

Deep learning

DNN

Vincent Lefieux

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références



Plan

Introduction

Introduction

Neurone formel

Neurone formel

Fonctions
d'activation

Fonctions d'activation

Perceptron
multicouche

Rétro-
propagation

Perceptron multicouche

Architectures
classiques

Pratique des
DNN

Références

Rétro-propagation

Architectures classiques

Pratique des DNN

Plan

Introduction

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Point de vue biologique I

- ▶ Un **réseau de neurones** (artificiels) est un algorithme inspiré du fonctionnement des neurones du **cerveau humain**.
- ▶ Le **cerveau humain** comprend de l'ordre de **100 milliards** de neurones, chacun étant **connecté à environ 1 000 autres** par des synapses.



©Getty / Science Photo Library - KTSDESIGN

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

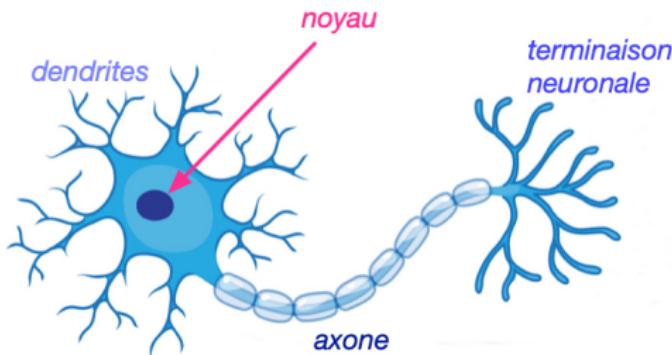
Architectures classiques

Pratique des DNN

Références

Point de vue biologique II

- De manière extrêmement simplifiée, un neurone est une cellule cérébrale permettant de collecter, traiter et transmettre des signaux électriques.
- Il est notamment composé :
 - de **dendrites** : les « entrées » du neurone,
 - d'un **axone** : la « sortie » du neurone.



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

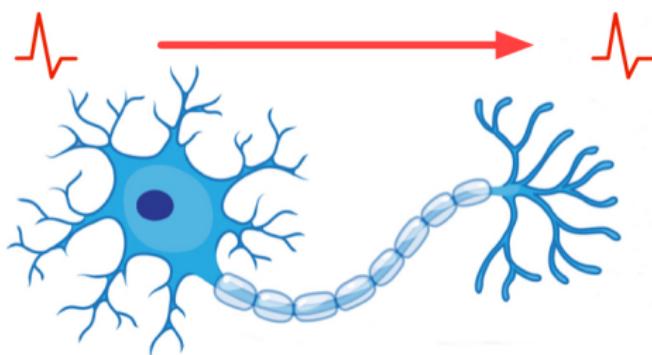
Architectures
classiques

Pratique des
DNN

Références

Point de vue biologique III

- ▶ De manière extrêmement simplifiée, un neurone est une cellule cérébrale permettant de collecter, traiter et transmettre des signaux électriques.
- ▶ Il est notamment composé :
 - ▶ de **dendrites** : les « entrées » du neurone,
 - ▶ d'un **axone** : la « sortie » du neurone.



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

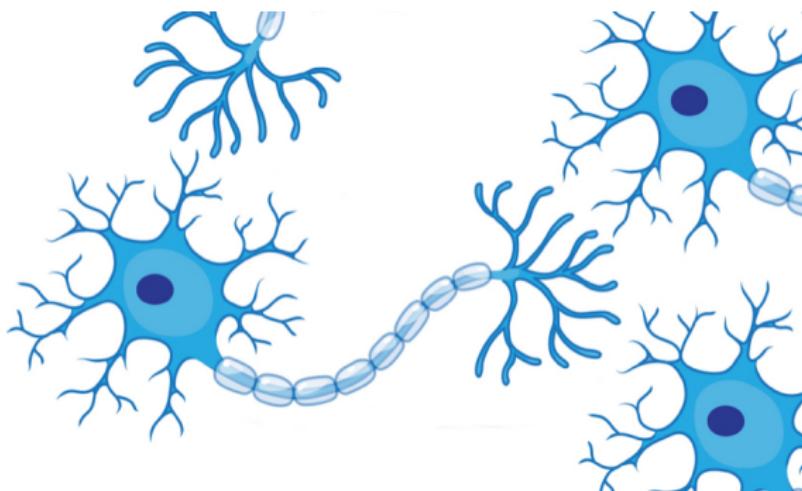
Architectures
classiques

Pratique des
DNN

Références

Point de vue biologique IV

- ▶ De manière extrêmement simplifiée, un neurone est une cellule cérébrale permettant de collecter, traiter et transmettre des signaux électriques.
- ▶ Il est notamment composé :
 - ▶ de **dendrites** : les « entrées » du neurone,
 - ▶ d'un **axone** : la « sortie » du neurone.



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Eléments chronologiques

- ▶ 1943 : **neurone formel**
McCulloch (neurophysiologiste) et Pitts (logicien).
(McCulloch et Pitts, 1943)
- ▶ 1958 : **perceptron simple**
(Rosenblatt, 1958)
- ▶ 1974-1986 : **perceptron multicouche** et **rétro-propagation**
Rumelhart, Hinton et Williams, suite à des travaux de
Werbos.
(Werbos, 1974), (Rumelhart et collab., 1986)
- ▶ Fin des années 1990 : **deep learning**
Bengio, Hinton et LeCun.
(Le Cun et collab., 1989)

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Plan

Neurone formel

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Définition

- ▶ Un **neurone formel** est composé :
 - ▶ d'**entrées** X_1, \dots, X_p dans \mathbb{R} , auxquelles sont associés des **poids** (synaptiques) $\omega_1, \dots, \omega_p$ et un **biais** ω_0 ,
 - ▶ d'une **sortie** (réponse) Y dans \mathbb{R} .
- ▶ Plusieurs opérations sont successivement effectuées par le neurone :
 1. Somme pondérée des entrées avec les **poids** $(\omega_j)_{j \in \{1, \dots, p\}}$.
 2. Ajout du **biais** ω_0 .
 3. Transformation par une **fonction d'activation** φ .
- ▶ La réponse obtenue au final est :

$$Y = \varphi \left(\omega_0 + \sum_{j=1}^p \omega_j X_j \right).$$

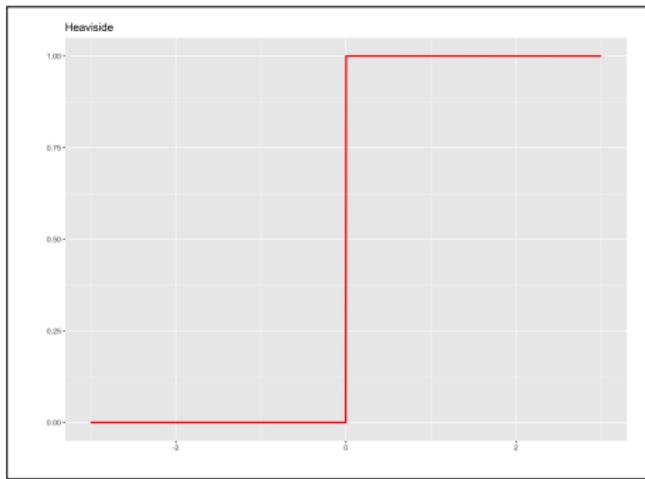
On note **a** le résultat des 2 premières étapes :

$$a = \omega_0 + \sum_{j=1}^p \omega_j X_j.$$

Fonction d'activation d'Heaviside

- ▶ Dans la première version du neurone formel, la fonction d'activation retenue était celle de **Heaviside** (ou échelon unité ou marche - *step*) :

$$\varphi(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases}.$$



- ▶ Si $\sum_{j=1}^p \omega_j X_j$ dépasse le seuil $-\omega_0$, la sortie du neurone vaut 1, sinon 0 (on parle alors de neurone éteint).

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

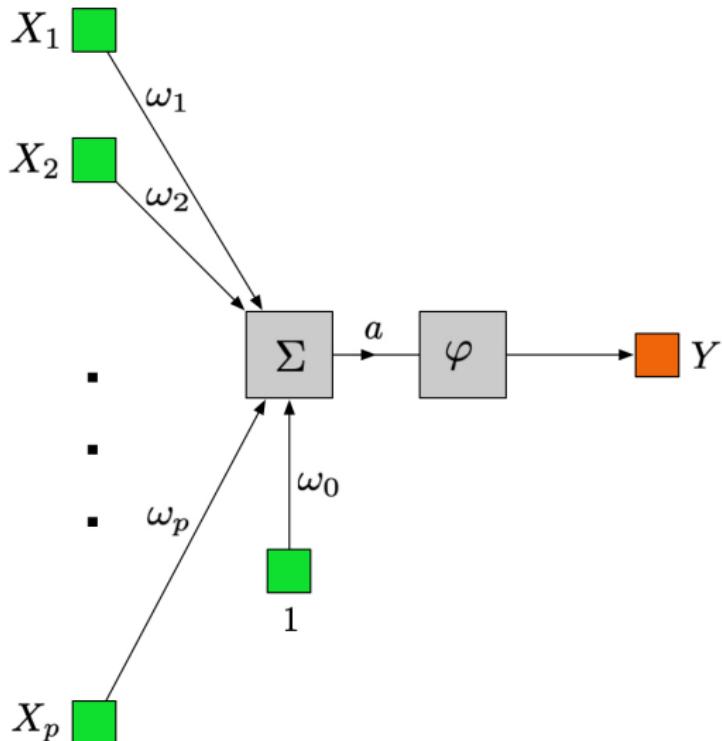
Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Représentation I



p entrées

1 sortie

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

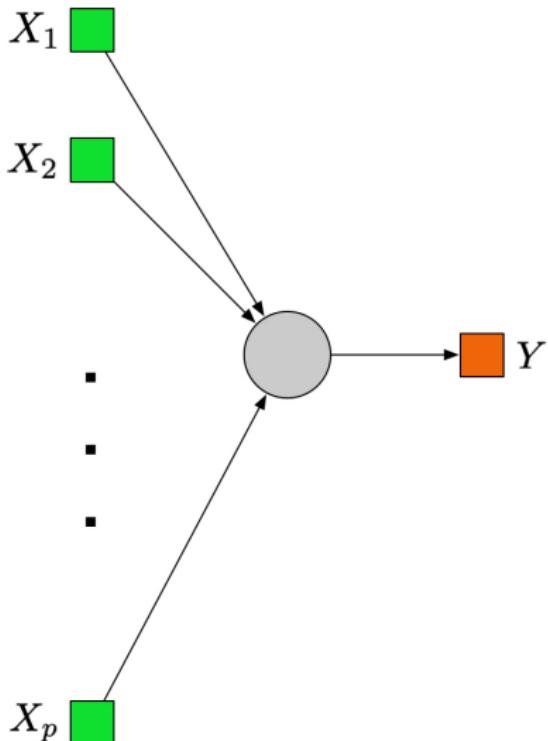
Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Représentation II



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Perceptron (simple) I

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

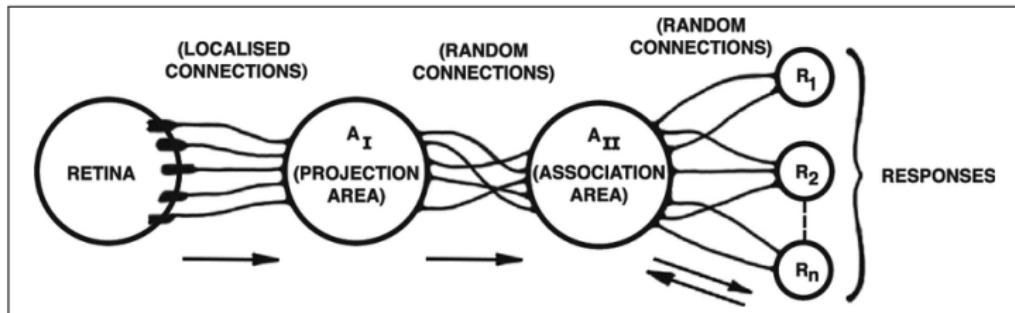
Réto-propagation

Architectures classiques

Pratique des DNN

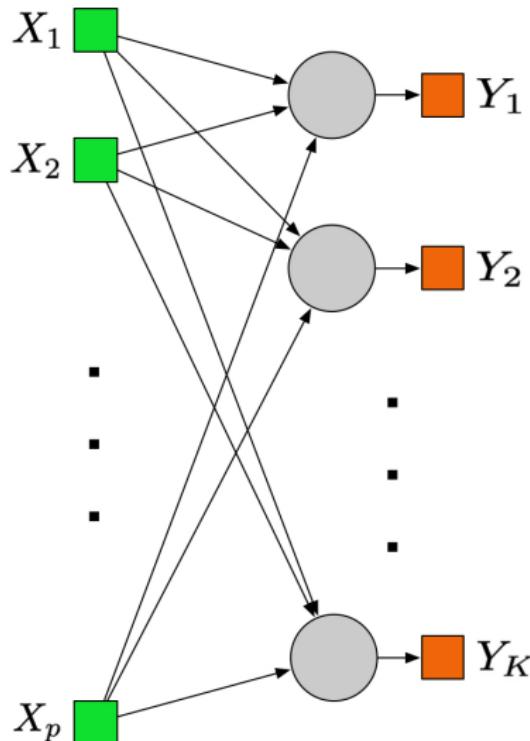
Références

- Le **perceptron** (simple) est un ensemble de neurones formels, reliés aux mêmes entrées auxquels, on adjoint une règle d'apprentissage pour les poids et les biais.



(Rosenblatt, 1958)

Perceptron (simple) II



p entrées

K sorties

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Réto-
propagation

Architectures
classiques

Pratique des
DNN

Références

Plan

Fonctions d'activation

Introduction

Neurone formel

**Fonctions
d'activation**

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Choix d'une fonction d'activation

- ▶ Le choix de la **fonction d'activation** est important, il est conditionné par des propriétés de régularité et d'interprétabilité.
- ▶ La facilité d'obtention de la **dérivée** est intéressante d'un point de vue calculatoire.
- ▶ Parmi les plus utilisées, on trouve :
 - ▶ la **linéaire** pour la couche de sortie dans le cadre d'une régression,
 - ▶ la **sigmoïde** pour la couche de sortie dans le cadre d'une classification supervisée binaire,
 - ▶ la **ReLU** pour les couches cachées,
 - ▶ la **tangente hyperbolique** pour les LSTM.

