APPRENTISSAGE NON-SUPERVISÉ

Pr. Fabien Moutarde
Centre de Robotique (CAOR)
MINES ParisTech (Ecole des Mines de Paris)
PSL Research University

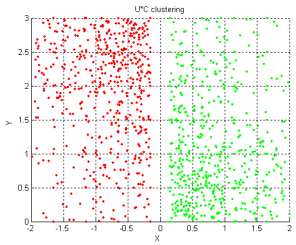
`Fabien.Moutarde@mines-paristech.fr`

`http://people.mines-paristech.fr/fabien.moutarde`

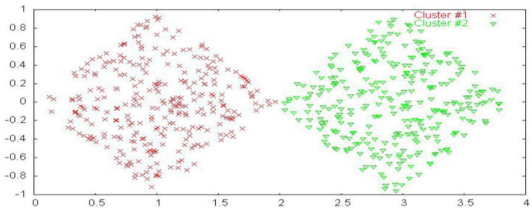
Machine-learning *typology*

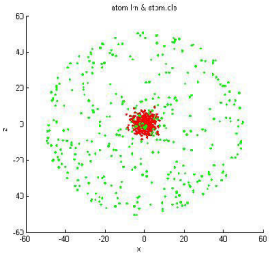
- **What kind of (mathematical) model?**
  → polynom/spline, decision tree, neural net, kernel machine, …
- **Availability of target output data?**
  → ***supervised*** learning v.s. ***unsupervised*** learning
- **Permanent adaptability?**
  → ***offline*** learning v.s. ***online (life-long)*** learning
- **Which objective function?**
  → cost function (quadratic error, …), implicit criterium, …
- ***How*** **to find the best-fitting model?**
  → algorithm type (exact solving, gradient descent, quadratic optimization, heuristics, …)

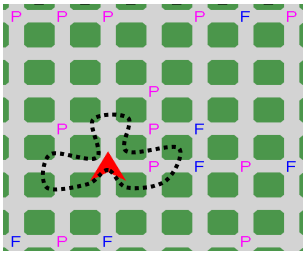
Exemples apprentissage NON-supervisé



Analyse répartition données (clustering)







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

Apprentissage *NON* supervisé *à partir de données*

Base d'exemples de type « entrée seule» :
$X = \{x_1, x_2, \dots, x_n\}$
($x_i \in \Re^d$, souvent avec d « grand »)

H famille de modèles mathématiques
[chaque $h \in H$ → agent avec comportement $y=h(x)$]

Hyper-paramètres pour l'algorithme d'apprentissage

ALGORITHME D'APPRENTISSAGE

$h \in H$ telle que critère J(h,X) soit vérifié ou optimisé

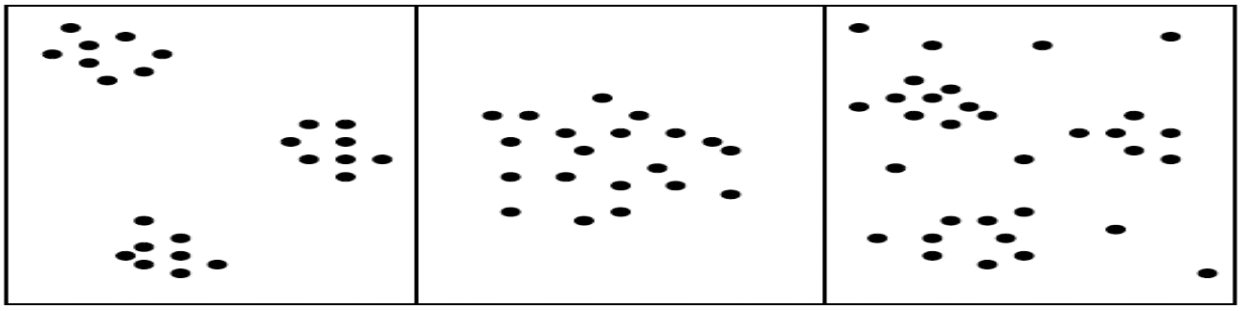
Exemple typique : le « clustering »

