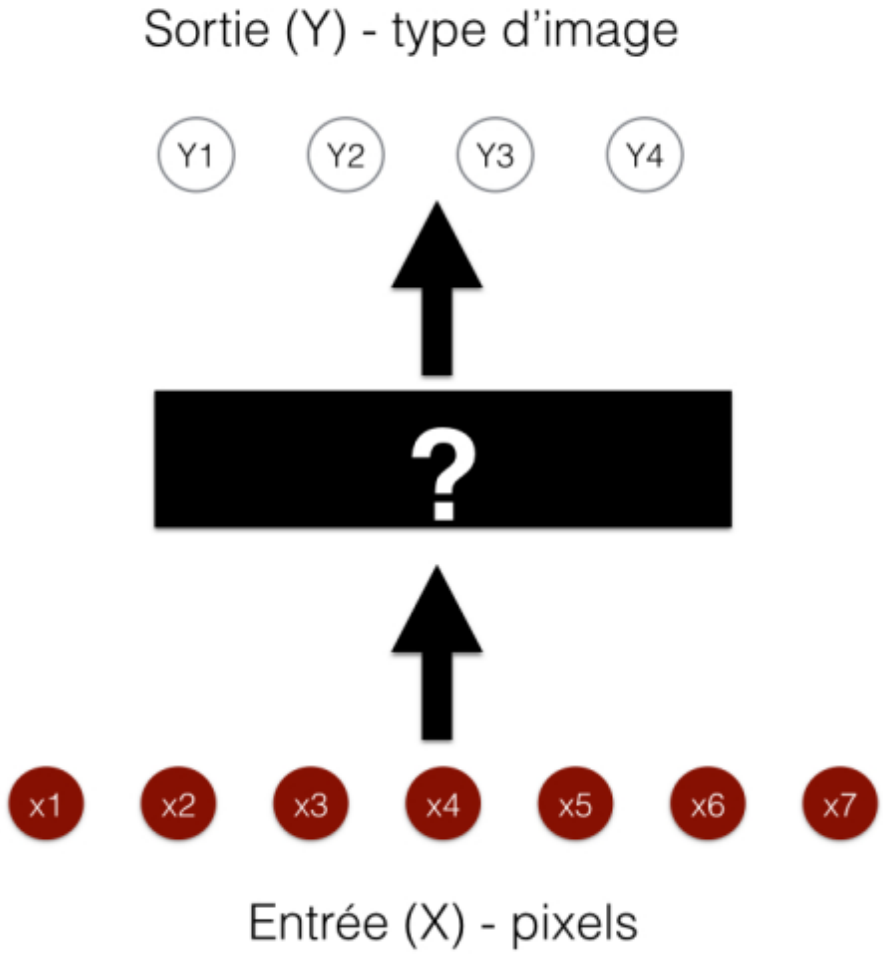
APPRENTISSAGE SUPERVISÉ : PERCEPTRONS

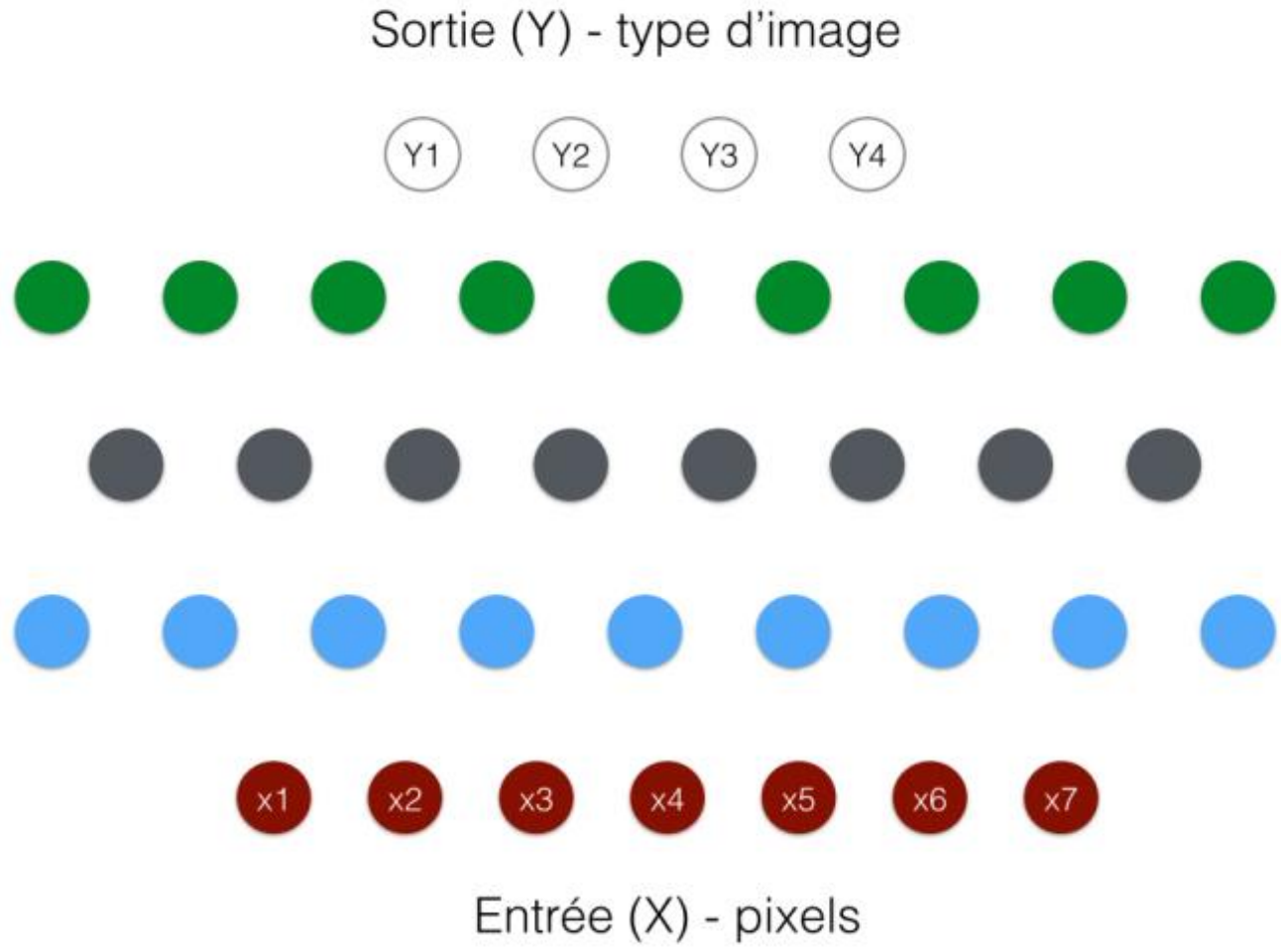
HISTOIRE

- Premières idées dans les années 50 : basé sur le fait de "copier" le fonctionnement des neurones du cerveau.
- Modèle classique : **perceptron multi-couches** introduit en 1986.
- **Théorème (Cybenko, 1989)** : Toute fonction continue bornée est estimable, avec une précision arbitraire, par un réseau avec une couche intermédiaire.
- Aujourd'hui : extension au **deep learning** avec en particulier les réseaux convolutifs pour les images.
- **Black box** par excellence : il est très compliqué de les interpréter et de les metre au point.

