Réseaux de neurones

ENS'IA

Ensimag 2019-2020

11 mars 2020

ENS'IA

Qui sommes nous ?

- Association fondée en mai 2019
- Promouvoir l'intelligence artificielle et son apprentissage
- Partager les connaissances entre élèves

ENS'IA

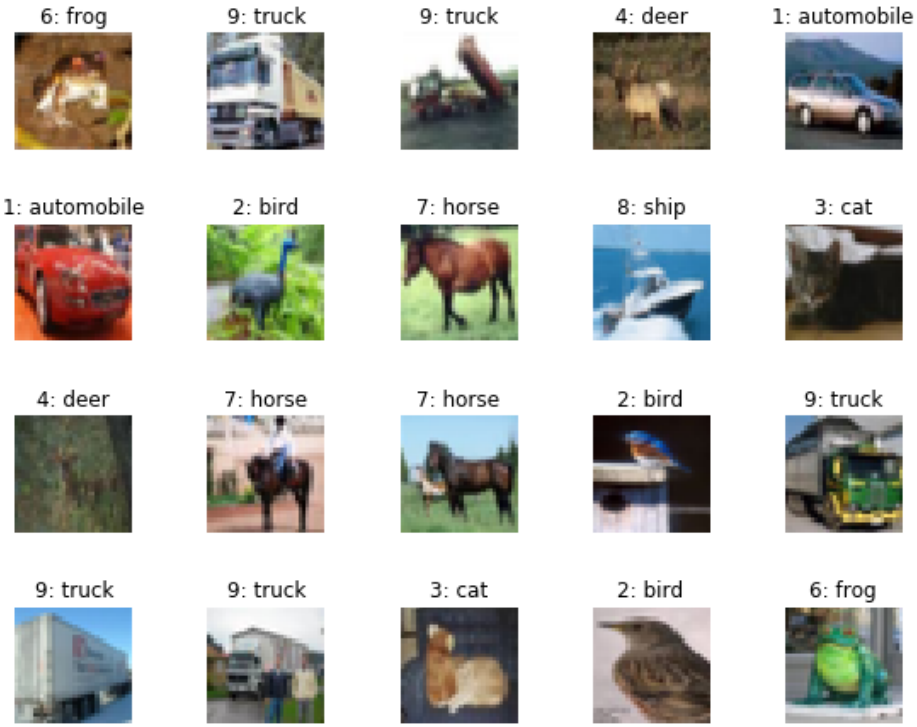
Qui sommes nous ?

- Association fondée en mai 2019
- Promouvoir l'intelligence artificielle et son apprentissage
- Partager les connaissances entre élèves

Les membres :

- Clément Domps (MOSIG) - Président
- Lucas Sort (MMIS) - Vice président
- Joana Lemercier (IF) - Secrétaire générale
- Alexandre Audibert (MMIS) - Comptable

Rappel

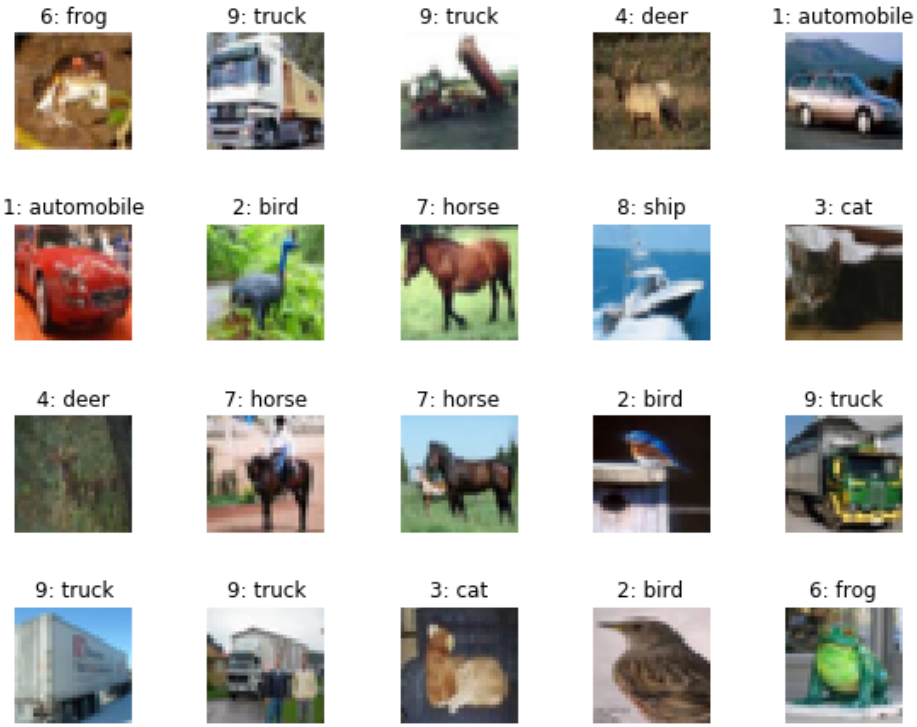


Classification d'images

$\rightarrow$ Supervised learning : x et y connus

$\rightarrow$ Unsupervised learning : que x

Rappel



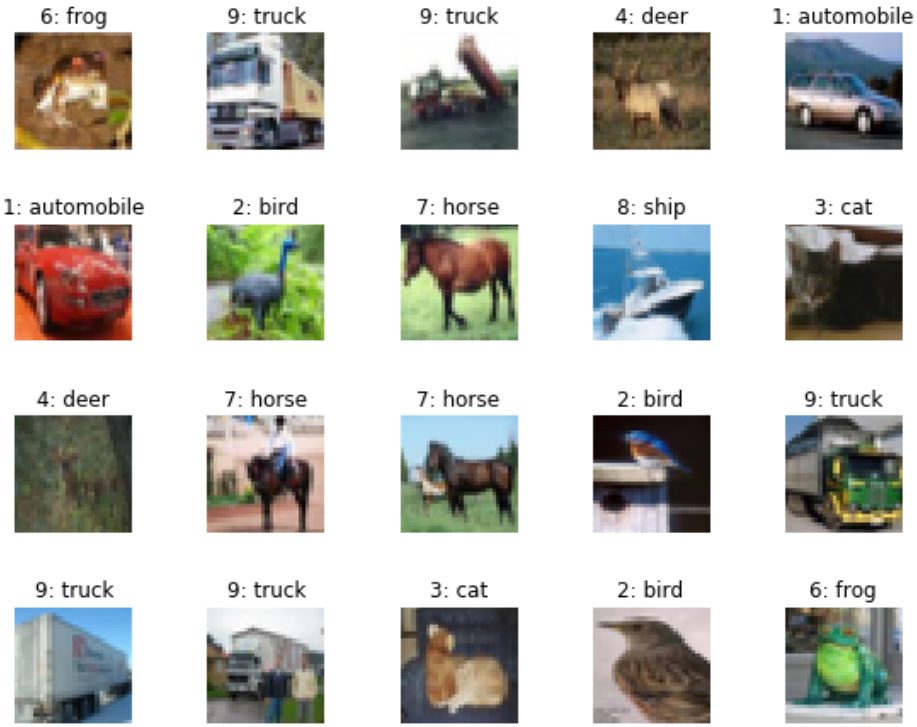
Classification d'images

→ Supervised learning : x et y connus

→ Unsupervised learning : que x

Performances ?

