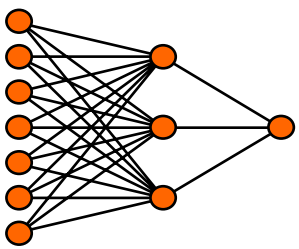
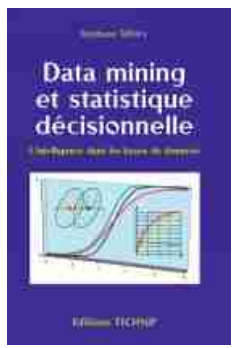
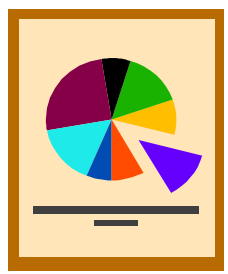
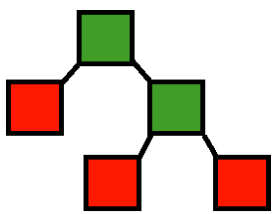
Stéphane Tufféry

DATA MINING
& STATISTIQUE DÉCISIONNELLE









Plan du cours

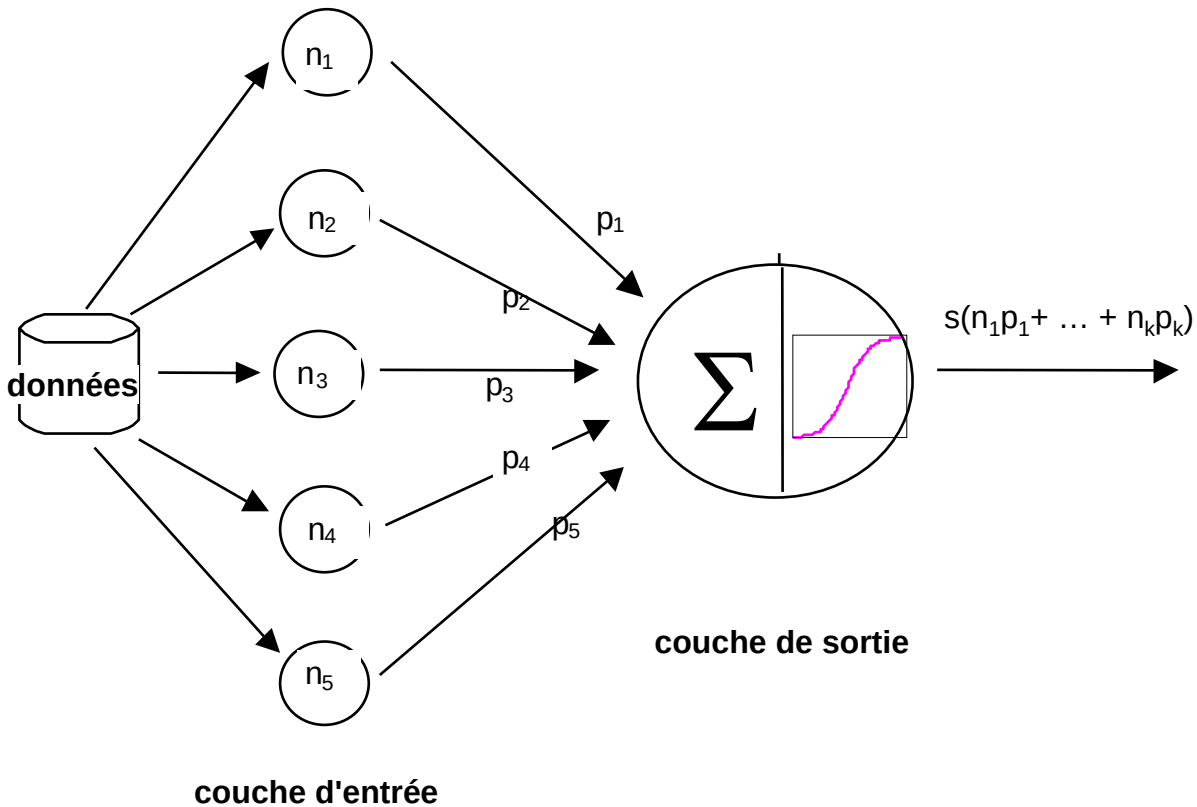
- Qu'est-ce que le data mining ?
- A quoi sert le data mining ?
- Les 2 grandes familles de techniques
- Le déroulement d'un projet de data mining
- Coûts et gains du data mining
- Facteurs de succès - Erreurs - Consulting
- L'analyse et la préparation des données
- Techniques descriptives de data mining
- Techniques prédictives de data mining 1
- *Techniques prédictives de data mining 2 : réseaux de neurones, SVM et algorithmes génétiques*
- Logiciels de statistique et de data mining
- Informatique décisionnelle et de gestion
- CNIL et limites légales du data mining
- Le text mining
- Le web mining

Technique de classement ou de prédiction :

Les réseaux de neurones

Les réseaux de neurones

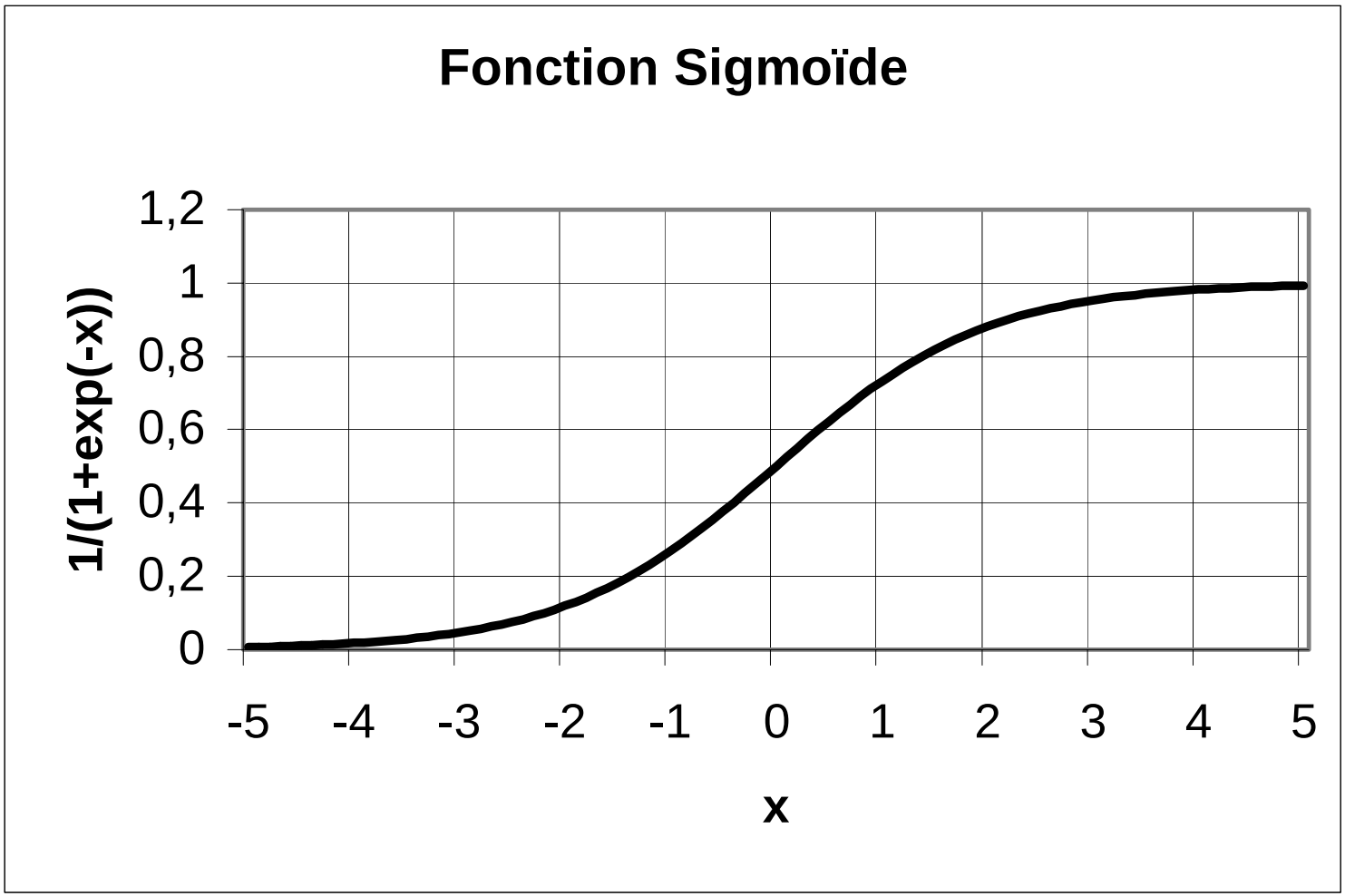
- Un **réseau de neurones** : ensemble de nœuds connectés entre eux, chaque variable correspondant à un nœud
- Le plus courant est le perceptron :