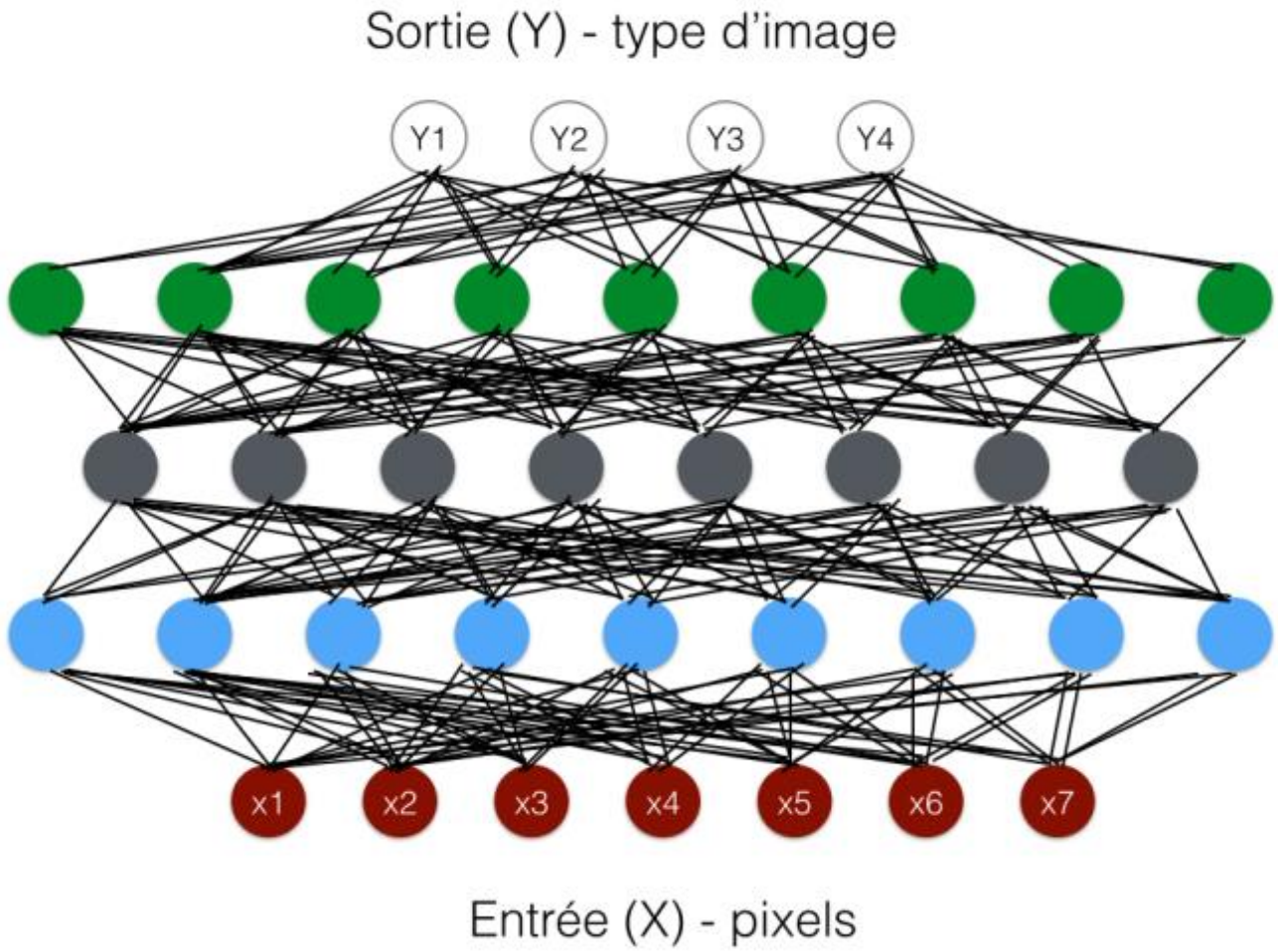
PRINCIPE


Sortie (Y) - type d'image
Y1
Y2
Y3
Y4
?
x1
x2
x3
x4
x5
x6
x7
Entrée (X) - pixels

PRINCIPE


Sortie (Y) - type d'image
Y1
Y2
Y3
Y4
x1
x2
x3
x4
x5
x6
x7
Entrée (X) - pixels

PRINCIPE


Sortie (Y) - type d'image
Y1
Y2
Y3
Y4
x1
x2
x3
x4
x5
x6
x7
Entrée (X) - pixels

PRINCIPE

Le réseau est composé:

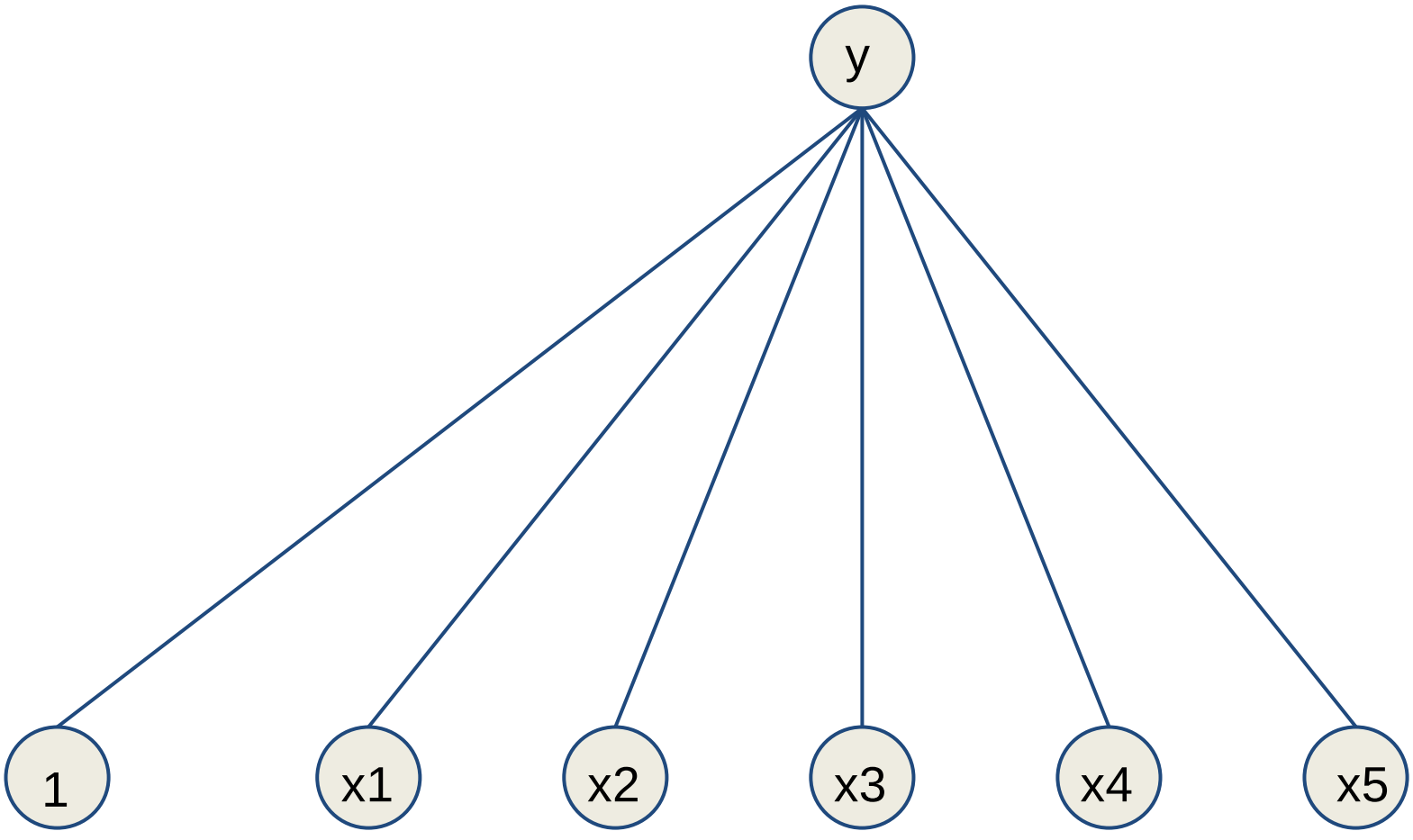
- d'une couche d'entrée (par exemple, des pixels)
- d'une couche de sortie (ce qu'on veut prédire)
- de couches intermédiaires appelées couches cachées

A partir de la première couche cachée, chaque "neurone" est une fonction de tous les neurones de la couche précédente.

Les neurones de la couche de sortie qui donnent les résultats sont donc une fonction (de fonctions de fonctions de fonctions…) des neurones de la couche d'entrée.

