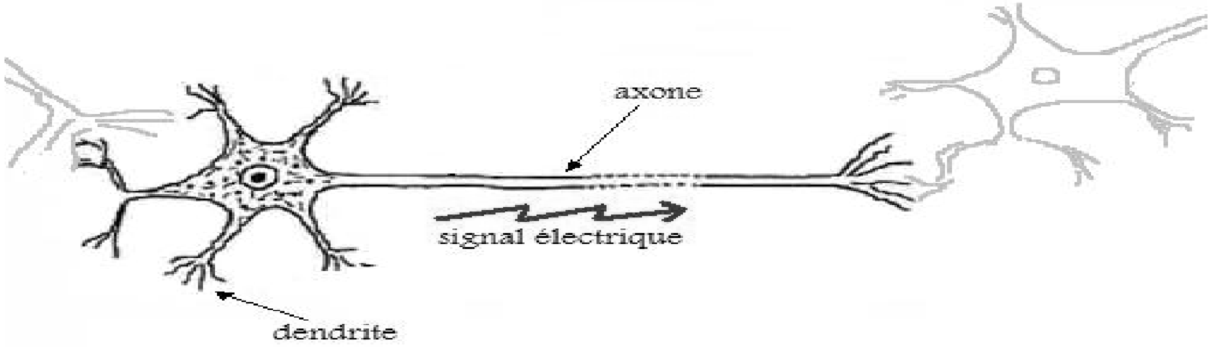
Réseaux de neurones articificiels
Perceptron simple et multi-couches

Application du réseaux de neurones à
l'apprentissage supervisé

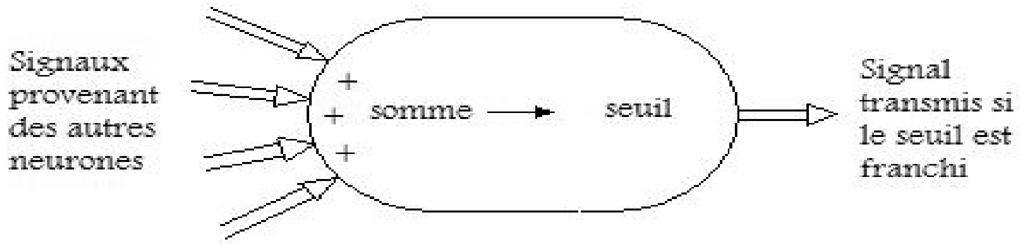
Ricco RAKOTOMALALA

Métaphore biologique

Fonctionnement du cerveau
Transmission de l'information et apprentissage



Idées maîtresses à retenir

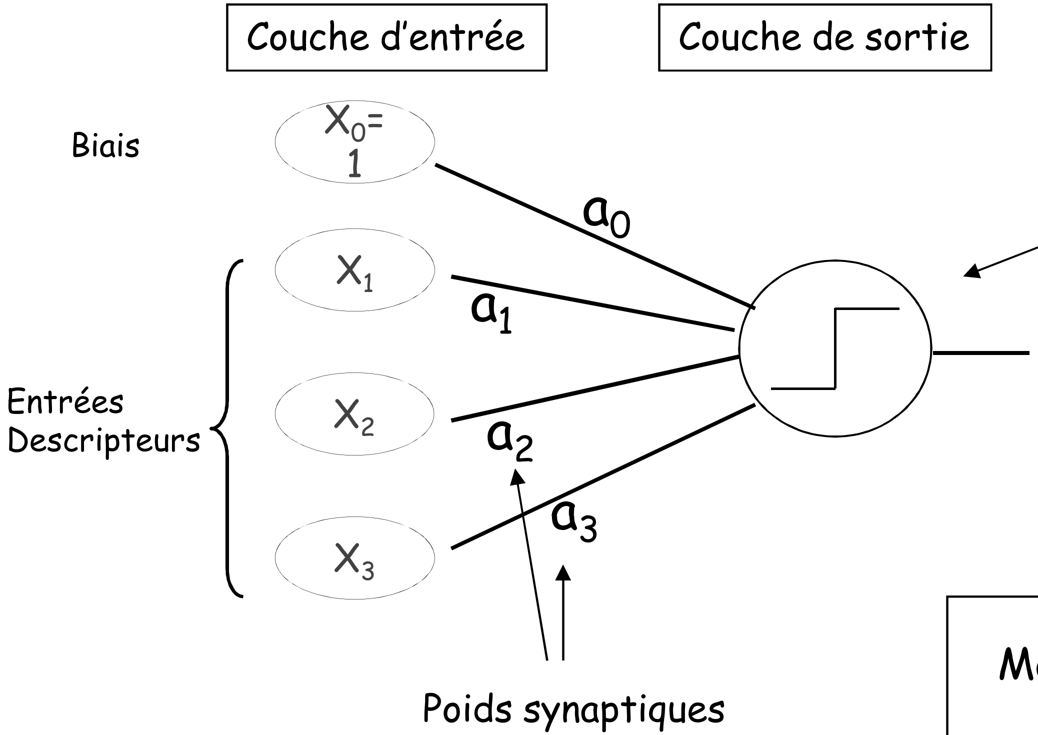


• Réception d'une information (signal)
• Activation + Traitement (simple) par un neurone
• Transmission aux autres neurones (si seuil franchi)
• A la longue : renforcement de certains liens → APPRENTISSAGE

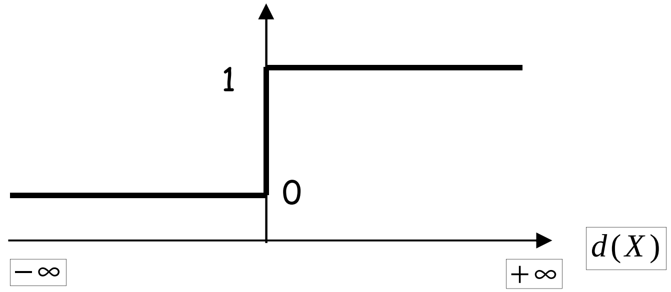
Modèle de Mc Colluch et Pitts
Le perceptron Simple

