

Deep learning

DNN

Vincent Lefieux

- Introduction
- Neurone formel
- Fonctions d'activation
- Perceptron multicouche
- Rétro-propagation
- Pratique du perceptron multicouche
- Références



Plan

Introduction

Introduction

Neurone formel

Neurone formel

Fonctions d'activation

Fonctions
d'activation

Perceptron multicouche

Perceptron
multicouche

Rétro-propagation

Rétro-
propagation

Pratique du perceptron multicouche

Pratique du
perceptron
multicouche

Références

Plan

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Cognitivisme et connexionnisme

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

L'IA a été influencée par des courants de pensée, issus de la cybernétique, parmi lesquels :

- ▶ Le **cognitivisme** : il fait l'analogie entre la pensée et un processus de traitement de l'information. Il s'agit d'une approche par réduction (« top down »). Ce paradigme domine les sciences cognitives du milieu des années 1950 aux années 1980.
- ▶ Le **connexionnisme** : il modélise la pensée ou le comportement comme un processus issu de réseaux d'unités simples interconnectées. Il s'agit d'une approche systémique (« bottom up »). Ce paradigme supplante le cognitivisme dans les années 1980.

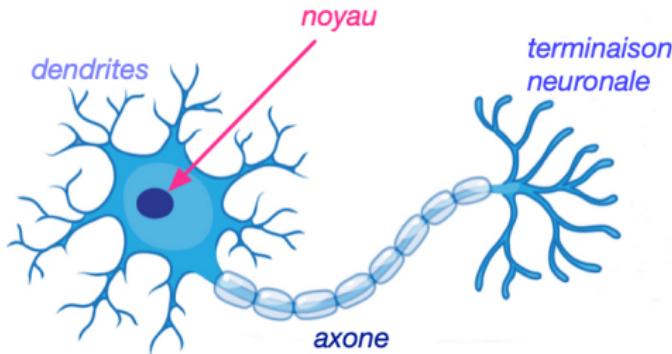
Point de vue biologique I

- Un **réseau de neurones** (artificiels) est un algorithme inspiré du fonctionnement des neurones du **cerveau humain**.



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

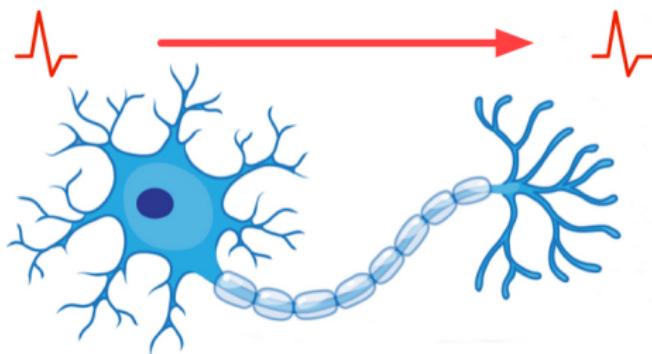
Rétro-
propagation

Pratique du
perceptron
multicouche

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

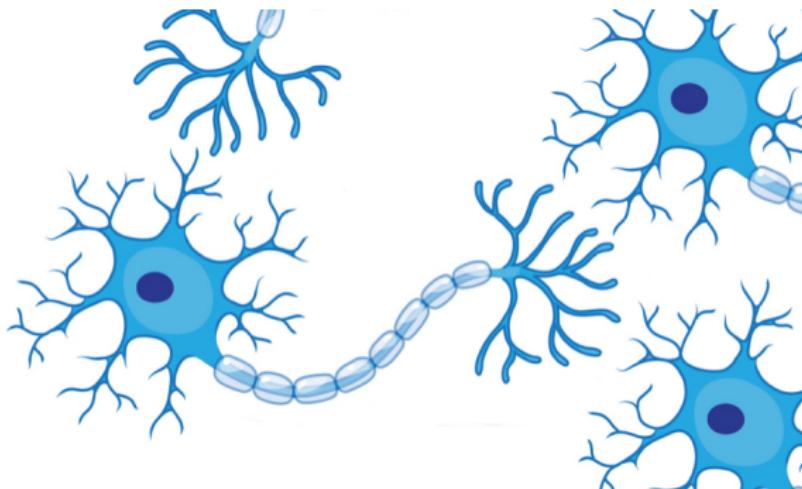
Rétro-
propagation

Pratique du
perceptron
multicouche

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

Pratique du
perceptron
multicouche

Références

Les réseaux de neurones (artificiels) : éléments d'histoire I

- ▶ 1943 : **neurone formel** : McCulloch (neurophysiologiste) et Pitts (logicien).
(McCulloch et Pitts, 1943)
- ▶ 1958 : **perceptron (simple)** : Rosenblatt.
(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

Pratique du perceptron multicouche

Références

Les réseaux de neurones (artificiels) : éléments d'histoire II

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références



De gauche à droite, Trenchard More, John McCarthy, Marvin Minsky, Oliver Selfridge et Ray Solomonoff, présents lors de la conférence de Dartmouth sur l'IA en 1956 (2006).

Les réseaux de neurones (artificiels) : éléments d'histoire III

- ▶ Une histoire avec des « **hivers** »...
- ▶ ...des printemps
 - ▶ de gros volumes de **données** (*big data*) labellisées,
 - ▶ des **moyens de calculs** puissants (ex : GPU),
 - ▶ des **algorithmes** plus performants.

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Les réseaux de neurones (artificiels) : éléments d'histoire IV

Introduction

Neurone formel

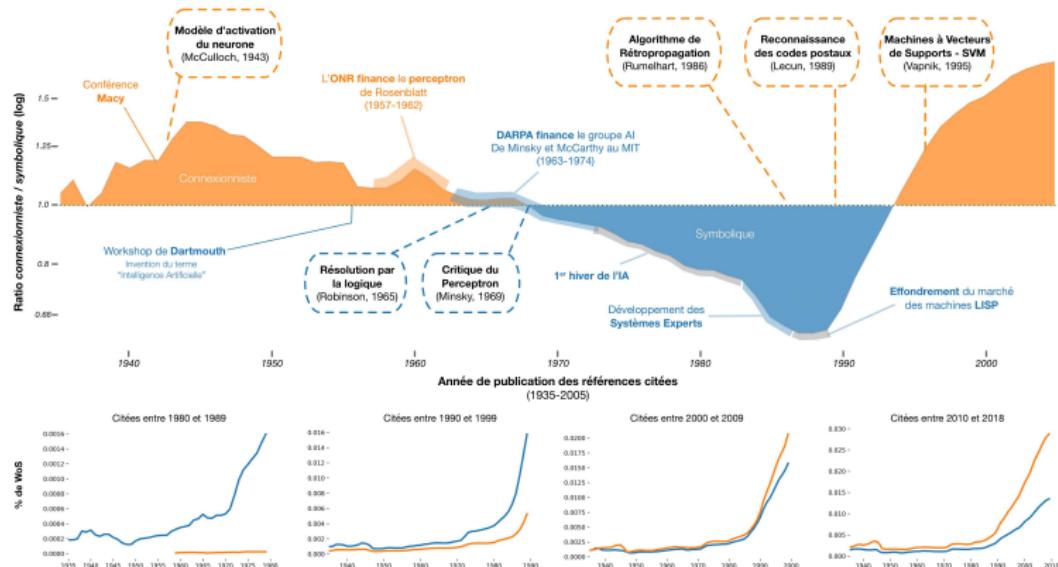
Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références



(Cardon et collab., 2018)

Plan

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

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 :

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

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

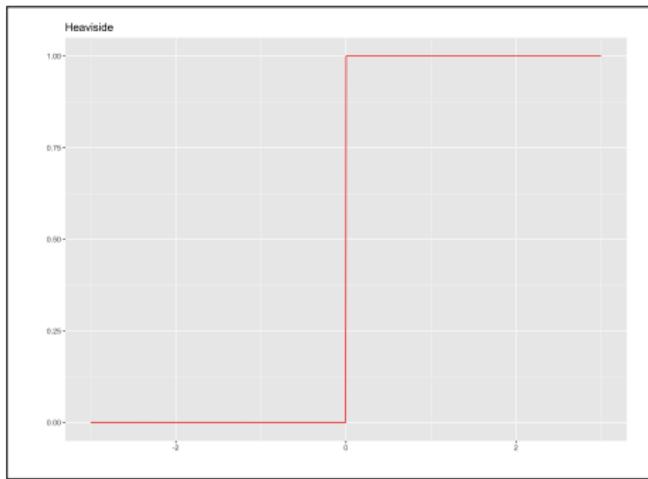
Pratique du perceptron multicouche

Références

Fonction d'activation d'Heaviside

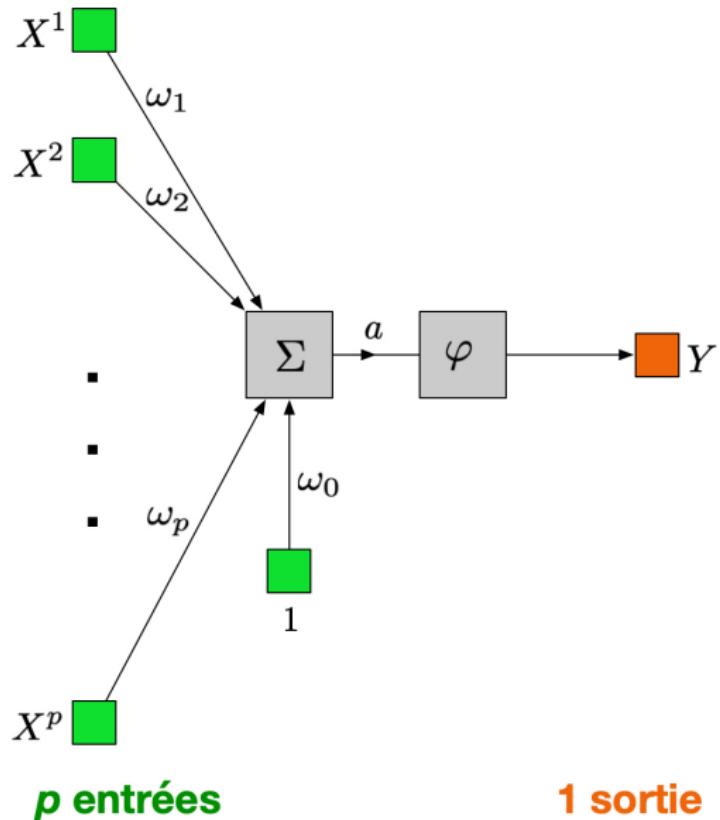
- ▶ Dans la première version du neurone formel, la fonction d'activation retenue était celle de **Heaviside** (ou échelon unité ou marche - *step*) :

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



- ▶ 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).