Principe du perceptron multicouches

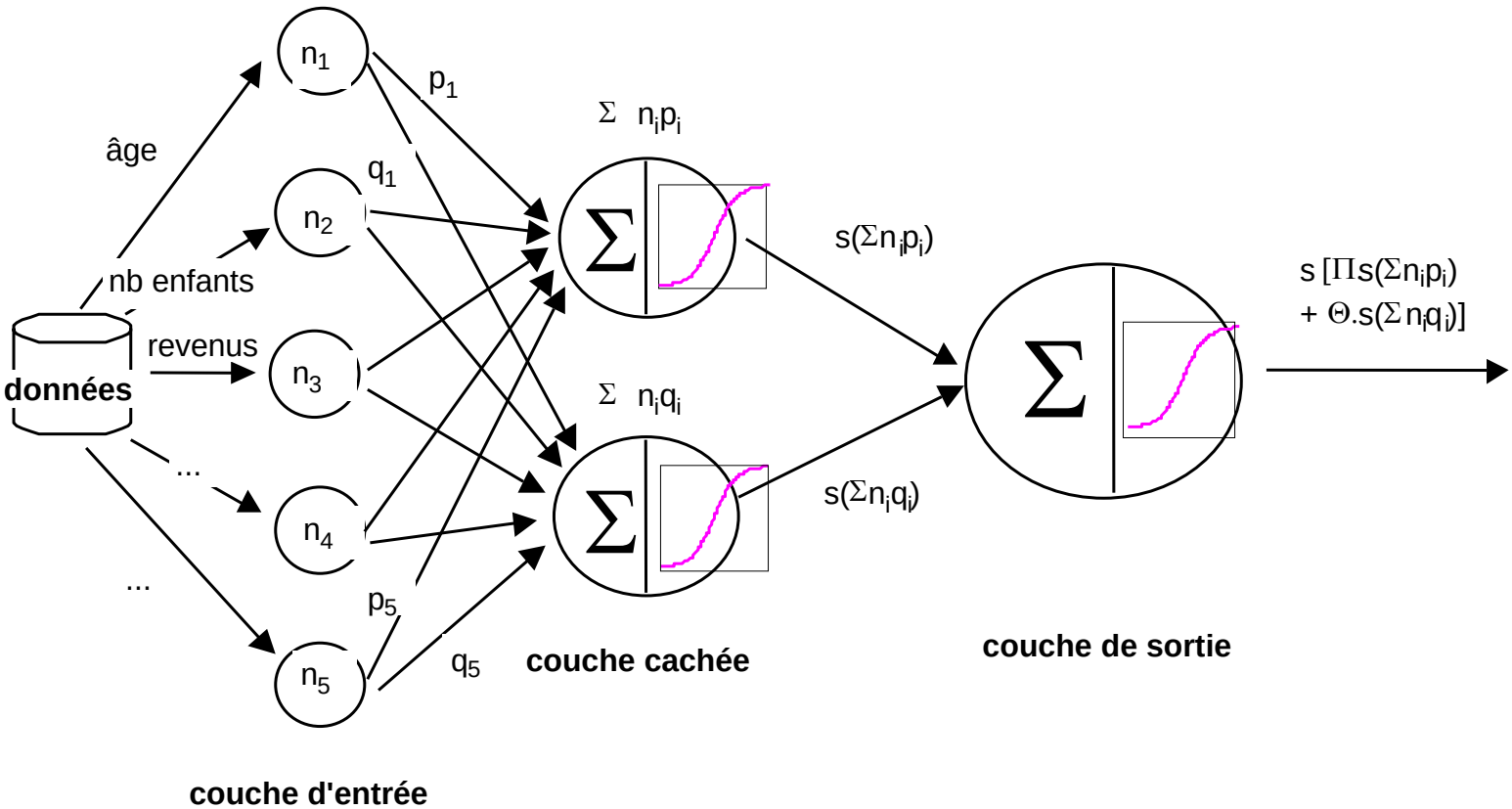
- Lors de l'**apprentissage du réseau de neurones**, pour chaque exemple présenté en entrée, la valeur renvoyée (« rétropropagée ») par le nœud de sortie est comparée à la valeur réelle, et les poids p_i sont ajustés
- La fonction de *combinaison* $\Sigma_i n_i p_i$ (produit scalaire) est suivie d'une fonction de *transfert* : souvent la sigmoïde
  - $s(x) = 1 / [1 + \exp(-x)]$
- L'échantillon d'apprentissage est parcouru plusieurs fois. L'apprentissage s'achève lorsque 1) une solution optimale a été trouvée ou 2) un nb fixé d'itérations a été atteint
- L'apprentissage se fait en ajustant 1 à 1 chaque poids (rétropropagation), ou par modification aléatoire des poids suivie d'un mécanisme de sélection (algorithme génétique)

La fonction logistique



Les réseaux à couches cachées

- On augmente le pouvoir de prédiction en ajoutant une ou plusieurs couches cachées entre les couches d'entrée et de sortie

