Majeure Machine Learning

Deep Learning Introduction

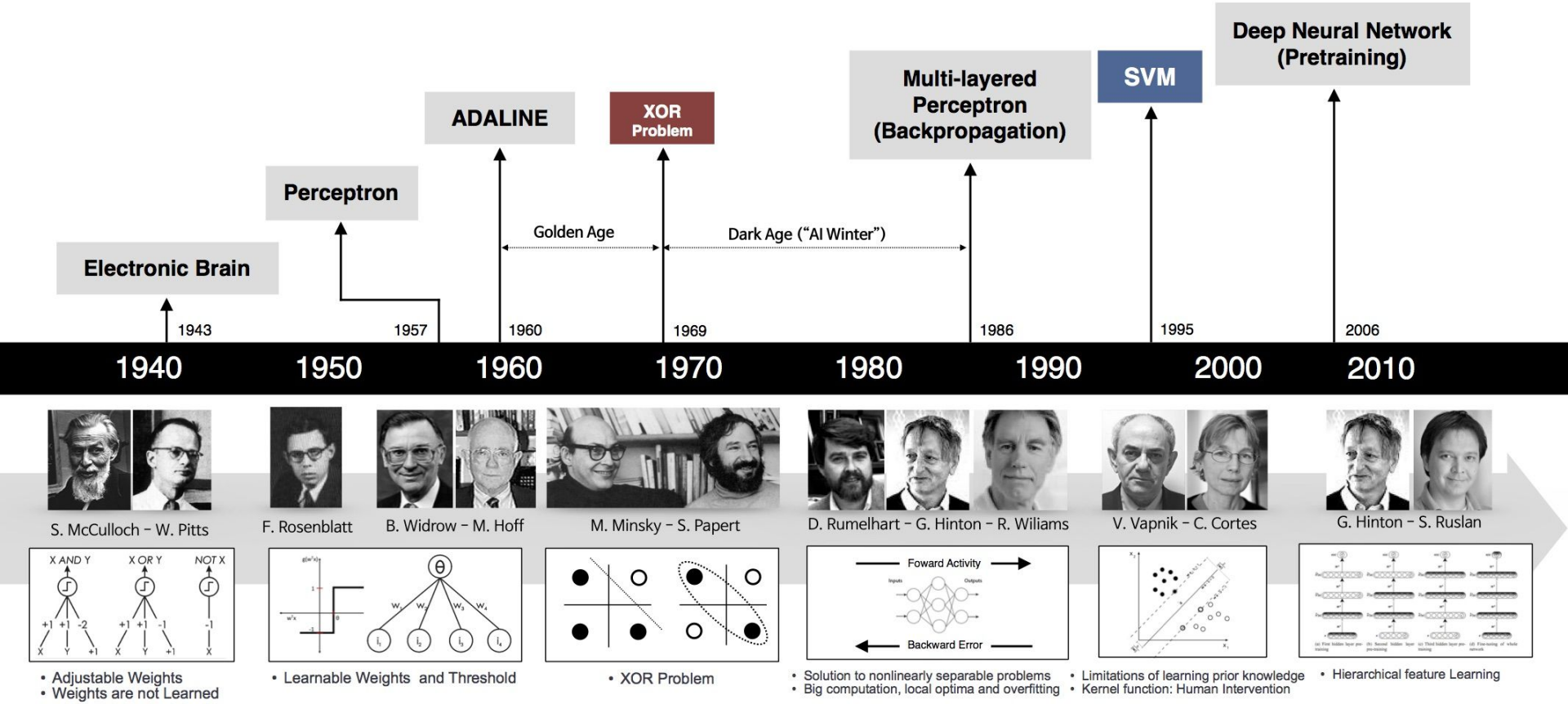
Contenu

- Principe de Deep Learning
- Réseau de neurones
- Fonctions d’activation
- Backpropagation

Ce que vous devrez savoir faire

- Comprendre ce qu’est le Deep Learning
- Comprendre ce qu’est un neurone artificiel
- Comprendre ce qu’est un réseau de neurones
- Comprendre l’intuition de la Backpropagation

Deep Learning



- Adjustable Weights
- Weights are not Learned

- Learnable Weights and Threshold

- XOR Problem

- Solution to nonlinearly separable problems
- Big computation, local optima and overfitting

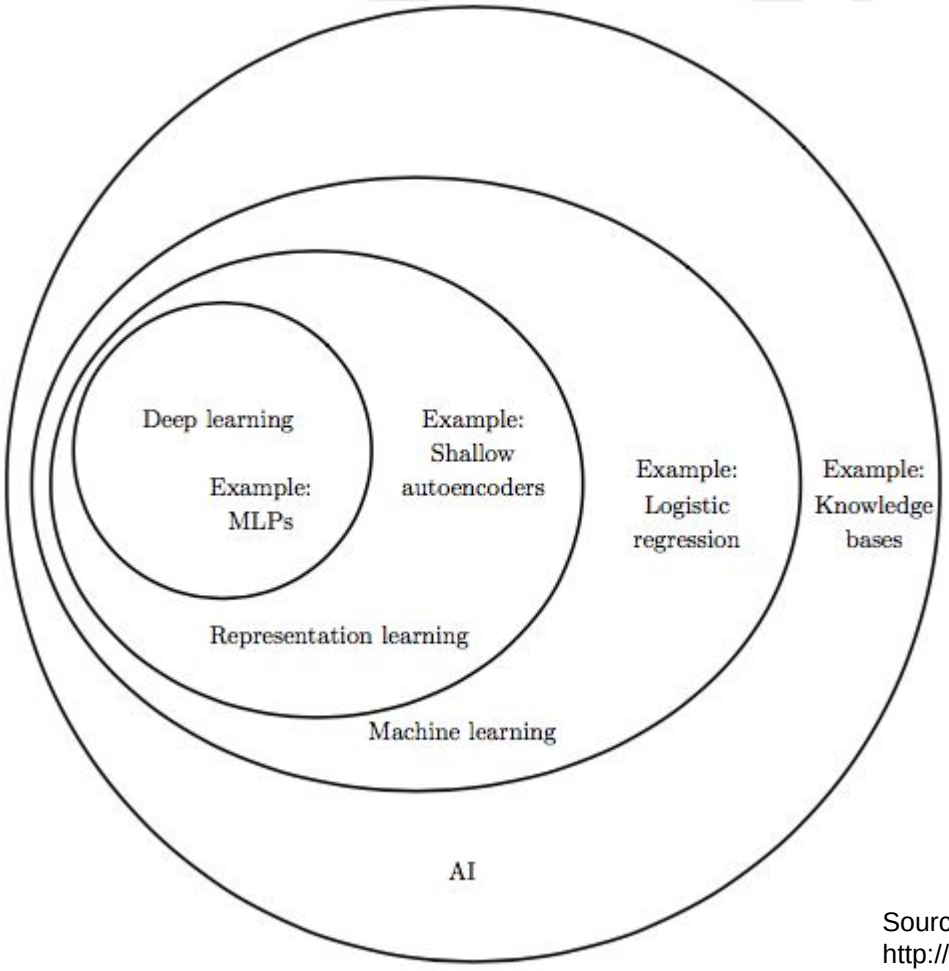
- Limitations of learning prior knowledge
- Kernel function: Human Intervention