Problème à deux classes (positif et négatif)

$$Y \in \{1\ (+), 0\ (-)\}$$



Fonction de transfert
Fonction à seuil -- Fonction de Heaviside



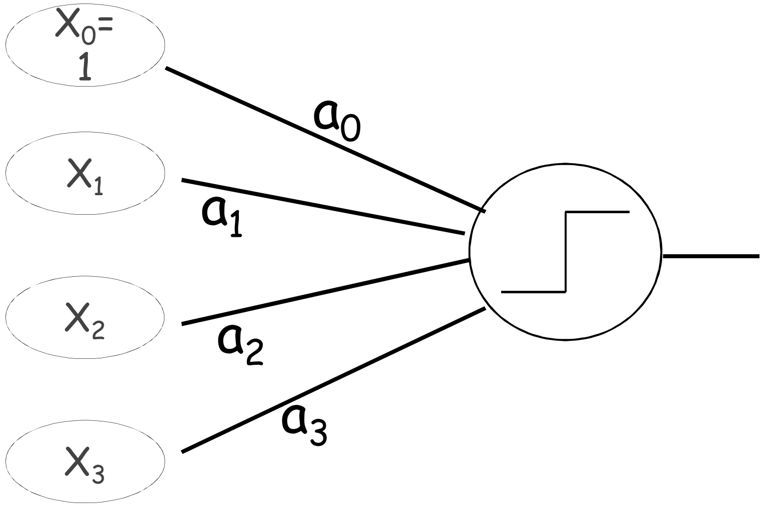
Modèle de prédiction et règle d'affectation

$$d(X) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3$$

$$\text{Si } d(X) > 0 \text{ Alors } Y = 1 \text{ Sinon } Y = 0$$

Le perceptron simple est un modèle de prédiction linéaire !

Apprentissage du perceptron simple



? Comment calculer les poids synaptiques à partir d'un fichier de données (Y ; X1, X2, X3)

Faire le parallèle avec la régression et les moindres carrés
Un réseau de neurones peut être utilisé pour la régression
(fonction de transfert avec une sortie linéaire)

(1) Quel critère optimiser ?

(2) Comment procéder à l'optimisation ?

(1) Minimiser l'erreur de prédiction

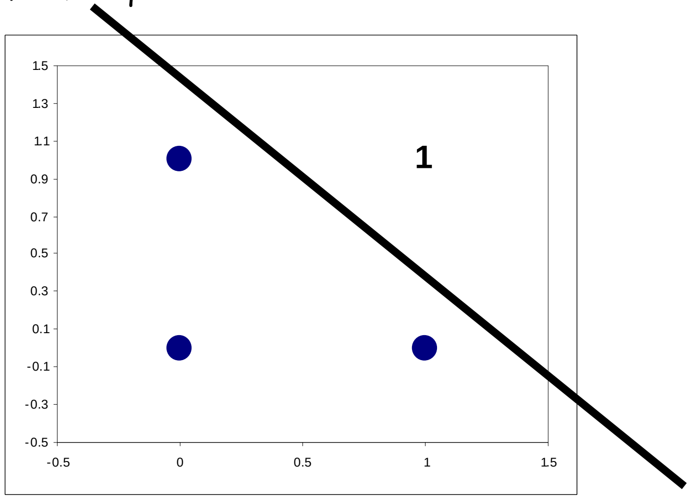
(2) Principe de l'incrémentalité

Exemple – Apprentissage de la fonction AND (ET logique)

Exemple révélateur – Les premières applications proviennent de l'informatique

X1	X2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Données



Représentation dans le plan

Principales étapes :

1. Mélanger aléatoirement les observations
2. Initialiser aléatoirement les poids synaptiques
3. Faire passer les observations unes à unes
    - Calculer l'erreur de prédiction pour l'observation
    - Mettre à jour les poids synaptiques
4. Jusqu'à convergence du processus

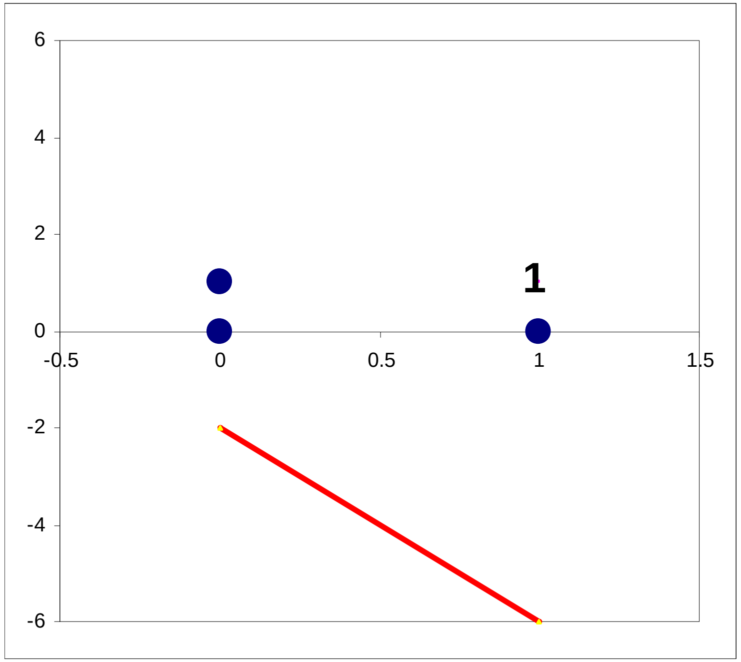
Une observation peut passer plusieurs fois !