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

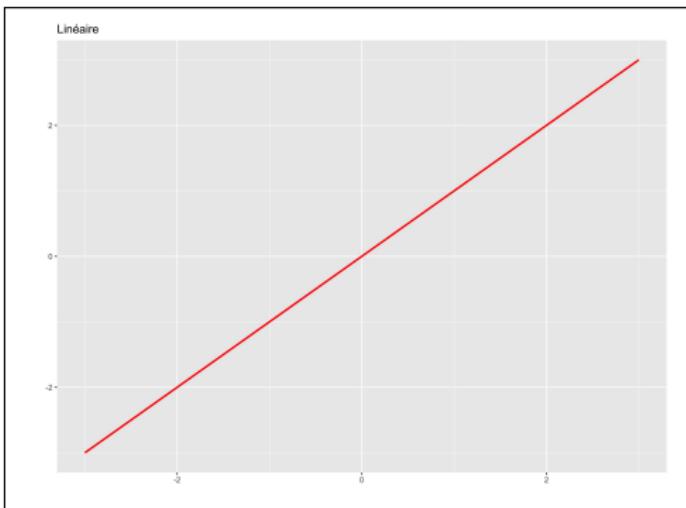
Pratique des
DNN

Références

Fonction d'activation linéaire

- ▶ La fonction **linéaire** (ou identité) est :

$$\varphi(x) = x .$$



- ▶ A valeurs dans \mathbb{R} , elle est infiniment continûment dérivable, de dérivée :

$$\varphi'(x) = 1 .$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

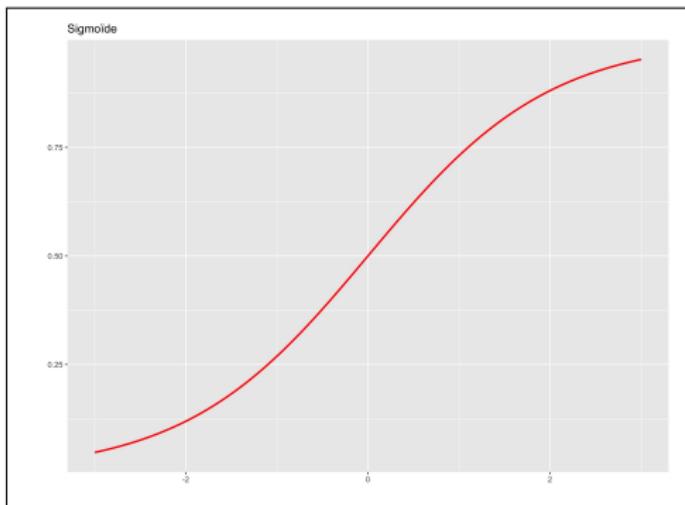
Pratique des DNN

Références

Fonction d'activation sigmoïde

- ▶ La fonction **sigmoïde** (ou logistique ou marche douce - *soft step*) est :

$$\varphi(x) = \frac{1}{1 + e^{-x}} .$$



- ▶ A valeurs dans $[0, 1]$, elle est infiniment continûment dérivable, de dérivée :

$$\varphi'(x) = \varphi(x)[1 - \varphi(x)] .$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

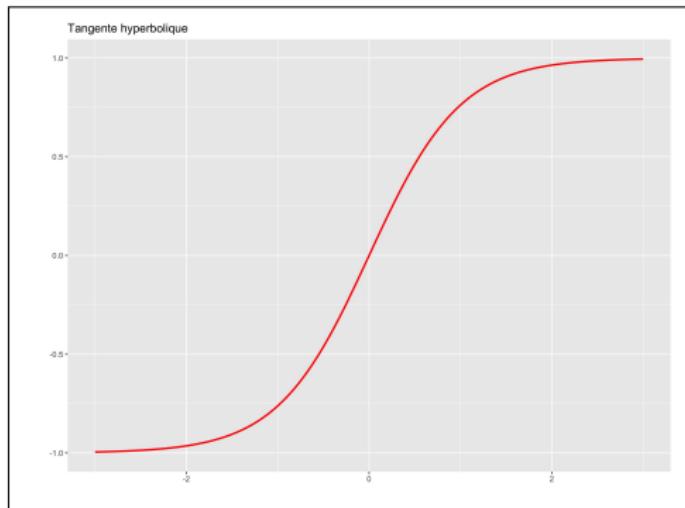
Pratique des DNN

Références

Fonction d'activation tangente hyperbolique

- ▶ La fonction tangente hyperbolique est :

$$\varphi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} .$$



- ▶ A valeurs dans $[-1, 1]$, elle est infiniment continûment dérivable, de dérivée :

$$\varphi'(x) = 1 - \varphi^2(x) .$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

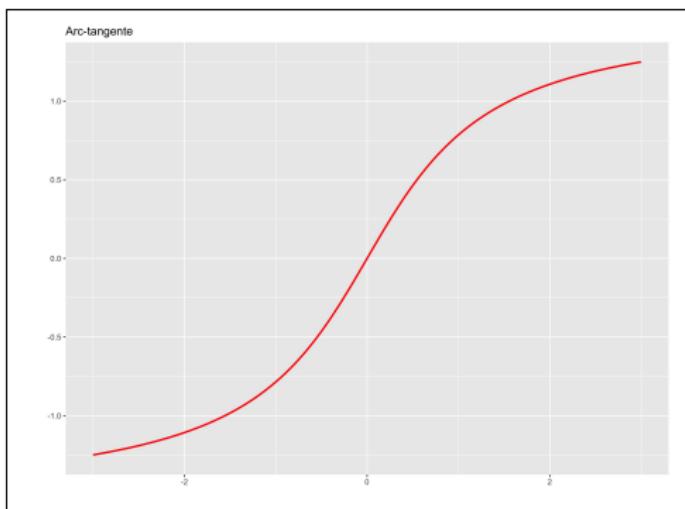
Pratique des DNN

Références

Fonction d'activation arc-tangente

- ▶ La fonction **arc-tangente** est :

$$\varphi(x) = \tan^{-1}(x) .$$



- ▶ A valeurs dans $[-\frac{\pi}{2}, \frac{\pi}{2}]$, elle est infiniment continûment dérivable, de dérivée :

$$\varphi'(x) = \frac{1}{1+x^2} .$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

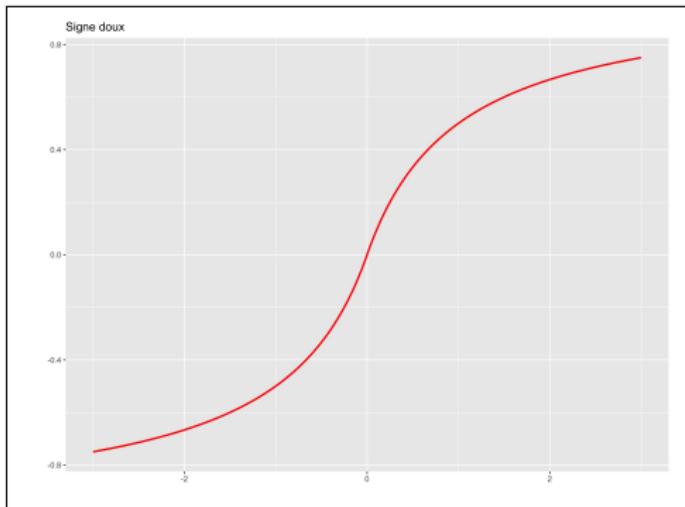
Pratique des DNN

Références

Fonction d'activation signe doux

- La fonction **soft sign** (« signe doux ») est :

$$\varphi(x) = \frac{x}{1 + |x|} .$$



- A valeurs dans $[-1, 1]$, elle est continûment dérivable, de dérivée :

$$\varphi'(x) = \frac{1}{(1 + |x|)^2} .$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

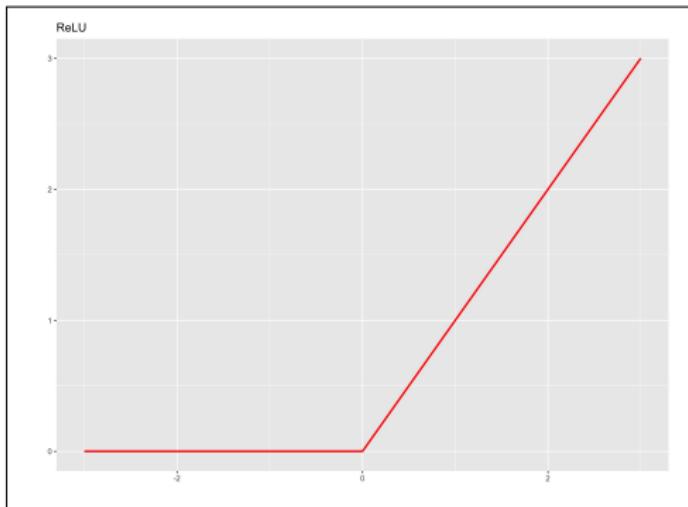
Pratique des DNN

Références

Fonction d'activation ReLU

- ▶ La fonction **ReLU** (*Rectified Linear Unit* ou rampe) est :

$$\varphi(x) = \begin{cases} x & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases}.$$



- ▶ A valeurs dans \mathbb{R}^+ , sa dérivée vaut :

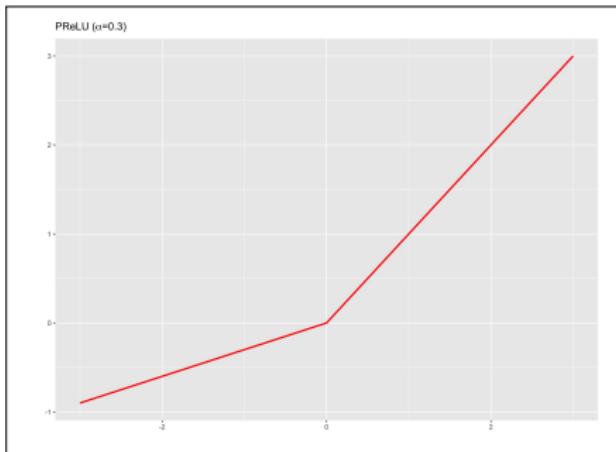
$$\varphi'(x) = \begin{cases} 1 & \text{si } x > 0 (\geq 0) \\ 0 & \text{si } x < 0 \end{cases}.$$

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

Fonction d'activation PReLU

- ▶ La fonction **PReLU** (*Parametric Rectified Linear Unit*) est :

$$\varphi(x) = \begin{cases} x & \text{si } x \geq 0 \\ \alpha x & \text{si } x < 0 \end{cases} \quad \text{avec } \alpha \in \mathbb{R}^+.$$



- ▶ A valeurs dans \mathbb{R} , sa dérivée vaut :

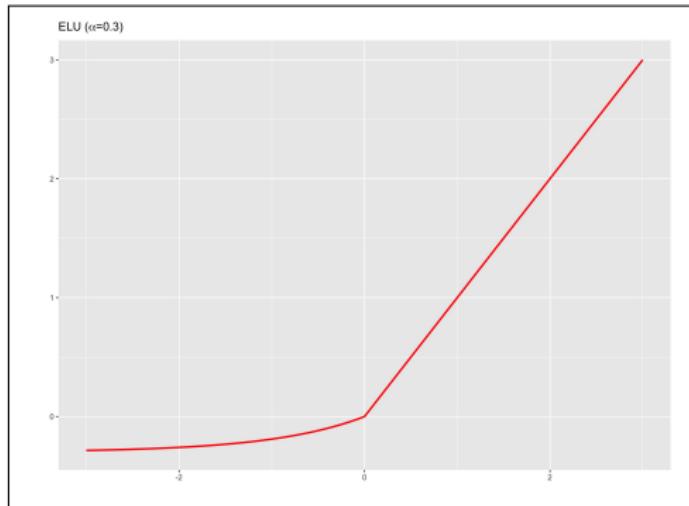
$$\varphi'(x) = \begin{cases} 1 & \text{si } x > 0 (\geq 0) \\ \alpha & \text{si } x < 0 \end{cases} .$$

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

Fonction d'activation ELU

- ▶ La fonction **ELU** (*Exponential Linear Unit*) est :

$$\varphi(x) = \begin{cases} x & \text{si } x \geq 0 \\ \alpha(e^x - 1) & \text{sinon} \end{cases} \quad \text{avec } \alpha \in \mathbb{R}^+.$$



- ▶ A valeurs dans $]-\alpha, +\infty[$, sa dérivée vaut :

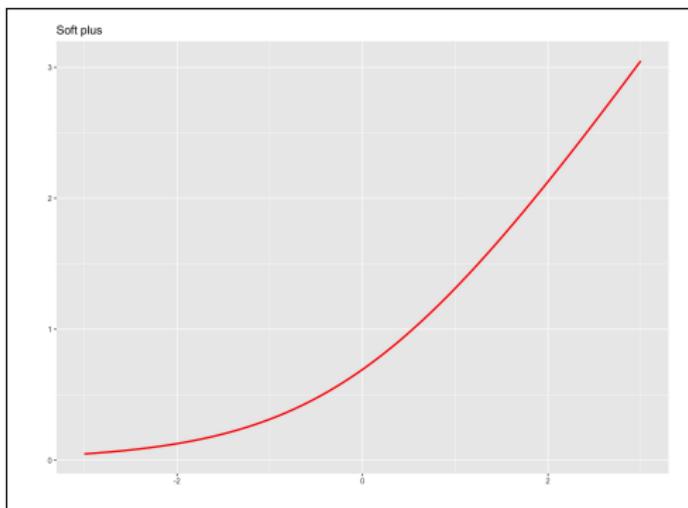
$$\varphi'(x) = \begin{cases} 1 & \text{si } x > 0 (\geq 0) \\ \varphi(x) + \alpha & \text{si } x < 0 \end{cases} .$$

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

Fonction d'activation soft plus

- ▶ La fonction *soft plus* est :

$$\varphi(x) = \ln(1 + e^x) \quad \text{avec } \alpha \in \mathbb{R}^+.$$



- ▶ A valeurs dans \mathbb{R}^+ , elle est infiniment continûment dérivable, de dérivée :

$$\varphi'(x) = \frac{1}{1 + e^{-x}}.$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Plan

Perceptron multicouche

Introduction

Neurone formel

Fonctions
d'activation

**Perceptron
multicouche**

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Limites du neurone formel

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

- ▶ Le neurone formel ou le perceptron simple ne permettent de traiter correctement que des problèmes linéairement séparables.
- ▶ Afin de traiter des **problèmes non-linéairement séparables** (tels que que le XOR (« OU exclusif »)), on doit ajouter des couches aux réseaux de neurones.