- Hierarchical feature Learning

Définition

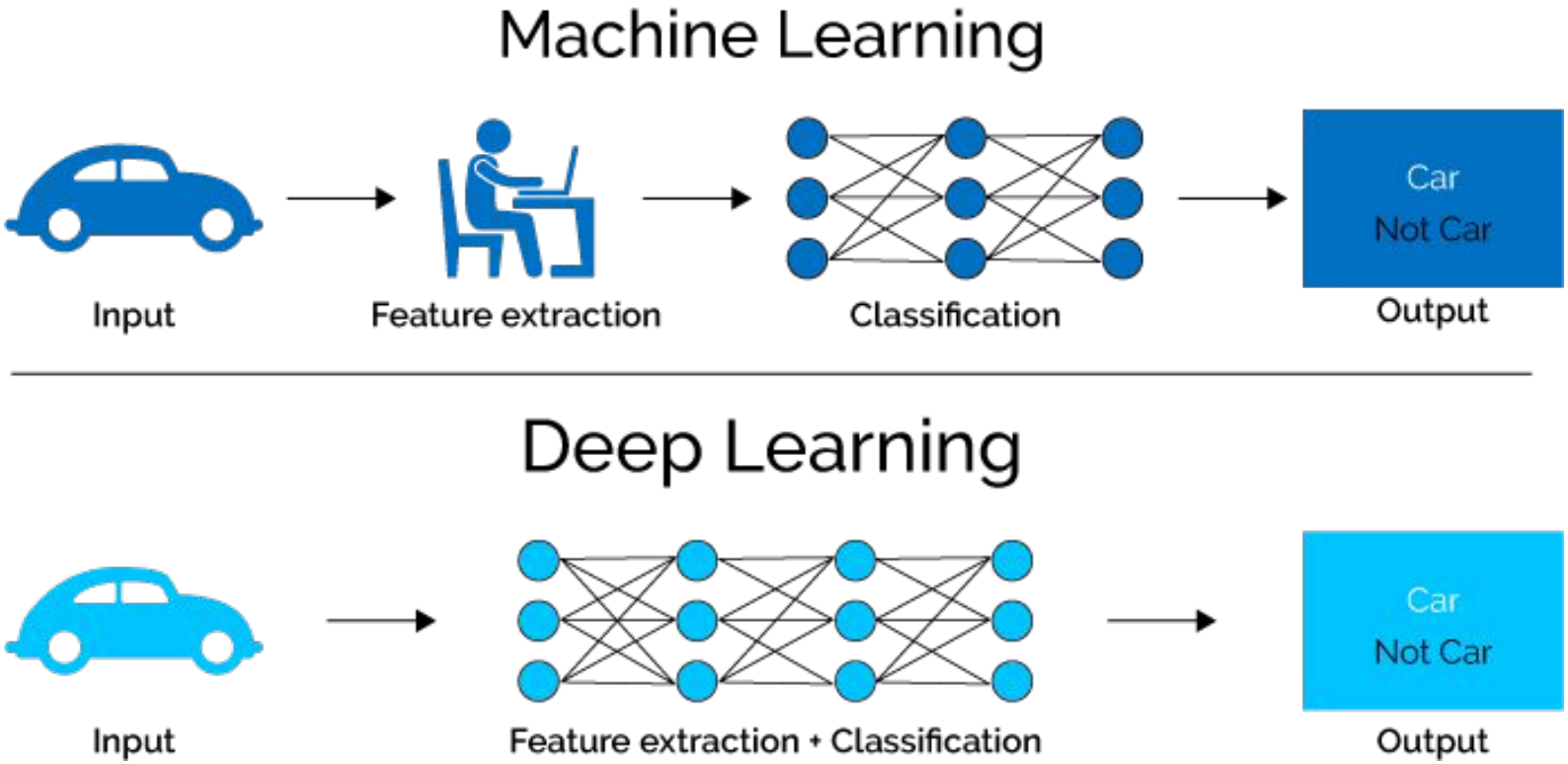
“L'apprentissage profond est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d’abstraction des données grâce à des architectures articulées de différentes transformations non linéaires.”
Wikipedia