- $h(x) \in C=\{1,2, \dots, K\}$ [chaque i ←→ « cluster »]
- $J(h,X) : dist(x_i, x_j)$ « plus faible pour x_i, x_j tq $h(x_i)=h(x_j)$ que pour des x_i, x_j tq $h(x_i) \neq h(x_j)$ »

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



Proximité et distance

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 :
  $d^2(x_1;x_2) = \sum_i (x_{1i}-x_{2i})^2 = (x_1-x_2).{}^t(x_1-x_2)$ *[norme L_2]*
- distance de ***Manhattan*** :
  $d(x_1;x_2) = \sum_i |x_{1i}-x_{2i}|$ *[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= matr. covariance]*

Types de clustering

- **Clustering par *agglomération***
  - Regroupement Hiérarchique Ascendant (Agglomerative Hierarchical Clustering)
- **Clustering par *partitionnement***
  - Partitionnement Hiérarchique Descendant
  - Partitionnement spectral (séparation dans espace de vecteurs propres de Matrice adjacence)
  - K-means
- **Clustering par *modélisation***
  - Mélange de gaussiennes (GMM)
  - Cartes de Kohonen (Self-Organizing Maps, SOM)

- **Clustering basé sur la densité**

Regroupement H. Ascendant

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 en 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

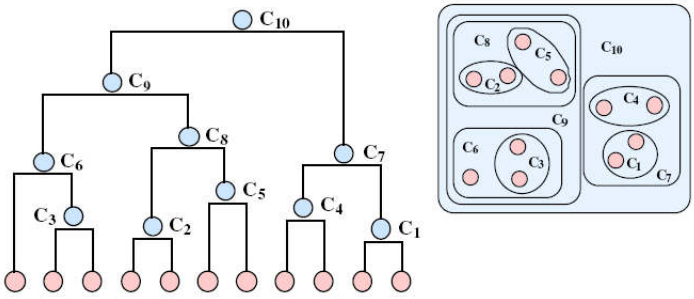
Dissemblance entre 2 clusters ??

- distance ***mini*** (entre plus proches voisins) :
  $\min(d(i,j)\ i\in C_1\ \&\ j\in C_2)$
- distance ***maxi*** : $\max(d(i,j)\ i\in C_1\ \&\ j\in C_2)$
- distance ***moyenne*** :
  $(\Sigma_{i\in C1 \& j\in C2}\, d(i,j))/(card(C_1)\times card(C_2))$
- distance des ***centres de gravité*** : $d(b_1;b_2)$
- distance de ***Ward*** :
  $sqrt(n_1n_2/(n_1+n_2))\times d(b_1;b_2)$ *[où $n_i=card(C_i)$]*

Chaque mesure → variante ≠ de RHA

- distMin (ppV) → ***single-linkage***
- distMax → ***complete-linkage***

RHA: Dendrogramme



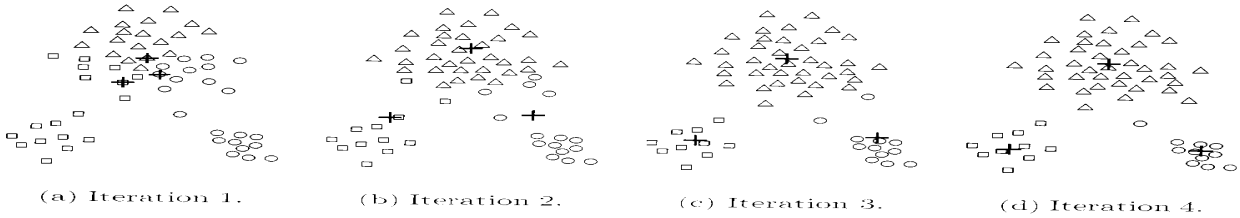
- dendrogramme = représentation des fusions successives
- hauteur d'un cluster dans le dendrogramme = similarité entre les 2 clusters avant fusion (sauf exception avec certaines mesures de similarité...)