Fonctions logiques

Introduction

Neurone formel

Fonctions d'activation

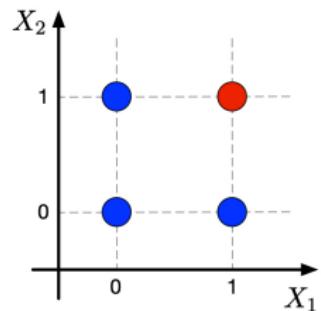
Perceptron multicouche

Rétro-propagation

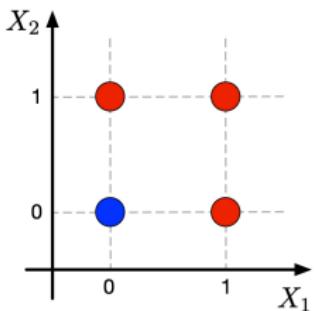
Architectures classiques

Pratique des DNN

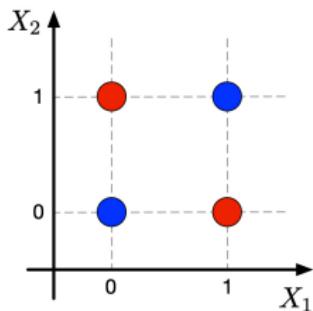
Références



AND



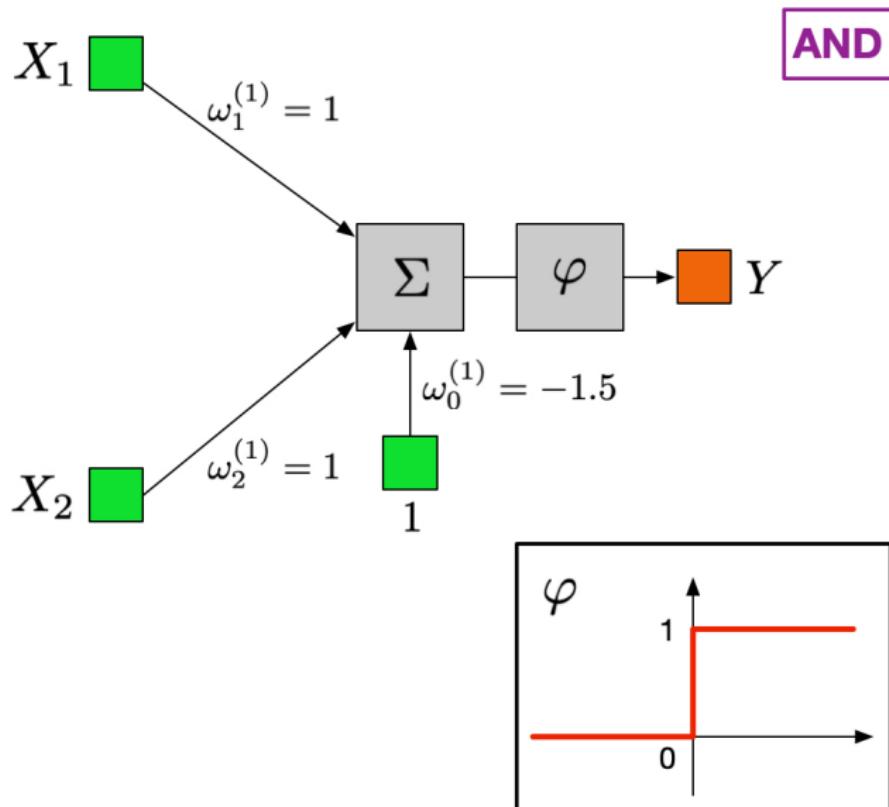
OR



XOR

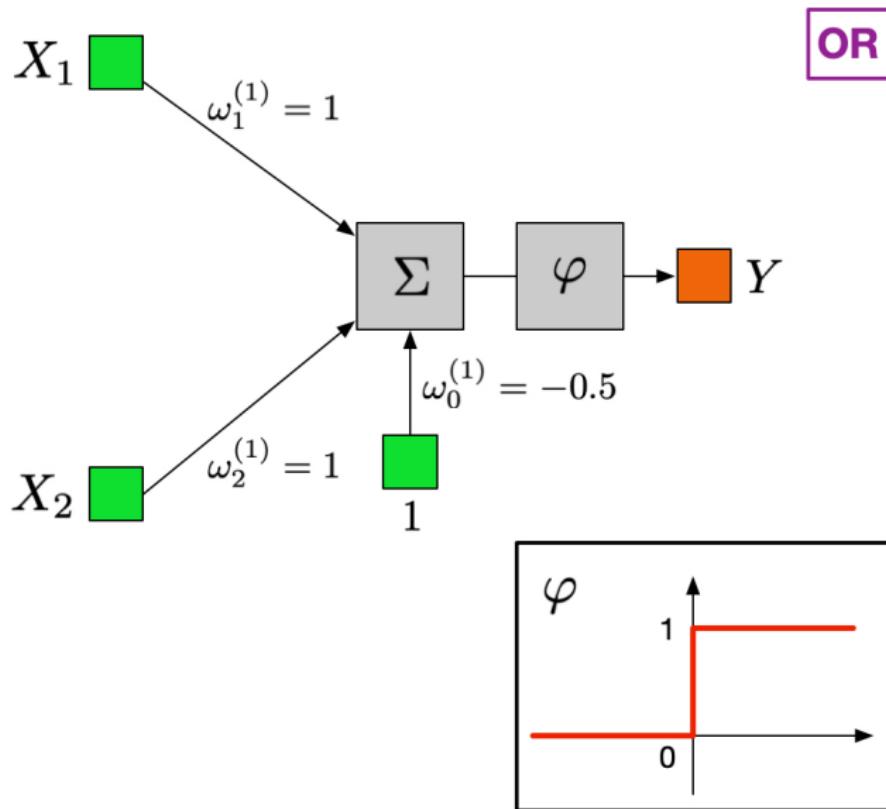
$$Y = 0 \quad Y = 1$$

Fonction AND



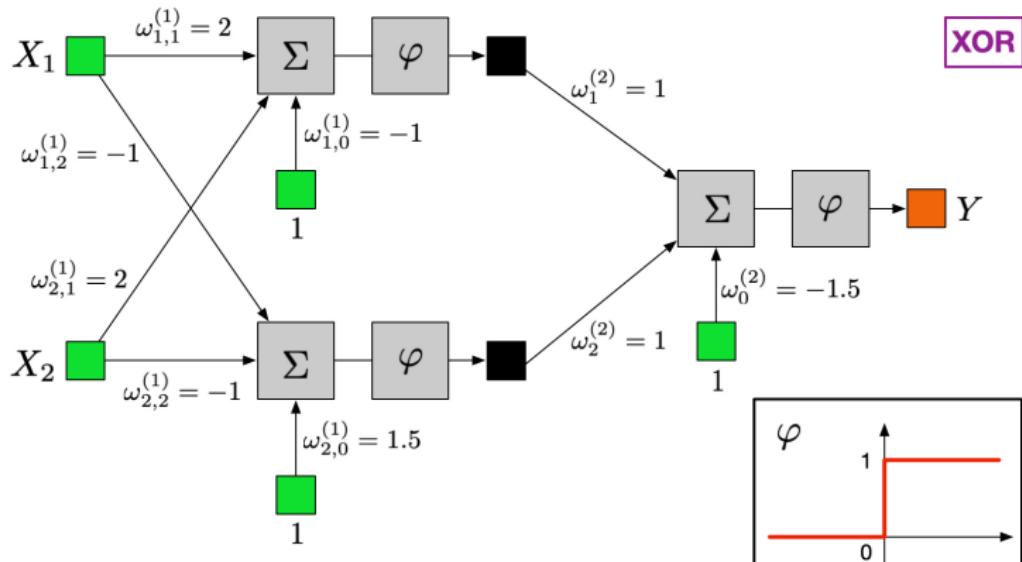
- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références

Fonction OR



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références

Fonction XOR



Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Architecture

- ▶ Un **perceptron multicouche** (**MLP** : *Multi-Layer Perceptron*) est constitué par des couches de neurones, chaque neurone d'une couche étant relié aux neurones de la couche adjacente.
- ▶ On parle de réseau de neurones **profond** (**deep**) ou dense, au final noté **DNN**, dès qu'on a plus de 2 couches : plus le nombre de couches est élevé, plus le réseau est dit « profond ».
- ▶ On parle d'architecture **feed-forward** (à propagation avant) : il n'y a pas de boucle de retour vers les couches plus basses du réseau.
- ▶ Les couches de neurones intermédiaires sont appelées **couches cachées** : plus leur nombre est important, plus le risque de sur-apprentissage est important.
- ▶ Pour les problématiques de traitement d'images, on peut trouver jusqu'à 30 couches cachées... .

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

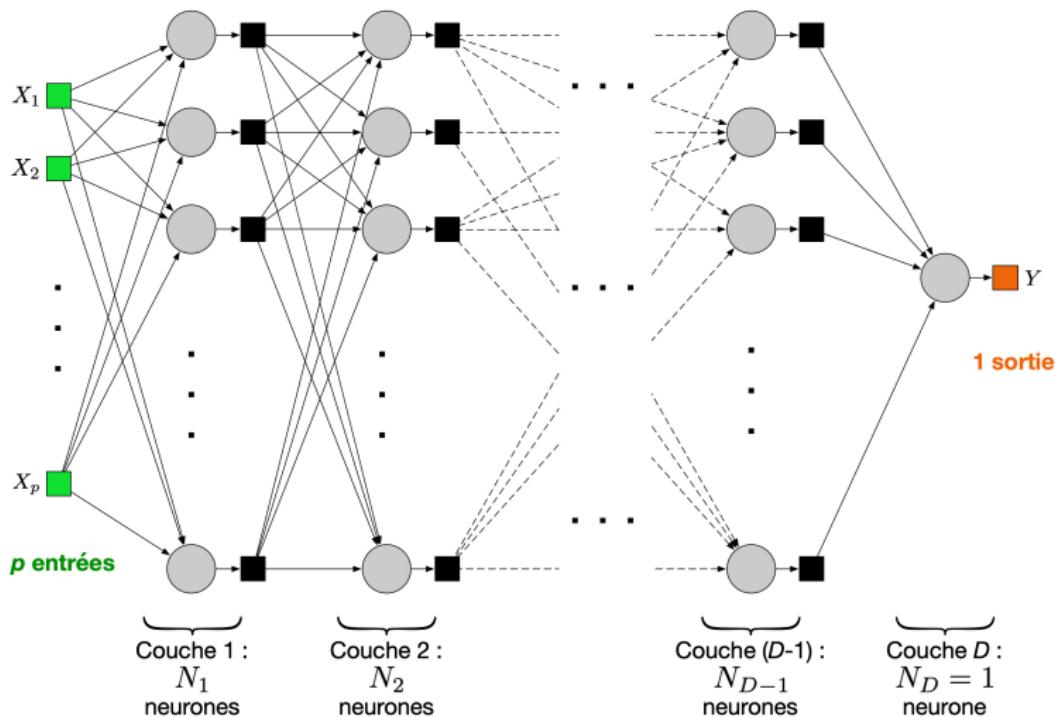
Rétro-
propagation

Architectures
classiques

Pratique des
DNN

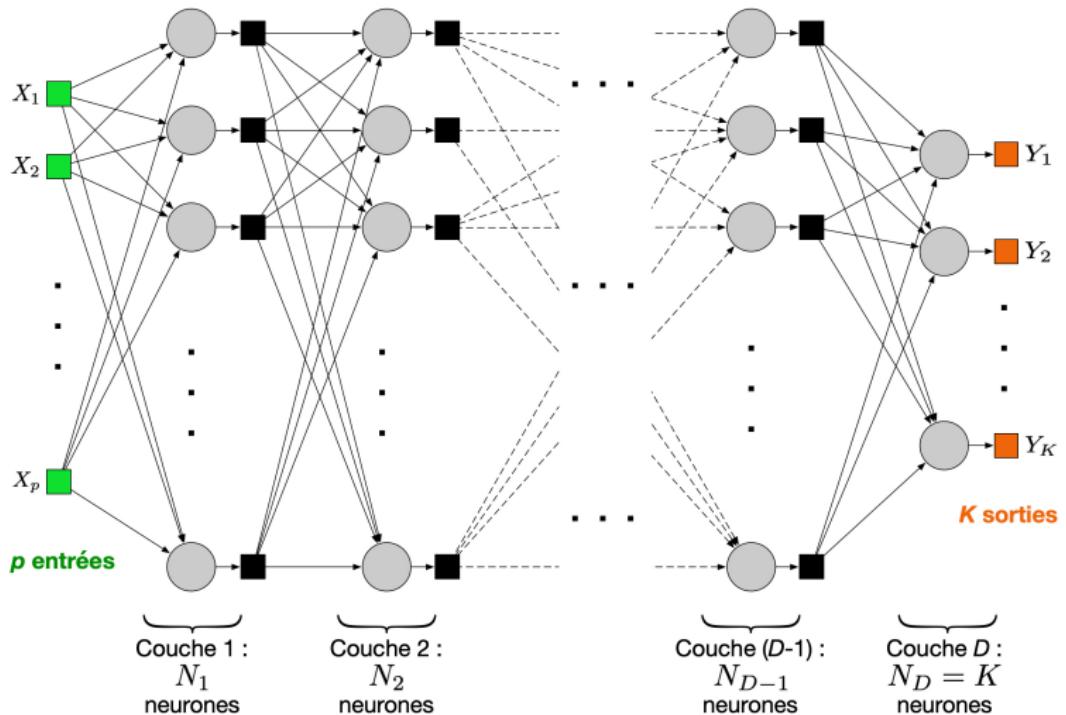
Références

Représentation I



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références

Représentation II



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références

Théorème d'approximation universelle

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

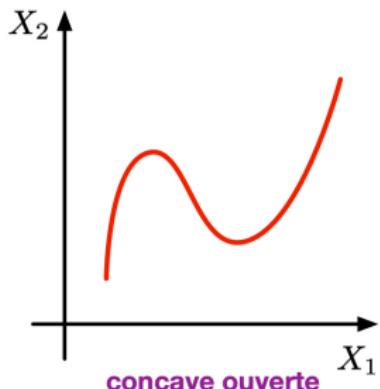
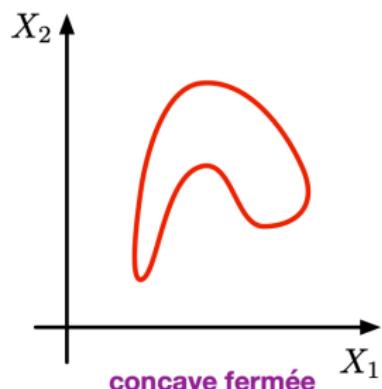
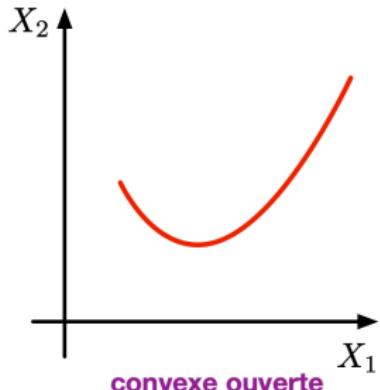
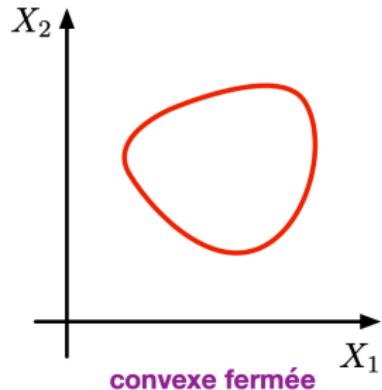
Architectures
classiques

Pratique des
DNN

Références

- ▶ Le **théorème d'approximation universelle** indique que toute fonction continue peut s'approximer par un réseau de neurones avec une couche cachée.
(Cybenko, 1989), (Hornik, 1991)
- ▶ La complexité des problèmes peut néanmoins exiger beaucoup (trop) de neurones, conduisant à des réseaux de neurones « *fat* » ; on leur préfère en pratique des réseaux de neurones profonds « *deep* ».

Frontières de décision



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références

Estimation des poids et des biais

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

- ▶ L'apprentissage des poids et des biais s'effectue classiquement à l'aide de l'**algorithme de retropropagation** (*backpropagation*).
- ▶ Dans le domaine des réseaux de neurones, on emploie usuellement le terme d'**entraînement**.

Plan

Rétro-propagation

Principe

Taux d'apprentissage

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Principe

Taux d'apprentissage

Architectures
classiques

Pratique des
DNN

Références

Plan

Rétro-propagation

Principe

Taux d'apprentissage

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Principe

Taux d'apprentissage

Architectures
classiques

Pratique des
DNN

Références

Idée

- ▶ L'erreur sur un neurone de la couche cachée $d \in \{1, \dots, D - 1\}$ dépend de l'erreur des neurones de la couche suivante ($d + 1$).
- ▶ C'est ce constat qui est à l'origine de l'idée de **rétro-propagation** (*backward propagation* ou *backpropagation*) des erreurs : on amende les valeurs des poids et des biais de chaque neurone en rétro-propageant l'erreur observée sur la dernière couche.
- ▶ On applique la **descente du gradient** au risque empirique.
- ▶ La **propagation avant** (*forward propagation*) précède la rétro-propagation.

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Principe

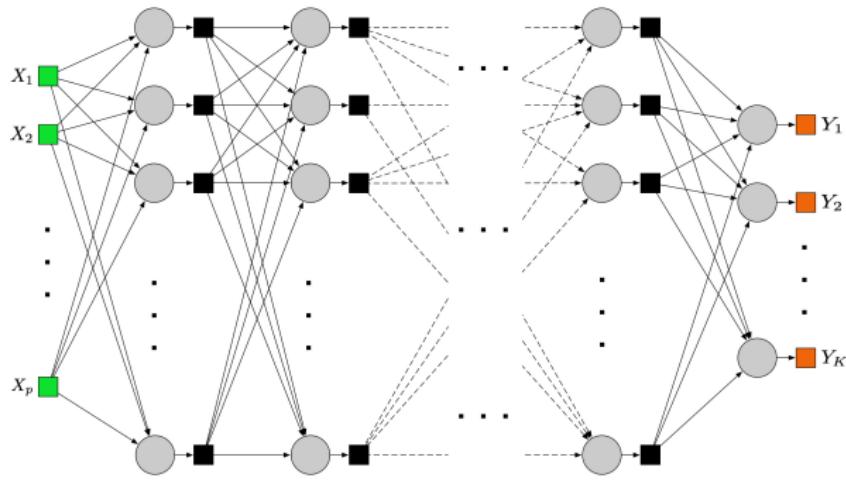
Taux d'apprentissage

Architectures classiques

Pratique des DNN

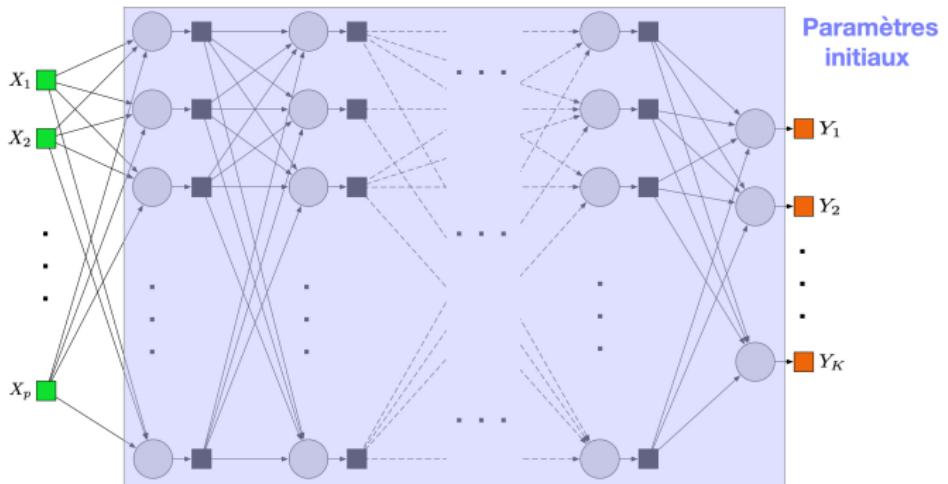
Références

Pour une architecture fixée



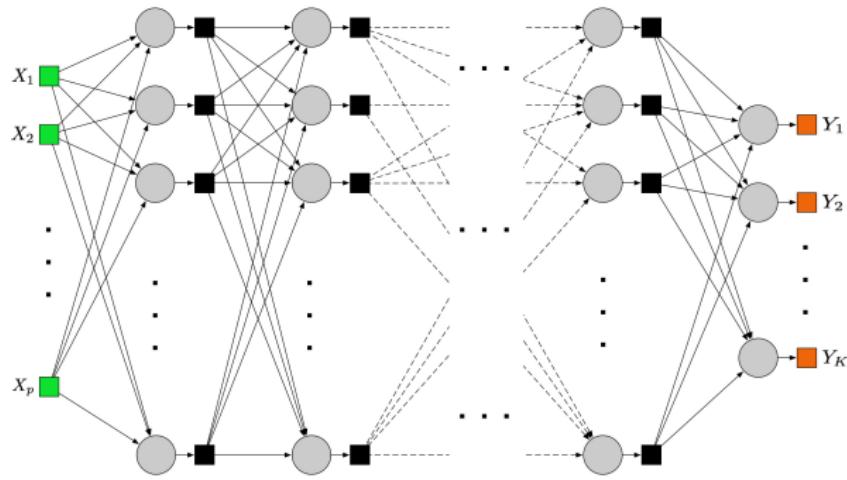
- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

Initialisation



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

Propagation avant...

