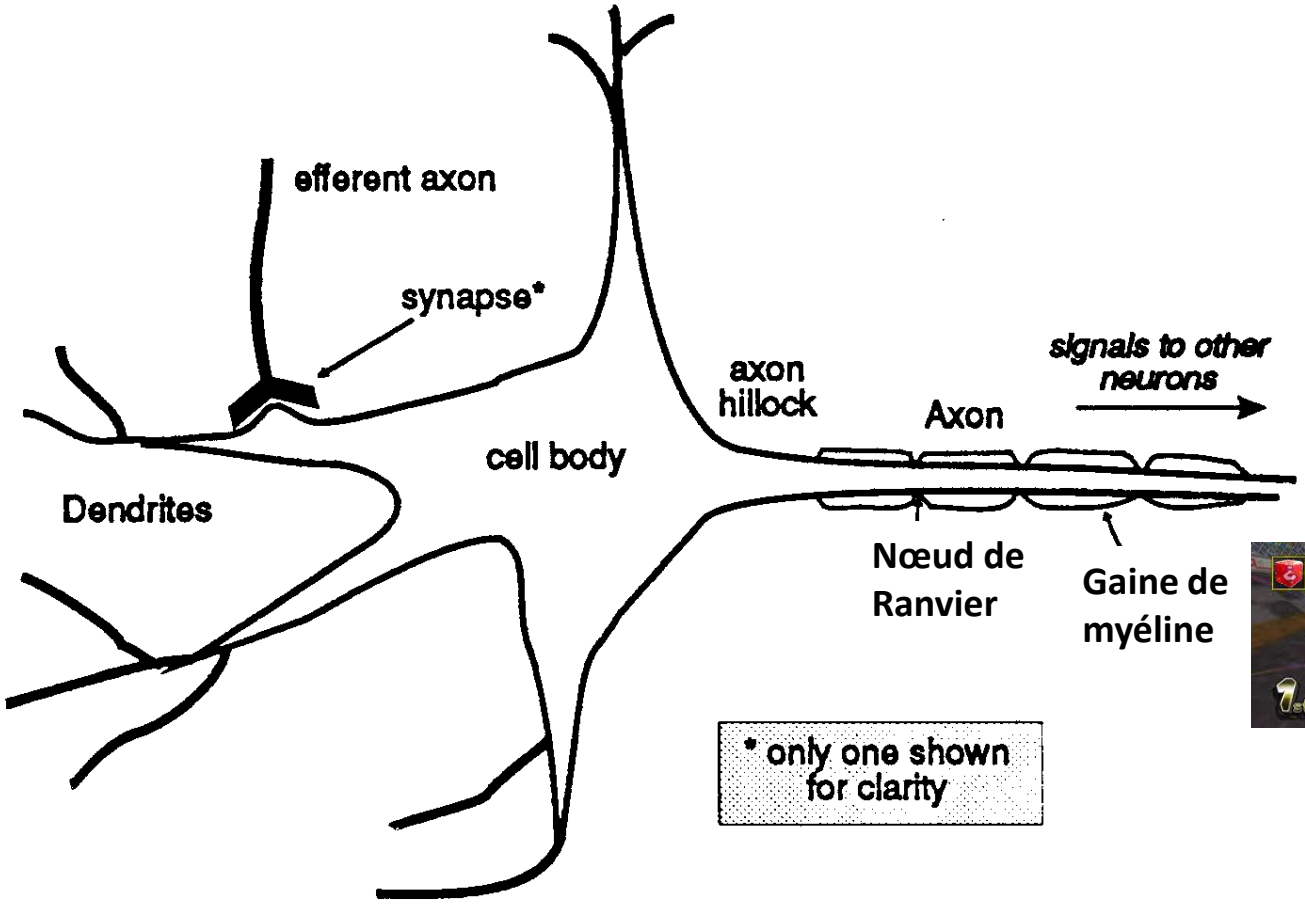
Réseaux de neurones

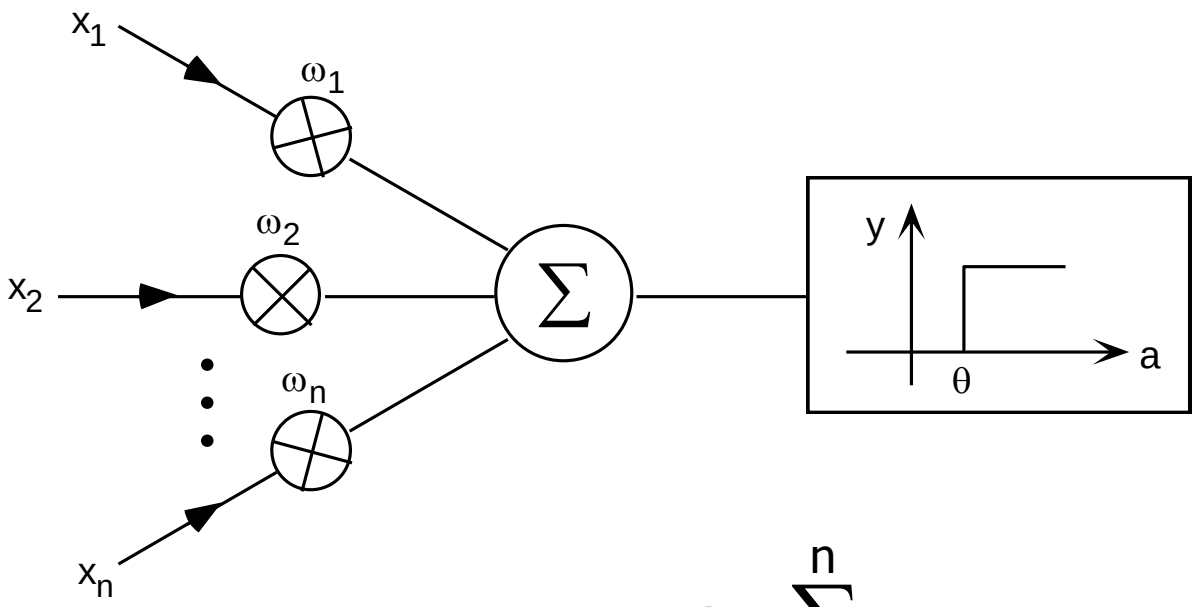
- Introduction aux réseaux de neurones
- Le perceptron
- La rétropropagation

- Travaux sur les RN remontent aux années 1940.
- Ces modèles primitifs (connexionnistes) furent appelés réseaux neuronaux car ils essayaient de modéliser fidèlement le réseau de cellules du cerveau (les neurones).
- Entre 1950-1970 il y a eu un épanouissement de ce domaine.
- Un déclin à la fin de cette période à cause de :
  - Limitations méthodologiques et technologiques.
  - Attaques excessives pessimistes sur l'utilité potentielle de ces modèles [Minsky, Pupert, 1969].
- Des chercheurs ont continué dans ce domaine comme T. Kohonen, S. Grossberg, J. Anderson et K. Fukushima.
- Réapparition après découverte de résultats théoriques importants dans les débuts 80, principalement la « error-back propagation ».


efferent axon
synapse*
axon hillock
signals to other neurons
Axon
cell body
Dendrites
Nœud de Ranvier
Gaine de myéline
* only one shown for clarity

Un réseau de neurones « formel » est un assemblage interconnecté d'éléments simples : neurones.

La capacité de traitement d'un réseau de neurones réside dans les connexions interunités ou encore poids. Les poids sont déterminés selon un processus d'adaptation/apprentissage.



$$a = \sum_{i=1}^{n} w_i x_i$$

La fonction de sortie du neurone est calculée en appliquant la fonction de sortie à la valeur d'activation.

Exemples

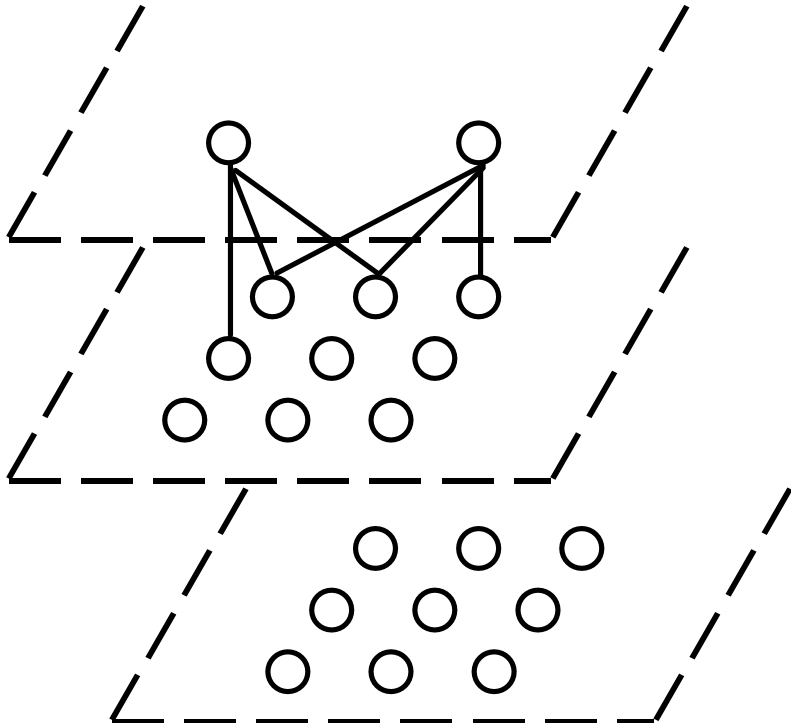
Linéaire
Bornée
Binaire
Binaire avec seuil
Sigmoïde

C'est une procédure locale qui décrit comment les poids des connexions varient en fonction du temps.

$$w_i(t+1) = w_i(t) + \eta r(w_i(t),\ x,\ d_i(t))x$$

Un neurone est connecté à la sortie de plusieurs autres neurones.

Les neurones auxquels il est connecté et la valeur des poids de connexions forment le champ récepteur du neurone.



Les RN se distinguent par les caractéristiques suivantes :

- La nature des connexions
- La prise en compte du temps (réseaux à délai)
- La présence de couches de neurones (neurones cachés)
- Le type du supervision (non supervisé, supervisé)
- L'algorithme d'apprentissage (hebbien, compétitif…)
- La complexité de la classification (linéaire, non linéaire)
- La nature des problèmes (association, contrôle, classification, optimisation…)

La règle de Hebb

$$\Delta w_{ij} = w_i(t+1) - w_i(t) = \eta\, a_i\, a_j$$

Règle de corrélation

$$\Delta w_{ij} = \eta\, d_i a_j$$

Règle du Winner-Takes-All (WTA)

$$\Delta w_k = \eta\,(x - w_k)$$

Minimisation de l'erreur: la règle delta

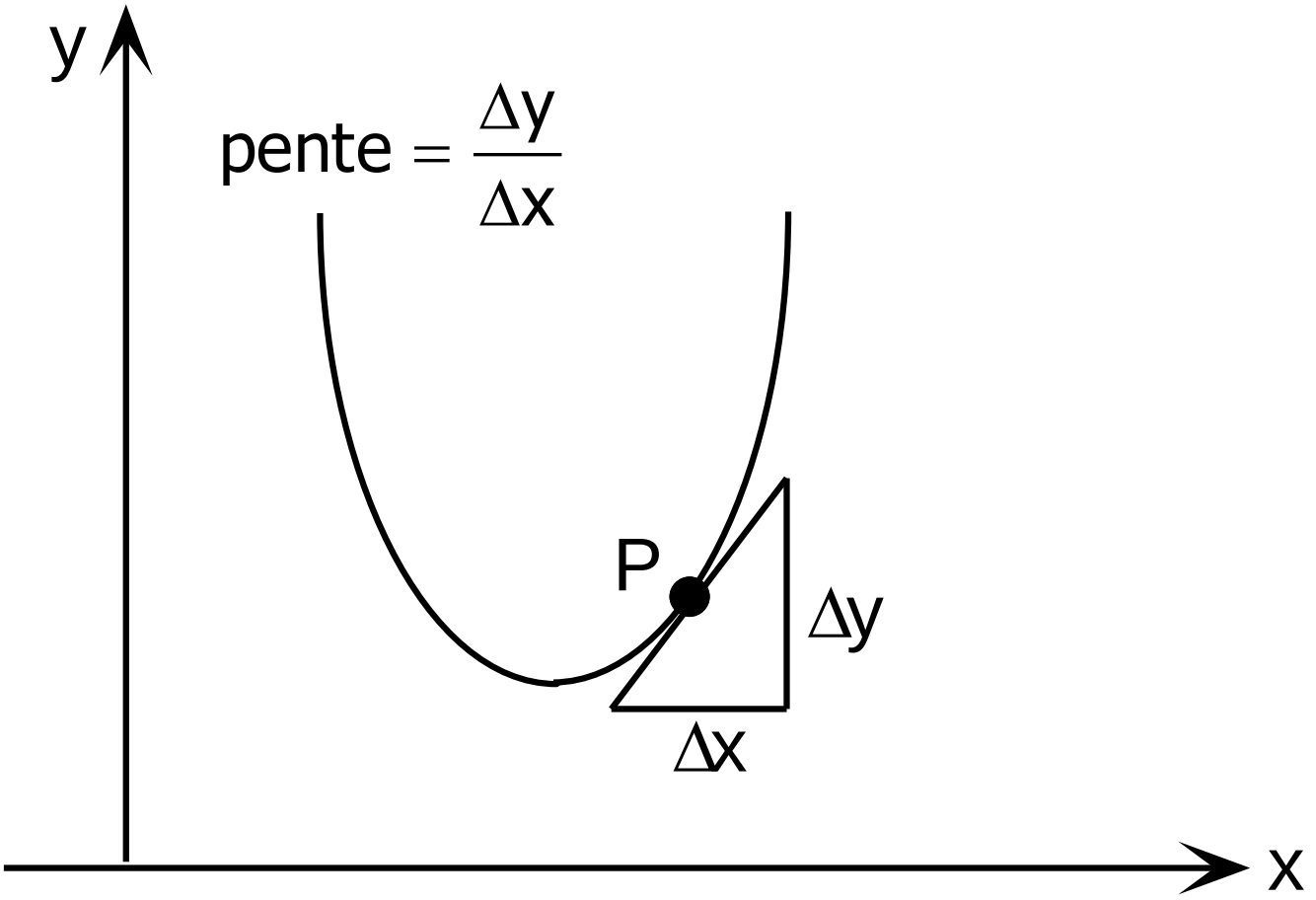
$$\Delta_p w_j = \eta\left(d^p - a^p\right)x_j$$