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 $c_1, \ldots, c_K$ au hasard parmi $\{x_1, \ldots, x_n\}$**
  - **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 :** $c_k = \sum_{x \in C_k} x \Big/ card(C_k)$

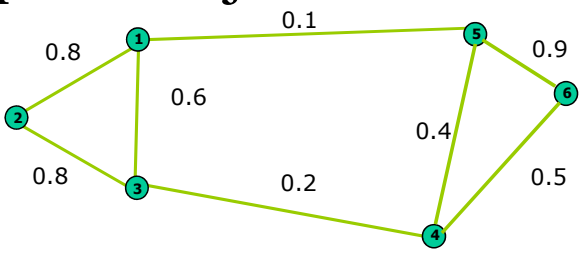
    [Ceci revient à minimiser $D = \sum_{k=1}^{K} \sum_{x \in C_k} dist(c_k, x)^2$ **]**



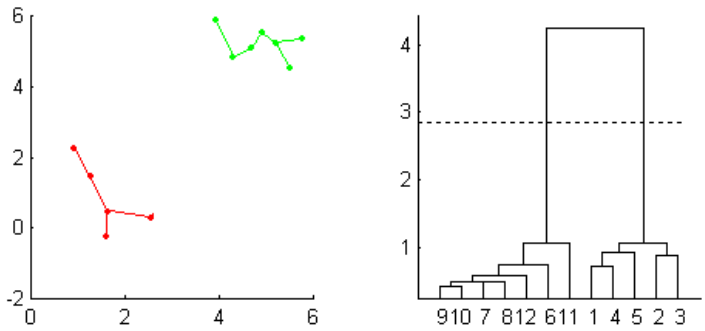
Partitionnement spectral

- Principe = passer par ***graphe*** d'adjacence

nœuds = points données
val. arêtes = similarités
(ds [0;1], 1 ←→ même pt)



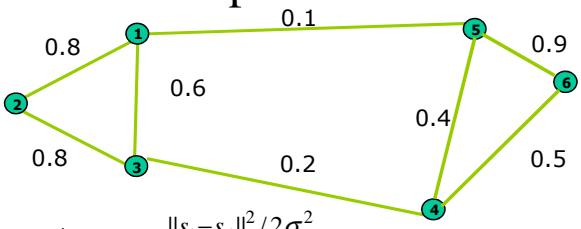
➔ ***algos de partitionnement de graphe (min-cut, etc…) permettent séparer points en groupes***



Ex: sur arbre couvrant minimal (Minimal Spanning Tree)
supprimer arêtes de + petite à + grande → single-linkage clusters

Partitionnement spectral : algo

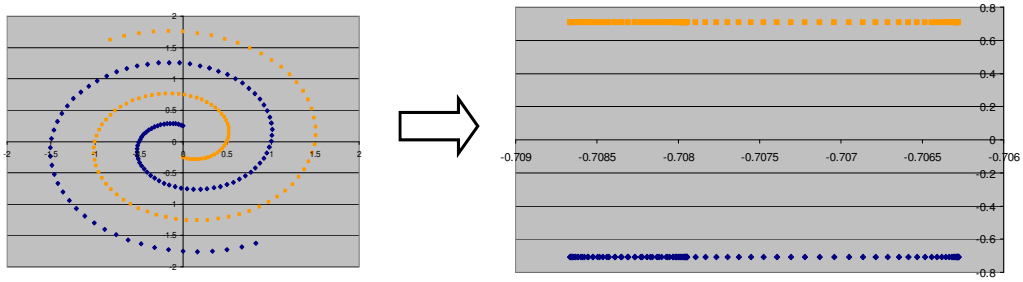
- Calculer matrice « Laplacienne » L=D-A du graphe adjacence



$$A_{ij} = e^{-\|s_i - s_j\|^2 / 2\sigma^2}$$

	x_1	x_2	X_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.1	-0.4	-0.5
x_5	-0.1	0	0	-0.4	1.4	-0.9
x_6	0	0	0	-0.5	-0.9	1.4

