Réseaux neuromimétiques

Alain Grumbach
Jean Marie Nicolas

SI221 : Méthodes de classification

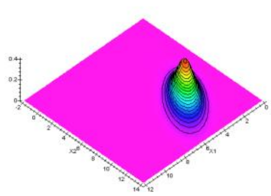
- Bayes : Connaissance des lois
  - Lois connues
- Automatique: Non hiérarchique
- Méthodes structurelles

Classification bayésienne : rappels

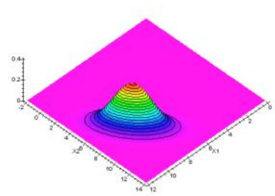
- **R échantillons indépendants : X_p, $p \in [1,R]$**
  - Vecteurs d'état de dimension N : $X_{i,p}$, $i \in [1,N]$, $p \in [1,R]$
- **Classification en *c* classes**
  - Le nombre de classes *c* est connu
  - On connaît les lois pour chaque classe
- **Pour chaque échantillon p, on connaît la sortie désirée (étiquette) : d_p, $p \in [1,R]$**

- **On connaît tout sur tout**

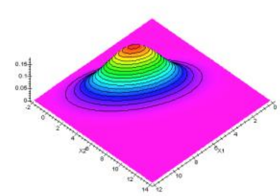
Trois classes, lois gaussiennes



$$m_1 = \begin{pmatrix} 4 \\ 9 \end{pmatrix} \qquad s_1 = \begin{pmatrix} 2 & 2 \\ 2 & 5 \end{pmatrix}$$

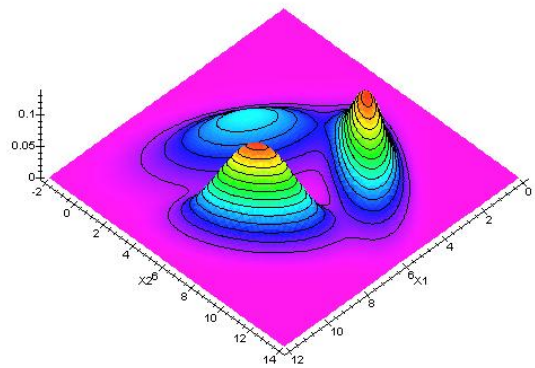


$$m_1 = \begin{pmatrix} 8.5 \\ 7.5 \end{pmatrix} \qquad s_1 = \begin{pmatrix} 2 & -2 \\ -2 & 5 \end{pmatrix}$$

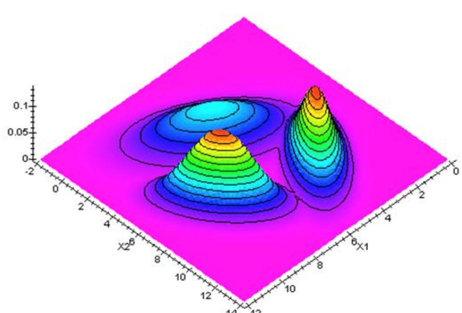


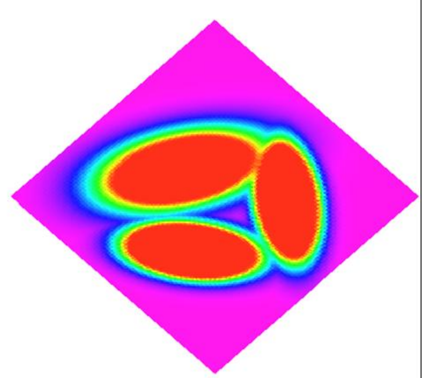
$$m_1 = \begin{pmatrix} 6 \\ 3.5 \end{pmatrix} \qquad s_1 = \begin{pmatrix} 7 & -4 \\ -4 & 7 \end{pmatrix}$$

Mélange équiprobable (P(ω_i)=1/3)

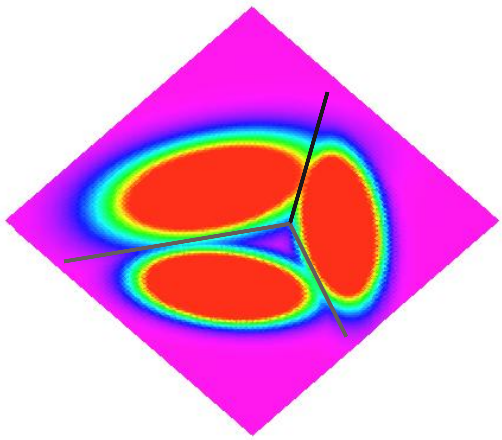


Décision bayésienne





Décision bayésienne : simplification

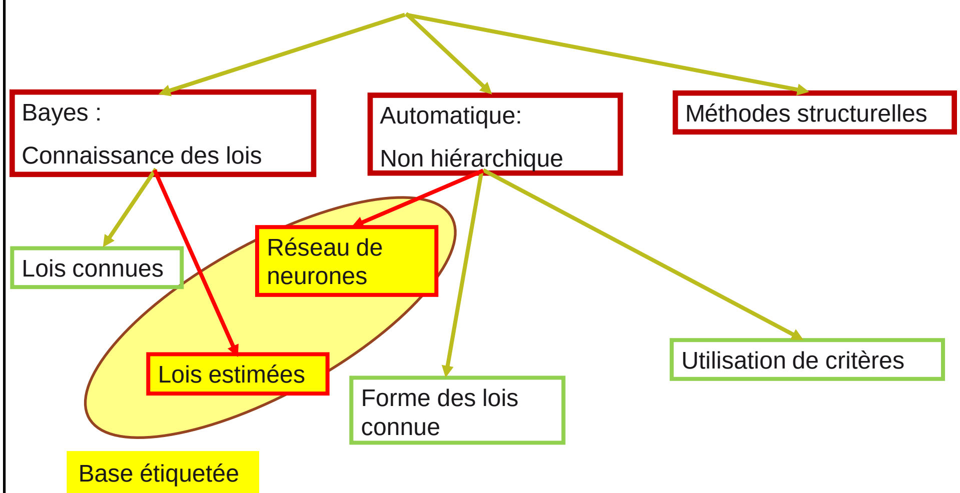


Approximation : séparatrice linéaire ?

Classification bayésienne

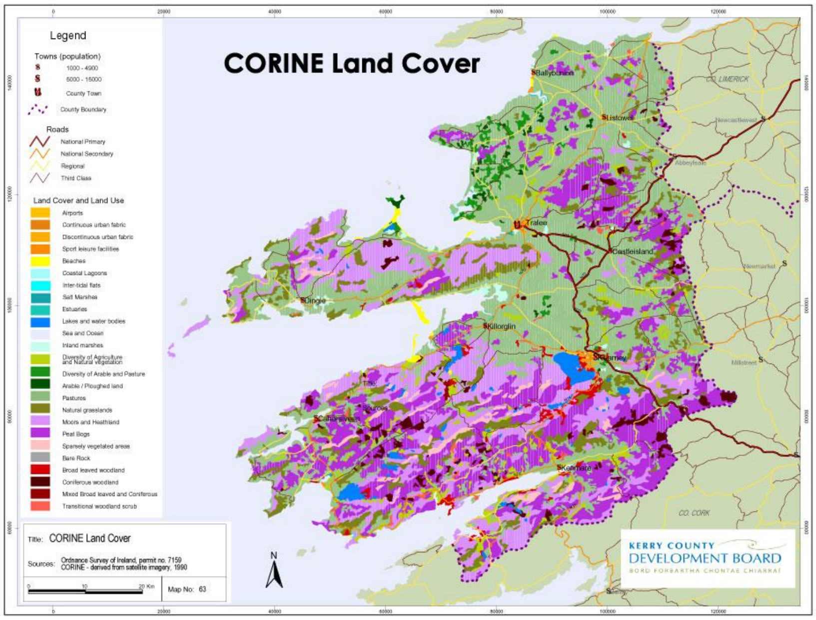
- **R échantillons indépendants : X_p, $p \in [1,R]$**
  - Vecteurs d'état de dimension d : $X_{i,p}$, $i \in [1,N]$
- **Classification en *c* classes**
  - Le nombre de classes *c* est connu
  - ~~On connaît les lois pour chaque classe~~
- **Pour chaque échantillon, on connaît la sortie désirée : $d_p, p \in [1,R]$**
  - Bases d'apprentissage et de test

Méthodes de classification

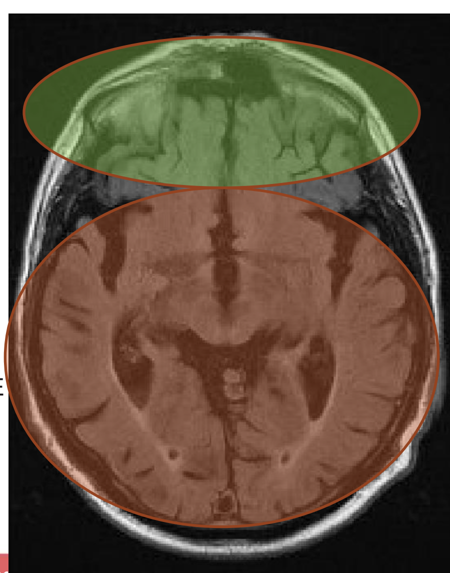


Base servant à l'apprentissage

- **Le cadre des réseaux neuromimétiques :**
  - Aucune hypothèse sur les données
  - Existence d'une base servant à l'apprentissage :
    - Pour chaque échantillon, on connaît la sortie désirée
    - Rôle essentiel de l'expert
- **Possible découpage de la base servant à l'apprentissage :**
  - Apprentissage
  - Reconnaissance (ou généralisation)
  - Test sur données non utilisées en apprentissage et en reconnaissance!!



