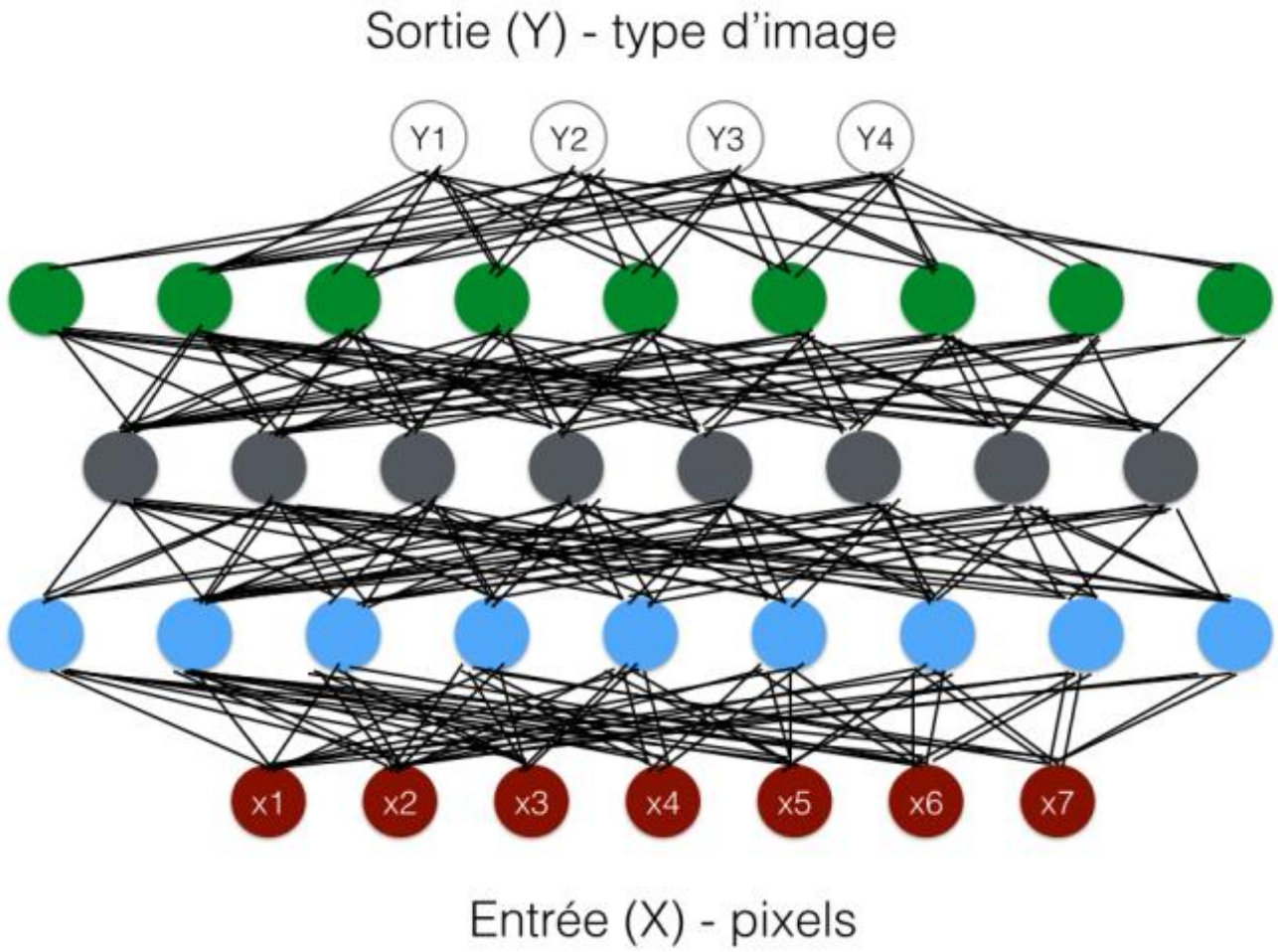
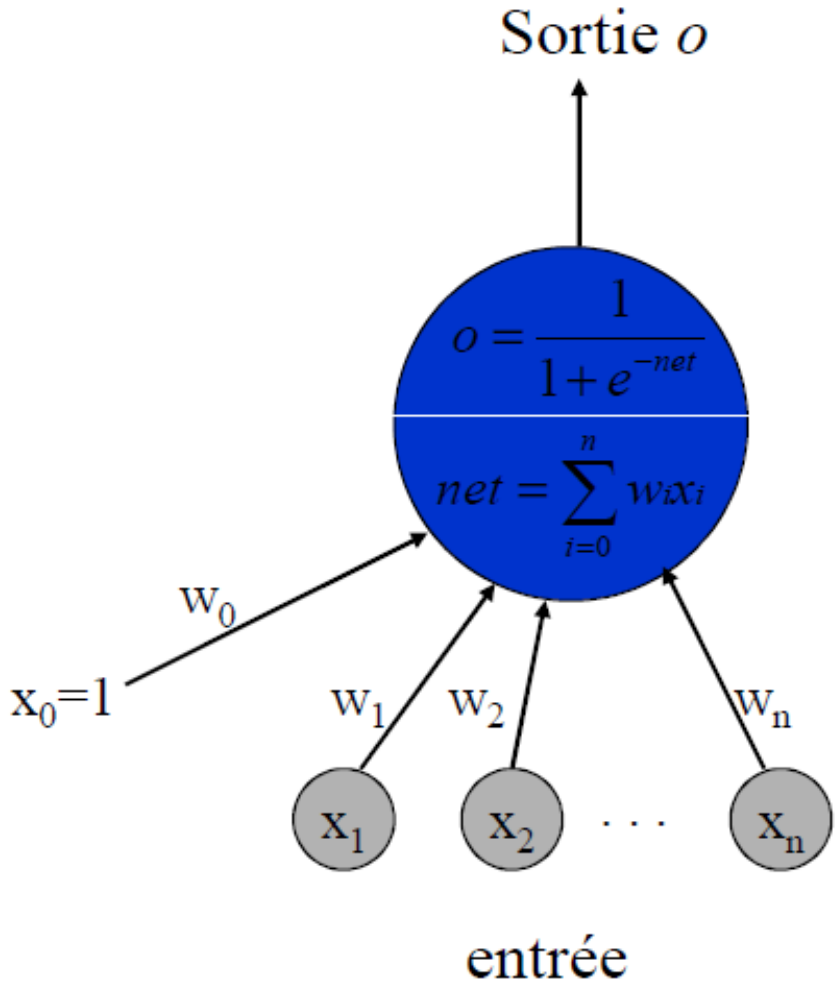
APPRENTISSAGE SUPERVISÉ : PERCEPTRONS MULTI COUCHES

