

Réseaux de neurones

ENS'IA

Ensimag 2019-2020

14 octobre 2020

Qui sommes nous ?

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

Qui sommes nous ?

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

Les membres :

- Clément Doms (MOSIG) - Président
- Lucas Sort (MMIS) - Vice président
- Zaineb Tiour (IF) - Vice président 2
- Joana Lemerrier (IF) - Secrétaire générale
- Stepan Lebedev (ISI) - Vice Secrétaire
- Alexandre Audibert (MSIAM) - Comptable

Pourquoi nous rejoindre ?

- Te la péter à la machine à café
- Impressionner tes grand-parents
- Rajouter une ligne sur ton CV
- Eventuellement apprendre à faire de l'IA

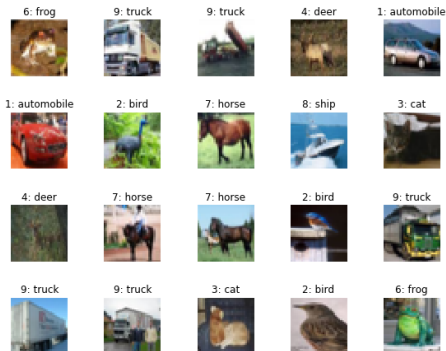
Pas besoin d'être expert pour nous aider

Le programme :

- Introduction
- Mon premier neurone
- Mon premier réseau de neurones
- Ma première convolution
- Challenge IA

Et après ? → à vous de nous dire

Rappel

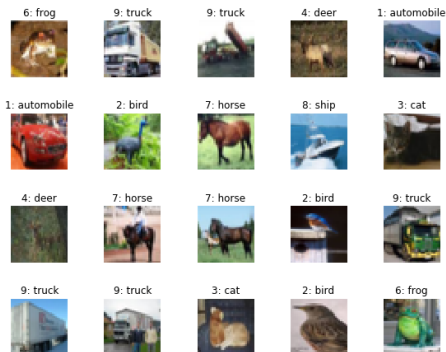


Classification d'images

→ Supervised learning : x et y connus

→ Unsupervised learning : que x

Rappel



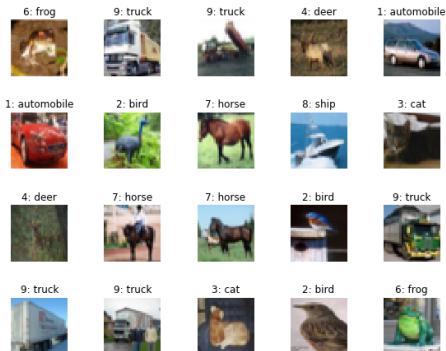
Classification d'images

→ 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

$$x \rightarrow y$$

But

$$f(x) = y$$

But

$f(x) = y$
Comment trouver f ?

But

$f(x) = y$
Comment trouver f ?

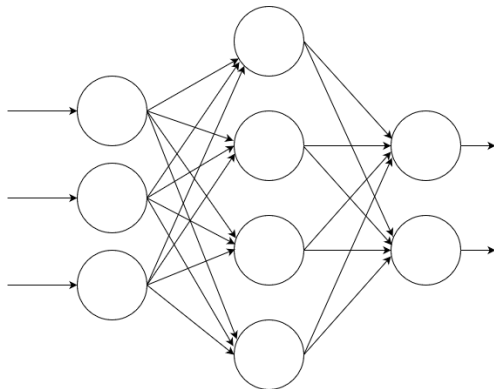
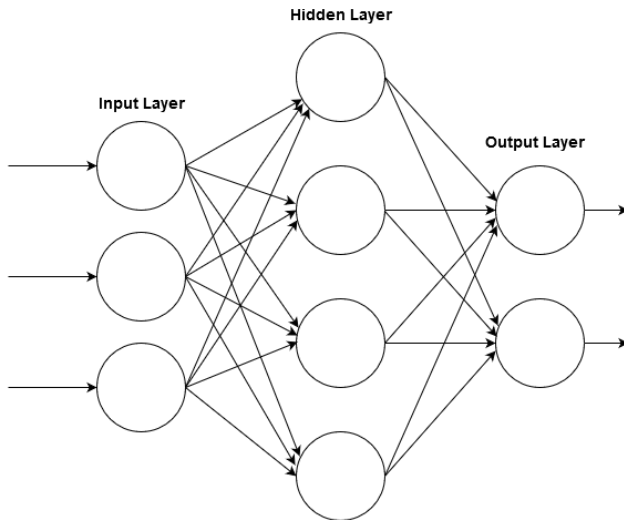