Exemple sur un IRM de cerveau

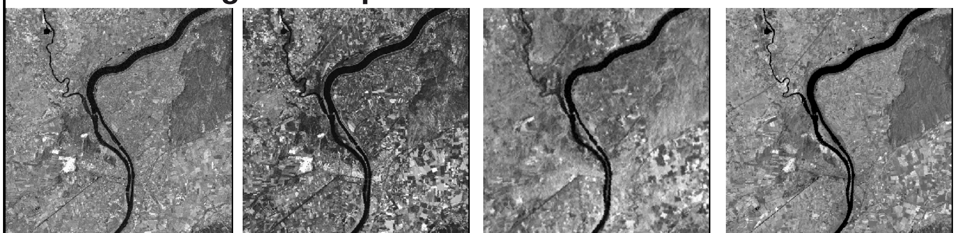


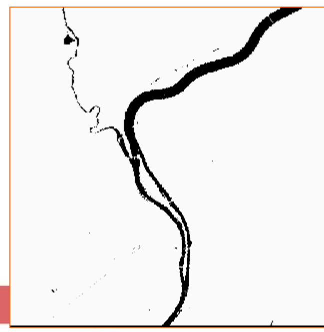
BASE DE GENERALISATION

BASE D'APPRENTISSAGE

Le TP « Réseaux de neurones »

- **Une image multispectrale Landsat sur Tarascon**



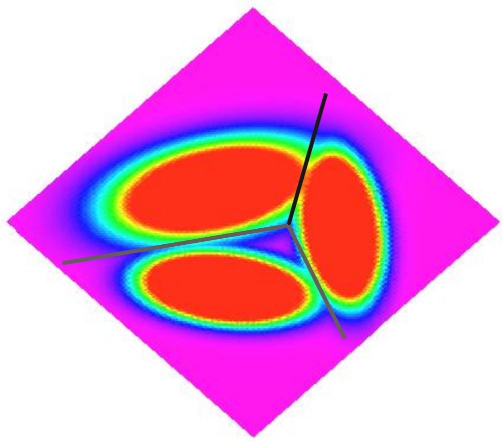


Rôle de ces bases

- **La base d'apprentissage :**
  - On compare la sortie désirée et la sortie calculée
  - L'erreur permet de trouver la bonne architecture du système de classification

- **La base de reconnaissance (généralisation) :**
  - On compare la sortie désirée et la sortie calculée
  - L'erreur n'intervient pas dans la recherche de la meilleure architecture
  - L'erreur permet de juger des performances de l'architecture sur des individus n'ayant pas servi à définir l'architecture

Recherche d'une séparatrice linéaire



- **Apprentissage :** Définir les séparatrices linéaires
- **Reconnaissance :**
  - Tester la qualité de la qualité de la classification
  - Remettre en cause l'apprentissage

Un séparateur linéaire « automatique »
Le Perceptron

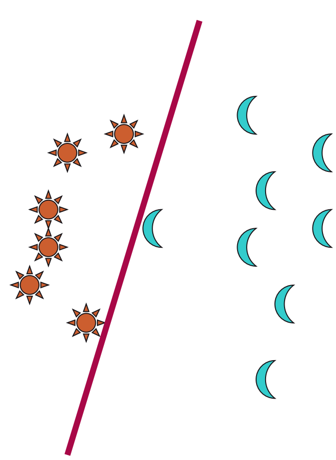
Un séparateur linéaire : le perceptron

- **Séparation par hyperplans**
  - Problèmes à c classes : c(c-1)/2 hyperplans
- **Problème à deux classes : 1 hyperplan**
  - Classe C^+ et classe C^-
  - Séparation par l'hyperplan : $\boxed{W^t x + b = 0}$
  - Propriété souhaitée

$$\forall x \in C^+ \; W^t x + b > 0$$

$$\forall x \in C^- \; W^t x + b < 0$$

Exemple linéairement séparable



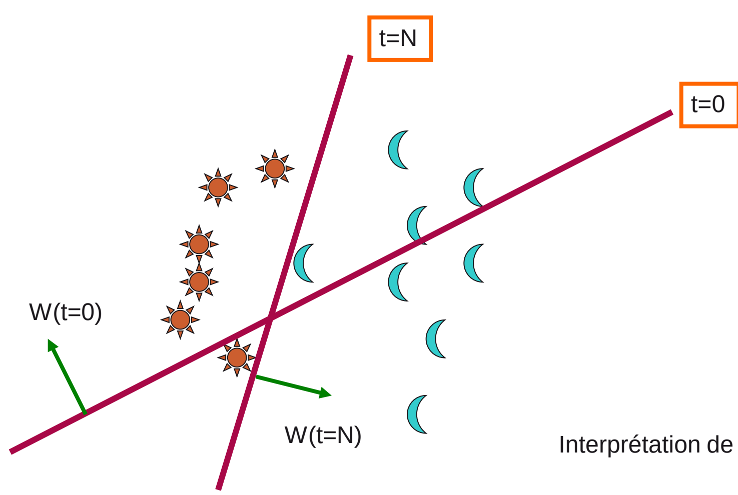
Prolégomène : la règle du perceptron

- **Règle du Perceptron (Rosenblatt, 1957) :**
  - Initialiser les poids
  - Modifier itérativement les poids si la sortie n'est pas égale à la sortie désirée par la règle

$$\boxed{W(t+1) = W(t) + \eta X}$$

- **W définit un hyperplan séparateur si le problème est linéairement séparable**

Exemple linéairement séparable



Interprétation de W

Test d'arrêt

- **Lorsque tous les vecteurs sont bien classés**
  - Ceci requiert que le problème soit linéairement séparable
- **Lorsque tous les vecteurs sont « à peu près » bien classés**
  - Ceci requiert la définition d'une métrique (définition de l'erreur)
- **Au bout d'un certain nombre d'itérations (« époques »).**

Erreur de classification

- **Très classiquement une erreur quadratique :**
  - Expression bien connue
  - Expression dérivable

$$J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k} \right)^2$$

- **Ensemble des mal classés : Y(W)**

$$\tilde{J}(W) = \sum_{\substack{k=1 \\ k \in Y(W)}}^{R} \left(W^t X_k - d_{X_k} \right)^2$$

  - Non dérivable (fonction d'appartenance)

Critères sur la qualité de la classification : base d'apprentissage

- **Nombre de bien classés :**
  - Taux de bonne classification : τ_A
  - R_A individus :

$$\boxed{\tau_A = \frac{\text{Nombre d'individus bien classés}}{R_A}}$$

- **Erreur quadratique :**
  - Diminue pour la base d'apprentissage : n'arrive jamais à 0 en pratique !!
- **En général, sur une base d'apprentissage, on peut arriver à des valeurs proches de 100% pour τ !!**

Critères sur la qualité de la classification : base de reconnaissance (généralisation)

- **Utiliser l'architecture définie par l'utilisation de la base d'apprentissage**
- **Nombre de bien classés :**
  - Taux de bonne classification : τ_G
  - R_G individus :

$$\boxed{\tau_G = \frac{\text{Nombre d'individus bien classés}}{R_G}}$$

- **Erreur quadratique :**
  - Au début de l'apprentissage : diminue
  - Ensuite : peut remonter !!
  - Notion de sur-apprentissage

Critères sur la qualité de la classification : base de reconnaissance (généralisation)

- **Utiliser le perceptron défini sur la base d'apprentissage**
- **Nombre de bien classés :**
  - Taux en % de bonne classification
- **Erreur quadratique sur la base de test :**
  - Diminue d'abord durant l'apprentissage
  - MAIS peut remonter : « surapprentissage »

Mise en œuvre (cf TP)

$$\boxed{W^t x + b = 0}$$

- **Vecteur d'entrée : x (dimension N)**
- **Séparatrice linéaire : vecteur a (dimension N)**
- **Prise en compte d'un offset**
- **Un artifice de réalisation : rajouter une dimension**

$$\boxed{\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \rightarrow \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{pmatrix}}$$

L'offset b passe dans le vecteur

$$\boxed{W^t x + b = 0}$$

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \rightarrow \vec{\tilde{x}} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} w_1 & w_2 & \dots & w_n & -b \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{pmatrix} = 0$$

Interprétation connectionniste

- Equation de l'hyperplan

$$W^t x + b = 0$$

- Si X appartient à la classe C⁺

$$W^t X + b > 0$$

- Si X appartient à la classe C⁻

$$W^t X + b < 0$$

Interprétation connexioniste

- Equation de l'hyperplan

$$\boxed{W^t x + b = 0}$$

- Ecriture en somme pondérée

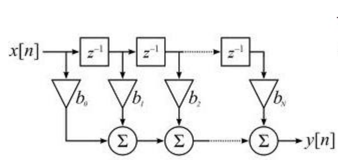
$$\boxed{a = \sum_{k=1}^{N+1} w_k \tilde{x}_k}$$

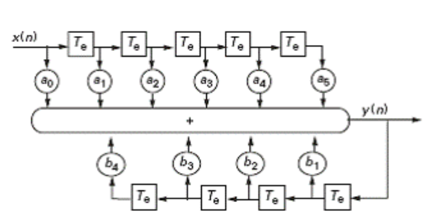
$$\boxed{\vec{\tilde{x}} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \\ 1 \end{pmatrix}}$$

« automate » de calcul
Element « processeur »

$$\boxed{a_j = \sum_{k=1}^{N} w_{kj} x_k}$$

- Analogue à la somme pondérée des filtres linéaires



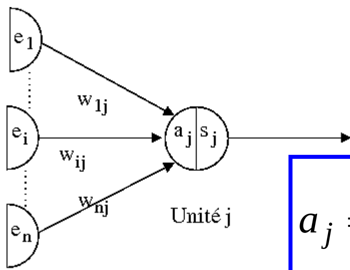


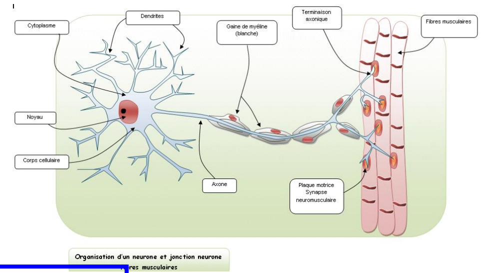
- Autre notation :

$$\boxed{\begin{aligned} p_j &= w \otimes x \\ &= \langle W | X \rangle \end{aligned}}$$