- Trouver et trier valeurs propres de L (symétrique => valeurs propres réelles≥0, et vecteurs propres ⊥
- Projeter pts $s_i \in \Re^d$ sur k vect propres de + gdes valeurs propres
  → nouvelle représentation $x_i \in \Re^k$, où séparation + facile



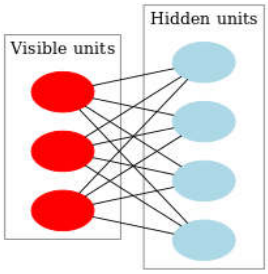
Autres algos non-supervisés pour apprentissage de *répartition des exemples*

- **Apprendre DISTRIBUTION DE PROBABILITE :**
  - ***Restricted Boltzmann Machine (RBM)***

- **Apprendre sorte de « PROJECTION » DANS ESPACE DE FAIBLE DIMENSION (« Manifold Learning ») :**
  - ***Analyse en Composantes Principale (ACP) non-linéaire***
  - ***Auto-encodeurs***
  - ***Cartes topologiques de Kohonen***
  - …

Restricted Boltzmann Machine

- **Proposé par Smolensky (1986) + *Hinton vers 2005***
- **Permet apprendre distribution de probas des exemples**
- **Réseau de neurones BINAIRES à 2 couches (graphe bi-partite) et *connections bidirectionnelles***
- **Utilisation :** $P(v,h) = \frac{1}{Z} e^{-E(v,h)}$ $P(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}$



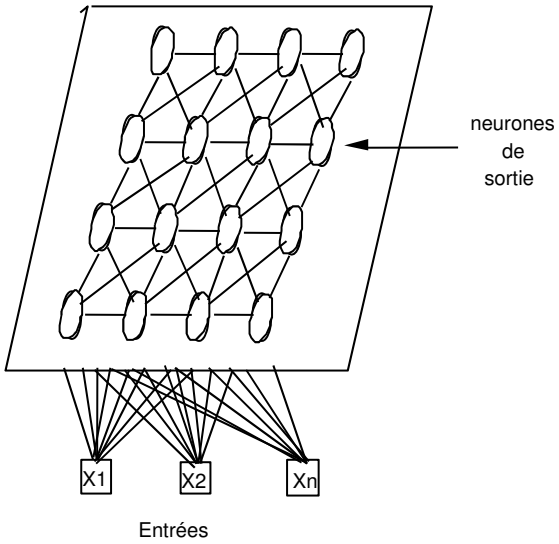
où $E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$ **= énergie**

- **Apprentissage : maximiser produit des probas $\Pi_i P(v_i)$ par descente de gradient par *Contrastive Divergence***

$\Delta w_{i,j} = \epsilon(vh^{\intercal} - v'h'^{\intercal})$ **où v' = reconstruction à partir h et h' déduit de v'**

Apprentissage *NON* supervisé : cartes auto-organisatrices de Kohonen

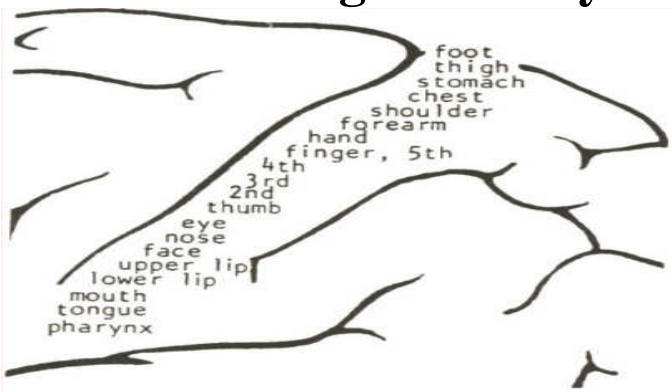
Réseau de neurones particulier



…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.