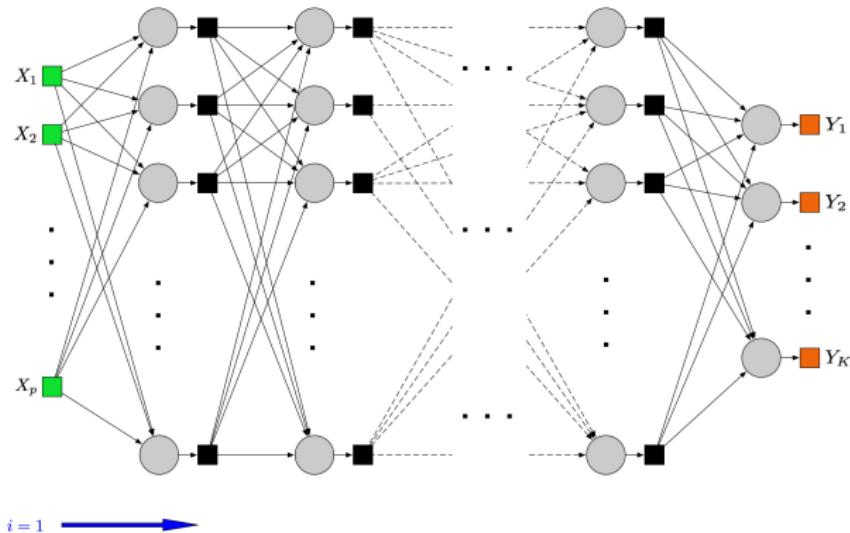


$i = 1$

Forward

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

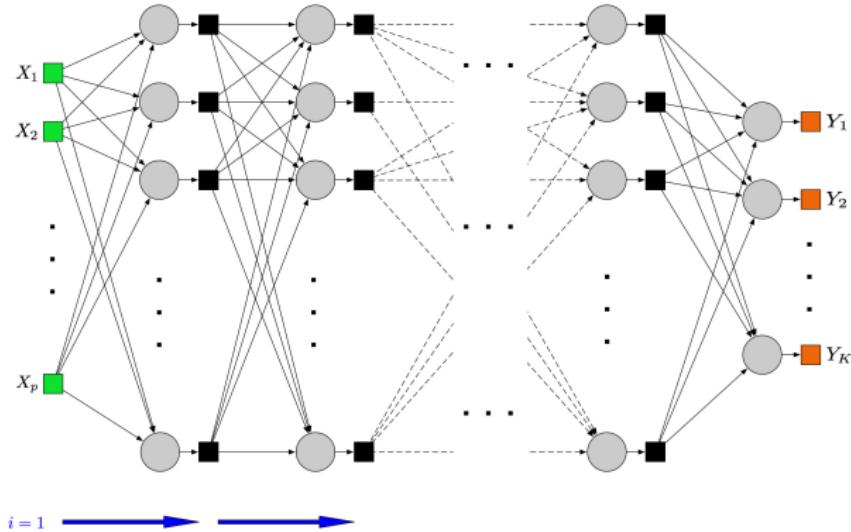
Propagation avant...



Forward

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

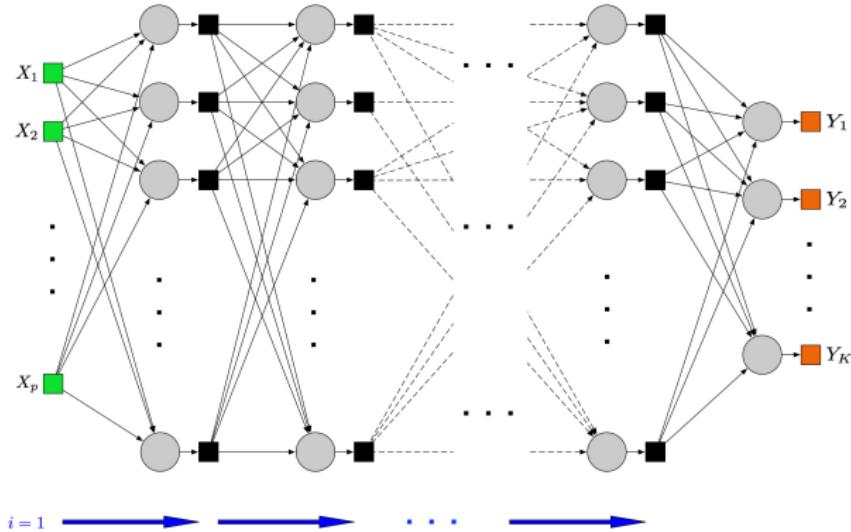
Propagation avant...



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

Forward

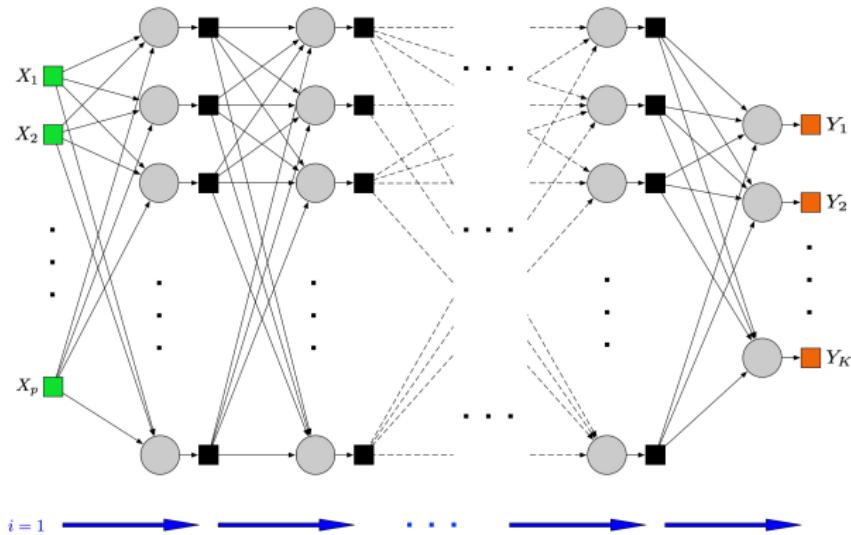
Propagation avant...



Forward

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

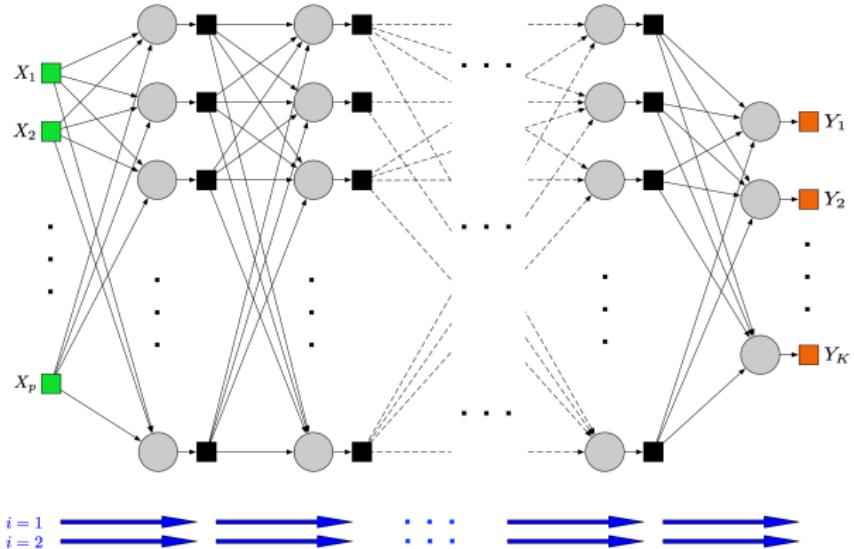
Propagation avant...



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

Forward

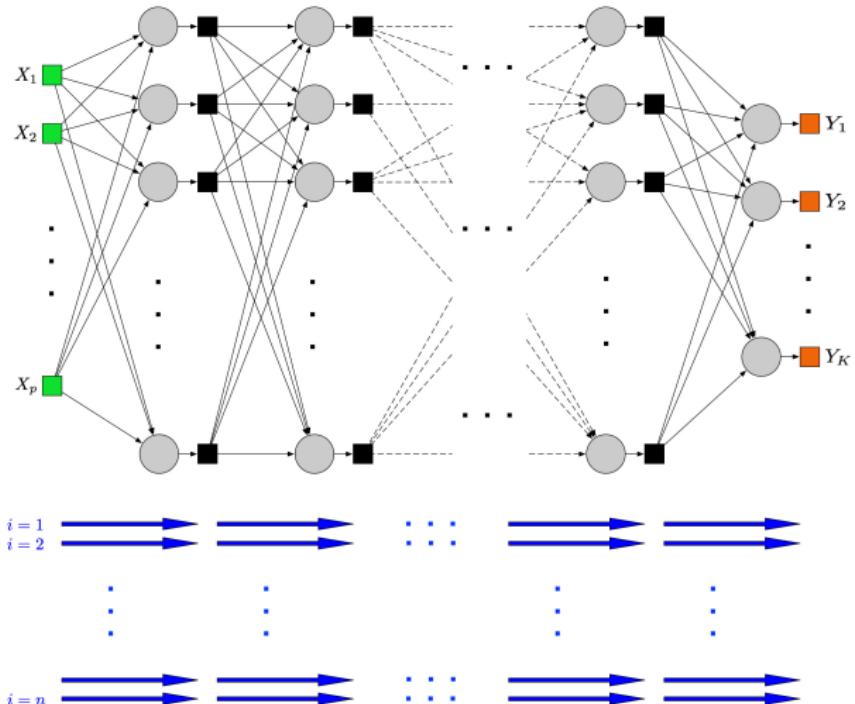
Propagation avant...



Forward

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

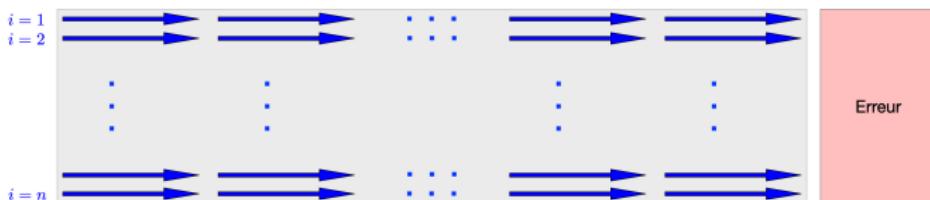
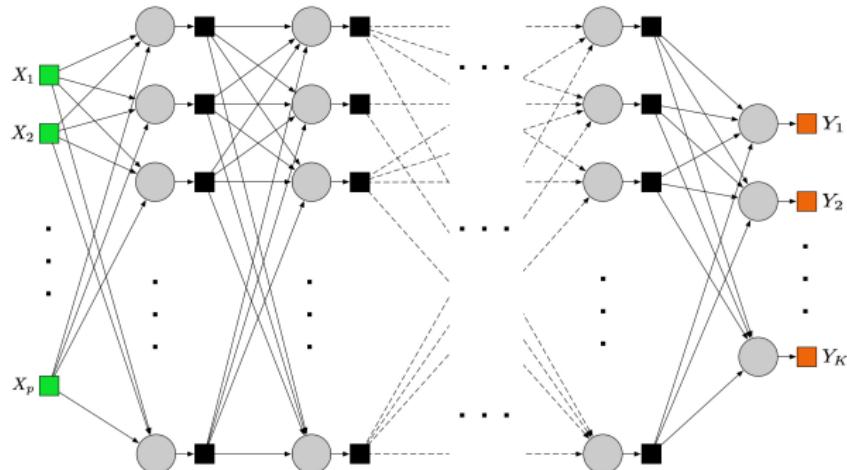
Propagation avant...



Forward

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

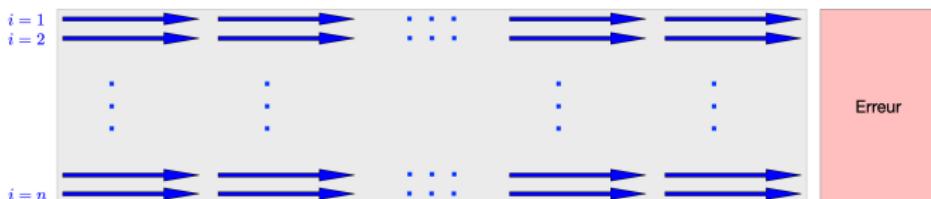
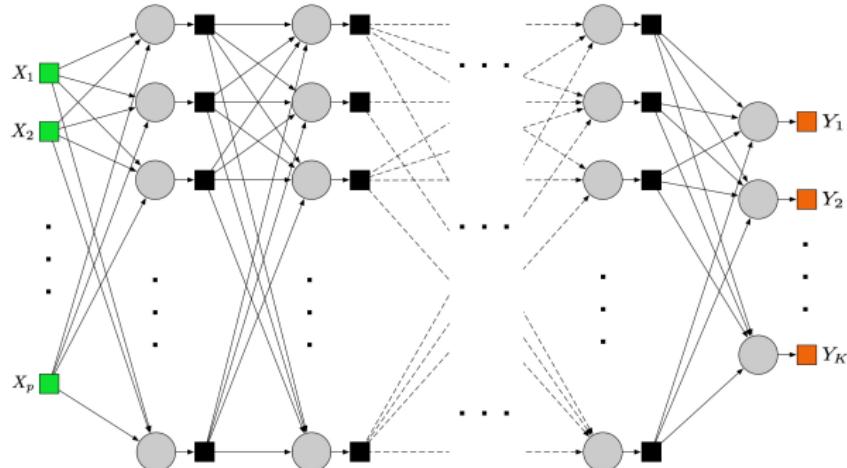
Calcul d'erreur



Forward

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Réto-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

... puis retro-propagation

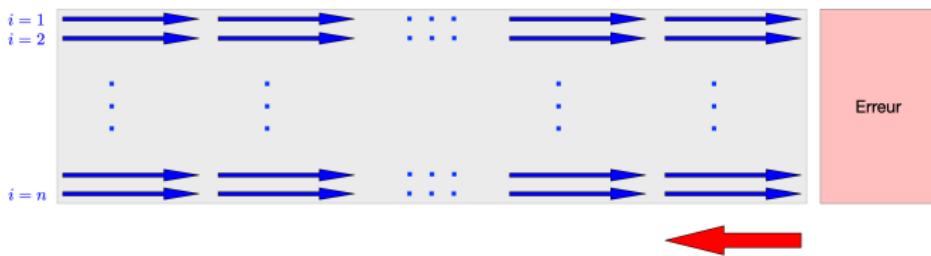
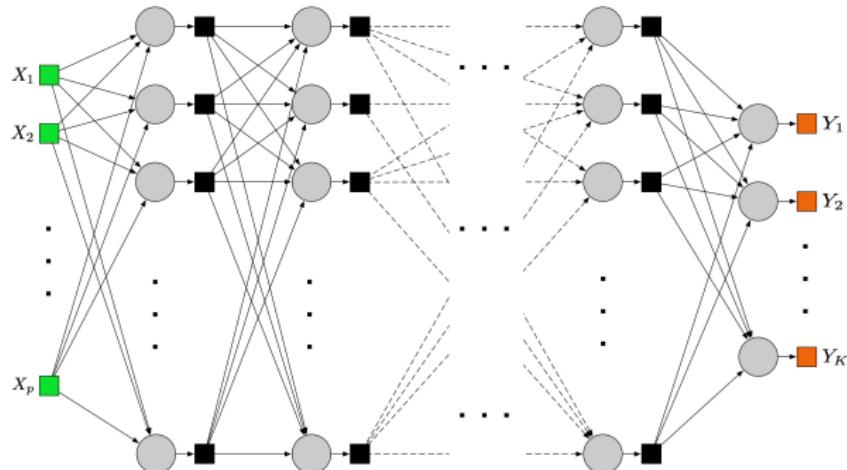