Exemple AND (1)

Initialisation aléatoire des poids : $a_0 = 0.1; a_1 = 0.2; a_2 = 0.05$

Frontière :

$$0.1 + 0.2x_1 + 0.05x_2 = 0 \Leftrightarrow x2 = -4.0x_1 - 2.0$$



Règle de mise à jour des poids

Pour chaque individu que l'on fait passer (Principe de l'incrémentalité)

$$a_j \leftarrow a_j + \Delta a_j$$

avec

$$\Delta a_j = \eta(y - \hat{y})x_j$$

Force du signal

Erreur

Détermine s'il faut réagir ou non

Constante d'apprentissage

Détermine l'amplitude de l'apprentissage

Quelle est la bonne valeur ?

Trop petit → lenteur de convergence

Trop grand → oscillation

En général autour de 0.05 ~ 0.15 (0.1 dans notre exemple)

Exemple AND (2)

Observation à traiter

$$\begin{cases} x_0 = 1 \\ x_1 = 0 \\ x_2 = 0 \\ y = 0 \end{cases}$$

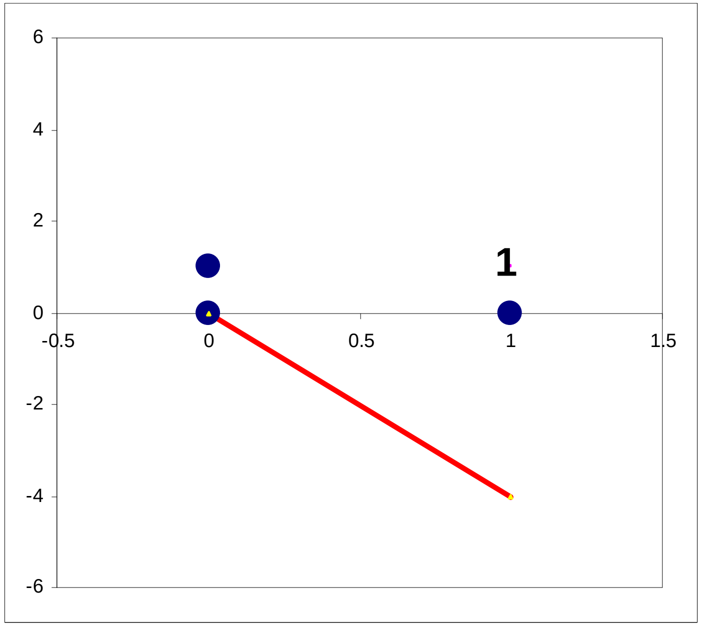
Appliquer le modèle

$$0.1\times1+0.2\times0+0.05\times0=0.1$$
$$\Rightarrow \hat{y}=1$$

Màj des poids

$$\begin{cases} \Delta a_0 = 0.1\times(-1)\times1 = -0.1 \\ \Delta a_1 = 0.1\times(-1)\times0 = 0 \\ \Delta a_2 = 0.1\times(-1)\times0 = 0 \end{cases}$$

Nouvelle frontière : $0.0 + 0.2x_1 + 0.05x_2 = 0 \Leftrightarrow x_2 = -4.0x_1 + 0.0$



Exemple AND (3)

Observation à traiter

$$\begin{cases} x_0 = 1 \\ x_1 = 1 \\ x_2 = 0 \\ y = 0 \end{cases}$$

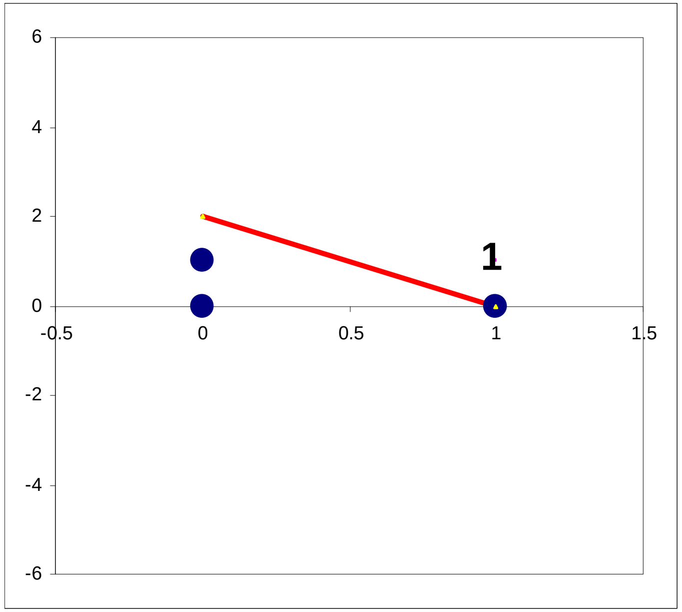
Appliquer le modèle

$$0.0 \times 1 + 0.2 \times 1 + 0.05 \times 0 = 0.2$$
$$\Rightarrow \hat{y} = 1$$

Màj des poids

$$\begin{cases} \Delta a_0 = 0.1 \times (-1) \times 1 = -0.1 \\ \Delta a_1 = 0.1 \times (-1) \times 1 = -0.1 \\ \Delta a_2 = 0.1 \times (-1) \times 0 = 0 \end{cases}$$

Nouvelle frontière : $-0.1 + 0.1x_1 + 0.05x_2 = 0 \Leftrightarrow x_2 = -2.0x_1 + 2.0$