Représentation I



Introduction

Neurone formel

Fonctions
d'activation

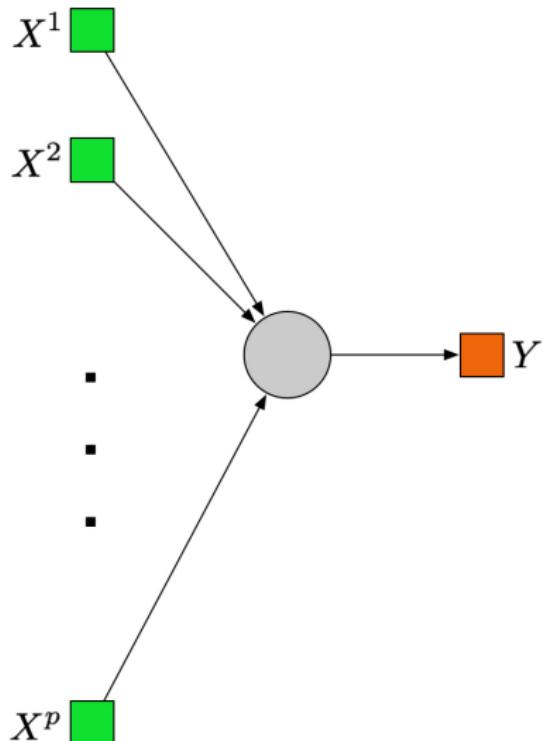
Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Représentation II



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Perceptron (simple) I

- 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.

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

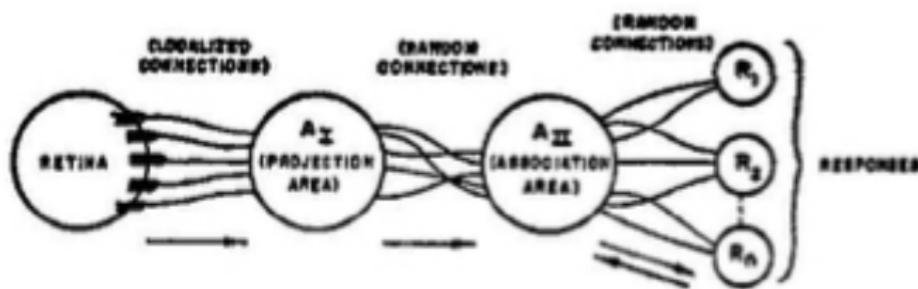
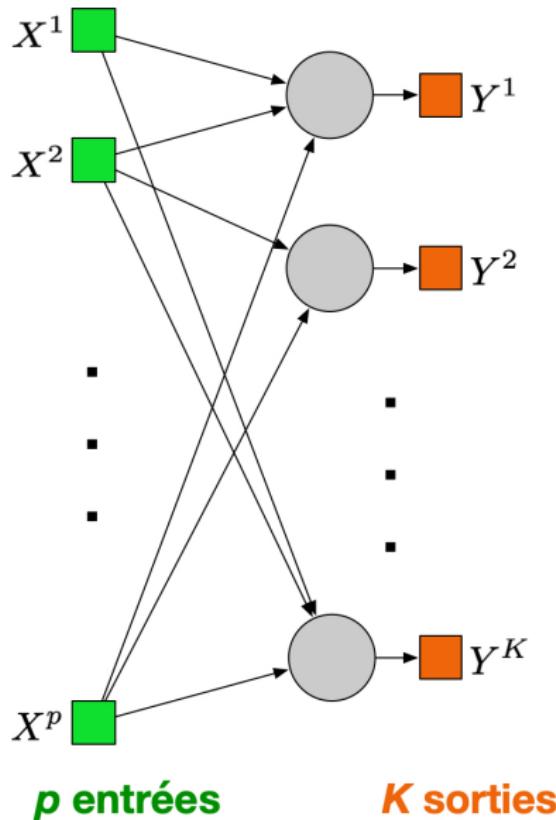


FIG. 1. Organization of a perceptron.

(Rosenblatt, 1958)

Perceptron (simple) II



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Plan

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Introduction

Neurone formel

**Fonctions
d'activation**

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Choix d'une fonction d'activation

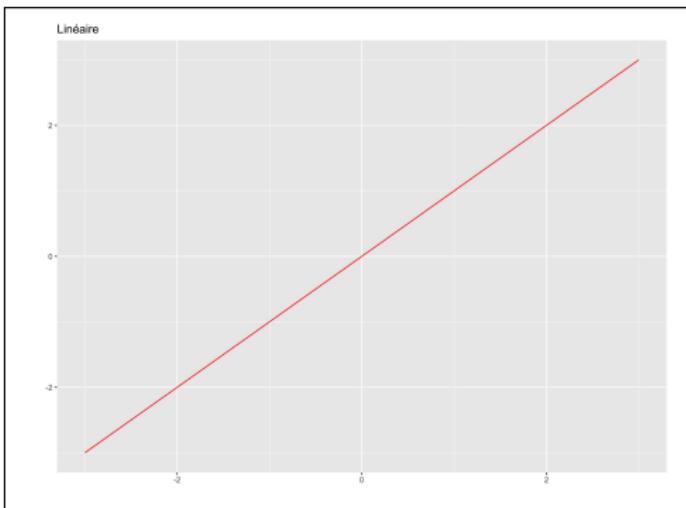
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

- ▶ 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 **sigmoïde**, la tangente hyperbolique et **ReLU**.
- ▶ Les **fonctions d'activation polynomiales** (incluant la **fonction d'activation linéaire**) limitent la portée du neurone, on les considère très peu en pratique.