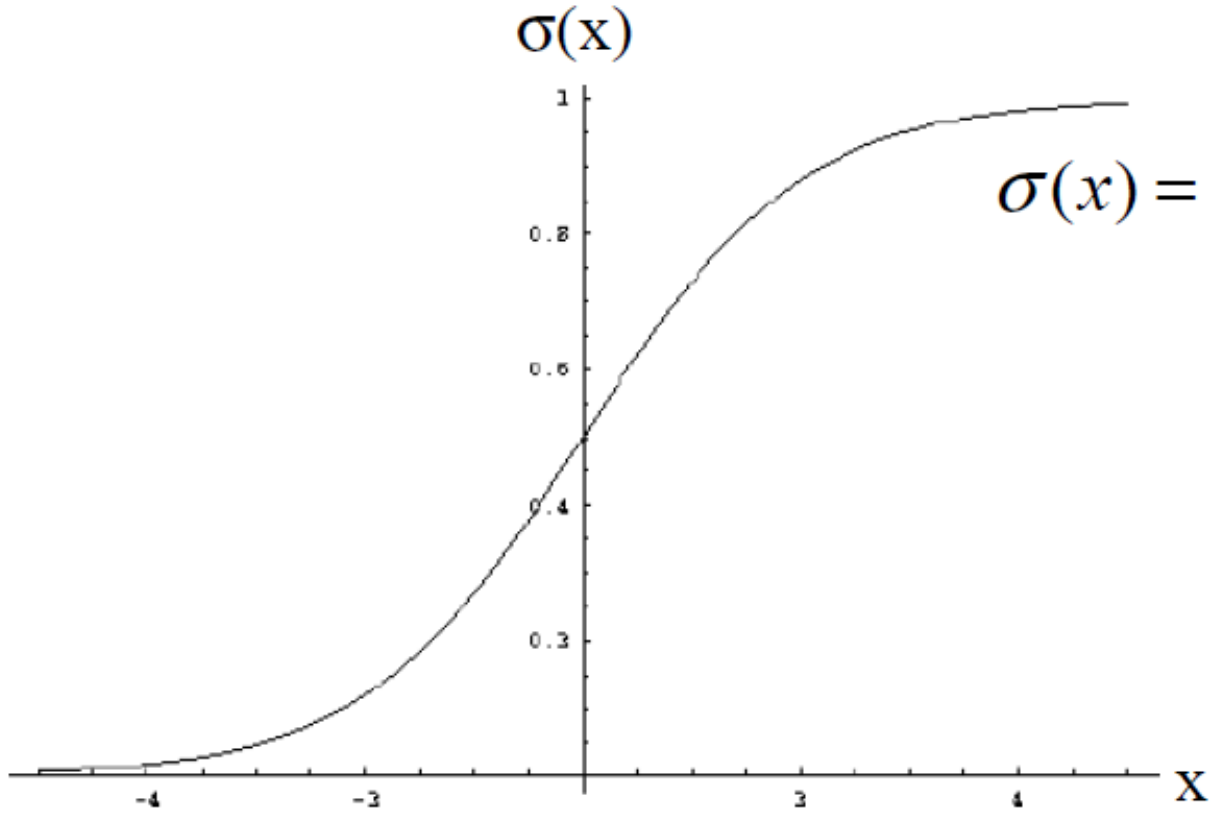
PRINCIPE


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

UNITE AVEC SIGMOIDE



FONCTION SIGMOIDE



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

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

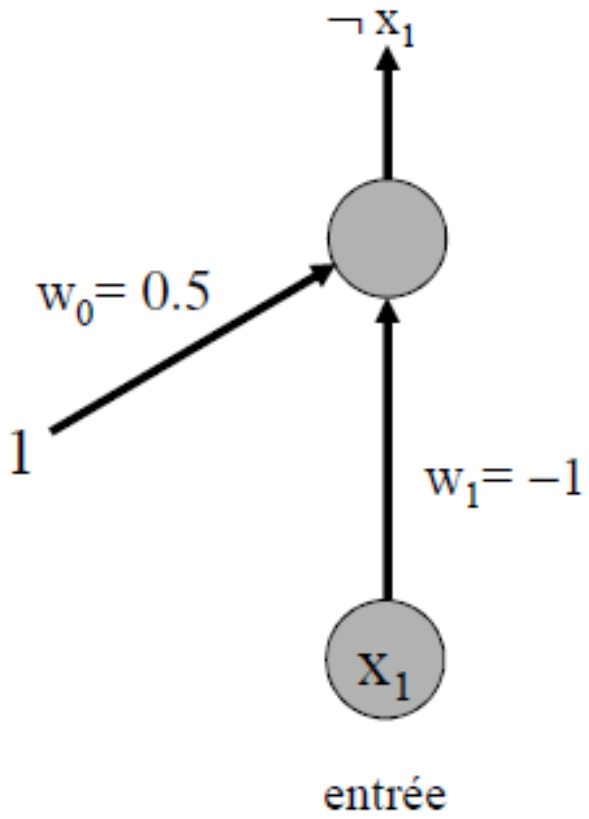
DESCENTE DE GRADIENT

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

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

Exemple: négation logique

entrée x1	sortie
0	1
1	0



Exemple avec 1 poids (sans le neurone biais)

On suppose que la fonction d'activation du neurone de sortie est linéaire.