Définition

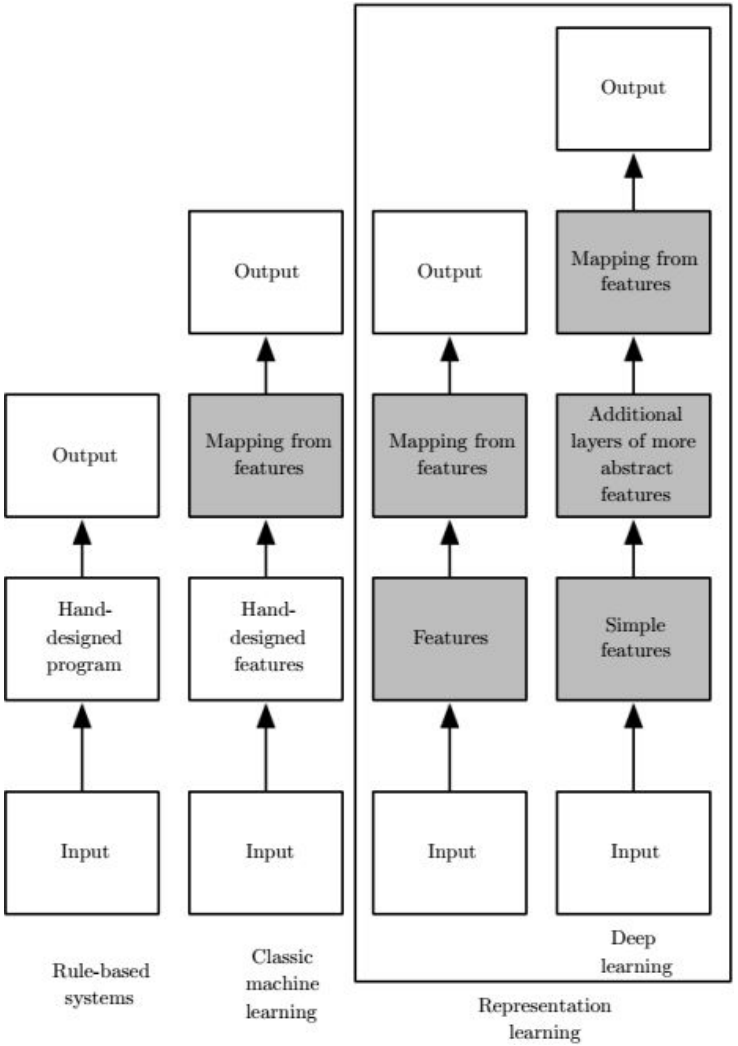


Source :
http://www.deeplearningbook.org

Définition

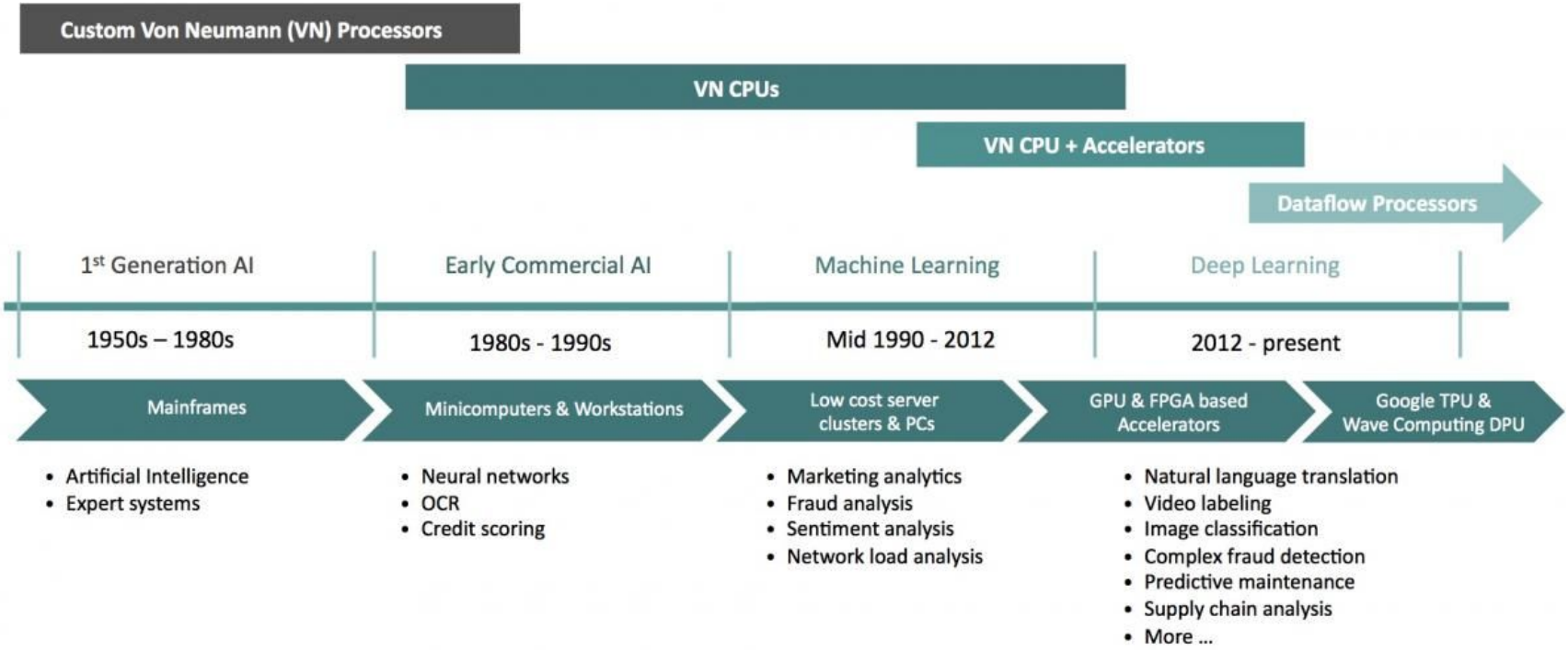

Machine Learning
Input
Feature extraction
Classification
Car
Not Car
Output
Deep Learning
Input
Feature extraction + Classification
Car
Not Car
Output

