$  [et éventuellement le voisinage  $V(t)$ ]
  - modifier les poids :
 
$$w_i(t+1) = w_i(t) + \alpha(t) (X - w_i(t)) \beta(i, g, t)$$
 avec  $\beta(i, g, t) = 1$  si  $i \in V(t)$  et 0 sinon (cas voisinage limité),  
 ou bien  $\beta(i, g, t) = \exp(-\text{dist}(i, g)^2 / \sigma(t)^2)$  [par exemple]
- $t = t+1$
- Convergence de l'algorithme :  
conditions sur  $\alpha(t)$  ( $1/t$  convient)

*[Voir démo]*

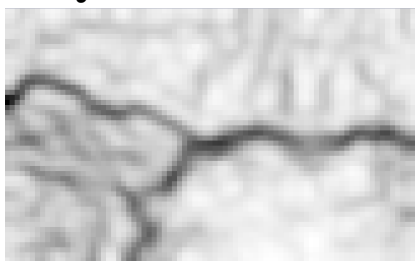
# Exemple d'application de Kohonen



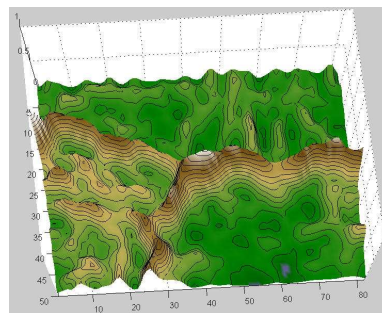
Résultat d'un apprentissage sur une base où chaque exemple est un pays, représenté par un vecteur de 39 indicateurs de la qualité de vie (état de santé, espérance de vie, nutrition, services éducatifs...)

# Utilisation de Kohonen pour clustering

- Analyse des distances entre neurones de carte (U-matrix)

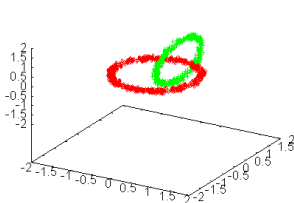


Niveau de gris  
(+ sombre = + gde distance)



Idem en « vue 3D »  
de courbes de niveau

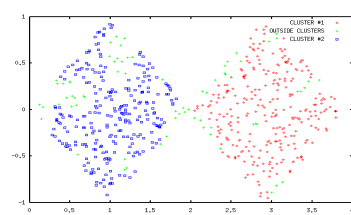
➔ Possibilité de segmentation automatisée qui fournit un *clustering sans a priori* (nb et formes amas) sur données



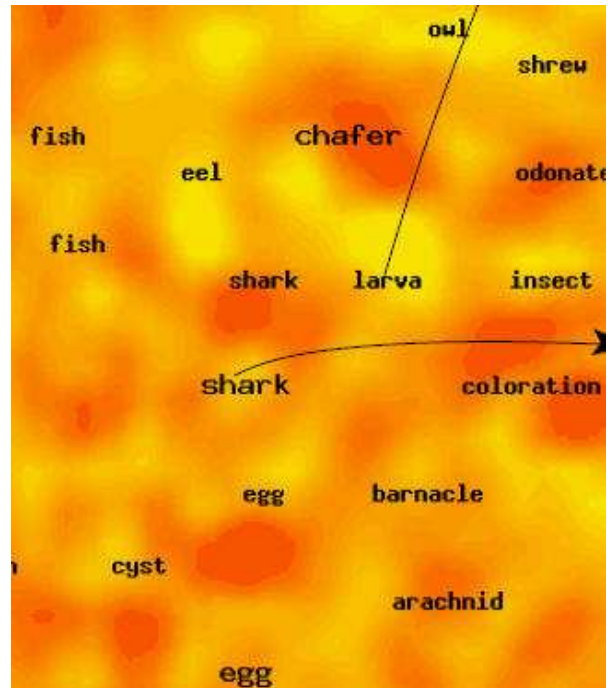
Exemple « chainLink »



Exemple « twoDiamonds »

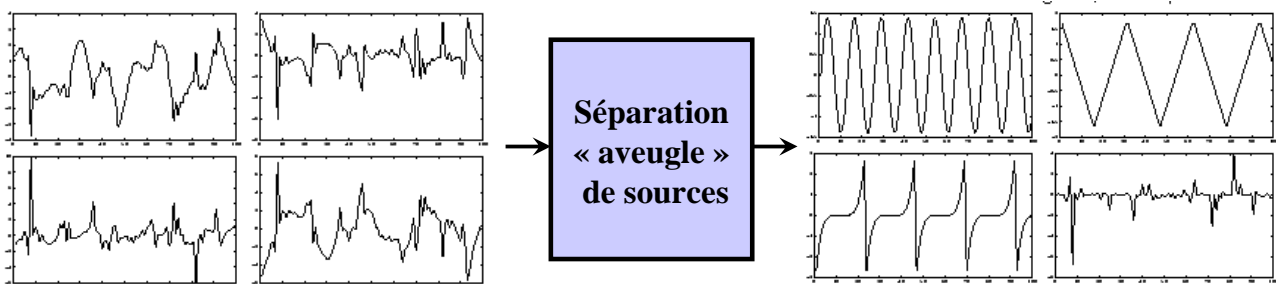


- Chaque document représenté  $\approx$  comme un histogramme des mots contenus
- A droite extrait d'une carte obtenue avec tous les articles de l'Encyclopedia Universalis...



