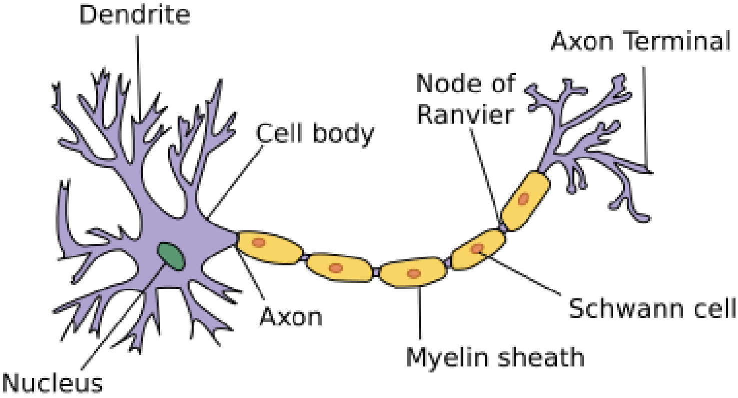
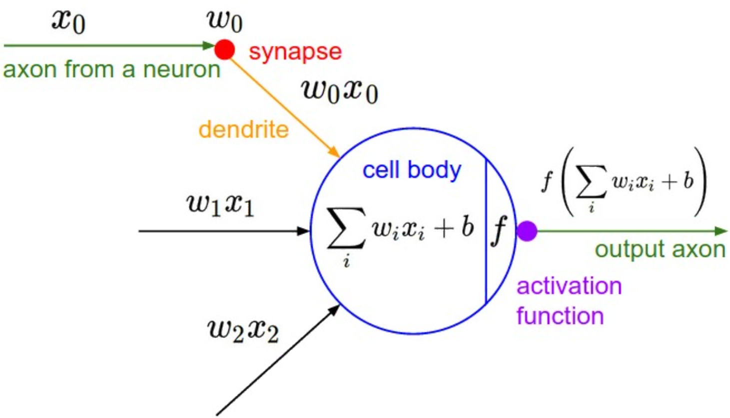
Machine Learning

Réseau de Neurones

Réseau de Neurones



Réseau de Neurones



Réseau de Neurones

$$\sigma\left(\begin{bmatrix}x_1 & x_2 & \dots & x_d\end{bmatrix} * \begin{bmatrix}w_{11} & w_{12} & \dots & w_{1n}\\ w_{21} & w_{22} & \dots & w_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ w_{d1} & w_{d2} & \dots & w_{dn}\end{bmatrix} + \begin{bmatrix}b_1 & b_2 & \dots & b_n\end{bmatrix}\right)$$

$$= \begin{bmatrix}o_1 & o_2 & \dots & o_n\end{bmatrix}$$

où :

X est une donnée en entrée de dimension **d**,

w et b sont les paramètres à trouver des **n** neurones de notre modèle.

σ la fonction d'activation et

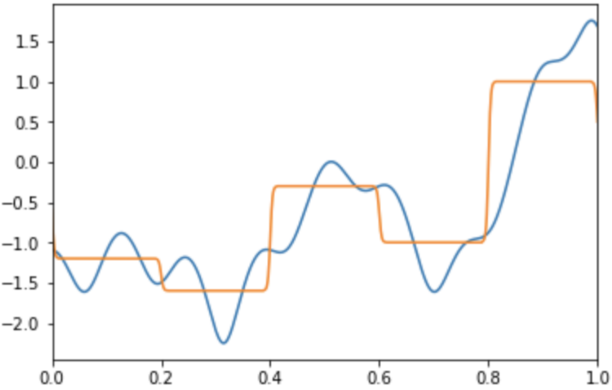
O la sortie du réseau

Réseaux de neurones : Fonction d'activation σ

- Sigmoïde
- Tanh
- Softmax
- ReLU
- ...

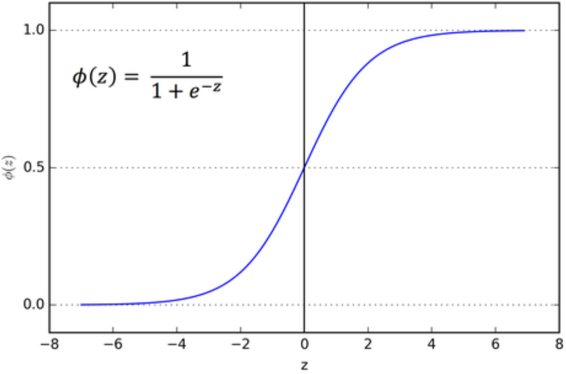
Réseaux de neurones : un Potentiel infini

