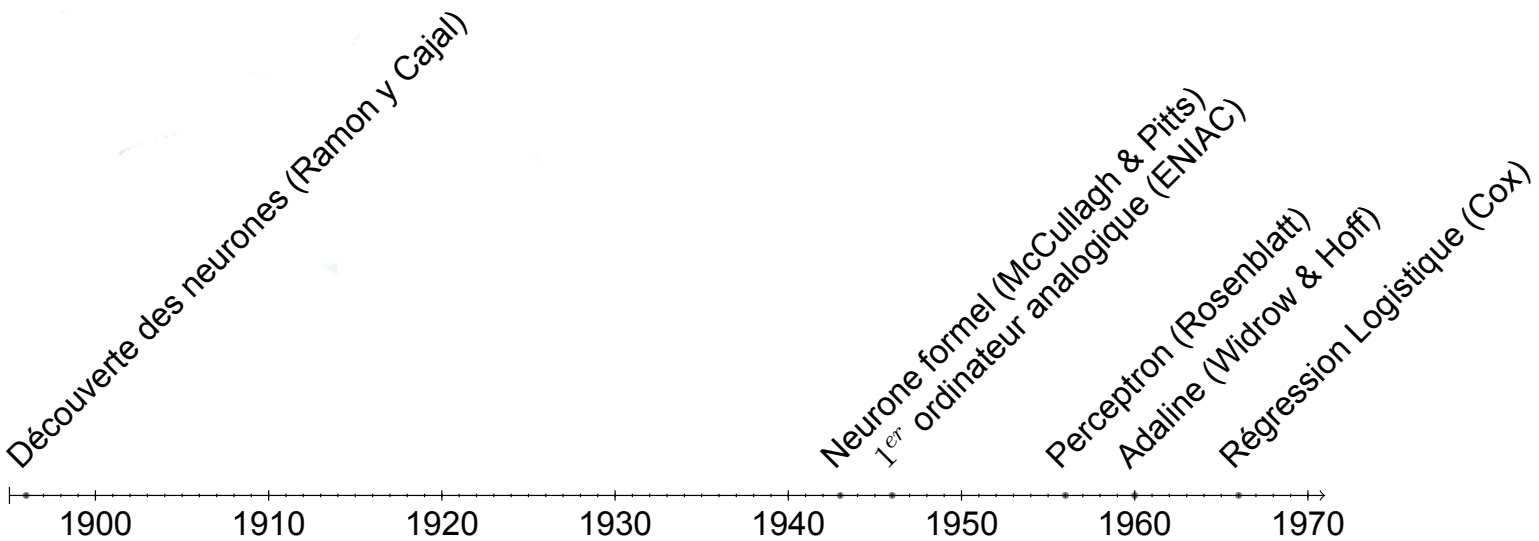
Introduction au Machine Learning

Massih-Reza Amini, Alexandre Audibert, Aurélien Gauffre

Université Grenoble Alpes
Laboratoire d'Informatique de Grenoble
Prenom.Nom@univ-grenoble-alpes.fr

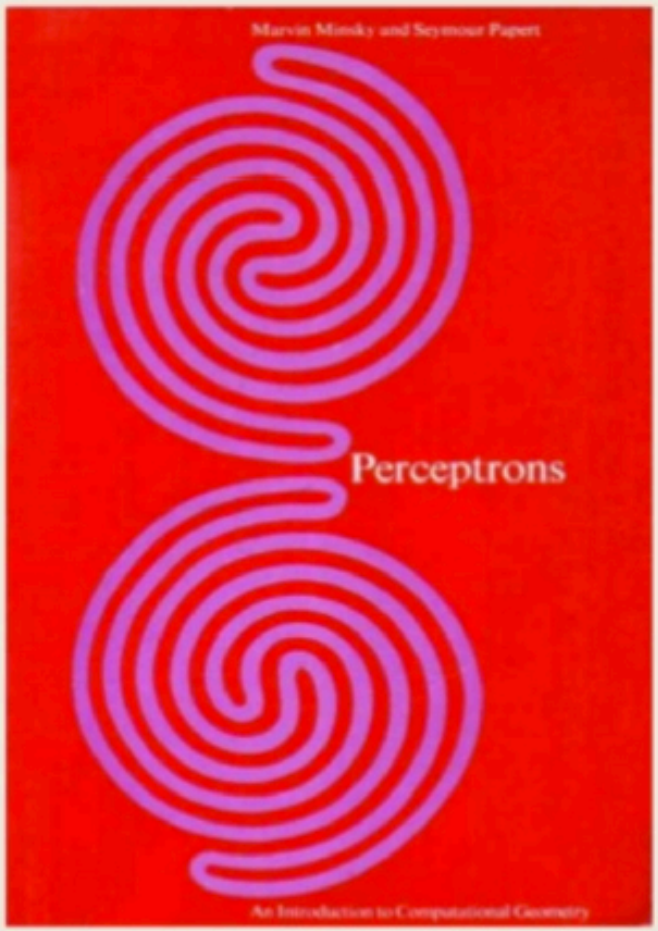
Les algorithmes basés sur le neurone formel