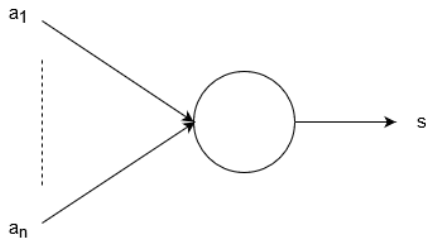
FIGURE 1 – Réseau de neurones

Réseau de neurones



→ Succession de couches de neurones

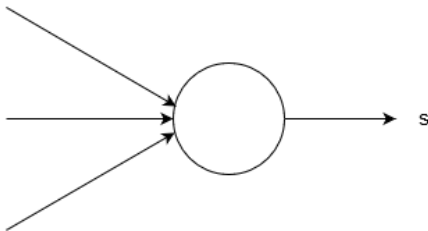
Perceptron



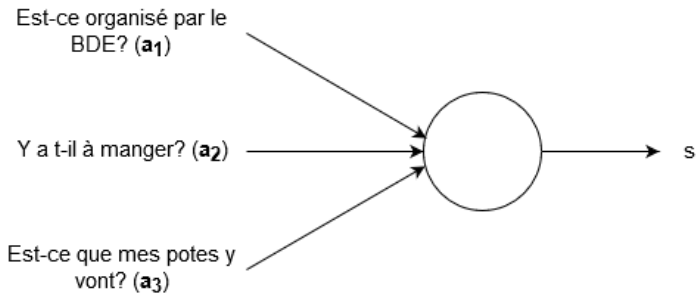
$$a_1, \dots, a_n, s \in 0, 1$$

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

Perceptron - Exemple

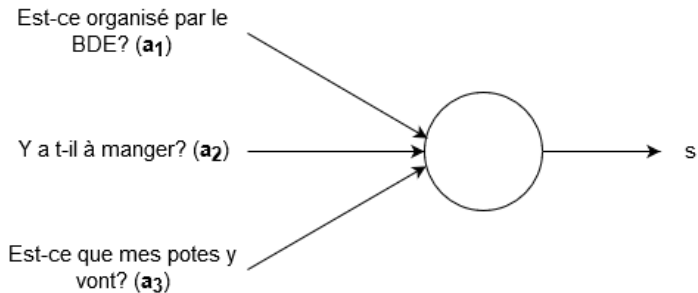


Perceptron - Exemple



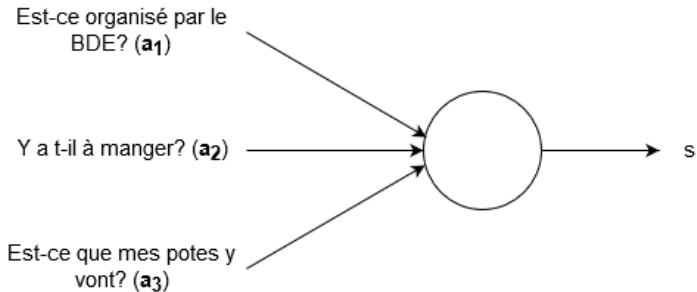
$$\text{Si } a_1 + a_2 + a_3 = 3, s = 1$$

Perceptron - Exemple



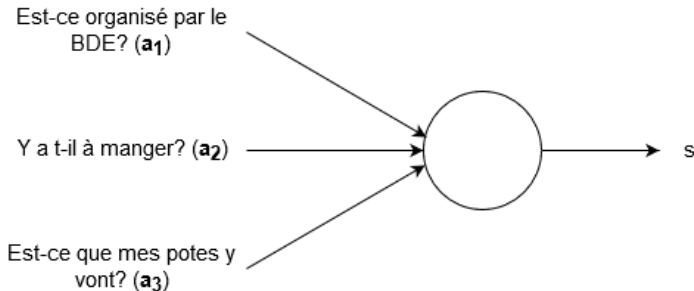
Si $a_1 + a_2 + a_3 = 3$, $s = 1$
On va à la soirée!