Neurone formel

McCulloch et Pitts (1943)





$$\boxed{a_j = \sum_{k=1}^{N} w_{kj} x_k}$$

Neurone formel j
Entrées e_i $i \in [1,n]$
Activité a_i
Sortie s_i

$$a_j = \sum_{i=1}^{N} w_{ij} e_i$$

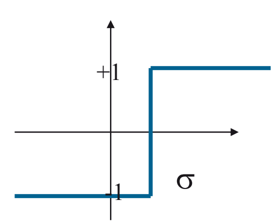
$$s_j = f\left(a_j\right)$$

Poids synaptique w_{ij}
Fonction d'activation *f*

Charge : + ou -

Neurone formel
La fonction d'activation

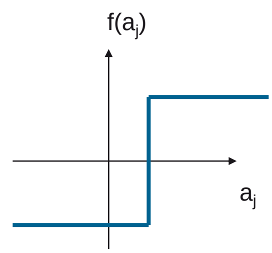
- **Fonction d'activation :**
  - Charges + ou – dans les neurones biologiques
- **Sortie binaire**
  - 0 ou 1
  - -1 ou 1 (cf charge +e ou –e)
  - Non dérivable



Interprétation du Perceptron

- (x_k) un vecteur d'entrée ($k \in [1,N]$)
- (s_j) sorties « binaires » (+1 ou -1), $j \in [1,R]$
- (d_j) sorties désirées(+1 ou -1), $j \in [1,R]$
- Poids $(w_{kj})=W_j \quad k\in [1,N]$
- Fonction seuil f

$$\boxed{\begin{aligned} a_j &= \sum_{k=1}^{N} w_{kj} x_k \\ s_j &= f\left(a_j\right) \end{aligned}}$$



Interprétation du perceptron

- **Ajustement des poids des neurones :**
  - Seulement si l'échantillon est mal classé

$$\boxed{W(t+1)=W(t)+\eta X \qquad \eta=\pm 1}$$

  - Pas de prise en compte de l'erreur quadratique

Mécanisme d'apprentissage : La règle du Perceptron (Rosenblatt, 1957)

- Classification en 2 classes
- Vecteur d'entrée : X
- Ensemble des mal classés : Y(W)
- Fonction de coût J(W)
- Calcul de J(W) sur les vecteurs X mal classés :

$$J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k}\right)^2 = \sum_{k=1}^{R} \left(\left(W^t X_k\right)^2 + d_{X_k}^2 - 2 d_{X_k} W^t X_k\right)$$

$$J(W) = \sum_{X \in Y(W)} -c(X)\, W^t X$$

c(X) = 1 si X∈C+
c(X) = -1 si X∈C-

La règle du Perceptron Rosenblatt, 1957

$$J(W) = \sum_{X \in Y(W)} -c(X)\, W^t X$$

c(X) = 1 si X∈C+
c(X) = -1 si X∈C-

- Modifier W dans le sens opposé à son gradient

$$\nabla J(W) = \sum_{X \in Y(W)} -c(X)\, X$$

c(X) = 1 si X∈C+
c(X) = -1 si X∈C-

Widrow-Hoff

- **Variante de l'algorithme du Perceptron**
- **Coût aux moindres carrés : on prend tous les échantillons**

$$J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k}\right)^2$$

- **Fonction quadratique : on sait dériver et exprimer le gradient**

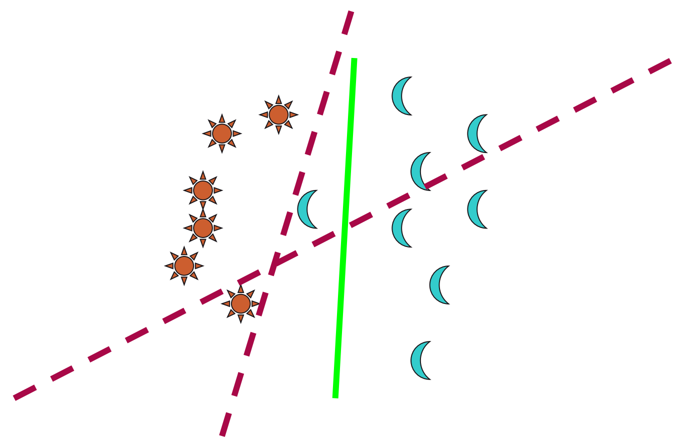
Widrow-Hoff

$$J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k}\right)^2$$

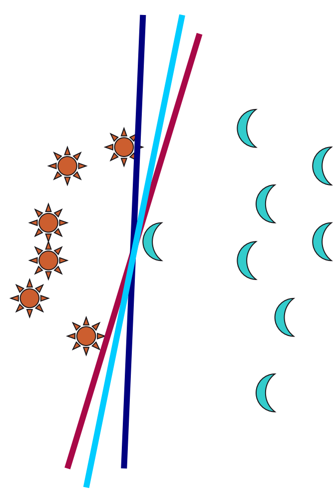
- **Variante de l'algorithme du Perceptron :**
  - Initialiser les poids
  - Modifier itérativement les poids si la <u>sortie</u> n'est pas égale à la sortie désirée

$$W(t+1) = W(t) + \eta\left(d_X - W^t(t)X\right)X$$

Exemple linéairement séparable : critère d'homogénéïté des classes

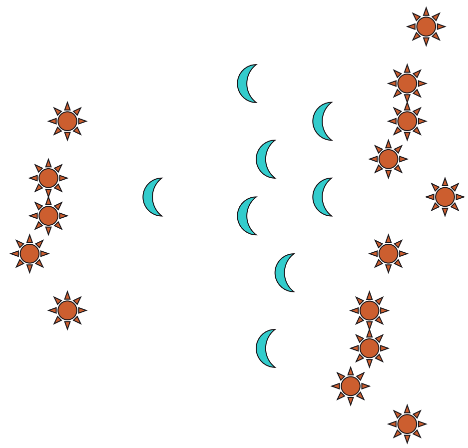


Exemple linéairement séparable MAIS est ce bien le « bon » hyperplan ??



- Notion de marge
- Principe des SVM

MAIS….
Exemple non linéairement séparable



Une question :
inverser le système ???

$$J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k} \right)^2$$

$$\frac{\partial J(W)}{\partial w_{ij}} = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k} \right) w_{ij} X_{k,j}$$

- Annuler la dérivée pour tous les w_{ij}
- Si vecteur d'état de dimension N, alors NxN coefficients à rechercher
- Généralement mal conditionné
- Résultats parfois discutables

Une vision entropique

- R échantillons dans la base d'apprentissage
- R « quantum » d'informations
- Dimension du vecteur d'état : N
- NxN coefficients pour la matrice W
- NxN « quantum » d'informations pour W

Bilan en 1965

- **Bonne classification pour des données séparables linéairement :**
  - Mécanisme d'apprentissage empirique mais efficace
  - L'apprentissage s'arrête dès que l'on a trouvé le séparateur linéaire
  - On ne sait rien si le problème n'est pas linéairement séparable !!
- **Variantes « signal » : ADALINE (Widrow)**
- **Echec dans le cas général (non séparable linéairement)**
- **La page se tourne vers l'Intelligence Artificielle**
- **Mais….**

Bilan en 1965 : les pistes restantes

- **Modifier les entrées :**
  - Entrée quadratique : le réseau ALN
  - Le réseau HOPI
  - MADALINE
- **Vers la combinaison de perceptrons**

Exemple du disque

- **Séparer l'intérieur de l'extérieur d'un disque**
  - Non linéairement séparable
- **Changement de variable**

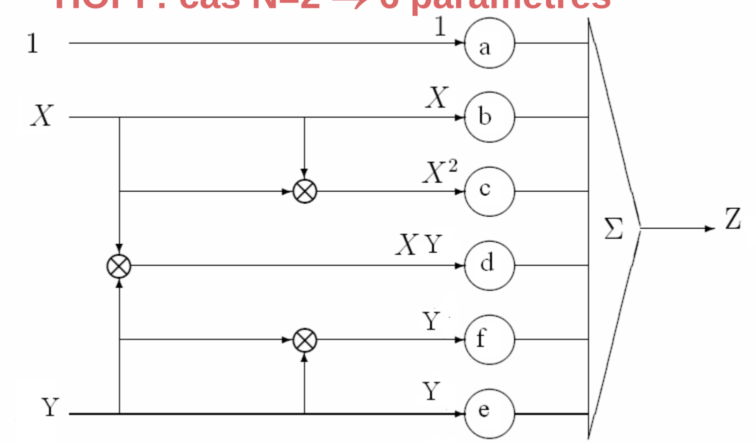
$$\boxed{\rho = \sqrt{x^2 + y^2}}$$

- **Séparatrice linéaire sur ρ**
- **$\rightarrow$On a changé l'espace d'états pour que la séparatrice soit linéaire**

Le réseau HOPI

- **High Order Polynomial Input**
- **Proposer en entrée :**
  - La variable
  - Son carré
  - Des termes croisés

HOPI : cas N=2 → 6 paramètres



$$\boxed{Z = a + bX + cX^2 + dXY + eY + fY^2}$$

- **Critère d'erreur quadratique sur la base d'apprentissage→dérivable !!**

Calcul du HOPI

$$Z_k = a + bX_k + cX_k^2 + dX_kY_k + eY_k + fY_k^2$$

- Pour chaque entrée Z_k on connaît la valeur cible $\underline{Z}_k$
- Pour la base de R données, on a l'erreur quadratique :

$$\boxed{\begin{aligned} E &= \sum_R \left(Z_k - \underline{Z}_k\right)^2 \\ &= \sum_R \left(a + bX_k + cX_k^2 + eY_k + fY_k^2 + dX_kY_k - \underline{Z}_k\right)^2 \end{aligned}}$$