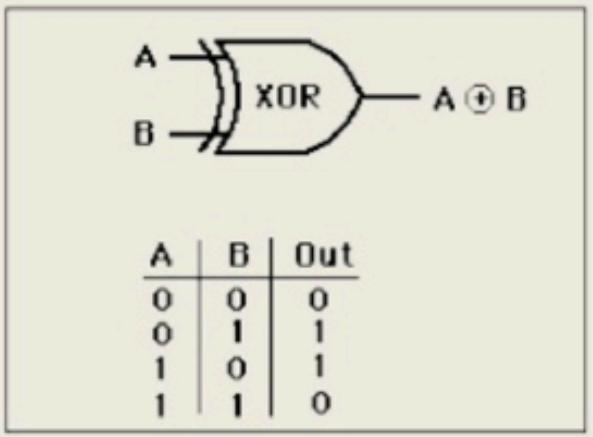
Les algorithmes basés sur le neurone formel

- Peu de problèmes sont linéairement séparables

1969: Perceptrons can't do XOR!



http://www.i-programmer.info/images/stories/BabBag/AI/book.jpg



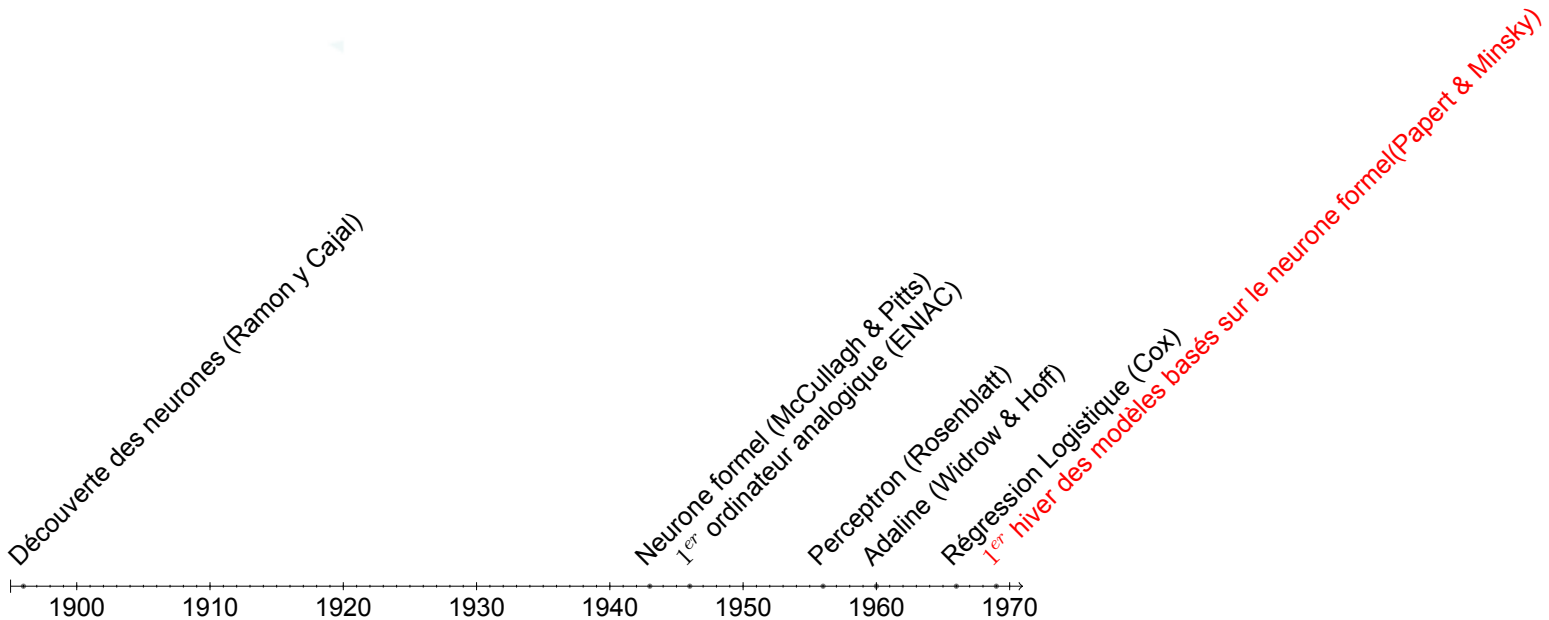
http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/ietron/xor.gif