Rappel



Classification d'images
→ Supervised learning : x et y connus
→ Unsupervised learning : que x

Performances ? Bof.

But

$$x \to y$$

But

$$f(x) = y$$

But

$$f(x) = y$$

Comment trouver f ?

But

$$f(x) = y$$

Comment trouver f ?

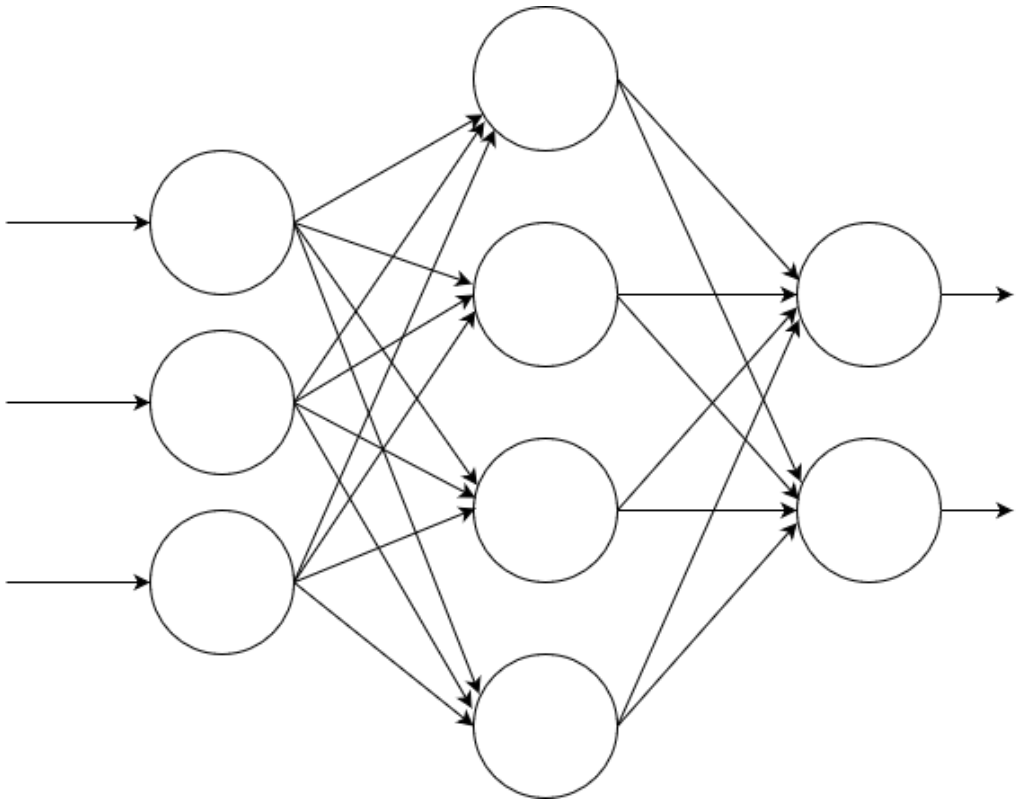
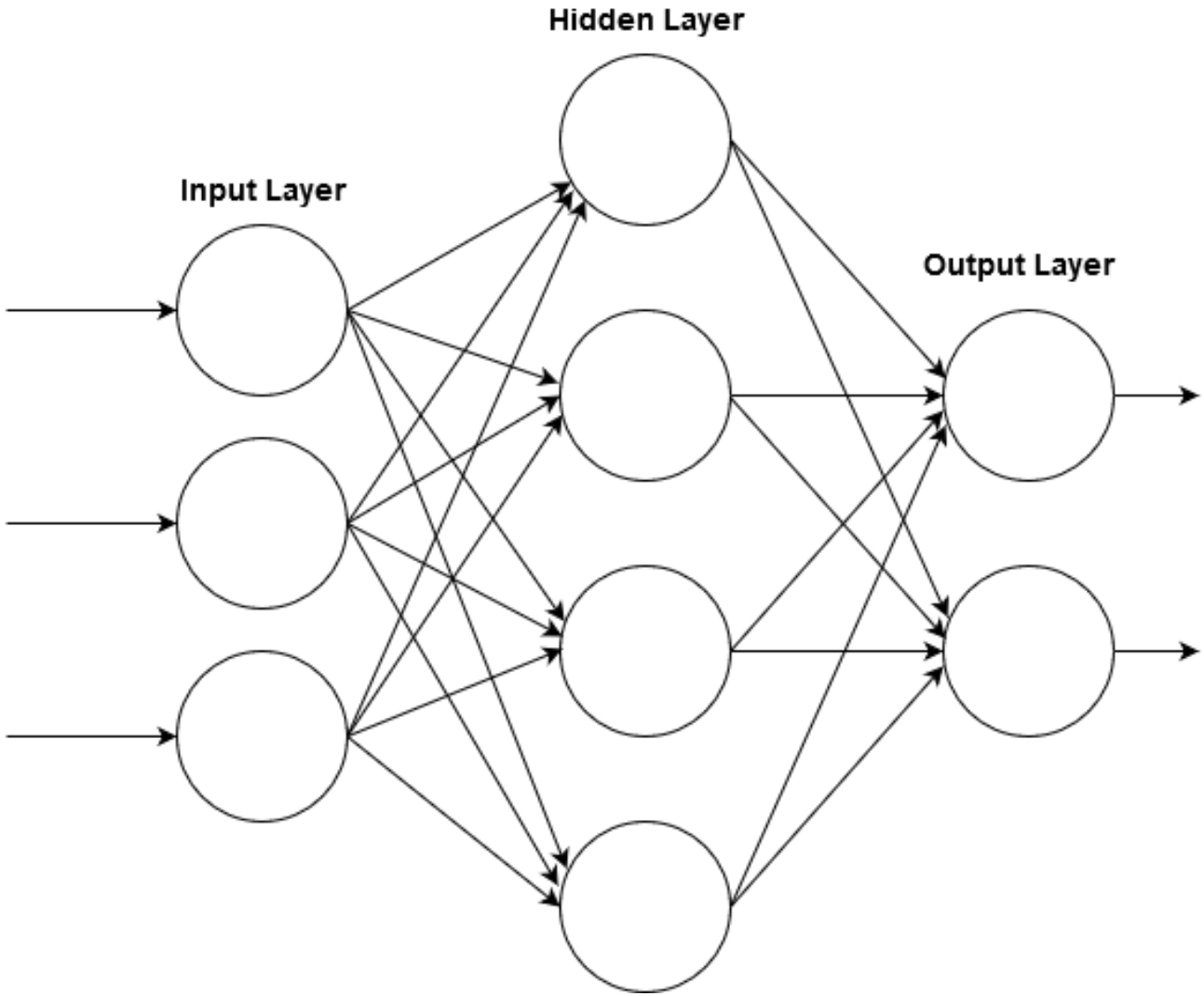