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

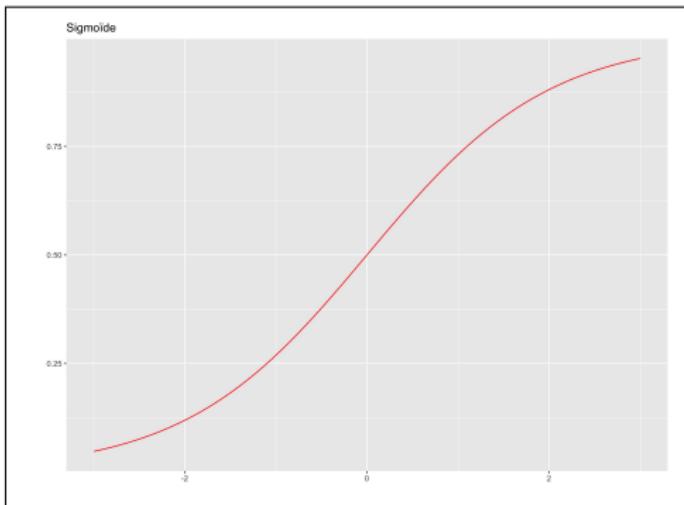
Pratique du perceptron multicouche

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

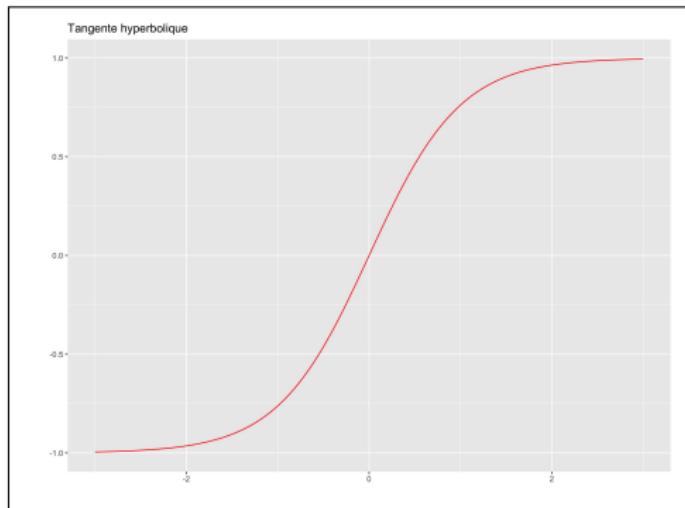
Pratique du perceptron multicouche

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

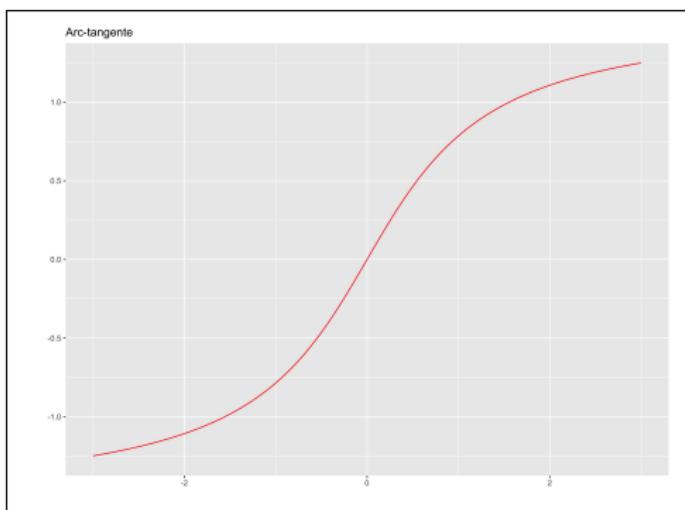
Pratique du perceptron multicouche

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

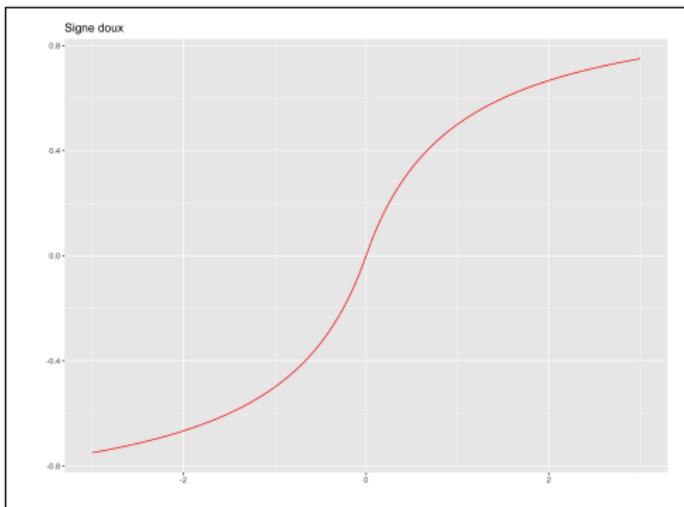
Pratique du perceptron multicouche

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

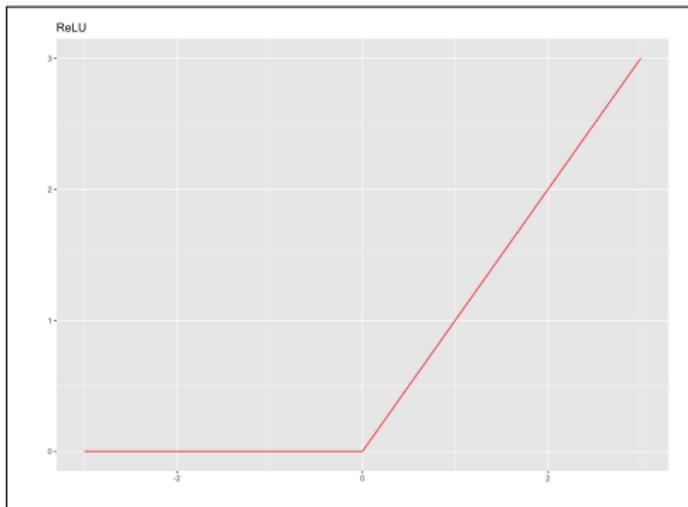
Pratique du perceptron multicouche

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

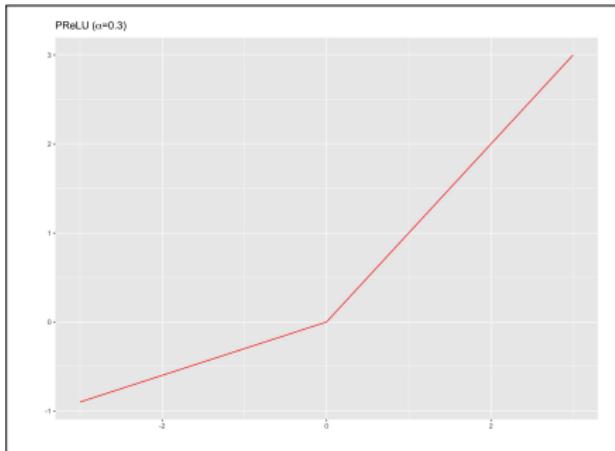
Pratique du perceptron multicouche

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

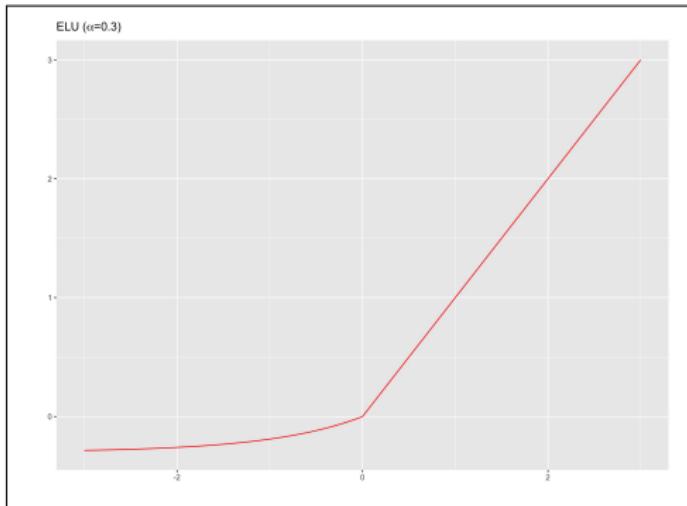
Pratique du perceptron multicouche

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

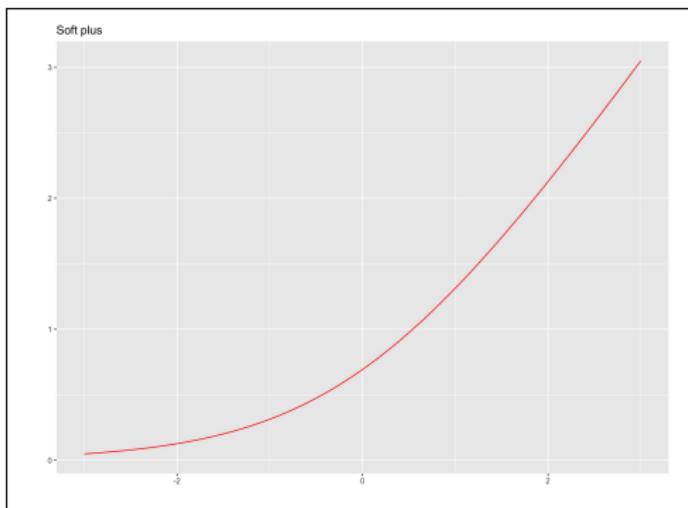
Pratique du perceptron multicouche

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

Pratique du perceptron multicouche

Références

Plan

Introduction

Introduction

Neurone formel

Neurone formel

Fonctions d'activation

Fonctions
d'activation

Perceptron multicouche

Perceptron
multicouche

Rétro-propagation

Rétro-
propagation

Pratique du perceptron multicouche

Pratique du
perceptron
multicouche

Références

Limites du neurone formel

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
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