Critère erreur quadratique

- J erreur quadratique
- Rappel : W opérateur linéaire

$$\boxed{J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k}\right)^2}$$

- Ici l'opérateur fait intervenir les entrées à la puissance 1 et 2 ainsi qu'un terme croisé : non linéaire

On minimise l'erreur pour chaque paramètre

$$\boxed{E = \sum_R \left(a + bX_k + cX_k^2 + eY_k + fY_k^2 + dX_kY_k - \underline{Z}_k\right)^2}$$

- Par exemple, pour le paramètre b :

$$\frac{\partial E}{\partial b} = \frac{\partial \sum_R \left(a + bX_k + cX_k^2 + eY_k + fY_k^2 + dX_kY_k - \underline{Z}_k\right)^2}{\partial b}$$

$$= \sum_R \left(a + bX_k + cX_k^2 + eY_k + fY_k^2 + dX_kY_k - \underline{Z}_k\right)X_k$$

- Pour atteindre le minimum, la dérivée doit être nulle, ce qui donne pour la variable b :

$$\frac{\partial E}{\partial b} = \sum_R \left(a + bX_k + cX_k^2 + eY_k + fY_k^2 + dX_kY_k - \underline{Z}_k\right)X_k = 0$$

$$\sum_R \left(a + bX_k + cX_k^2 + eY_k + fY_k^2 + dX_kY_k\right)X_k = \sum_R \underline{Z}_k X_k$$

$$a\sum_R X_k + b\sum_R X_k^2 + c\sum_R X_k^3 + e\sum_R Y_k + f\sum_R Y_k^2 + d\sum_R X_k^2 Y_k = \sum_R \underline{Z}_k X_k$$

HOPI : les solutions

- On effectue cette annulation de la dérivée pour les 6 variables : a, b, c, d, e et f
- On a alors un système linéaire de 6 équations à 6 inconnues
- La solution existe en général

$$\tilde{a} \quad \tilde{b} \quad \tilde{c} \quad \tilde{d} \quad \tilde{e} \quad \tilde{f}$$

- Généralisation : pour tout (X,Y)

$$\boxed{Z = \tilde{a} + \tilde{b}X + \tilde{c}X^2 + \tilde{d}XY + \tilde{e}Y + \tilde{f}Y^2}$$

Interprétation entropique

- 6 coefficients
- R données dans la base d'apprentissage
- 6 << R : cela doit « bien » généraliser

1980 : Le Perceptron.. Le retour…

Réseaux Connexionnistes

Réseaux Neuro-mimétiques

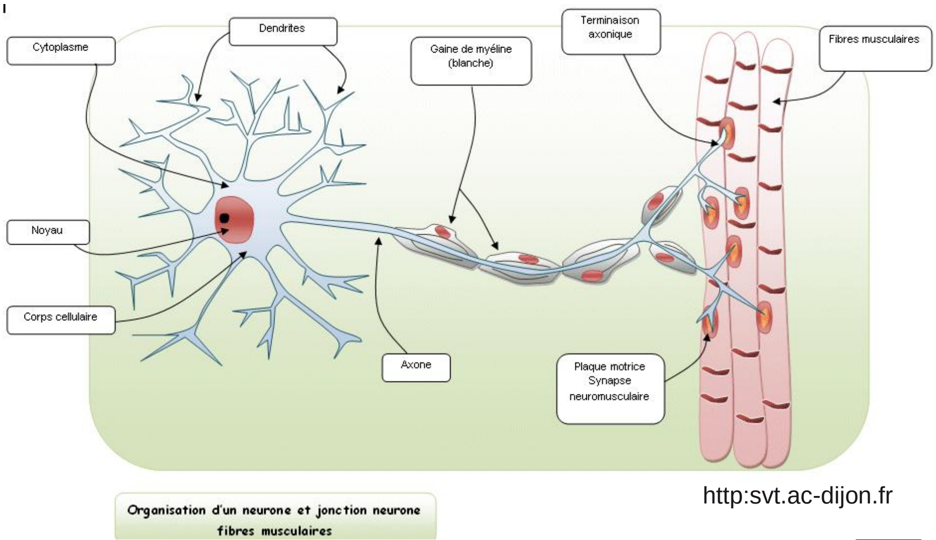
Réseaux Neuronaux

Connectionist Networks

Le fil de l'histoire

1943 :	le neurone formel	McCulloch et Pitts
1960 :	la règle delta	Widrow et Hoff
1962 :	le Perceptron	Rosenblatt
1969 :	les limites du Perceptron	Minsky et Papert
1980 :	le neo-connexionnisme	
1986 :	l'apprentissage	Le Cun et al.

Un neurone

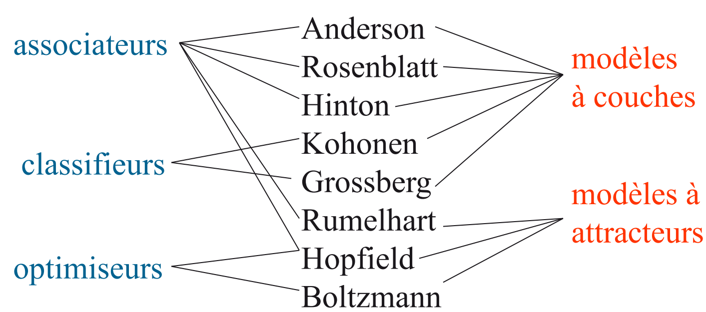


http:svt.ac-dijon.fr

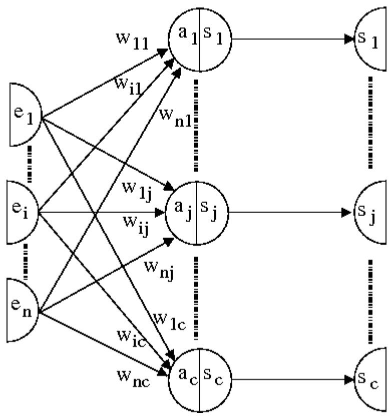
Les neurones en réseau….

Nouvelle architecture :
Le perceptron multicouche

Le réseau des réseaux



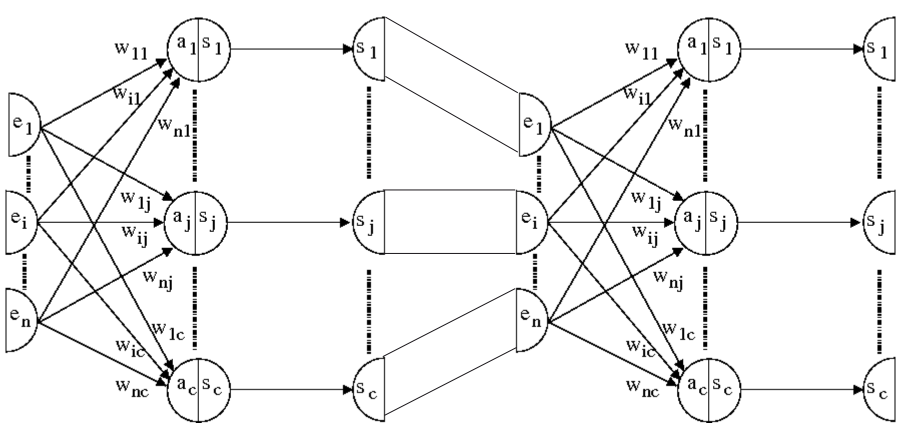
Neurone formel



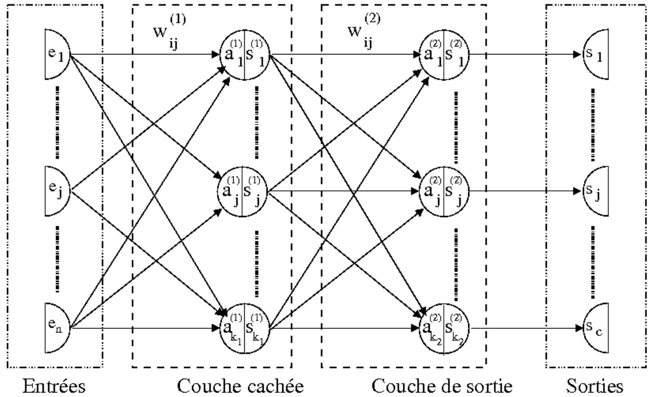
$$a_j = \sum_{i=1}^{n} w_{ij} e_i$$

$$s_j = f\left(a_j\right)$$

Utiliser des « couches » de neurones



Exemple à deux couches



- **Notion de « couche cachée »**

Classification « non linéaire » possible

- **Le cas du « et »:**
  - Linéairement séparable

0	1
0	0

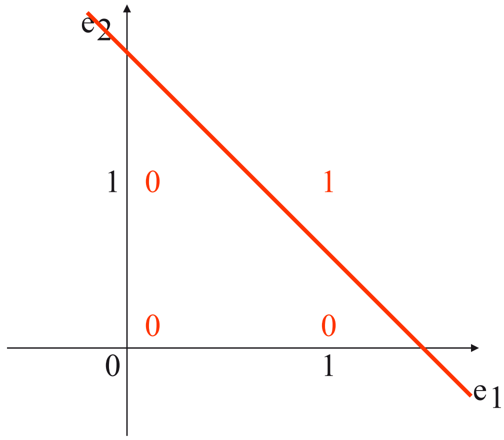
- **Le cas du « ou exclusif »**
  - Pas de solution par le perceptron
  - Rôle du « multicouche »