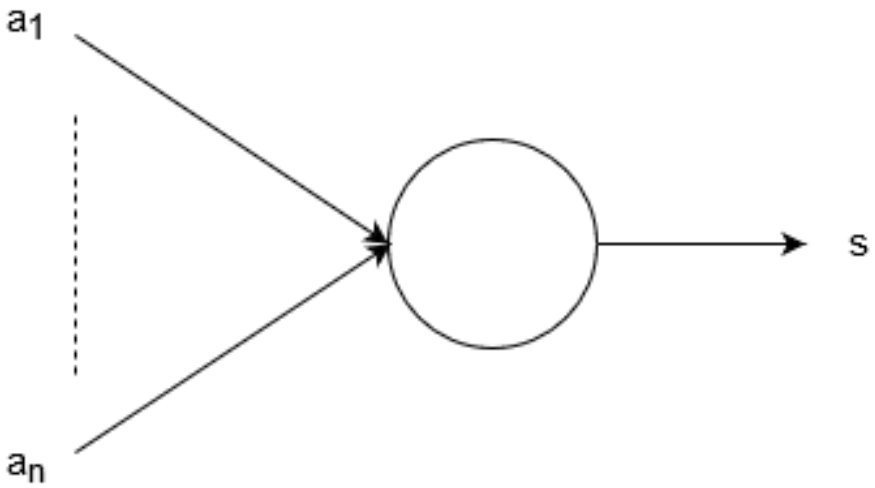
FIGURE 1 – Réseau de neurones

Réseau de neurones



→ Succession de couches de neurones

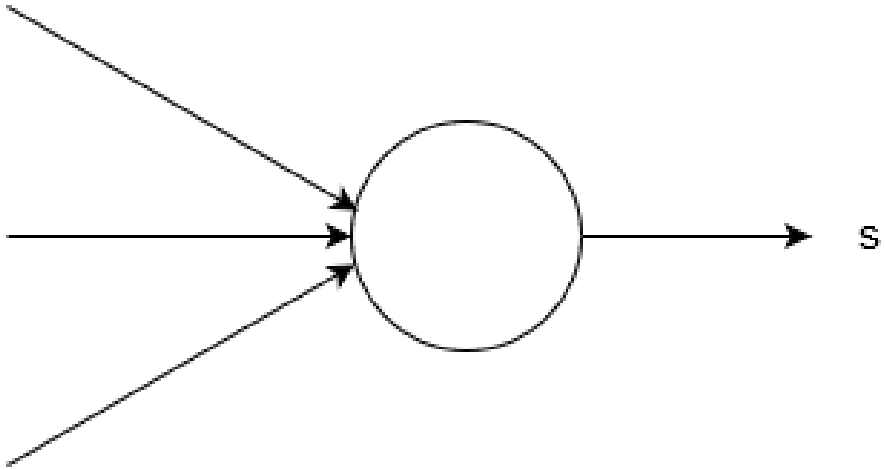
Perceptron



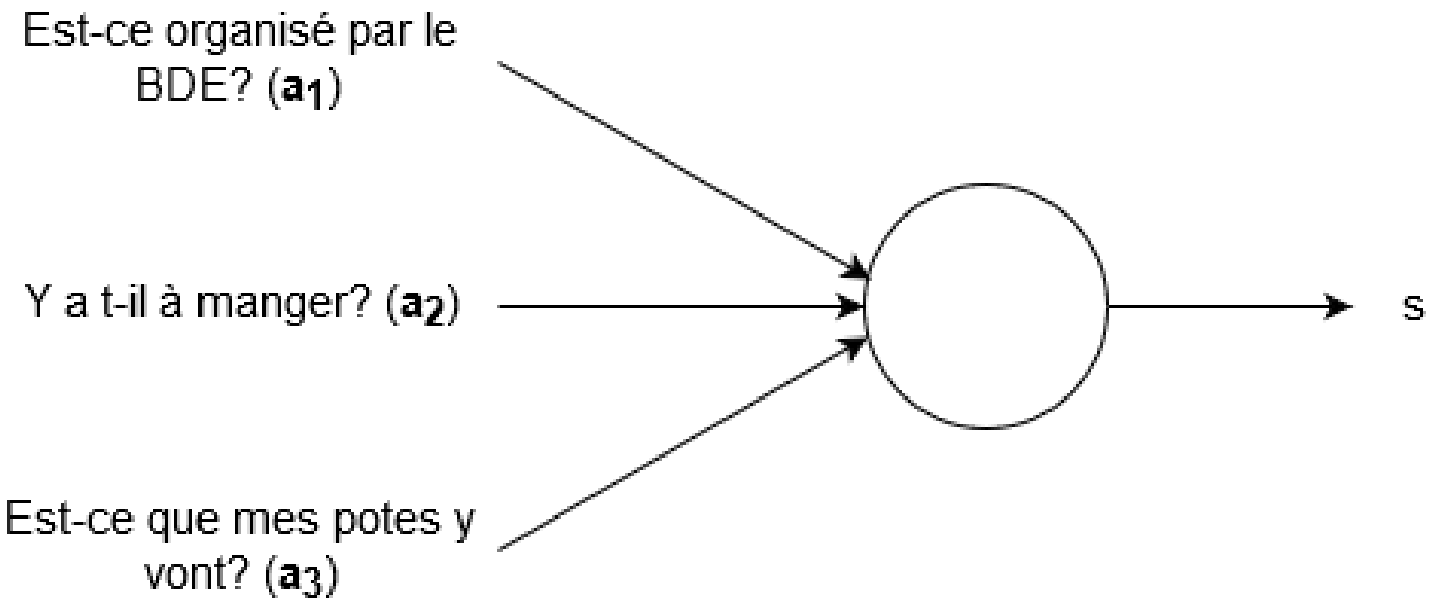
$a_1, ..., a_n, s \in 0, 1$

$$s = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} a_i * w_i + b > 0 \\ 0 & \text{sinon.} \end{cases}$$