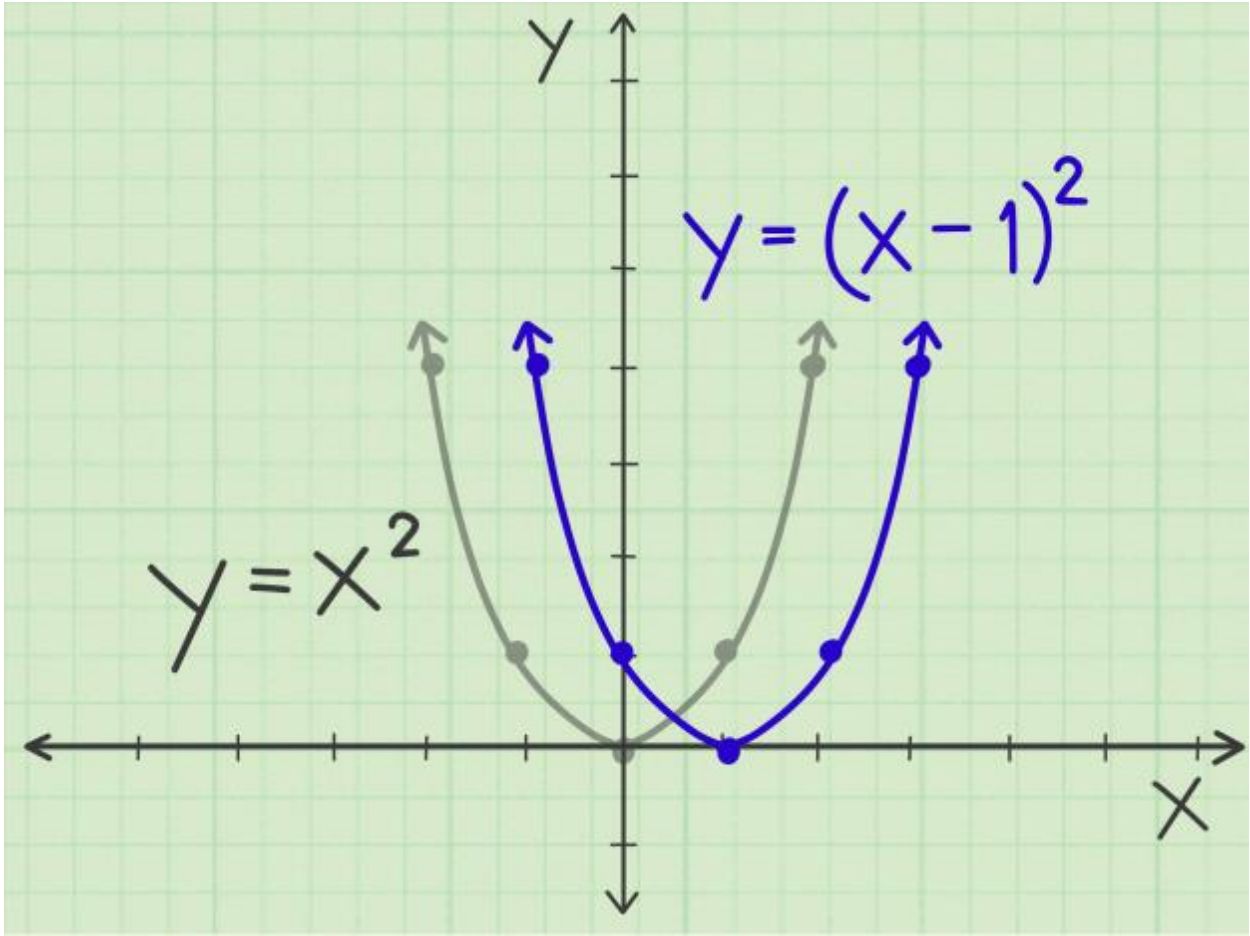
$0.5 * E(W) = (1 – 0*w_1))^2 + (0 – 1*w_1)^2$
$E(W) = 2 * (1 + w_1^2)$

En général, nous cherchons à minimiser E par rapport aux poids W. On voit ci-dessus qu'il s'agit d'une parabole.