Définition

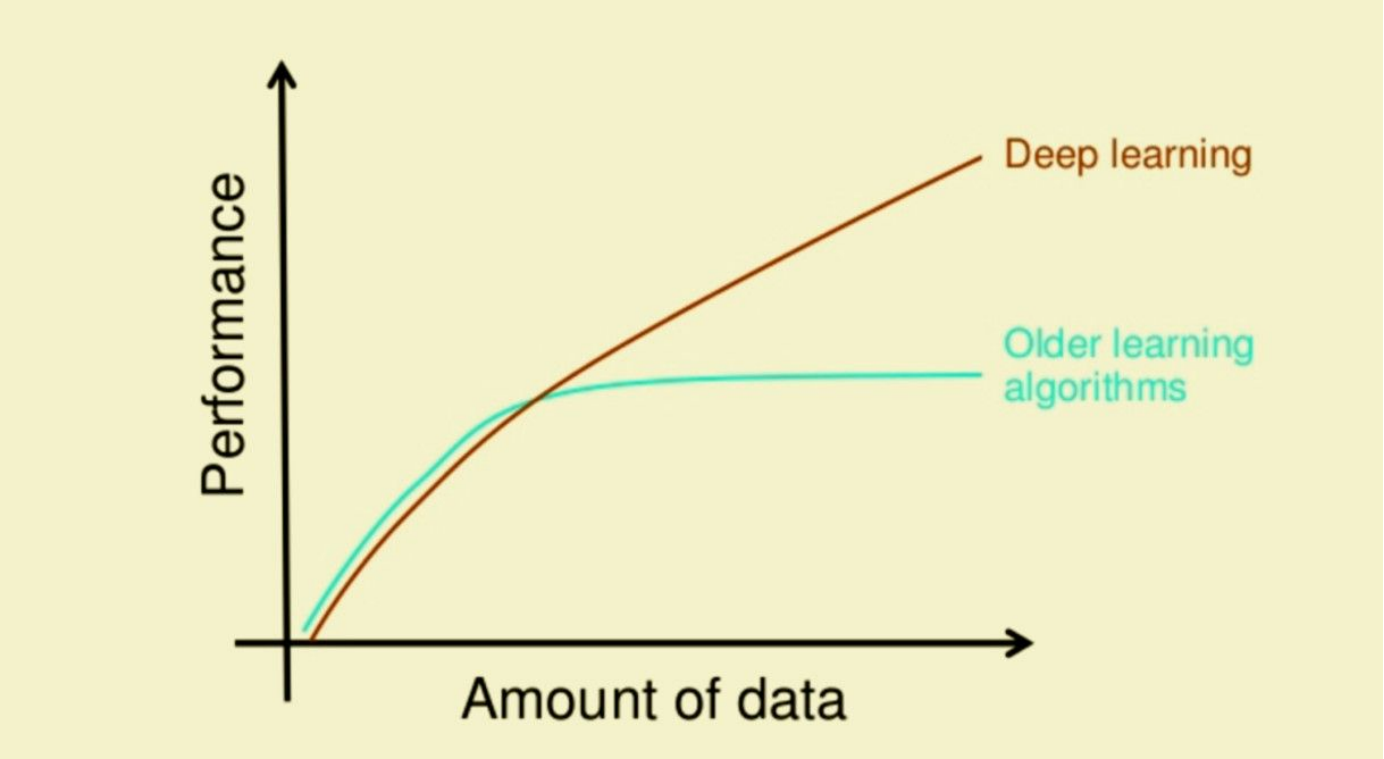


Source : http://www.deeplearningbook.org

Pourquoi un tel essor depuis 2010

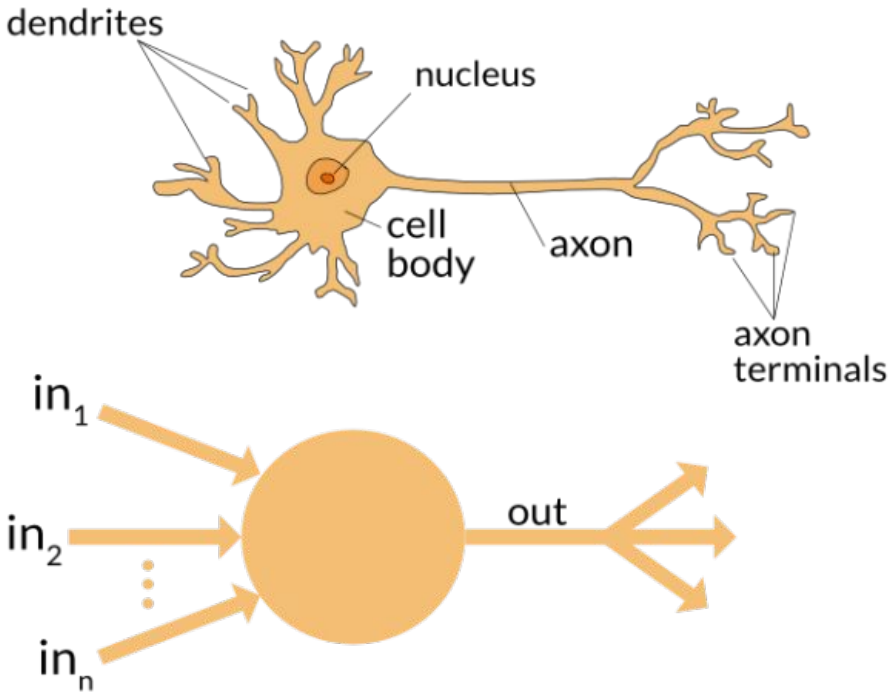


Le deep learning aujourd'hui



Les réseaux de neurones

Une inspiration biologique



Le neurone artificiel (Perceptron)

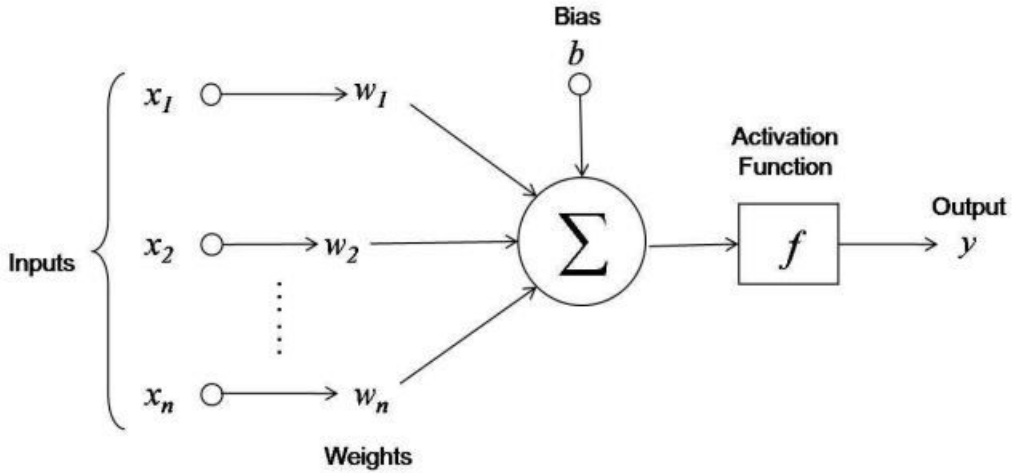
Pré-activation :

