Réseaux de Neurones:
- Modèles & Architectures
- Apprentissage

Modèles & Architecture

- Neurone formel de McCulloch & Pitts
- Modèle non-linéaire d'un neurone
- Fonctions d'activation
- Architectures de réseaux
- Représentation des connaissances
- Apprentissage: Perceptron
- Apprentissage: Rétro-propagation

Quelques notes historiques

références les plus importantes:

- J.J. Hopfield

  Neural Networks and physical systems with emergent collective computational abilities

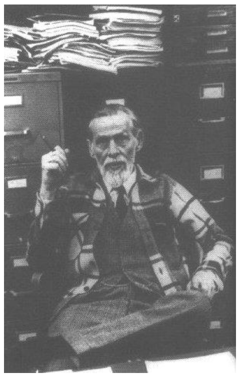
  Proc. Natl. Acad. of Sciences, vol. 79, 1554-1558, 1982.

- D.E. Rumelhart & J.L. McClelland

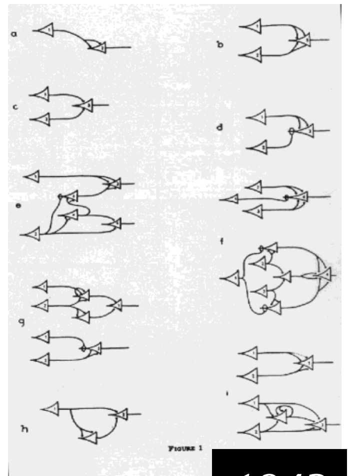
  Parallel Distributed Processing: Exploration in the Microstructure of Cognition

  MIT Press, Cambridge, 1986.

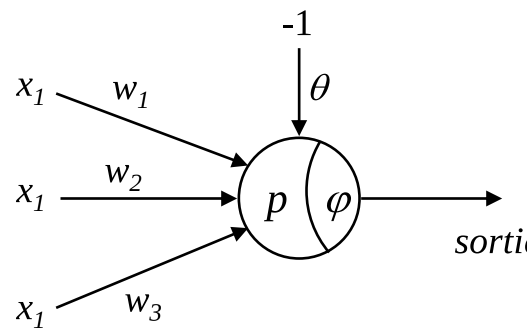
Neurone formel de McCulloch & Pitts

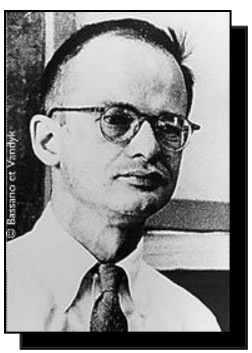


Warren S. McCulloch



1943



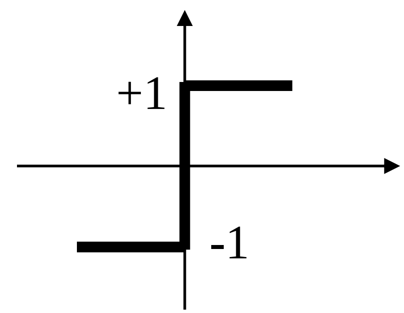


William Pitts

$$p = \sum_i w_i \, x_i$$

si $p > \theta$ then *sortie* = +1

else *sortie* = -1



fonction Signum

Interprétation géométrique

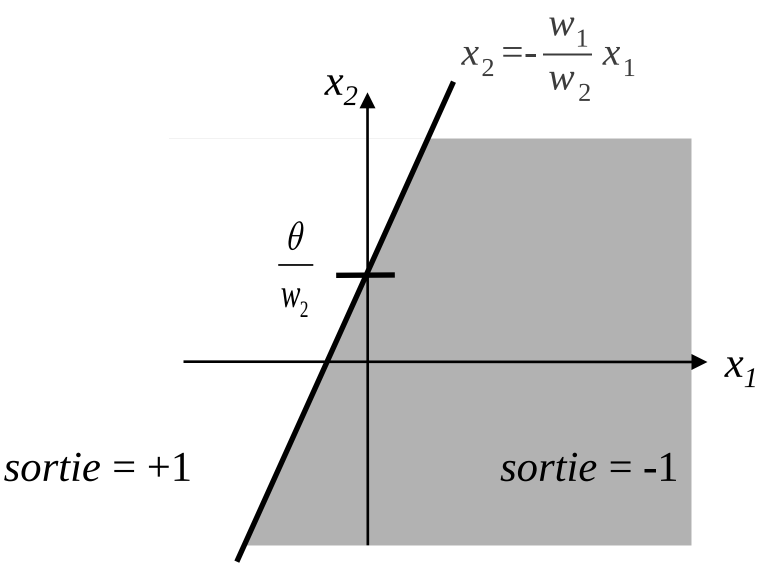
Soit un neurone formel de McCulloch&Pitts avec deux entrées x_1 et x_2

sa *sortie* est +1 si: $w_1 x_1 + w_2 x_2 > \theta$

sa *sortie* est -1 si: $w_1 x_1 + w_2 x_2 \leq \theta$

Géométriquement ces deux équations partitionnent le plan (x_1,x_2) par une droite définie par l'équation suivante:

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



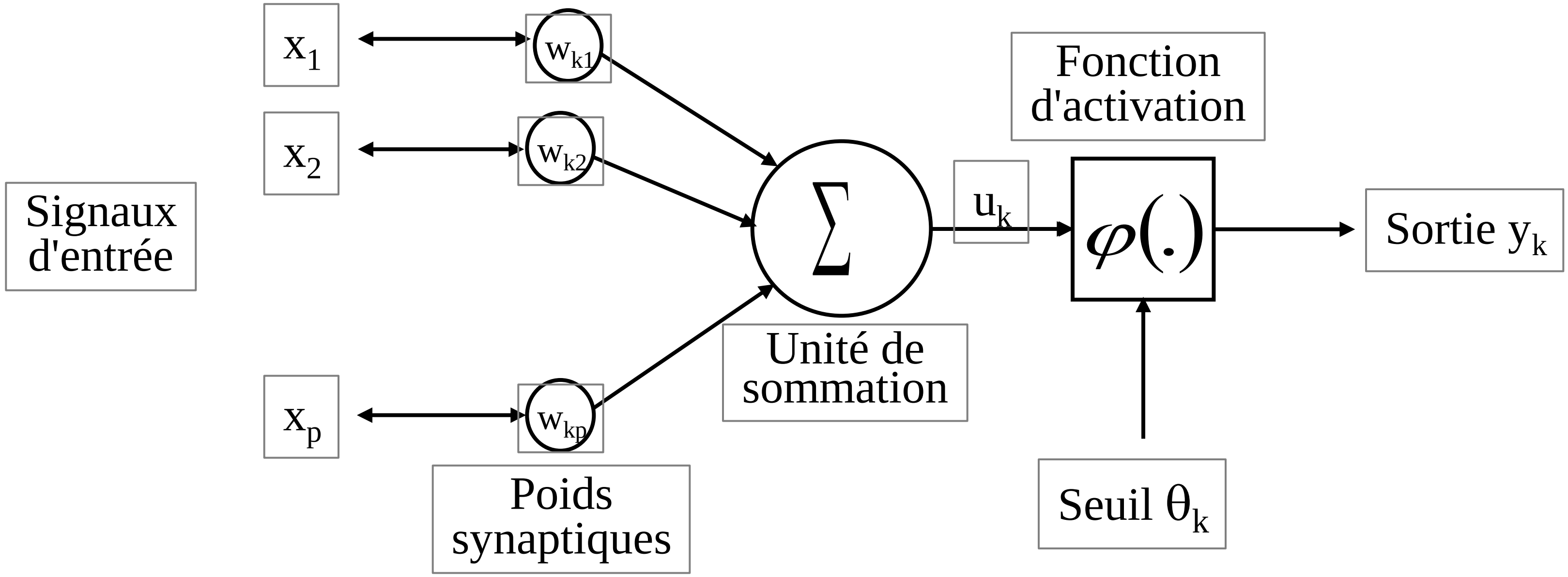
Modèle non-linéaire de neurone

3 éléments fondamentaux:

- Un ensemble de *synapses* caractérisées par un poids w_{kj}
  - $w_{kj} > 0 \Rightarrow$ synapse excitatrice,
  - $w_{kj} < 0 \Rightarrow$ synapse inhibitrice,
- un *additionneur* pour sommer les signaux d'entrée,
- une *fonction d'activation* pour limiter l'amplitude de la valeur de sortie.

Modèle non-linéaire de neurone (cont.)

(version moderne du modèle de the McCulloch & Pitts)



Modèle non-linéaire de neurone (cont.)

Ce modèle est décrit mathématiquement par:

(1) $$u_k = \sum_{j=1}^{p} w_{kj} x_j \quad \text{et} \quad y_k = \varphi(u_k - \theta_k)$$

où:

$x_1, x_2, \ldots, x_p$ sont les *entrées*,

$w_{k1}, w_{k2}, \ldots, w_{kp}$ sont les *poids synaptiques* du neurone k,

u_k est la sortie de *l'additionneur*,

θ_k est le *seuil*,

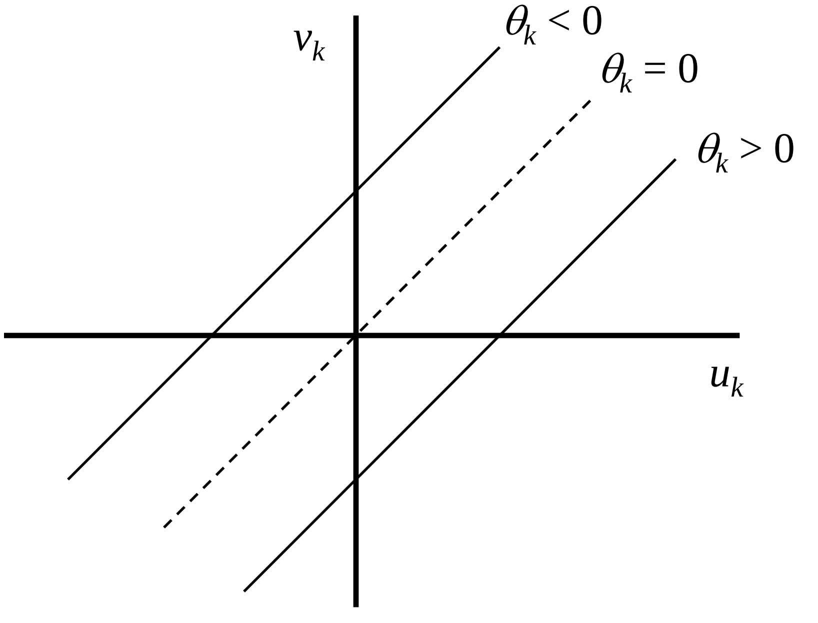
$\varphi(.)$ est la *fonction d'activation*,

y_k est la valeur de *sortie* du neurone.

Effets de la valeur de seuil

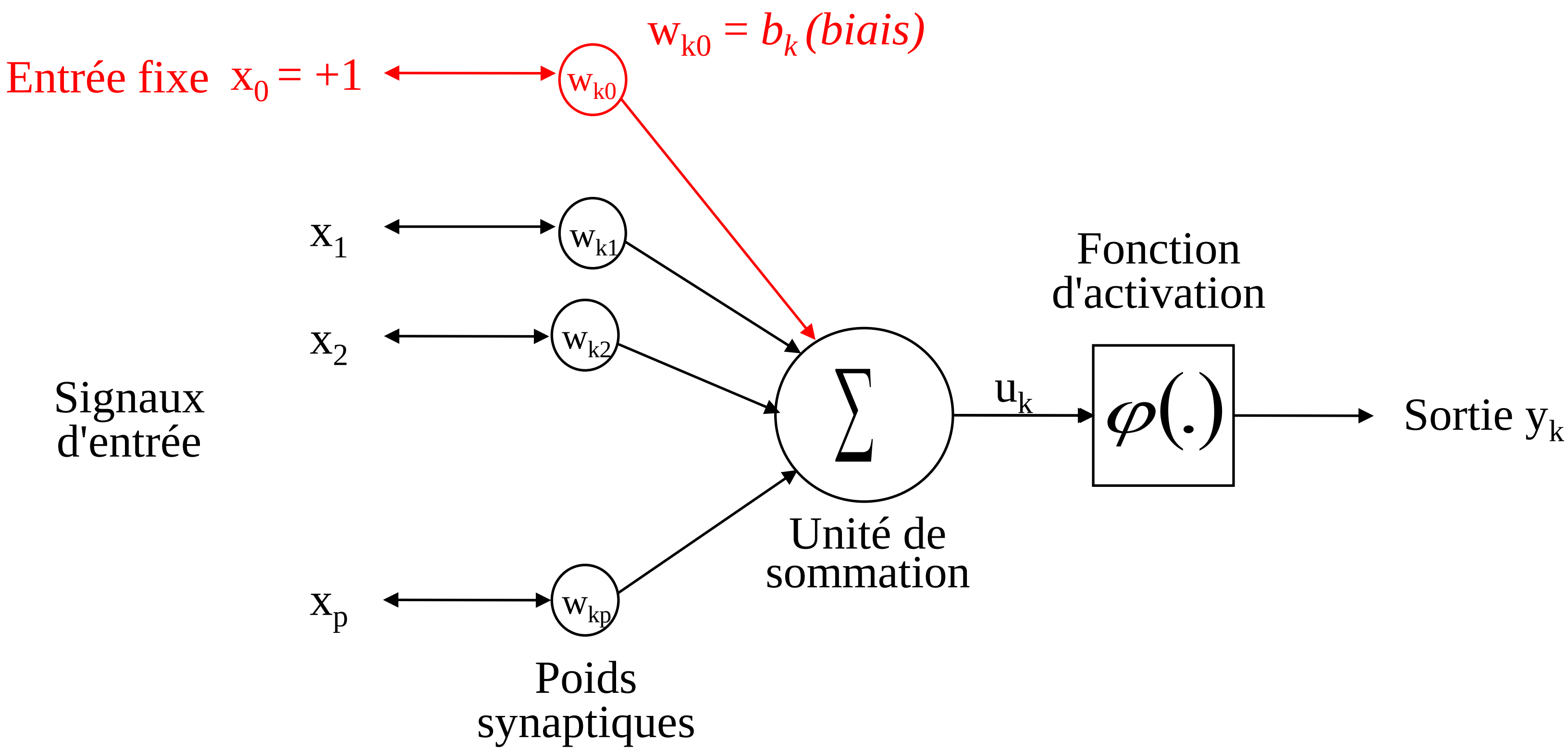
La valeur de seuil agit comme une *transformation affine* sur la valeur de sortie u_k :

v_k = potentiel d'activation du neurone k



$$v_k = u_k - \theta_k$$ ou en utilisant l'équ. (1) $$v_k = \sum_{j=1}^{p} w_{kj} x_j - \theta_k$$