La règle de Delta se base sur le principe de la minimisation de l'erreur :

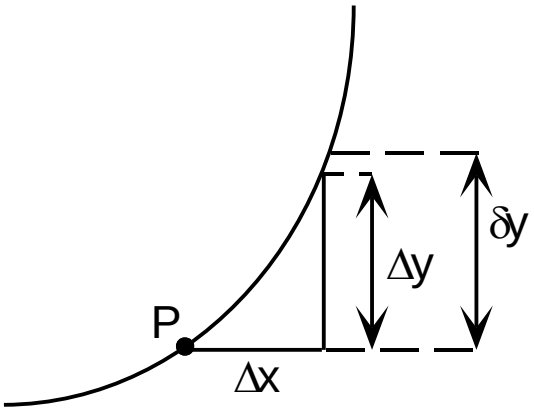
- Le minimum d'une fonction : descente du gradient
- Descente du gradient par l'erreur

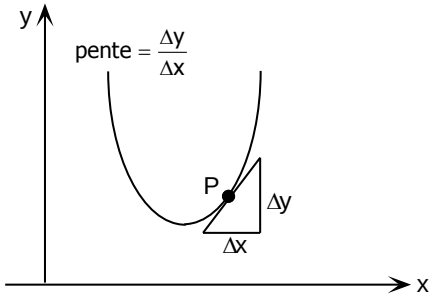
Le minimum d'une fonction : descente du gradient

Objectif : trouver le minimum d'une fonction



Le descente du gradient : procédure





Si Δx est très petit, alors Δy reste presque égal à δy

$$\delta y \cong \Delta y = \frac{\Delta y}{\Delta x}\Delta x$$

$$\delta y \cong \text{pente} \cdot \Delta x$$

Si on pose $\Delta x = -\eta \cdot \text{pente}$

$$\delta y = -\eta(\text{pente})^2$$

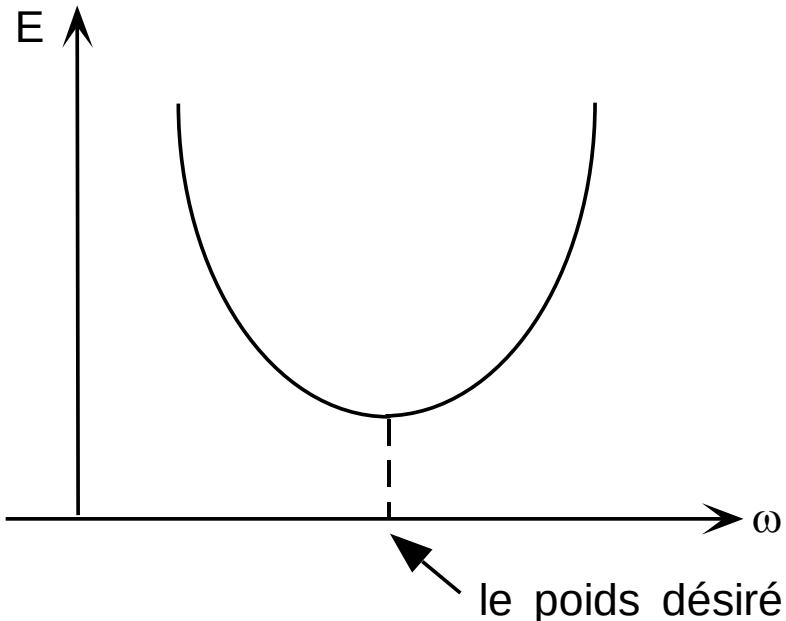
répéter cette étape pour approcher le minimum de la fonction

Comment utiliser cette technique pour entraîner un réseau?

Descente du gradient par l'erreur : principe

Principe :

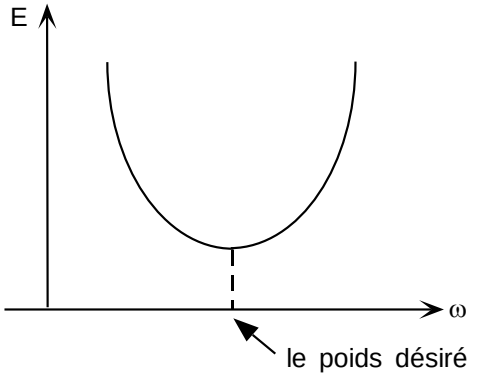
- calculer l'erreur à chaque nouvelle forme « p » présentée au réseau
- appliquer la descente du gradient pour calculer l'erreur en fonction du poids.



Descente du gradient par l'erreur : technique

Soit une activation de la forme :

$$a = \sum_j w_j x_j + \theta$$



L'erreur quadratique (LMS) E_p sur la forme p :

$$E_p = \frac{1}{2}\left(d^p - a^p\right)^2$$

P : indice de la forme d'entrée

d^p : sortie désirée pour l'entrée p

a^p : activation pour la forme p

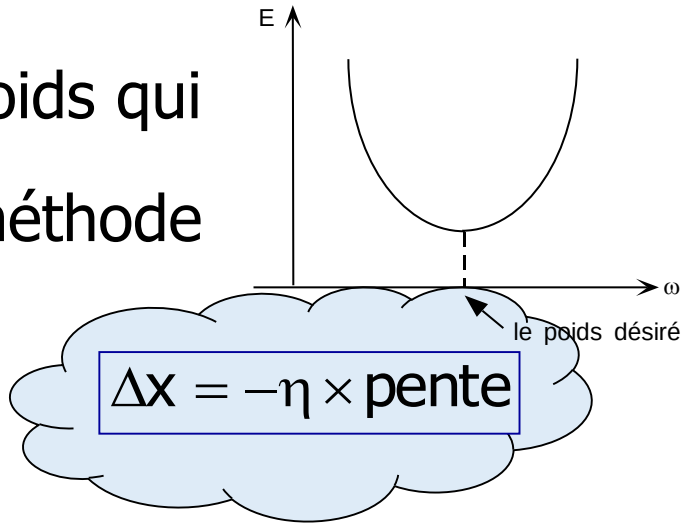
L'erreur totale E :

$$E = \sum_p E_p$$

$$E = \sum_p \frac{1}{2}\left(d^p - a^p\right)^2 = \frac{1}{2}\sum_p \left(d^p - a^p\right)^2$$

Descente du gradient par l'erreur : technique

La méthode LMS consiste à trouver les poids qui minimisent la fonction d'erreur par la méthode **descente du gradient**



La règle d'apprentissage (règle delta) est définie par

$$\Delta_p w_j = -\eta \ * \ \left(\text{pente de } E_p \text{ par rapport à } w_j\right)$$

Objectif immédiat

Calcul de la pente de E_p % w_j

Descente du gradient par l'erreur : technique

$$\Delta_p w_j = -\eta \; * \left(\text{pente de } E_p \text{ par rapport à } w_j\right)$$

$$\frac{\partial E^p}{\partial w_j} = \frac{\partial E^p}{\partial a^p} \cdot \frac{\partial a^p}{\partial w_j}$$

$$\frac{\partial a^p}{\partial w_j^p} = x_j$$

$$\frac{\partial E^p}{\partial a^p} = \frac{\partial}{\partial a^p}\left[\frac{1}{2}\left(d^p - a^p\right)^2\right]$$

$$= -\left(d^p - a^p\right) \quad \Rightarrow \quad \frac{\partial E^p}{\partial w_j} = -\left(d^p - a^p\right) \cdot x_j$$

Donc finalement

$$\frac{\partial E^p}{\partial w_j} = -\left(d^p - a^p\right) \cdot x_j$$

Descente du gradient par l'erreur : technique

$$\Delta_p w_j = -\eta \left[-\left(d^p - a^p\right) \right] x_j$$
$$= \eta \left(d^p - a^p\right) x_j$$

$$\Delta_p w_j = -\eta * \frac{\partial E^p}{\partial w_j}$$
$$= \eta \cdot \left(d^p - a^p\right) \cdot x_j$$

$$\Delta_p w_j = \eta \, \delta^p x_j \quad \text{avec} \quad \delta^p = d^p - a^p$$

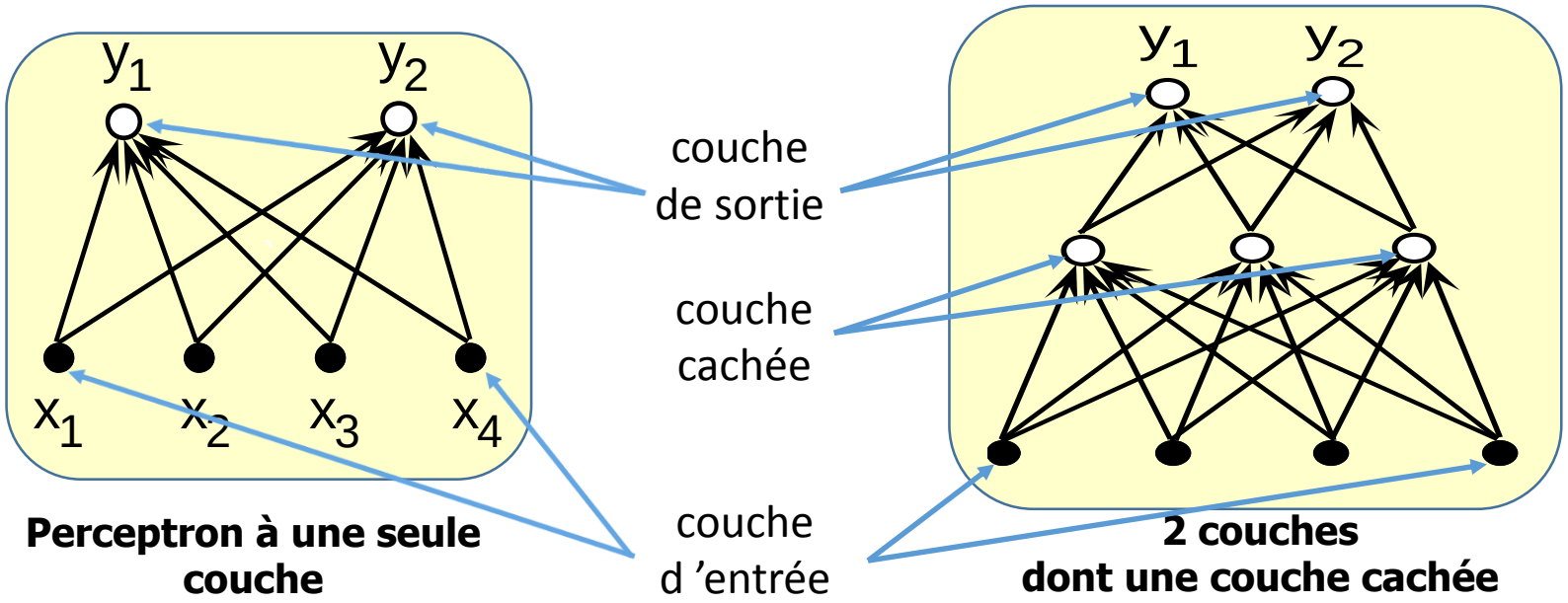
Δ = la différence entre la sortie désirée et la sortie actuelle pour la forme p

Actualisation des poids

$$w_j^p(t+1) = w_j^p(t+1) + \Delta_p w_j$$

Le perceptron proposé par Rosemblat

Il est du type feed-forward



Conventions

- compter les unités de la couche d'entrée
- ne pas compter la couche d'entrée

Adopter

- on choisi de **ne pas les compter**
- on dira qu'un réseau avec une couche cachée est un réseau à deux couches

À l'étude

dans notre étude, on se restreint à un réseau à 1 couche : ***perception simple***

Problème

- On considère le cas linéairement séparable
- On voudrait déterminer les poids appropriés

Procédure

- Commencer avec une simple initialisation des poids
- Utiliser la règle **Delta** pour l'entraînement

- Entrer les exemples d'apprentissage un à un
- Trouver la sortie pour chaque exemple
- Si la sortie obtenue correspond à la sortie on ne fait rien
- sinon, on adapte nos poids selon

$$\Delta_p w_j = \eta \left(d^p - a^p \right) x_j$$

L'adaptation se fait selon :

$$w_j^{\text{new}} = w_j^{\text{old}} + \Delta_p w_j$$

η taux d 'apprentissage

d^p sortie désirée

Le but du perceptron est d'apprendre une transformation

$$T : \{-1, 1\}^N \rightarrow \{-1, 1\}$$

La sortie du perceptron est :