Exemple AND (4) – Définir la convergence

Observation à traiter

$$\begin{cases} x_0 = 1 \\ x_1 = 0 \\ x_2 = 1 \\ y = 0 \end{cases}$$

Appliquer le modèle

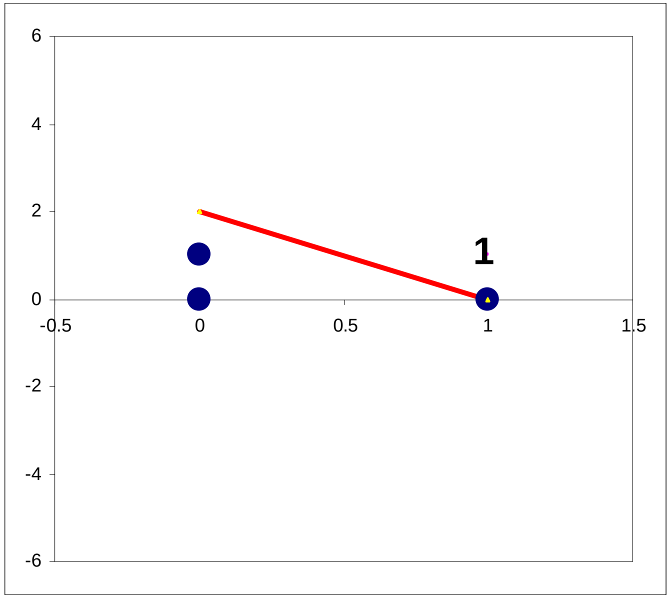
$$-0.1\times 1 + 0.1\times 0 + 0.05\times 1 = -0.05$$
$$\Rightarrow \hat{y} = 0$$

Màj des poids

$$\begin{cases} \Delta a_0 = 0.1\times(0)\times 1 = 0 \\ \Delta a_1 = 0.1\times(0)\times 0 = 0 \\ \Delta a_2 = 0.1\times(0)\times 1 = 0 \end{cases}$$

Pas de correction ici ? Pourquoi ?
Voir sa position dans le plan !

Nouvelle frontière : $-0.1 + 0.1x_1 + 0.05x_2 = 0 \Leftrightarrow x_2 = -2.0x_1 + 2.0$



Remarque : Que se passe-t-il si on repasse l'individu (x1=1 ; x2=0) ?

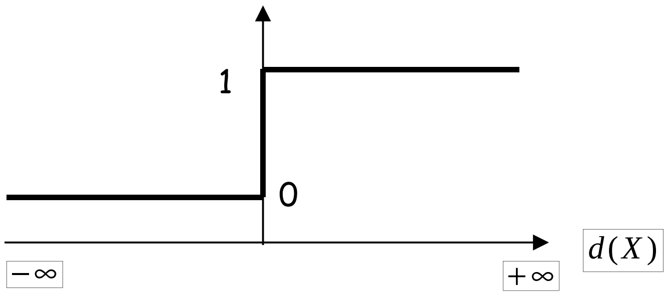
Convergence ?

(1) Plus aucune correction effectuée en passant tout le monde
(2) L'erreur globale ne diminue plus « significativement »
(3) Les poids sont stables
(4) On fixe un nombre maximum d'itérations
(5) On fixe une erreur minimale à atteindre

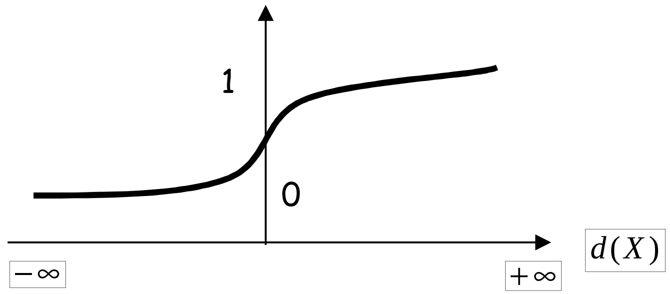
Évaluation de P(Y/X) – Fonction de transfert sigmoïde

Le Perceptron propose un classement Y/X
Dans certains cas, nous avons besoin de la probabilité P(Y/X) ex. Scoring

Fonction de transfert
Fonction à seuil -- Fonction de Heaviside



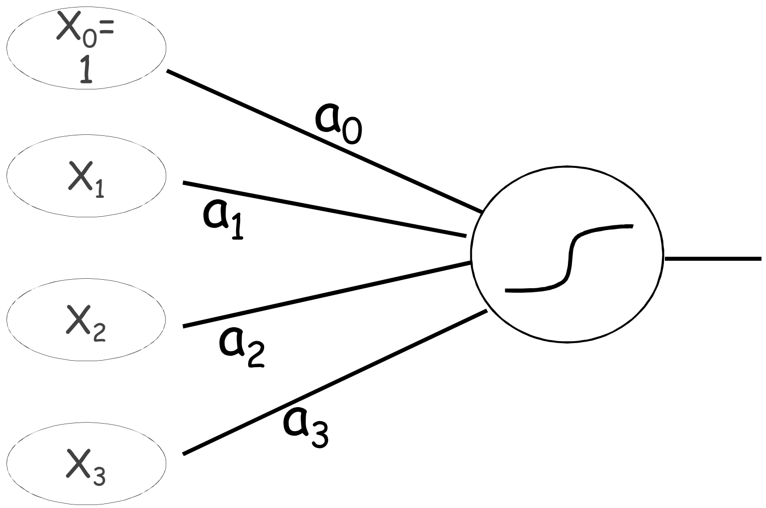
Fonction de transfert
Fonction sigmoïde – Fonction logistique



$$g(v) = \frac{1}{1+e^{-v}}$$
$$v = d(X)$$

La régle de décision devient : Si $g(v) > 0.5$ Alors Y=1 Sinon Y=0

Conséquences d'une fonction de transfert continue et dérivable
Modification du critère à optimiser



Sortie du réseau

$$\hat{y} = g(v) = f\left[d(x)\right]$$

Critère à optimiser : critère des moindres carrés

$$E = \frac{1}{2}\sum_{\omega \in \Omega}\left(y(\omega) - \hat{y}(\omega)\right)^2$$

Mais toujours fidèle au principe d'incrémentalité, l'optimisation est basé sur la descente du gradient !