Forward

Backward

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

... puis retro-propagation

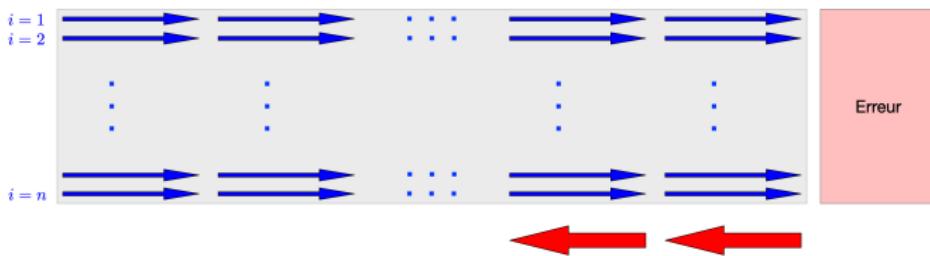
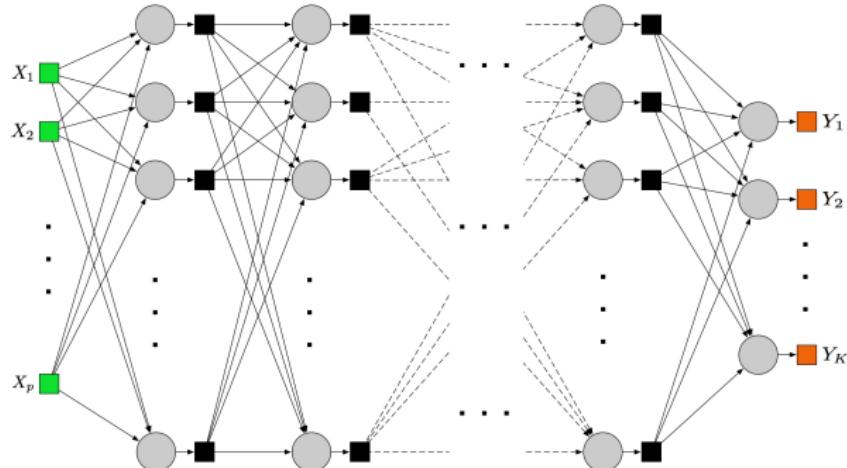


Forward

Backward

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

... puis retro-propagation

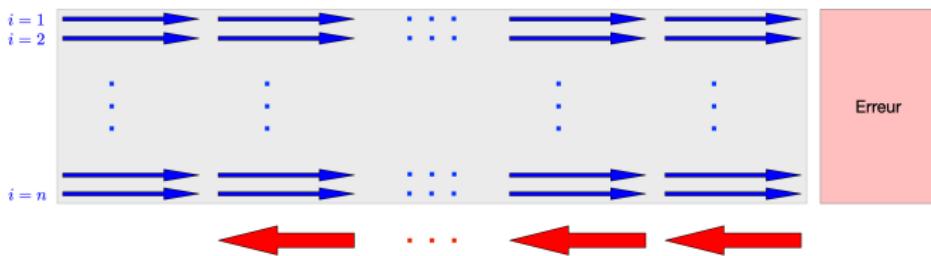
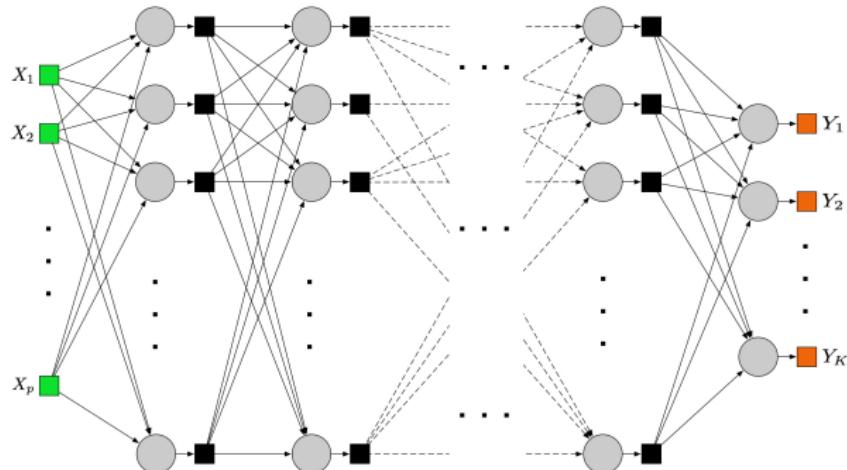


Forward

Backward

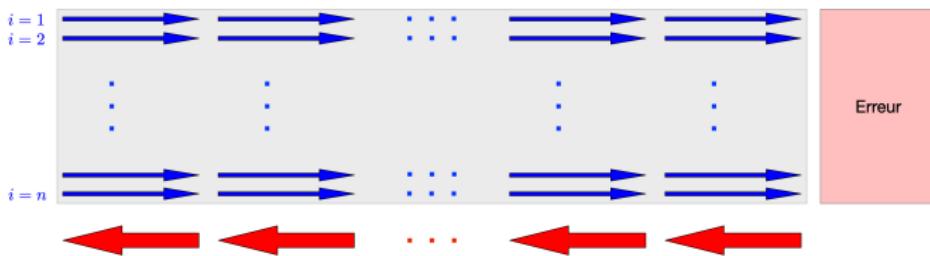
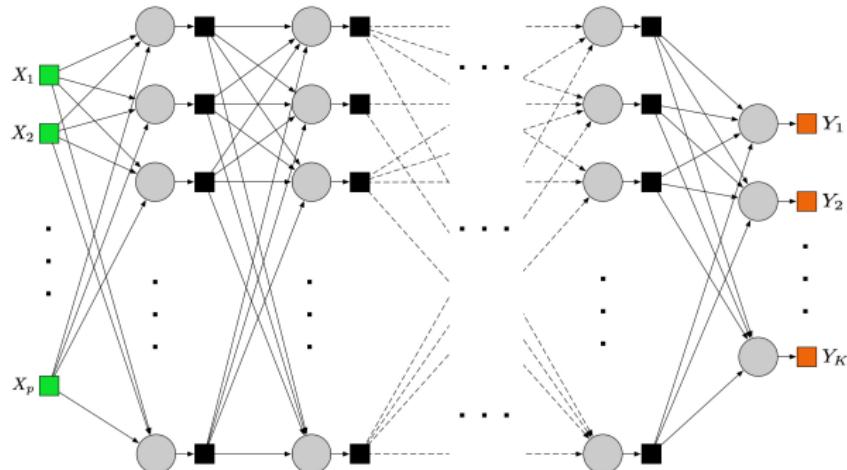
- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

... puis retro-propagation



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

... puis rétro-propagation

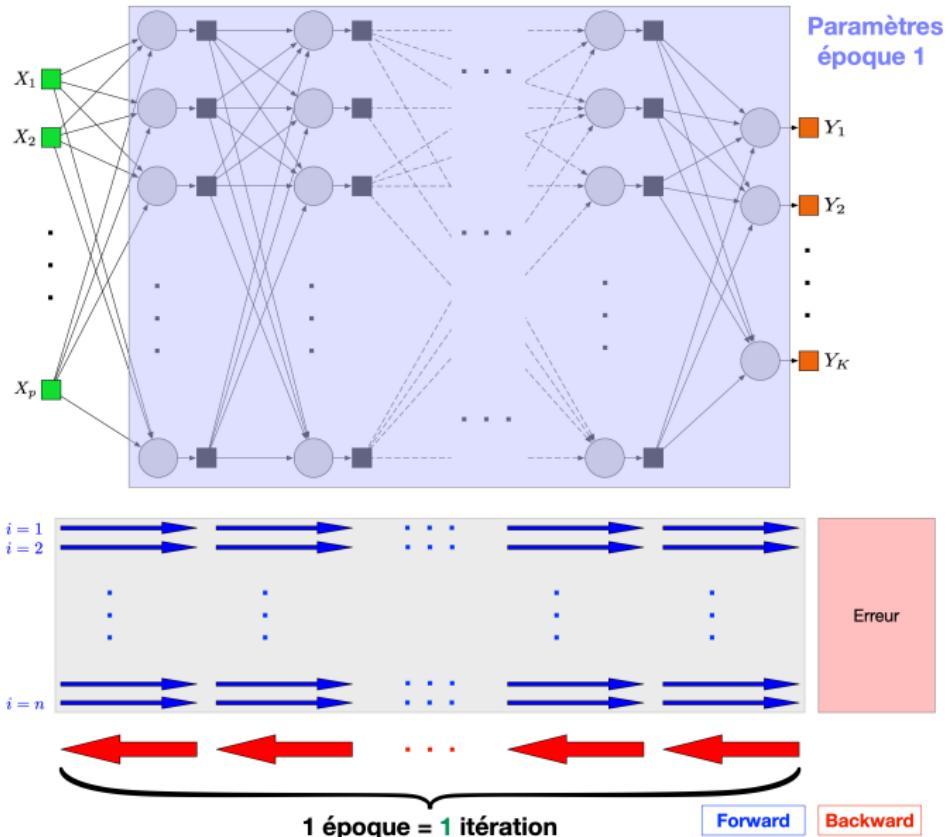


Forward

Backward

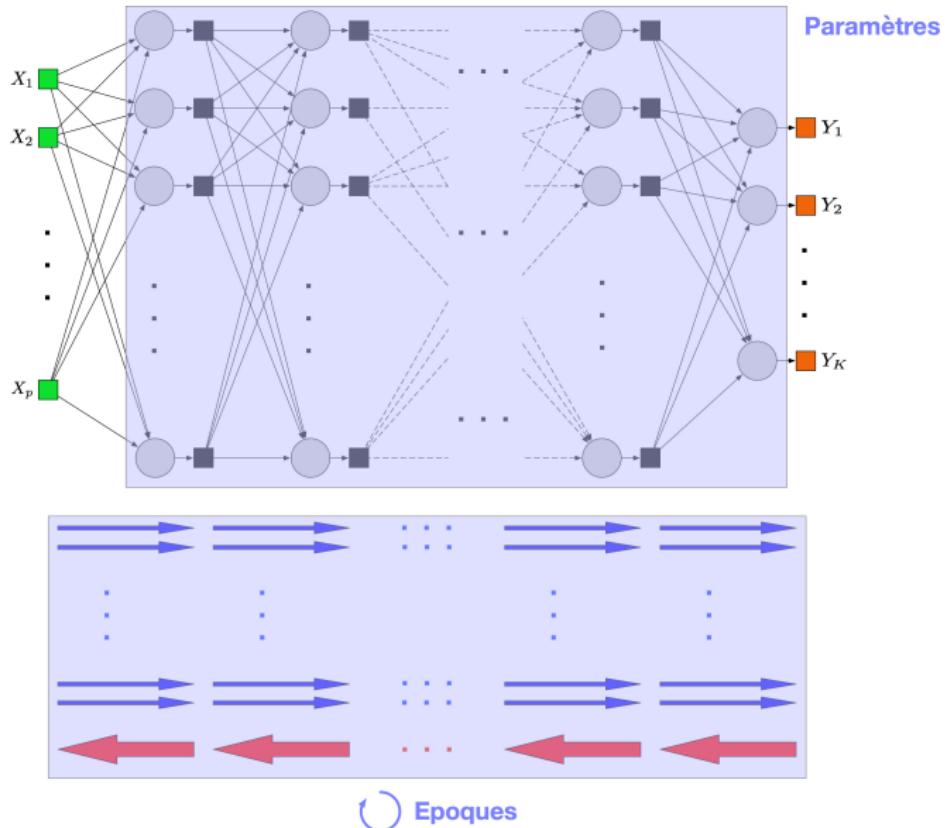
- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

Une époque



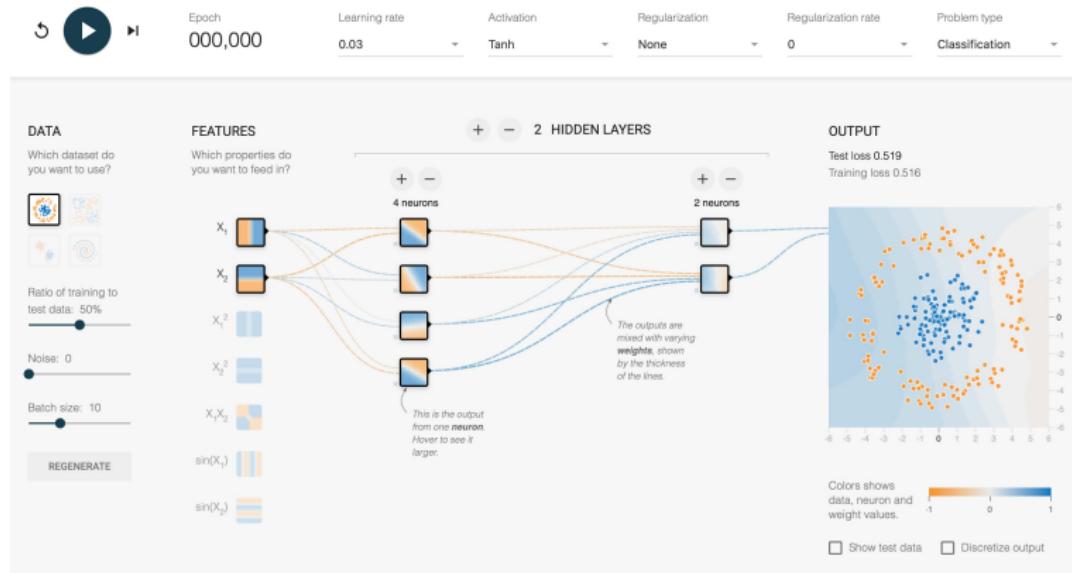
- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Réto-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

Plusieurs époques



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe**
- Taux d'apprentissage
- Architectures classiques
- Pratique des DNN
- Références

Un site pédagogique



<https://playground.tensorflow.org/>

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Principe

Taux d'apprentissage

Architectures classiques

Pratique des DNN

Références

Plan

Rétro-propagation

Principe

Taux d'apprentissage

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Principe

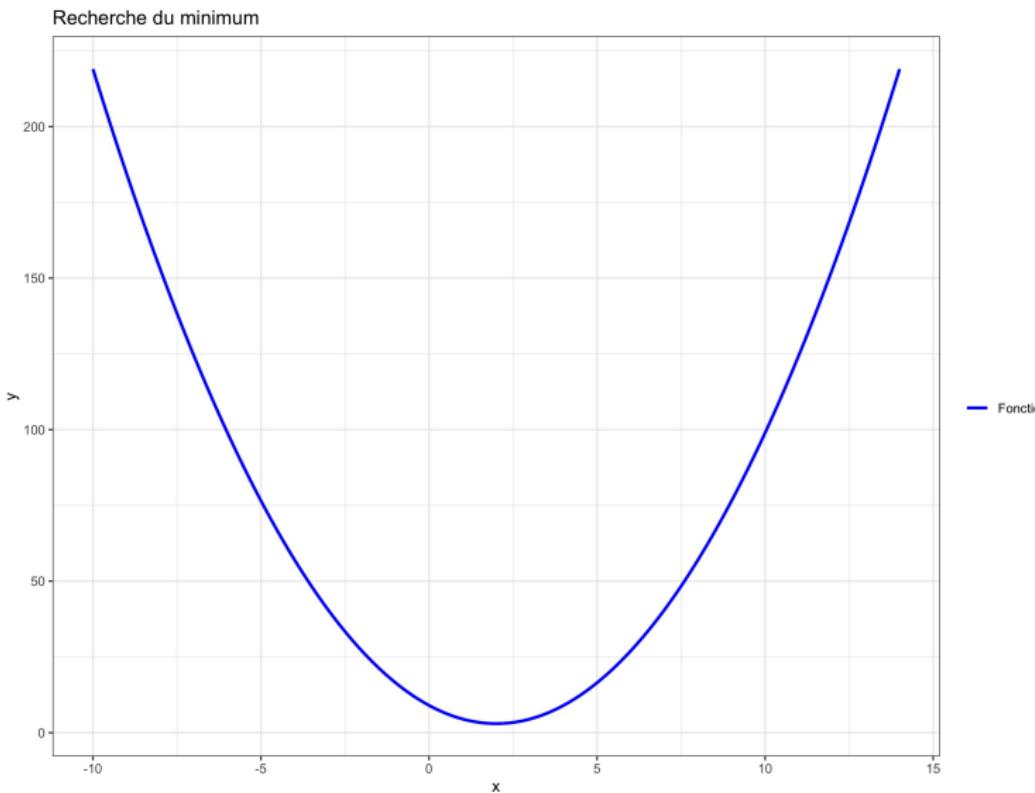
Taux d'apprentissage

Architectures
classiques

Pratique des
DNN

Références

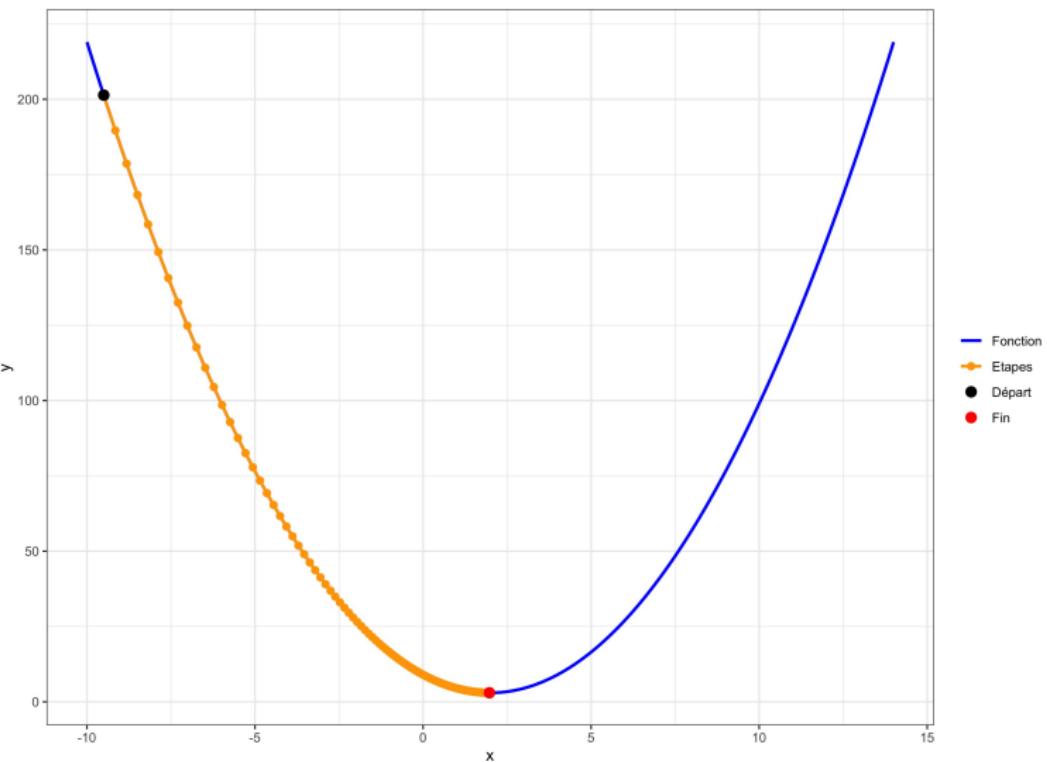
Taux d'apprentissage : exemple 1 |



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Principe
Taux d'apprentissage
Architectures classiques
Pratique des DNN
Références