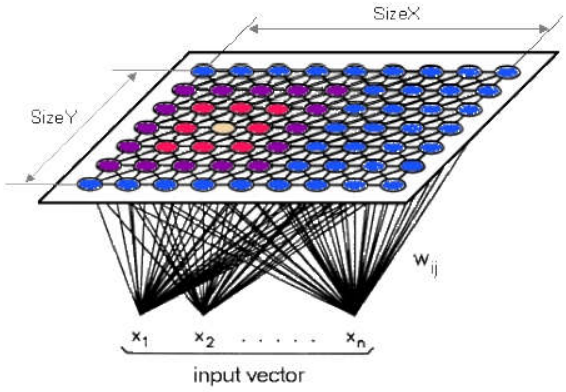


MOTIVATIONS EN CLASSIFICATION / ANALYSE DE DONNEES

- **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 $\Re^d$), 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 $\Re^d$ → carte

Principe de l'algorithme de Kohonen

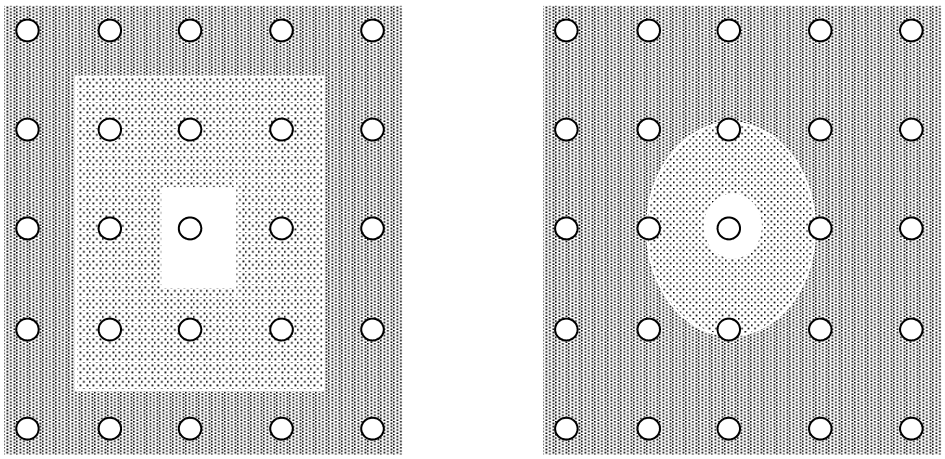
- La réponse d'une cellule i de poids $W_i = (w_{i1}, \dots, w_{in})$ à une forme $X = (x_1, \dots, x_n)$ est la distance euclidienne de X à W_i.

- 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

Voisinage pour carte de Kohonen

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 $\beta(i,g,t)=\exp(-dist(i,g)^2/\sigma(t)^2)$ [par exemple]