EXEMPLE

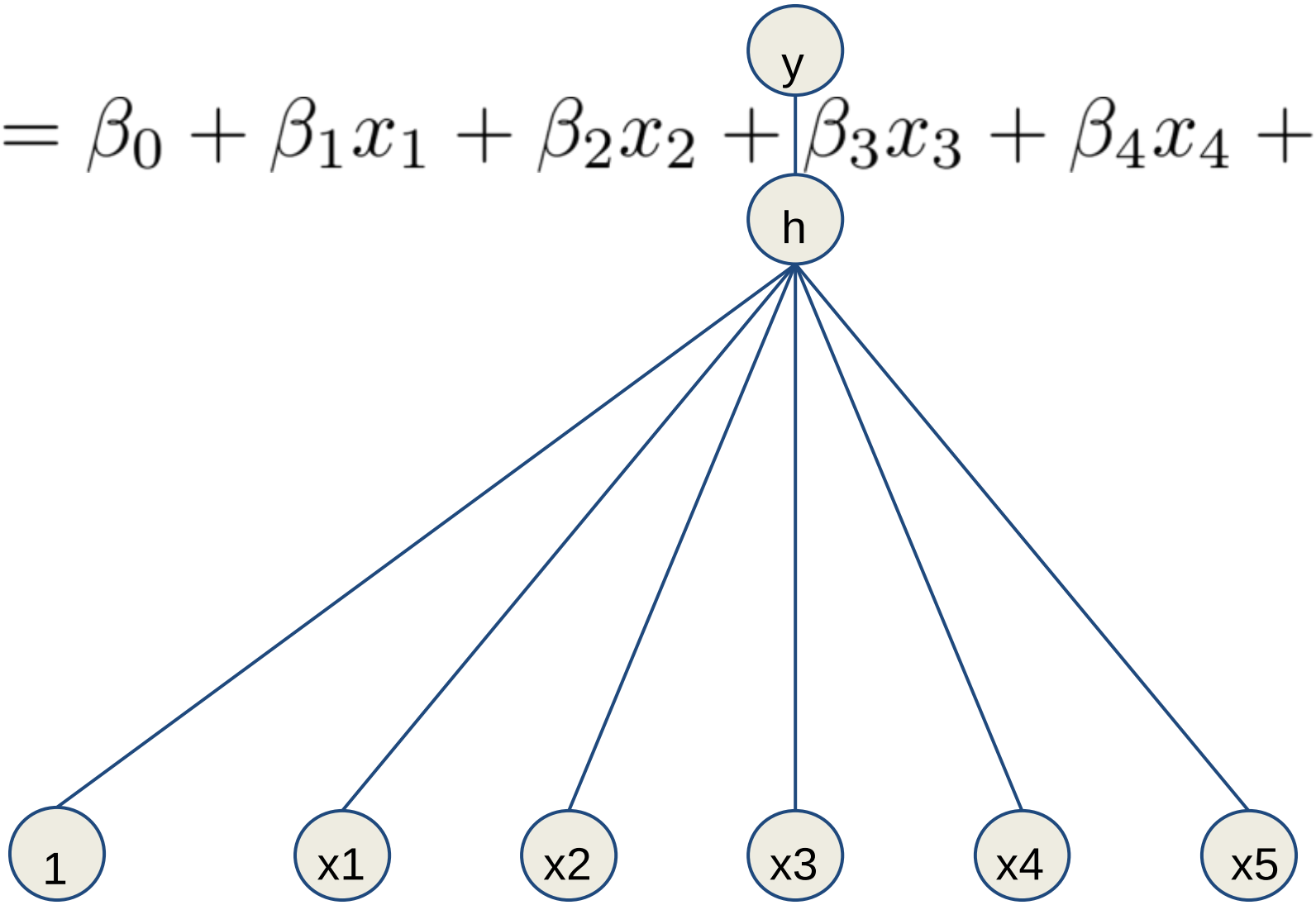
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5$$