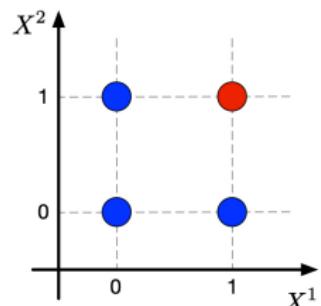
Fonctions d'activation

Perceptron multicouche

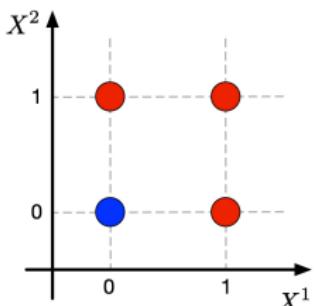
Rétro-propagation

Pratique du perceptron multicouche

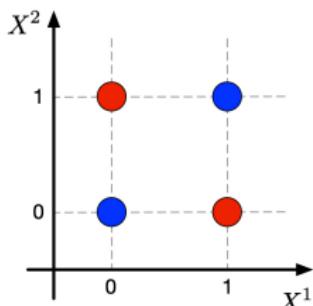
Références



AND



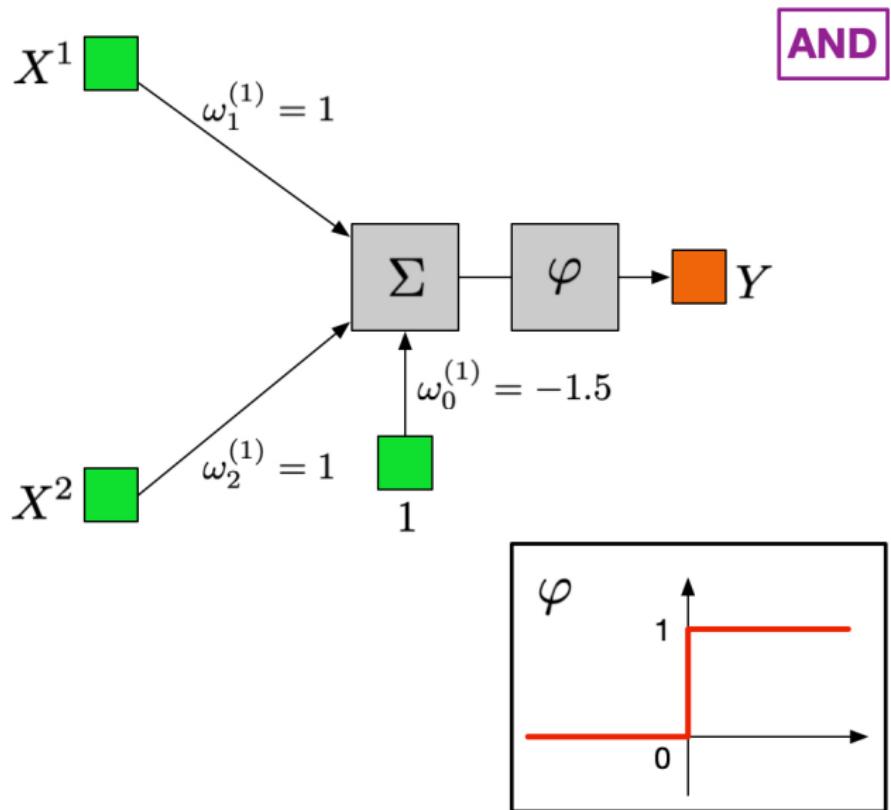
OR



XOR

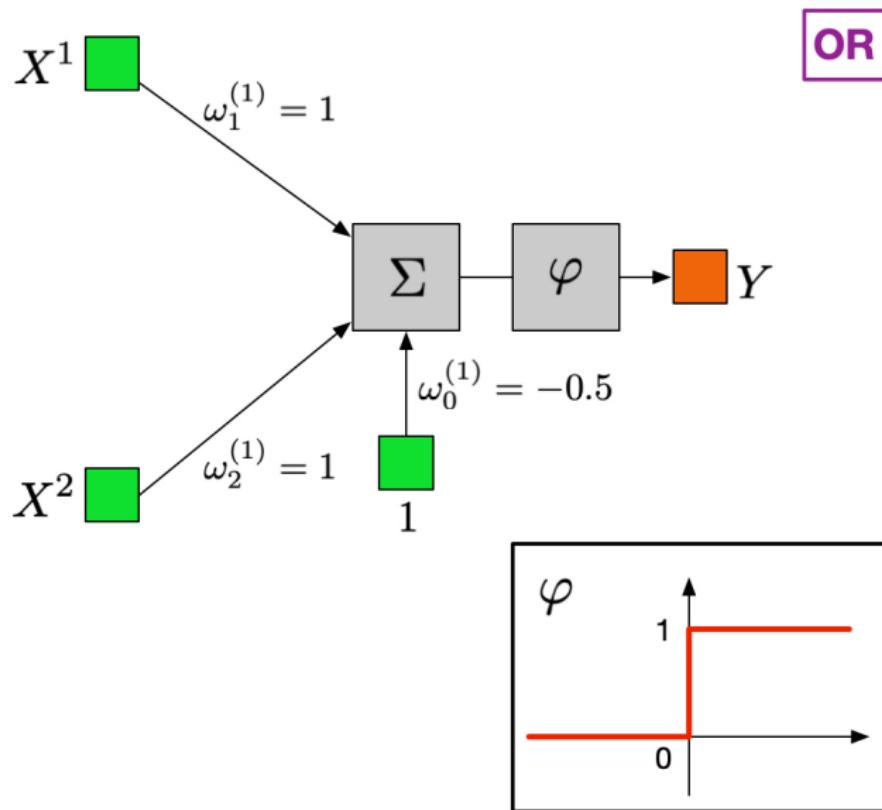
$$\boxed{Y = 0 \quad Y = 1}$$

Fonction AND



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Fonction OR



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Fonction XOR

Introduction

Neurone formel

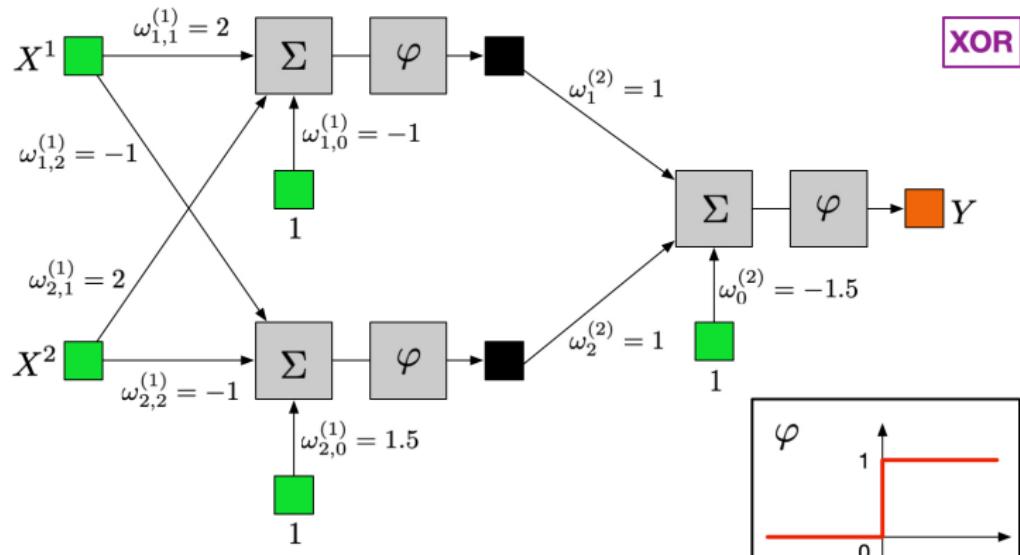
Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références



Architecture

- ▶ Un **perceptron multicouche** (**MLP** : *MultiLayer 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** (**DNN** : *Deep Neural Network*) 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 30 couches cachées...

Introduction

Neurone formel

Fonctions d'activation

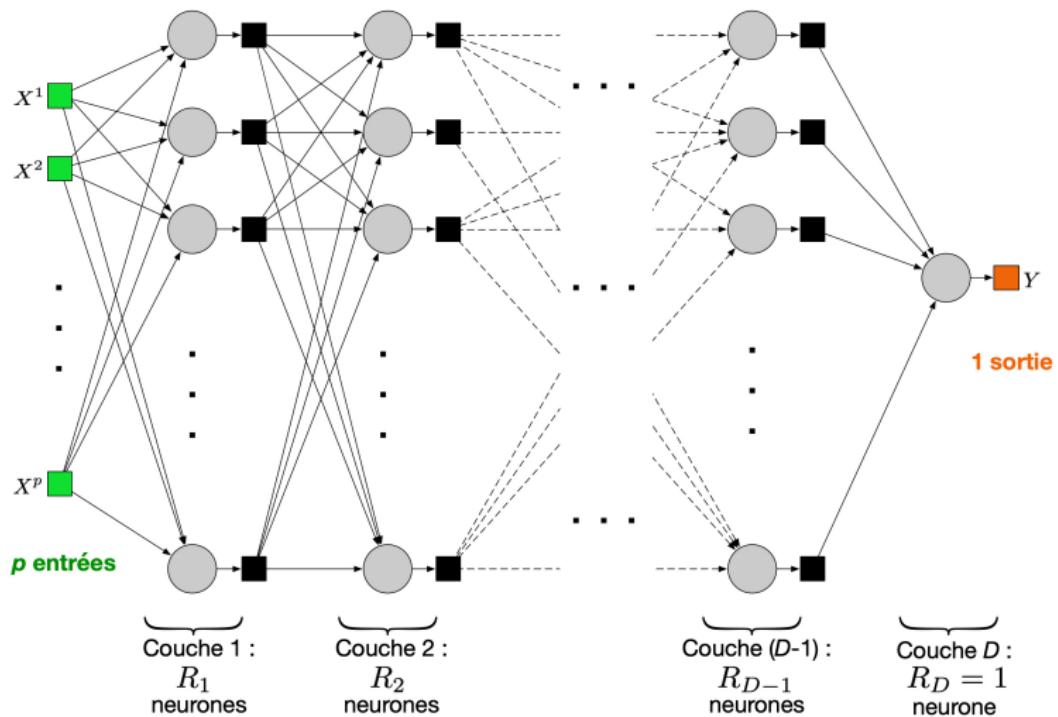
Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Représentation I



Introduction

Neurone formel

Fonctions d'activation

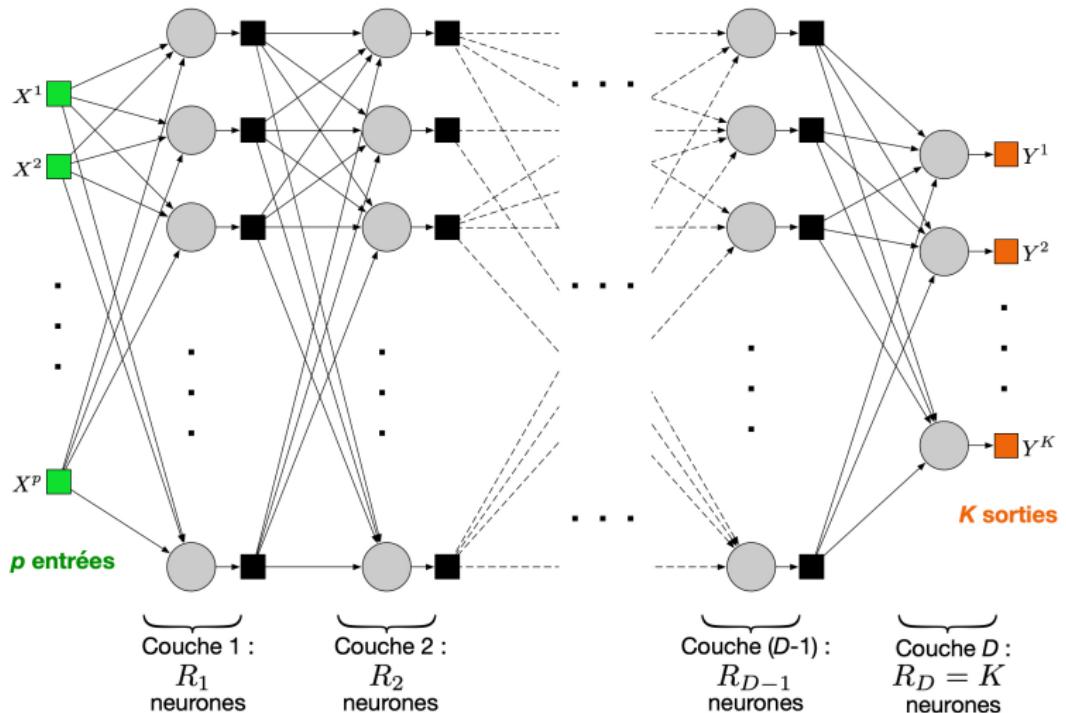
Perceptron multicouche

Réto-propagation

Pratique du perceptron multicouche

Références

Représentation II



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Théorème d'approximation universelle

- ▶ 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* ».