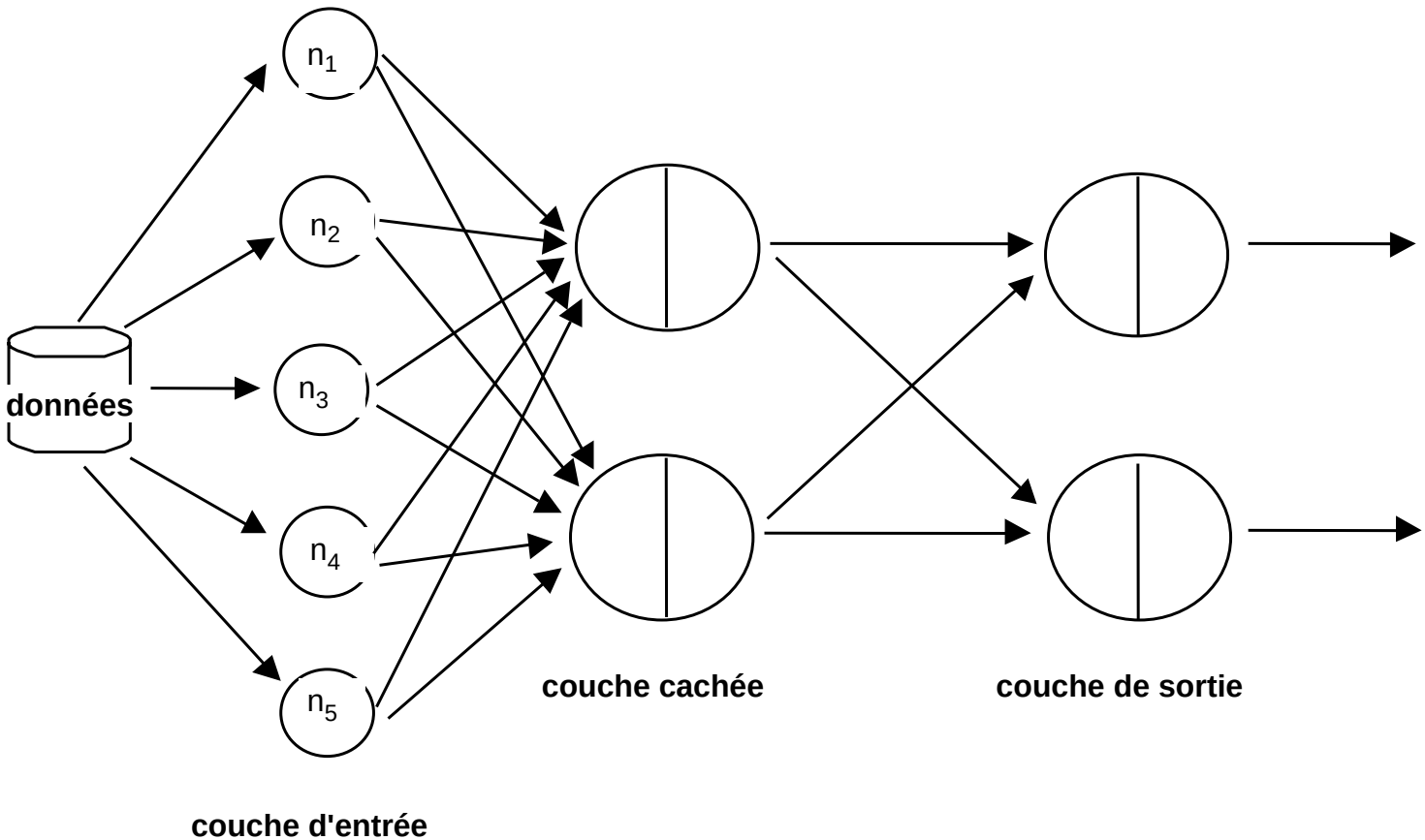


Les réseaux à couches cachées

- Le pouvoir de prédiction augmente avec le nombre de nœuds des couches cachées
  - le nb de couches cachées est très généralement 1 ou 2
  - lorsque ce nombre = 0, le réseau effectue une régression linéaire ou logistique (selon la fonction de transfert)
- Mais ce dernier doit néanmoins être limité pour que le réseau de neurones ne se contente pas de mémoriser l'ensemble d'apprentissage mais puisse le généraliser
  - sinon, il y a **sur-apprentissage**
- Le fait que toutes les valeurs soient comprises entre 0 et 1 permet de prendre en entrée d'un nœud la sortie d'un nœud précédent
- Autre but de la normalisation des valeurs : éviter que les données avec de grandes valeurs « écrasent » les autres

Les réseaux à plusieurs sorties

- La couche de sortie du réseau peut parfois avoir plusieurs nœuds, lorsqu'il y a plusieurs valeurs à prédire.



Différents réseaux de neurones

- Le **Perceptron multicouches** est utilisé pour prédire une variable cible continue ou discrète
- Le **réseau à fonction radiale de base** (« radial basis function » RBF) est aussi utilisé pour prédire une variable cible continue ou discrète
- Le **réseau de Kohonen** effectue les analyses typologiques (clustering, recherche de segments)
- Réseaux par **estimation de densité de probabilité** de Speckt (1990)
  - PNN (probabilistic neural networks) : classement
  - GRNN (general regression neural networks) : régression
- « Analyse discriminante généralisée » : les **Support Vector Machines** de Vladimir Vapnik (1998) sont utilisées pour prédire une variable cible discrète

Mise en œuvre d'un réseau

- Les étapes dans la mise en œuvre d'un réseau de neurones pour la prédiction ou le classement sont :
  - identification des données en entrée et en sortie
  - normalisation de ces données
  - constitution d'un réseau avec une topologie adaptée
  - apprentissage du réseau
  - test du réseau
  - application du modèle généré par l'apprentissage
  - dénormalisation des données en sortie.

Quelques règles empiriques

- Il faut 5 à 10 individus pour ajuster chaque poids
- On recommande d'avoir 1 unité cachée (réseau RBF) ou 1 ou 2 (réseau PMC)
- Un réseau à n unités d'entrée, 1 unité cachée, m unités dans la couche cachée et 1 unité de sortie a n.m+m poids
- Il faut donc un échantillon d'au moins 5(n.m+m) individus pour l'apprentissage
- La valeur de m est généralement comprise en n/2 et 2n
- On a intérêt à diminuer n (en utilisant l'ACP par ex)
- Pour un classement, m $\geq$ nombre de classes
- L'échantillon d'apprentissage ne doit pas être trié selon un ordre significatif, qui pourrait induire le réseau en erreur
- L'échantillon d'apprentissage doit couvrir tous les cas

Algorithmes d'apprentissage 1/2

- Levenberg-Marquardt
  - très performant (converge plus rapidement et vers une meilleure solution que la rétro-propagation)
  - ne fonctionne qu'avec un seul nœud en sortie
  - requiert une grande capacité mémoire, proportionnelle à n^2, avec n = nb de nœuds
    - => limité à des petits réseaux (peu de variables)
- Descente du gradient conjugué
  - ses performances se rapproche de Levenberg-Marquardt en terme de convergence
  - applicable à des réseaux + complexes que Levenberg-Marquardt (avec éventuellement plusieurs sorties)