1991 : **Kurt Hornik** : Théorème d'approximation universelle



Réseaux de neurones : Fonction d'activation σ

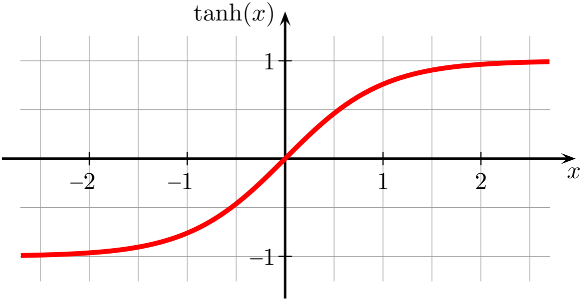
Sigmoïde



$$\frac{\partial \phi(x)}{\partial x} = \phi(x) * (1 - \phi(x))$$

Réseaux de neurones : Fonction d'activation σ

$$\tanh(x) = \frac{1 - \exp -2 * x}{1 + \exp -2 * x}$$



$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh^2(x)$$

Réseaux de neurones : Fonction d'activation σ

$$Softmax(x_j) = \frac{\exp x_j}{\sum_{i=1}^{n} \exp x_i}$$

donc :

$$\sum_{j=1}^{n} Softmax(x_j) = 1$$

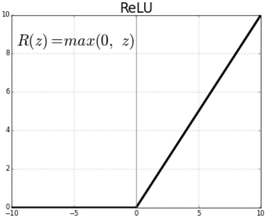
dérivée (ou jacobien car le softmax est une fonction de $\mathbb{R}^n \to \mathbb{R}^n$) :

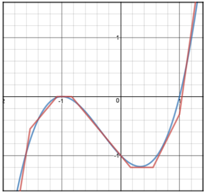
$$D_j S_i = S_i(\delta_{ij} - S_j)$$

où $D_j S_i$ est la dérivée partielle de la i-ième sortie par rapport à la j-ième entrée

δ_{ij} est le delta de Kronecker

Réseaux de neurones : Fonction d'activation σ

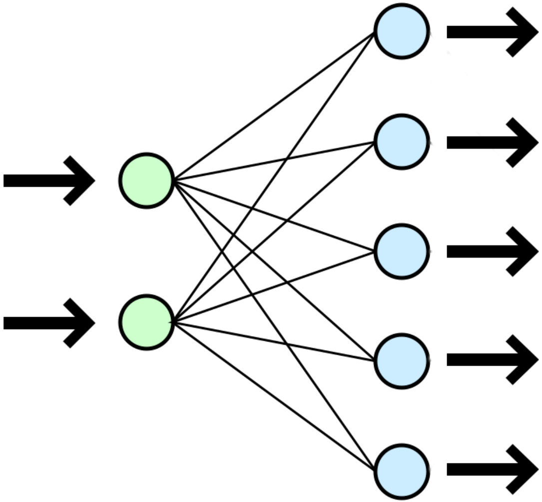




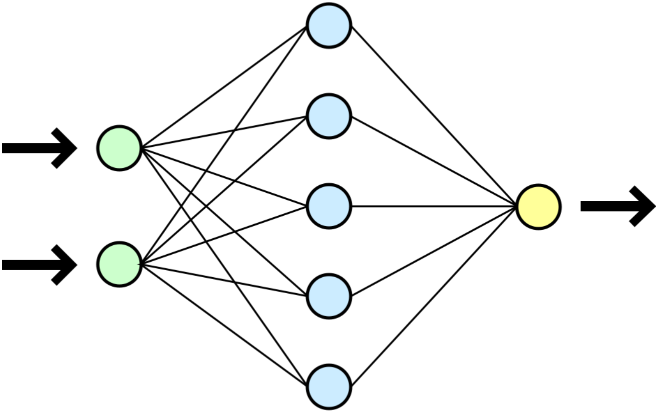
$$n_1(x) = Relu(-5x - 7.7)$$
$$n_2(x) = Relu(-1.2x - 1.3)$$
$$n_3(x) = Relu(1.2x + 1)$$
$$n_4(x) = Relu(1.2x - .2)$$
$$n_5(x) = Relu(2x - 1.1)$$
$$n_6(x) = Relu(5x - 5)$$

$$Z(x) = -n_1(x) - n_2(x) - n_3(x) + n_4(x) + n_5(x) + n_6(x)$$

Réseau de Neurones



Réseau de Neurones



Réseau de Neurones