Introduction

Neurone formel

Fonctions
d'activation

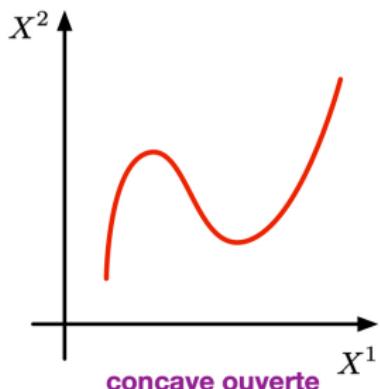
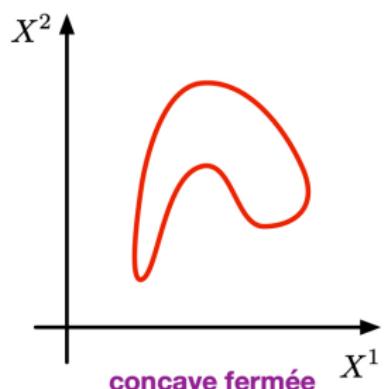
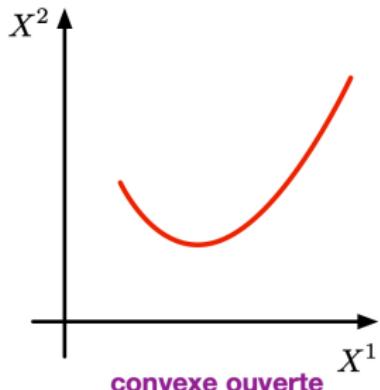
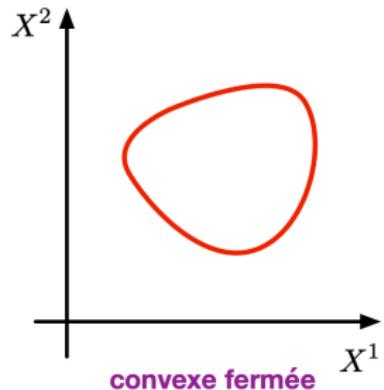
Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Frontières de décision



Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Estimation des poids et des biais

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
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

Introduction

Introduction

Neurone formel

Neurone formel

Fonctions d'activation

Fonctions
d'activation

Perceptron multicouche

Perceptron
multicouche

Rétro-propagation

Rétro-
propagation

Pratique du perceptron multicouche

Pratique du
perceptron
multicouche

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

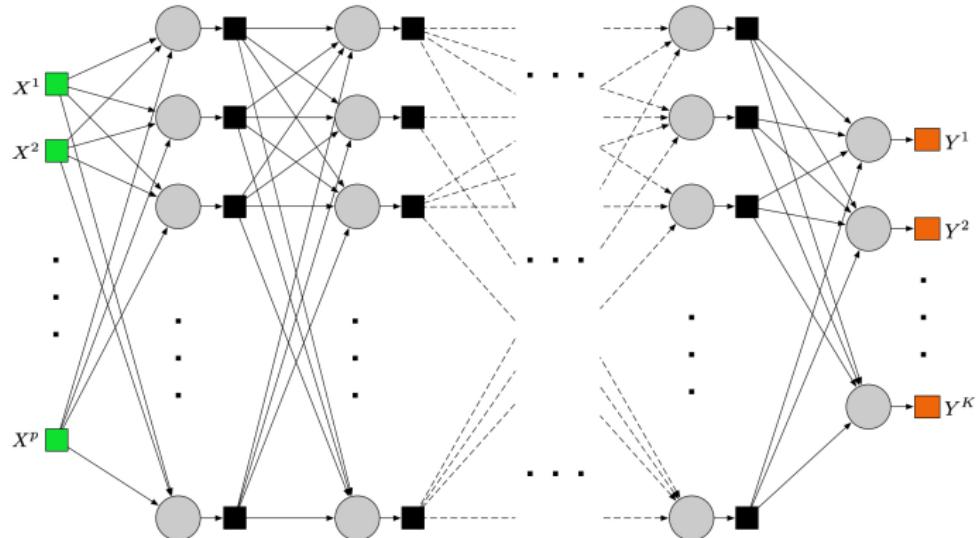
Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Propagation avant...



Introduction

Neurone formel

Fonctions
d'activation

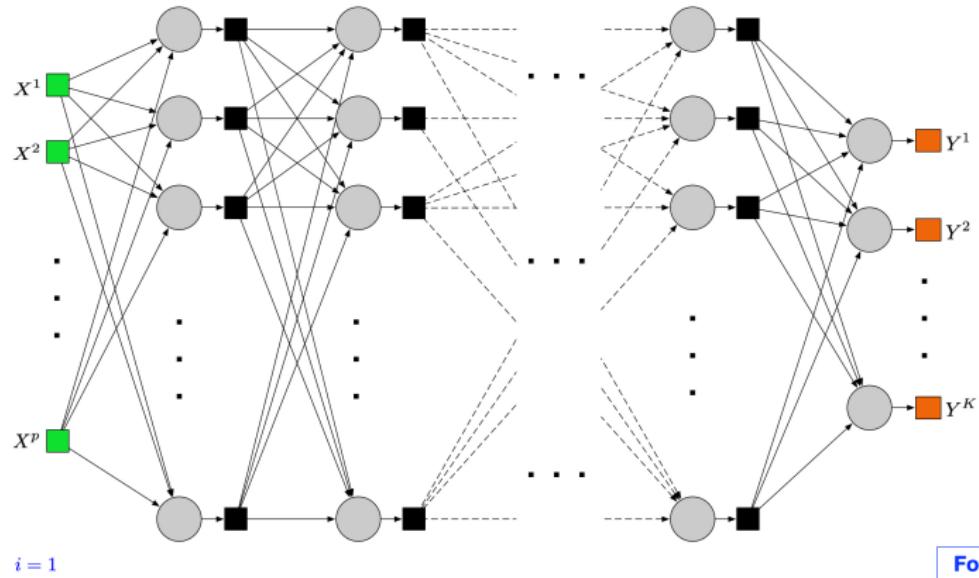
Perceptron
multicouche

Rétrō-
propagation

Pratique du
perceptron
multicouche

Références

Propagation avant...



Introduction

Neurone formel

Fonctions
d'activation

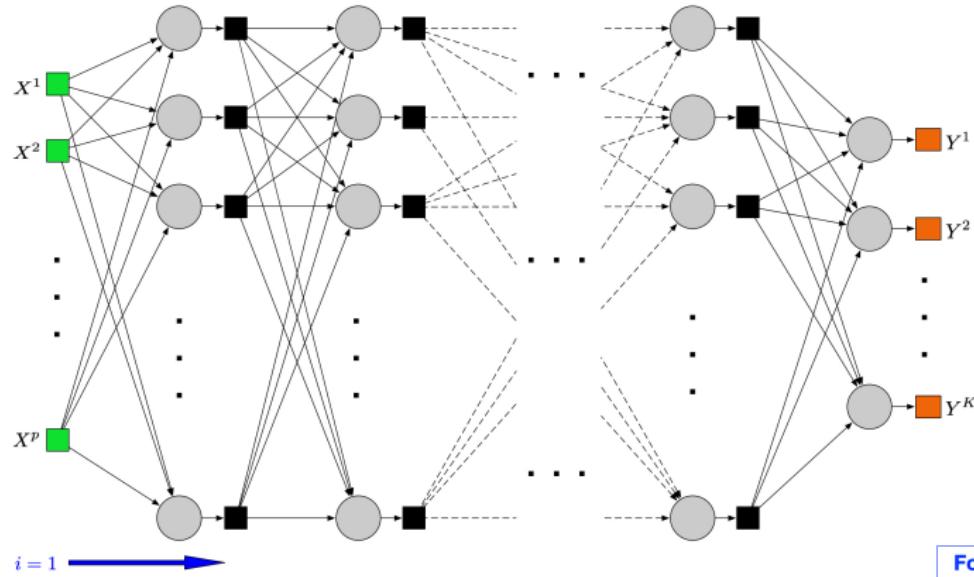
Perceptron
multicouche

Réto-
propagation

Pratique du
perceptron
multicouche

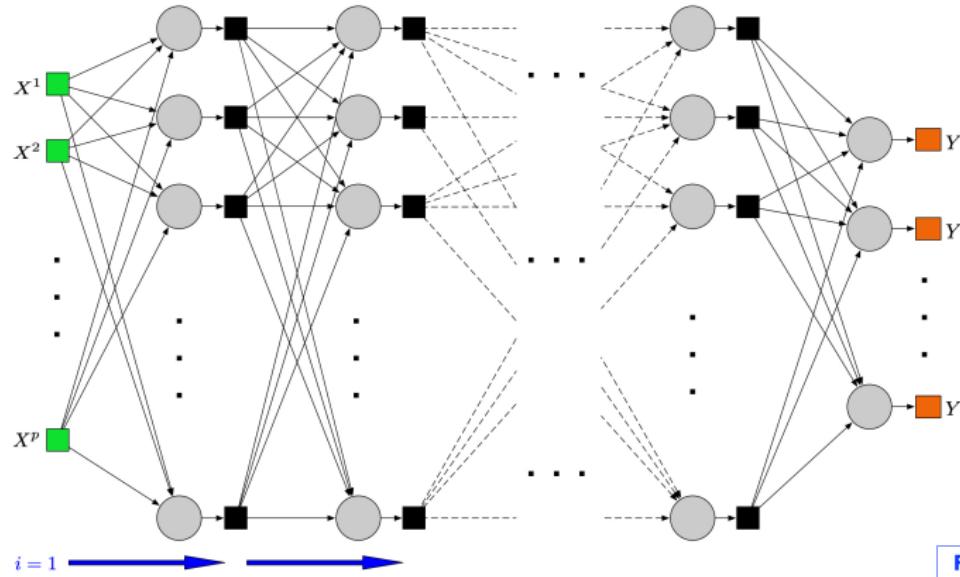
Références

Propagation avant...