Perceptron - Exemple

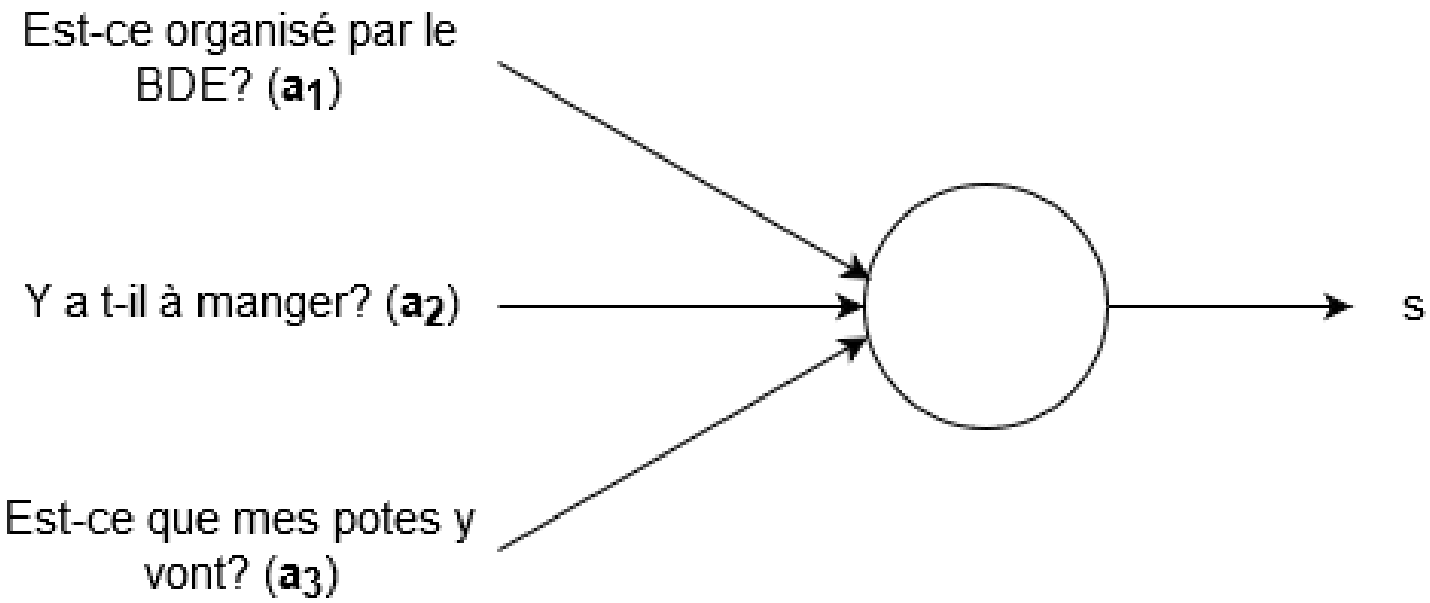


Perceptron - Exemple



$$\text{Si } a_1 + a_2 + a_3 = 3, s = 1$$

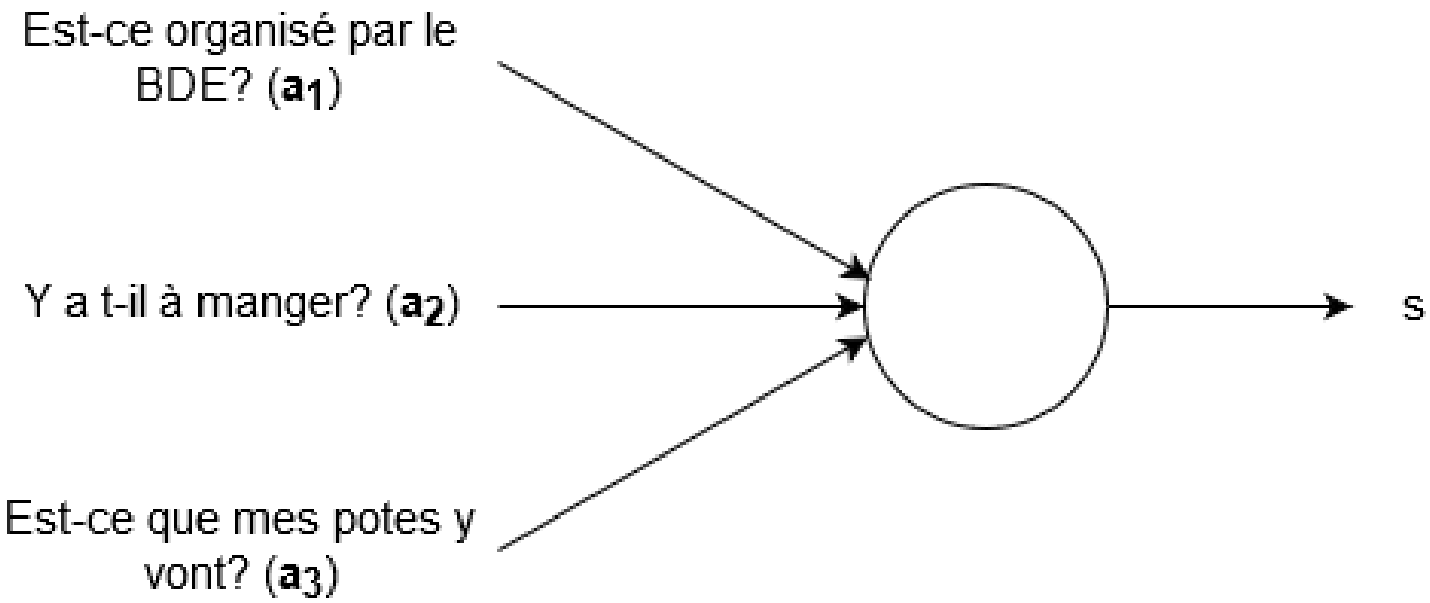
Perceptron - Exemple



Si $a_1 + a_2 + a_3 = 3, s = 1$

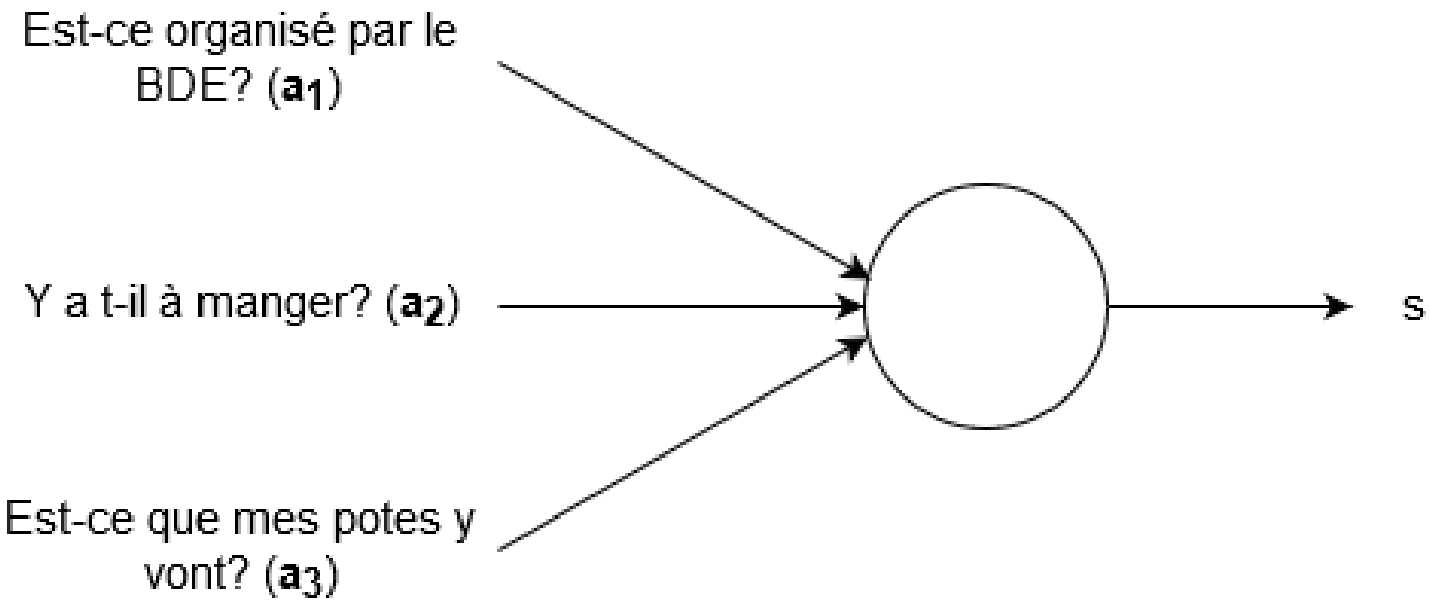
On va à la soirée !