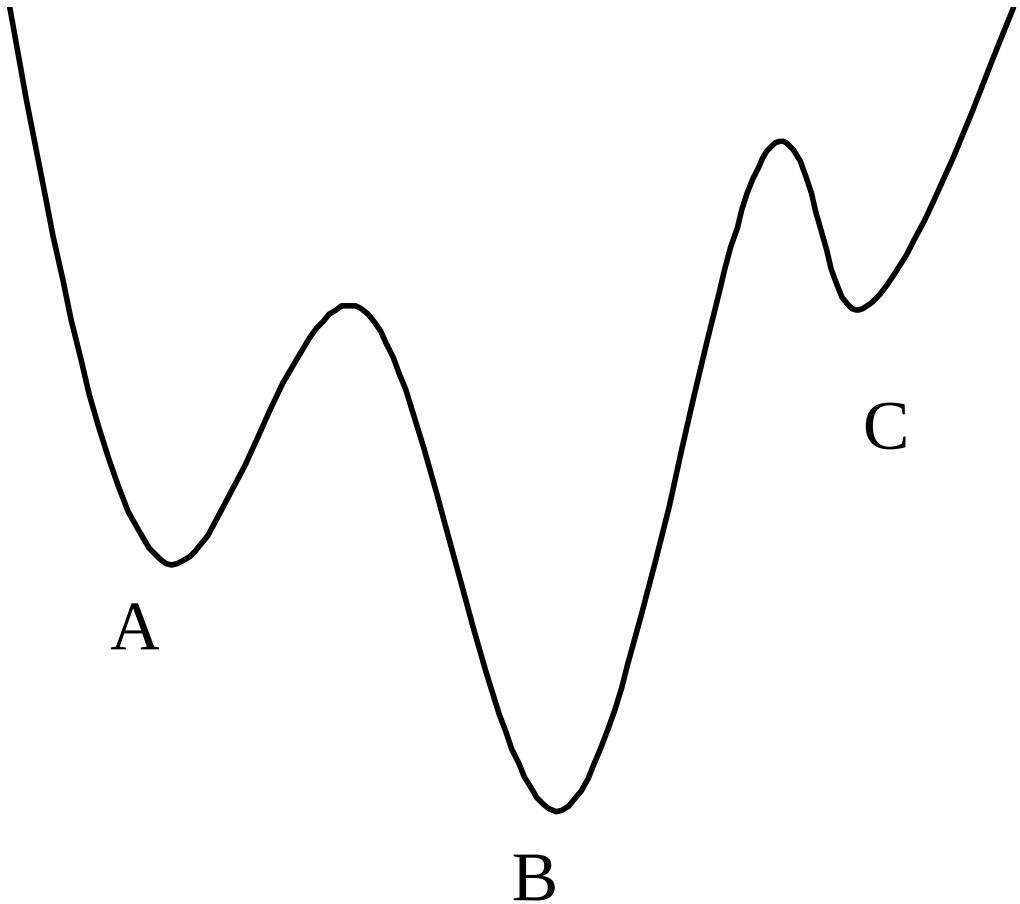
Algorithmes d'apprentissage 2/2

- Rétro-propagation du gradient
  - le plus ancien et le plus répandu
  - moins fiable (sensible aux minimas locaux)
  - utilisé sur de grands volumes de données
- Propagation rapide (« quick propagation »)
  - pas toujours meilleurs que la rétro-propagation et parfois pire
  - difficile à paramétrer
- Quasi-Newton
- Algorithmes génétiques

Rétropropagation du gradient

- Fonction d'erreur : mesure écart entre valeur attendue et valeur fournie par le réseau
- Chaque n-uplet (p_1, p_2, …, p_n) de poids peut être représenté dans un espace à n+1 dimensions, la dernière dimension représentant la fonction d'erreur ε
- Surface d'erreur : ensemble des valeurs (p_1, p_2, …, p_n,ε)
- Modèles linéaires => surface d'erreur mathématiquement bien définie
- Réseaux de neurones => surface d'erreur complexe
- Il faut trouver son point le plus bas
- Rétropropagation du gradient : déplacement sur la surface d'erreur en suivant la ligne de plus grande pente

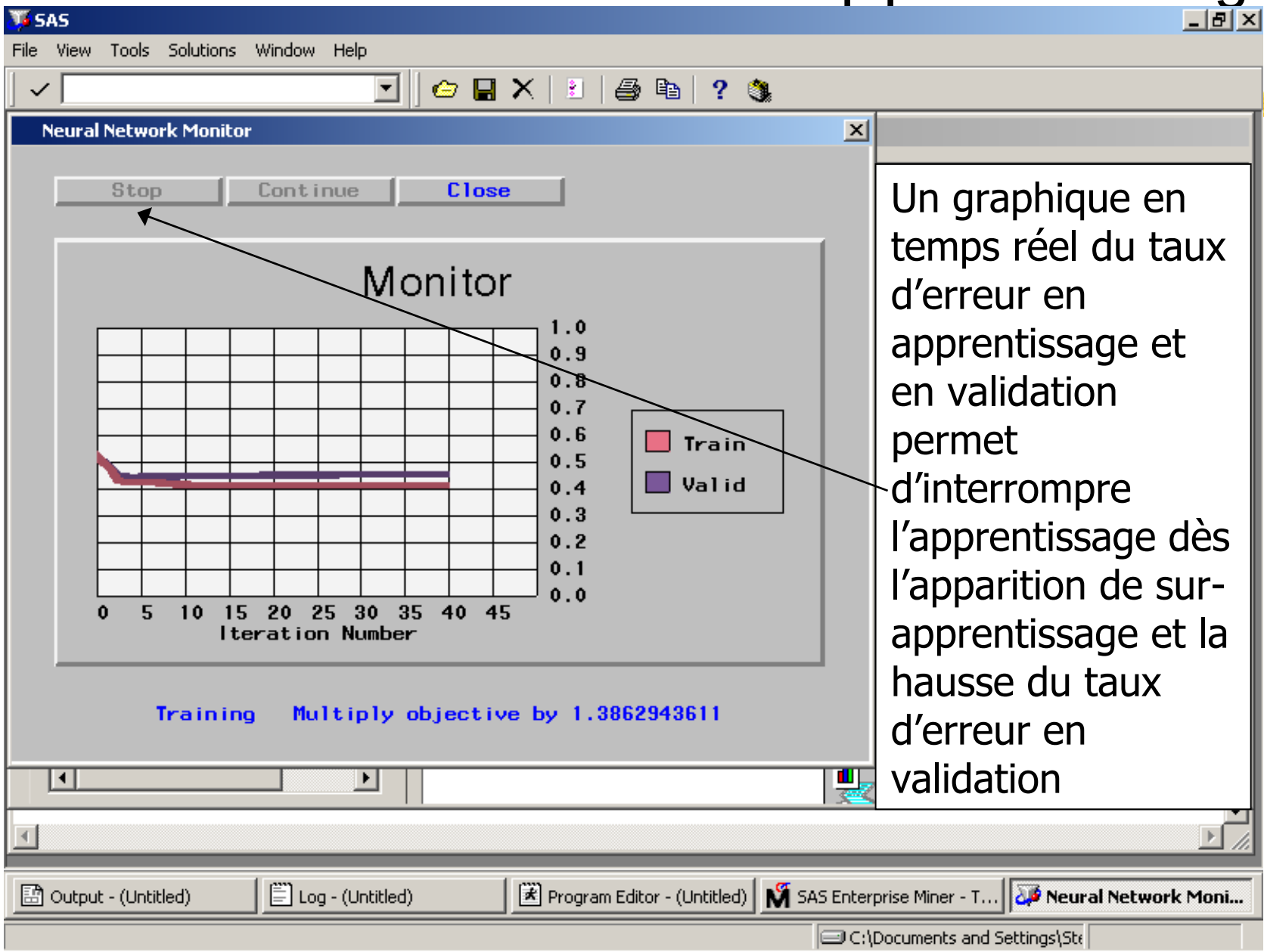
Convergence vers une mauvaise solution



Paramètres de rétropropagation

- Taux d'apprentissage : contrôle l'importance de la modification des poids durant le processus d'apprentissage
  - = contrôle la vitesse de déplacement
  - + il est élevé, + l'apprentissage est rapide mais + le réseau risque de converger vers une solution globalement non optimale
- Moment : agit comme un paramètre d'amortissement en réduisant les oscillations et en aidant à atteindre la convergence
  - = contrôle la rapidité des changements de direction sur la surface d'erreur
  - + il est faible, + le réseau « s'adapte au terrain » mais + l'influence des données extrêmes sur les poids se fait sentir

Contrôle interactif de l'apprentissage



Un graphique en temps réel du taux d'erreur en apprentissage et en validation permet d'interrompre l'apprentissage dès l'apparition de sur-apprentissage et la hausse du taux d'erreur en validation

Réseaux PMC et RBF

	réseau →	PMC	RBF
	« poids »	poids p_i	centre ω_i
couche(s)	fonction combinaison	produit scalaire $\Sigma_i\ p_i x_i$	distance euclidienne $\Sigma_i\ (x_i - \omega_i)^2$
cachée(s)	fonction transfert	logistique $s(X) = 1/(1 + \exp(-X))$	gaussienne $\Gamma(X) = \exp(-X^2/2\sigma^2)$
	nb couches cachées	≥ 1	$= 1$
couche	fonction combinaison	produit scalaire $\Sigma_k\ p_k x_k$	combinaison linéaire de gaussiennes $\Sigma_k\ \lambda_k \Gamma_k$ (voir ci-après)
de sortie	fonction transfert	logistique $s(X) = 1/(1 + \exp(-X))$	fonction linéaire $f(X) = X$
	rapidité	plus rapide en mode « application du modèle »	plus rapide en mode « apprentissage du modèle »