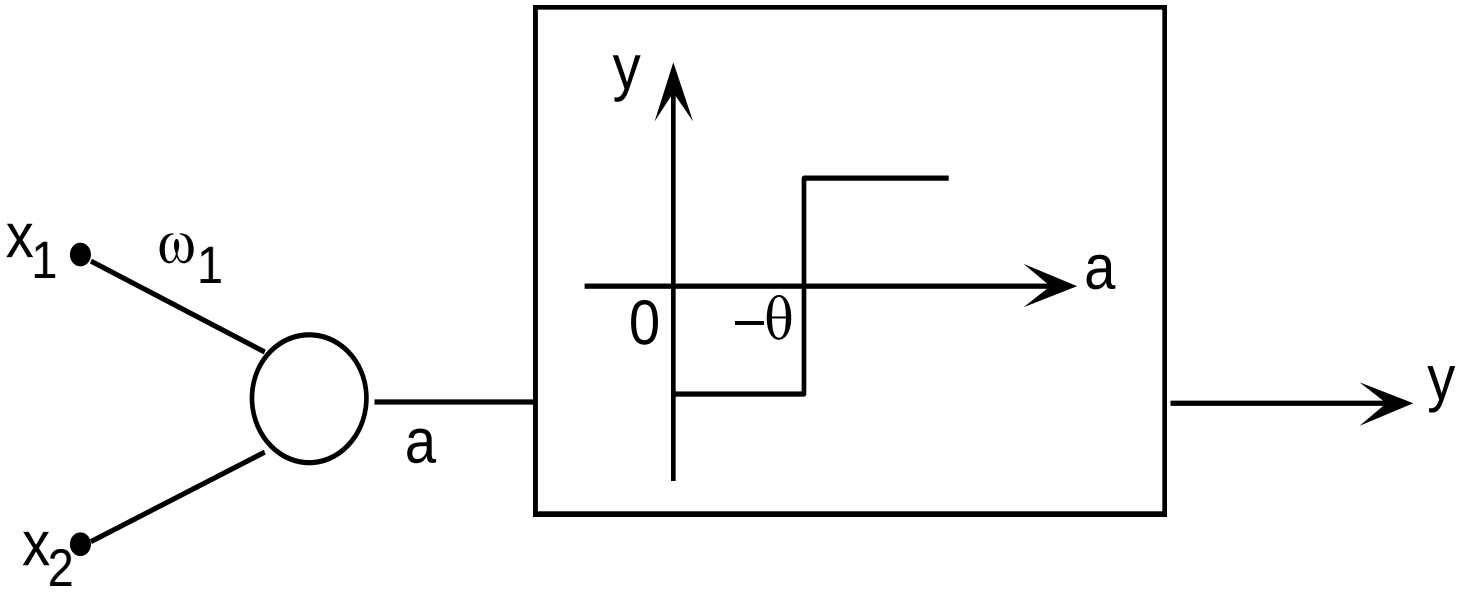
$$y = F\left(\sum_{i=1}^{N} w_i x_i + \theta\right)$$

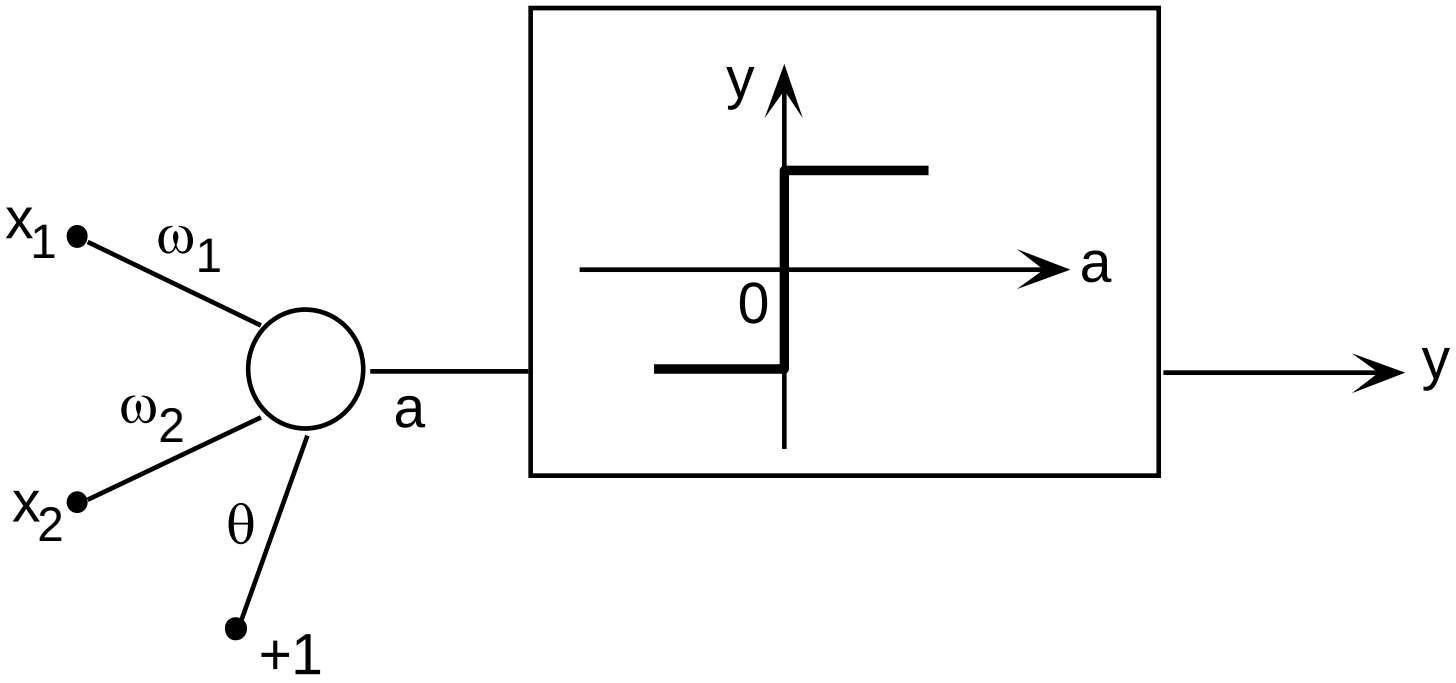
Fonction d'activation

activation

Exemple : fonction logique

$$F(a) = \begin{cases} 1 & \text{si } a > 0 \\ -1 & \text{sinon} \end{cases}$$


x1
ω1
x2
a
y
a
0
–θ
y


x1
ω1
ω2
x2
θ
+1
a
y
0
a
y

Concevoir un perceptron pour générer le ET logique

La table de vérité du ET

x_0	x_1	**y**
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	+1

Concevoir un perceptron pour générer le ET logique

$$a = w_1 x_1 + w_2 x_2 + \theta$$

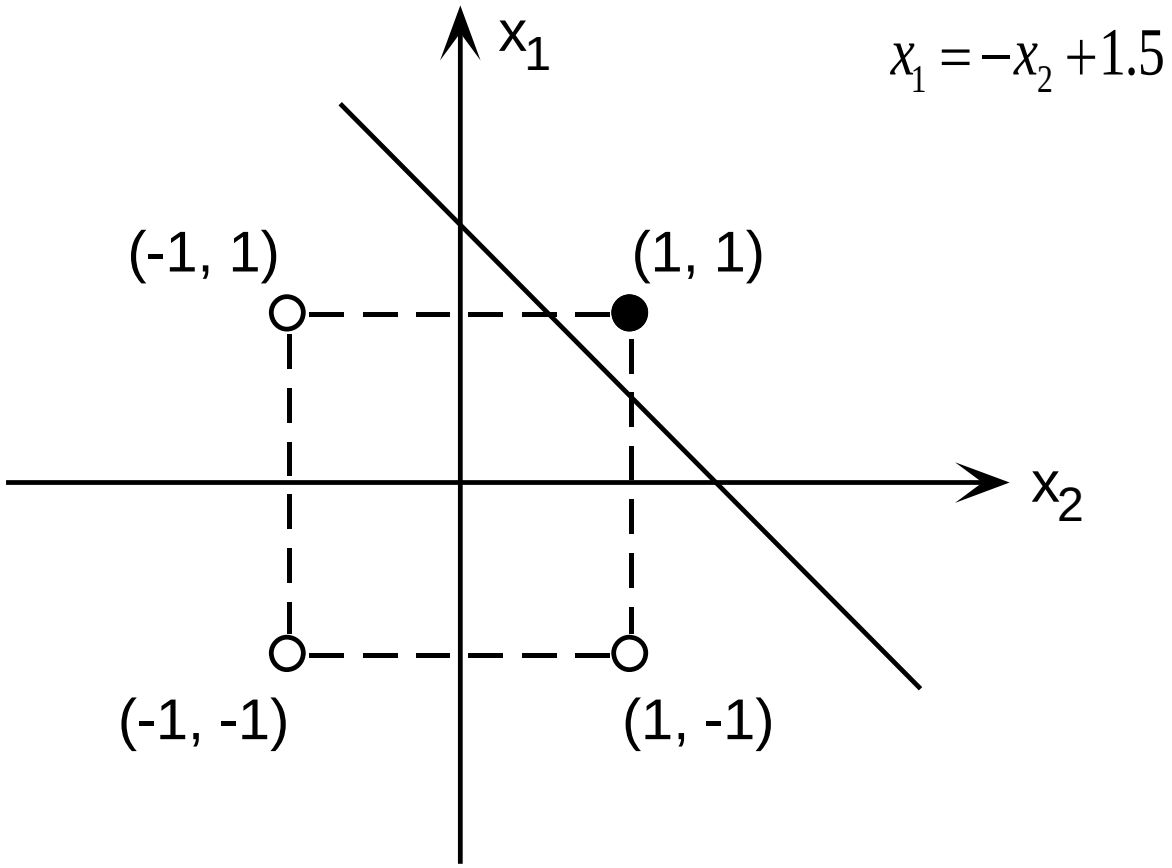
$$y = \begin{cases} 1 & a > 0 \\ -1 & a \leq 0 \end{cases}$$

$$w_1 = 1 \quad w_2 = 1 \quad \theta = -1{,}5$$

$$w_1 x_1 + w_2 x_2 + \theta = 0 \quad \rightarrow \quad x_1 = -\frac{w_2}{w_1} x_2 - \frac{\theta}{w_1}$$