Minsky & Papert

https://constructingkids.files.wordpress.com/2013/05/minsky-papert-71-csolomon-x640.jpg

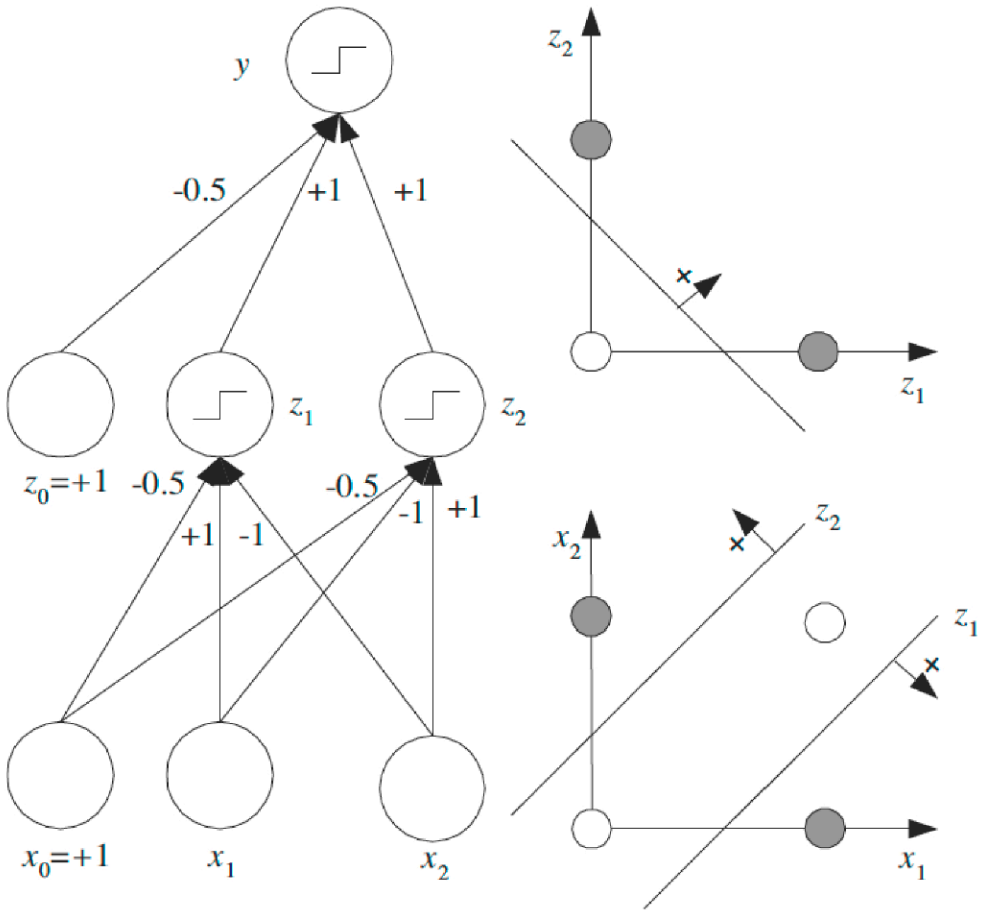
Les algorithmes basés sur le neurone formel

- Ceci a marqué le 1^{er} hiver des modèles à base du neurone formel;
  $\Rightarrow$ Abandon des modèles linéaires,
  - Développement de l'IA symbolique,
  - À la recherche des modèles non-linéaires.