$$a(x) = b + \sum_i w_i x_i$$

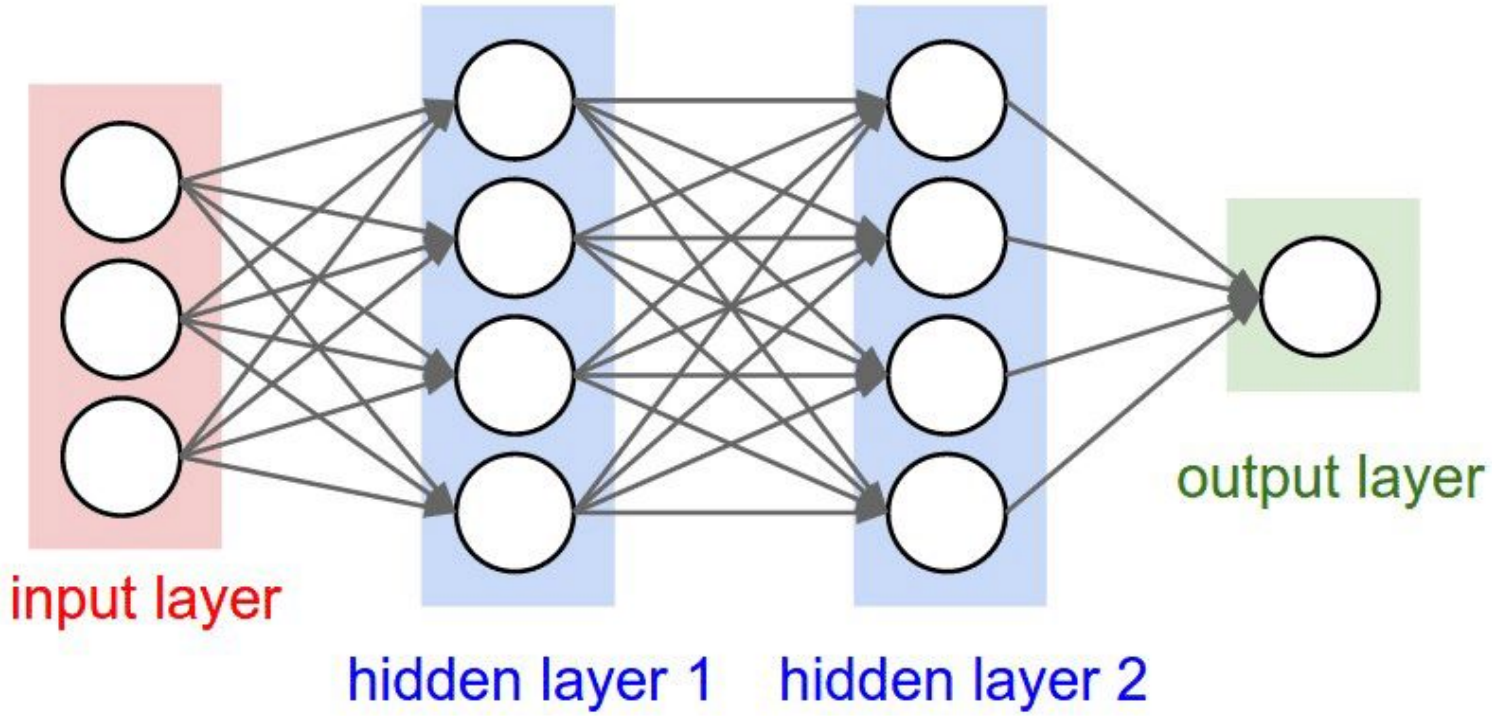
Activation (output) :

$$h(x) = g(a(x)) = g(b + \sum_i w_i x_i)$$

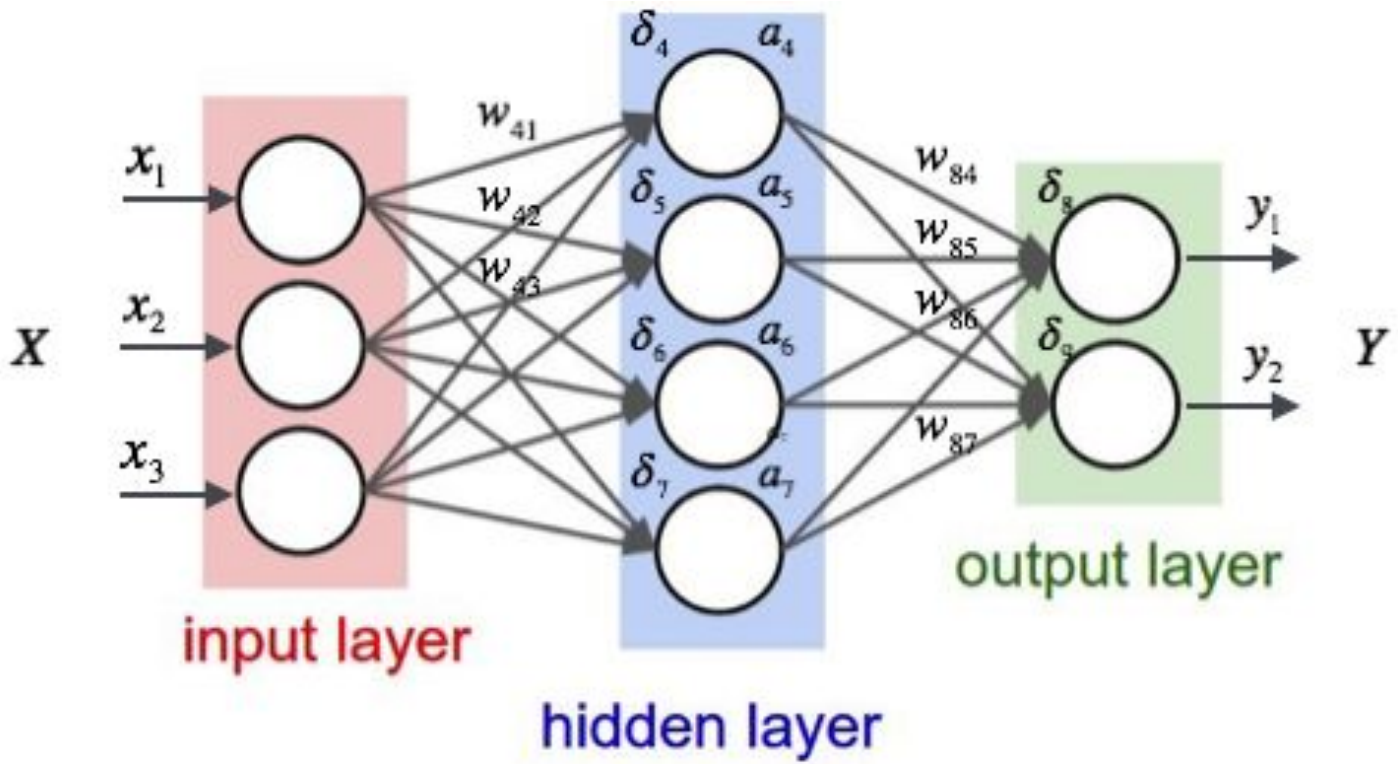
- **w** => poids
- **b** => biais
- **g(.)** => fonction d'activation