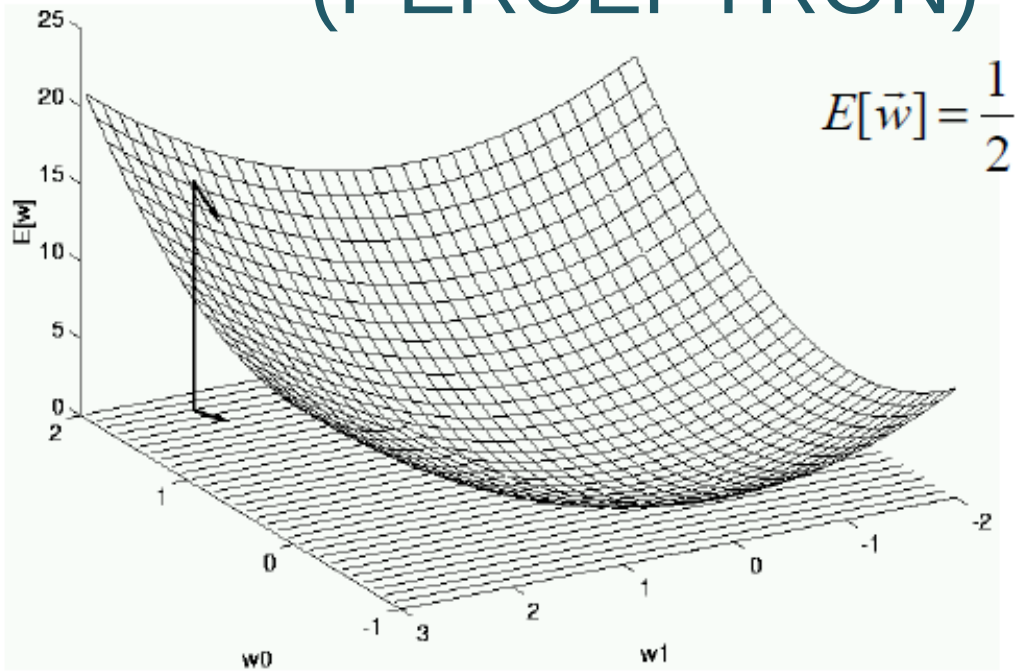
La stratégie de base est de suivre la pente la plus forte, c'est-à-dire le gradient.

Parabole (1 dimension)

Aller vers le minimum en faisant des petits pas par rapport à la tangente.



DESCENTE DE GRADIENT (PERCEPTRON)



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

Gradient:

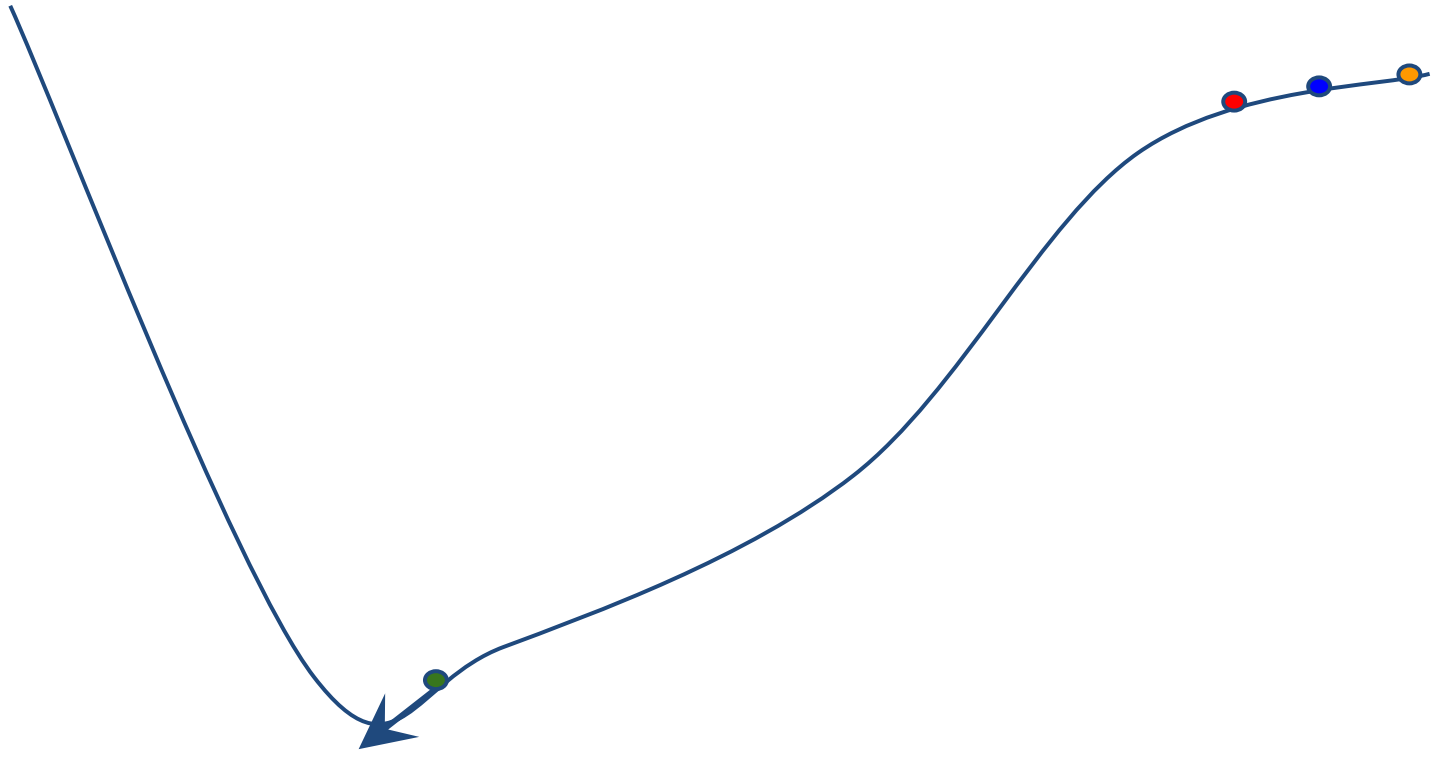
$$\nabla E[\vec{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right]$$

Règle d'apprentissage:

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

DESCENTE DE GRADIENT

Idée de la descente de gradient : faire des petits pas dans la direction de la pente jusqu'à atteindre le minimum.