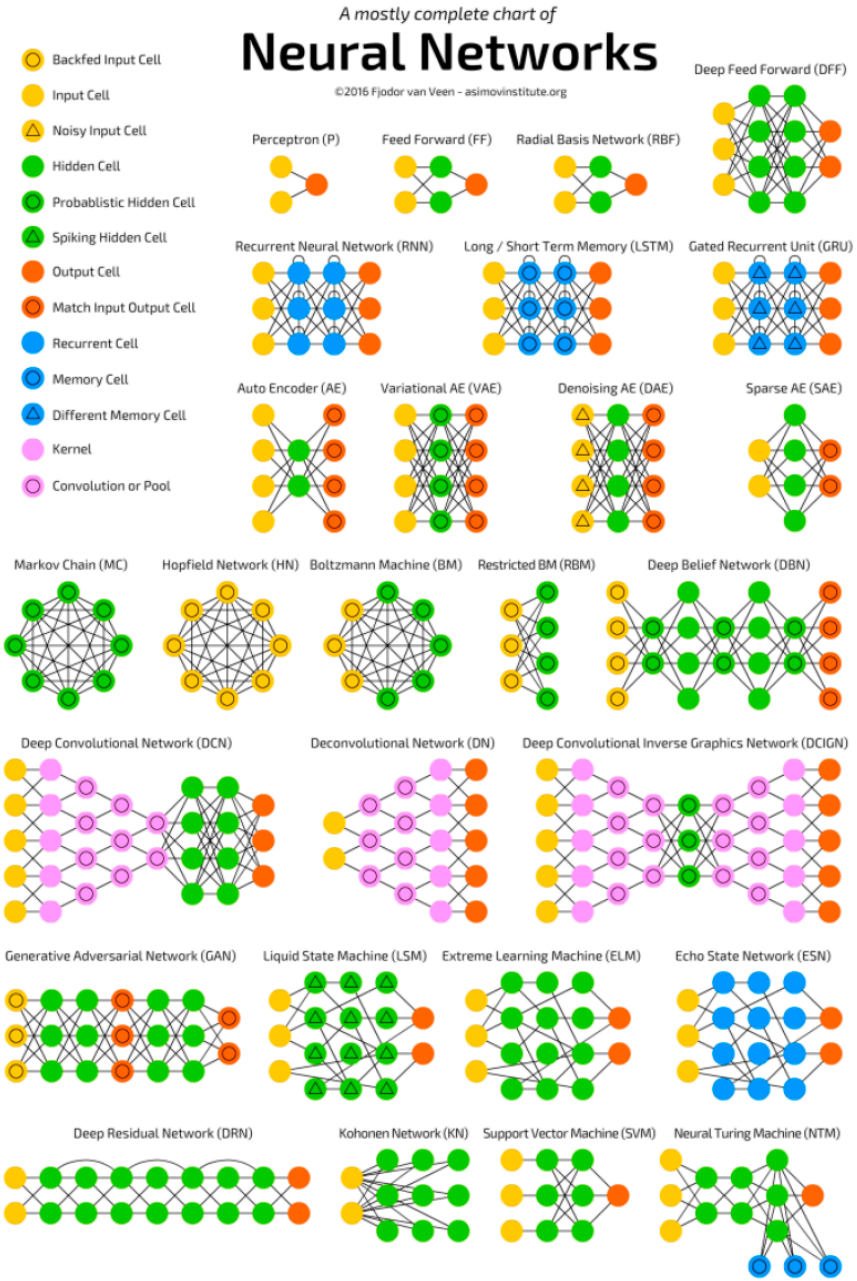
Réseaux de neurones artificiels (1985)

- Résoudre des problèmes non-linéaires en combinant des neurones formels: problème XOR