Perceptron - Exemple



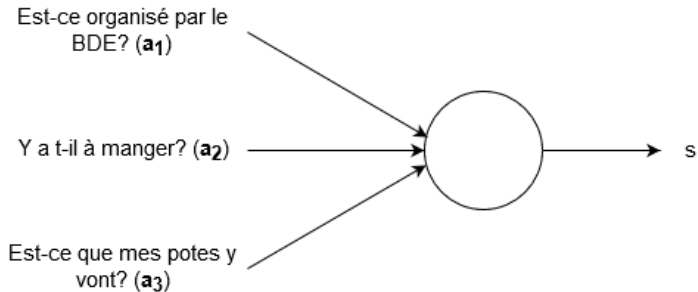
Si $a_1 + a_2 + a_3 > threshold$, $s = 1$

Perceptron - Exemple



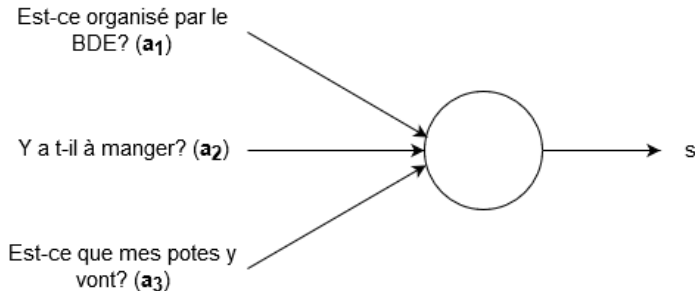
Si $a_1 + a_2 + a_3 > threshold$, $s = 1$
On va à la soirée !

Perceptron - Exemple



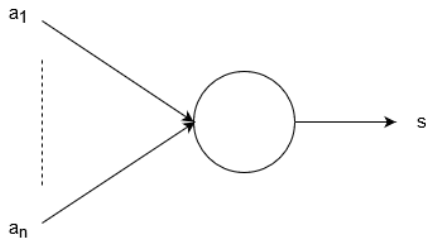
$$\text{Si } a_1 * w_1 + a_2 * w_2 + a_3 * w_3 + b > 0, s = 1$$

Perceptron - Exemple



Si $a_1 * w_1 + a_2 * w_2 + a_3 * w_3 + b > 0, s = 1$
On va à la soirée!

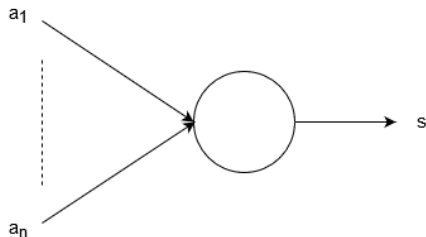
Perceptron



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

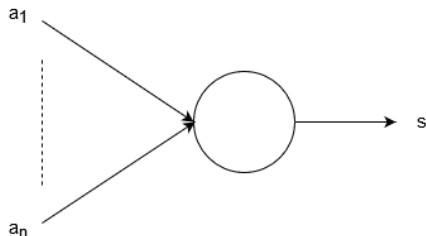
$$s = \begin{cases} 1 & \text{si } \sum_{i=0}^n a_i * w_i + b > 0 \\ 0 & \text{sinon.} \end{cases}$$

Perceptron



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

Perceptron



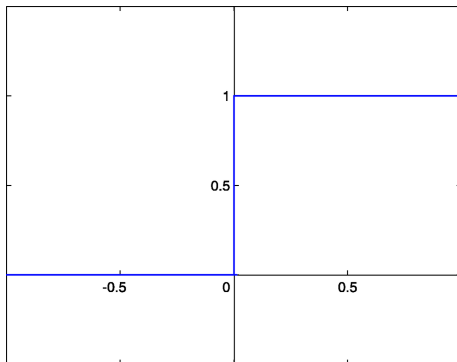
- Capable de reproduire des portes logiques !
- Trouver les w et b à la main c'est pénible

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

Perceptron

Comment apprendre ?

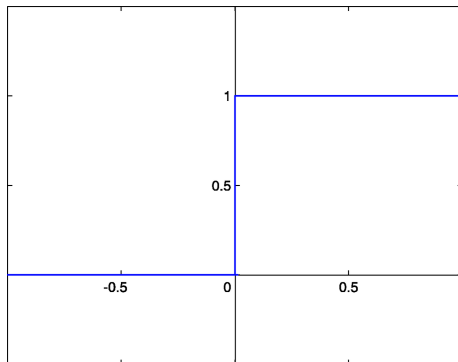
Petit changement des w et des $b \rightarrow$ petit changement de la sortie



Perceptron

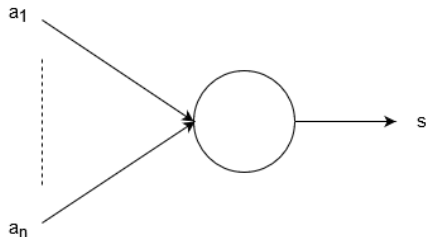
Comment apprendre ?