Taux d'apprentissage : exemple 1 II

Recherche du minimum : $\eta = 0.01$



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Principe

Taux d'apprentissage

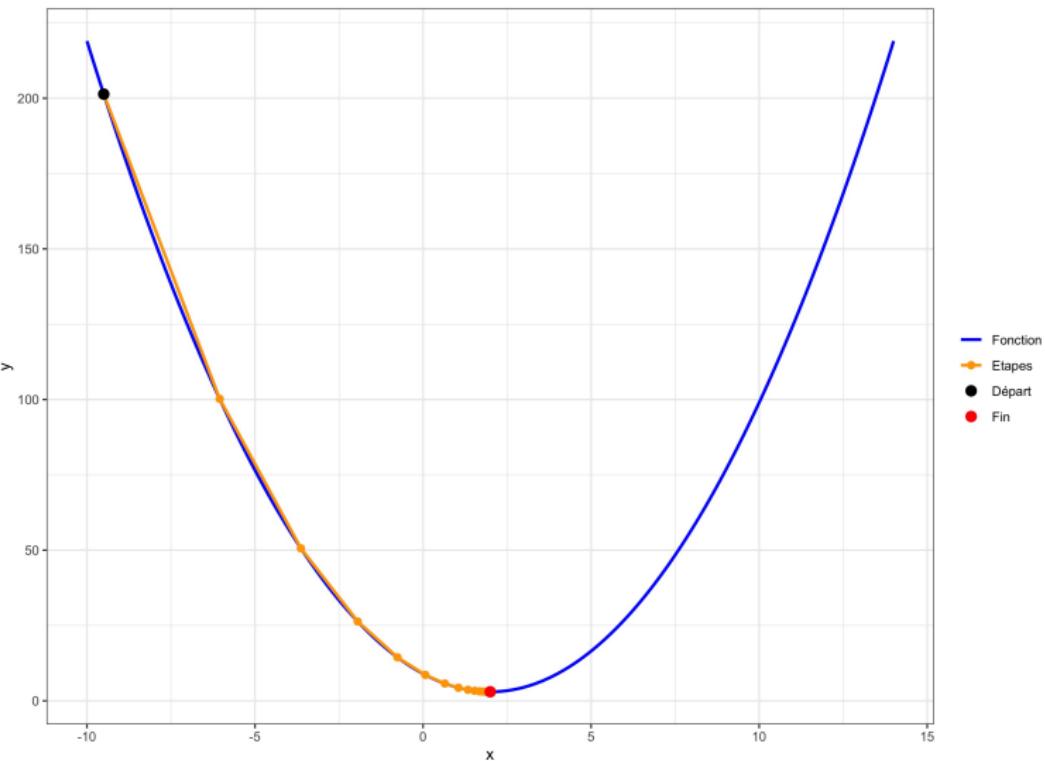
Architectures
classiques

Pratique des
DNN

Références

Taux d'apprentissage : exemple 1 III

Recherche du minimum : $\eta = 0.1$



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Principe

Taux d'apprentissage

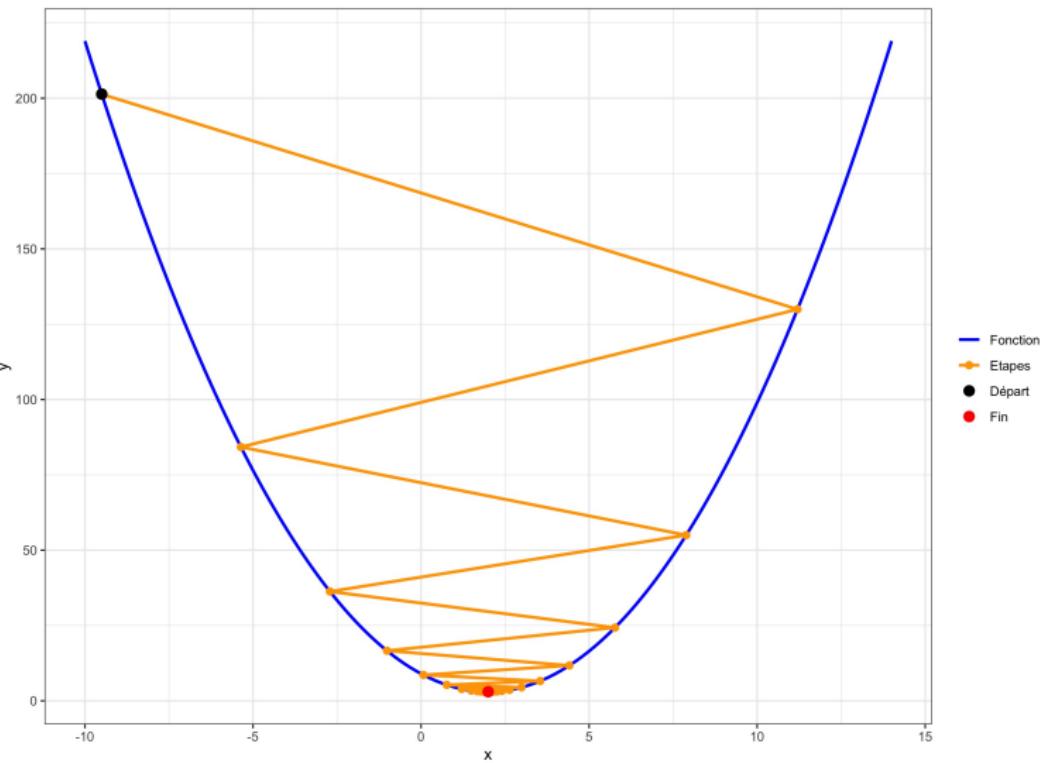
Architectures
classiques

Pratique des
DNN

Références

Taux d'apprentissage : exemple 1 IV

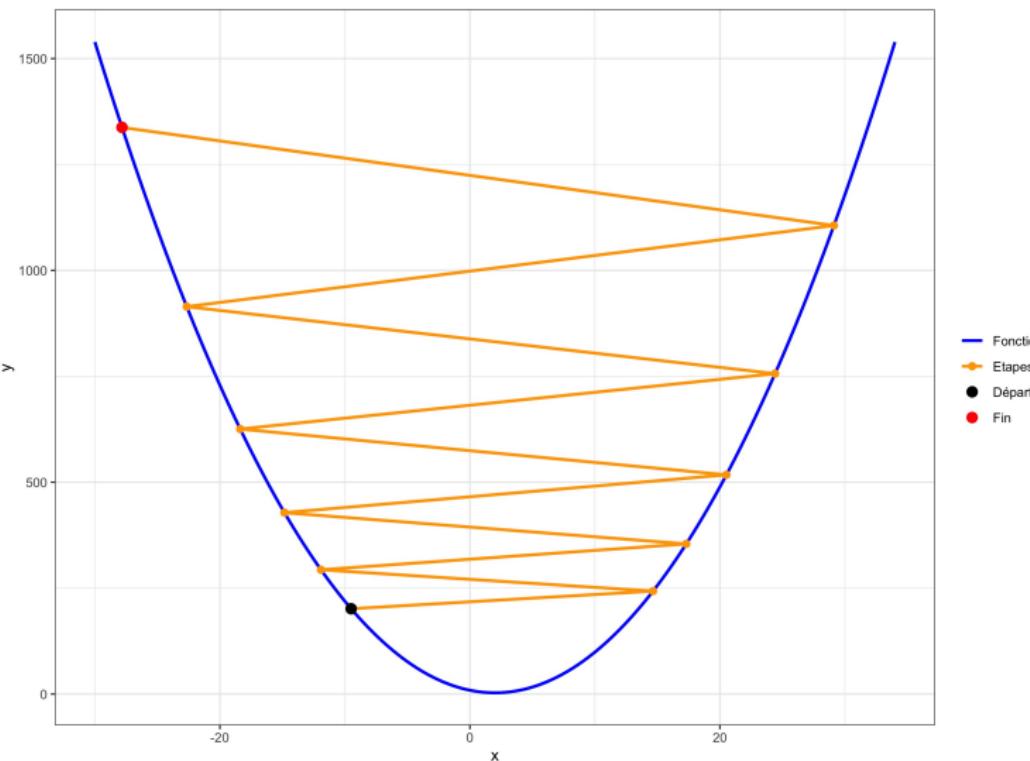
Recherche du minimum : $\eta = 0.6$



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage**
- Architectures classiques
- Pratique des DNN
- Références

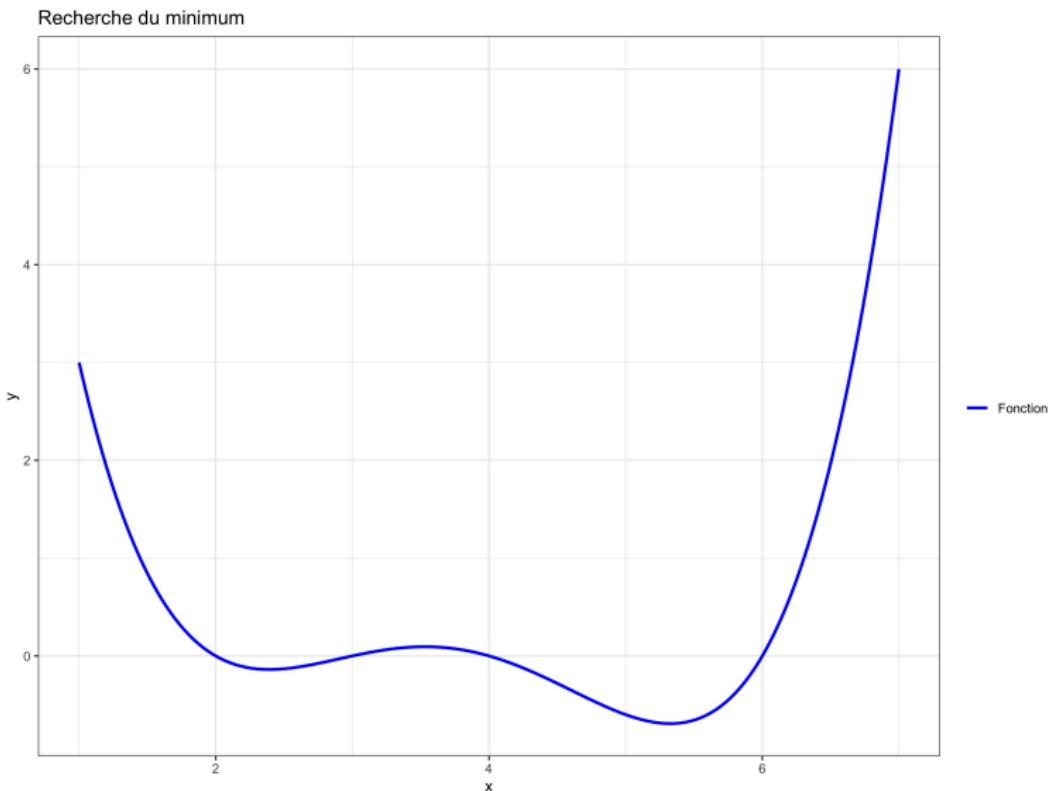
Taux d'apprentissage : exemple 1 V

Recherche du minimum : $\eta = 0.7$



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage**
- Architectures classiques
- Pratique des DNN
- Références

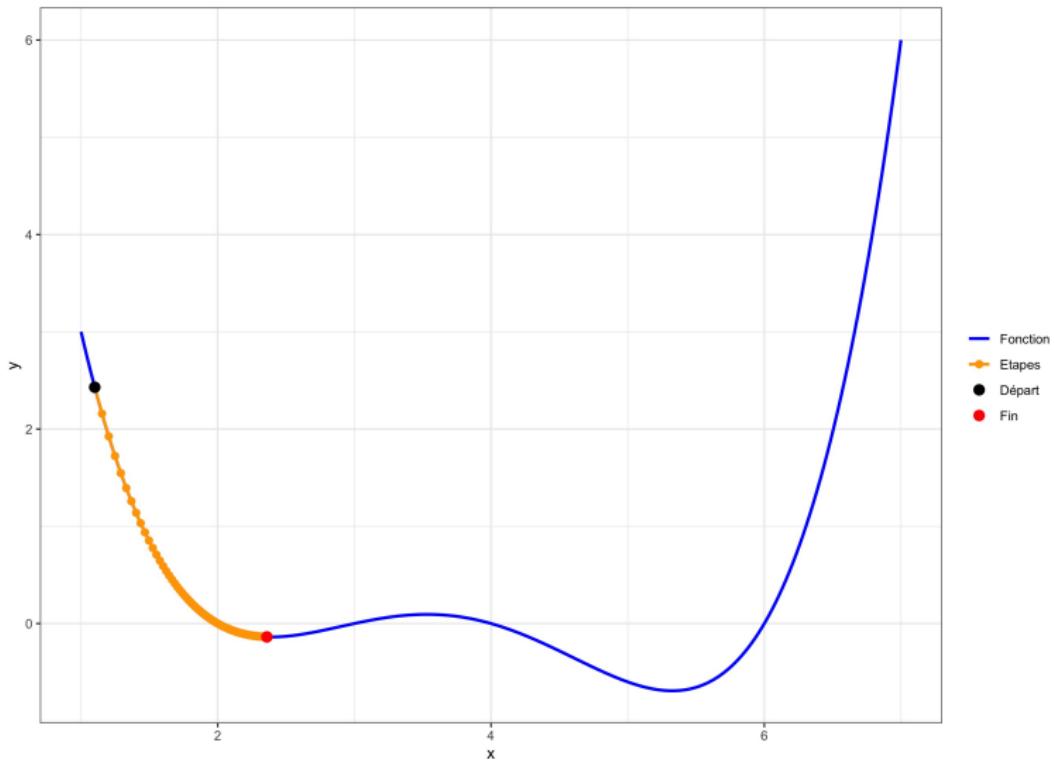
Taux d'apprentissage : exemple 2 I



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Principe
Taux d'apprentissage
Architectures classiques
Pratique des DNN
Références