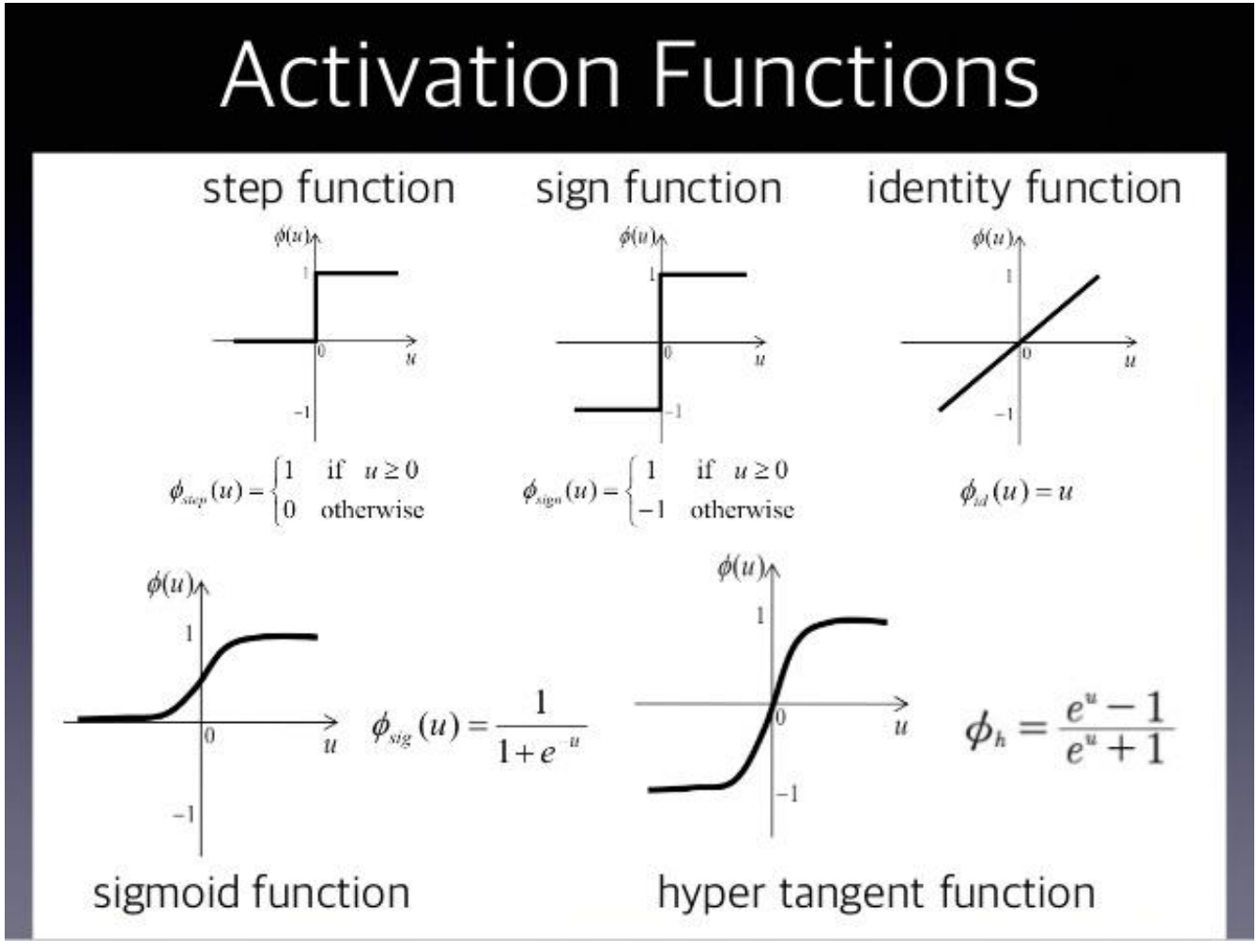
EXEMPLE

$$y = \phi(h)$$

$$h = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5$$



FONCTIONS D'ACTIVATION

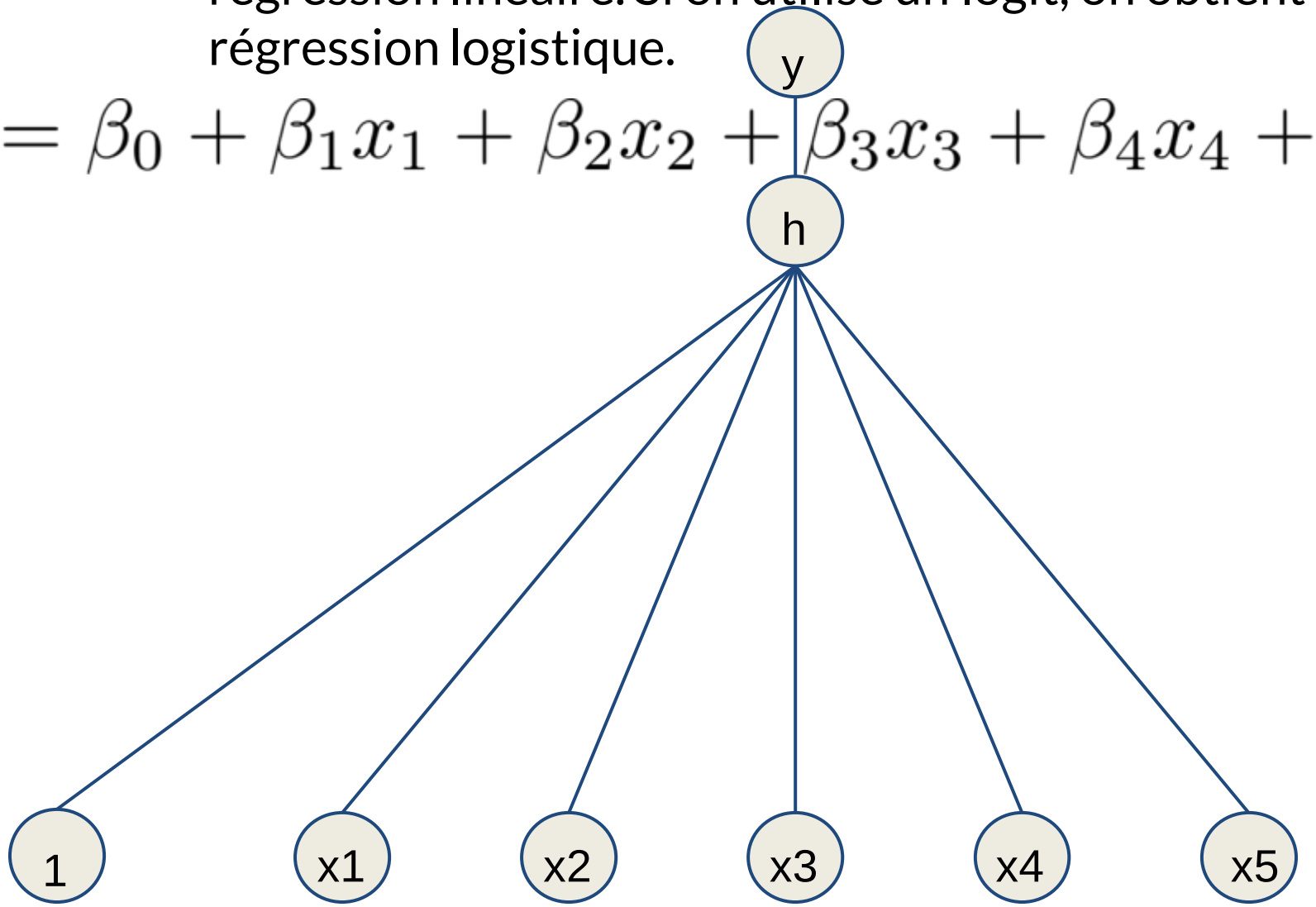


Source: [slideshare, présentation de Sung-ju Kim]

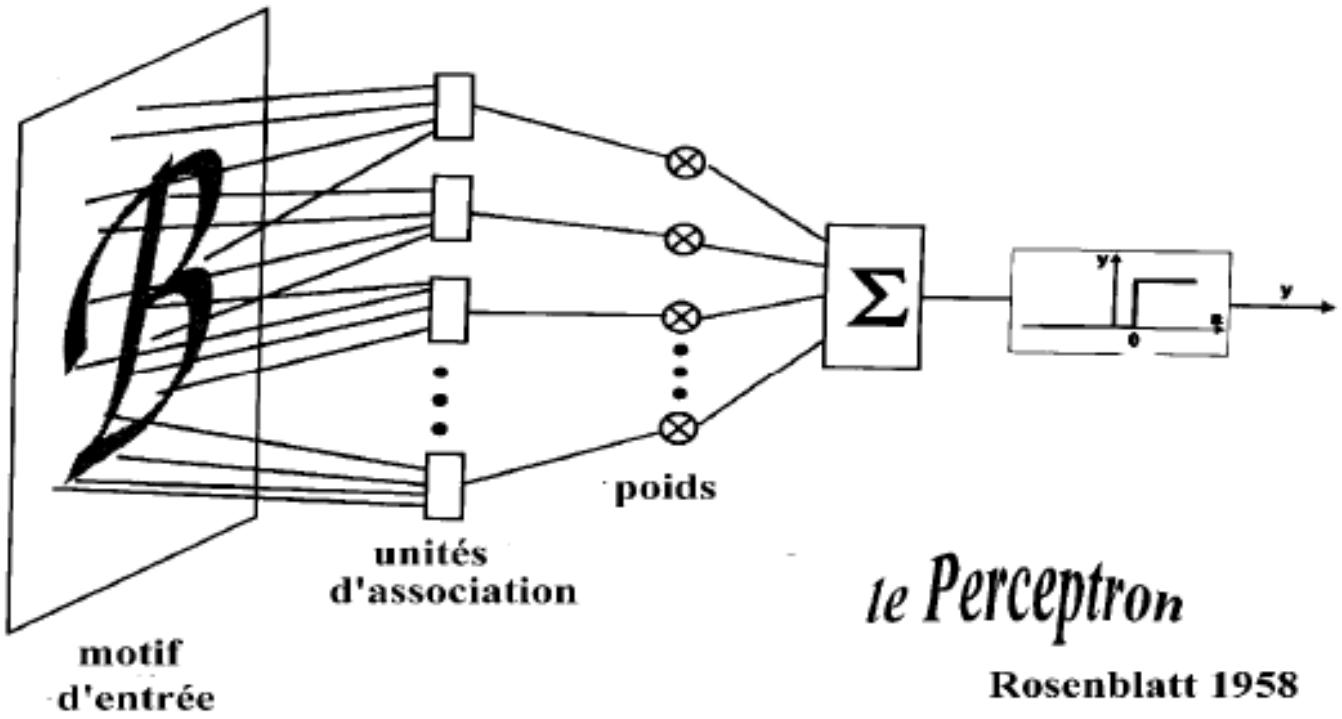
EXEMPLE

$y = \phi(h)$ Si on utilise l'activation linéaire, on obtient une régression linéaire. Si on utilise un logit, on obtient une régression logistique.

$$h = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5$$

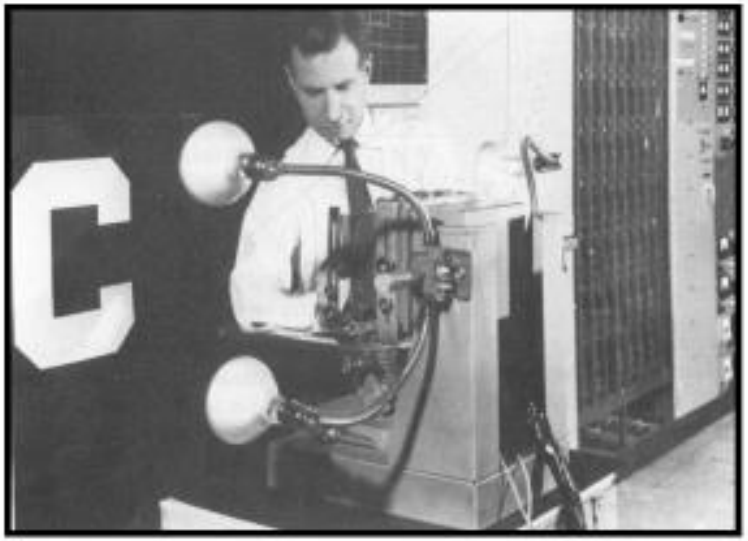


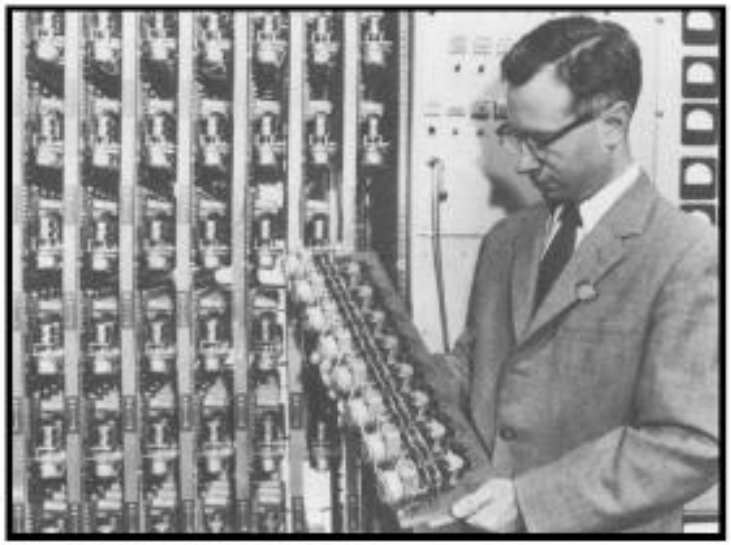
Le Perceptron (F. Rosenblatt, 1958)