Descente du gradient

Fonction de transfert sigmoïde dérivable

$$g'(v) = g(v)(1 - g(v))$$

Optimisation : dérivation de la fonction objectif par rapport aux coefficients

$$\frac{\partial E}{\partial a_j} = -\sum_i [y(\omega) - \hat{y}(\omega)] \times g'[v(\omega)] \times x_j(\omega)$$

Règle de mise à jour des coefficients pour un individu (Règle de Widrow-Hoff ou Règle Delta)

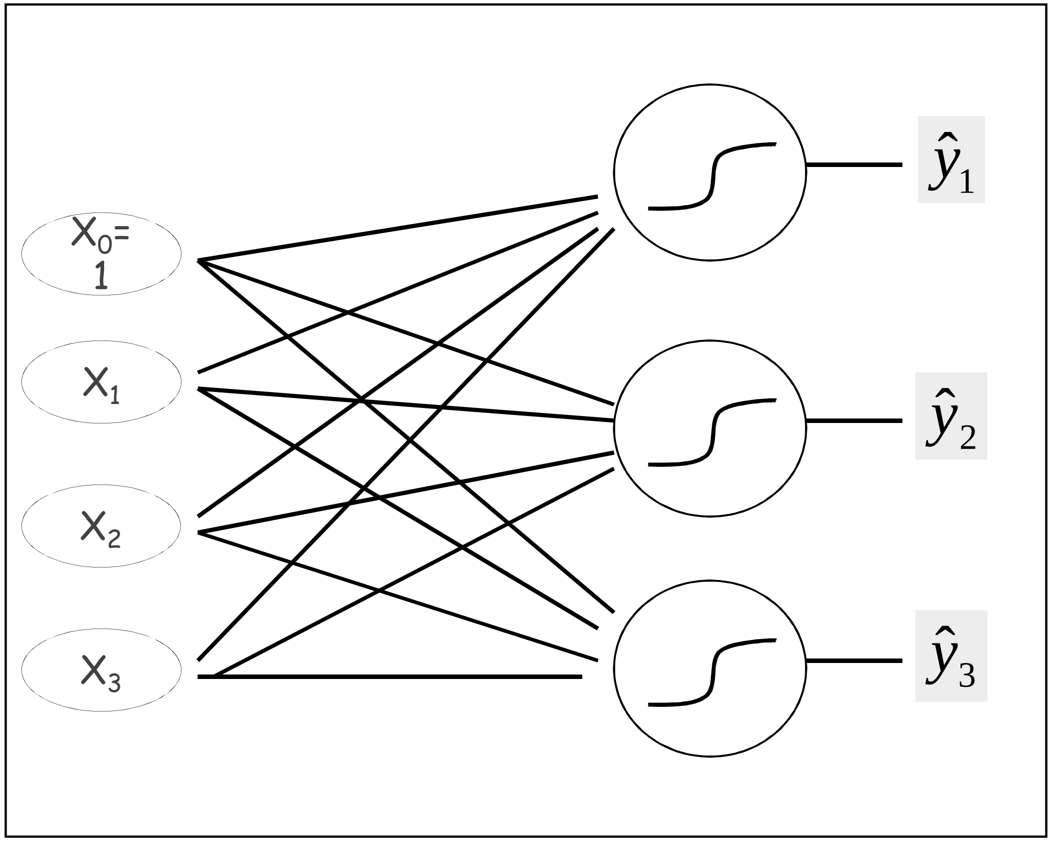
$$a_j \leftarrow a_j + \eta \underbrace{(y - \hat{y}) g'(v) x_j}$$

Gradient : màj des poids dans la direction qui minimise E

+

La convergence vers le minimum est bonne dans la pratique
Capacité à traiter des descripteurs corrélés (pas d'inversion de matrice)
Capacité à traiter des problèmes à très grande dimension (lignes x colonnes)
Mise à jour facile si ajout de nouveaux individus dans la base

Problème à K classes -- Que faire quand Y possède plus de 2 modalités ?



(1) Codage disjonctif complet de la sortie

$$y_k = 1 \;\; ssi \;\; y = y_k$$

(2) « Output » pour chaque sortie

$$\hat{y}_k = g[v_k]$$

avec $v_k = a_{0,k} + a_{1,k} x_1 + \cdots + a_{J,k} x_J$

(3) P(Y/X) $P(Y = y_k \,/\, X) \propto g[v_k]$

(4) Règle de décision

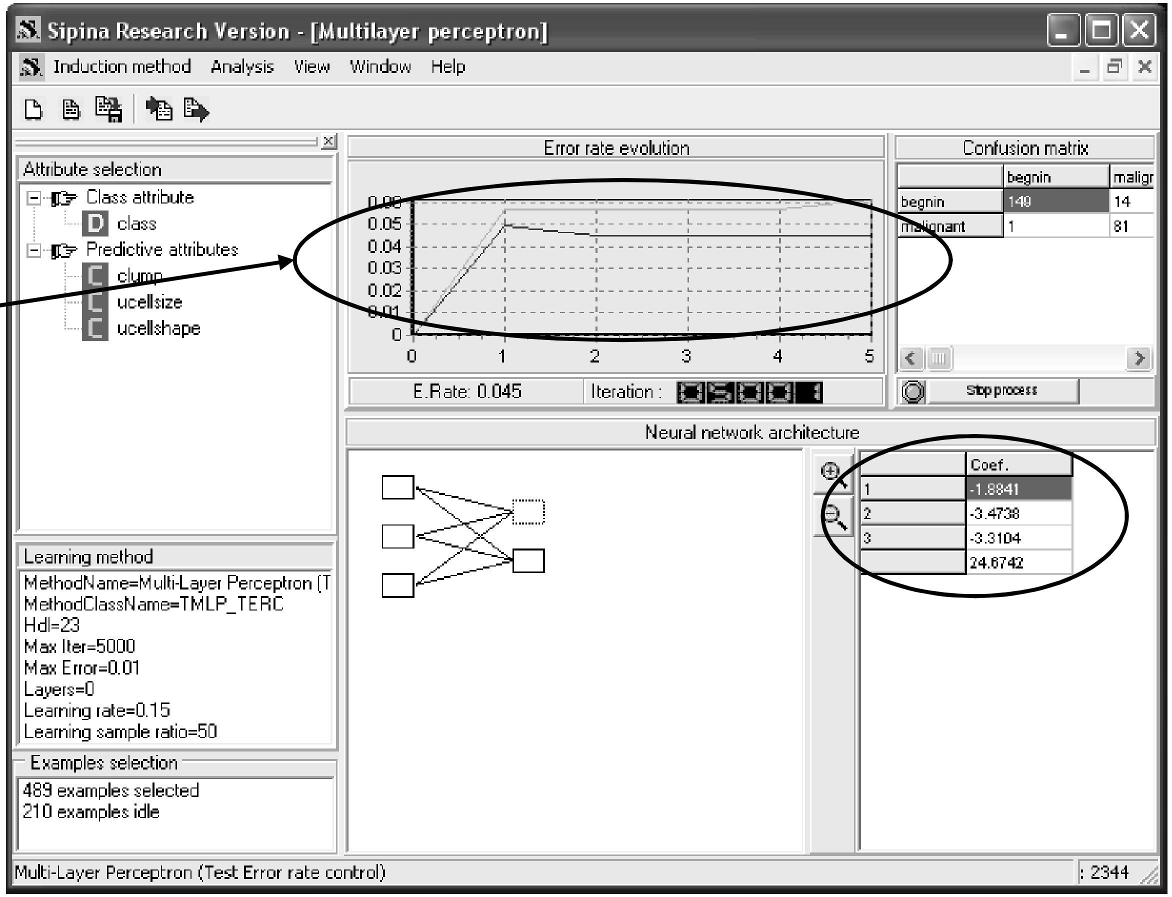
$$\hat{y} = y_{k^*} \;\; ssi \;\; k^* = \arg\max_k \hat{y}_k$$