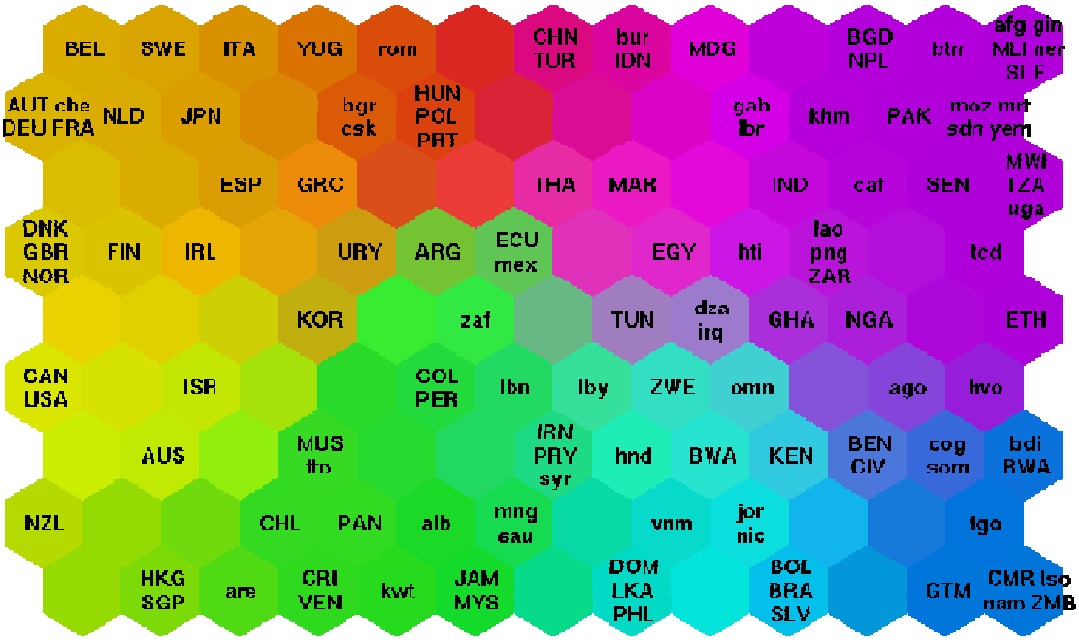
- t = t+1

- Convergence de l'algorithme :

conditions sur $\alpha(t)$ (1/t convient)

[Voir démo]

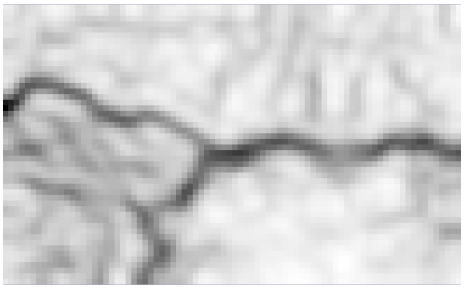
Exemple d'application de Kohonen



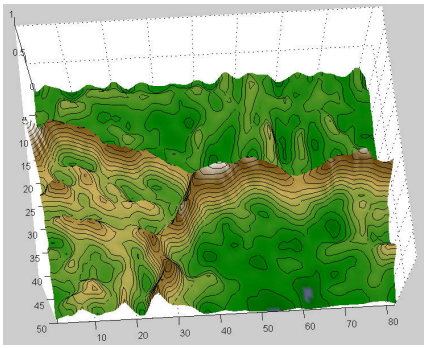
Résultat d'un apprentissage sur une base où chaque exemple est un pays, représenté par un vecteur de 39 indicateurs de la qualité de vie (état de santé, espérance de vie, nutrition, services éducatifs…)

Utilisation de Kohonen pour clustering

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

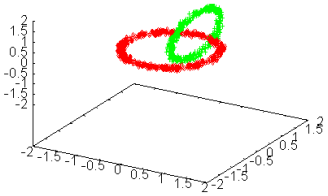


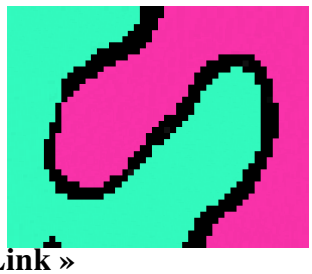
Niveau de gris
(+ sombre = + gde distance)



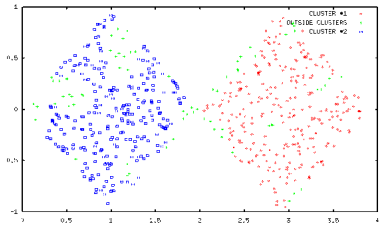
Idem en « vue 3D »
de courbes de niveau

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





Exemple « chainLink »