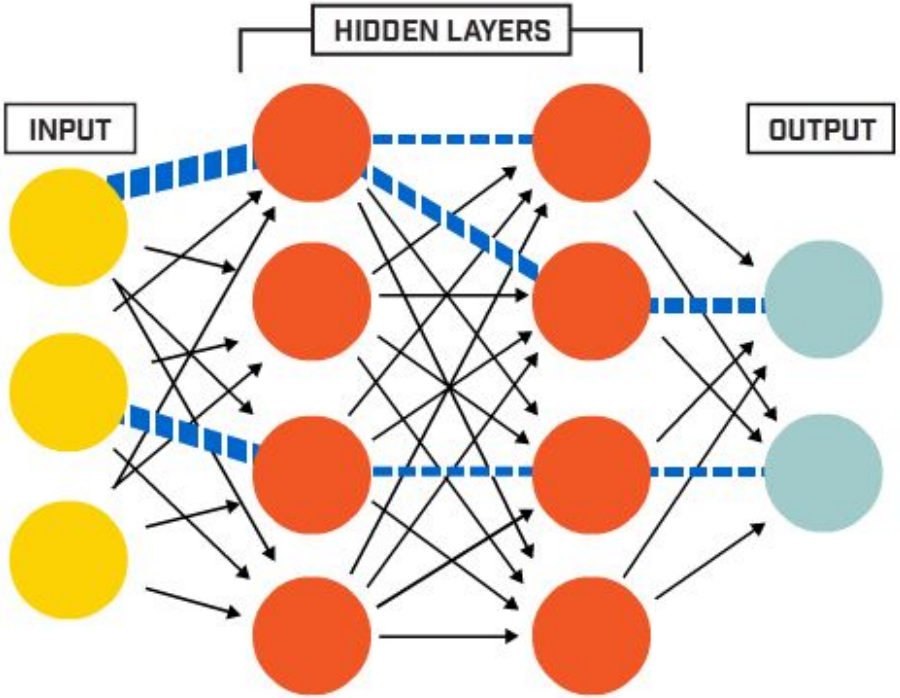
Feed Forward Neural Network (Multi Layer Perceptron)



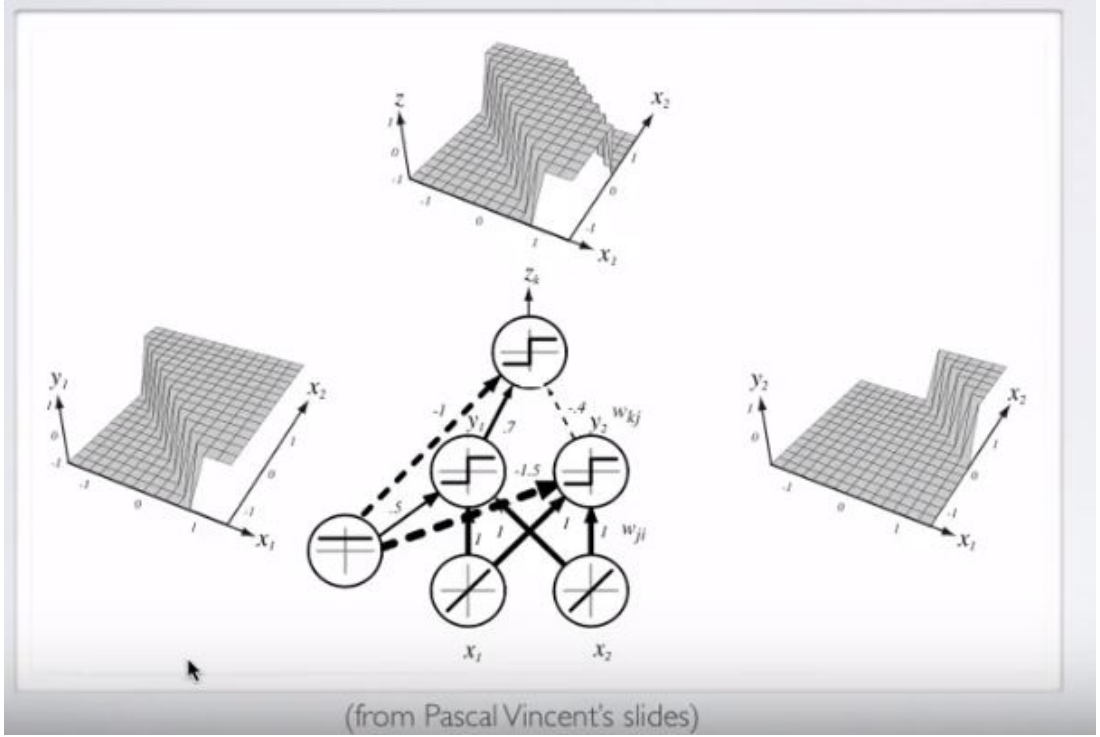
Feed Forward Neural Network



Feed Forward Neural Network



Intuition de capacité



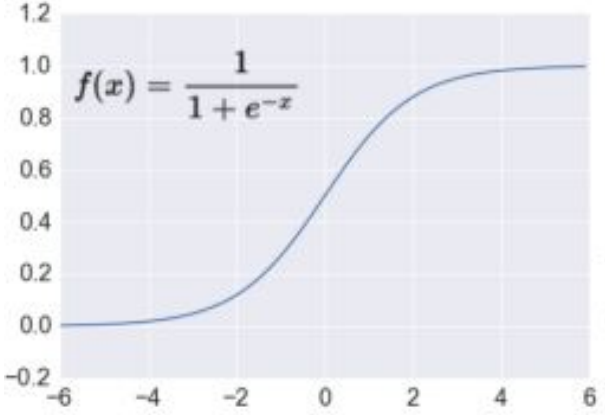
(from Pascal Vincent's slides)

Démo - Playground

DEMO

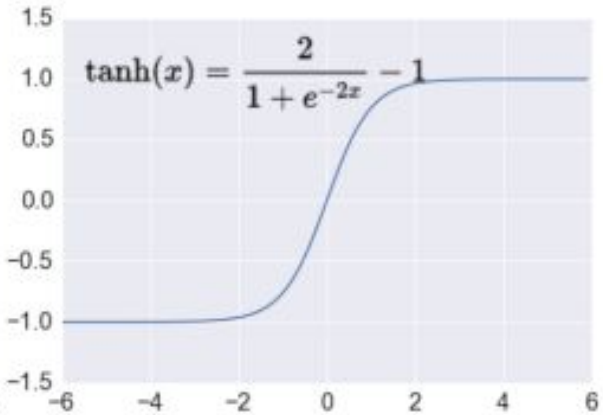
Fonctions d'activation - Couches cachées

Sigmoid



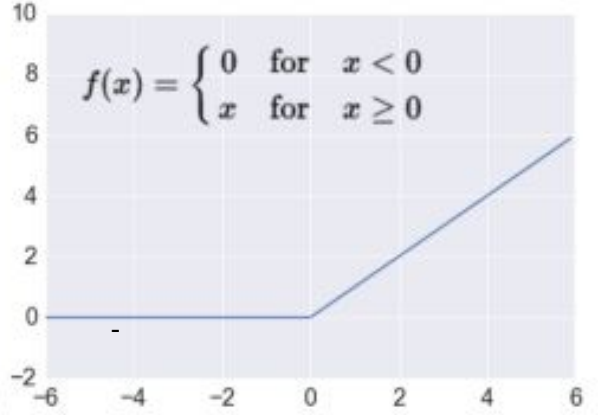
- Très utilisé jusqu'à il y a quelques années
- Problème : Gradient Vanishing