Minimisation de l'erreur quadratique
En traitant K réseaux en parallèle !

$$E = \frac{1}{2} \sum_{\omega} \sum_{k=1}^{K} \left(y_k(\omega) - \hat{y}_k(\omega) \right)^2$$

Pratique du Perceptron (Démo sur les cancers du sein)

Évolution erreur



Poids

Quelques conseils

Ramener les descripteurs sur la même échelle
Standardisation, Normalisation, etc.

(Éventuellement) Subdiviser les données en 3 parties :
Training + Validation + Test

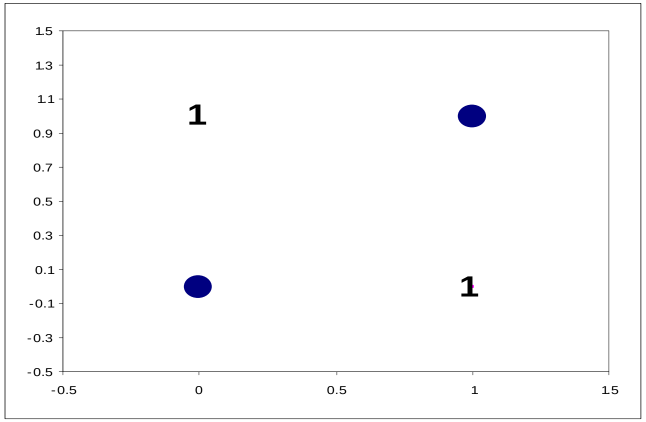
Attention au paramétrage, notamment de la règle d'arrêt

Mort et résurrection du Perceptron – Le problème du XOR

(1)

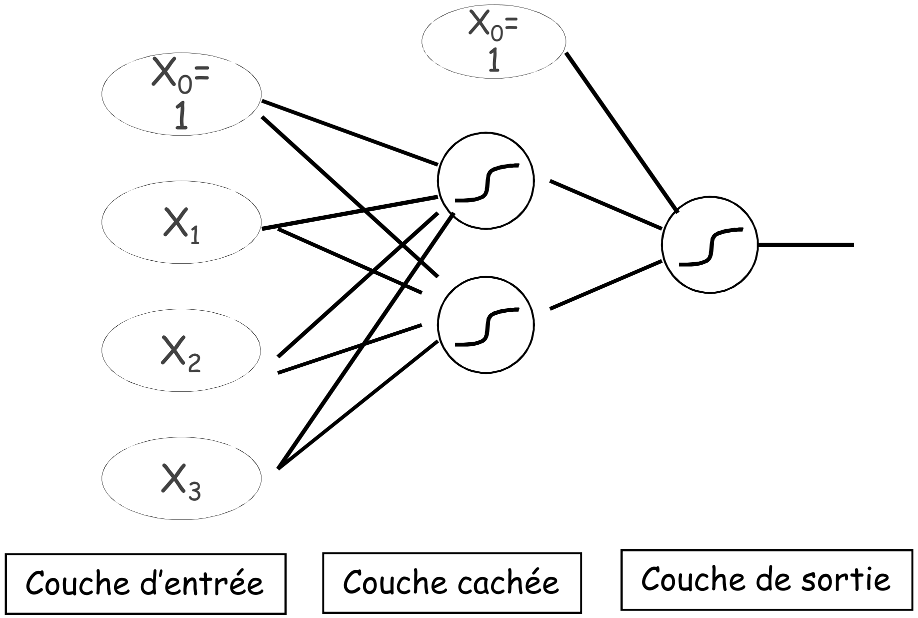
X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Données



Non séparable linéairement (Minsky & Papert, 1969)

(2)