Réponse globale du réseau à chaque individu (x_i)

$$\sum_{k=1}^{\text{nb de noeuds cachés}} \lambda_k \exp\left[-\frac{1}{2} \frac{\sum_{i=1}^{\text{nb de noeuds en entrée}} (x_i - \omega_i^k)^2}{\sigma_k^2} \right]$$

RBF : choix des paramètres initiaux

- Solution simple
  - nombre de nœuds : + grand qu'avec le perceptron
    - parfois plusieurs centaines
  - répartition uniforme des ω^k
  - σ_j deux fois la distance moyenne entre les ω^k
- Solution élaborée : adaptation aux données
  - dans les zones de forte densité en observations (adaptation aux X) et dans les zones où la donnée à prédire varie rapidement (adaptation aux Y) :
    - ω^k plus nombreux
    - σ_k plus faibles
  - partitionnement par k-means ou Kohonen pour fixer les ω^k
  - k-plus proches voisins pour fixer les σ_k (prendre la distance moyenne aux k-plus proches voisins)

RBF : choix des paramètres initiaux

- Importance du choix des σ_k
  - si trop grands => manque de précision du réseau
  - si trop petits => sur-apprentissage => mauvaise généralisation du réseau

Comparaison PMC - RBF

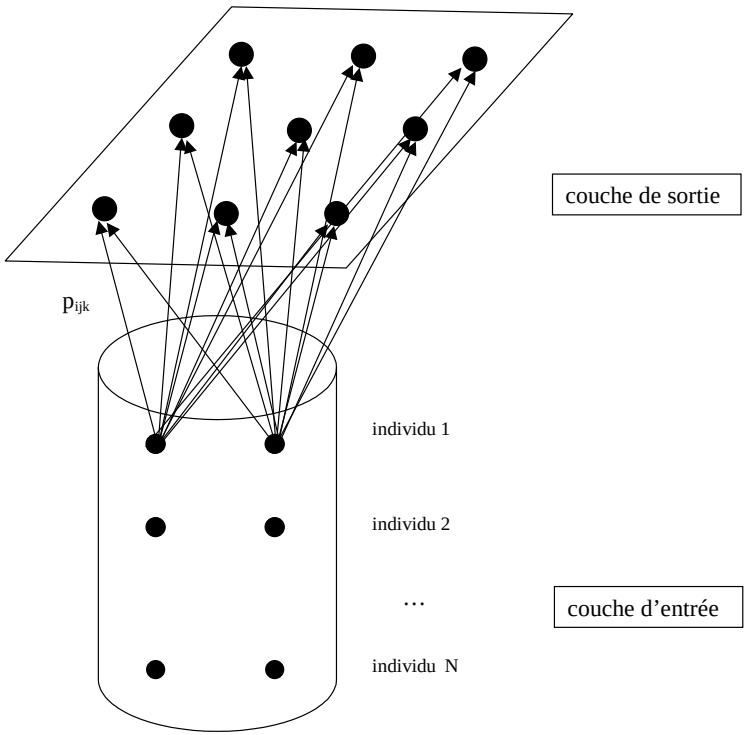
- PMC
  - meilleure capacité de généralisation, notamment sur données bruitées
  - risque de convergence vers un optimum local non global
  - paramètres plus difficiles à régler

- RBF
  - apprentissage plus rapide
  - plus facile de trouver les bons paramètres
  - moins bonne capacité de généralisation, surtout si l'échantillon d'apprentissage ne couvre pas toutes les configurations possibles
  - ce sont un peu les avantages et inconvénients des réseaux à estimation de densité de probabilité

Technique de classification :

Les réseaux de neurones de Kohonen

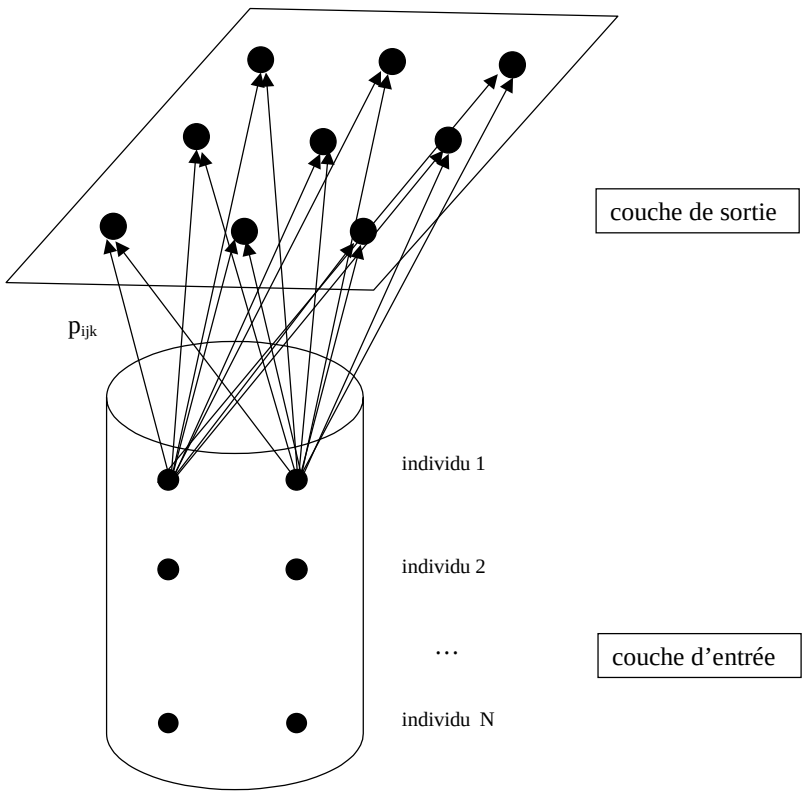
Le réseau de Kohonen



Le + utilisé des réseaux de neurones à apprentissage non supervisé

- Les nœuds de la couche d'entrée correspondent aux variables de classification et servent à présenter les individus
- Les nœuds de la couche de sortie sont disposés sur une grille
- La forme et la taille (par ex : rectangulaire de taille *l*x*m*) de la grille sont généralement choisies par l'utilisateur mais peuvent aussi évoluer au cours de l'apprentissage
- Chaque nœud d'entrée est connecté à tous les nœuds de sortie, avec une pondération p_{ijk}