Modèle étendu (alternative)



Types de fonctions d'activation

La fonction d'activation définit la valeur de sortie d'un neurone en fonction des valeurs de ses entrées.

3 types de fonctions d'activation:

- *Fonction à seuil*

$$y_k = \varphi(v_k) = \begin{cases} 1 & \text{si } v_k \geq 0 \\ 0 & \text{si } v_k < 0 \end{cases}$$

- *Fonction 3 marches*

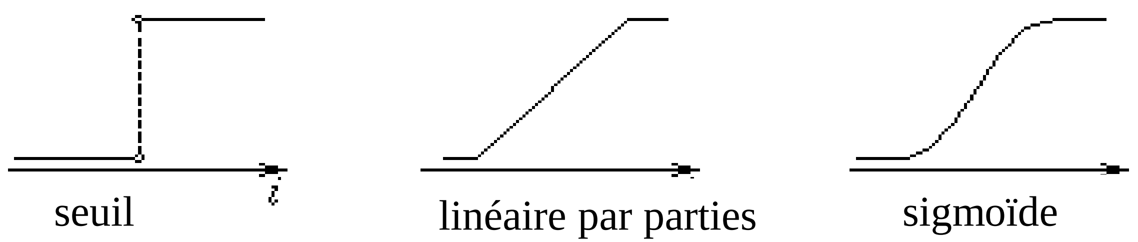
$$\varphi(v) = \begin{cases} 1 & v \geq \alpha \\ v & \alpha > v > \beta \\ 0 & v \leq \beta \end{cases}$$

Types de fonctions d'activation *(cont.)*

- *Fonction sigmoïde*

$$\varphi(v) = \frac{1}{1+e^{-av}} \qquad \text{ou} \qquad \varphi(v) = \tanh\left(\frac{v}{2}\right) = \frac{1-e^{-v}}{1+e^{-v}}$$

- *graphiquement*



Définition mathématique

Un réseau neuronal est un *graphe orienté* constitué de *nœuds* liés par des *liens synaptiques* et *d'activation* caractérisé par les propriétés suivantes:

- un neurone est représenté par:

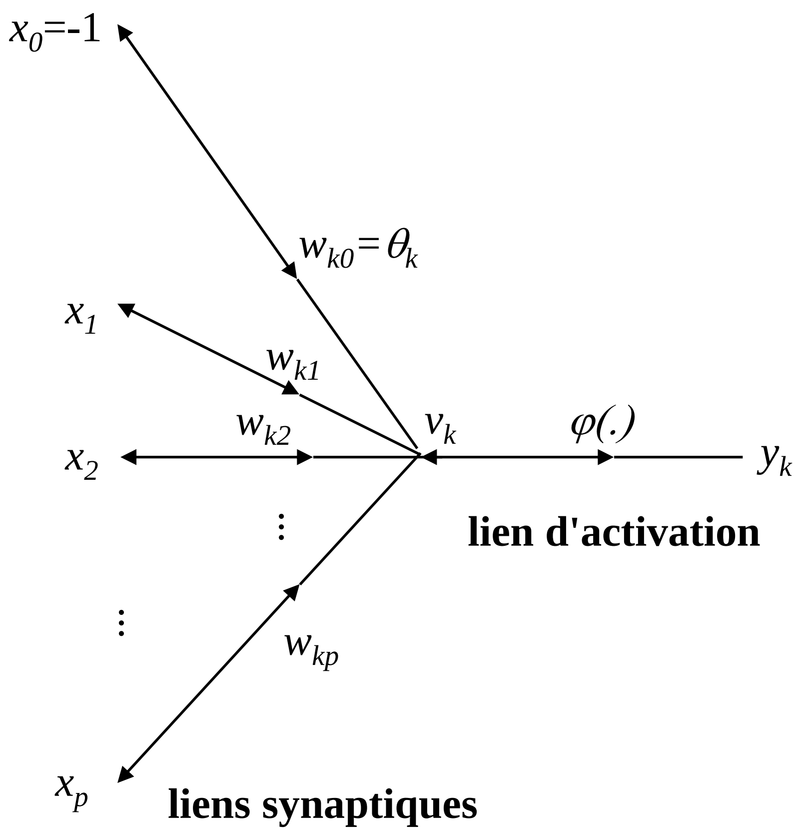
  un ensemble de liens synaptiques linéaires,

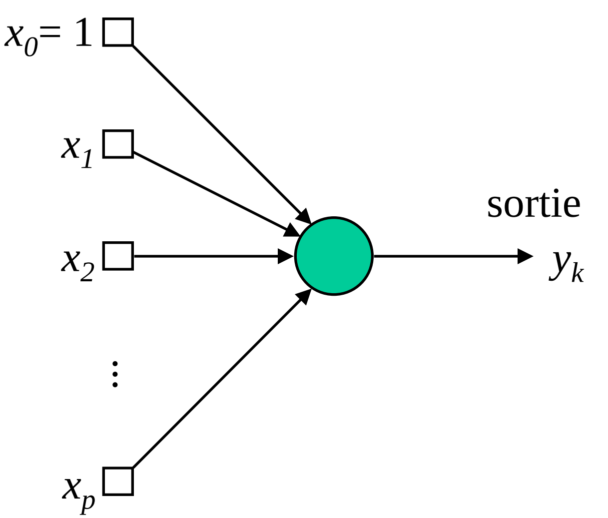
  un lien d'activation non-linéaire,

  un seuil

- les liens synaptiques pondèrent leurs signaux d'entrée,
- la somme pondérée des signaux d'entrée = niveau d'activité interne du neurone,
- le lien d'activation transforme le niveau d'activité interne en valeur de sortie (= variable d'état du neurone).