TanH



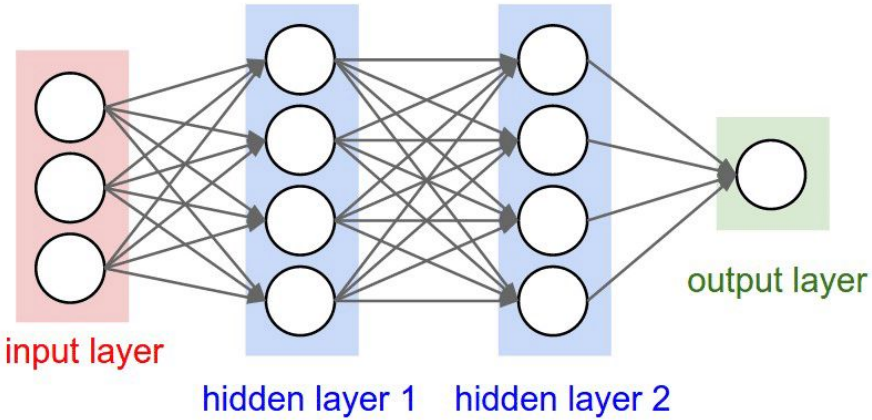
- Très utilisé dans les RNNs
- Même problème que la Sigmoid

ReLU



- Référence actuelle pour les FeedForwards
- Sparse (peut être égale à 0) => bien pour l'optimisation et le stockage
- Plus de problème de Gradient Vanishing sur la partie positive
- Problème de "Diying"

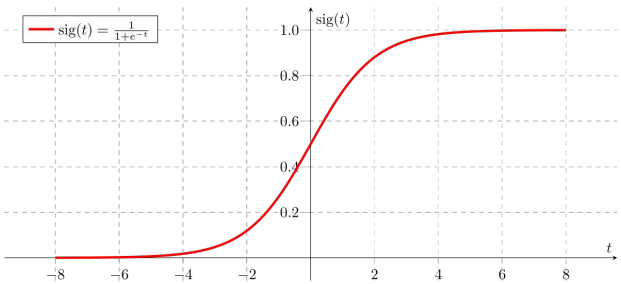
Fonctions d'activation (Régression) - Output



$$o(x) = b + \sum_i w_i h_i^2$$

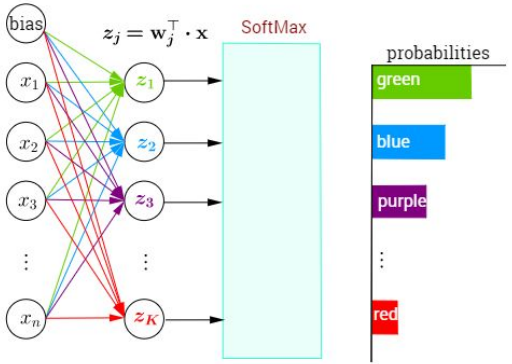
Fonctions d'activation (Classification) - Output

Binaire



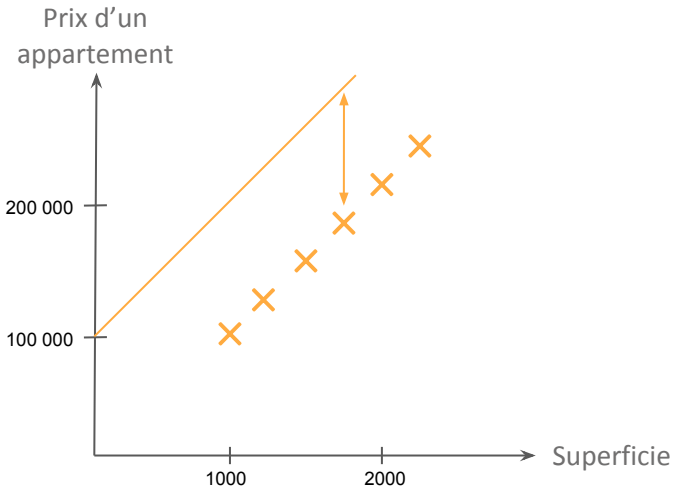
Sigmoid : $\sigma(x) = \frac{1}{1+e^{-x}}$

Multi-class



Softmax : $p(y = j|\mathbf{x}) = \frac{e^{(\mathbf{w}_j^T \mathbf{x} + b_j)}}{\sum_{k \in K} e^{(\mathbf{w}_k^T \mathbf{x} + b_k)}}$

Cost Function - Rappel



$$J(\theta) = 1/N \sum_{n=1}^{N} (\hat{y}(x_n, \theta) - y(x_n))^2$$

=> Objectif : Minimiser la Cost function

Fonction Coût : Régression

$$J(\theta) = 1/N \sum_{n=1}^{N} (\hat{y}(x_n, \theta) - y(x_n))^2$$

Fonction Coût : Classification

Cross Entropy :

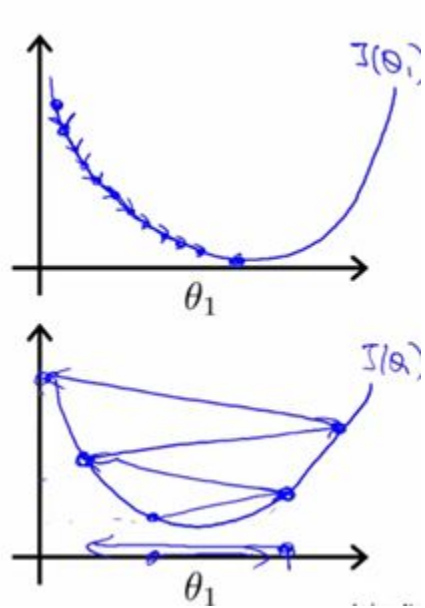
$$C(\theta) = \sum_{i=1}^{n} y^i log(\hat{y}^i) + (1 - y^i) log(1 - \hat{y}^i)$$

=> Objectif : Minimiser la Cost function

Descente de gradients - rappel

$$J(\theta) = 1/N \sum_{n=1}^{N} (\hat{y}(x_n, \theta) - y(x_n))^2$$

Dérivée partielle : $\frac{\partial J(\theta)}{\partial \theta_1} = 1/N \sum_{n=1}^{N} (\hat{y}(x_n, \theta) - y(x_n)) x_n$