Perceptron - Exemple



$$\text{Si } a_1 + a_2 + a_3 > threshold, s = 1$$

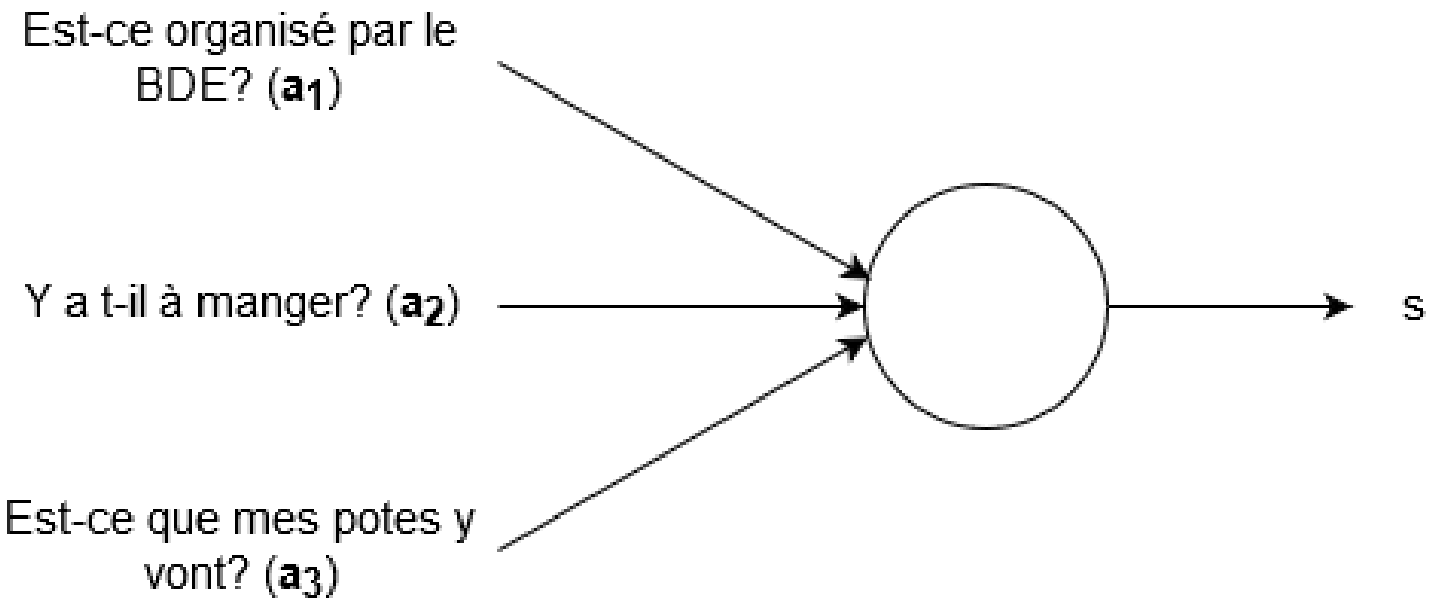
Perceptron - Exemple



Si $a_1 + a_2 + a_3 > threshold, s = 1$

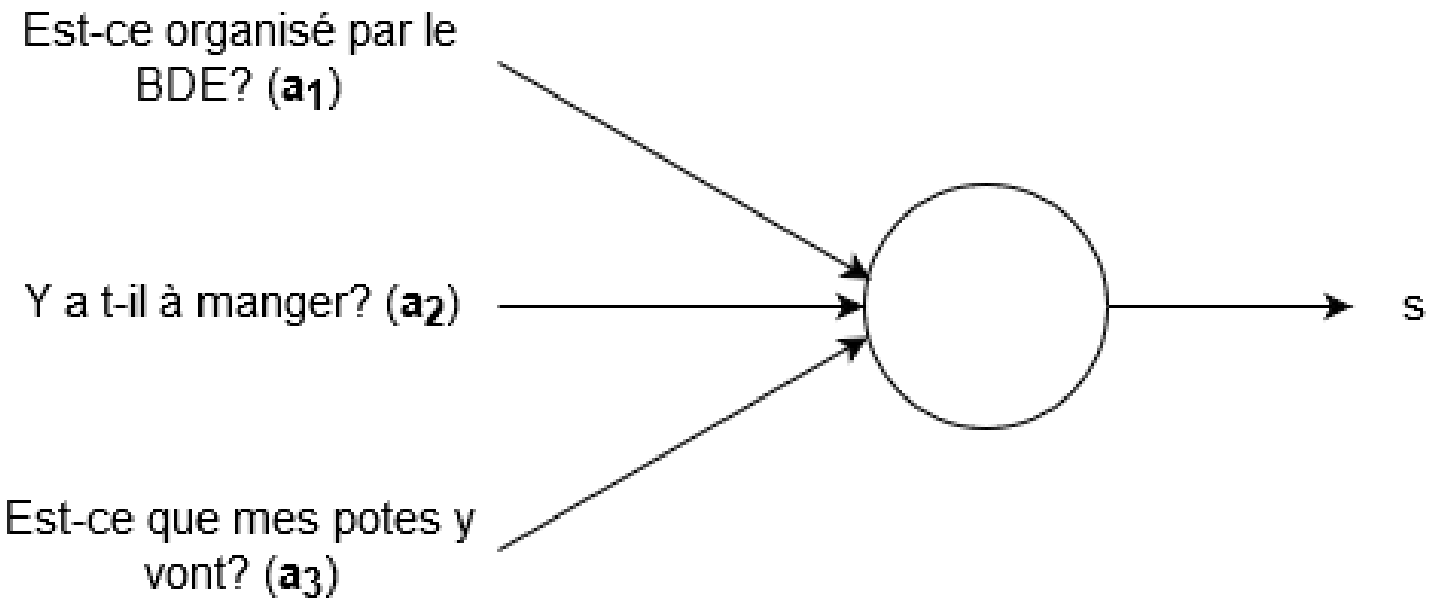
On va à la soirée !

Perceptron - Exemple



$$\text{Si } a_1 * w_1 + a_2 * w_2 + a_3 * w_3 + b > 0, s = 1$$

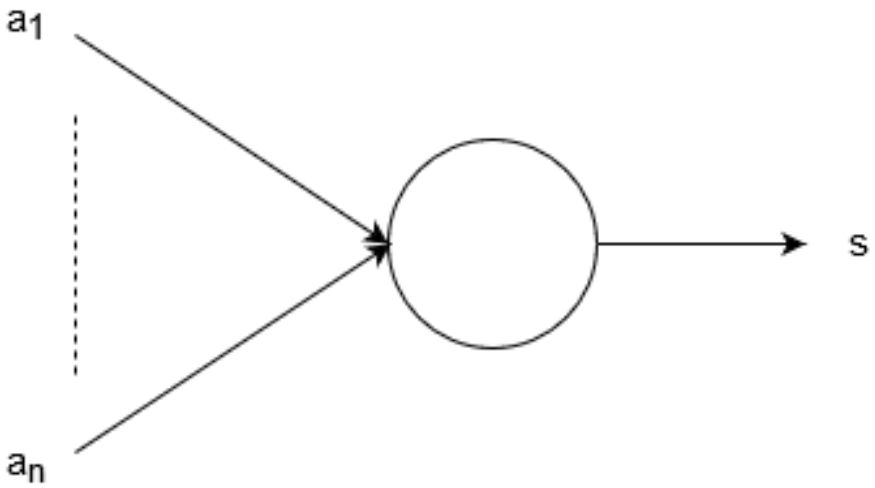
Perceptron - Exemple



$$\text{Si } a_1 * w_1 + a_2 * w_2 + a_3 * w_3 + b > 0, s = 1$$

On va à la soirée !