DESCENTE DE GRADIENT

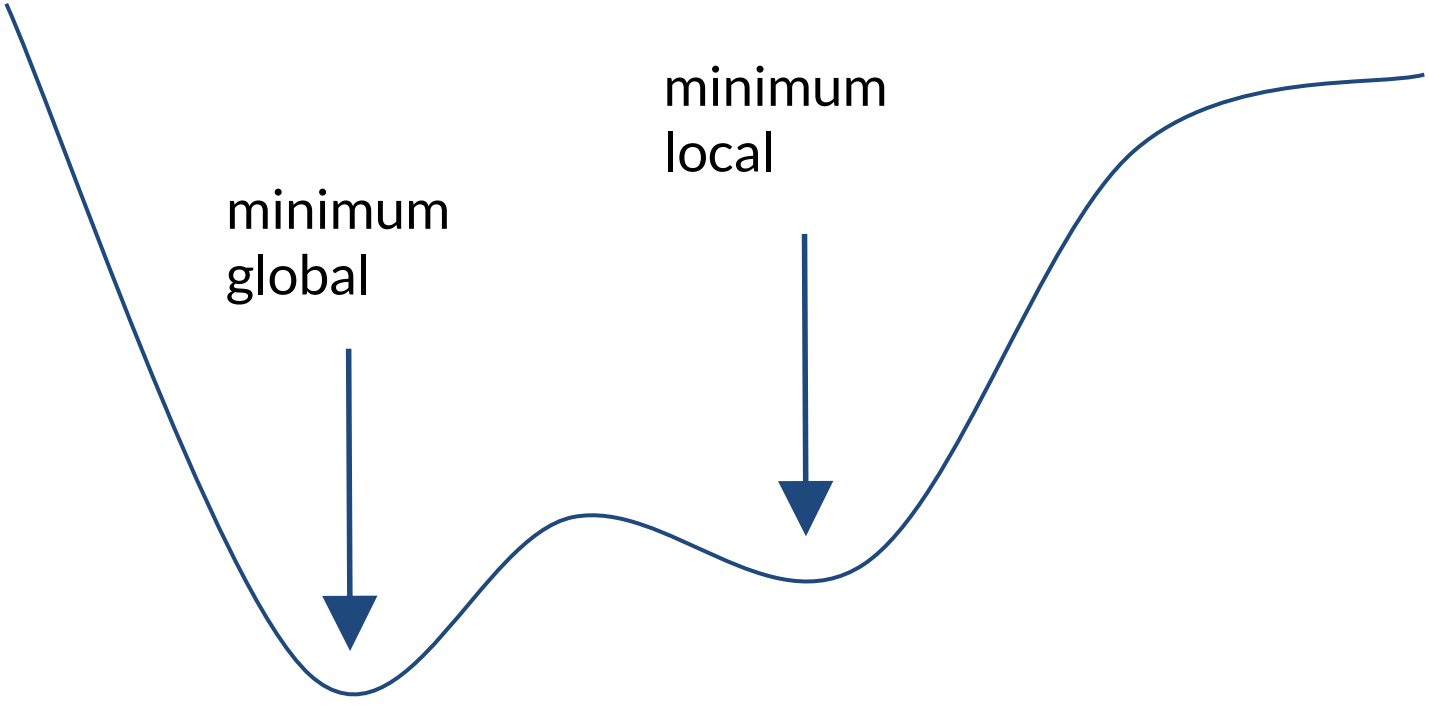
Idée de la descente de gradient: faire des petits pas dans la direction de la pente jusqu'à atteindre le minimum.

La taille des pas est ce qu'on appelle le taux d'apprentissage ou encore learning rate. Il nous dit à quel point ce que nous venons de voir doit influencer la modification des poids.

Il est généralement noté η.

DESCENTE DE GRADIENT

Avec un Perceptron (simple) notre fonction est convexe, mais cela n'est pas vrai lorsqu'on rajoute des couches intermédiaires.
Si le learning rate est trop petit, on peut rester bloqué dans un minimum local.
S'il est trop grand, on peut passer à côté du minimum global.



EXEMPLE

Je reçois une nouvelle image, je prédis grâce à mon réseau qu'il s'agit d'un chat.
Une fois au niveau de la sortie, je m'aperçois que c'est en fait un camion.
Je rétro-propage cette information pour modifier la valeur de mes (nombreux) paramètres (les poids).
Plus le réseau voit d'exemples étiquetés, plus il emmagasine de connaissances et devient "bon".

DESCENTE DU GRADIENT (SUITE)

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_e (t_e - o_e)^2$$

$$= \frac{1}{2}\sum_e \frac{\partial}{\partial w_i}(t_e - o_e)^2$$

$$= \frac{1}{2}\sum_e 2(t_e - o_e)\frac{\partial}{\partial w_i}(t_e - o_e)$$

$$= \sum_e (t_e - o_e)\frac{\partial}{\partial w_i}(t_e - \sigma(\vec{w}\cdot\vec{x}_e))$$

$$= \sum_e (t_e - o_e)(-\frac{\partial}{\partial w_i}\sigma(\vec{w}\cdot\vec{x}_e))$$

$$= -\sum_e (t_e - o_e)\sigma(\vec{w}\cdot\vec{x}_e)(1-\sigma(\vec{w}\cdot\vec{x}_e))x_{i,e}$$

$$= -\sum_e (t_e - o_e)o_e(1-o_e)x_{i,e}$$

Remarque

Si la fonction d'activation est l'identité au lieu de la sigmoïde, la dérivée partielle par rapport à w_i est plus simple et on retombe sur la règle du Perceptron.