1	0
0	1

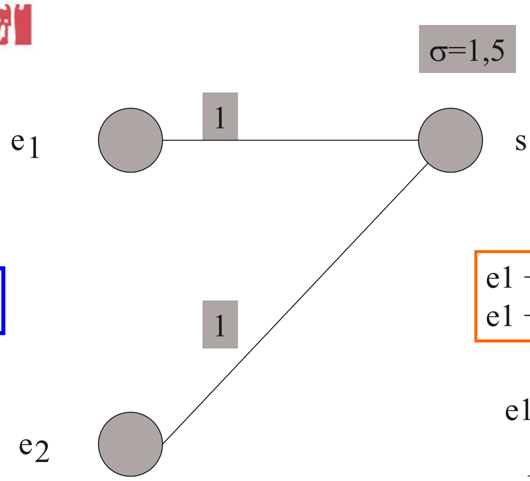
Cas d'école : le « et »

0	1
0	0

et



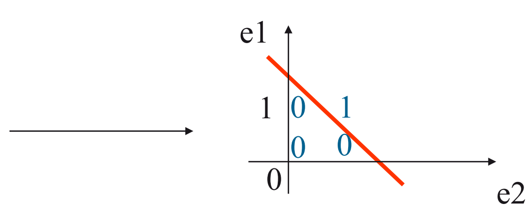
Séparatrice : pente=-1



et

$$e1 + e2 - 1{,}5 > 0 \rightarrow 1$$
$$e1 + e2 - 1{,}5 < 0 \rightarrow 0$$

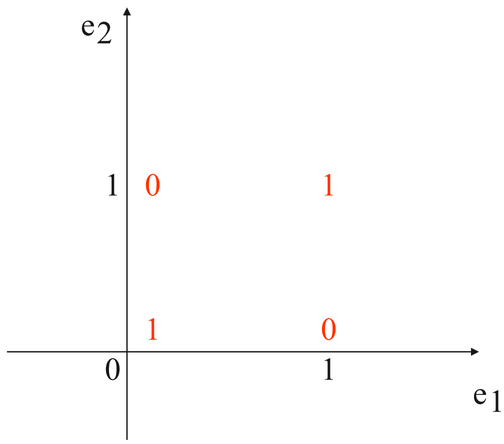
f : fonction à seuil
sur [0 … 1]



Cas d'école : le « ou exclusif »

1	**0**
0	**1**

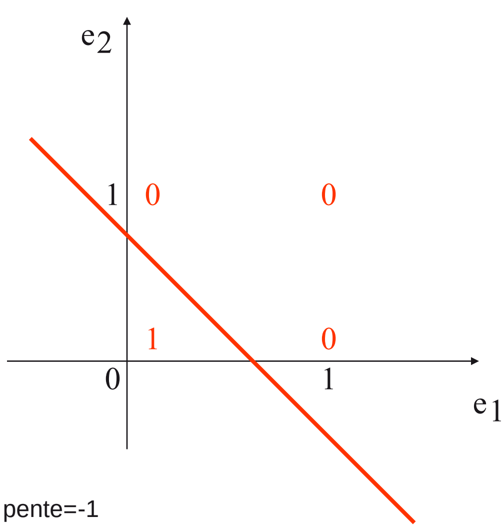
ou



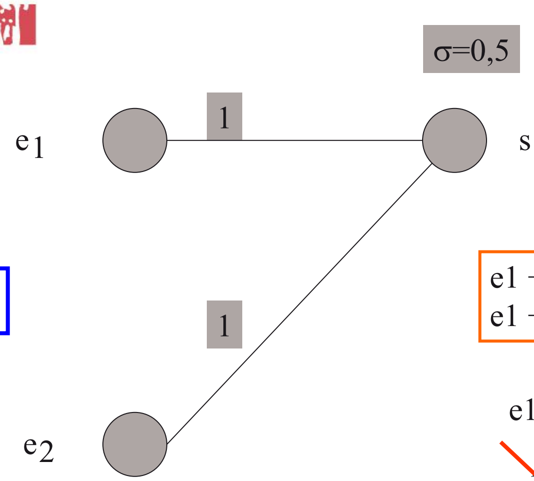
Cas d'école : un autre « et »

0	**0**
1	**0**

et

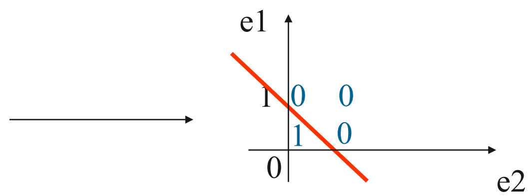


Séparatrice : pente=-1

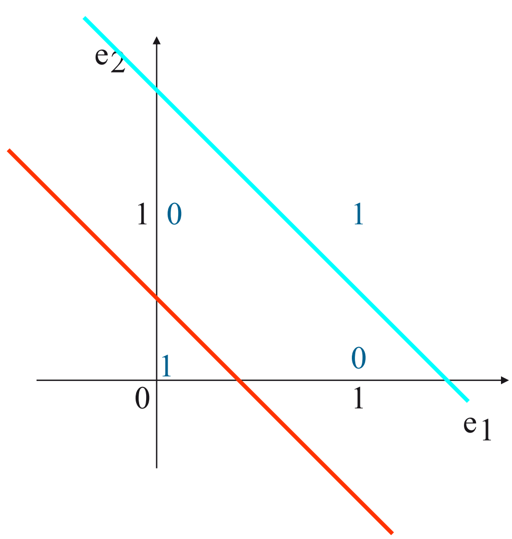


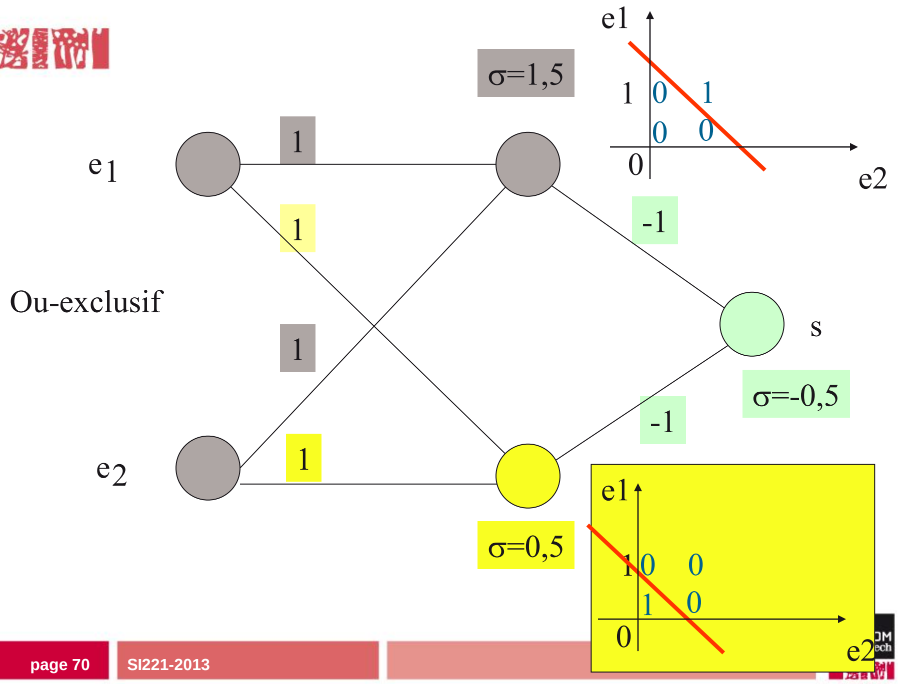
et

$e1 + e2 - 0{,}5 > 0 \rightarrow 0$
$e1 + e2 - 0{,}5 < 0 \rightarrow 1$

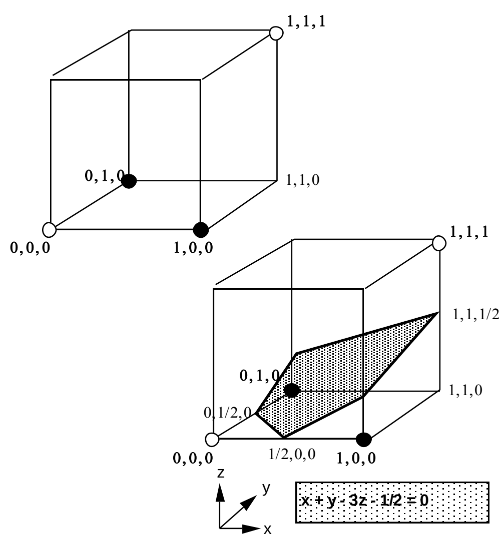


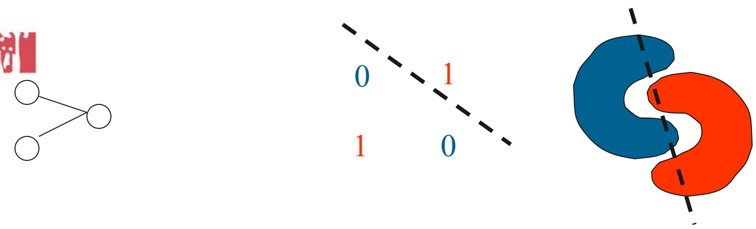
Ou-exclusif



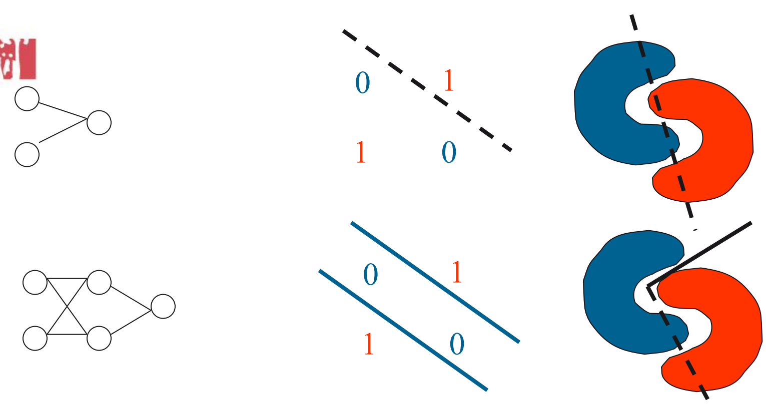


Changement de dimension de l'espace !!