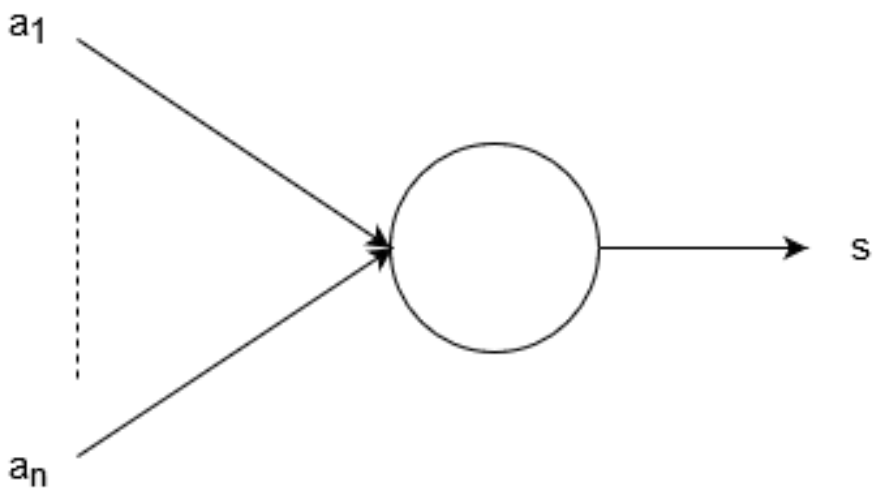
Perceptron



$a_1, ..., a_n, s \in 0, 1$

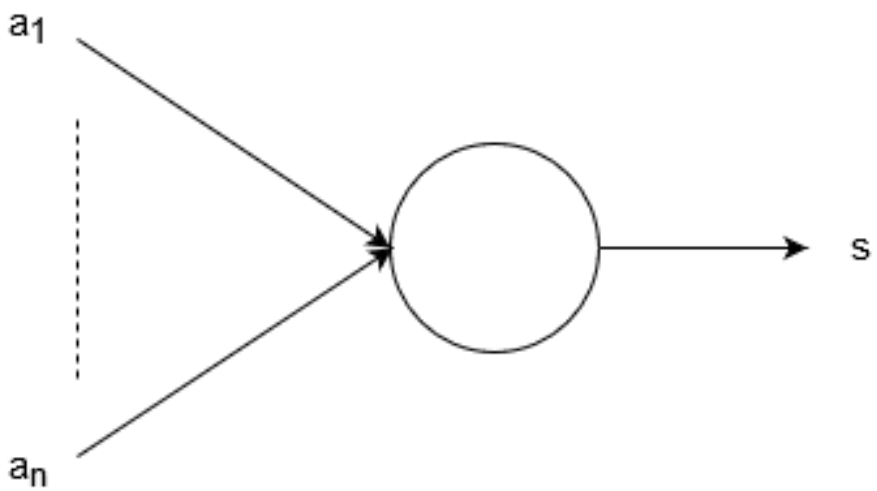
$$s = \begin{cases} 1 & \text{si } \sum_{i=0}^{n} a_i * w_i + b > 0 \\ 0 & \text{sinon.} \end{cases}$$

Perceptron



→ Capable de reproduire des portes logiques !
→ Trouver les w et b à la main c'est pénible

Perceptron

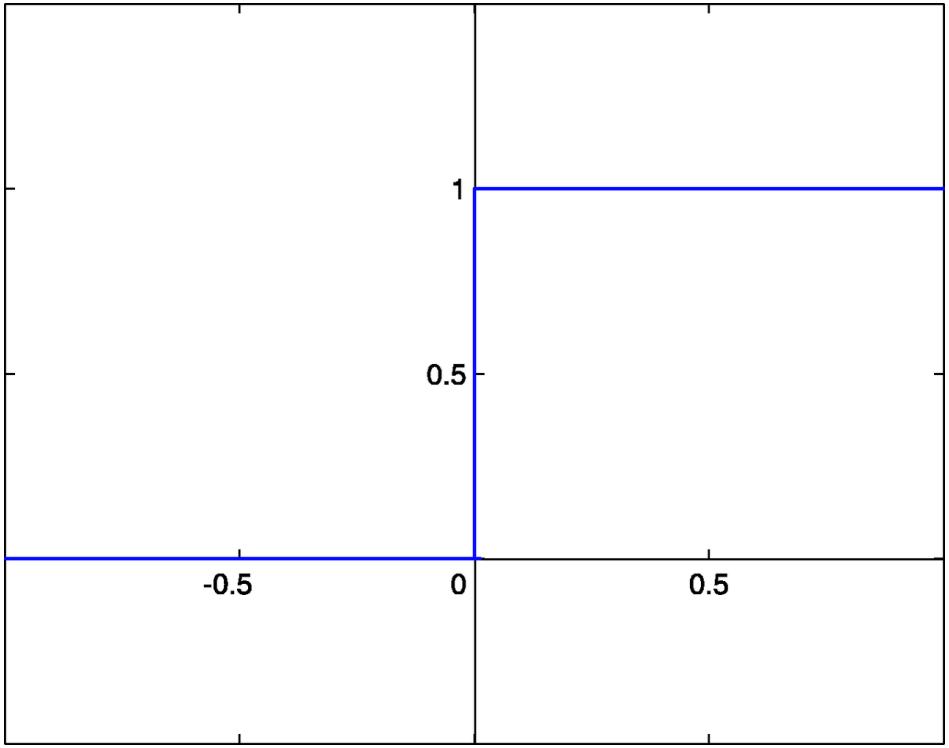


→ Capable de reproduire des portes logiques !
→ Trouver les w et b à la main c'est pénible

Il faut "**apprendre**" les w et les b.