$$= -x_2 + 1.5$$

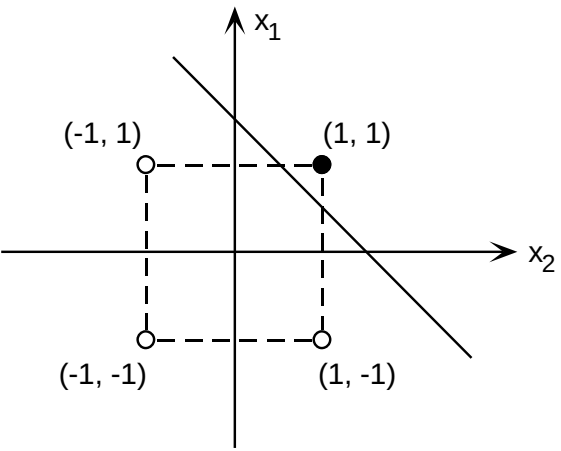
Plan de séparation



$$x_1 = -x_2 + 1.5$$

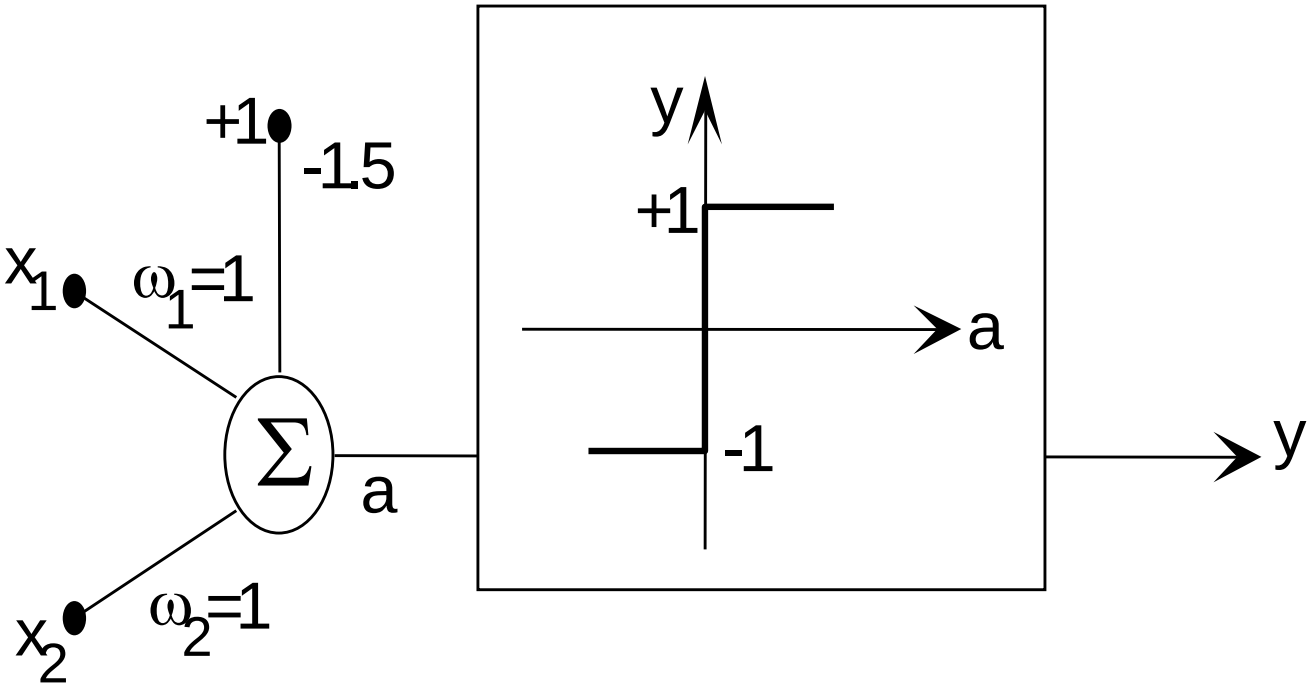
Vérification

$$a = w_1 x_1 + w_2 x_2 + \theta = x_1 + x_2 - 1.5$$



x_1	x_2	a	y
-1	-1	-3.5	-1
-1	1	-1.5	-1
1	-1	-1.5	-1
1	1	0.5	+1

Configuration du perceptron

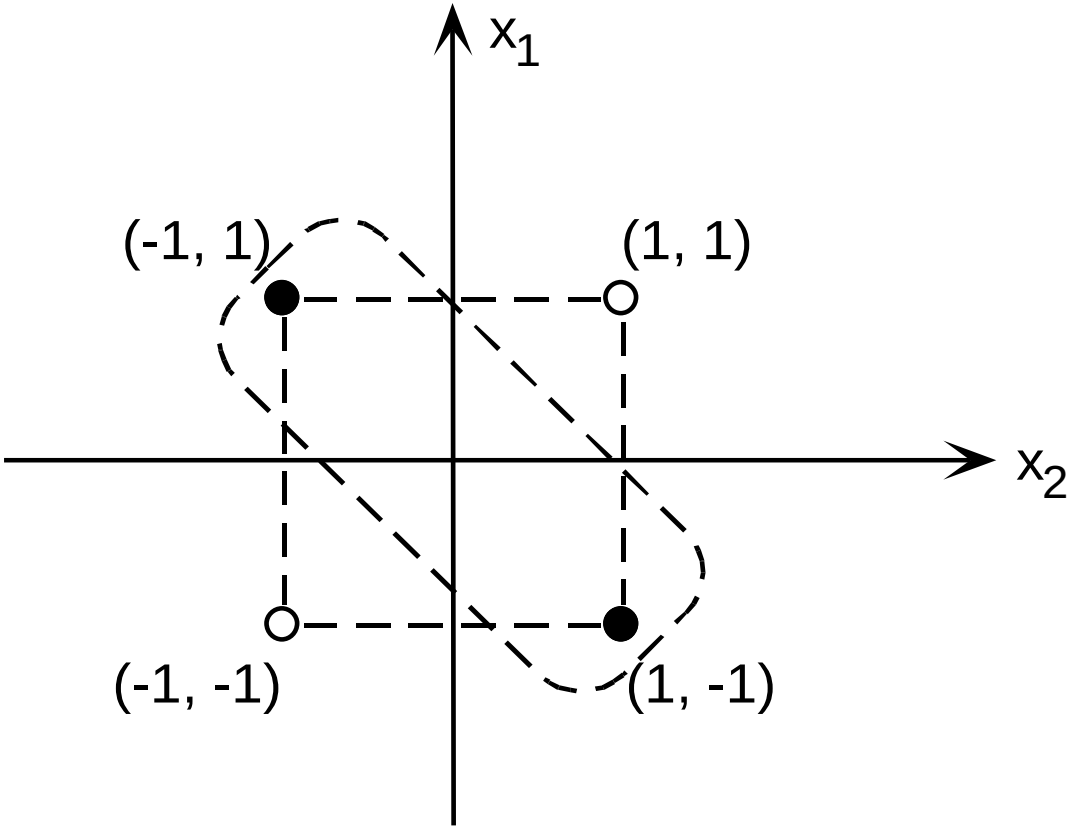


Caractéristique du XOR

La table de vérité du XOR est

x_0	x_1	**y**
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

Plan de séparation ?

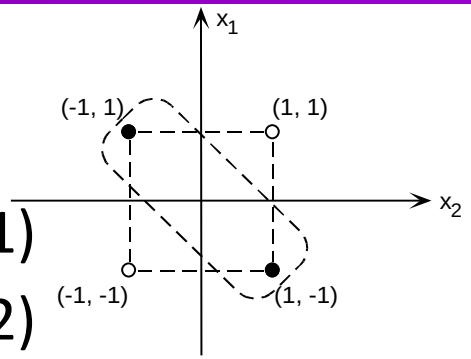


Données non linéairement séparables

Impossible de représenter cette fonction par un simple perceptron

Résultat algébrique

$$a = w_1 x_1 + w_2 x_2 + \theta$$



Classe -1

$x_1 = -1 \quad x_2 = -1 \quad -w_1 - w_2 + \theta < 0 \quad \rightarrow \quad -w_1 - w_2 < -\theta$ (1)

$x_1 = 1 \quad x_2 = 1 \quad w_1 + w_2 + \theta < 0 \quad \rightarrow \quad w_1 + w_2 < -\theta$ (2)

Classe +1

$x_1 = -1 \quad x_2 = 1 \quad -w_1 + w_2 + \theta > 0 \quad \rightarrow \quad -w_1 + w_2 > -\theta$ (3)

$x_1 = 1 \quad x_2 = -1 \quad w_1 - w_2 + \theta > 0 \quad \rightarrow \quad w_1 - w_2 > -\theta$ (4)

$$(2) \text{ et } (3) \quad \Leftrightarrow \quad \begin{cases} w_1 + w_2 < -\theta \\ -w_1 + w_2 > -\theta \quad \Rightarrow \quad w_1 - w_2 < \theta \end{cases} \quad \Leftrightarrow \quad w_1 < 0$$

$$(1) \text{ et } (4) \quad \Leftrightarrow \quad \begin{cases} -w_1 - w_2 < -\theta \quad \Rightarrow \quad w_1 + w_2 > \theta \\ w_1 - w_2 > -\theta \end{cases} \quad \Leftrightarrow \quad w_1 > 0$$

Marche pas

Problème

Il y a plus de formes que d'entrées

Dans le cas du XOR :

- 4 formes (-1, -1) , (-1, 1) , (1, -1) , (1, 1)
- 2 entrées

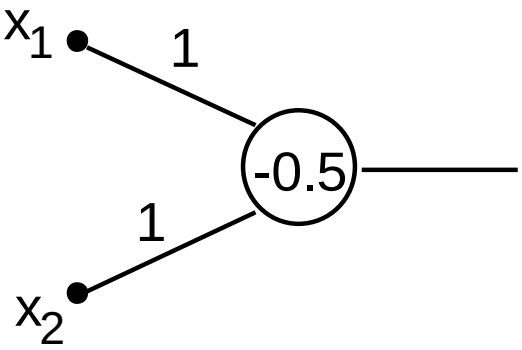
La non-séparabilité linéaire

Objectif

Pour résoudre le problème de la **non-séparabilité**, on peut faire une extension au **perceptron multicouche**

Stratégie

- Démarrer avec une configuration minimale (1 couche et un neurone)
- Rajouter des neurones et des couches
- Critère d'arrêt : classification satisfaisante



$$a = w_1 x_1 + w_2 x_2 + \theta$$

$$F(a) = \begin{cases} 1 & \text{si } a > 0 \\ -1 & \text{sinon} \end{cases}$$

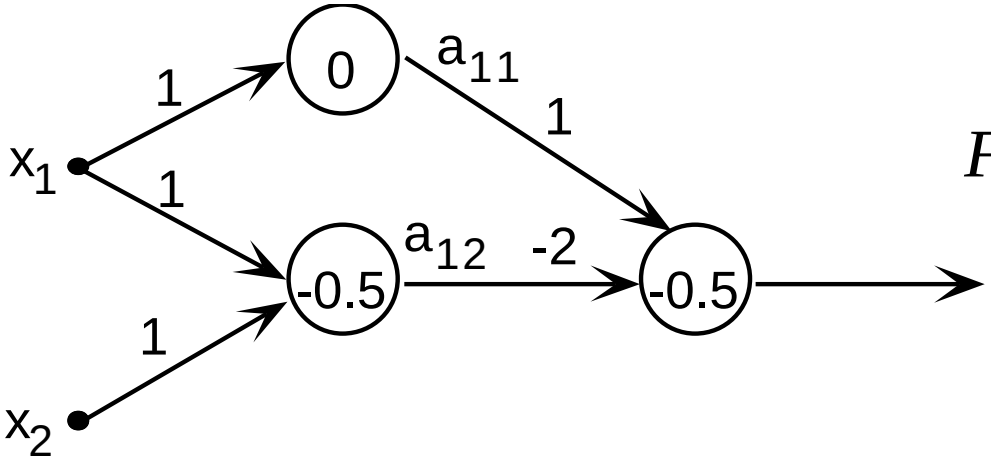
x_1	x_2	**a**	$y = F(a)$	**d**
-1	-1	-2.5	-1	-1
-1	+1	-0.5	-1	+1

Marche pas

1ere solution

Ajouter une deuxième couche

$$a = w_1 x_1 + w_2 x_2 + \theta$$



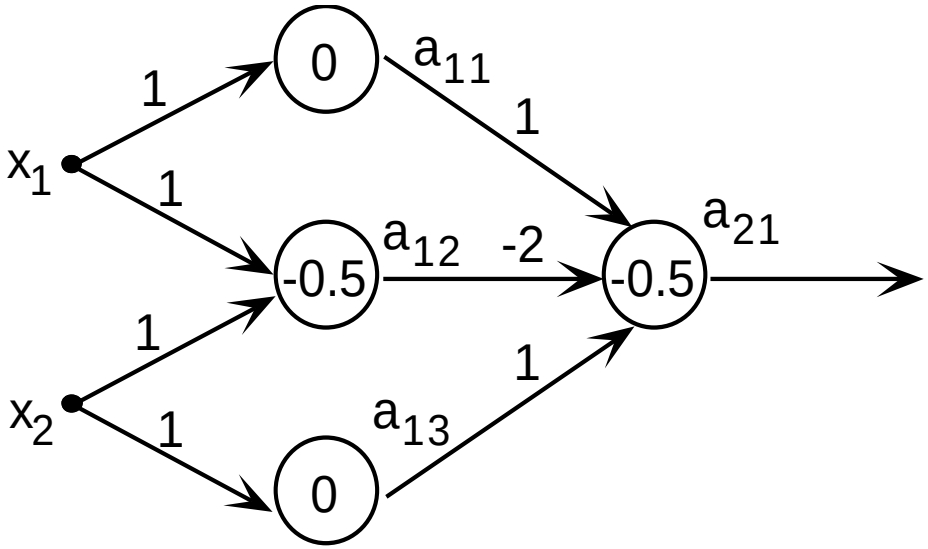
$$F(a) = \begin{cases} 1 & \text{si } a > 0 \\ -1 & \text{sinon} \end{cases}$$

Entrée		Couche cachée				Couche de sortie		
x_1	x_2	a_{11}	$F(a_{11})$	a_{12}	$F(a_{12})$	a	F(a)	d
-1	-1	-1	-1	-2.5	-1	0.5	+1	-1

Marche pas

2eme solution

Ajouter une unité dans la couche cachée



$$a = w_1 x_1 + w_2 x_2 + \theta$$

$$F(a) = \begin{cases} 1 & \text{si } a > 0 \\ -1 & \text{sinon} \end{cases}$$

Entrée		Couche cachée						Couche de sortie		
x_1	x_2	a_{11}	$F(a_{11})$	a_{12}	$F(a_{12})$	a_{13}	$F(a_{13})$	a_{21}	$F(a_{21})$	d
-1	-1	-1	-1	-2.5	-1	-1	-1	-0.5	-1	-1
-1	+1	-1	-1	-0.5	-1	+1	+1	+1.5	+1	+1
+1	-1	+1	+1	-0.5	-1	-1	-1	+1.5	+1	+1
+1	+1	+1	+1	+1.5	+1	+1	+1	-0.5	-1	-1

Ça Marche!

Lors de l'apprentissage, construire le réseau couche par couche en itérant le processus suivant :

Unité maîtresse

- Construire une unité réalisant une classification partielle des représentations de motifs. À la fin de cette étape, si la classification n'est pas terminée, certains motifs sont mal classés

Unités auxiliaires

- Rajouter à l'unité maîtresse précédente d'autres unités

Complexité

Le nombre d'unités d'entrée et de sortie dépend du problème à traiter

Nombre d'unités

Faire un compromis :

- Optimisant l'apprentissage
- Évitant le sur-apprentissage qui sera la conséquence d'un trop grand nombre d'unités cachées

Hardware

Possible de produire des circuits dédiés

Problème

Le perceptron ne présente pas de solution au problème de l'ajustement des poids de l'entrée à la couche cachée

Solution

L'erreur des unités de la couche cachée est déterminée en propageant les erreurs des unités de sortie vers les unités cachées

Rétropropagation (back-propagation)

Note

La rétropropagation est une généralisation de la règle delta pour des fonctions d'activation non linéaires et des réseaux multicouches

Apprentissage

La caractéristique fondamentale de l'apprentissage est la règle du gradient stochastique.