Diagrammes "signal-flow" & architecturaux





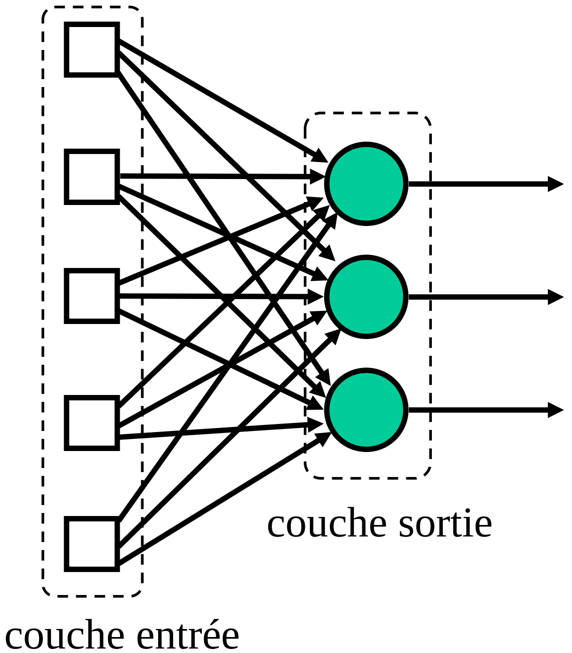
Graphe "signal-flow"

Graphe architectural

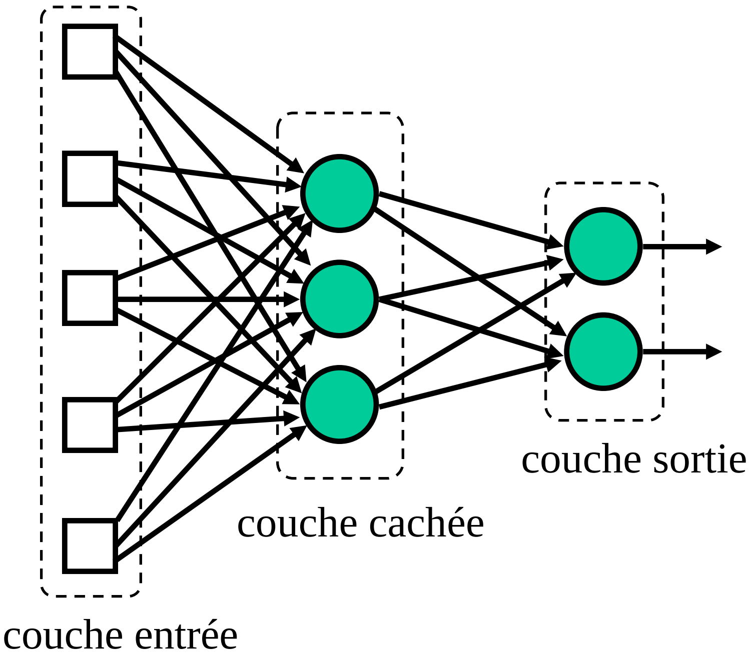
Architectures de réseaux

- Il y a 4 différentes classes d'***architectures***:
  - **réseaux monocouche "feedforward",**
    - une couche d'entrés de nœuds-source,
    - une couche de sortie,
    - "feedforward": entrée ⇒ sortie (pas vice versa)
  - **réseaux multicouche "feedforward",**
    - une couche d'entrés de nœuds-source,
    - une ou plusieures couches cachées,
    - une couche de sortie,
  - **réseaux récurrents,**
    - au moins une boucle de rétro-action,
  - **structures en treillis,**
    - neurones organisés en matrice.

Architectures "feed-forward"

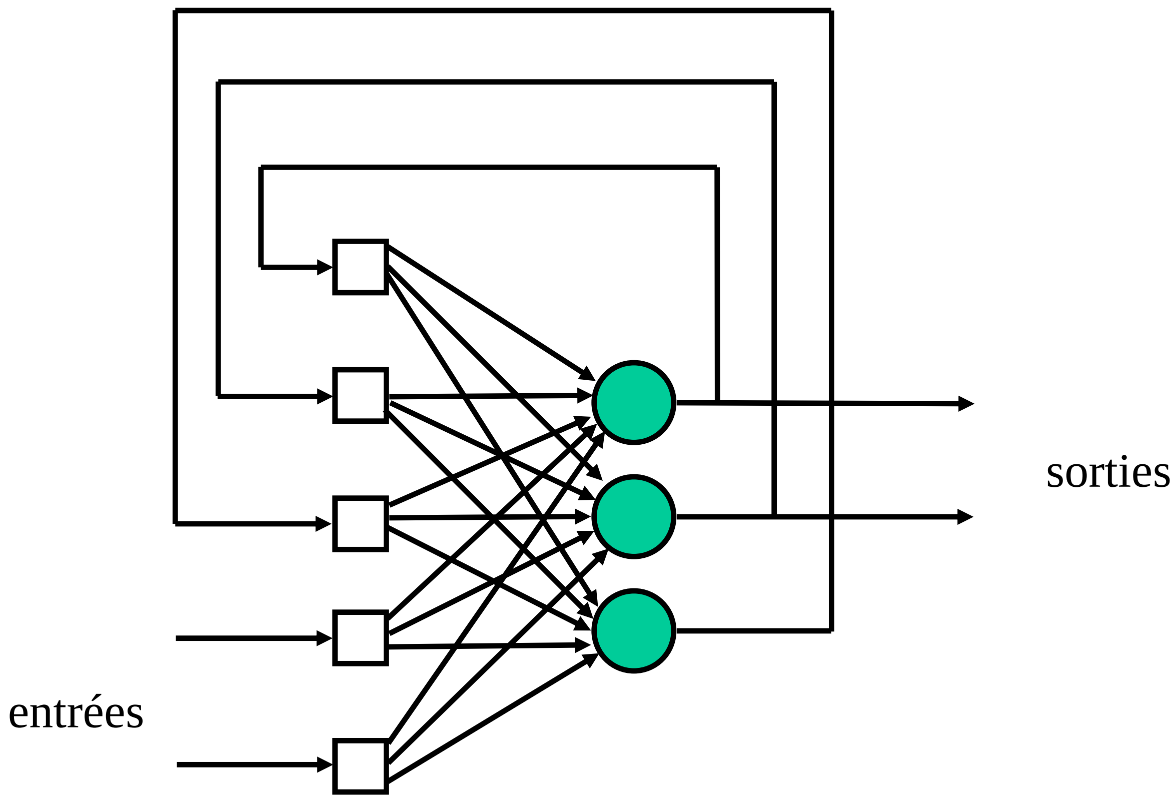


Réseau monocouche "feedforward"