Taux d'apprentissage : exemple 2 II

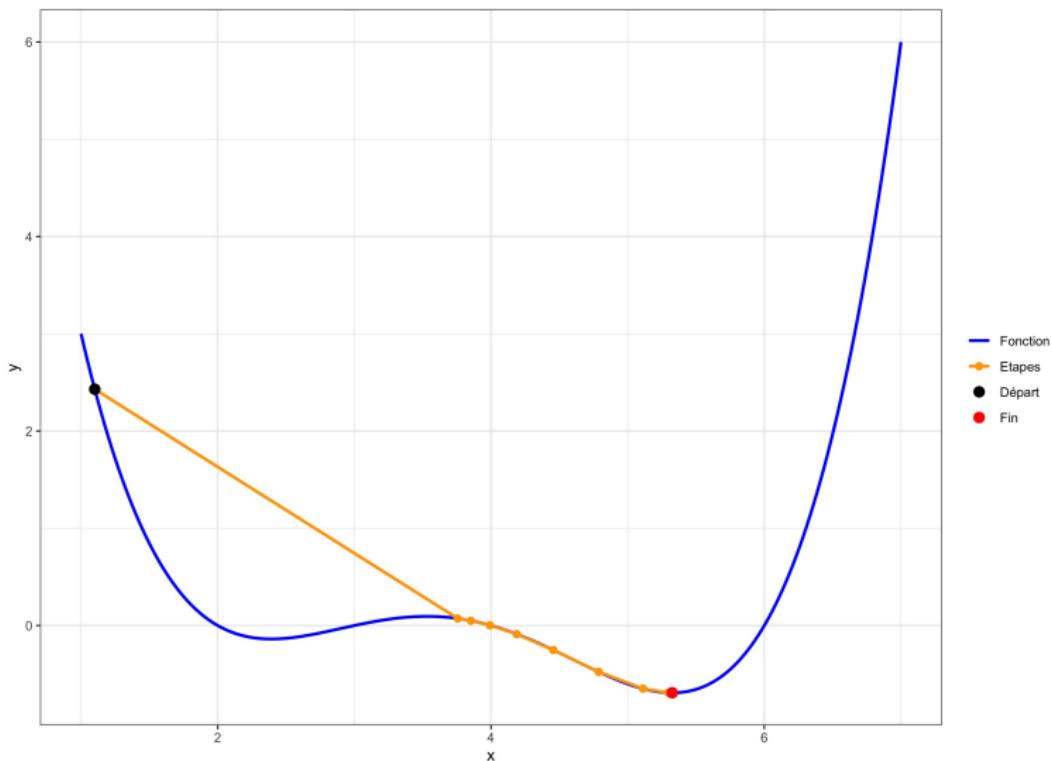
Recherche du minimum : $\eta = 0.01$



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage**
- Architectures classiques
- Pratique des DNN
- Références

Taux d'apprentissage : exemple 2 III

Recherche du minimum : $\eta = 0.5$



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Principe
- Taux d'apprentissage**
- Architectures classiques
- Pratique des DNN
- Références

Plan

Architectures classiques

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

**Architectures
classiques**

Pratique des
DNN

Références

Régression

- ▶ Dans le cas de la régression, les **fonctions d'activation** usuellement retenues sont :
 - ▶ **ReLU** pour les **couches cachées**,
 - ▶ **linéaire** pour la **couche de sortie**.
- ▶ On a, pour $i \in \{1, \dots, n\}$, $y_i \in \mathbb{R}$.
- ▶ A partir d'une entrée $x_i = (x_{i1}, \dots, x_{ip})^\top$ pour $i \in \{1, \dots, n\}$, on obtient la sortie $\hat{y}_i \in \mathbb{R}$.
- ▶ Le risque empirique usuellement retenue est l'**erreur quadratique** (divisée par un facteur 2 pour des commodités d'écriture) :

$$R_n = \frac{1}{2} \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 .$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Classification supervisée binaire

- ▶ Dans le cas de la classification supervisée binaire, les **fonctions d'activation** usuellement retenues sont :
 - ▶ ReLU pour les **couches cachées**,
 - ▶ **sigmoïde** pour la **couche de sortie**.
- ▶ On a, pour $i \in \{1, \dots, n\}$, $y_i \in \{0, 1\}$.
- ▶ A partir d'une entrée $x_i = (x_{i1}, \dots, x_{ip})^\top$ pour $i \in \{1, \dots, n\}$, on obtient la sortie $\hat{y}_i \in [0, 1]$.
- ▶ Le risque empirique usuellement retenue est l'**entropie croisée binaire** :

$$R_n = -\frac{1}{n} \sum_{i=1}^n [y_i \ln (\hat{y}_i) + (1 - y_i) \ln (1 - \hat{y}_i)] .$$

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

Entropie croisée : exemple

y_i	\hat{y}_i	Entropie croisée	Erreur de classification ($s = \frac{1}{2}$)
0	0.05	0.05	0
0	0.45	0.60	0
0	0.55	0.80	1
0	0.95	3.00	1
1	0.05	3.00	1
1	0.45	0.80	1
1	0.55	0.60	0
1	0.95	0.05	0

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Classification supervisée non-binaire

- ▶ Dans le cas d'une classification supervisée à K classes avec $K > 2$, les **fonctions d'activation** usuellement retenues sont :

- ▶ ReLU pour les **couches cachées**,
- ▶ softmax pour la **couche de sortie** :

$$\forall \ell \in \{1, \dots, K\} : y_\ell = \varphi_\ell(a_1, \dots, a_K) = \frac{e^{a_\ell}}{\sum_{k=1}^K e^{a_k}} \in [0, 1]$$

- ▶ On a, pour $i \in \{1, \dots, n\}$ et $k \in \{1, \dots, K\}$,
 $y_{ik} \in \{0, 1\}$ et $\sum_{i=1}^K y_{ik} = 1$.
- ▶ A partir d'une entrée $x_i = (x_{i1}, \dots, x_{ip})^\top$ pour
 $i \in \{1, \dots, n\}$, on obtient K sorties $\hat{y}_{ik} \in [0, 1]$:

$$\hat{k} = \arg \max_{\ell \in \{1, \dots, K\}} \hat{y}_{i\ell} .$$

- ▶ Le risque empirique usuellement retenue est l'**entropie croisée** :

$$R_n = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K y_{ik} \ln(\hat{y}_{ik}) .$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Plan

Pratique des DNN

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

**Pratique des
DNN**

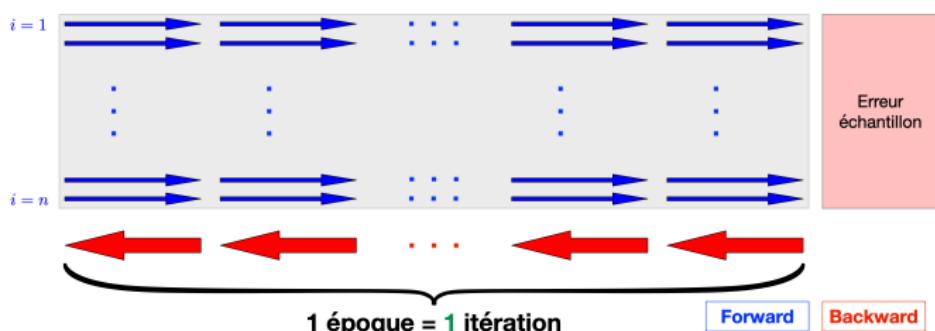
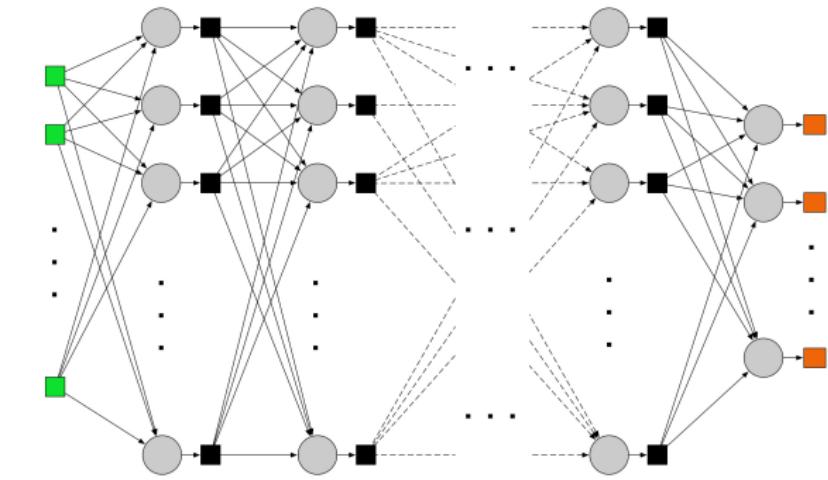
Références

Apprentissage en ligne vs. (par lot) complet

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

- ▶ L'**apprentissage (par lot) complet** (*full batch learning*) considère l'ensemble des observations et corrige les poids (une fois par époque) à partir du risque empirique calculé sur tout l'échantillon.
- ▶ L'**apprentissage en ligne** (*online learning*) considère les observations l'une après l'autre et corrige les poids (n fois par époque) à partir du risque empirique calculé sur l'observation incorporée.

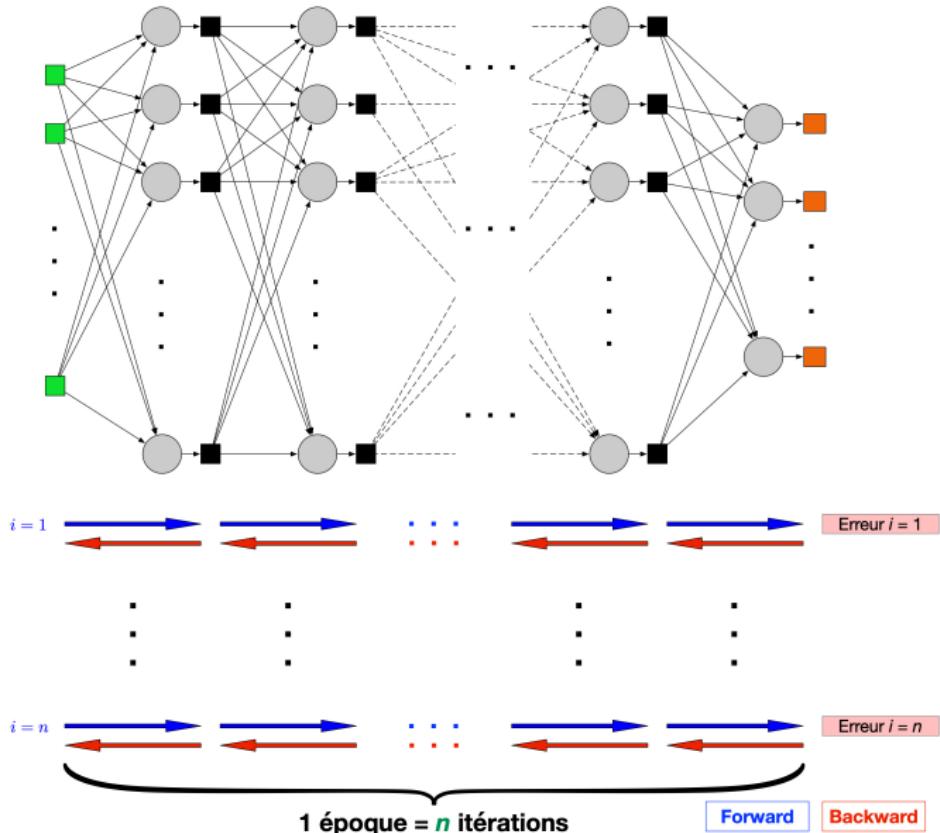
Apprentissage (par lot) complet



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Réto-propagation
- Architectures classiques
- Pratique des DNN

Références

Apprentissage en ligne



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Réto-propagation
- Architectures classiques
- Pratique des DNN
- Références

Motivations pour des (mini-)lots

- ▶ L'apprentissage (par lot) complet est plus stable que l'apprentissage en ligne mais nécessite de charger toutes les données en mémoire.
- ▶ Pour pallier en partie les instabilités de l'apprentissage en ligne, on présente les observations dans un ordre aléatoire à chaque étape (itération).
- ▶ Un compromis, utile face à de gros volumes de données, est l'apprentissage par (mini-)lot ((mini-)batch learning), dans lequel la taille de ces mini-lots est fixée par l'utilisateur.

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

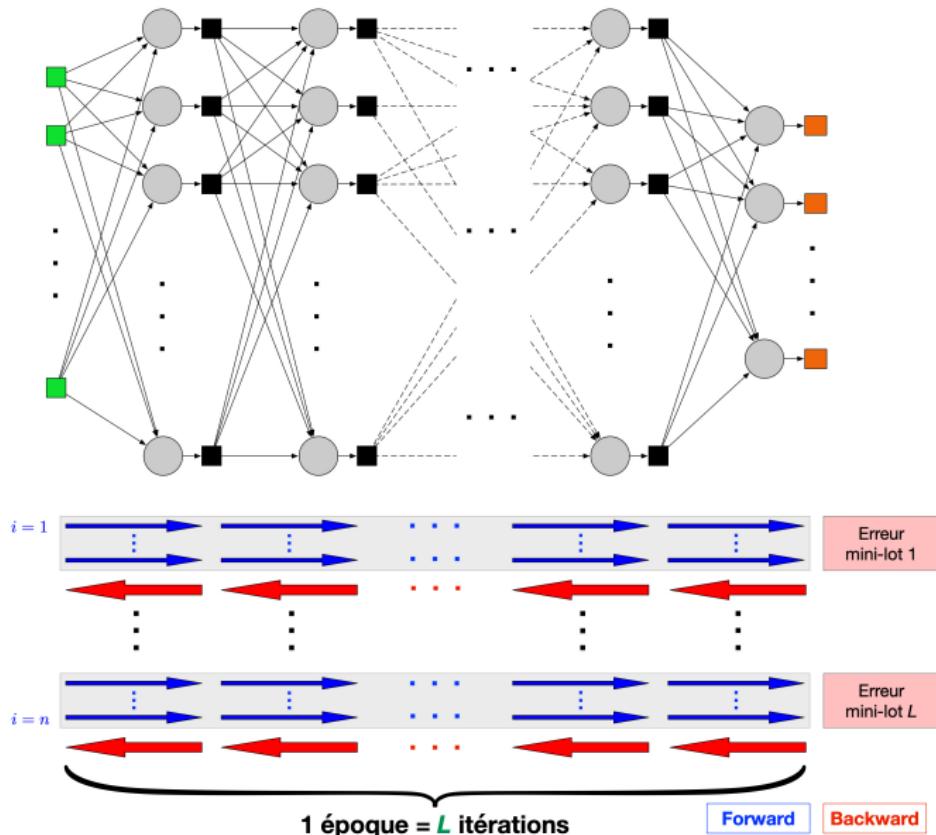
Architectures classiques

Pratique des DNN

Références

Apprentissage par (mini-)lot

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références



Epoque et itérations

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

- ▶ Une **époque** correspond à un passage avant de **tout l'échantillon** à travers le réseau de neurones.
- ▶ Le nombre d'**itérations** est le nombre de lots nécessaires à une époque :
 - ▶ 1 pour l'apprentissage (par lot) complet,
 - ▶ n pour l'apprentissage en ligne,
 - ▶ L pour l'apprentissage par (mini-)lot, où L désigne le nombre de (mini-)lots (contenant approximativement $\lceil \frac{n}{L} \rceil$ observations).

Initialisation des poids et des biais

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

- ▶ Pour débuter la propagation avant, il faut disposer de **valeurs initiales** pour les **poids** et les **biais**.
- ▶ On les initialise à l'aide d'un **tirage aléatoire**, à partir par exemple d'une loi :
 - ▶ $\mathcal{U}(-[\frac{1}{2}, \frac{1}{2}])$,
 - ▶ $\mathcal{N}(0, 1)$,
 - ▶ $\mathcal{N}\left(0, \frac{1}{N_{d-1}}\right)$ pour les neurones de la couche $d \in \{2, \dots, D\}$.

Normalisation des données

- ▶ Pour la plupart des fonctions d'activation, le gradient est quasiment nul pour de faibles ou grandes valeurs en entrées, ce qui conduit au mieux à un apprentissage très lent. On parle alors de **saturation** des neurones.
- ▶ On effectue à cet effet une **normalisation des covariables** via :

- ▶ Un **centrage-réduction** pour des covariables quantitatives :

$$X_j^* = \frac{X_j - \bar{X}_j}{s_{X_j}} .$$

- ▶ Une **normalisation min-max** pour les pixels d'une image :

$$X_j^* = \frac{X_j}{255} .$$

- ▶ On applique aux données de validation et de test la même normalisation que celle appliquée sur les données d'apprentissage (les mêmes paramètres!).

Astuce dans le cas de la classification supervisée binaire

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

- ▶ Dans le cas de la **classification supervisée binaire**, on préfère souvent considérer comme valeurs de sortie $\{0.05, 0.95\}$ plutôt que $\{0, 1\}$, toujours dans l'idée de pallier ce problème de saturation.

A propos de l'optimisation I

- ▶ Le risque empirique considéré dans un perceptron multi-couche admet de nombreux **minima locaux** dans lesquels il est aisé de « tomber » si on n'y prête pas attention.
- ▶ On observe des phénomènes de **disparition du gradient** (*vanishing gradient*), ou à l'inverse, d'**explosion du gradient** (*exploding gradient*), dans les couches basses du réseau de neurones.
- ▶ Afin de pallier certains de ces problèmes, il est possible :
 - ▶ de choisir une **fonction d'activation adéquate** (ex : ReLU),
 - ▶ d'adopter un **taux d'apprentissage adaptatif**,
 - ▶ d'ajouter un terme d'inertie ou **momentum** dans la correction des poids : à chaque itération on modifie les poids en tenant compte en sus des changements de poids effectués à l'itération précédente.