Règle de delta généralisée

Caractéristiques

Les unités sont connectées aux unités du niveau suivant.
Le réseau est sans rétroaction

Rétropropagation (back-propagation)

Limitation

Données non linéairement séparables
Apprentissage de la couche cachée

Objectif

Trouver une méthode pour corriger les erreurs des unités cachées

Caractéristiques

Généralisation de la règle de Delta
Propagation de l'erreur de la sortie vers l'entrée

Calcul du signal d'erreur par le principe de rétroaction (règle de delta généralisée)

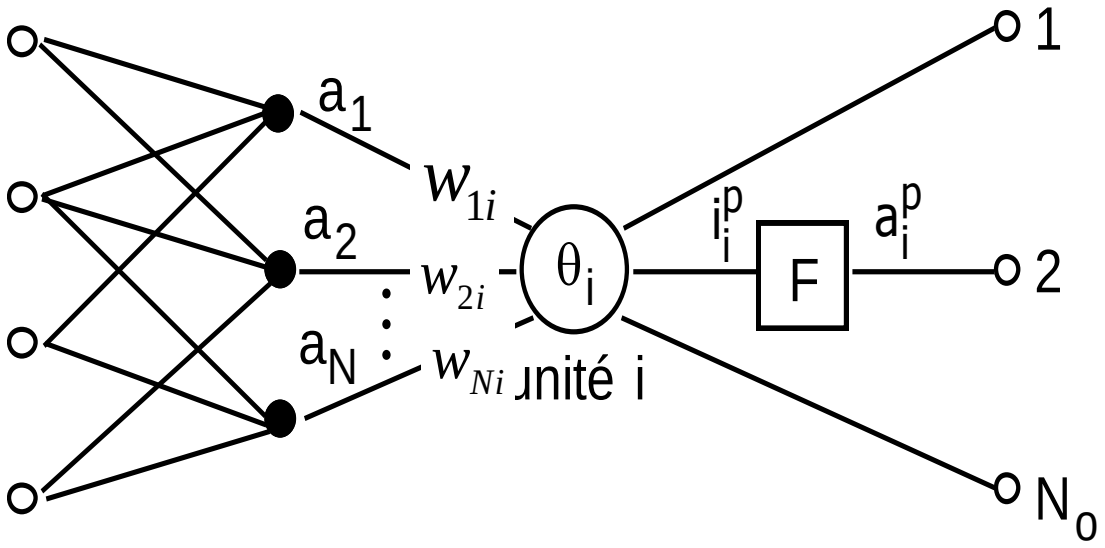
- on soumet au réseau des exemples
- On connaît leur famille d 'appartenance
- La sortie désirée
- Correction des poids en comparant

(Sortie obtenue , Sortie désirée)

Calcul de l'activation

$$i_i^p = \sum_k w_{ki} a_k^p + \theta_i$$

$$a_i^p = F\left(i_i^p\right)$$



$$\Delta_p w_{ij} = -\eta \times \left(\text{pente de } E^p \,\%\, w_{ij}\right)$$

$$= -\eta \frac{\partial E^p}{\partial w_{ij}}$$

L'activation est fonction de toutes les entrées

Règle de delta

$$E^p = \frac{1}{2}\sum_{i=1}^{N}\left(d_i^p - a_i^p\right)^2$$

$$\Delta_p w_{ij} = -\eta \frac{\partial E^p}{\partial w_{ij}}$$

$$= -\eta \frac{\partial E^p}{\partial i_i^p} \cdot \frac{\partial i_i^p}{\partial w_{ij}}$$

$$= -\eta\left(-\delta_i^p\right)\cdot a_j^p = \eta \delta_i^p a_j^p$$

$$\Delta_p w_{ij} = \eta\, \delta_j^p\, a_j^p$$

Simple perceptron $\left(d^p - a^p\right)$

Problème

Trouve une technique de mise à jour des poids dans le cas d'un réseau multicouche

Solution

Le calcul de δ se fait par un calcul récursif en propageant l'erreur de la sortie vers l'entrée.

Démo

$$\delta_i^p = -\frac{\partial E^p}{\partial i_i^p}$$

$$= -\frac{\partial E^p}{\partial a_i^p} \cdot \frac{\partial a^p}{\partial i_i^p}$$

Erreur fonction de la sortie d'activation

La sortie en fonction de l'entrée

Calcul des termes $\frac{\partial a_i^p}{\partial i_i^p}$ et $\frac{\partial E^p}{\partial a_i^p}$

Règle de delta généralisée

$$\delta_i^p = -\frac{\partial E^p}{\partial a_i^p} \cdot \frac{\partial a^p}{\partial i_i^p}$$

Calcul de $\frac{\partial a_i^p}{\partial i_i^p}$

$$a_i^p = F\left(i_i^p\right) \quad \Rightarrow \quad \frac{\partial a_i^p}{\partial i_i^p} = F'\left(i_i^p\right)$$

Calcul de $\frac{\partial E^p}{\partial a_i^p}$

Deux cas :

- i unité de sortie
- i unité de la couche cachée

Calcul de $\frac{\partial E^p}{\partial a_i^p}$

Rappel

$$\frac{\partial a_i^p}{\partial i_i^p} = F'\left(i_i^p\right)$$

$$\delta_i^p = -\frac{\partial E^p}{\partial a_i^p} \cdot \frac{\partial a^p}{\partial i_i^p}$$

Si i unité de sortie

$$\frac{\partial E^p}{\partial a_i^p} = -\left(d_i^p - a_i^p\right)$$

$$\delta_i^p = \left(d_i^p - a_i^p\right) \cdot F'\left(i_i^p\right)$$

$$\Delta_p w_{ij} = \eta \delta_i^p a_j^p$$

$$= \eta \left(d_i^p - a_i^p\right) \cdot F'\left(i_i^p\right) a_j^p$$

Calcul de $\frac{\partial E^p}{\partial a_i^p}$

Si i unité de la couche cachée

$$\frac{\partial E^p}{\partial a_i^p} = \sum_{h=1}^{N} \frac{\partial E^p}{\partial i_h^p} \cdot \frac{\partial i_h^p}{\partial a_i^p}$$

$$= \sum_{h=1}^{N} \frac{\partial E^p}{\partial i_h^p} \cdot \frac{\partial}{\partial a_i^p} \sum_{k=1}^{N_h} w_{kh} a_k^p,$$

$$= \sum_{h=1}^{N} \frac{\partial E^p}{\partial i_h^p} \cdot w_{ih}$$

$$= \sum_{h=1}^{N} \left(-\delta_h^p\right) \cdot w_{hi} = -\sum_{h=1}^{N} \delta_h^p \, w_{ih}$$

N_h : nombre d'unité dans la couche cachée

Calcul de δ_i^p

$$\delta_i^p = -\frac{\partial E^p}{\partial a_i^p} \cdot \frac{\partial a^p}{\partial i_i^p}$$

Rappel

$$\frac{\partial a_i^p}{\partial i_i^p} = F'\left(i_i^p\right)$$

$$\frac{\partial E^p}{\partial a_i^p} = -\sum_{h=1}^{N} \delta_h^p \, w_{ih}$$

$$\delta_i^p = -\left[-\sum_{h=1}^{N} \delta_h^p \, w_{ih}\right] F'\left(i_i^p\right)$$

$$= F'\left(i_i^p\right) \sum_{h=1}^{N} \delta_h^p \, w_{ih}$$

$$\Delta_p w_{ij} = \eta \, \delta_j^p \, a_i^p$$

$$= \eta \left(F'\left(i_i^p\right) \sum_{h=1}^{N} \delta_h^p \, w_{ih} \right) a_i^p$$

Règle de delta généralisée

Actualiser les poids selon

$$\Delta_p w_{ij} = \eta \delta_j^p a_i^p$$

Si i unité de sortie

$$\delta_i^p = \left(d_i^p - a_i^p\right) F_i'\left(i_i^p\right)$$

Si i unité de la couche cachée

$$\delta_i^p = F'\left(i_i^p\right) \sum_{h=1}^{N} \delta_h^p w_{ih}$$

N:nombre d'unités de la couche supérieure directement connectées à i

Comment calculer F'?

Sigmoïde

$$a_i^p = F\left(i_i^p\right) = \frac{1}{1+e^{-i_i^p}} \Rightarrow F'\left(i_i^p\right) = \frac{\partial}{\partial i_i^p}\left[\frac{1}{1+e^{-i_i^p}}\right] = a_i^p\left(1-a_i^p\right)$$

i unité de sortie

$$\delta_i^p = \left(d_i^p - a_i^p\right)F_i'\left(i_i^p\right) = \left(d_i^p - a_i^p\right)a_i^p\left(1-a_i^p\right)$$

i unité de la couche cachée

$$\delta_i^p = F'\left(i_i^p\right)\sum_{h=1}^{N}\delta_h^p w_{hi} = a_i^p\left(1-a_i^p\right)\sum_{h=1}^{N}\delta_h^p w_{hi}$$

En partant de l'entrée vers la sortie

Pour chaque couche c, c=1, …, C

{

Pour chaque unité i=1, …, N_c

{

$$i_i^c = \sum_k^{N_c} \omega_{ki}^c x_k + \theta_i^c \quad c = 1$$

$$= \sum_k^{N_c} \omega_{ki}^c a_k^{c-1} + \theta_i^c \quad c > 1$$

} $a_i^{(c)} = F\left(i_i^{(c)}\right)$

}

En partant de la sortie vers l'entrée

Pour chaque couche c, c=C, …, 1

{

Pour chaque unité i=1, …, N_c

{

Si i unité de sortie $\delta_i^{(c)} = \left(d_i^{(c)} - a_i^{(c)}\right) F'\left(i_i^{(c)}\right)$

i unité de la couche cachée $\delta_i^{(c)} = F'\left(i_i^{(c)}\right) \sum_{h=1}^{N} \delta_h^{(c+1)} w_{ih}^{(c+1)}$

}

}

En partant de l'entrée vers la sortie

Pour chaque couche c, c=1, …, C

{

$$\Delta \omega_{ij}^{(c)} = \begin{cases} \eta \, \delta_j^{(c)} \, a_i^{(c-1)} & c > 1 \\ \eta \, \delta_j^{(c)} \, x_i & c = 1 \end{cases}$$

}

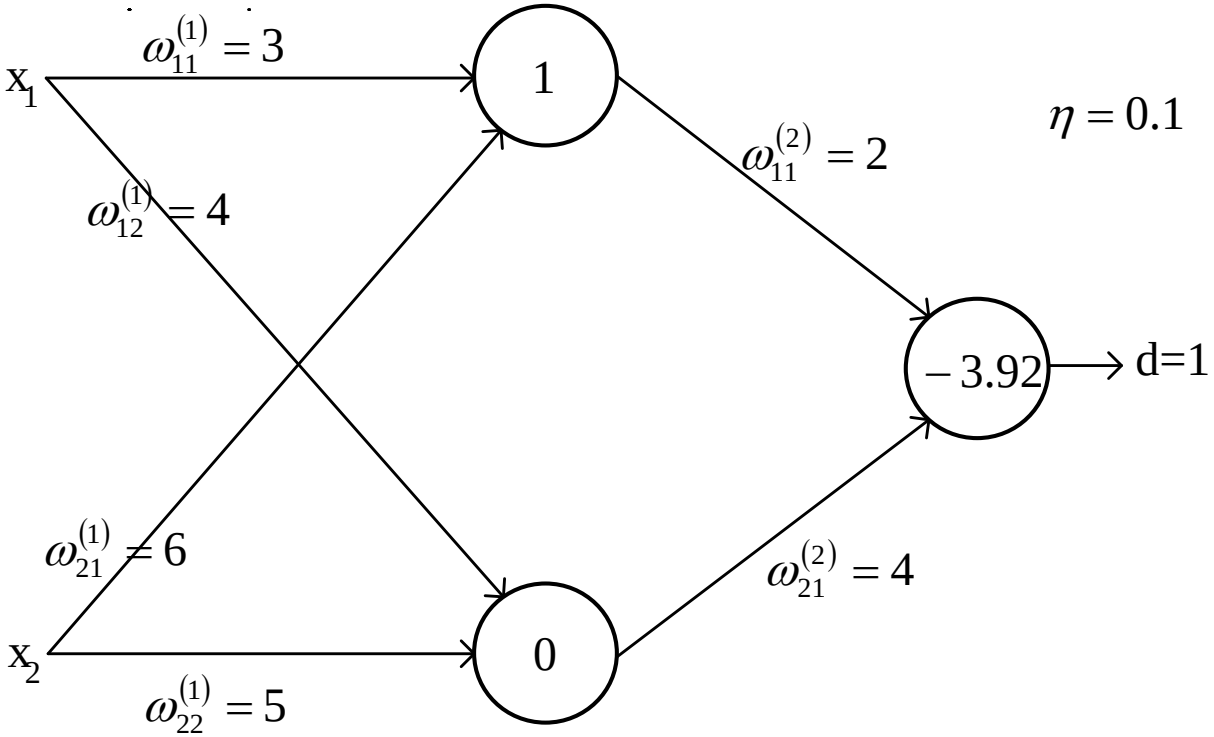
En partant de l'entrée vers la sortie

Pour chaque couche c, c=1, …, C

{

$$\omega_{ij}^{new} = \omega_{ij}^{old} + \Delta\omega_{ij}$$

}



Apprentissage

Entrée $(x_1, x_2) = (1, 0)$

Trouver les nouvelles valeurs des poids du réseau en appliquant la règle de delta généralisée

Entraînement

Long processus d'entraînement
⇒ utilisation non optimale de η et α

Structure rigide

Nombre de couches et d'unités sont fixes et ne peuvent changer

Paralysie du réseau

aucun apprentissage ne sera alors possible
les poids peuvent atteindre des valeurs asymptotiques

Minimum local

la MLP est basée sur la descente du gradient
⇒ peut converger vers un minimum local

Problème

Les poids sont modifiés par cette formule pour fournir exactement les sorties désirées pour le dernier exemple présenté sans tenir compte des corrections apportées précédemment pour obtenir les résultats désirés pour les exemples présentés préalablement.

L'apprentissage conduit à des oscillations des valeurs des poids alors que l'on considère que l'apprentissage est fini lorsque les poids sont stabilisés.