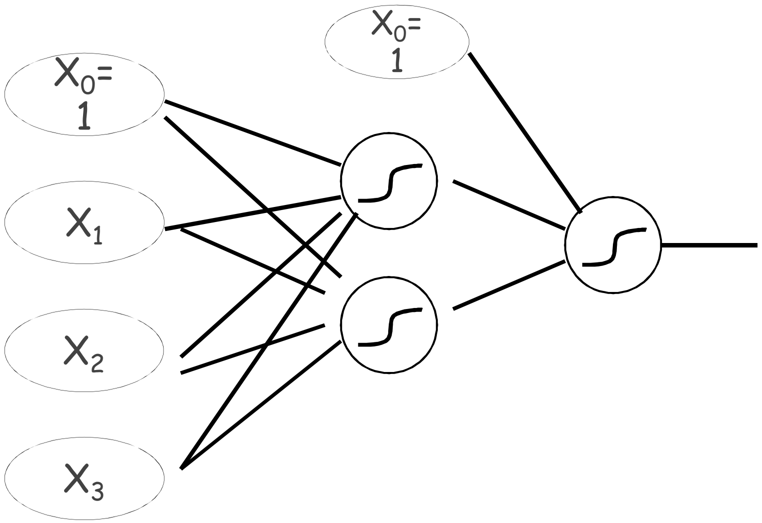
Une combinaison de séparateurs linéaires permet de produire un séparateur global non-linéaire **!**

(Rumelhart, 1986)

Perceptron Multi-Couches (PMC)

On peut avoir plusieurs couches cachées

PMC – Formules et propriétés



Passage C.Entrée → C.Cachée

$$v_1 = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3$$
$$v_2 = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$$

Sortie de la C.Cachée

$$u_1 = g(v_1) = \frac{1}{1 + e^{-v_1}}$$
$$u_2 = g(v_2) = \frac{1}{1 + e^{-v_2}}$$

Passage C.Cachée → C.Sortie

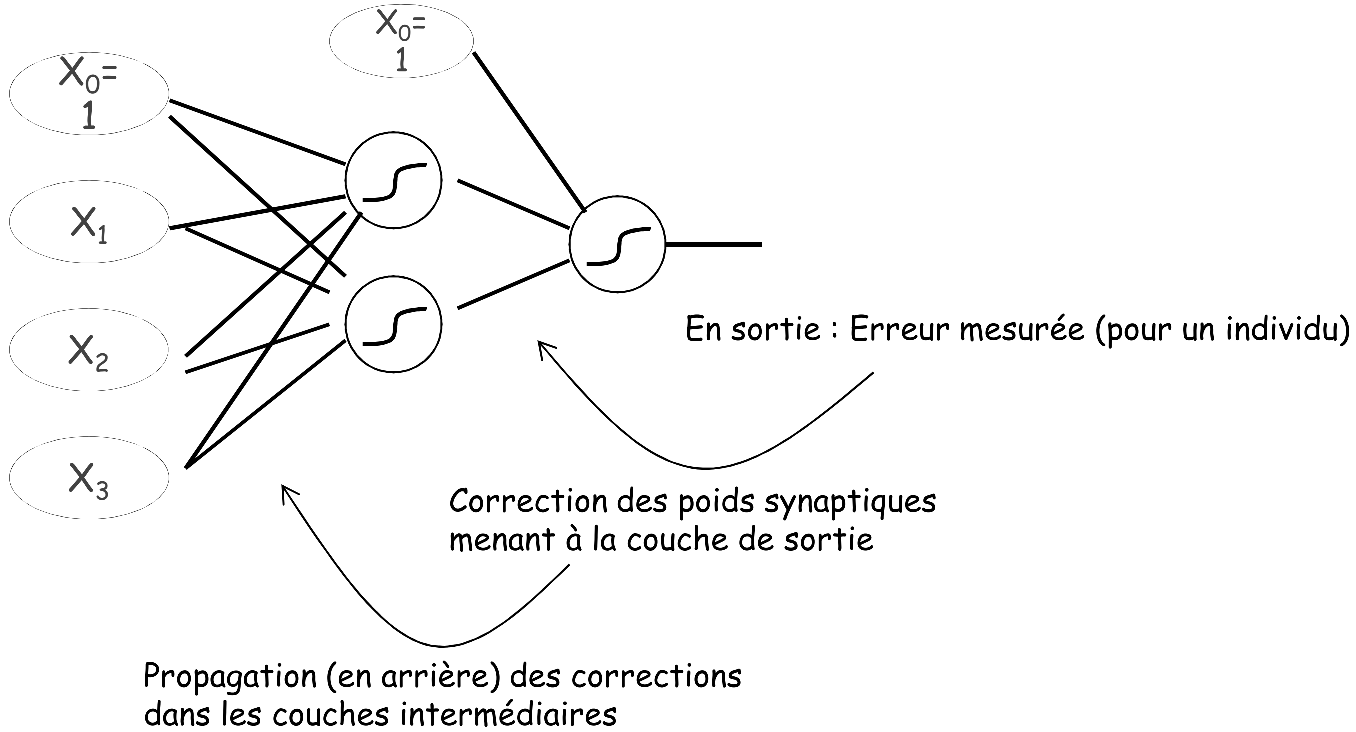
$$z = c_0 + c_1 u_1 + c_2 u_2$$

Sortie du réseau

$$\hat{y} = g(z) = \frac{1}{1 + e^{-z}}$$

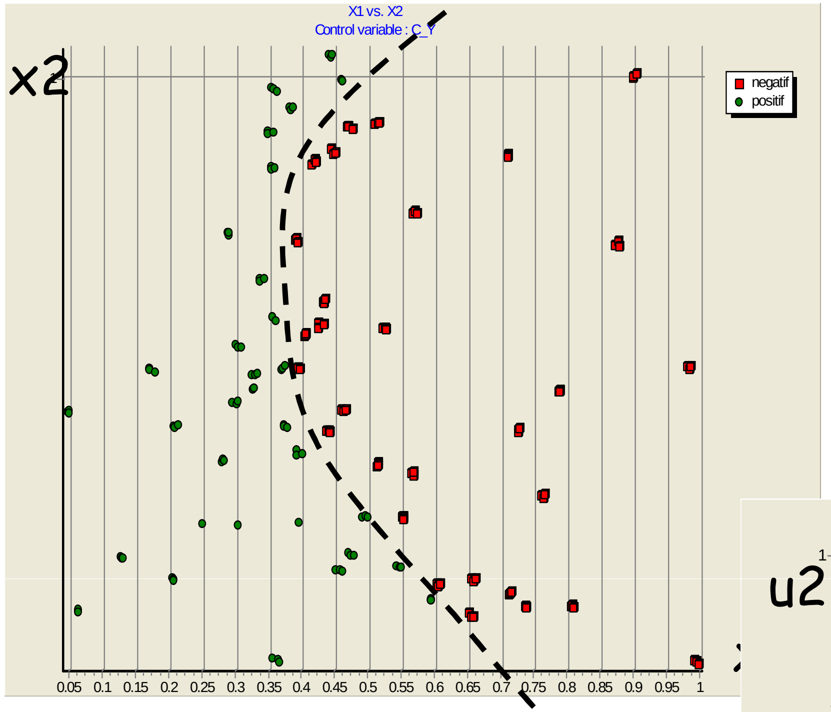
Propriété fondamentale : Le PMC est capable d'approximer toute fonction booléenne existante pourvu que l'on fixe convenablement le nombre de neurones dans la couche cachée **!**