En effet, si on a :

$w_0 + w_1x_1 + w_2x_2 + \ldots + w_ix_i + \ldots w_nx_n$

La dérivée partielle par rapport à w_i est x_i.

AJUSTEMENT DES POIDS

- Le terme d'erreur $(t_e - o_e)$ est connu pour les unités de la couche de sortie. Pour ajuster les poids entre la couche cachée et celle de sortie, on peut utiliser le processus de descente de gradient.

- Pour ajuster les poids entre la couche d'entrée et la couche cachée, il faut trouver le moyen d'estimer les erreurs commises par les unités de la couche cachée.

RETRO-PROPAGATION DE L'ERREUR

Idée: chaque unité cachée est responsable pour une fraction de l'erreur commise par chacune des unités de sortie.

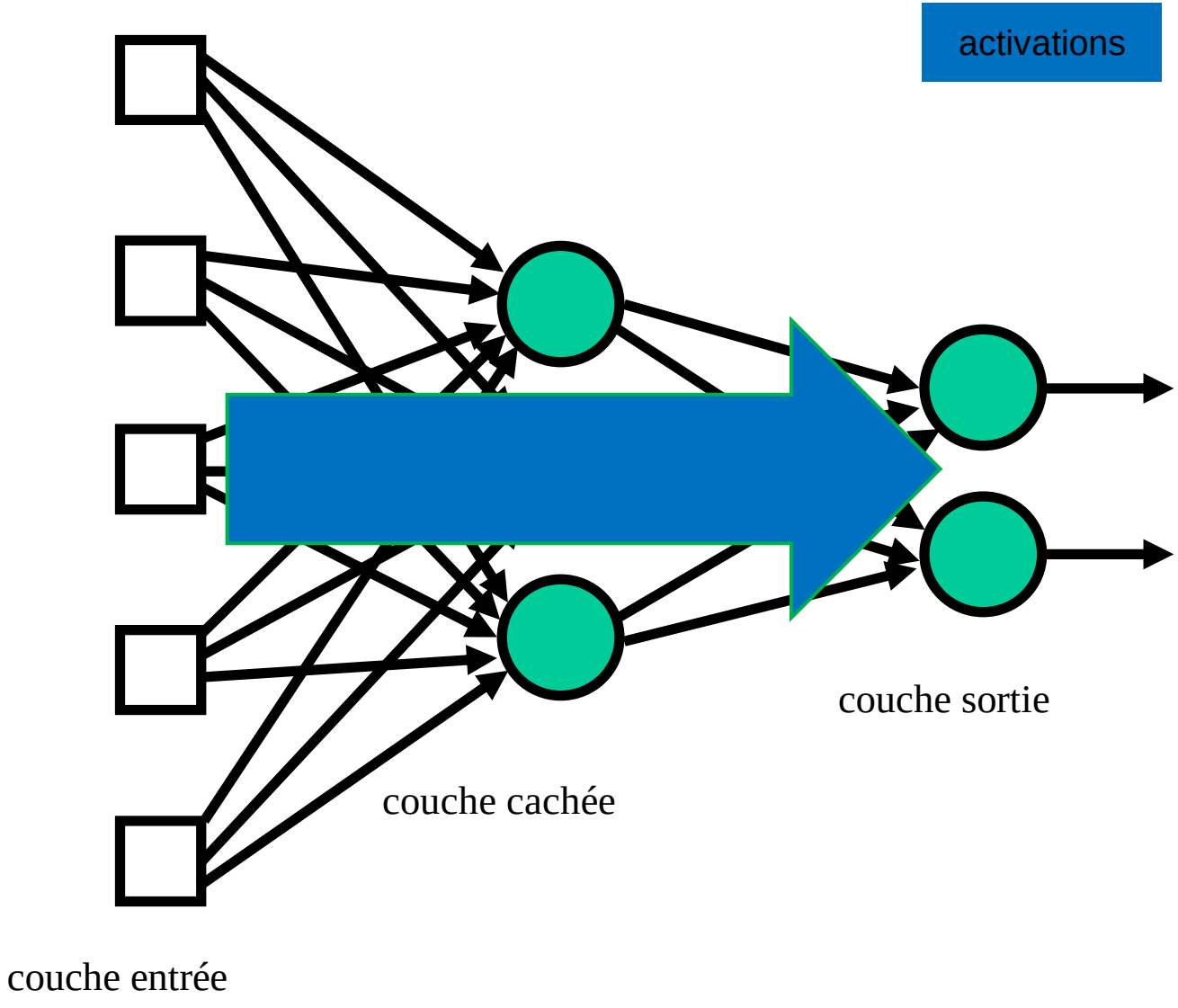
$$\text{erreur pour l'unité cachée } j = \sum_{i \in sorties} w_{ij} \delta_i$$