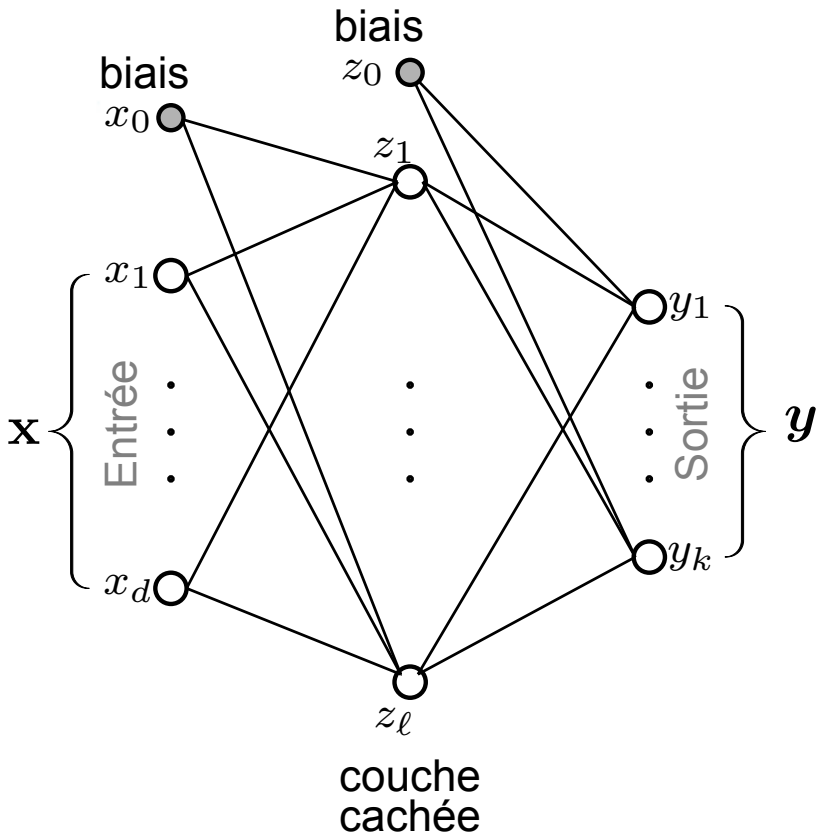
Réseaux de neurones artificiels

- Un réseau de neurones est un graphe orienté pondéré de neurones formels,
- Lorsque deux neurones sont connectés (liés par une arête orientée du graphe), la sortie du neurone de tête est utilisée comme entrée par le neurone de queue
- Les trois types de neurones sont :
  - neurones d'entrée (connectés à l'entrée)
  - neurones de sortie
  - neurones cachés


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Perceptron Multi-couches (PMC)

- Un PMC est un réseau de neurones acyclique, où les neurones sont structurés en couches successives, commençant par une couche d'entrée et finissant par une couche de sortie.



PMC: phase de propagation

Pour le modèle ci-dessus, la valeur z_j, de la $j^{\text{ème}}$ unité de la couche cachée pour une observation $\mathbf{x} = (x_i)_{i=1\ldots d}$ en entrée est obtenue par composition :

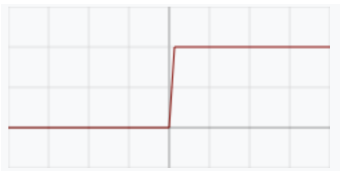
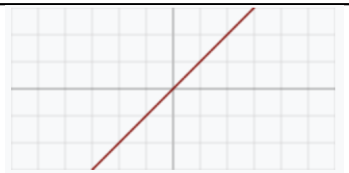
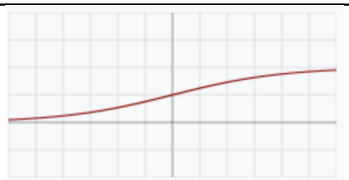
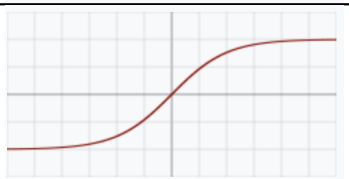
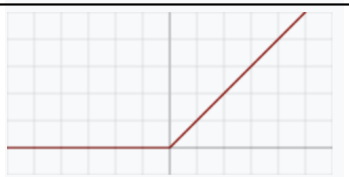
- d'un produit scalaire entre $\mathbf{x}$ et le vecteur de poids $\boldsymbol{w}_{j.}^{(1)} = (w_{ji}^{(1)})_{i=1,\ldots,d}; j \in \{1,\ldots,\ell\}$ liant $\mathbf{x}$ à cette unité j^{th}; ajouté du biais $w_{j0}^{(1)}$:

$$\forall j \in \{1,\ldots,\ell\}, a_j = \langle \boldsymbol{w}_{j.}^{(1)}, \mathbf{x} \rangle + w_{j0}^{(1)}$$
$$= \sum_{i=0}^{d} w_{ji}^{(1)} x_i$$