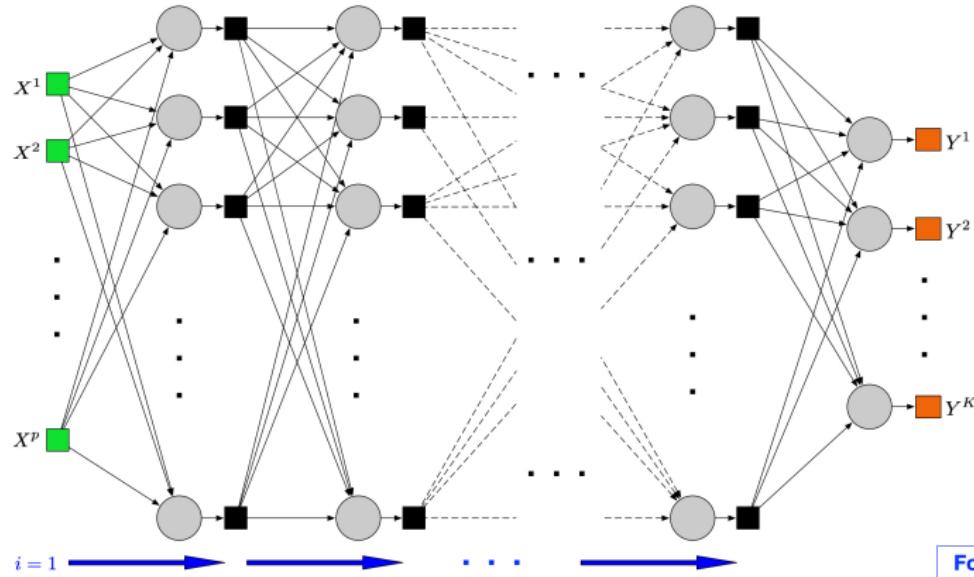
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Propagation avant...



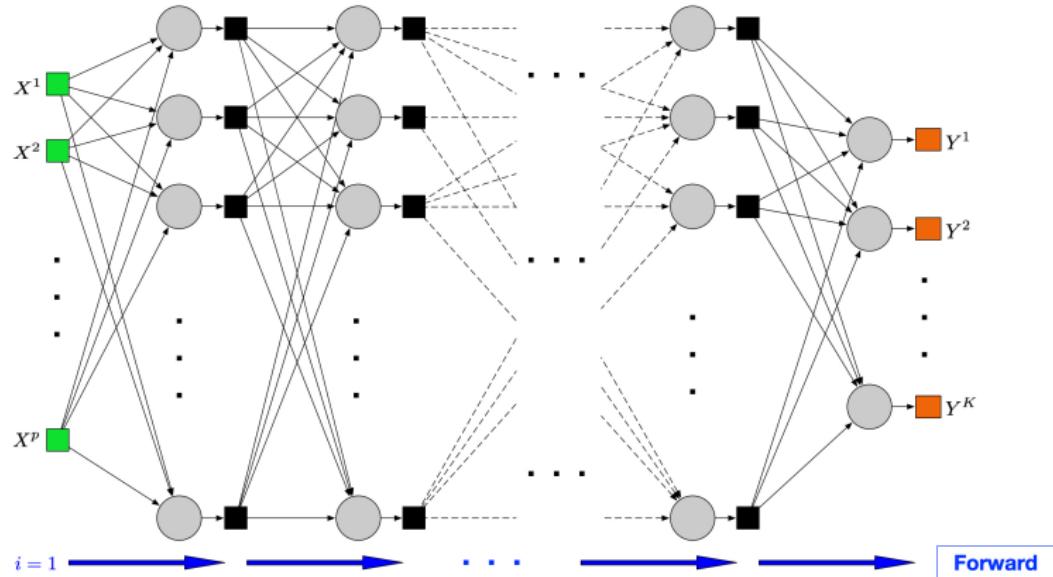
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Propagation avant...



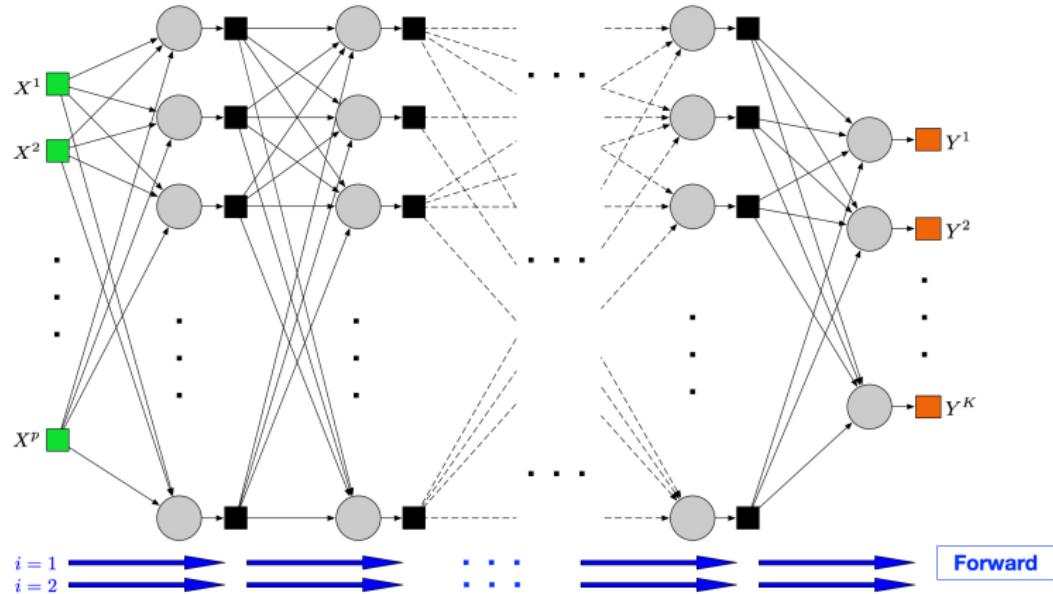
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Propagation avant...



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Propagation avant...



Introduction

Neurone formel

Fonctions d'activation

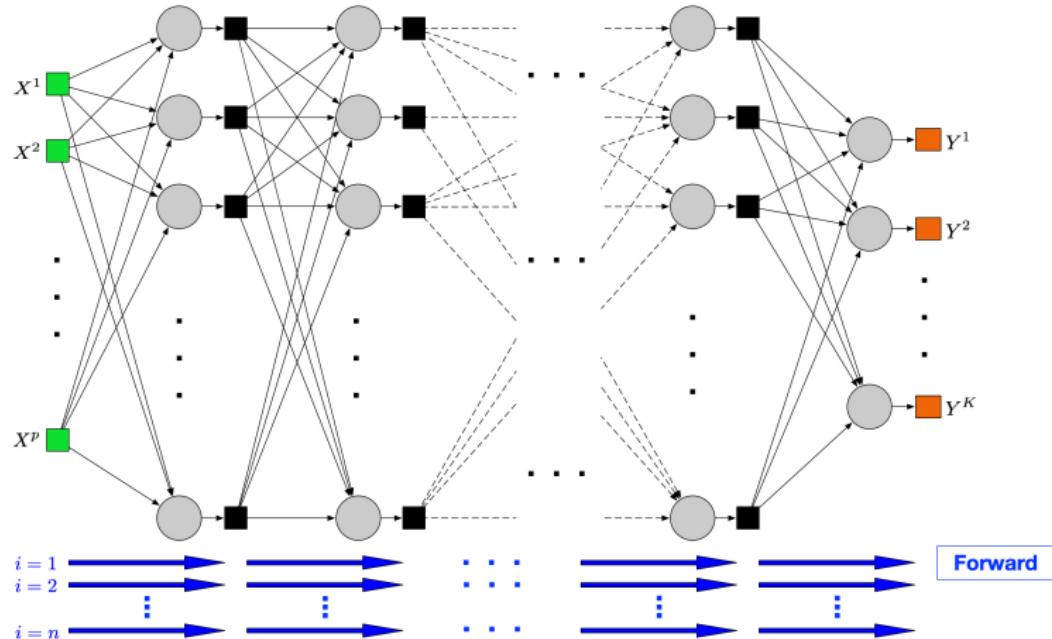
Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

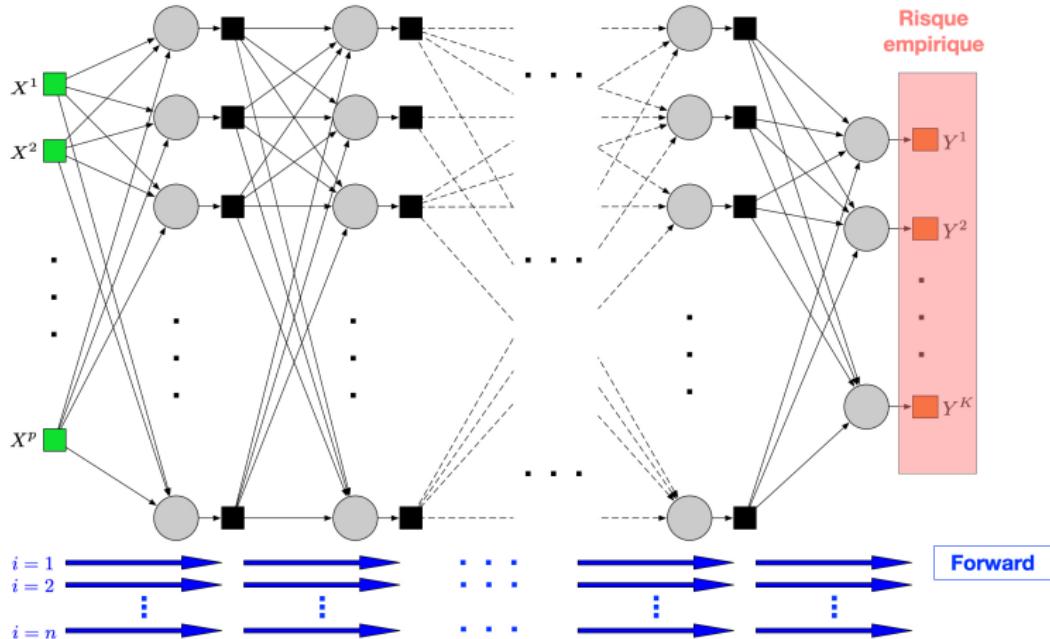
Références

Propagation avant...