où δ_i est l'erreur pour l'unité de sortie i

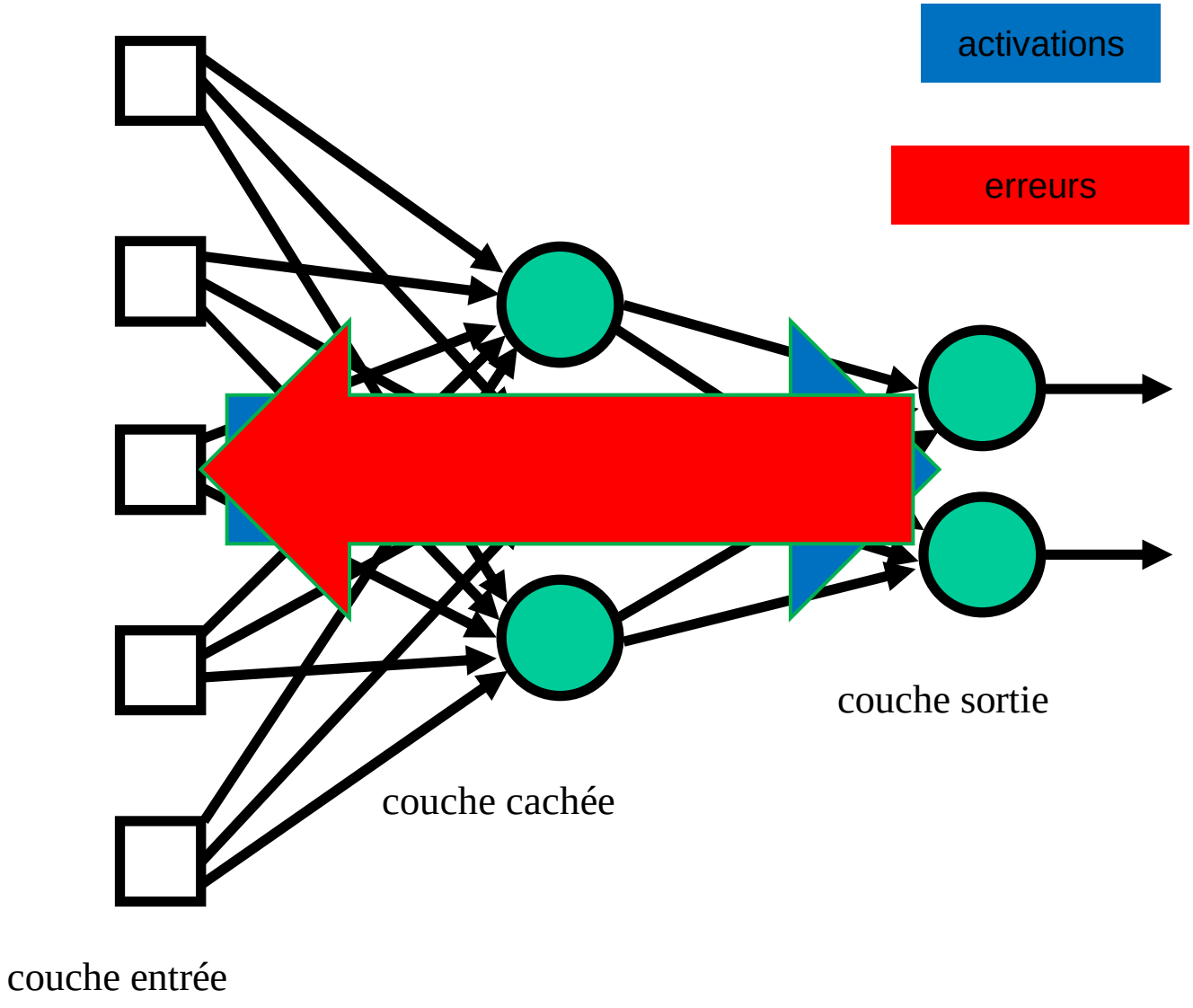
ALGORITHME DE RETRO-PROPAGATION

- Initialiser tous les poids du réseau avec de petites valeurs aléatoires.
- Pour chaque exemple de l'ensemble d'apprentissage faire:
  - pour chaque unité cachée h, calculer: $o_h = \sigma(\sum_i w_{hi} x_i)$
  - pour chaque unité de sortie k, calculer: $o_k = \sigma(\sum_k w_{kh} x_h)$
  - pour chaque unité de sortie k, calculer: $\delta_k = o_k(1-o_k)(t_k - o_k)$
  - pour chaque unité cachée h, calculer: $\delta_h = o_h(1-o_h)\sum_k w_{hk}\delta_k$
  - mettre à jour chaque poids du réseau: $w_{ij} \leftarrow w_{ij} + \eta\delta_j x_{ij}$

ALGORITHME DE RETRO-PROPAGATION (suite)