Solution

Pour pallier cet inconvénient, on ajoute au terme précédent un terme proportionnel à la variation des poids.

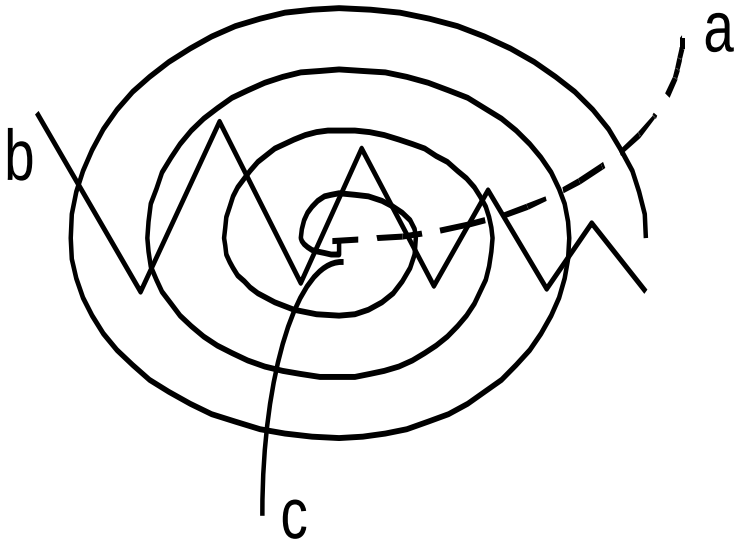
Si $\Delta w_{ij}(n)$ est la variation des poids après la présentation du n^{ième} exemple, alors :

$$\Delta w_{ij}(n+1) = \eta \delta_j^p a_i^p + \alpha \Delta w_{ij}(n)$$

$0 < \eta < 1$, taux d'apprentissage

$0 < \alpha < 1$, le momentum

~0.5

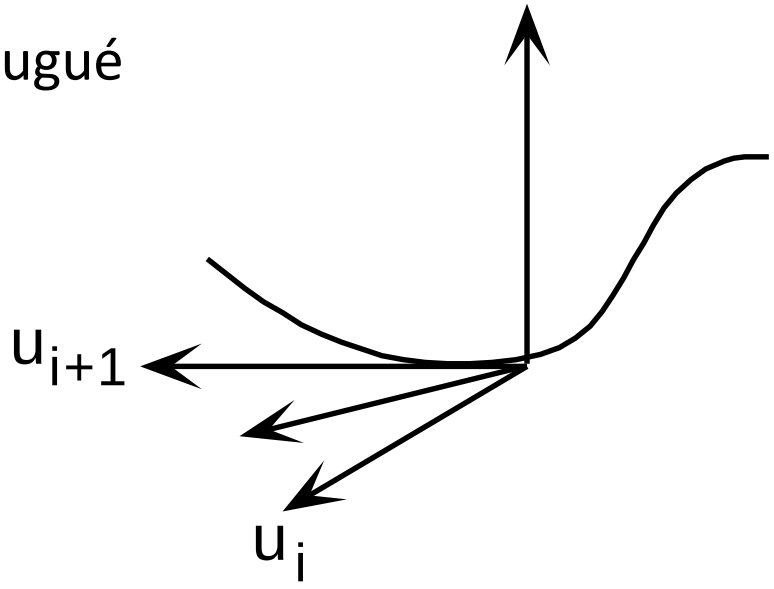


a : taux d'app. petit
b : grand taux d'app.
c : grand taux d'app. + momentum

Descente dans l'espace des poids

Éviter de tomber dans un minimum local

- Utilisation du recuit simulé (Simulated annealing)
- Minimisation du gradient conjugué



$$Si \ \frac{\partial E(n+1)}{\partial w_{ij}} \ et \ \frac{\partial E(n)}{\partial w_{ij}} \ \text{sont de même signe}$$

$$\eta_{ij}(n+1) = u\,\eta_{ij}(n)$$

Sinon

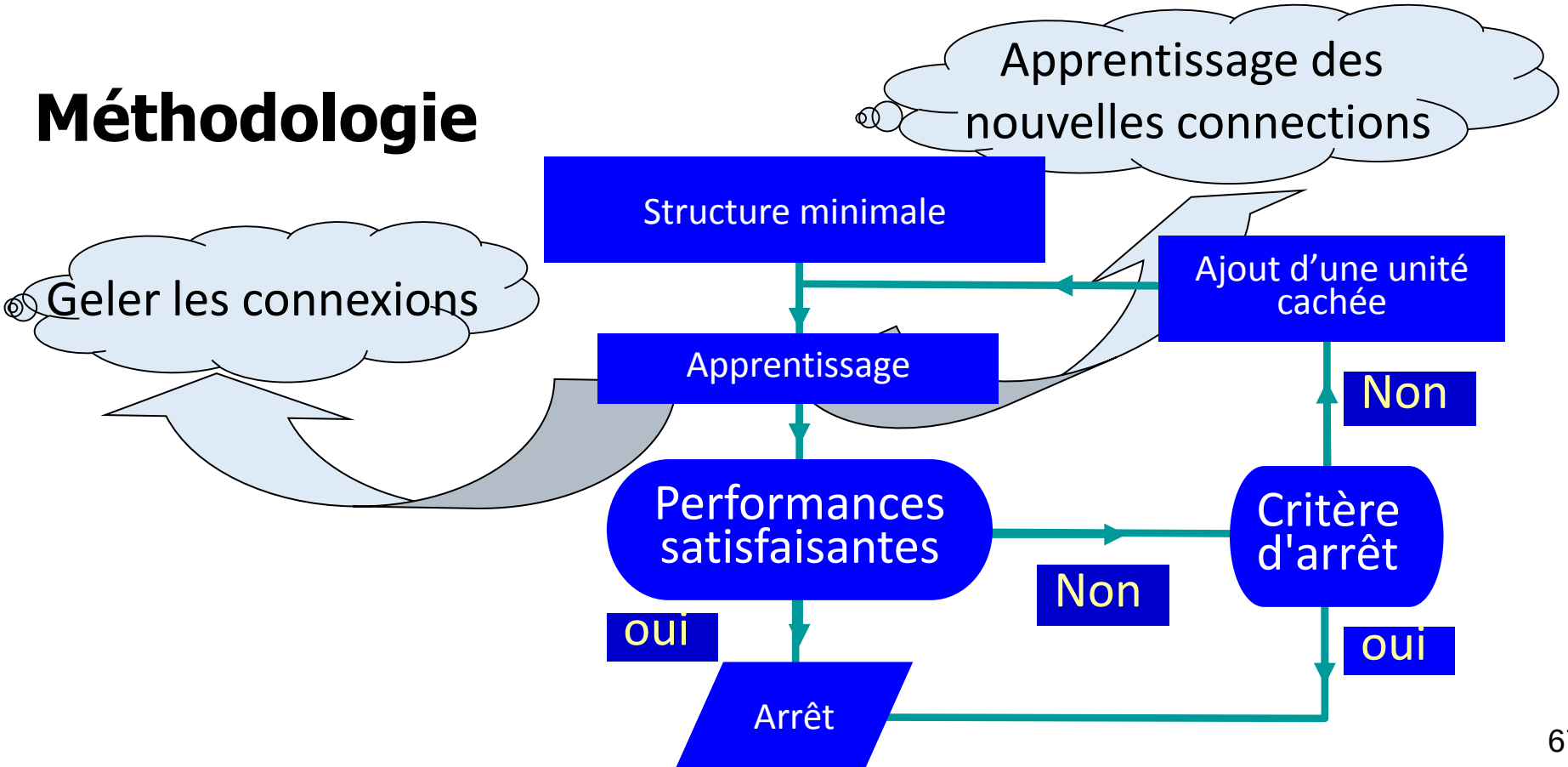
$$\eta_{ij}(n+1) = d\,\eta_{ij}(n)$$

u et d sont des constantes positives de valeurs légèrement plus grandes et plus petites que 1 respectivement.

Limitation

Ne s'applique qu'à un réseau dont le nombre d'unités cachées est fixé

Méthodologie



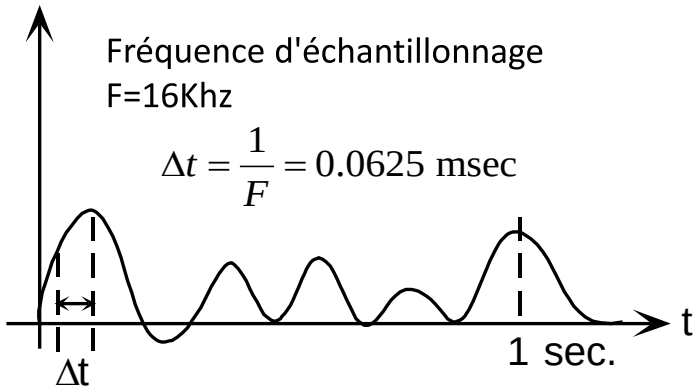
Problème de la reconnaissance de la parole

Deux difficultés dans la conception

Choix et structuration des entrées

Quantité de données dans 1 sec

$$n\Delta t = 1 \text{ sec} \Rightarrow n = \frac{1\text{sec}}{\Delta t} = \frac{1}{0{,}0000625} = 16000$$



Codage sur 8 bits

$$1 \text{ échantillon} \rightarrow 8\text{bits}$$

$$16000 \text{ échantillons} \rightarrow x = 128K\text{bits/Sec}$$

Qualité du signal

128Kbits/sec est largement suffisant

Transition du signal

La succession des phonèmes ➔ distorsions

Quantité de données

Suffisamment d'unités
Bon temps d'apprentissage (distorsions)

Séquentialité du signal

Permet de prendre l'information contextuelle

- *Filtrage*
- *Échantillonage 10KHz*
- *Toutes les 12.8 ms, on calcule sur 256 points le spectre de puissance sur 16 canaux de fréquence en utilisant la FFT*
- *Ce sont les valeurs de ces 16 canaux qui seront traitées comme information de base par le réseau.*

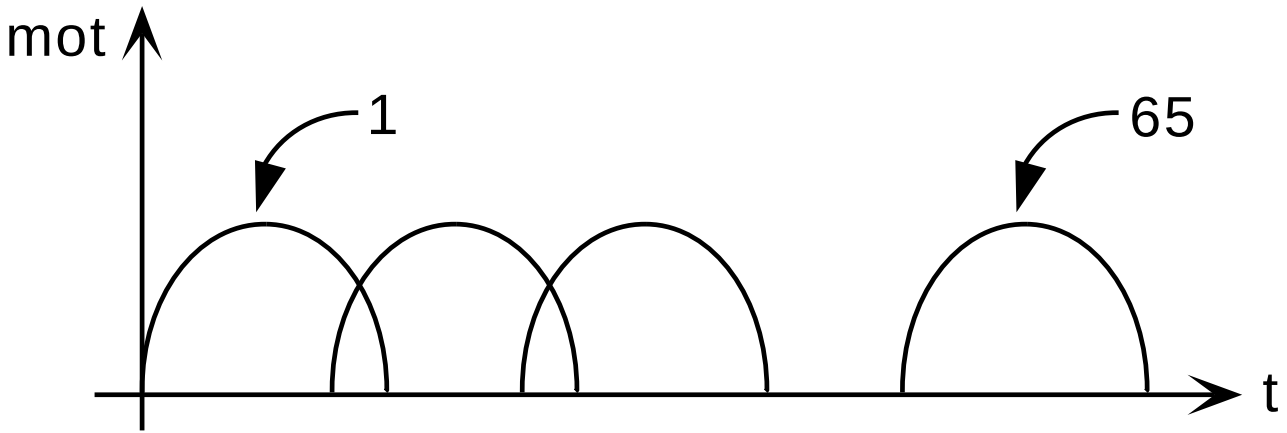
Exemple

Reconnaissance de 5 mots

Le mot le plus long : 832 ms, i.e. qu'il est prononcé à 65 intervalles de 12.8 ms