- et, une fonction d'activation, $\bar{H}(.)$ définie sur $\mathbb{R}$:

$$\forall j \in \{1,\ldots,\ell\}, z_j = \bar{H}(a_j)$$

Fonctions d'activation classiques

Nom	Définition	Graphe
Heaviside	$\bar{H}(z) = \begin{cases} 1, & \text{si } z \geqslant 0 \\ 0, & \text{sinon.} \end{cases}$	
Linéaire	$\bar{H}(z) = z$	
Sigmoïde	$\bar{H}(z) = \frac{1}{1+e^{-z}}$	
Tanh	$\bar{H}(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	
Relu	$\bar{H}(z) = \begin{cases} z, & \text{si } z \geqslant 0 \\ 0, & \text{sinon.} \end{cases}$	

PMC: phase de propagation

Pour le modèle précédent, la valeur de la $j^{\text{ème}}$ unité de la couche cachée pour une observation $\mathbf{x} = (x_i)_{i=1\ldots d}$ en entrée est obtenue par composition:

- Les valeurs des unités de sortie $\boldsymbol{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_K(\mathbf{x}))$ sont obtenues de la même manière entre le vecteur de la couche cachée $z_j, j \in \{0, \ldots, \ell\}$ et les poids liant cette couche à la sortie; i.e. $\boldsymbol{w}_{k.}^{(2)} = (w_{kj}^{(2)})_{j=1,\ldots,\ell}; k \in \{1, \ldots, K\}$,
- la sortie prédite pour une observation $\mathbf{x}$ est une transformation composée de l'entrée, qui pour le modèle précédent est

$$\forall \mathbf{x}, \forall k \in \{1, \ldots, K\}, h_k(\mathbf{x}) = \bar{H}(a_k) = \bar{H}\left(\sum_{j=0}^{\ell} w_{kj}^{(2)} \times \bar{H}\left(\sum_{i=0}^{d} w_{ji}^{(1)} \times x_i\right)\right)$$

PMC: phase de rétro-propagation

- Une façon efficace pour estimer les paramètres des réseaux de neurones est l'algorithme de la retro-propagation de l'erreur [Rumelhart et al., 1986],
- Pour un problème de classification, un vecteur indicateur de classe est associé à chaque classe:

$$\forall (\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}, y = k \Leftrightarrow \mathbf{y}^\top = \left(\underbrace{y_1, \ldots, y_{k-1}}_{\text{all equal to 0}}, \underbrace{y_k}_{=1}, \underbrace{y_{k+1}, \ldots, y_K}_{\text{all equal to 0}} \right)$$

- Après la phase de propagation de l'information pour un exemple $(\mathbf{x}, y)$, une erreur est estimée entre la prédiction et la sortie désirée :
  - MSE: $\ell_{MSE}(\boldsymbol{h}(\mathbf{x}), \mathbf{y}) = \frac{1}{2} \times \sum_{k=1}^{K} (h_k(\mathbf{x}) - y_k)^2$
  - Cross-entropy: $\ell_{CE}(\boldsymbol{h}(\mathbf{x}), \mathbf{y}) = -\sum_{k=1}^{K} y_k \log h_k(\mathbf{x})$

PMC: phase de propagation

- Et, les poids sont corrigés de la sortie à l'entrée en utilisant l'algorithme de la descente de gradient

$$w_{ji} \leftarrow w_{ji} - \eta \frac{\partial \ell(\boldsymbol{h}(\mathbf{x}), \mathbf{y})}{\partial w_{ji}}$$

- En utilisant la règle de chaîne

$$\frac{\partial \ell(\boldsymbol{h}(\mathbf{x}), \mathbf{y})}{\partial w_{ji}} = \underbrace{\frac{\partial \ell(\boldsymbol{h}(\mathbf{x}), \mathbf{y})}{\partial a_j}}_{=\delta_j} \frac{\partial a_j}{\partial w_{ji}}$$

  where $\frac{\partial a_j}{\partial w_{ji}} = z_i$.