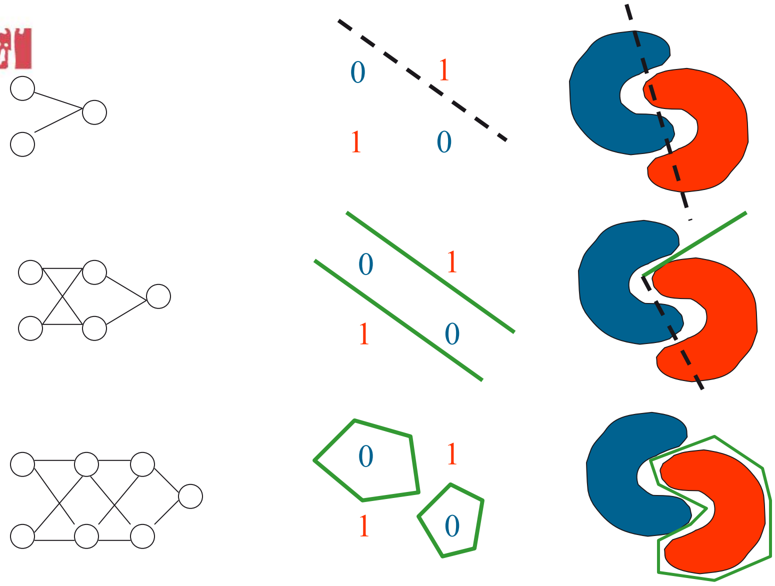




(D'après R.P. Lippmann 87)

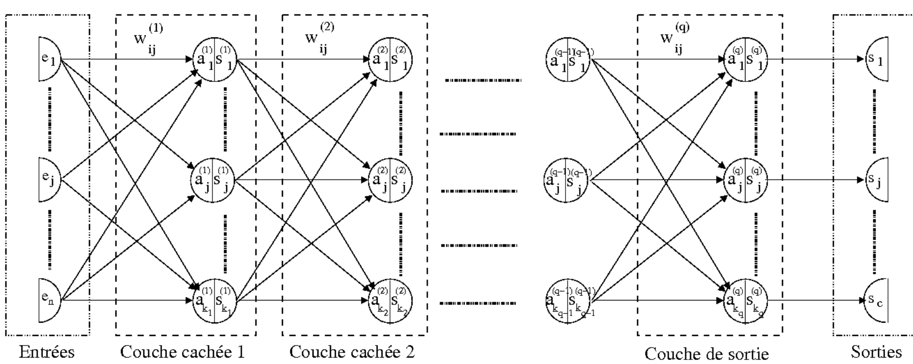


(D'après R.P. Lippmann 87)



(D'après R.P. Lippmann 87)

Réseaux à q couches



Théorème d'existence

- Avec deux couches cachées, on peut résoudre tout problème de classification non linéaire
- MAIS : on ne sait pas comment !!!

La sortie

- **En général**
  - Classification en c classes
  - Vecteur s_i de dimension c :

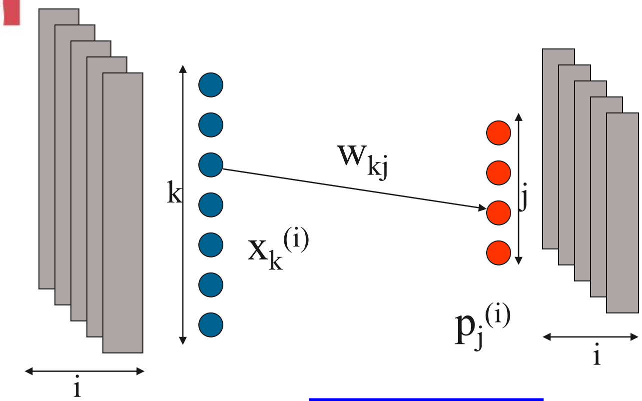
$$\begin{pmatrix}1\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\1\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\1\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Nouveaux principes d'apprentissage

Apprentissage

- Principe de Hebb
- Perceptrons (simples) : Règle delta
- Perceptrons multi-couches :
  Rétropropagation de gradient

Règle Delta



Propagation

$$p_j = \sum_{k=1}^{N} w_{kj} x_k$$

$$s_j = f\left(p_j\right)$$

Erreur quadratique et descente de gradient

$$E_i = \sum_{j=1}^{c} \left(f\left(\sum w_{kj} x_k\right) - d_j\right)^2$$

$$E = \sum_i E_i$$

Descente de gradient

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} = -\eta \sum_i \frac{\partial E_i}{\partial w_{kj}}$$

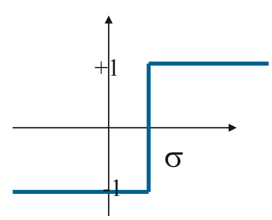
Fonction d'activation

$$E_i = \sum_{j=1}^{c} \left(f\left(\sum w_{kj} x_k\right) - d_j \right)^2$$

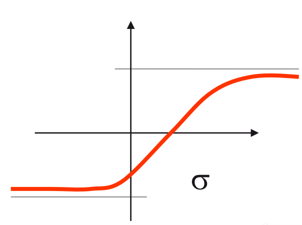
- **La fonction d'activation f doit être dérivable**
- **Introduction de la fonction sigmoïde**

Neurone formel
La fonction d'activation

- **Sortie binaire**
  - 0 ou 1
  - -1 ou 1 (cf charge +e ou –e)
  - Non dérivable



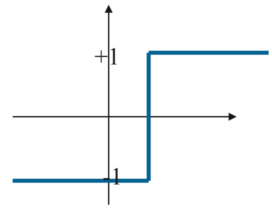
- **Fonction « sigmoïde »**
  - f(x) $\in$]-1;1[
  - Réglage : pente en y=0
  - Cas limite : fonction seuil
  - dérivable

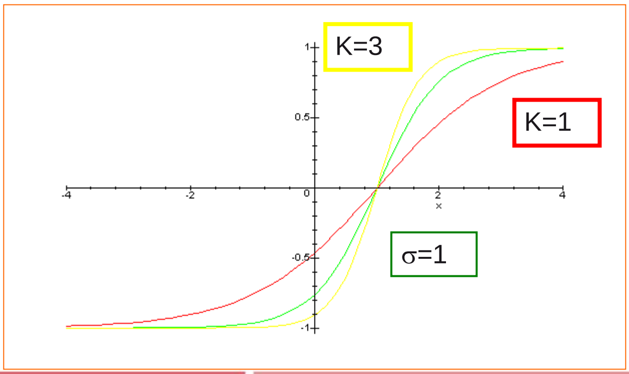


Fonction sigmoïde

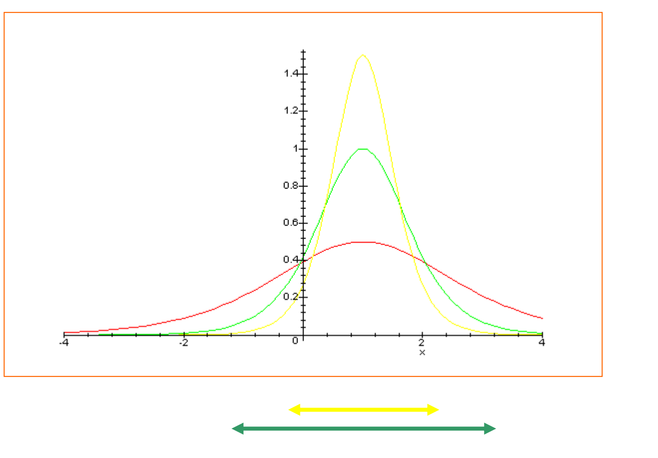
$$f[K,\sigma](x) = \frac{e^{K(x-\sigma)} - 1}{e^{K(x-\sigma)} + 1}$$

Limite :





Fonction sigmoïde : dérivable



Widrow-Hoff : nouvelles perspectives

- **Coût aux moindres carrés : on prend tous les échantillons**

$$J(W) = \sum_{k=1}^{R} \left(W^t X_k - d_{X_k}\right)^2$$

- **Fonction quadratique dérivable : on sait dériver et exprimer le gradient**
- **Correction des poids w_k en fonction du gradient**

$$E_i = \sum_{j=1}^{c} \left(f\left(\sum w_{kj} x_k\right) - d_j\right)^2 = \sum_{j=1}^{c} \left(p_{X_k} - d_j\right)^2$$

$$E = \sum_i E_i$$

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}}$$

$$a_j = \sum_{k=1}^{N} w_{kj} x_k$$

$$s_j = f\left(a_j\right)$$

$$\frac{\partial E_i}{\partial w_{kj}} = \sum_{j'=1}^{c} \frac{\partial}{\partial w_{kj}} \left(s_{j'} - d_{j'}\right)^2$$

$$= \sum_{j'=1}^{c} \frac{\partial}{\partial s_j} \left(s_{j'} - d_{j'}\right)^2 \frac{\partial s_{j'}}{\partial w_{kj}} = \frac{\partial}{\partial s_j} \left(s_j - d_j\right)^2 \frac{\partial s_j}{\partial w_{kj}}$$

$$= 2\left(s_j - d_j\right) \frac{\partial f\left(a_j\right)}{\partial w_{kj}} = 2\left(s_j - d_j\right) f'\left(a_j\right) \frac{\partial a_j}{\partial w_{kj}}$$

$$a_j = \sum_{k=1}^{N} w_{kj} x_k$$
$$s_j = f(a_j)$$
$$\frac{\partial a_j}{\partial w_{kj}} = x_k$$