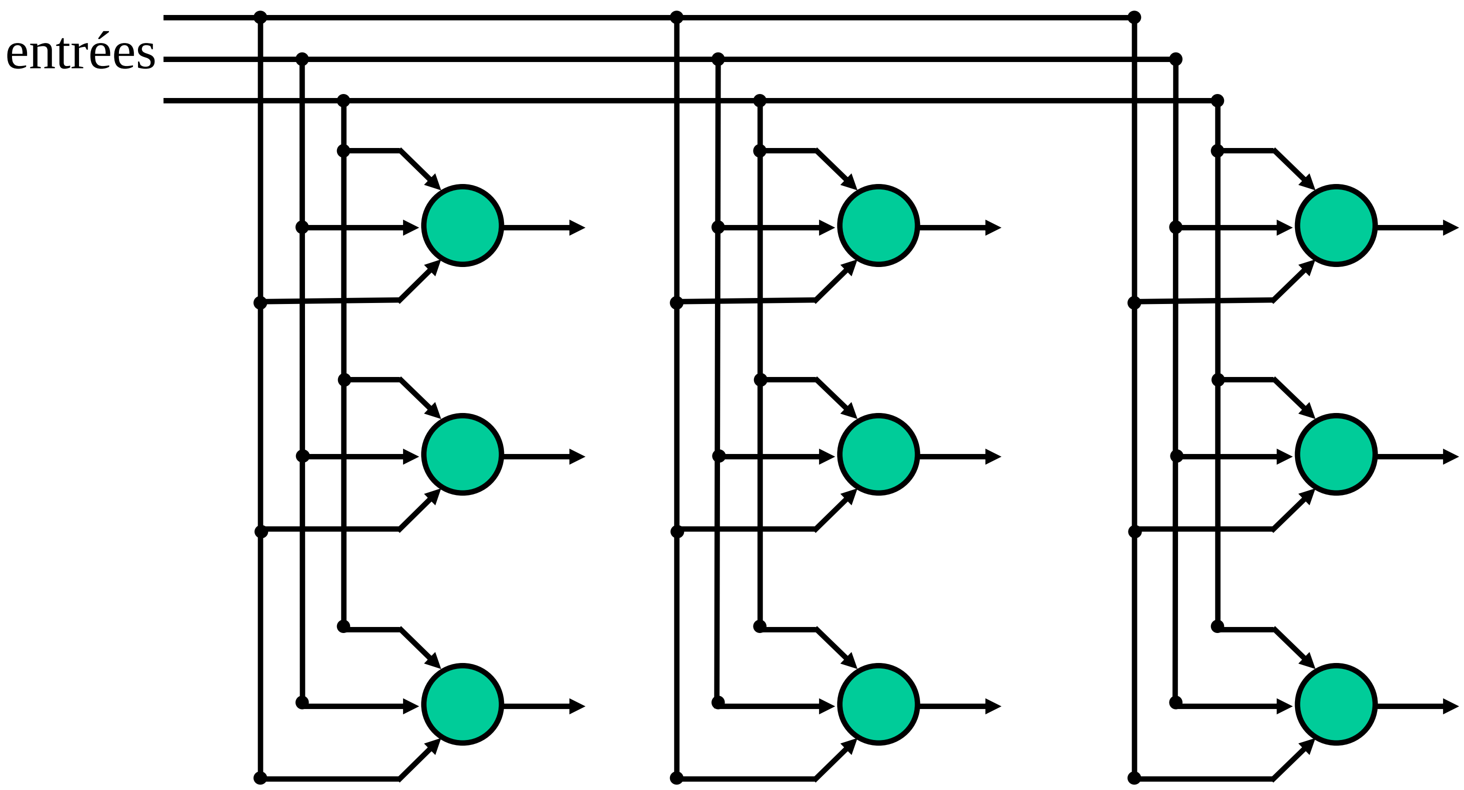
Réseau multicouche "feedforward"