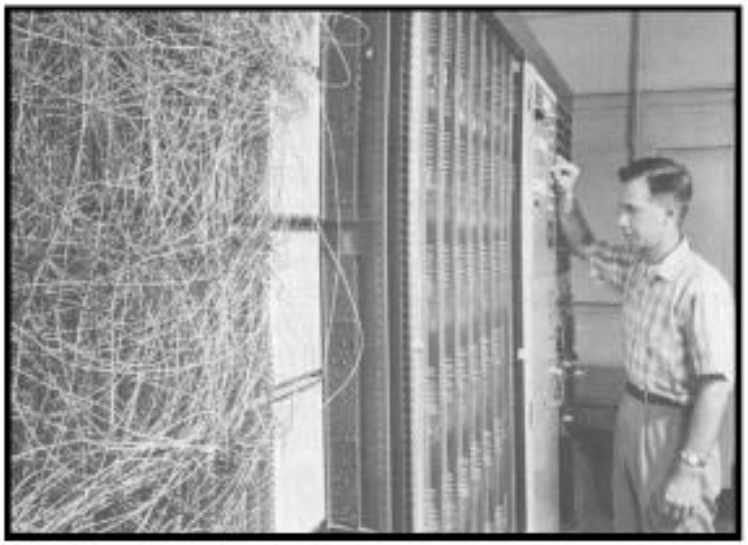


Le Mark I Perceptron

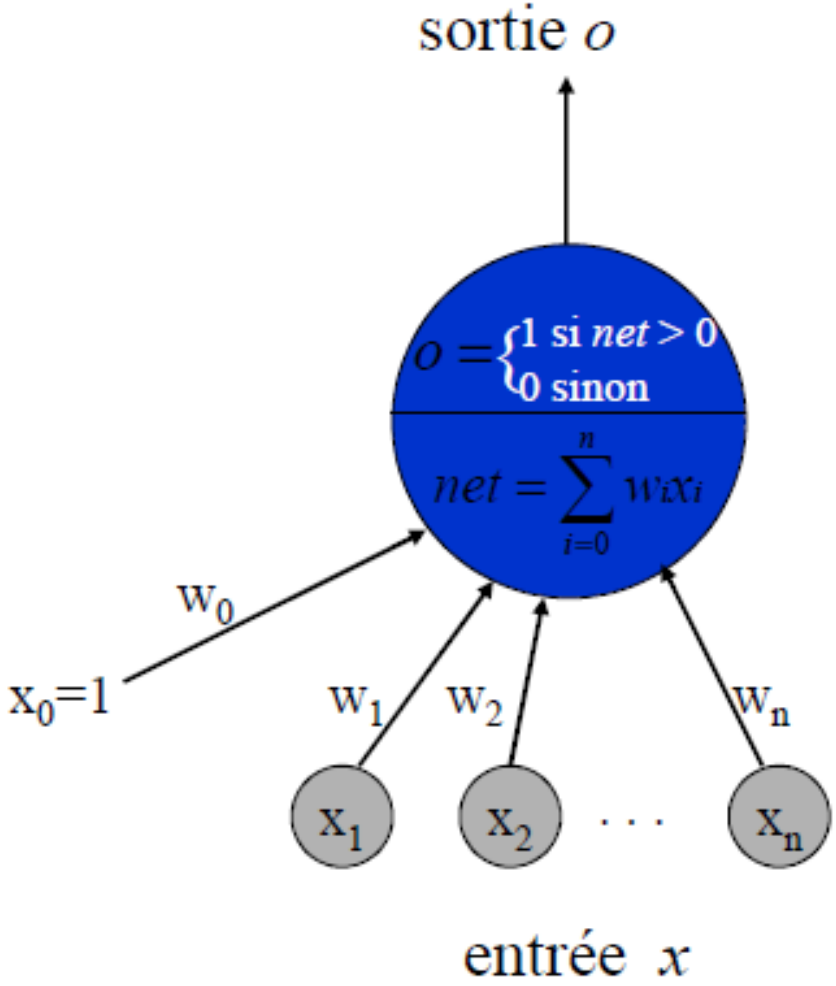
F. Rosenblatt





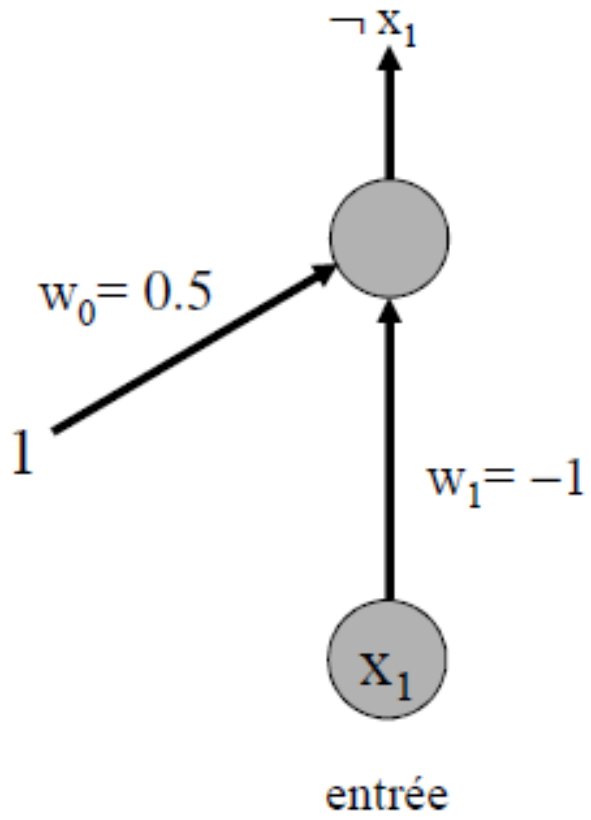


Le modèle "Perceptron"