Activation d'un nœud



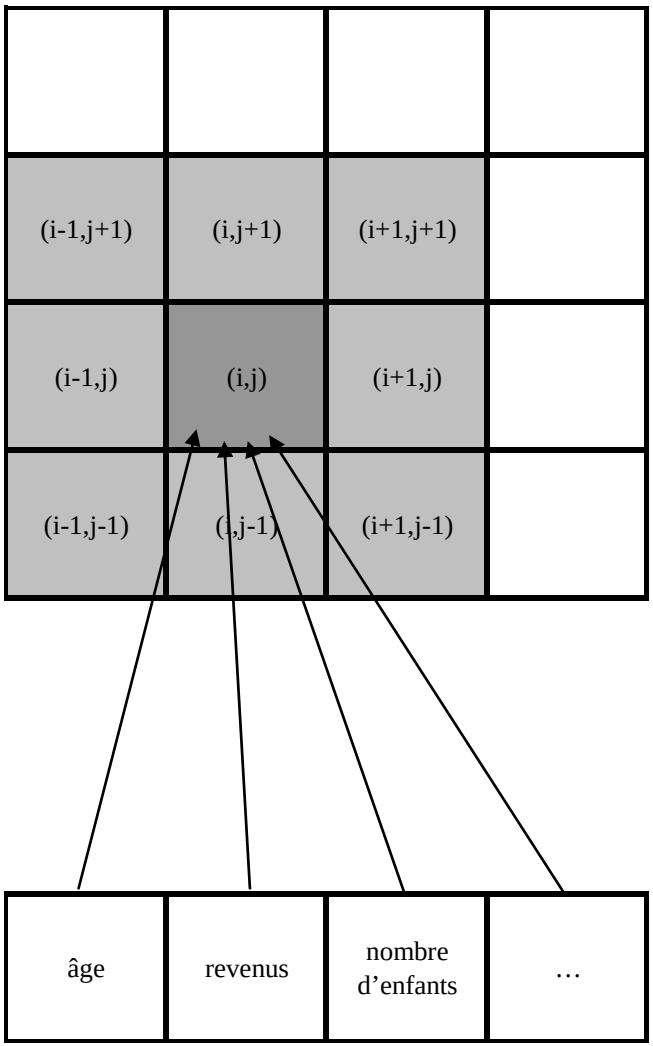
- Initialisation aléatoire des poids p_{ijk}

- La réponse d'un nœud (i,j) à un individu $(x_k)_{k\in[1,n]}$ est la distance euclidienne :

$$d_{ij}(x) = \sum_{k=1}^{n} \left(x_k - p_{ijk}\right)^2$$

- Le nœud retenu pour représenter (x_k) est le nœud pour lequel $d_{ij}(x)$ est minimum
- (i,j) et tous les nœuds voisins (I,J) voient leurs poids ajustés $p_{IJk} + \Theta.f(i,j;I,J).(x_k - p_{IJk})$ pour les rapprocher de (x_k)
  - Θ = taux d'apprentissage
  - f(i,j;I,J) = fct décroissante de la distance entre (i,j) et (I,J)
  - f(i,j;i,j) = 1

Apprentissage du réseau

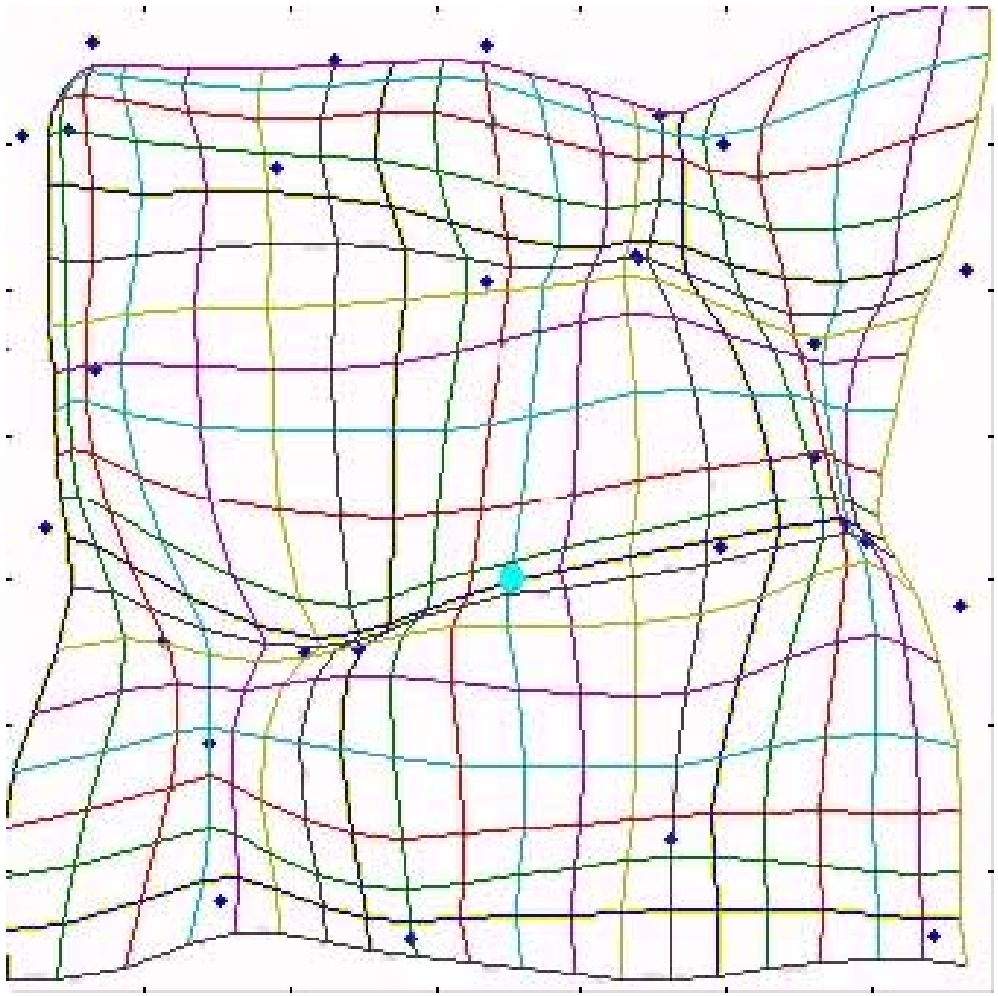


- Pour chaque individu, un seul nœud de sortie est activé (« le gagnant »)
- Le gagnant et ses voisins voient leurs poids ajustés
- En rapprochant les voisins, l'ajustement fait en sorte qu'à deux individus proches correspondent deux nœuds proches en sortie
- Des groupes (clusters) de nœuds se forment en sortie

Application d'un réseau de Kohonen

- Tout se passe comme si la grille du réseau était en caoutchouc et si on la déformait pour lui faire traverser le nuage des individus en s'approchant au plus près des individus.
  - ≠ avec un plan factoriel : c'est une projection non-linéaire
  - ≠ avec les autres méthodes de classification : réduction de la dimension
- Une fois que tous les individus de l'échantillonnage d'apprentissage ont été présentés au réseau et que tous les poids ont été ajustés, l'apprentissage est achevé.
- En phase d'application, le réseau de Kohonen fonctionne en représentant chaque individu en entrée par le nœud du réseau qui lui est le plus proche au sens de la distance définie ci-dessus. Ce nœud sera la classe de l'individu.

Représentation d'une carte de Kohonen



Utilisation des réseaux de Kohonen

- Synonymes : 1) carte de Kohonen 2) SOM (Self Organizing Map)
- Utilisation comme une « ACP » non linéaire
  - pour représenter sur une carte les groupes d'individus et comparer les groupes s'opposant sur la carte
- Utilisation comme pré-classification avant une CAH (voir la classification mixte)
  - on construit une carte de taille au moins 10 x 10 nœuds
  - on regroupe ensuite les 100 nœuds en un nb plus petit de classes qui doivent être connexes dans le plan
- Utilisation pour placer les prototypes d'un réseau de neurones RBF
- Éviter d'utiliser directement pour obtenir qq classes
  - voir les exemples suivants

Avantages des réseaux de neurones

- Aptitude à modéliser des structures complexes et des données irrégulières
  - prise en compte des relations non linéaires (interactions) entre les variables
- Assez bonne résistance aux données bruitées
- Aptitude à modéliser des problèmes très variés