Architecture récurrente



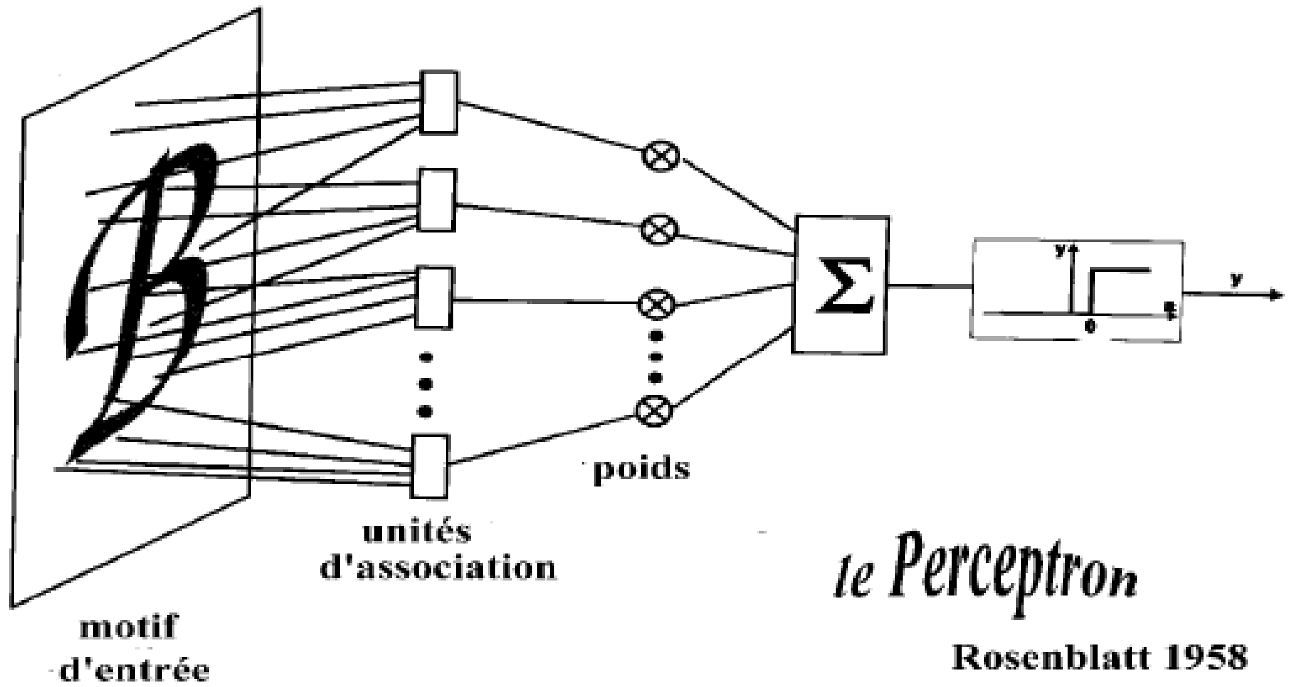
Réseau récurrent avec neurones cachés

Architecture en treillis



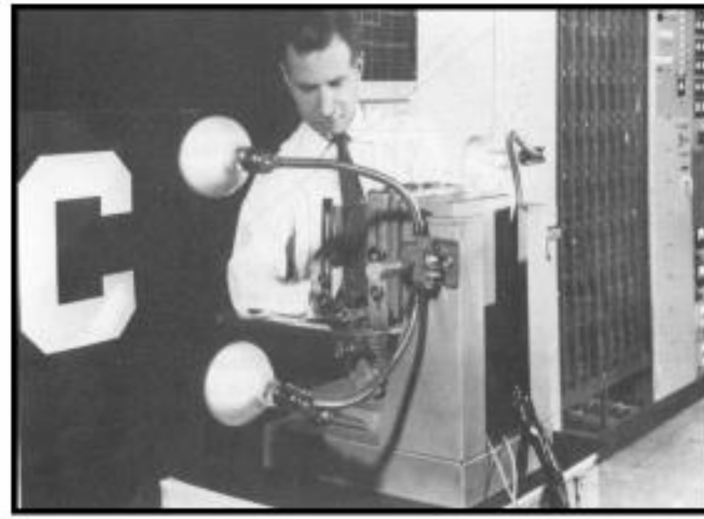
Réseau 2-D 3x3

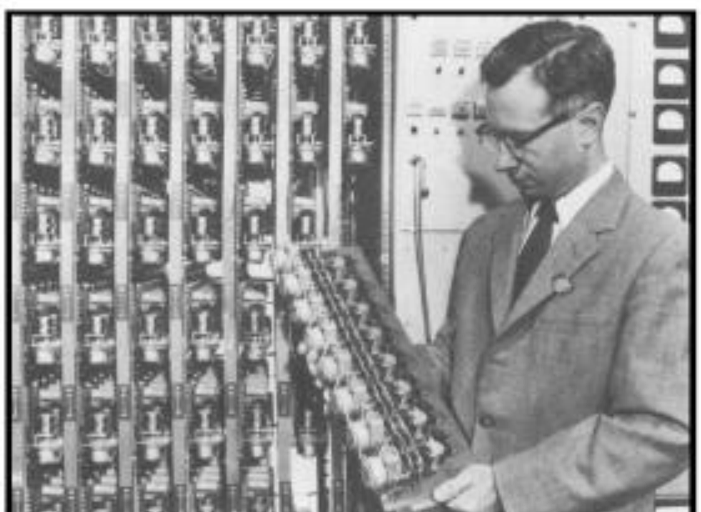
Le Perceptron (F. Rosenblatt, 1958)



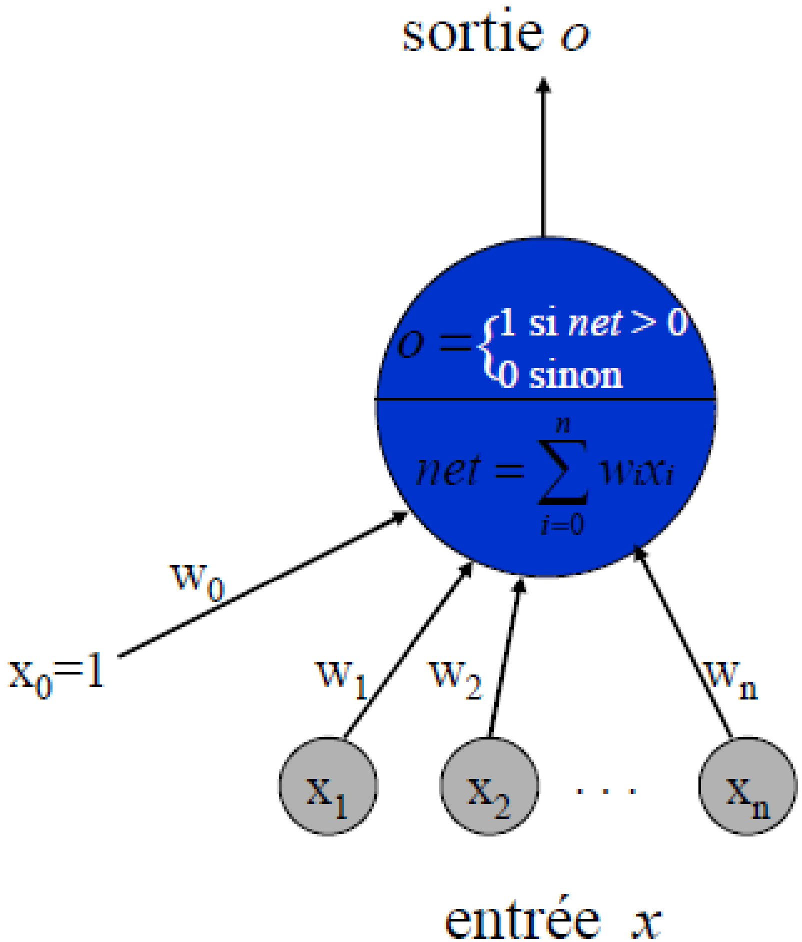
Le Mark I Perceptron

F. Rosenblatt



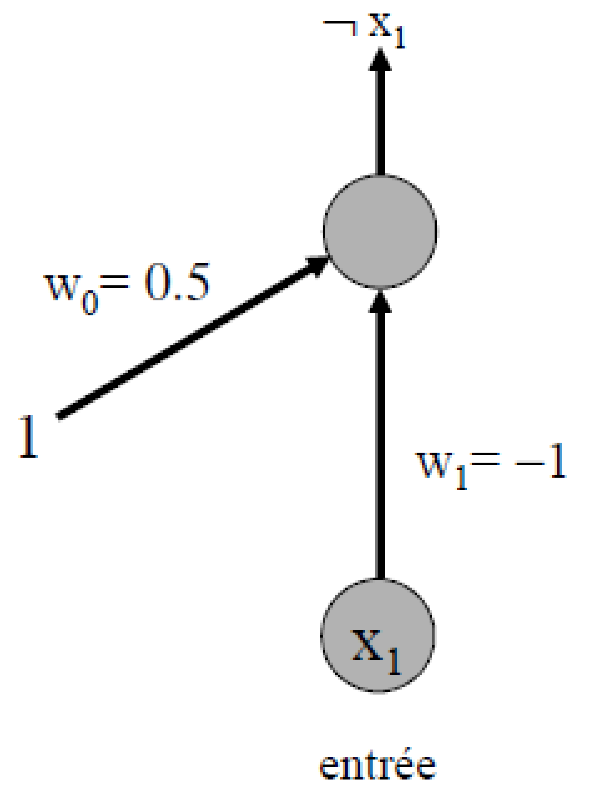


Le modèle "Perceptron"