Généraliser la règle de Widrow-Hoff – La descente du gradient

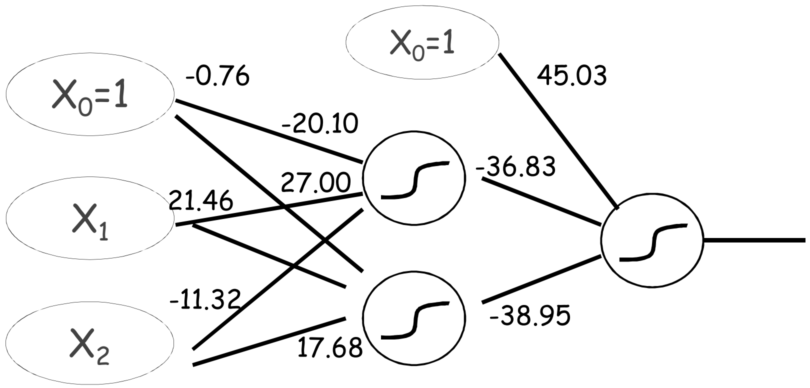


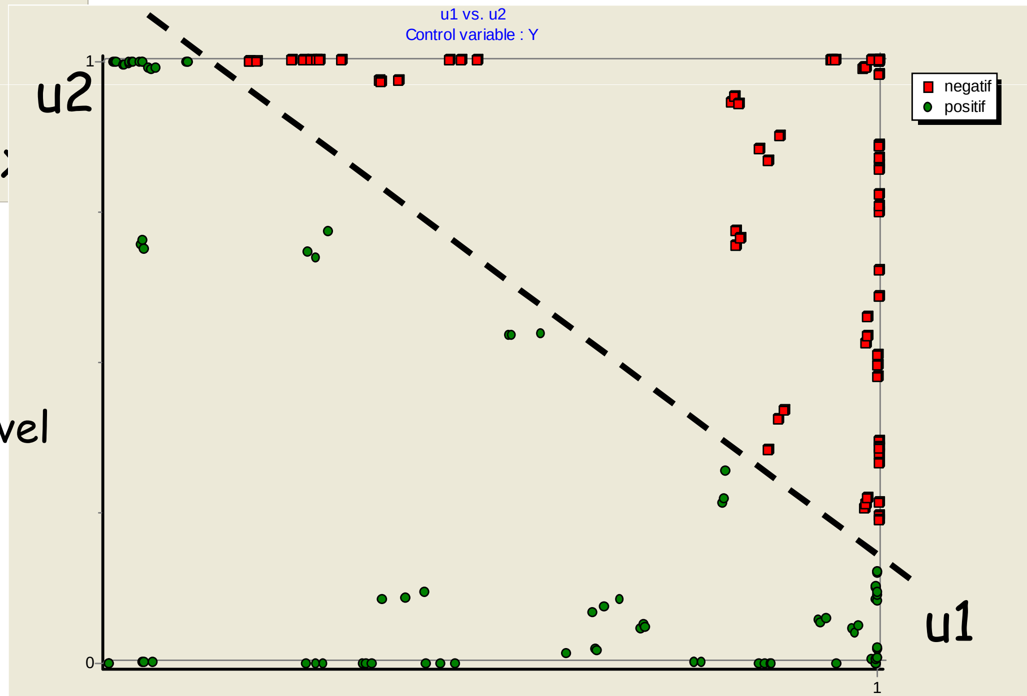
! L'algorithme de la rétro-propagation du gradient donne de bons résultats dans la pratique même si le risque de stagnation dans un optimum local n'est pas à négliger → normaliser ou standardiser impérativement les données et bien choisir la constante d'apprentissage

PMC – Un exemple de discrimination non linéaire



Construisons un réseau avec 2 couches cachées





La sortie des neurones de la couche cachée propose un nouvel espace de représentation où il est possible de discriminer linéairement les exemples !

PMC – Avantages et inconvénients

+

Classifieur très précis (si bien paramétré)

Incrémentalité

Scalabilité (capacité à être mis en œuvre sur de grandes bases)

-

Modèle boîte noire (causalité descripteur – variable à prédire)

Difficulté de paramétrage (nombre de neurones dans la couche cachée)

Problème de convergence (optimum local)

Danger de sur-apprentissage (utiliser impérativement un fichier de validation)

Références

Technique

• « Neural network »
Tutorial slides of Andrew Morre
http://www.autonlab.org/tutorials/neural.html

• « Apprentissage à partir d'exemples »
Notes de cours F. Denis & R. Gilleron – Lille 3
http://www.grappa.univ-lille3.fr/polys/apprentissage/

• « Machine Learning »
Tom Mitchell, Ed. Mc Graw-Hill International, 1997.

Culturel

• « Réseaux de neurones » sur WIKIPEDIA

• « Réseaux de neurones – Méthodologie et Applications »
Sous la direction de Gérard Dreyfus, Ed. Eyrolles, 2004.