Problèmes modélisés par les réseaux de neurones

- Analyse typologique
- Prédiction – classement
- Séries temporelles (prévision de cours boursiers)
- Reconnaissance de caractères optiques et de l'écriture manuscrite (sur des chèques, lettres, signatures)
- Reconnaissance/synthèse de la parole
- Jeu d'échecs (*Deep Blue* vainqueur de Kasparov en 1997)
- Reconnaissance des formes (prévention des pannes de machines par l'analyse de leurs vibrations)
- Reconnaissance des visages
- Analyse d'images (détecter si une gare est bondée)
- Traitement du signal
- Séries temporelles

Inconvénients des réseaux de neurones

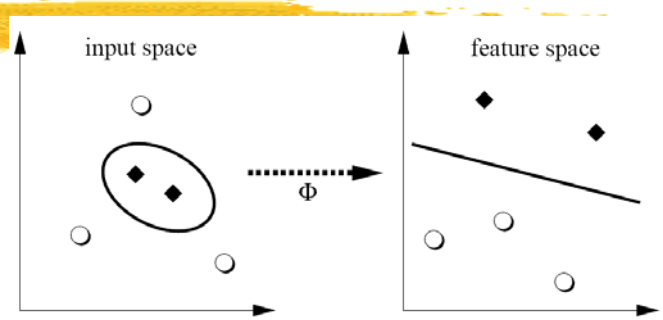
- Résultats totalement non explicites
  - rédhibitoire pour le diagnostic médical ou les pilotes automatiques d'avion
- Sensibilité aux individus hors norme
- Sensibilité à un trop grand nombre de variables non discriminantes
- Convergence vers la meilleure solution globale pas toujours assurée
- Difficulté d'utilisation correcte — paramètres nombreux et délicats à régler (nb et tailles des couches cachées, taux d'apprentissage, moment...)
- Ne s'appliquent naturellement qu'aux variables continues dans l'intervalle [0,1]
  - multiplication des nœuds pour les variables catégorielles

Nouvelle technique de classement :

Les Support Vector Machines

Séparateurs à Vaste Marge

Les Support Vector Machines



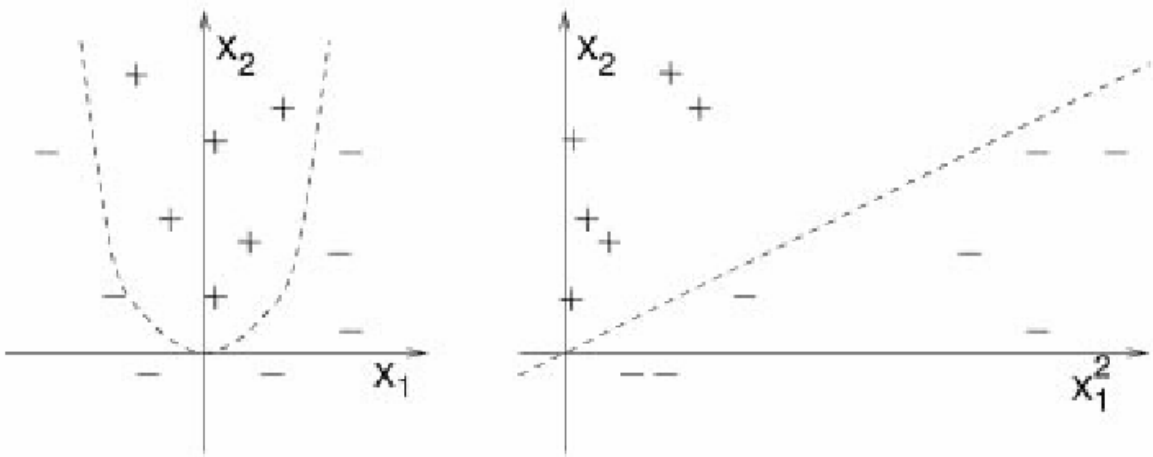
- Méthode récente (1998)
- Méthode de classement
- Deux étapes :
  - transformation non linéaire Φ pour passer dans un espace de dimension plus grande (voire infinie) que l'espace d'origine, mais doté d'un produit scalaire
  - dans cet espace, on cherche un séparateur linéaire f(x) = a·x+b (ex : fonction discriminante de Fisher), qui est un hyperplan optimal
    - séparant bien les groupes (précision du modèle)
      - $f(x) > 0 \Rightarrow$ classe A ; $f(x) \leq 0 \Rightarrow$ classe B
    - le + loin possible de tous les cas (robustesse du modèle)
  - on exprime $f(\Phi(x))$ sans faire intervenir explicitement Φ

Exemple de transformation

- Observer l'augmentation de la dimension

Input Space: $\vec{x} = (x_1, x_2)$ (2 Attributes)

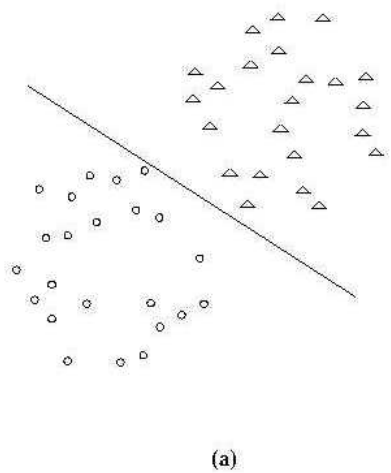
Feature Space: $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$ (6 Attributes)



Maximisation de la marge

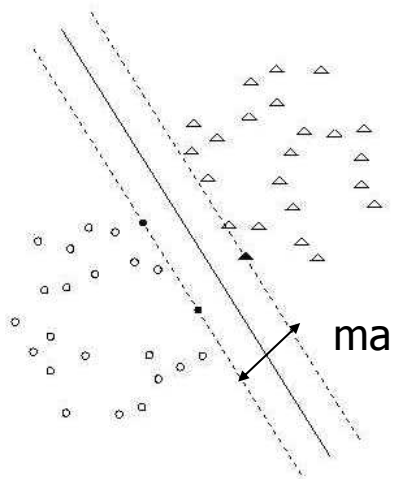
- Distance d'un point x à l'hyperplan = $\frac{|a \cdot x + b|}{\|a\|}$

séparation correcte



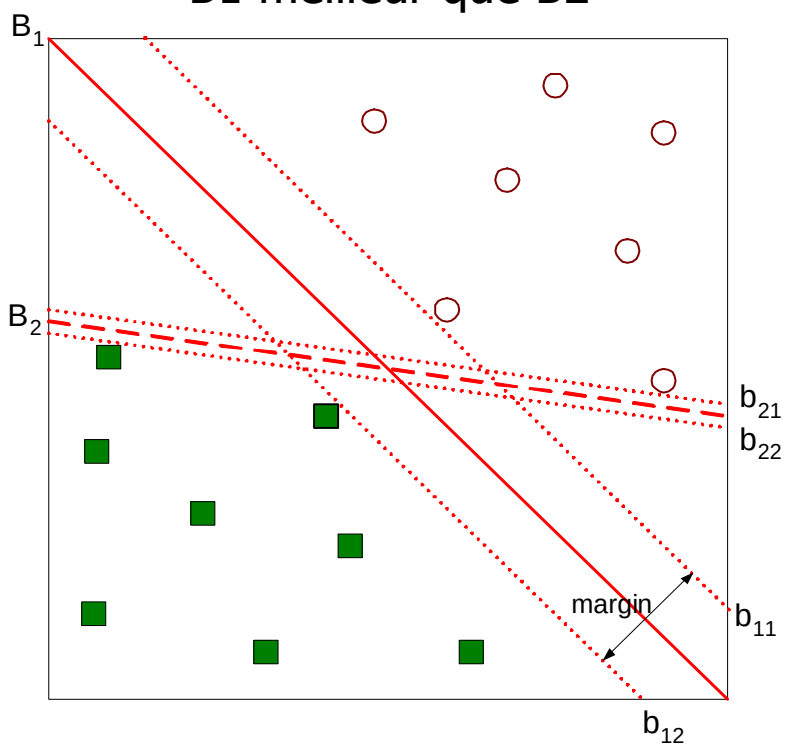
(a)

séparation optimale



(b)