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



Pas possible ici

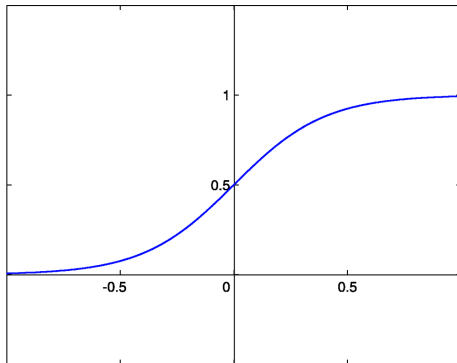
Sigmoid neuron



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

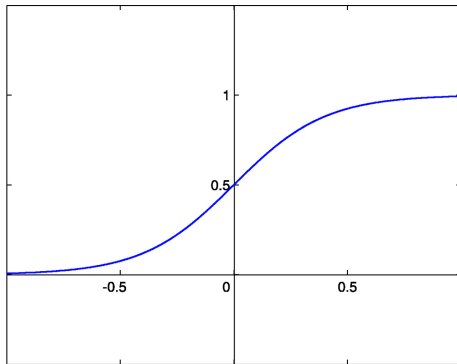
$$s = \sigma\left(\sum_{i=0}^n a_i * w_i + b\right) \text{ où } \sigma(x) = \frac{1}{1+e^{-x}}$$

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 ?

Objectif : minimiser l'erreur sur les prédictions :

Objectif : minimiser l'erreur sur les prédictions :

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

Problèmes à résoudre :

- Comment quantifier l'erreur ?

Problèmes à résoudre :

- Comment quantifier l'erreur ?

→ *Quadratic loss* :

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

Problèmes à résoudre :

- Comment quantifier l'erreur ?

→ *Quadratic loss* :

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

- Comment minimiser ?

Problèmes à résoudre :

- Comment quantifier l'erreur ?

→ *Quadratic loss* :

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

- Comment minimiser ?

→ *Backpropagation*

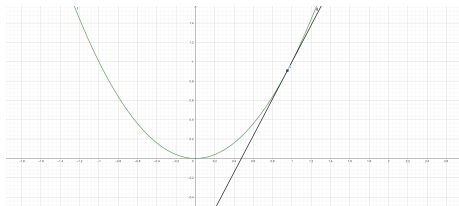
Algorithme du gradient (*Gradient Descent*)

Algorithme du gradient (*Gradient Descent*)

Idée : Atteindre le minimum d'une fonction de façon itérative

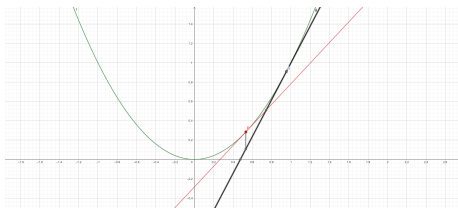
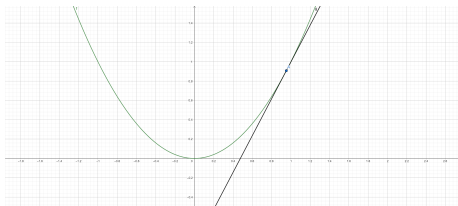
Algorithme du gradient (*Gradient Descent*)

Idée : Atteindre le minimum d'une fonction de façon itérative



Algorithme du gradient (*Gradient Descent*)

Idée : Atteindre le minimum d'une fonction de façon itérative



Pour chaque neurone :

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

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

Pour chaque neurone :

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

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

Objectif : Calculer ∇L

Pour chaque neurone :

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

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

Objectif : Calculer ∇L

→ Une couche ✓

Pour chaque neurone :

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

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

Objectif : Calculer ∇L

→ Une couche ✓

→ Multicouches :

Pour chaque neurone :

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

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

Objectif : Calculer ∇L

→ Une couche ✓

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

- **1ère approche**

Pour chaque entrée :

→ Calculer l'erreur

→ Calculer le gradient

→ Mettre à jour les paramètres

- **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

Résumé :

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