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}}$$

$$\frac{\partial E_i}{\partial w_{kj}} = 2(s_j - d_j) f'(a_j) \frac{\partial a_j}{\partial w_{kj}} = 2(s_j - d_j) f'(a_j) x_k$$

- **Cas où f est l'identité**

$$\frac{\partial E_i}{\partial w_{kj}} = (s_j - d_j) x_k$$

Règle Delta : à appliquer pour chaque exemple

$$\Delta w_{kj} = -\eta (s_j - d_j) x_k$$

Règle delta

- **Elaboration de la base d'exemple**
- **Définition de la structure**
- **Initialisation des poids (aléatoire)**

1. **Prendre la base d'exemple (« époque » n=0)**
   - Sélection d'un exemple et calcul de la sortie
   - Modification des poids $\Delta w_{kj} = -\eta (s_j - d_j) x_k$
   - Boucler sur les exemples
2. **Calcul de l'erreur E sur la base d'exemple**
3. **E > seuil :**
   - Incrémenter l'époque : n = n+1
   - Si n<Nmax : revenir en 1

Réseau multicouche : rétropropagation du gradient

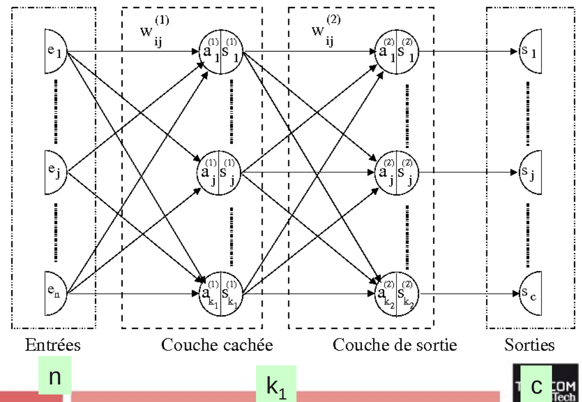
$$a_l^{(2)} = \sum_{j=1}^{k_1} w_{jl}^{(2)} s_j^{(1)}$$

$$s_l = f\left(a_l^{(2)}\right)$$

$$a_j^{(1)} = \sum_{k=1}^{n} w_{kj}^{(1)} e_k$$

$$s_j^{(1)} = f\left(a_j^{(1)}\right)$$

$$\frac{\partial E_i}{\partial w_{kj}} = 2\sum_{l=1}^{c}\left(s_l - d_l\right)\frac{\partial f\left(a_l\right)}{\partial w_{kj}}$$



Pour la dernière couche

- Refaire le calcul précédent

$$\frac{\partial E_i}{\partial w_{jl}^{(2)}} = 2\left(s_l - d_l\right) f'\left(a_l^{(2)}\right) s_j^{(1)}$$

Pour la couche cachée :

$$a_l^{(2)} = \sum_{j=1}^{k_1} w_{jl}^{(2)} s_j^{(1)}$$

$$s_l = f\left(a_l^{(2)}\right)$$

$$a_j^{(1)} = \sum_{k=1}^{n} w_{kj}^{(1)} e_k$$

$$s_j^{(1)} = f\left(a_j^{(1)}\right)$$

$$s_l = f\left(\sum_{j=1}^{k_1} w_{jl}^{(2)} f\left(a_j^{(1)}\right)\right) = f\left(\sum_{j=1}^{k_1} w_{jl}^{(2)} f\left(\sum_{k=1}^{n} w_{kj}^{(1)} e_k\right)\right)$$

Pour la couche cachée

$$s_l = f\left(\sum_{j=1}^{k_1} w_{jl}^{(2)} f\left(a_j^{(1)}\right)\right) = f\left(\sum_{j=1}^{k_1} w_{jl}^{(2)} f\left(\sum_{k=1}^{n} w_{kj}^{(1)} e_k\right)\right)$$

- **Dérivation des fonctions composées**
- **Possible puisque la sigmoïde est dérivable !!**

$$\frac{\partial E_i}{\partial w_{kj}^{(1)}} = 2e_k f'\left(a_j^{(1)}\right) \sum_{l=1}^{c} \left(w_{jl}^{(2)} \left(s_l - d_l\right) f'\left(a_l^{(2)}\right)\right)$$

- **On rétropropage le gradient !!!**

Rétropropagation du gradient

- Elaboration de la base d'exemple
- Définition de la structure
- Initialisation des poids (aléatoire)

1. **Prendre la base d'exemple (« époque » n=0)**
   - Sélection d'un exemple et calcul de la sortie
   - Modification des poids
   - Boucler sur les exemples

$$\boxed{\Delta w_{kj} = -\eta Z_{kj}}$$

2. **Calcul de l'erreur E sur la base d'exemple**
3. **E > seuil :**
   - Incrémenter l'époque : n = n+1
   - Si n<Nmax : revenir en 1

Base d'apprentissage

- **Sert à modifier les poids du réseau**
- **On présente les échantillons dans un ordre aléatoire**
  - Calcul du gradient de la dernière couche
  - Rétropropagation pour les couches cachées
  - Modification des poids
- **Critère d'arrêt**
  - Nombre d'époque

Caractérisation de l'architecture

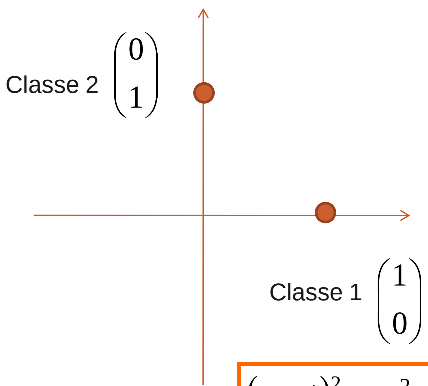
- **Utilisation de la base de généralisation**
- **Deux sortes d'erreur**
  - Erreur de classification
  - Erreur quadratique

Erreur de classification :
Rappel : matrice de confusion

		Classification		
		Classe 1	**Classe j**	**Classe c**
Référence	**Classe 1**	X_{11}	X_{1j}	X_{1c}
	Classe i	X_{i1}	X_{ij}	X_{ic}
	Classe c	X_{c1}	X_{nj}	X_{cc}

Le taux de bonne classification revient à ne s'intéresser qu'à la diagonale

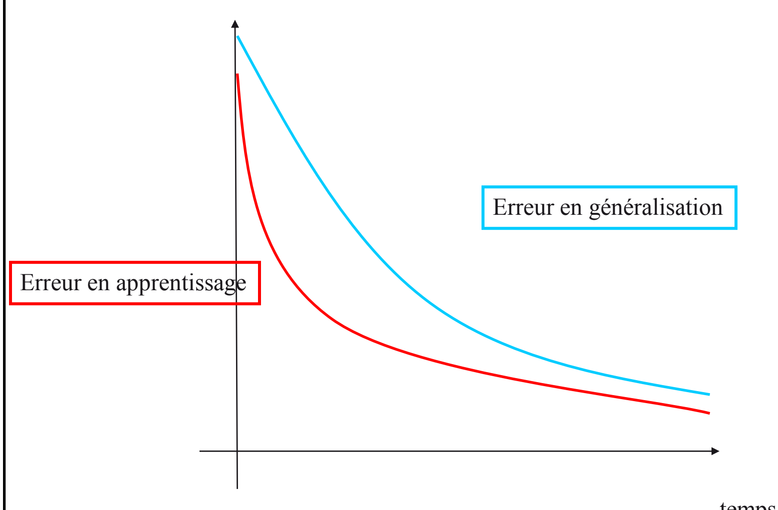
Exemple : deux classes espace de sortie en dimension 2



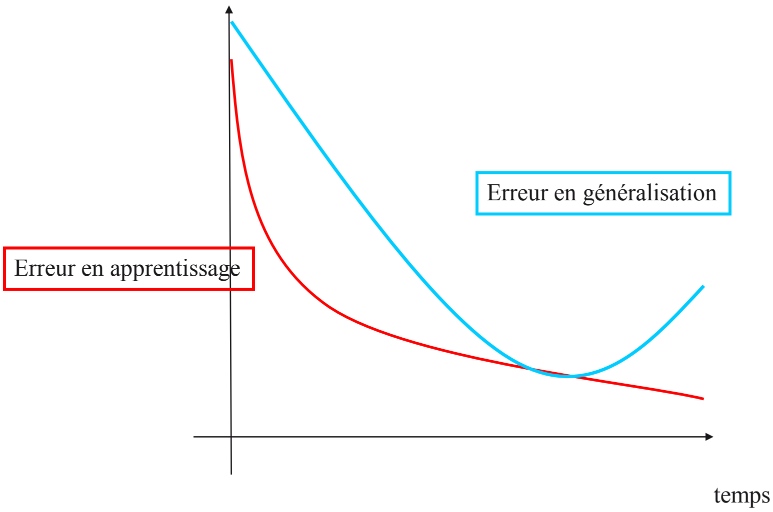
- Deux classes
- Deux étiquettes « objectif »
- A un point donné on cherche la classe la plus proche
- Si le point est mal classé, on peut définir une erreur

$$(x-1)^2 + y^2 = x^2 + y^2 + 1 - 2x$$
$$x^2 + (y-1)^2 = x^2 + y^2 + 1 - 2y$$

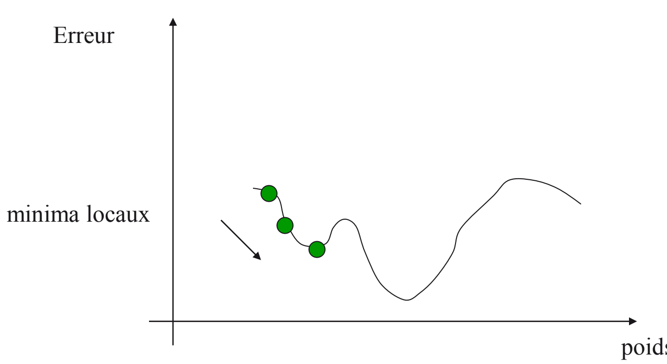
Erreur quadratique en fonction du temps (époque)