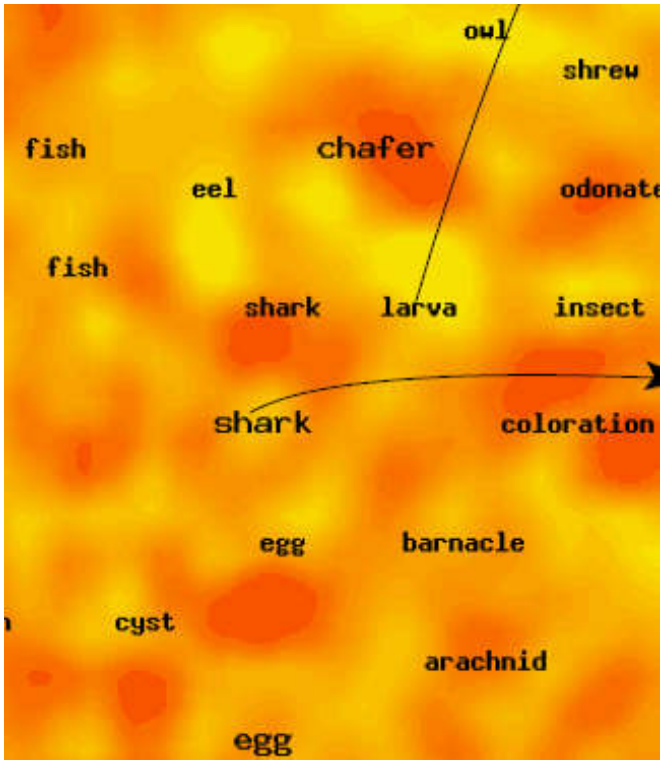


Exemple « twoDiamonds »

Application de Kohonen au « text-mining »

- **Chaque document représenté ≈ comme un histogramme des mots contenus**

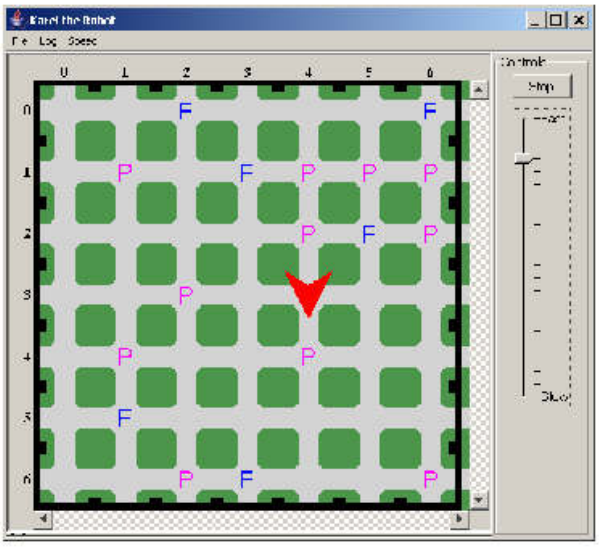
- **A droite extrait d'une carte obtenue avec tous les articles de l'Encyclopedia Universalis…**

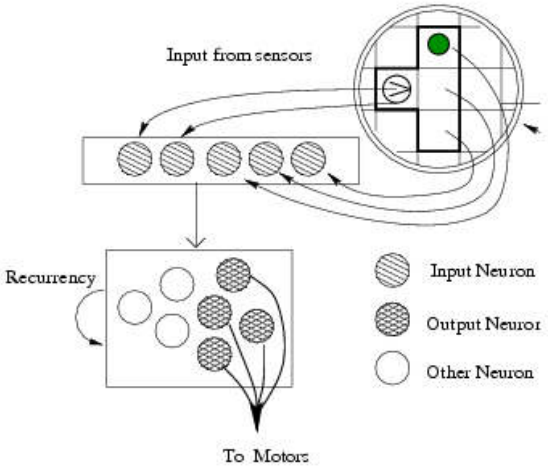


WebSOM (voir démo, etc… à `http://websom.hut.fi/websom`)

Autres types d'apprentissages non-supervisés

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

- ***The Elements of Statistical Learning*** *(2nd edition)*
  T. Hastier, R. Tibshirani & J. Friedman, Springer, 2009.
  http://statweb.stanford.edu/~tibs/ElemStatLearn/
- ***Deep Learning***
  I. Goodfellow, Y. Bengio & A. Courville, MIT press, 2016.
  http://www.deeplearningbook.org/
- ***Pattern recognition and Machine-Learning***
  C. M. Bishop, Springer, 2006.
- ***Introdution to Data Mining***
  P.N. Tan, M. Steinbach & V. Kumar, AddisonWeasley, 2006.
- ***Machine Learning***
  T. Mitchell, McGraw-Hill Science/Engineering/Math, 1997.
- ***Apprentissage artificiel : concepts et algorithmes***
  A. Cornuéjols, L. Miclet & Y. Kodratoff, Eyrolles, 2002.