WebSOM (voir démo, etc... à <http://websom.hut.fi/websom>)

## Apprentissage NON supervisé : séparation aveugle de sources



### Objectif :

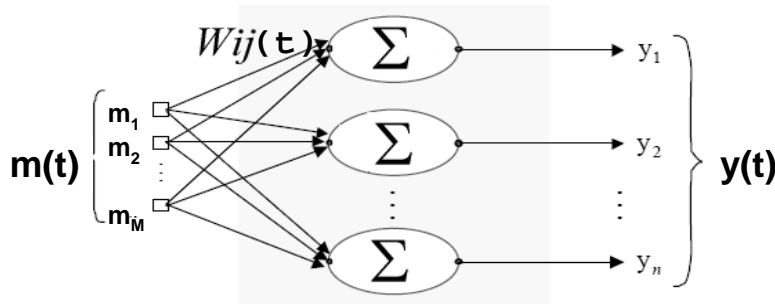
partant de  $M$  mélanges *différents* de  
 $M$  signaux *indépendants*, parvenir à  
reconstituer les signaux sources

(exemple : plusieurs locuteurs ou instruments, et autant de micros placés à des endroits différents)



**Critère à optimiser par l'agent :**  
**« indépendance » des  $y_i$**

- **Modèle : transformation linéaire (= une couche de neurones sommateurs sans biais) *variant avec le temps***



**Nombreuses variantes d'algorithmes de mise à jour des poids, correspondant parfois à la minimisation explicite d'une quantité (Information Mutuelle, corrélations croisées,...) parfois à une simple heuristique.**

Par exemple :

$$W(t+1) = W(t) + \mu(I - g_1(y)g_2(y)) \cdot W(t)$$

(Jutten et Héroult)

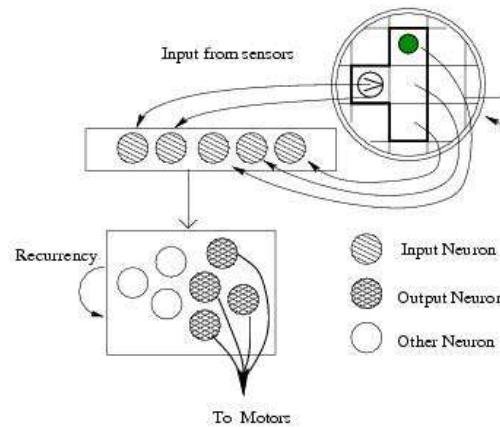
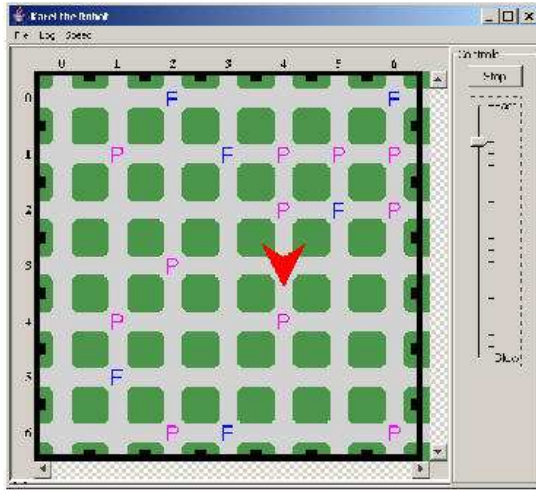
**OU**

$$W(t+1) = W(t) + \mu(I - y \cdot y^T - g(y) \cdot y^T - y \cdot g(y^T)) \cdot W(t)$$

(Cardoso, EASI)

[où  $m = \ll \text{pas} \gg$  ;  $I = \text{matrice identité}$  ;  $g_1, g_2 = \text{fonctions non-linéaires}$  ; [voir démo](#)]

Par exemple, trouver automatiquement pour un « robot » autonome un comportement qui réalise au mieux une tâche donnée



⇒ Apprentissage « par renforcement », et/ou heuristiques diverses (algorithmes évolutionnistes, ...)

## QUELQUES REFERENCES SUR L'APPRENTISSAGE ARTIFICIEL

- *Apprentissage artificiel : concepts et algorithmes*, A. Cornuéjols, L. Miclet & Y. Kodratoff, Eyrolles, 2002.
- *Pattern recognition and Machine-Learning*, Christopher M. Bishop, Springer, 2006.
- *Introduction to Data Mining*, P.N. Tan, M. Steinbach & V. Kumar, AddisonWesley, 2006.
- *Machine Learning*, Thomas Mitchell, McGraw-Hill Science/Engineering/Math, 1997.