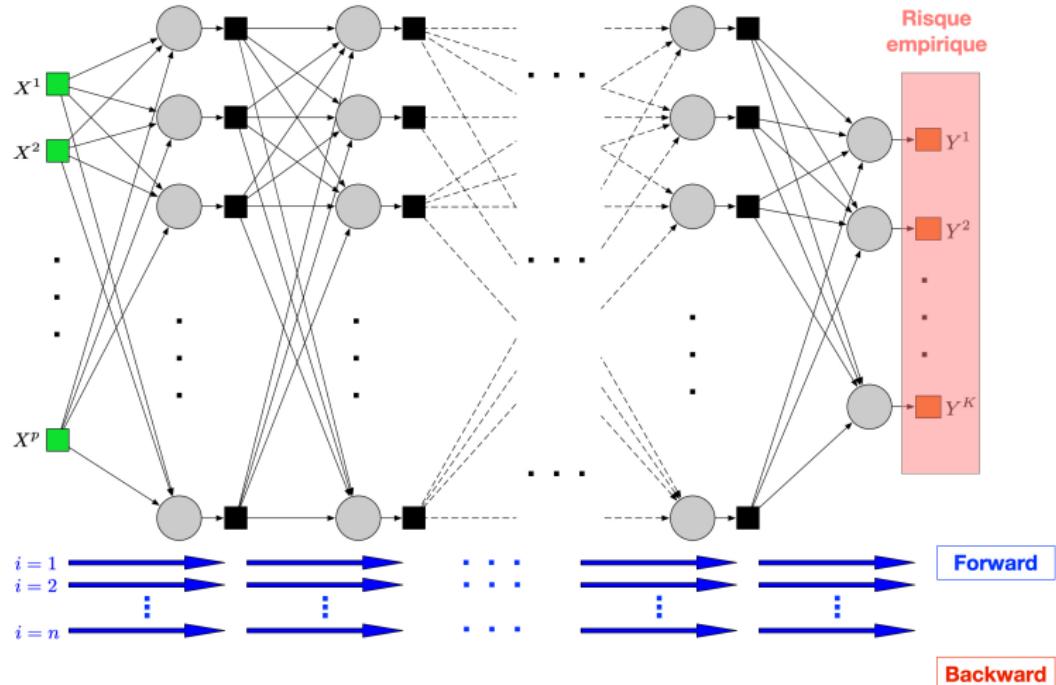
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Calcul d'erreur



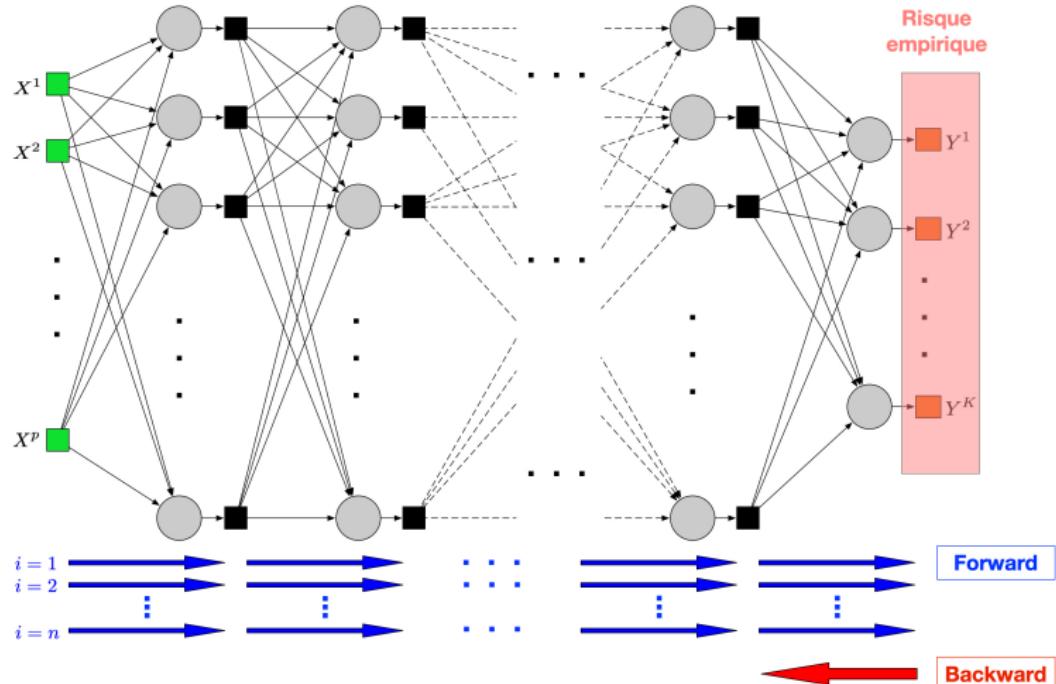
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

... puis retro-propagation



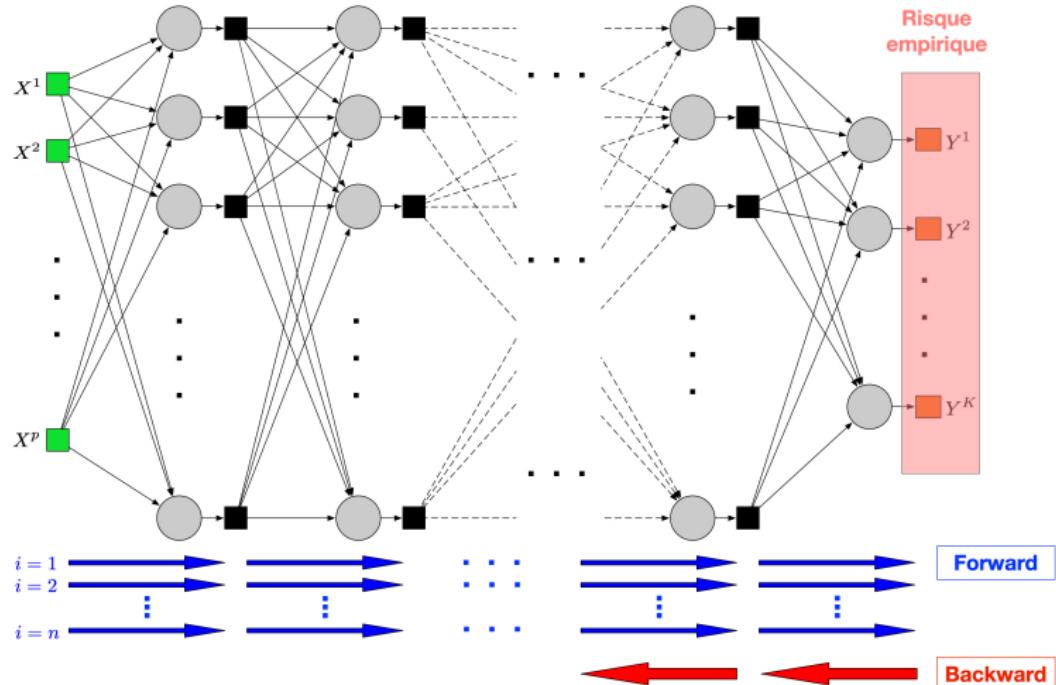
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

... puis retro-propagation



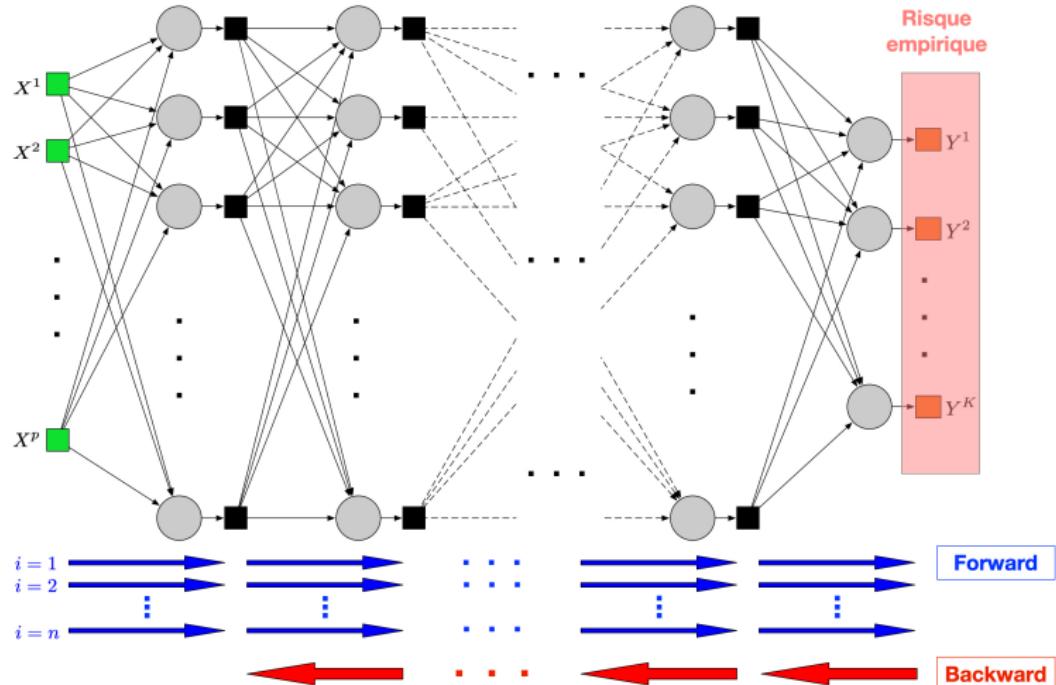
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