B1 meilleur que B2



Forme de la solution

- Étant donnés les points (x_i,y_i), avec $y_i = 1$ si $x_i \in A$ et $y_i = -1$ si $x_i \in B$, trouver le séparateur linéaire $f(x) = a \cdot x+b \Leftrightarrow$ trouver (a,b) satisfaisant simultanément les 2 conditions :
  - pour tout i, $y_i(a \cdot x_i + b) \geq 1$ (bonne séparation)
  - $\|a\|^2$ est minimum (marge maximale)
- La sol. f(x) s'exprime en fonction de produits scalaires $x \cdot x'$
- Après transformation Φ, la solution s'exprime en fonction de produits scalaires $\Phi(x) \cdot \Phi(x')$
- La quantité $k(x,x') = \Phi(x) \cdot \Phi(x')$ est appelée noyau
- C'est k et non Φ que l'on choisit. En s'y prenant bien, on peut calculer k(x,x') sans faire apparaître Φ
- Les calculs sont alors faits dans l'espace de départ, et sont beaucoup + simples et + rapides

Exemples de noyaux

- Linéaire
  - $k(x,x') = x \cdot x'$
- Polynomial
  - $k(x,x') = (x \cdot x')^d$
  - si d = 2, $x=(x_1,x_2)$ et $\Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$, alors $\Phi(x) \cdot \Phi(x') = (x_1x_1' + x_2x_2')^2 = (x \cdot x')^2$
- Gaussien (RBF)
  - $k(x,x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}}$
- Sigmoïdal
  - $k(x,x') = \tanh \{\kappa(x \cdot x') + \theta\}$

- Intérêt des SVM : précision des prédictions

Nouvelle technique de data mining :

Algorithmes génétiques

Algorithmes génétiques

- Les algorithmes génétiques cherchent à reproduire les mécanismes de la sélection naturelle, en sélectionnant les règles (<=> gènes) les mieux adaptées à la prédiction, et en les croisant et mutant jusqu'à obtention d'un modèle suffisamment prédictif.

- Déroulement en 3 phases :
  - génération aléatoire des règles initiales
  - sélection des meilleures règles
  - génération de nouvelles règles par mutation ou croisement
    - la phase 3 bouclant sur la phase 2 jusqu'à la fin du déroulement de l'algorithme.

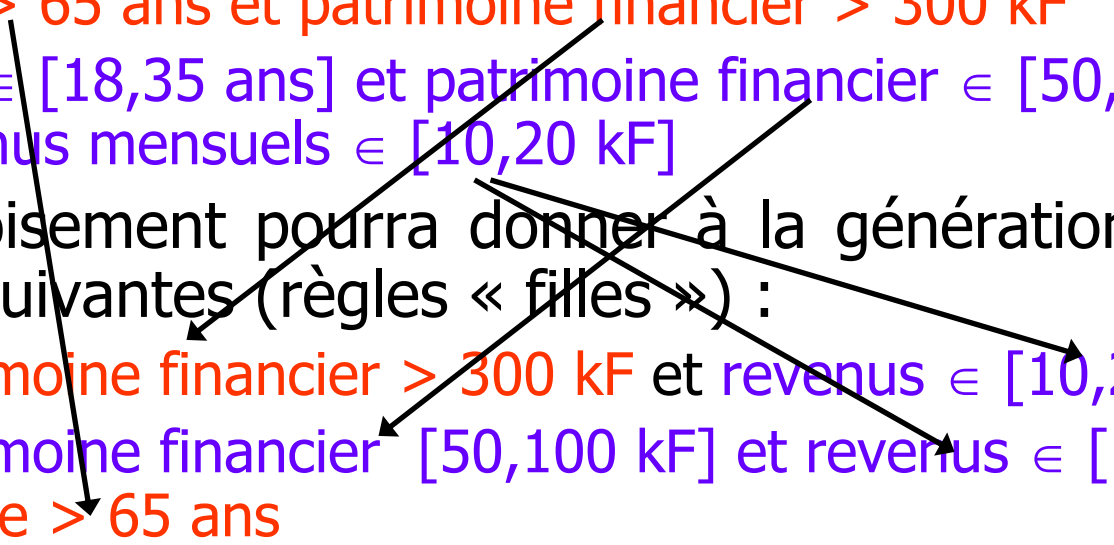
Génération initiale et sélection

- Les premières règles sont aléatoirement générées
- Elles doivent toutes être distinctes
- Chaque règle contient un nombre aléatoire de variables, chacune avec une modalité aléatoirement choisie.

- Les règles sont évaluées au vu de l'objectif à atteindre
- Les règles sont retenues avec une probabilité d'autant plus grande que la règle sera meilleure
- Les règles retenues doivent être satisfaites par un nombre minimum d'individus
- Certaines règles vont disparaître, tandis que d'autres seront sélectionnées plusieurs fois.

Croisement de règles

- Un *croisement* de deux règles (distinctes) est l'échange de certaines de leurs variables ou modalités pour donner deux nouvelles règles. Le croisement correspond à l'échange de place de deux sous-arbres.
- Si on prend les règles suivantes à la génération *n* :
  - âge > 65 ans et patrimoine financier > 300 kF
  - âge ∈ [18,35 ans] et patrimoine financier ∈ [50,100 kF] et revenus mensuels ∈ [10,20 kF]
- leur croisement pourra donner à la génération *n+1* les règles suivantes (règles « filles ») :
  - patrimoine financier > 300 kF et revenus ∈ [10,20 kF]
  - patrimoine financier ∈ [50,100 kF] et revenus ∈ [10,20 kF] et âge > 65 ans



Mutation d'une règle

- Une *mutation* est le remplacement par une autre d'une variable ou d'une modalité de la règle d'origine. Cela correspond au remplacement d'un nœud d'un arbre.
- Si on prend la règle suivante à la génération *n* :
  - âge $\in$ [36,50 ans] et revenus mensuels $\in$ [10,20 kF]
- une mutation pourra donner la règle « fille » :
  - âge $\in$ [36,50 ans] et patrimoine financier $\in$ [50,100 kF].

- Les mutations permettent de :
  - réintroduire des conditions intéressantes disparues par hasard
  - éviter une convergence trop précoce vers un optimum local.

Sélection des règles « filles »

- Les règles « filles » conservées pour être évaluées sont celles qui sont :
  - différentes des règles « mères »
  - différentes entre elles
  - satisfaites par un nombre minimum d'individus.
- Après évaluation, certaines des règles « filles » seront retenues pour poursuivre l'algorithme.
- L'algorithme s'achève lorsque :
  - un nombre préalablement fixé d'itérations a été atteint
  - OU : à partir d'une génération de rang n, les règles des générations n, n-1 et n-2 sont (presque) identiques.
- Le nombre d'itérations varie entre quelques dizaines et quelques centaines.

Utilisation des algorithmes génétiques

- Utilisation dans le calcul des poids des nœuds d'un réseau de neurones.
  - représenter l'ensemble de toutes les poids du réseau par un gène
  - partir de plusieurs ensembles de poids (c'est-à-dire : de gènes) possibles
  - sélectionner, croiser et muter les meilleurs gènes de génération en génération
  - jusqu'à obtention d'un ensemble de poids optimal.

- **Cet algorithme d'une grande complexité n'est utilisable que sur des volumes de données assez faibles.**