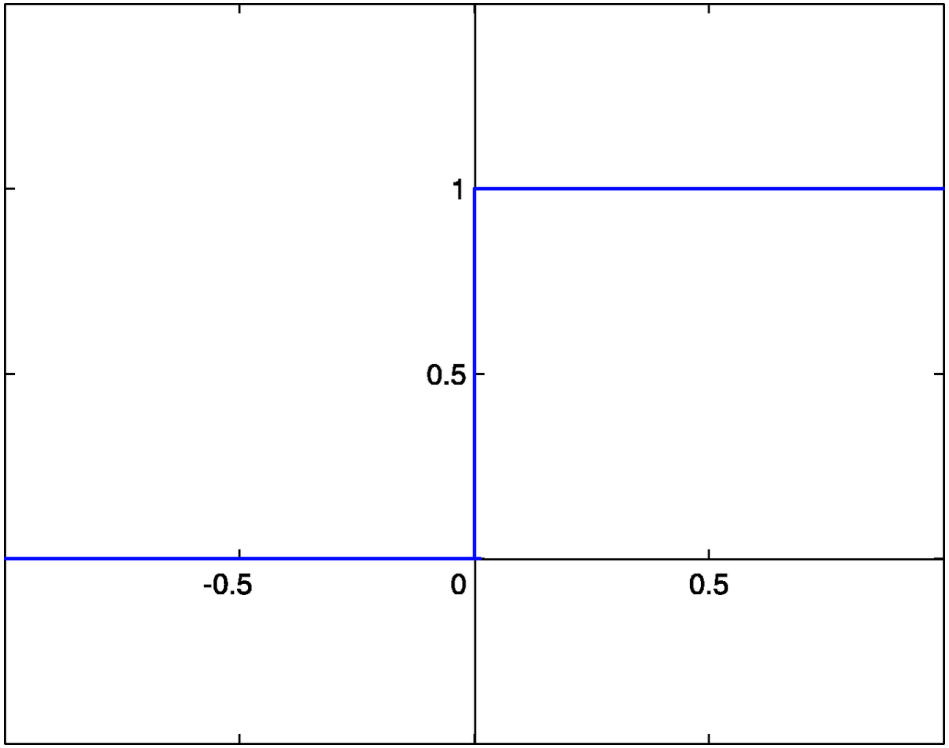
Perceptron

Comment apprendre ?
Petit changement des w et des b $\rightarrow$ petit changement de la sortie



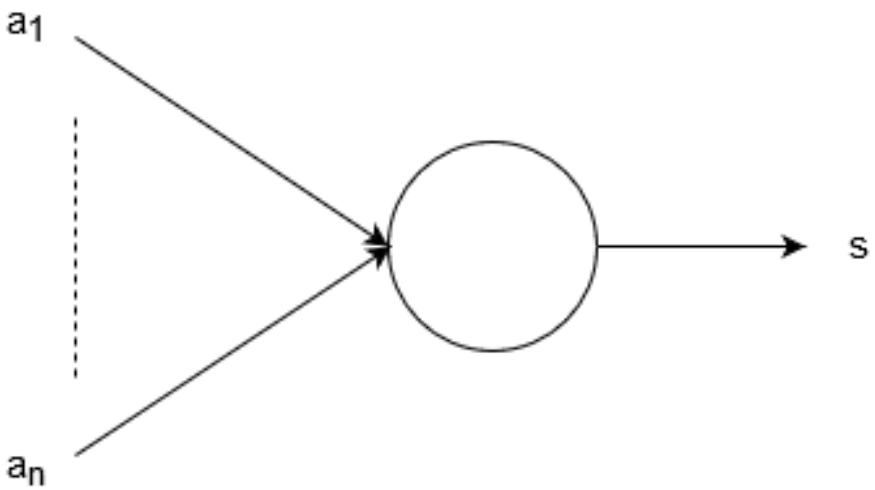
Perceptron

Comment apprendre ?
Petit changement des w et des b $\rightarrow$ petit changement de la sortie



Pas possible ici

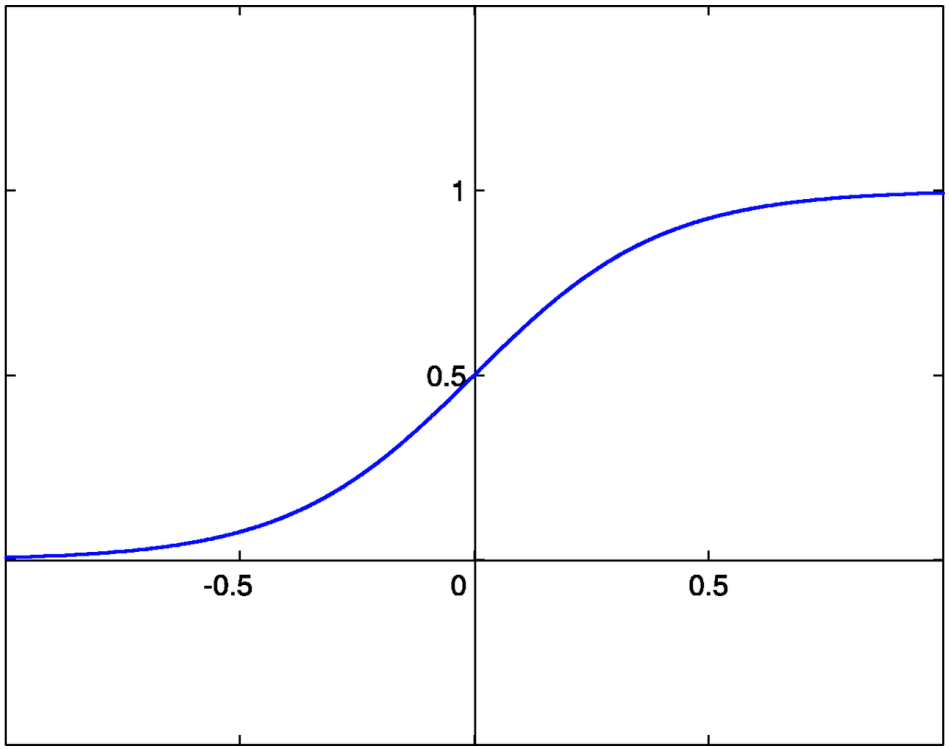
Sigmoid neuron



$a_1, ..., a_n \in [0, 1]$

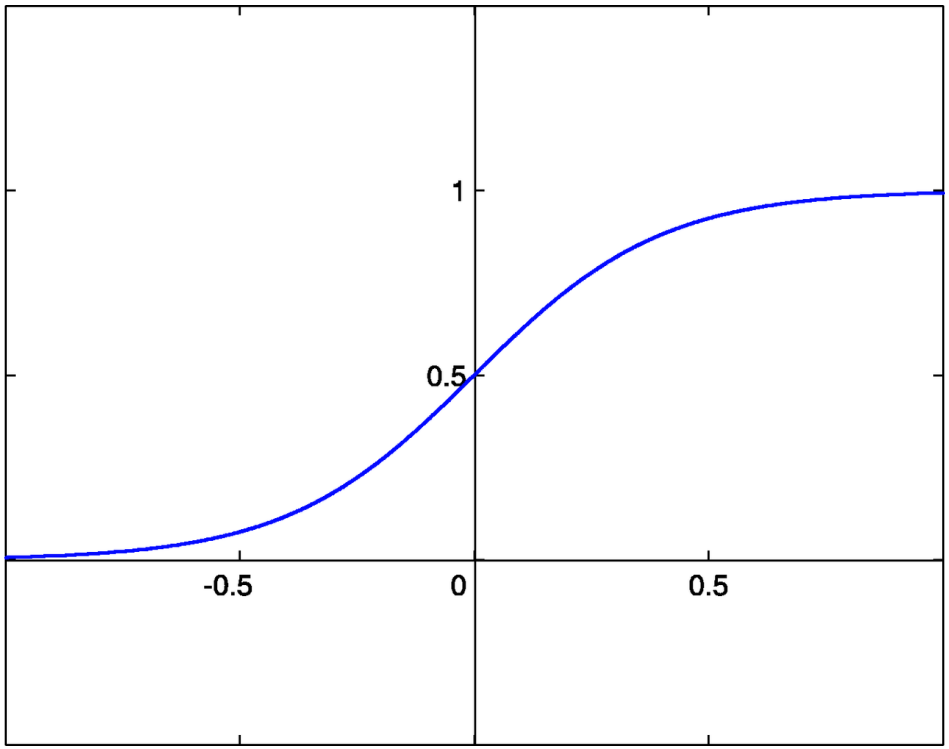
$s = \sigma(\sum_{i=0}^{n} a_i * w_i + b)$ où $\sigma(x) = \frac{1}{1+e^{-x}}$

Sigmoid neuron



Petit changement des w et des b $\rightarrow$ petit changement de la sortie ✓

Sigmoid neuron



Petit changement des w et des b $\rightarrow$ petit changement de la sortie ✓

Mais comment on entraîne ?