Une matrice de 16 × 65 unités

Entrée



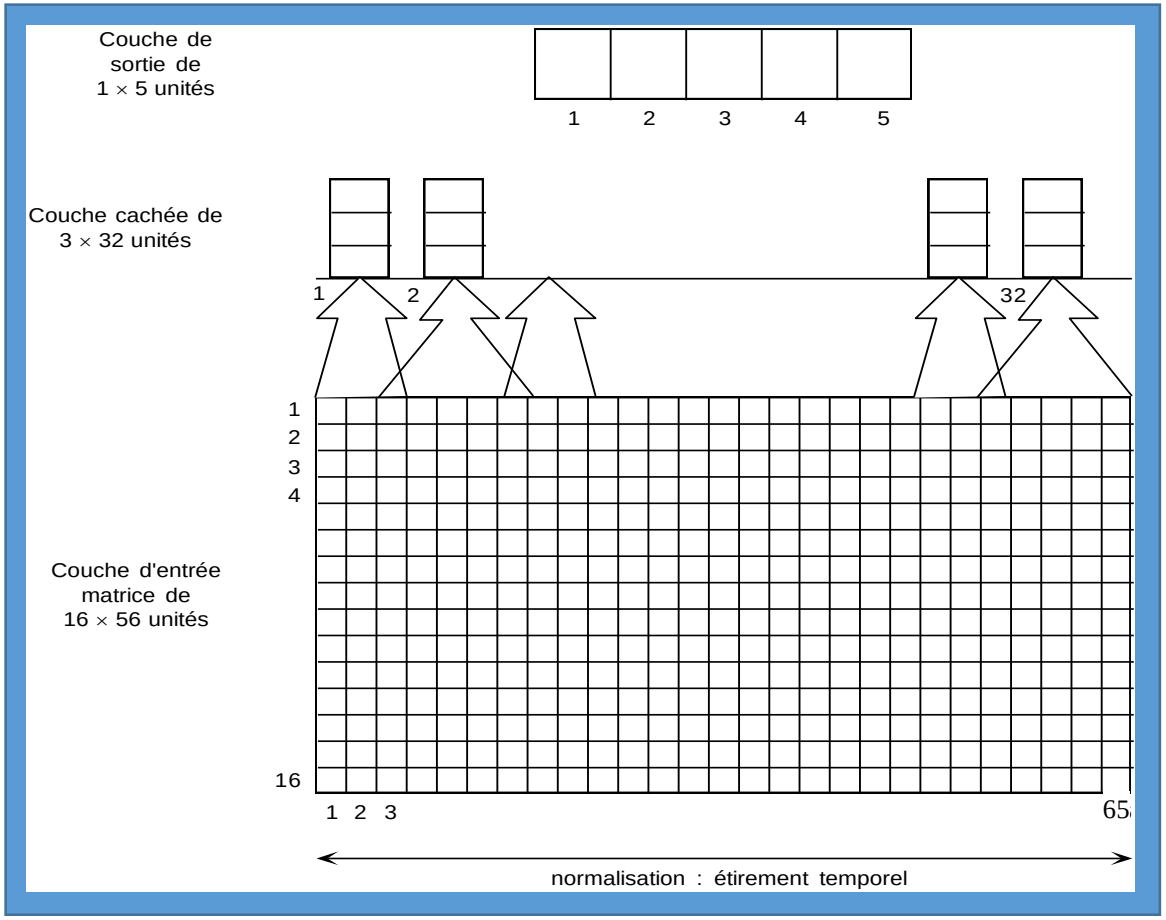
Exemple Reconnaissance de 5 mots

Couche Entrée 16×65 unités

Couche cachée 3×32 unités

Couche de sortie 1×5 unités

Structure Réseau à 2 couches


Couche de sortie de 1 × 5 unités
1 2 3 4 5
Couche cachée de 3 × 32 unités
1 2 32
1 2 3 4
Couche d'entrée matrice de 16 × 56 unités
16
1 2 3
65
normalisation : étirement temporel

Limitation La base d'entraînement inclut tous les mots possibles

NETtalk Proposé par Sejnowski et Rosenberg en 1987 pour la reconnaissance de textes

Permet de généraliser à des mots non connus (lors de l'entraînement)

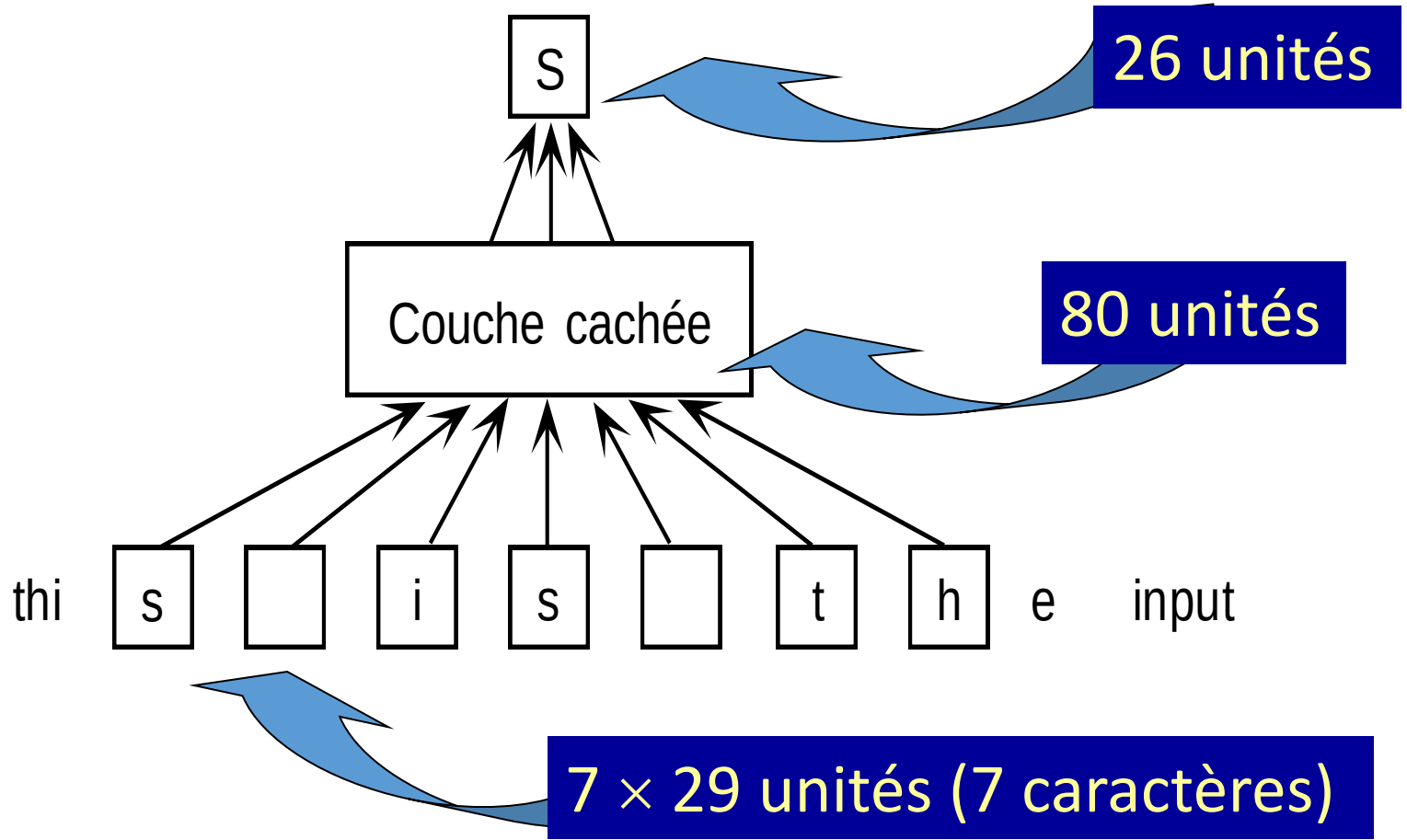
Architecture Réseau à 2 couches

Couche Entrée 7×29 unités (7 caractères)

Couche cachée 80 unités

Couche de sortie 26 unités

Structure Réseau à 2 couches


S
26 unités
Couche cachée
80 unités
thi
s
i
s
t
h
e
input
7 × 29 unités (7 caractères)

Entraînement Sur 1024 mots

Après 10 époques parole intelligible

Après 50 époques 95% sur données d'apprentissage
75% sur données de test

Sonar appareil de détection sous-marine par émission d'ondes sonores

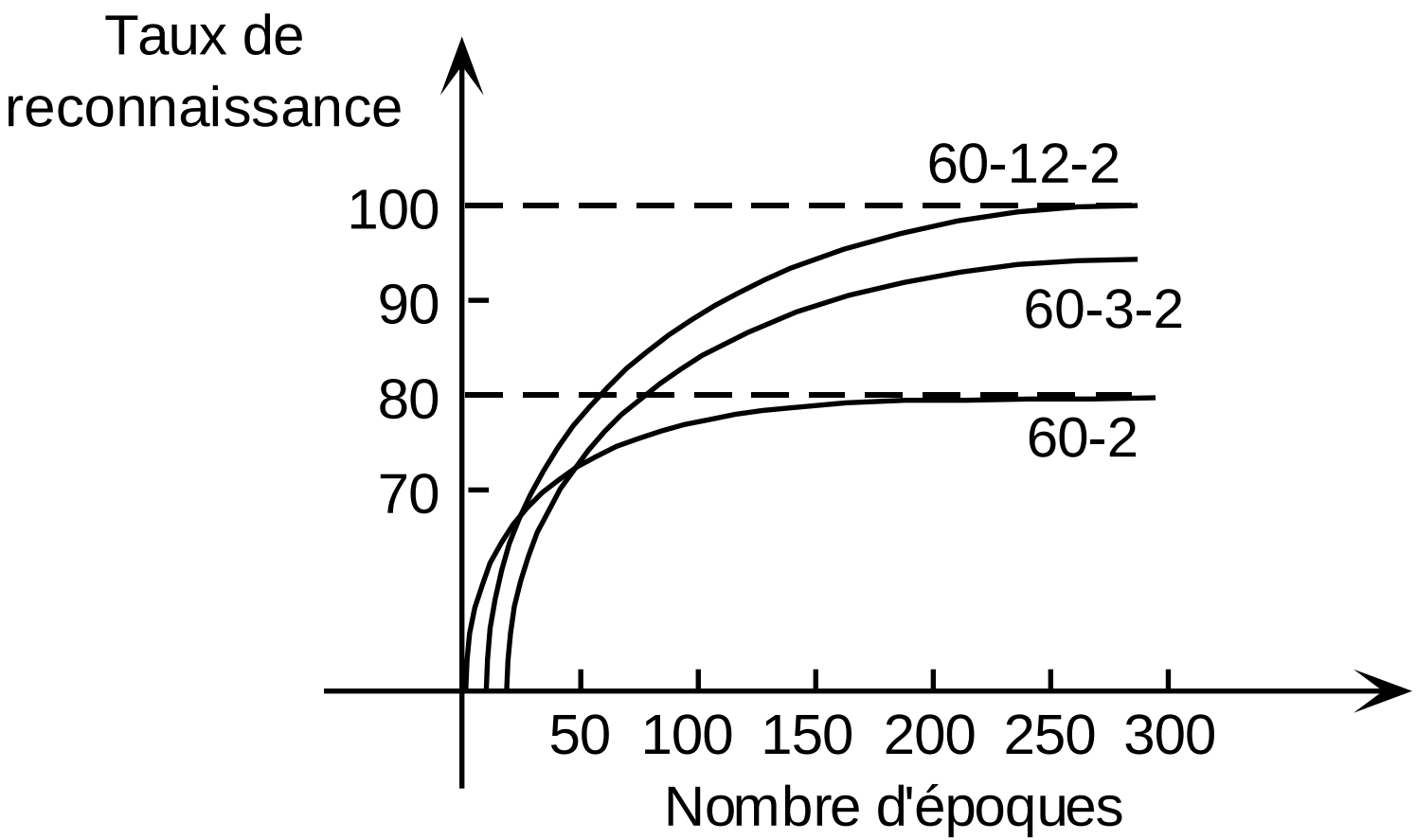
Réseau perceptron à 2 couches pour distinguer entre deux signaux sonar roches et métaux cylindriques

Couche d'entrée 60 unités

Couche cachée de 1 à 24 unités

Couche de sortie 2 unités
- roches
- métaux cylindriques

Structure Réseau à 2 couches

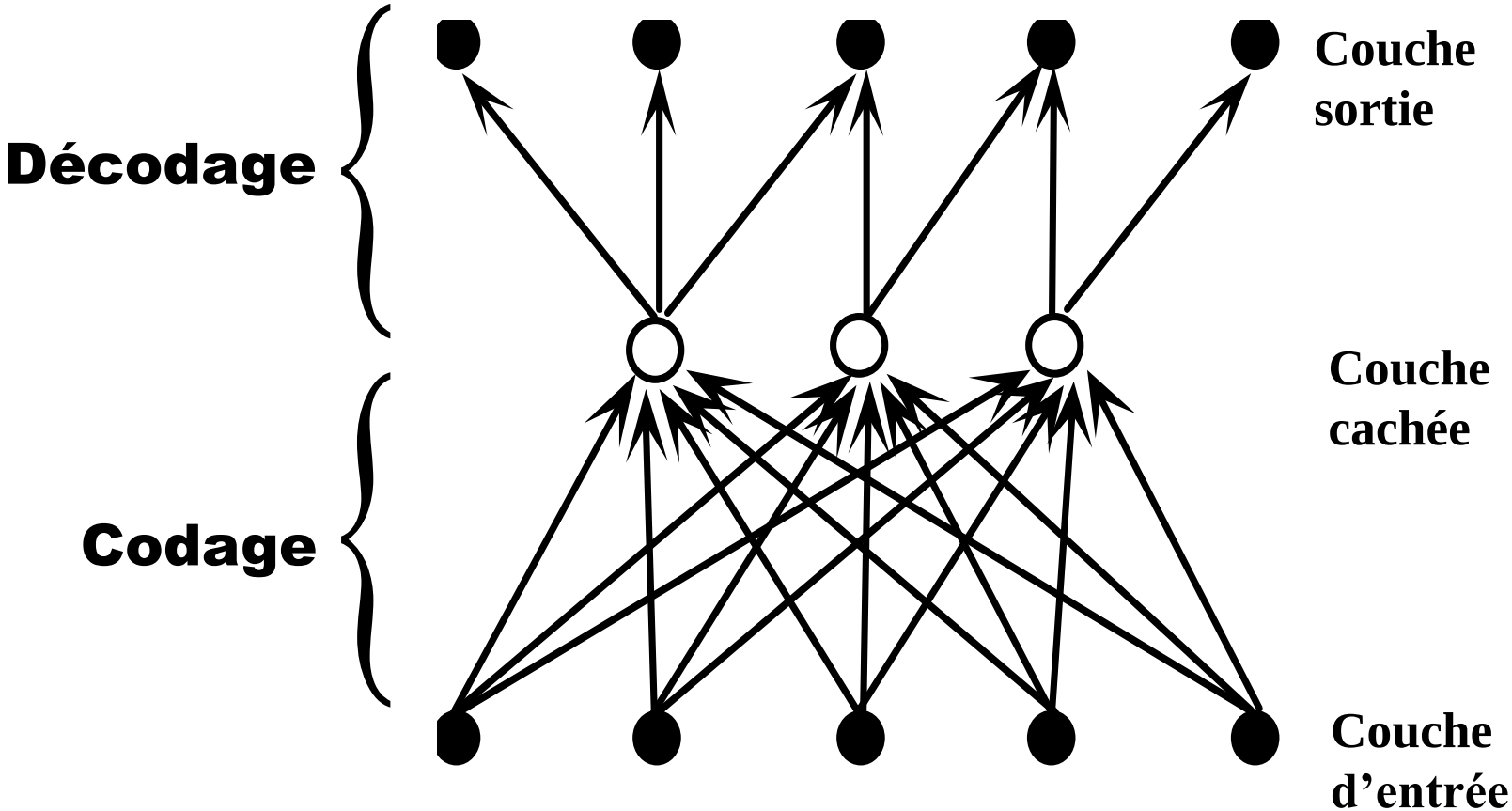

Taux de reconnaissance
100
90
80
70
60-12-2
60-3-2
60-2
50
100
150
200
250
300
Nombre d'époques