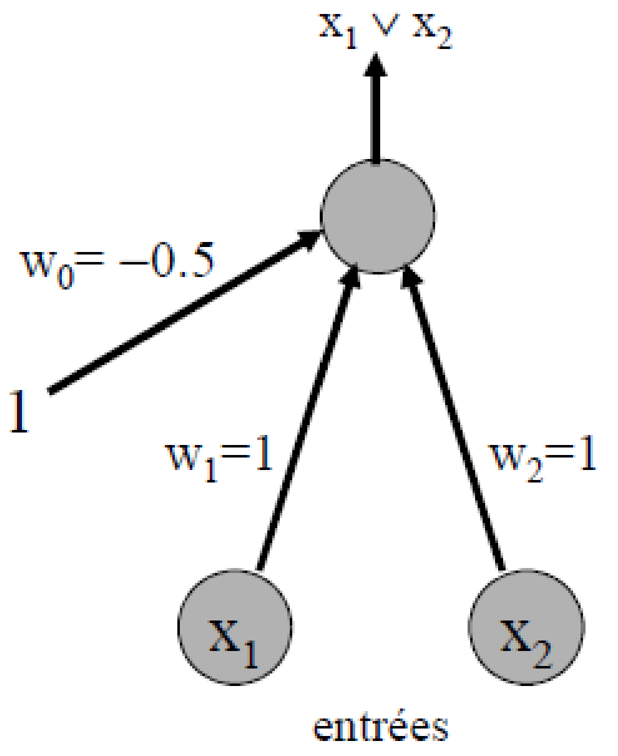
Exemple: négation logique

entrée x1	sortie
0	1
1	0



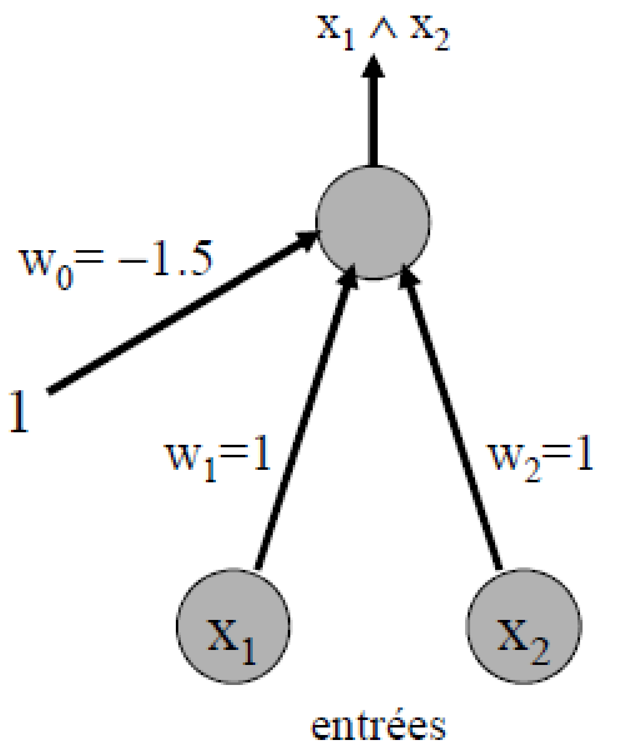
Exemple: "ou" logique

entrée x1	entrée x2	sortie
0	0	0
0	1	1
1	0	1
1	1	1



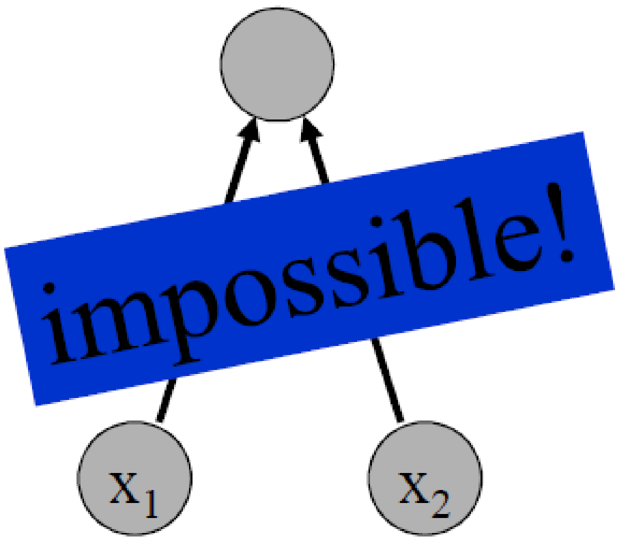
Exemple: "et" logique

entrée x1	entrée x2	sortie
0	0	0
0	1	0
1	0	0
1	1	1



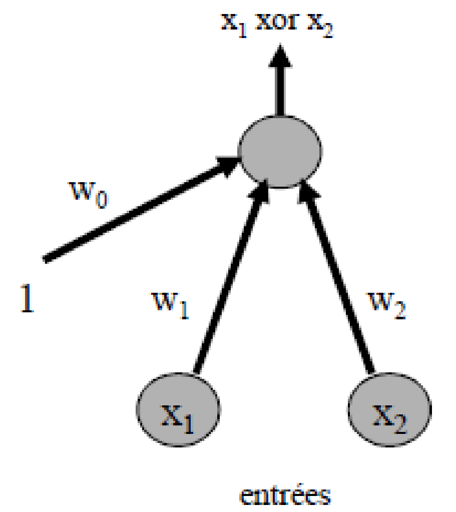
Exemple: "xor" logique (ou exclusif)

entrée x1	entrée x2	sortie
0	0	0
0	1	1
1	0	1
1	1	0



Pourquoi "impossible" ?

entrée x1	entrée x2	sortie
0	0	0
0	1	1
1	0	1
1	1	0



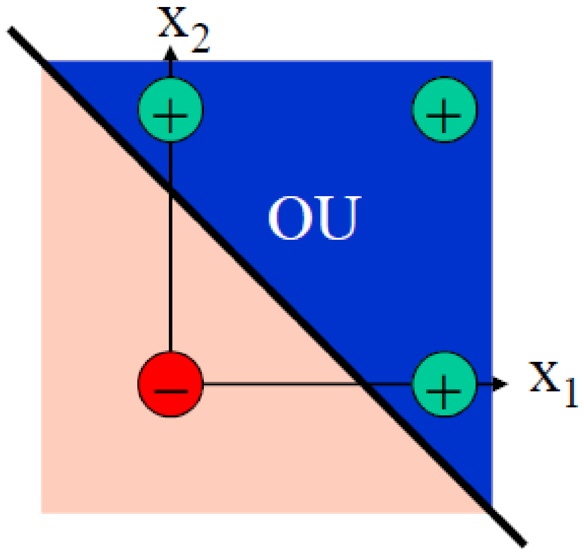
Le système d'équations à résoudre est:

$$w_0 + 0.w_1 + 0.w_2 \leq 0$$
$$w_0 + 0.w_1 + 1.w_2 > 0$$
$$w_0 + 1.w_1 + 0.w_2 > 0$$
$$w_0 + 1.w_1 + 1.w_2 \leq 0$$

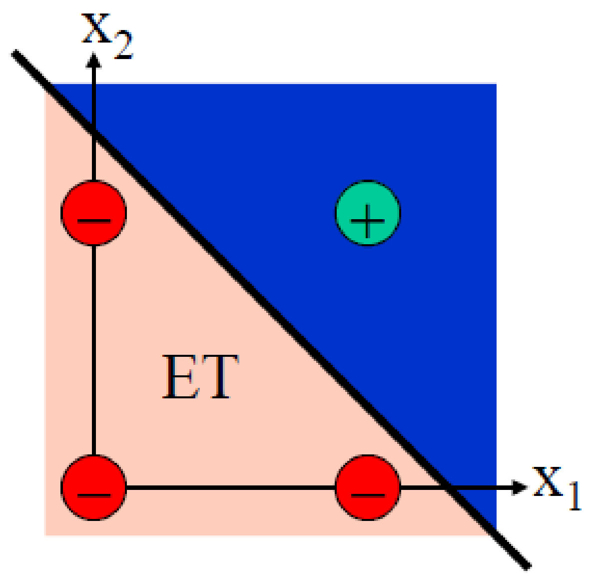
Il n'y a aucune valeur possible pour les coefficients w_0, w_1 et w_2 qui satisfasse les inégalités ci-contre.

xor ne peut donc pas être représenté!