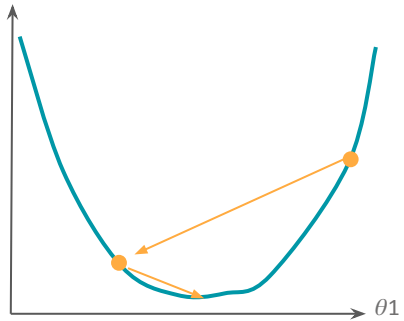
Pour i allant de 1 à nombre_choisi :

$$\theta_1 = \theta_1 - \alpha \frac{\partial J(\theta)}{\partial \theta_1}$$

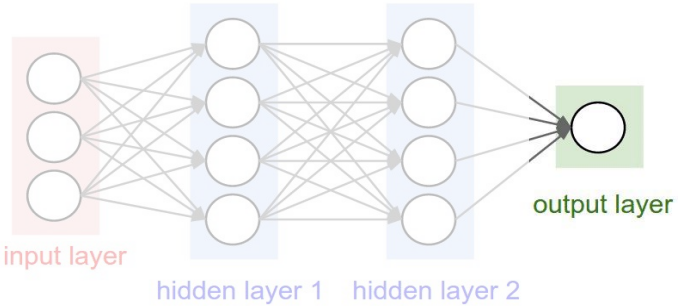
Avec α > 0, le pas d'avancement

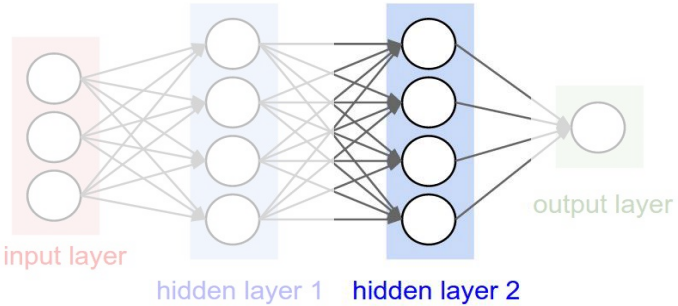
$\theta 1 = 5$
Dérivé partielle = 3
=> $\theta 1 = 5 - 1*3 = 2$

$\theta 1 = 2$
Dérivé partielle = -1
=> $\theta 1 = 2 - 1*(-1) = 2 + 1 = 3$



Descente de gradients - problématique des réseaux de neurones



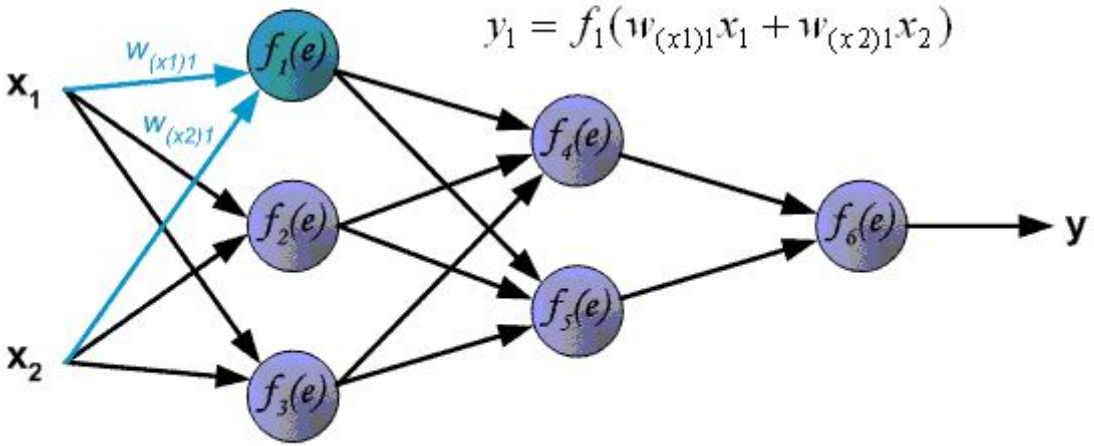


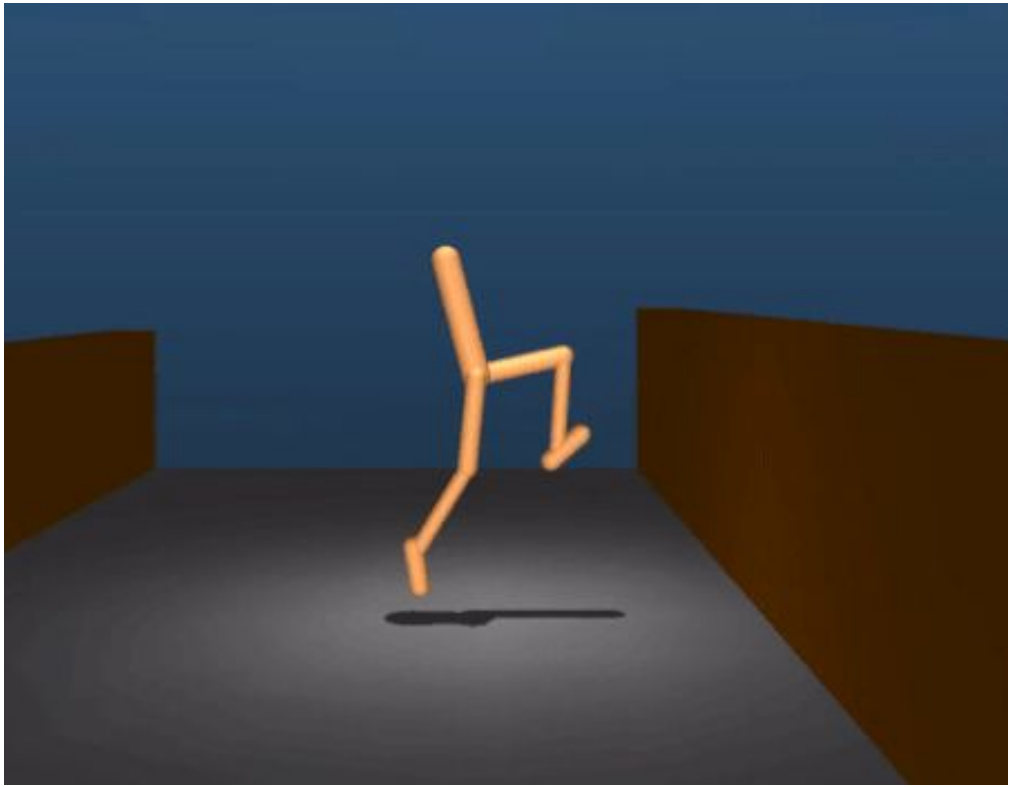
$$\hat{y}(x) - y(x)$$

=> Comment calculer les erreurs des couches précédentes ?

Backpropagation

FP





Fin du chapitre 5.1