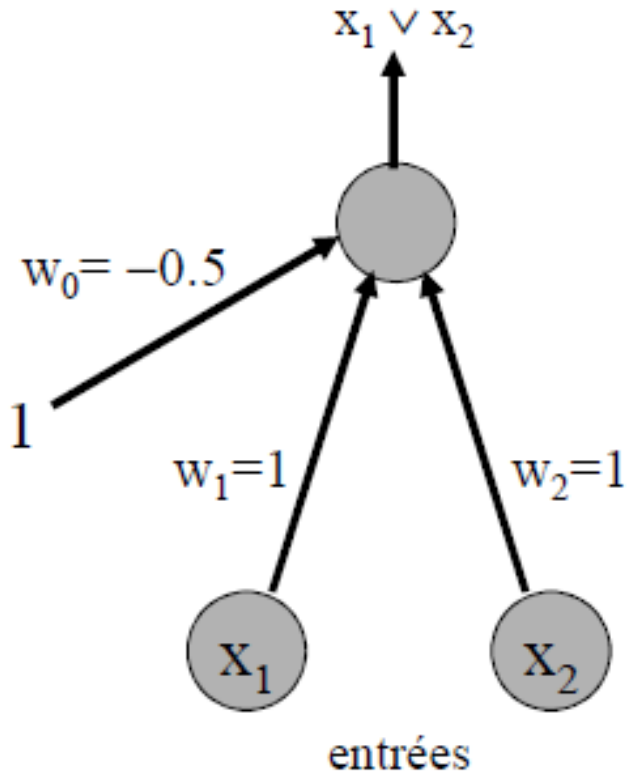
Exemple: négation logique

entrée x1	sortie
0	1
1	0



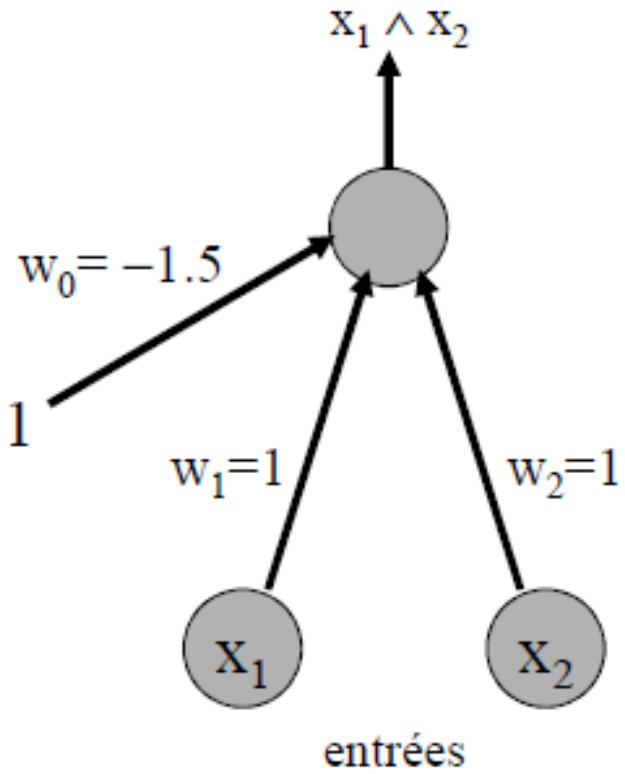
Exemple: "ou" logique

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



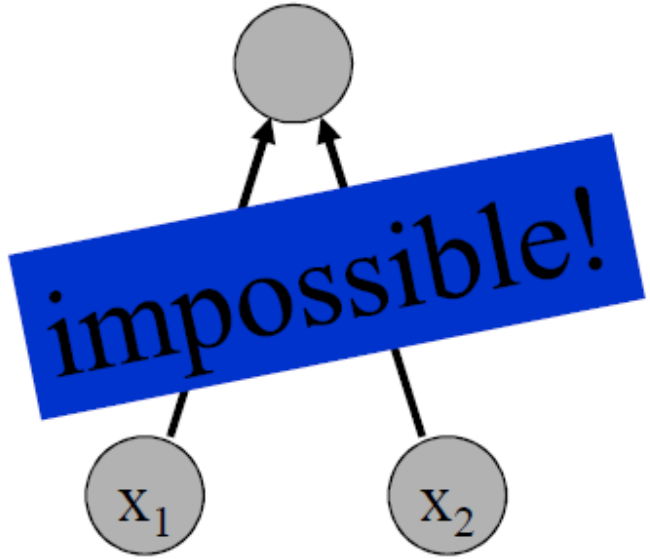
Exemple: "et" logique

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



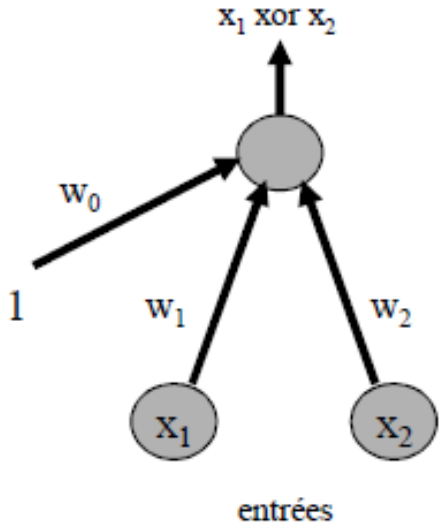
Exemple: "xor" logique (ou exclusif)

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



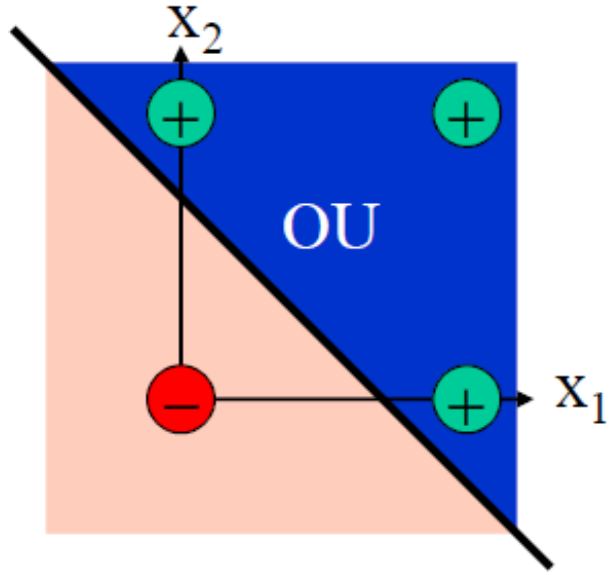
Le système d'équations à résoudre est:

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

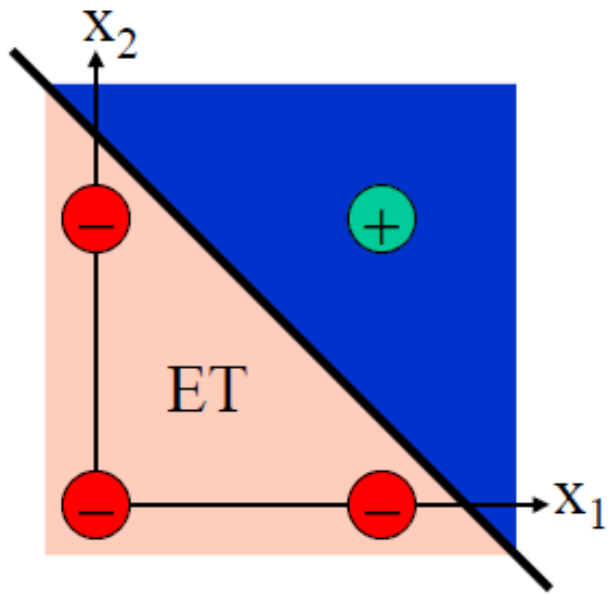
Il n'y a aucune valeur possible pour les coefficients w_0, w_1 et w_2 qui satisfasse les inégalités ci-contre.

xor ne peut donc pas être représenté!