Entraînement

Objectif : minimiser l'erreur sur les prédictions :

Entraînement

Objectif : minimiser l'erreur sur les prédictions :

$$\left\{ \omega, b | E(\omega, b) = \min_{\omega', b'} E(\omega', b') \right\}$$

Entraînement

Problèmes à résoudre :

- Comment quantifier l'erreur ?

Entraînement

Problèmes à résoudre :

- Comment quantifier l'erreur ?
  $\rightarrow$ *Quadratic loss* :

$$L = \frac{1}{n} \sum (desired - predicted)^2$$

Entraînement

Problèmes à résoudre :

- Comment quantifier l'erreur ?
  $\rightarrow$ *Quadratic loss* :
  $$L = \frac{1}{n} \sum (desired - predicted)^2$$
- Comment minimiser ?

Entraînement

Problèmes à résoudre :

- Comment quantifier l'erreur ?
  $\rightarrow$ *Quadratic loss* :
  $$L = \frac{1}{n} \sum (desired - predicted)^2$$
- Comment minimiser ?
  $\rightarrow$ *Backpropagation*

Entraînement

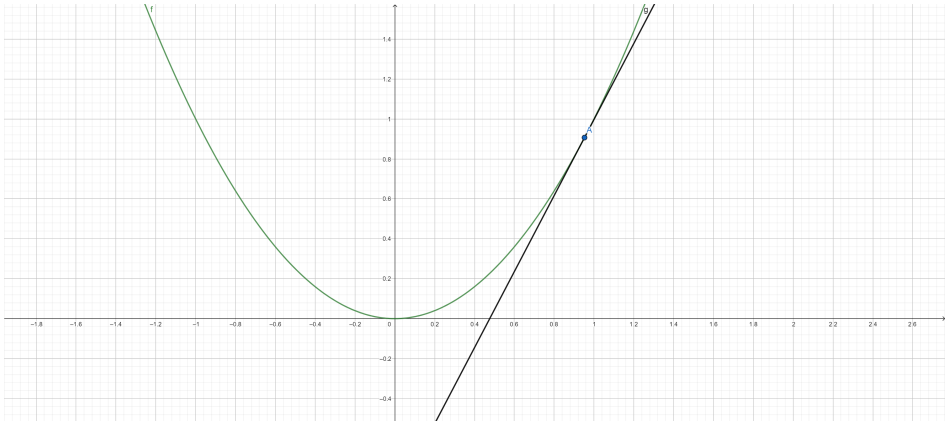
Algorithme du gradient (*Gradient Descent*)

Entraînement

Algorithme du gradient (*Gradient Descent*)
Idée : Atteindre le minimum d'une fonction de façon itérative

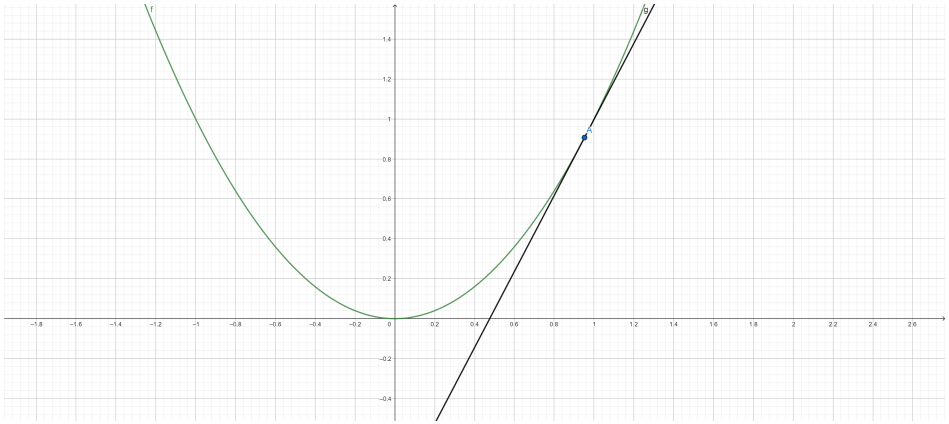
Entraînement

Algorithme du gradient (*Gradient Descent*)
Idée : Atteindre le minimum d'une fonction de façon itérative



Entraînement

Algorithme du gradient (*Gradient Descent*)
Idée : Atteindre le minimum d'une fonction de façon itérative



Entraînement

Pour chaque neurone :

$$\omega' = \omega - \eta \frac{\partial L}{\partial \omega}$$

$$b' = b - \eta \frac{\partial L}{\partial b}$$

Entraînement

Pour chaque neurone :

$$\omega' = \omega - \eta \frac{\partial L}{\partial \omega}$$

$$b' = b - \eta \frac{\partial L}{\partial b}$$

Objectif : Calculer ∇L

Entraînement

Pour chaque neurone :

$$\omega' = \omega - \eta \frac{\partial L}{\partial \omega}$$

$$b' = b - \eta \frac{\partial L}{\partial b}$$

Objectif : Calculer ∇L
$\rightarrow$ Une couche ✓

Entraînement

Pour chaque neurone :

$$\omega' = \omega - \eta \frac{\partial L}{\partial \omega}$$

$$b' = b - \eta \frac{\partial L}{\partial b}$$

Objectif : Calculer ∇L

$\rightarrow$ Une couche ✓

$\rightarrow$ Multicouches :

Entraînement

Pour chaque neurone :

$$\omega' = \omega - \eta \frac{\partial L}{\partial \omega}$$

$$b' = b - \eta \frac{\partial L}{\partial b}$$

Objectif : Calculer ∇L

$\rightarrow$ Une couche ✓

$\rightarrow$ Multicouches : Propagation du gradient en amont du réseau avec la règle de la chaîne : *Backpropagation* $\rightarrow$ **cs231**

Entraînement

- **1ère approche**
  Pour chaque entrée :
  → Calculer l'erreur
  → Calculer le gradient
  → Mettre à jour les paramètres

Entraînement

- **1ère approche**
  Pour chaque entrée :
  → Calculer l'erreur
  → Calculer le gradient
  → Mettre à jour les paramètres
- **2ème approche**
  Pour chaque ensemble d'entrée (*batch*) :
  → Calculer l'erreur moyenne
  → Calculer le gradient
  → Mettre à jour les paramètres

Entraînement

Résumé :
→ Sélectionner un *batch*
→ Pour chaque entrée calculer la sortie : *Forward propagation*
→ Calculer l'erreur moyenne
→ Calculer le gradient et modifier les paramètres : *Backpropagation*