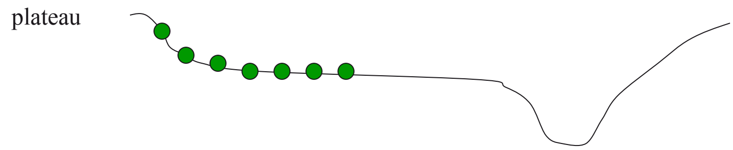


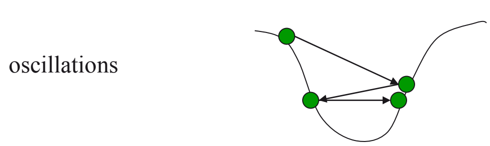
Erreur quadratique : surapprentissage

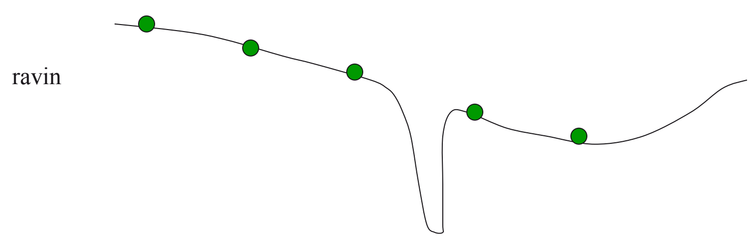


Problèmes




plateau


oscillations

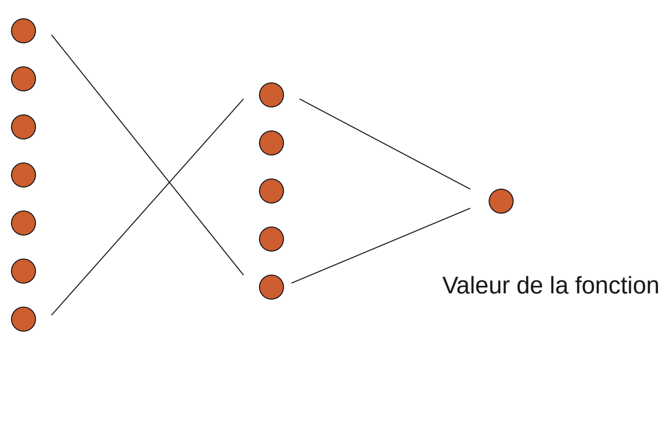


Critères d'apprentissage

- **Erreur quadratique**
  - Utilisée dans l'algorithme de rétropropagation
  - Utilisable pour la classe rejet
- **Erreur de classification**
  - En pourcentage de bonne classification
  - Résultat « opérationnel »

Autres réseaux

Approximation de fonctions F(x,y,…,z)



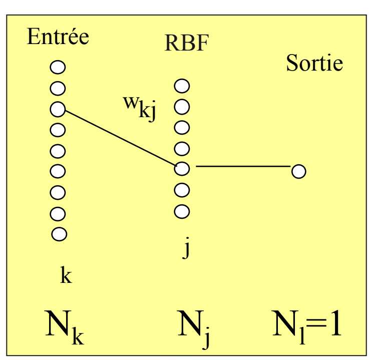
Autres réseaux : les RBF

- **Fonctions de Base Radiales :**
  - De R^n dans R
  - Symétrique autour d'un centre
  - Paramétrées par une « largeur »
- **Exemple : la gaussienne**

$$\boxed{\varphi_j\left[\mu_j,\sigma_j\right](x) = e^{\frac{-\left\|x-\mu_j\right\|^2}{2\sigma_j}}}$$

Réseaux de RBF

$$\boxed{y(x) = \sum_{j=1}^{N} w_j \varphi_j\left[\mu_j,\sigma_j\right](x)}$$



Modèle statistique de l'apprentissage

- Formalisme rigoureux
- Inégalité de Vapnik
- MSR : Minimisation Structurelle du Risque
- SVM : Machine à Vecteur de Support

Exemples

Réseaux neuromimétiques : Exemples d'application

- Analyse de caractères manuscrits
- Classification de bruits en acoustique sous marine
- Traitement d'images

Reconnaissance de caractères

Y. Le Cun et al.

AT & T Bell Labs

1990

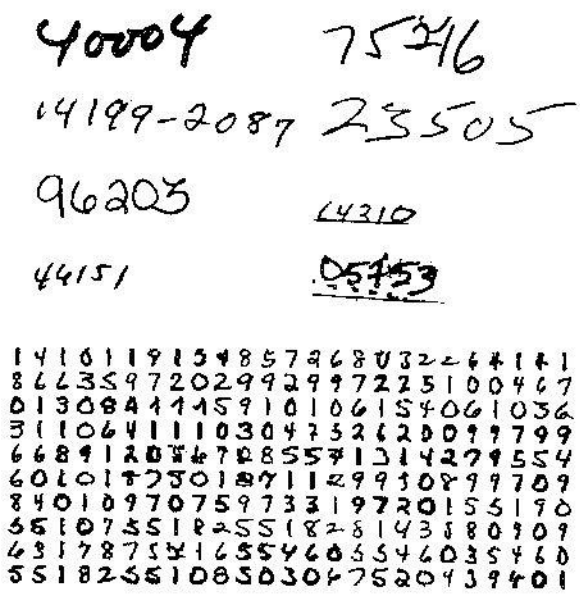
Le problème

Données : 9298 chiffres extraits de codes postaux manuscrits
3249 chiffres en caractères d'imprimerie (35 polices)

Base d'exemples : 7291 chiffres manuscrits
2549 chiffres en caractères d'imprimerie

Base de test : 2007 chiffres manuscrits
700 chiffres en caractères d'imprimerie

(les deux bases contiennent des exemples ambigus (naturels))



Architecture du réseau (1)

Entrées

```
nombre : 28 x 28 unités (dimensions de l'imagette)

codage :1 unité = 1 pixel
        niveaux de gris sur [-1,+1]
```

Sorties

```
nombre : 10 unités

codage : unité correspondant au caractère en entrée : +1
         autres unités : -1
```

Architecture du réseau (2)

Structure du réseau

```
4 couches cachées :4x24x24
                   4x12x12
                   12x8x8
                   12x4x4

connexions :spécifiques, non totales
            poids partagés
            98442 connexions
            2578 poids
```


Les performances

Apprentissage

9840 exemples

30 époques

3 jours sur Sparc 1

Performances

erreur 1,1 % sur base d'apprentissage

3,4 % sur base de test

toutes les erreurs : sur caractères manuscrits

taux de rejet pour atteindre 1% :

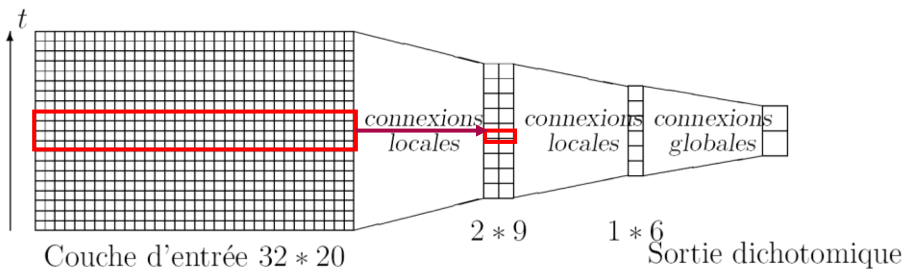
activité > seuil1
activité du second < seuil2
différence d'activités 1er-2ème> seuil3

moyenne : 5,7 %

manuscrits seulement : 9 %

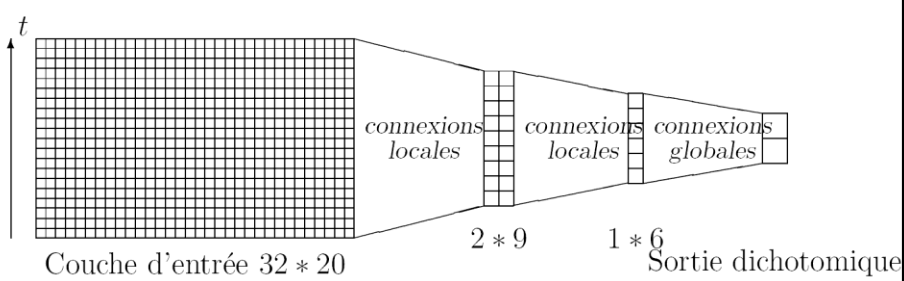
Architectures pour le traitement du signal Lemer et al., 1989

Traitement de bruits acoustiques sous marins



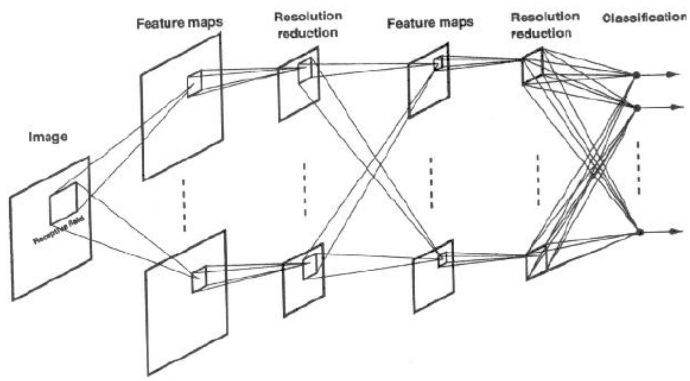
Architecture favorisant le déroulement du temps

Prétraitement pour la couche d'entrée



- **Transformée Temps Fréquence**
- **Transformée Temps Echelle (paquet d'ondelettes)**

Architecture pour le traitement des images



Conclusion,

- **Domaine « mature »**
- **Des plateformes accessibles :**
  - Matlab toolbox
  - SNNS
  - RNSat à TélécomParisTech