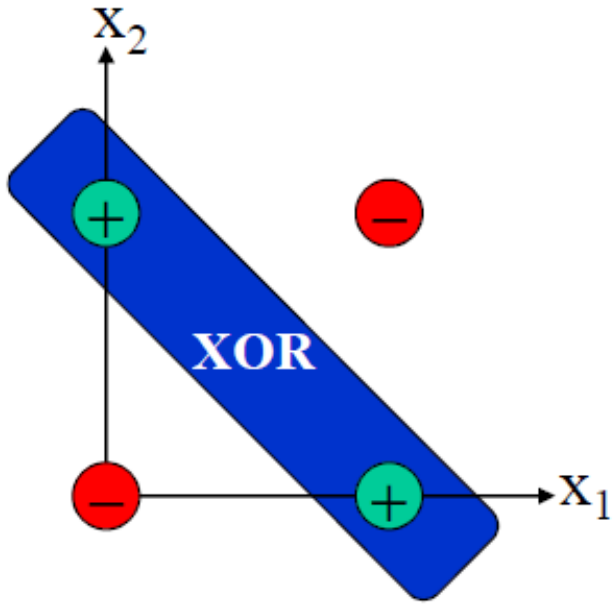
Séparabilité linéaire

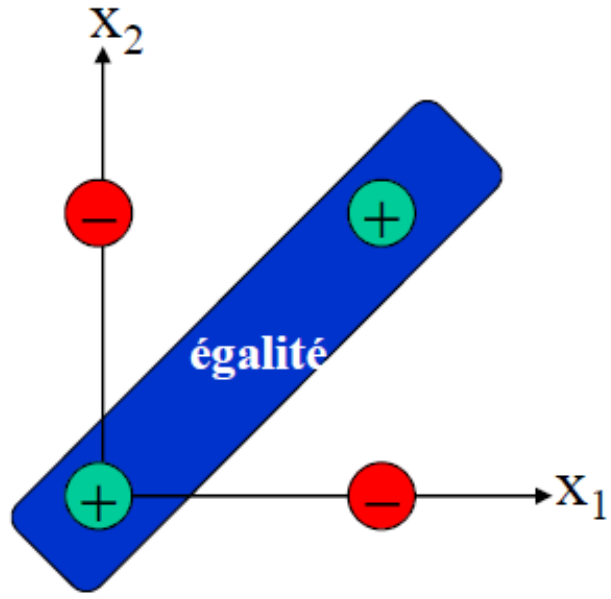


Séparabilité linéaire



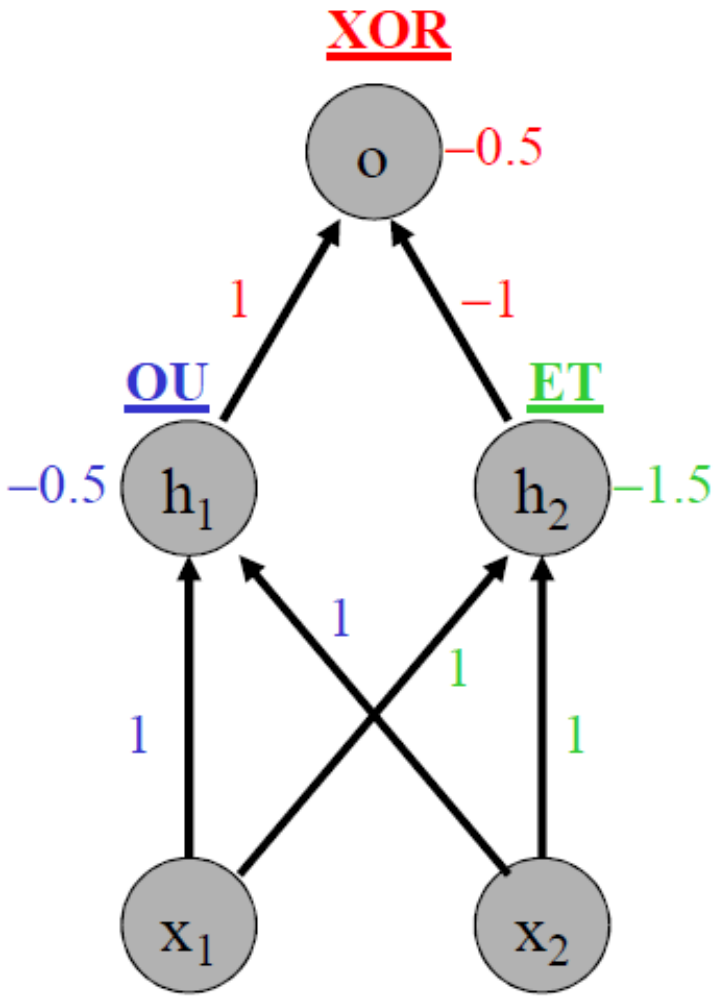
Non-séparabilité linéaire





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

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

Méthode

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

répéter jusqu'à convergence:

pour chaque exemple

calculer la valeur de sortie *o* du réseau.

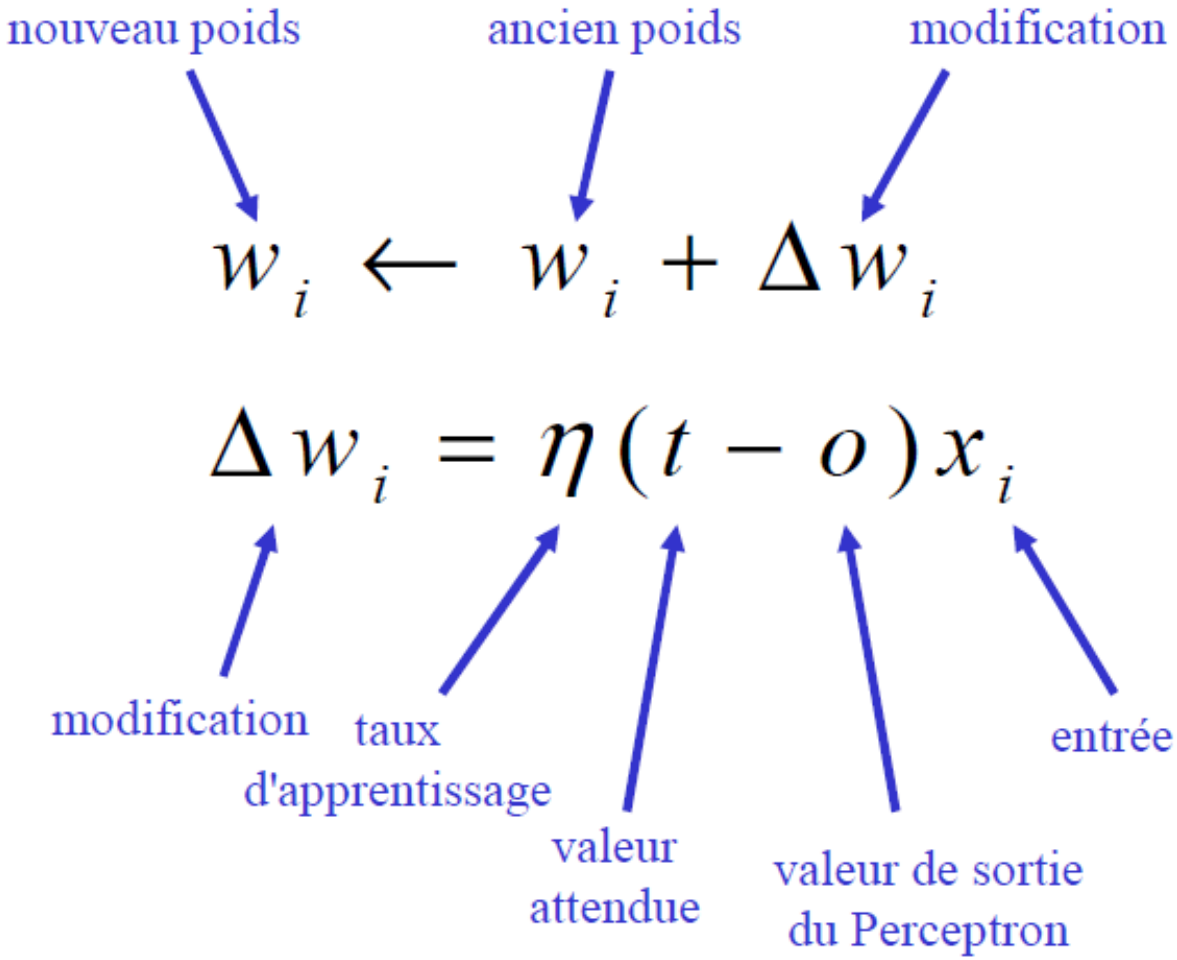
ajuster les poids:

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

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

Règle d'apprentissage du *Perceptron*

Algorithme d'apprentissage du Perceptron (résumé)



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:

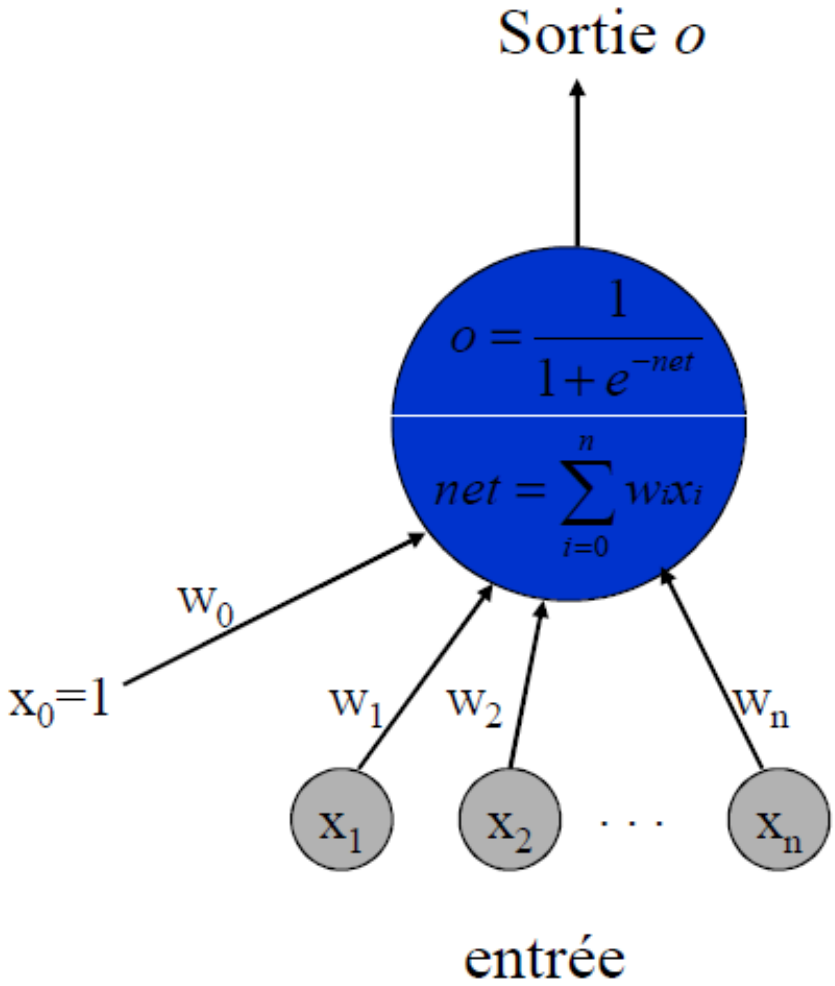
… les données d'apprentissage sont linéairement séparables
… le taux d'apprentissage η est suffisamment petit
… il n'y a pas d'unités "cachées" (une seule couche)

Réseaux multi-couches

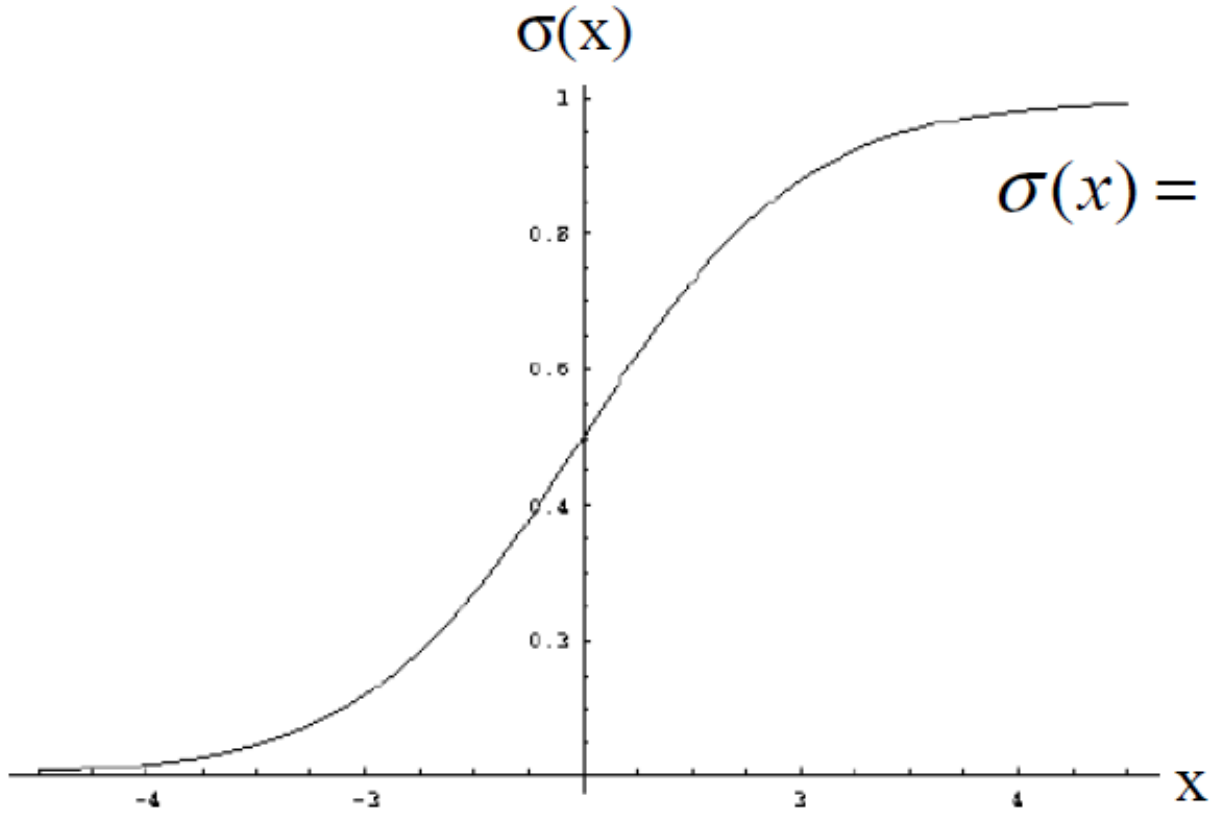
- Un réseau à 2 couches (une couche cachée) avec des unités à seuil peut représenter la fonction logique "ou exclusif" (xor).

- Les réseaux multi-couches "feedforward" peuvent être entraînés par rétro-propagation pour autant que la fonction de transition des unités soit différentiable (les unités à seuil ne conviennent donc pas).

Unité "sigmoïde"



Fonction sigmoïde



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

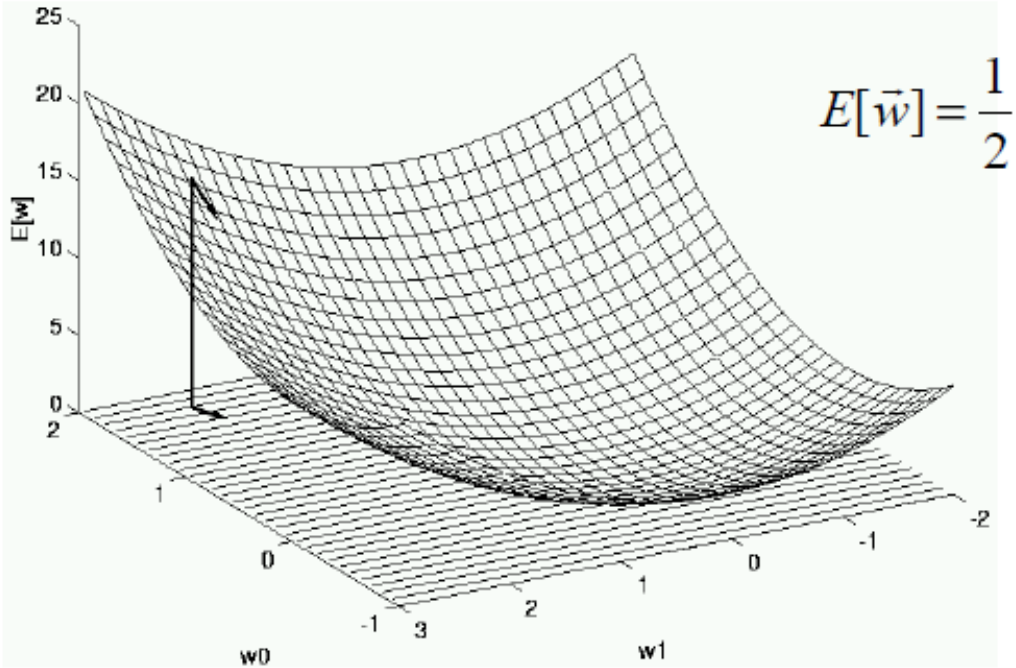
$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$ ⟵ Base pour le procédé de descente gradient

Descente de gradient

- Il faut apprendre les valeurs des poids w_i qui minimisent l'erreur quadratique

$$E[\vec{w}] = \frac{1}{2}\sum_{e}(t_e - o_e)^2$$

Descente de gradient (suite)



$$E[\vec{w}] = \frac{1}{2}\sum_{e}(t_e - o_e)^2$$

Gradient:

$$\nabla E[\vec{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$$

Règle d'apprentissage:

$$\Delta \vec{w} = -\eta \, \nabla E[\vec{w}] \qquad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

CONCLUSION SUR LE PERCEPTRON

Avantages :

Il représente une bonne base pour la compréhension des Perceptrons multi-couche.

Inconvénients :

Il ne peut apprendre que des problèmes de classification linéairement séparables.