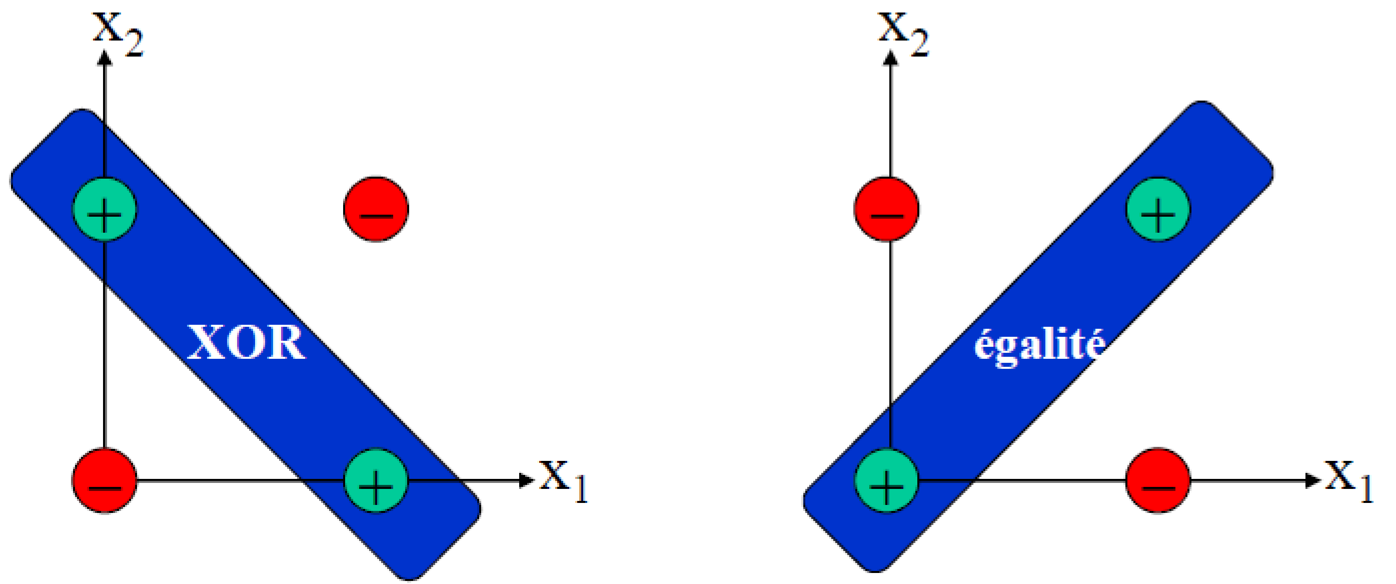
Séparabilité linéaire



Séparabilité linéaire

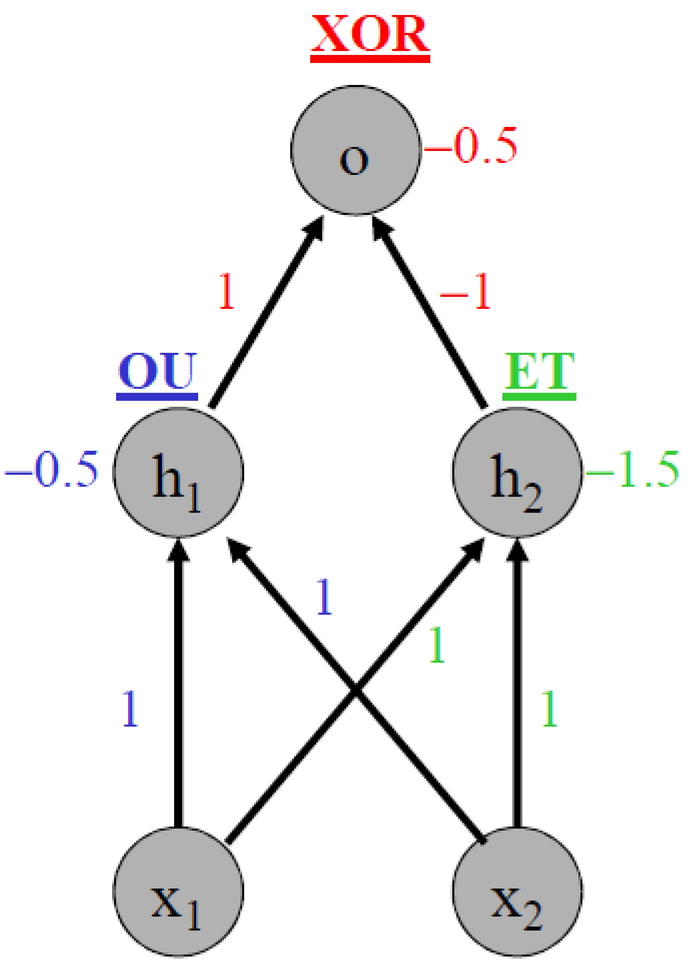


Non-séparabilité linéaire



Exemple: "xor" logique (revu)

entrée x1	entrée x2	sortie
0	0	0
0	1	1
1	0	1
1	1	0



Théorèmes du Perceptron

Théorème de représentation

Un réseau "feedforward" à une seule couche (Perceptron) peut uniquement représenter des fonctions linéairement séparables. C'est-à-dire celles pour lesquelles la surface de décision séparant les cas positifs des cas négatifs est un (hyper-)plan.

Théorème d'apprentissage (F. Rosenblatt)

Étant donné suffisamment d'exemples d'apprentissage, il existe un algorithme qui apprendra n'importe quelle fonction linéairement séparable.

Algorithme d'apprentissage du Perceptron

<u>Entrées</u>: ensemble d'apprentissage $\{(x_1, x_2, \ldots, x_n, t)\}$

<u>Méthode</u>

initialiser aléatoirement les poids w(i), 0<=i<=n

répéter jusqu'à convergence:

pour chaque exemple

calculer la valeur de sortie *o* du réseau.

ajuster les poids:

$$\Delta w_i = \eta\,(t - o)\,x_i$$

$$w_i \leftarrow w_i + \Delta w_i$$

Règle d'apprentissage du *Perceptron*

Algorithme d'apprentissage du Perceptron (résumé)

nouveau poids   ancien poids   modification

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta (t - o) x_i$$

modification   taux d'apprentissage   valeur attendue   valeur de sortie du Perceptron   entrée

Erreur quadratique

- La règle d'apprentissage du *Perceptron* effectue une descente de gradient dans l'espace des poids.

- Considérons une unité linéaire simple pour laquelle

$$o = w_0 + w_1 x_1 + \ldots + w_n x_n$$

- définissons l'erreur comme:

$$E[w_0, w_1, \ldots, w_n] = \frac{1}{2} \sum_{e \in Exemples} (t_e - o_e)^2$$

(erreur quadratique)

Convergence

La convergence est garantie car l'erreur E est une forme quadratique dans l'espace des poids. Elle possède donc un seul minimum global et la descente du gradient assure de le trouver.

Convergence si:

… les données d'apprentissage sont linéairement séparables
… le taux d'apprentissage η est suffisamment petit
… il n'y a pas d'unités "cachées" (une seule couche)