- Dans le cas où, la $j^{\text{ème}}$ unité est sur la couche de sortie, nous avons :

$$\delta_j = \frac{\partial \ell(\boldsymbol{h}(\mathbf{x}), \mathbf{y})}{\partial a_j}$$

PMC: phase de propagation

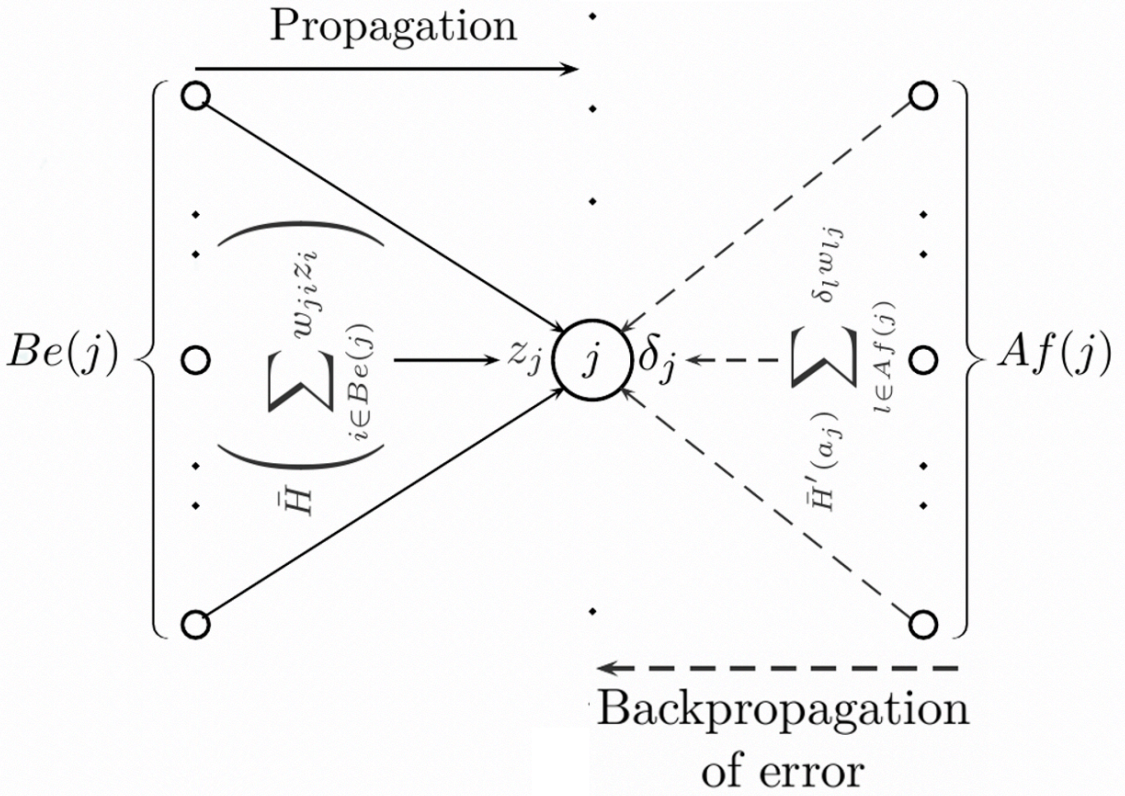
- Si la $j^{\text{ème}}$ unité est sur la couche cachée, nous avons en appliquant la règle de la chaine à nouveau :

$$\delta_j = \frac{\partial \ell(\boldsymbol{h}(\mathbf{x}), \mathbf{y})}{\partial a_j} = \sum_{l \in Af(j)} \frac{\partial \ell(\boldsymbol{h}(\mathbf{x}), \mathbf{y})}{\partial a_l} \frac{\partial a_l}{a_j}$$

$$= \bar{H}'(a_j) \sum_{l \in Af(j)} \delta_l \times w_{lj}$$

Où $Af(j)$ est l'ensemble des unités qui sont sur la couche succédant celle contenant l'unité j.

Backpropagation [Hinton & Rumelhart, 1986]



Référence

D. E. Rumelhart, G. E. Hinton and R. Williams
Learning internal representations by error propagation.
Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1986