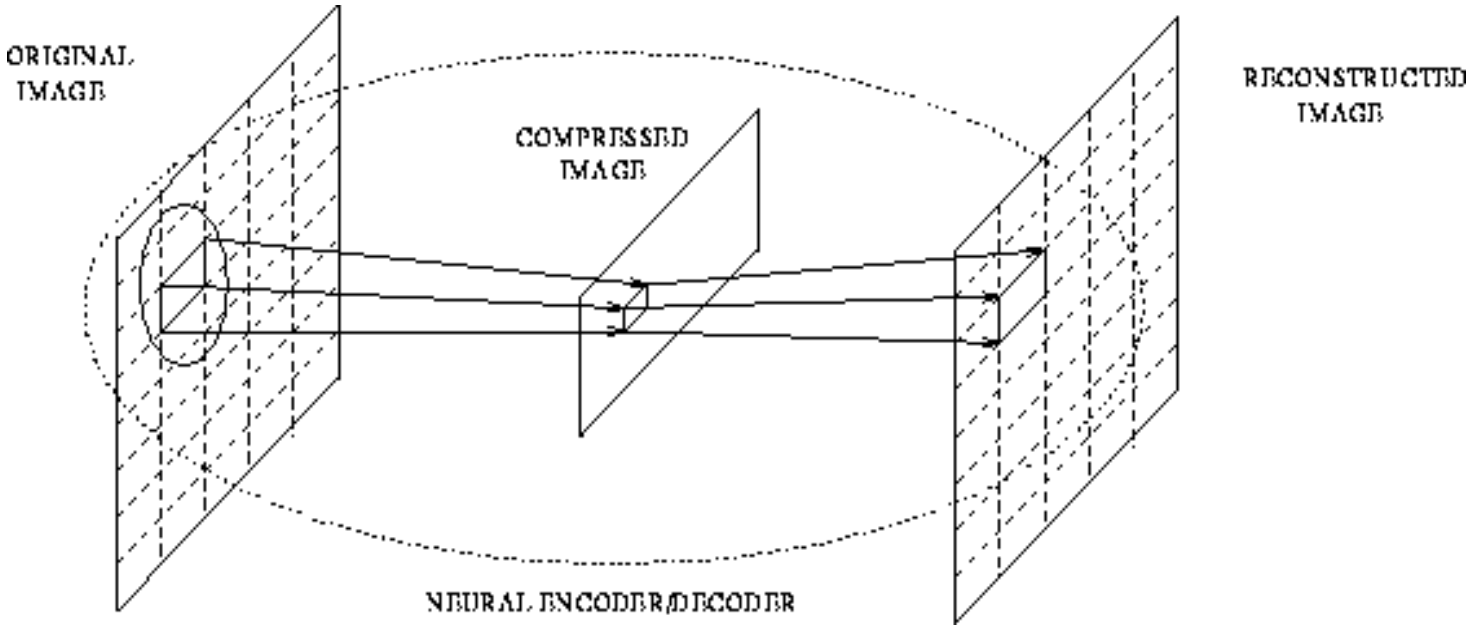
- Online Course, Sonar Signal Processing, PennState University
- Underwater Acoustics and Sonar Signal Processing , Institute of water acoustics, sonar engineering and signal theory
- High-speed processors for sonar

Problème nécessite une grande capacité de transmission

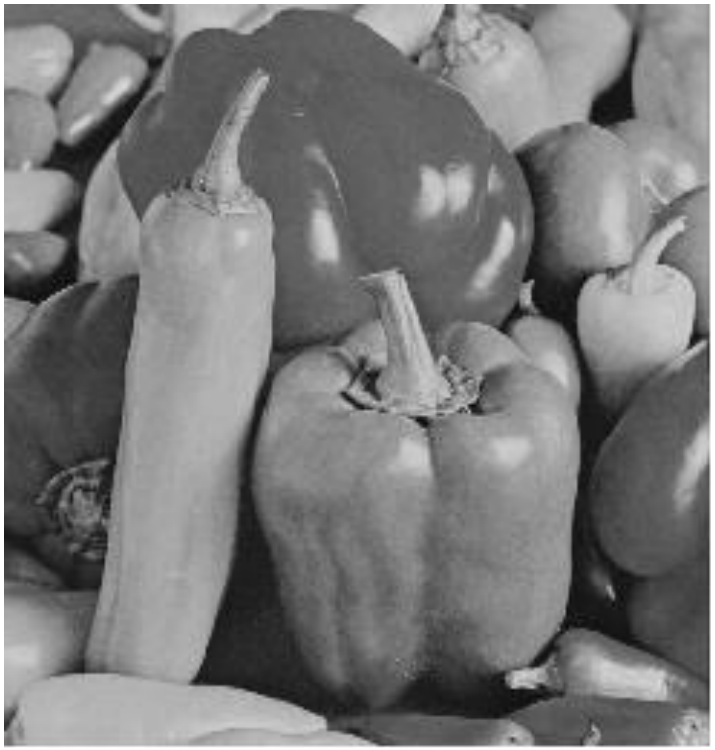
Pratique exploiter la redondance en codant l'image avec un nombre réduit de bits

Formulation codage par un problème d'apprentissage supervisé où l'entrée doit être la même que la sortie

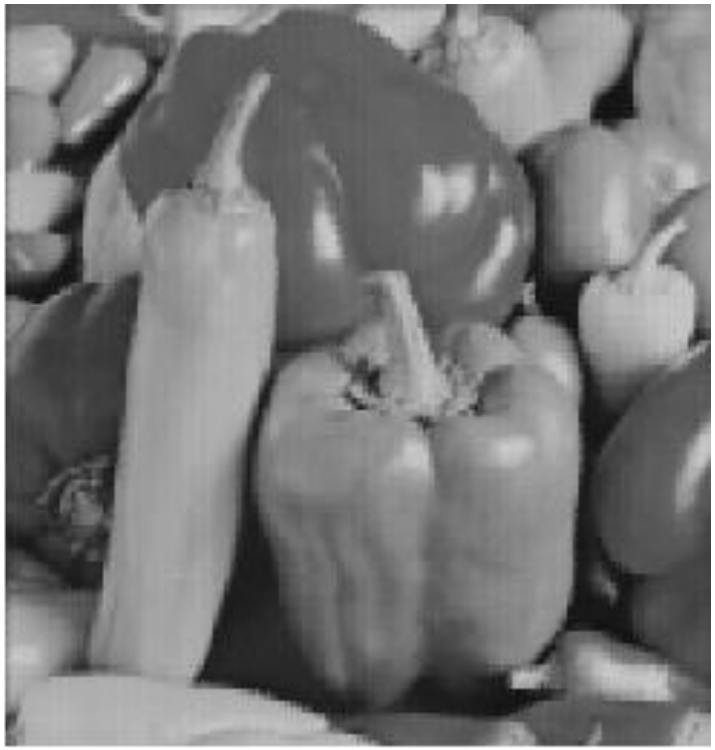

Décodage
Codage
Couche sortie
Couche cachée
Couche d'entrée



Department of Electrical Engineering,
Duke University, Durham, NC 27708-0291



PEPPERS original

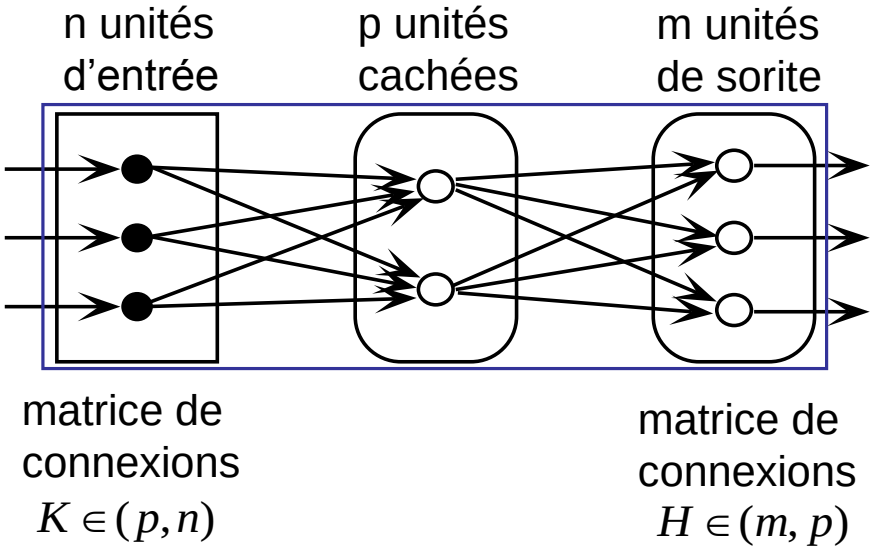


SNR = 27.82

Régression linéaire

Problématique

X : entrée $n \times N$
Y : sortie $m \times N$



Classification

Résolution en W du système **Y = WX**

Problématique

Il n'y a pas toujours de solution

Approche

Définir la fonction coût **f**

➔ problème d'optimisation qui minimise la fonction $f(Y - WX)$

Régression linéaire

Objectif problème d'optimisation qui minimise la fonction

f est une fonction quadratique $\mathbf{f(Y-WX)}$

$$Y = WX \quad \begin{cases} X \in (n, N) \\ Y \in (m, N) \end{cases}$$

Solution pseudo-inverse de Penrose :

$$W = Y \cdot X^t$$
$$= Y \cdot \left[X^t \left(X\, X^t \right)^{-1} \right]$$

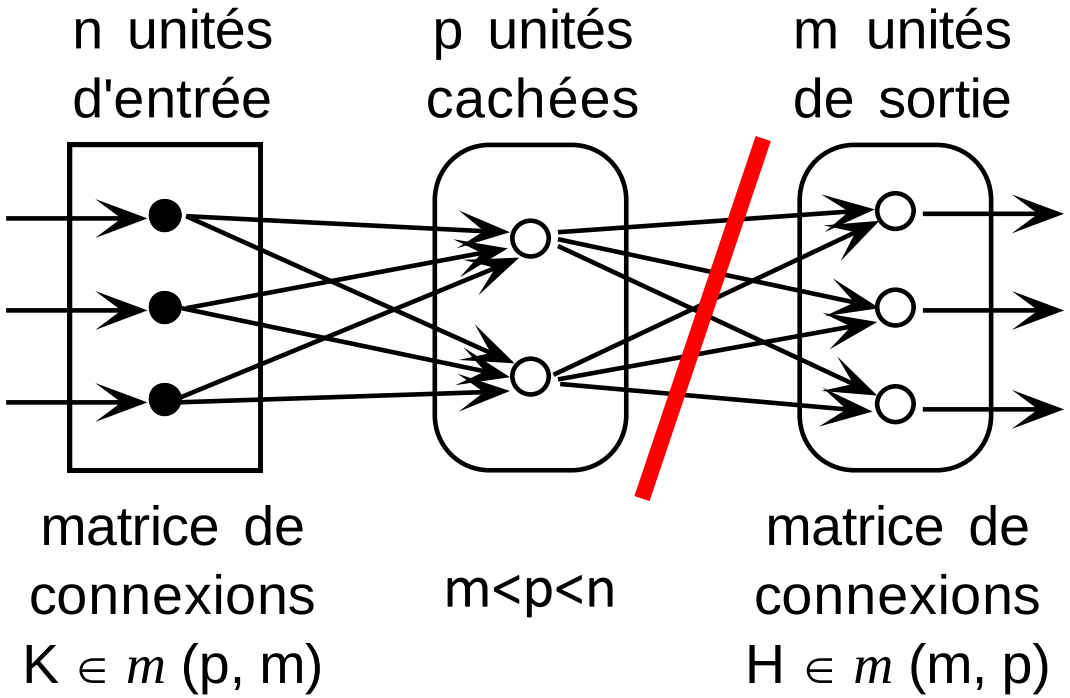
Analyse pratique solution non satisfaisante :

- problèmes de grande taille (inversion)
- prend en compte tous les exemples à la fois

Analyse discriminante

Objectif Trouver le meilleur sous-espaces **/**
la **projection** dans ce sous-espace **sépare** au mieux les vecteurs d'entrée

Approche



Retenir

Chaque unité de la couche cachée détecte une caractéristique contribuant à la classification

Point commun

Les résultats obtenus par la rétropropagation pourraient l'être par des méthodes plus traditionnelles d'analyse de données (analyse discriminante)

Avantage

la rétropropagation s'effectue de manière parallèle