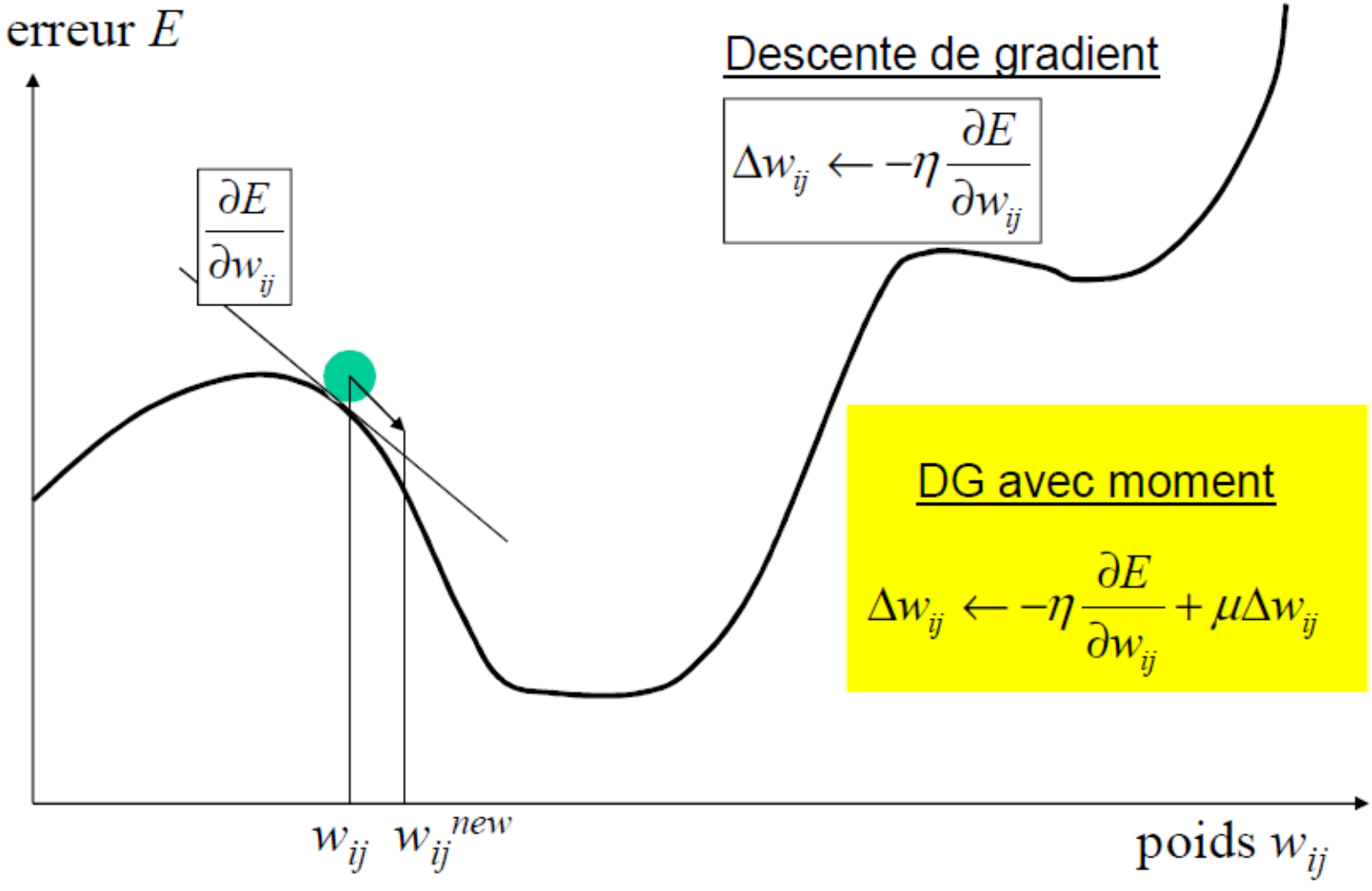
ALGORITHME DE RETRO-PROPAGATION (suite)



CARACTERISTIQUES DE LA RETRO-PROPAGATION

- Processus de descente de gradient effectué sur tout le vecteur des poids du réseau.
- Minimum d'erreur local, pas nécessairement global.
- Minimisation de l'erreur sur l'ensemble d'apprentissage; risque de sur-apprentissage comme pour les arbres de décision.
- Apprentissage nécessite des milliers d'itérations -- très lent!
- Décider de la topologie et fixer les valeurs des paramètres (par ex: taux d'apprentissage) est plus un *art* qu'une science.

ACCELERATION DE L'APPRENTISSAGE : LE MOMENT



PARAMÈTRES GLOBAUX

Il faut choisir un certain nombre de choses pour son réseau:

La complexité:

Le nombre de **couches** : plus elles sont nombreuses, plus on pourra prédire des choses compliquées, plus le temps de calcul est long. Si elles sont trop nombreuses, on tombe dans le **sur-apprentissage**.

Le nombre de **neurones** : idem !

Les **fonctions** d'activation et de sortie: on en propose un certain nombre. Elles dépendent en partie du type de sortie (par exemple : classification binaire vs multi-classes)

Le **learning rate** : le taux d'apprentissage. S'il est trop élevé, chaque exemple va influer beaucoup sur le réseau et on peut tomber dans un optimum **local**. S'il est faible, le réseau va être très long à optimiser. On choisit souvent un paramètre dynamique : il baisse au fur et à mesure de l'apprentissage.

BATCH ET ONLINE LEARNING

Bien que l'on ait accès à de puissants serveurs de calculs, on peut vouloir limiter l'utilisation de la mémoire pour entraîner ces algorithmes très lourds.

On peut alors envisager de ne fournir les données au réseau que **par batchs**. Ainsi, il ne traite qu'une seule partie des données à la fois, sans avoir besoin de stocker les autres. Il update les paramètres à la fin de chaque batch.

CONCLUSION SUR LE PERCEPTRON MULTI-COUCHES

Avantages :

Potentiellement extrêmement performants, en particulier dans certains domaines comme la reconnaissance d'images ou du langage.

Flexibles sur la complexité.

Inconvénients :

Peu/pas de résultats théoriques pour expliquer la performance.

Impossibles à interpréter.

Calculs extrêmement lourds.

Complexité difficile à calibrer : gros risques de **sur-apprentissage**.

BIBLIOGRAPHIE

- Hopfield, J. J. (1982). *Neural networks and physical systems with emergent collective computational abilities*. Proceedings of the National Academy of Sciences 79:2554-2558.
- Hertz, J., A. Krogh, and R. G. Palmer. (1991). *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley.
- Rumelhart, D. E., J. McClelland, and the PDP Research Group. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press.
- Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York: Oxford University Press.
- Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. New York: Macmillan College Publishing.
- Churchland, P. S., and T. J. Sejnowski. (1992). *The Computational Brain*. Cambridge, MA: MIT Press.
- Arbib, A. M. (1995). *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press.
- Sejnowski, T. (1997). *Computational neuroscience. Encyclopedia of Neuroscience*. Amsterdam: Elsevier Science Publishers.