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

A propos de l'optimisation II

- ▶ L'évaluation d'un gradient de somme peut être très coûteuse dans le cas d'un très grand ensemble d'apprentissage.
- ▶ La méthode de **descente de gradient stochastique** diminue le coût de calcul de chaque étape en tirant un échantillon aléatoire de l'ensemble des fonctions (donc un échantillon aléatoire des données d'apprentissage).
- ▶ **Adam** (*Adaptive Moment Estimation*) est une méthode de descente stochastique du gradient comprenant une méthode de taux d'apprentissage adaptatif et la méthode du momentum.
(Kingma et Ba, 2017)

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

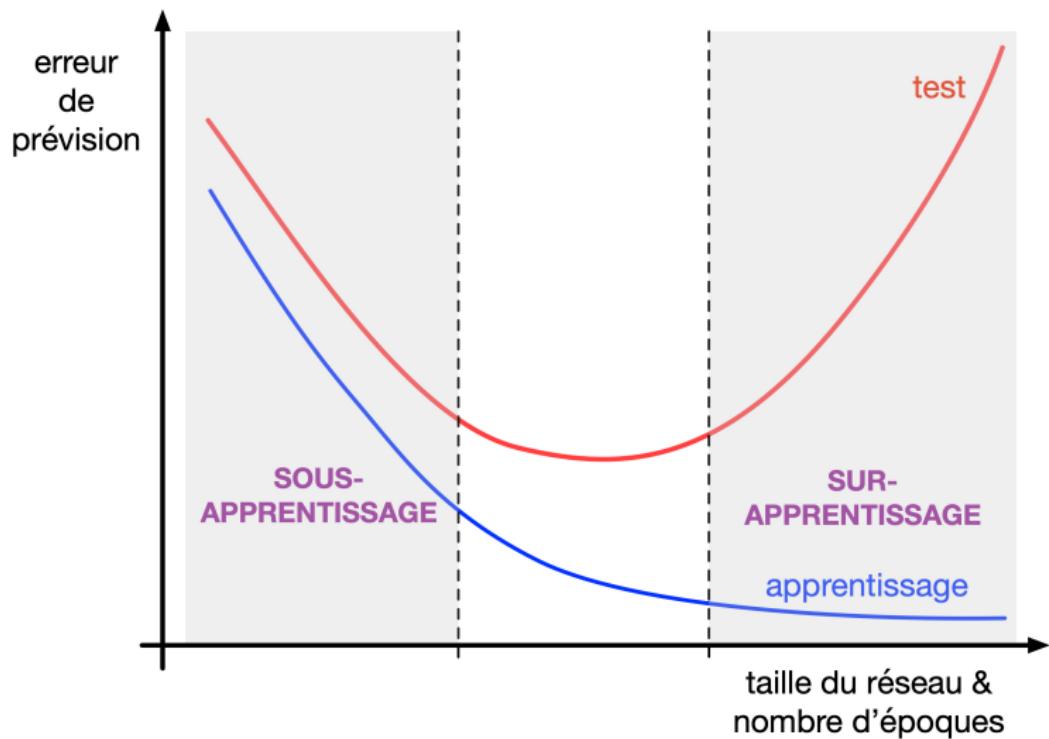
Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

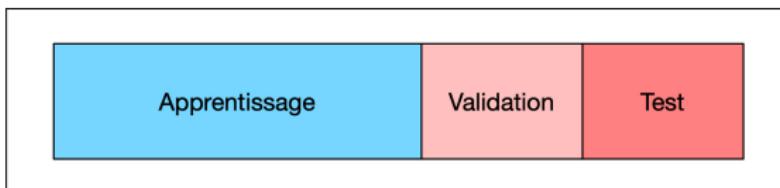
A propos du sur-apprentissage I



- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références

A propos du sur-apprentissage II

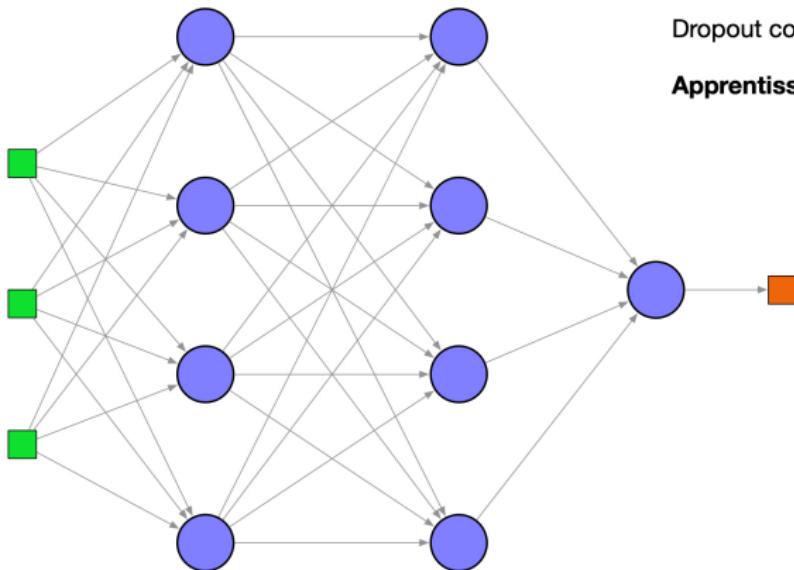
- ▶ Afin de limiter le risque de **sur-apprentissage**, il faut disposer d'un échantillon de validation et contrôler :
 - ▶ Le **nombre de couches et de neurones**.
 - ▶ Le **nombre d'époques**.



- ▶ Il est possible d'utiliser les techniques suivantes :
 - ▶ **Dégénération des poids (*weight decay*)** : régularisation du risque empirique avec une pénalité ridge ($\lambda \sum_{\ell} \omega_{\ell}^2$).
 - ▶ **Inhibition de neurones (*drop out*)** : certains neurones sont temporairement supprimés durant la rétro-propagation (avec une probabilité donnée).

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

Dropout I

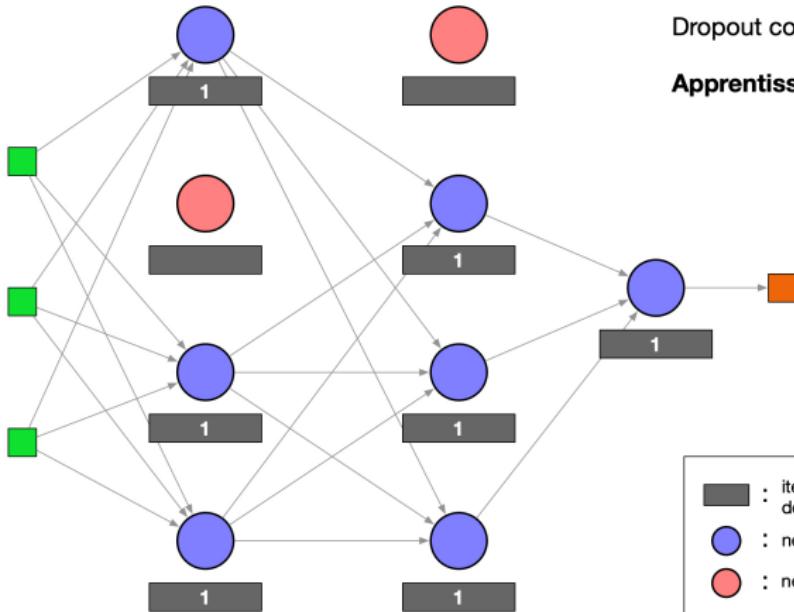


Dropout couches 1 & 2 : $p = \frac{1}{4}$

Apprentissage : initialisation

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références

Drop out II



Dropout couches 1 & 2 : $p = \frac{1}{4}$

Apprentissage : itération 1

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

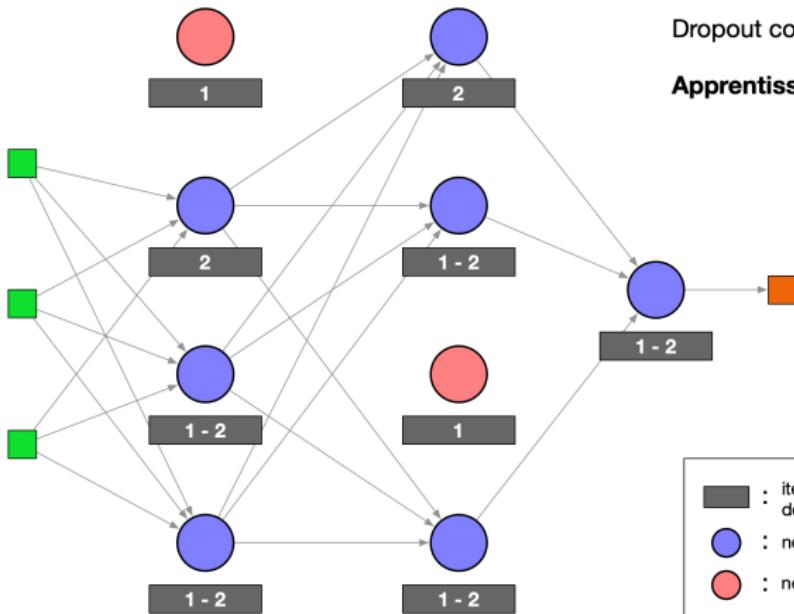
Rétro-propagation

Architectures classiques

Pratique des DNN

Références

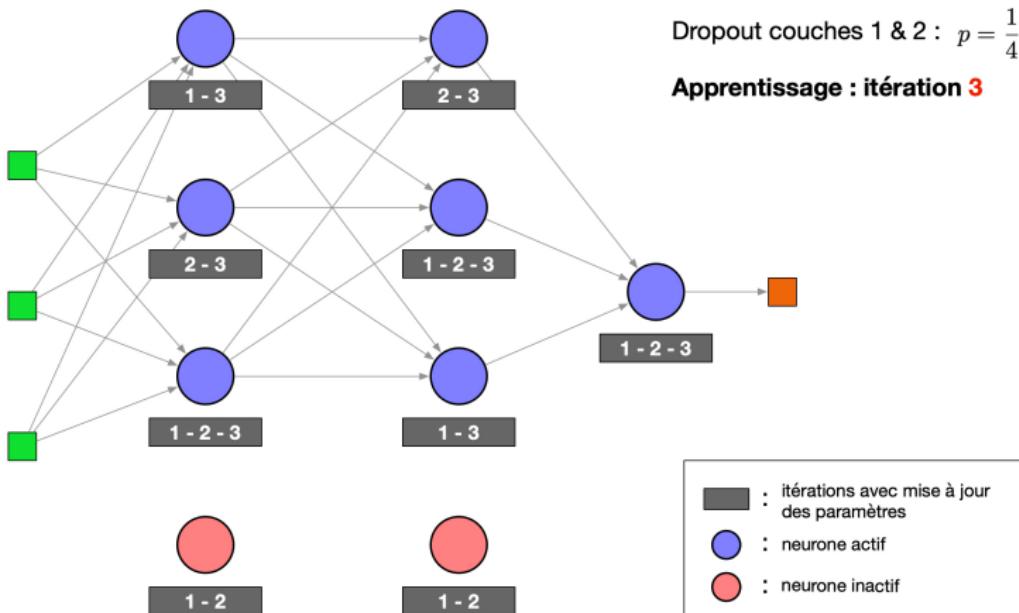
Dropout III



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références

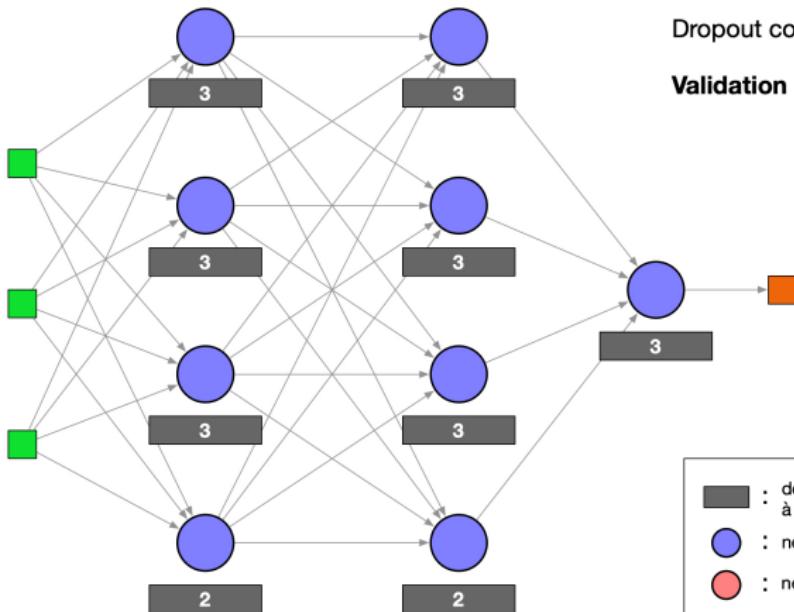
Dropout IV

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Architectures classiques
Pratique des DNN
Références



Dropout V

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Architectures classiques
- Pratique des DNN
- Références



Choix du nombre de couches et de neurones

- ▶ Les nombres de couches et de neurones sont les **hyperparamètres** des réseaux de neurones profonds.
- ▶ Il n'existe pas de solution analytique ou d'heuristique pour déterminer le nombre optimal de couches et de neurones.
- ▶ Expérimentalement, (Goodfellow et collab., 2016) recommandent de privilégier les réseaux de neurones profonds (*deep*) aux réseaux de neurones gras (*fat*).
- ▶ Il faut **expérimenter différentes architectures** en faisant varier le nombre de couches et de neurones tout en gardant à l'esprit le coût calculatoire.
- ▶ Si le *transfer learning* n'est pas adapté aux données structurées (contrairement aux images et aux textes), il peut s'avérer intéressant de tester un nombre de couches et de neurones semblable à celui de réseaux de neurones performants.

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

Sous Python

- ▶ Google



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Architectures
classiques

Pratique des
DNN

Références

- ▶ Meta



Moyens de calculs : CPU, GPU and TPU



Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

► **CPU** : Central Processing Unit.

- ▶ Nombre d'opérations par cycle : dizaines.
- ▶ Usage : programmation générale.
- ▶ Fabricants : Intel....

► **GPU** : Graphics Processing Unit.

- ▶ Nombre d'opérations par cycle : dizaines de milliers.
- ▶ Usage : apprentissage et inférence de modèles de machine learning (parallélisation).
- ▶ Fabricants : NVIDIA....

► **TPU** : Tensor Processing Unit.

- ▶ Nombre d'opérations par cycle : jusqu'à 128 000.
- ▶ Usage : apprentissage et inférence de modèles de machine learning dans TensorFlow.
- ▶ Fabricant : Google.

Références |

- Arias, S., E. Maldonado et J.-L. Parouty. 2022, «Fidle», URL <https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle/-/wikis/home>.
- Bishop, C. M. 1995, *Neural networks for pattern recognition*, Advanced Texts in Econometrics, Oxford University Press.
- Bishop, C. M. 2011, *Pattern recognition and machine learning*, Information Science and Statistics, Springer.
- Boyd, S. et L. Vandenberghe. 2003, *Convex optimization*, Cambridge University Press.
- Chollet, F. 2020, *L'apprentissage profond avec Python*, Machinelearning.fr.
- Chollet, F. 2021, *Deep learning with Python*, 2^e éd., Manning.
- Cybenko, G. 1989, «Approximation by superpositions of a sigmoidal function», *Mathematics of Control, Signals and Systems*, vol. 2, n° 4, p. 303–314.

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Références II

Gagné, C. 2022, «Introduction à l'apprentissage automatique», URL
[https://chgagne.github.io/iaa-ulaval/.](https://chgagne.github.io/iaa-ulaval/)

Goodfellow, I., Y. Bengio et A. Courville. 2016, *Deep learning*, Adaptive Computation and Machine Learning, MIT Press.

Hornik, K. 1991, «Approximation capabilities of multilayer feedforward networks», *Neural Networks*, vol. 4, n° 2, p. 251–257.

Kingma, D. P. et J. Ba. 2017, «Adam : a method for stochastic optimization», *arXiv preprint arXiv :1412.6980*.

Le Cun, Y., B. Boser, J. S. Denker, D. Henderson, W. Hubbard et L. D. Jackel. 1989, «Backpropagation applied to handwritten zip code recognition», *Neural computation*, vol. 1, n° 4, p. 541–551.

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références

Références III

- McCulloch, W. et W. Pitts. 1943, «A logical calculus of ideas immanent in nervous activity», *Bulletin of Mathematical Biophysics*, vol. 5, p. 115–133.
- Rosenblatt, F. 1958, «The perceptron : probabilistic model for information storage and organization in the brain», *Psychological Review*, vol. 65, n° 6, p. 386–408.
- Rumelhart, D. E., G. E. Hinton et R. J. Williams. 1986, «Learning representations by back-propagating errors», *Nature*, vol. 323, p. 533–536.
- Werbos, P. 1974, *Beyond regression : new tools for prediction and analysis in the behavioral sciences*, thèse de doctorat, Harvard University.

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Architectures classiques

Pratique des DNN

Références