... puis retro-propagation



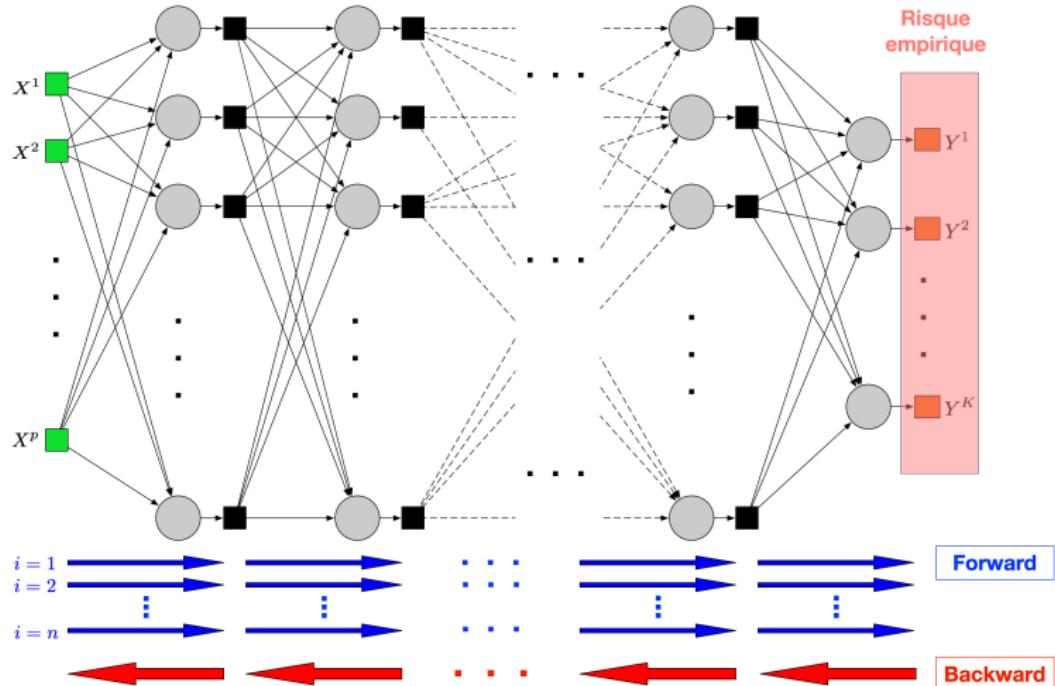
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

... puis retro-propagation



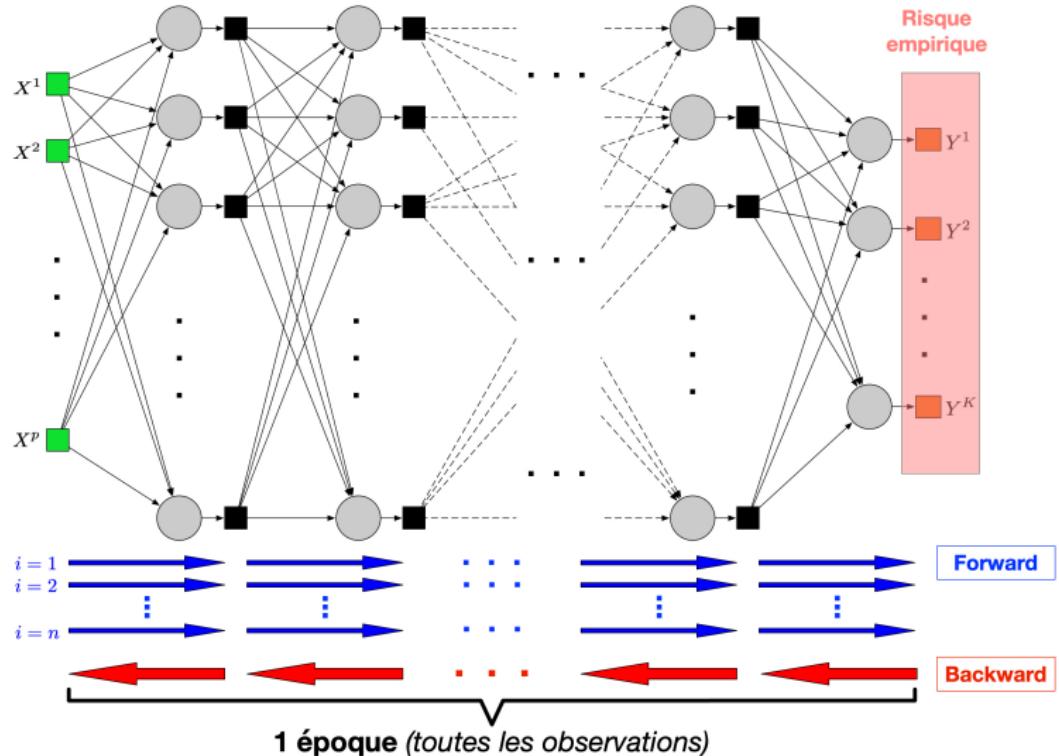
Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

... puis retro-propagation



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Toute une époque



Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Cas de la régression

- ▶ Pour des poids, des biais et une fonction d'activation donnés, à partir d'une entrée $x_i = (x_i^1, \dots, x_i^p)^T$ pour $i \in \{1, \dots, n\}$, on obtient une sortie \hat{y}_i en sortie du réseaux de neurones.
- ▶ Les **fonctions d'activation** usuellement retenues sont :
 - ▶ pour les **couches cachées** : ReLU,
 - ▶ pour la **couche de sortie** : la fonction d'activation **linéaire**.
- ▶ L'erreur 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

Pratique du perceptron multicouche

Références

Cas de la classification supervisée binaire

- ▶ Pour des poids, des biais et une fonction d'activation donnés, à partir d'une entrée $x_i = (x_i^1, \dots, x_i^p)^\top$ pour $i \in \{1, \dots, n\}$, on obtient une sortie \hat{y}_i en sortie du réseaux de neurones.
- ▶ Les **fonctions d'activation** usuellement retenues sont :
 - ▶ pour les **couches cachées** : ReLU,
 - ▶ pour la **couche de sortie** : la fonction d'activation **sigmoïde**.
- ▶ L'erreur 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
Pratique du perceptron multicouche
Références

Cas de la classification supervisée à K classes

- ▶ On se place dans le cas où $K > 2$.
- ▶ Pour des poids, des biais et une fonction d'activation donnés, à partir d'une entrée $x_i = (x_i^1, \dots, x_i^p)^\top$ pour $i \in \{1, \dots, n\}$, on obtient K sorties \hat{y}_i^k en sortie du réseaux de neurones.
- ▶ Les **fonctions d'activation** usuellement retenues sont :
 - ▶ pour les **couches cachées** : ReLU,
 - ▶ pour la **couche de sortie** : la fonction d'activation softmax :

$$\varphi_j(y^1, \dots, y^K) = \frac{e^{y^j}}{\sum_{k=1}^K e^{y^k}} .$$

- ▶ L'erreur usuellement retenue est l'**entropie croisée** :

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

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Exemple

- ▶ On se place dans le cadre de la **régression**, avec la **fonction d'activation sigmoïde** sur les couches intermédiaires (pour des commodités d'écriture) et la **fonction d'activation linéaire** sur la couche de sortie.
- ▶ Pour le neurone $j \in \{1, \dots, R_d\}$ de la couche $d \in \{1, \dots, D\}$, et une entrée x_i , on note $o_j^{(d),i}$ sa sortie :

$$o_j^{(d),i} = \varphi \left(a_j^{(d),i} \right)$$

où :

$$a_j^{(d),i} = \omega_{0,j}^{(d),i} + \sum_{\ell=1}^{R_{d-1}} \omega_{\ell,j}^{(d)} o_{\ell}^{(d-1),i} .$$

Introduction

Neurone formel

Fonctions d'activation

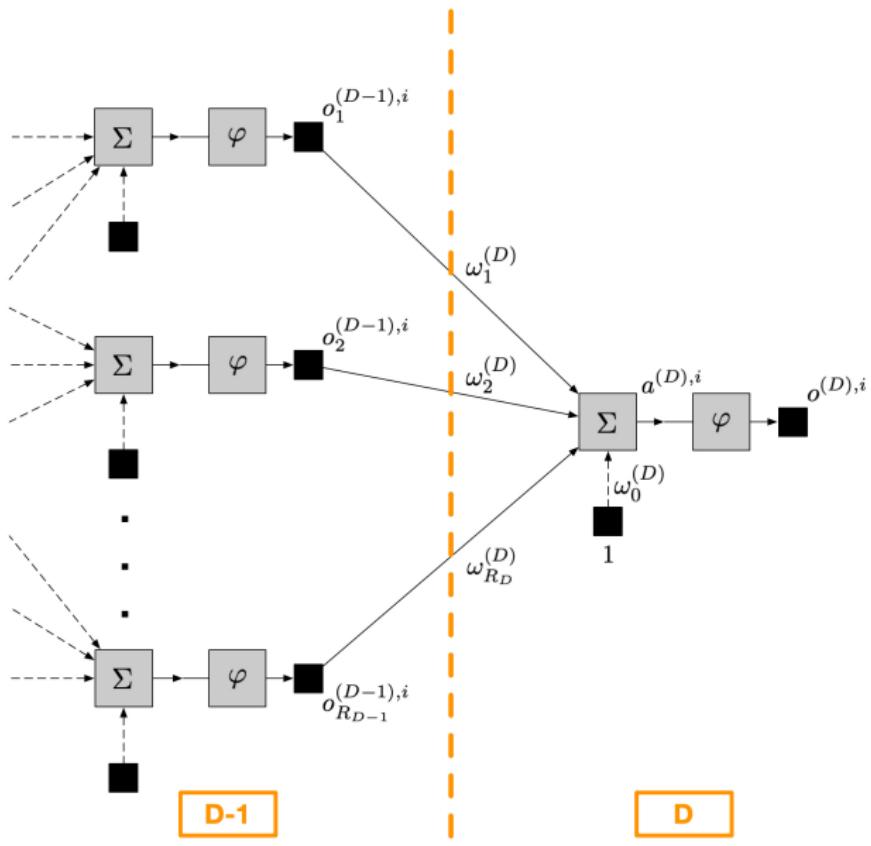
Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Exemple : couche de sortie I



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Réto-
propagation

Pratique du
perceptron
multicouche

Références

Exemple : couche de sortie II

- ▶ On considère le neurone de la couche D .
- ▶ Pour l'observation $i \in \{1, \dots, n\}$, l'erreur en sortie du neurone vaut :

$$e^{(D),i} = y_i - o^{(D),i}$$

et le risque empirique (erreur quadratique) vaut :

$$R_n^{(D),i} = \frac{1}{2} \left(e^{(D),i} \right)^2 = \frac{1}{2} \left(y_i - o^{(D),i} \right)^2 .$$

- ▶ Pour l'échantillon d_n , le risque empirique vaut :

$$R_n^{(D)} = \frac{1}{n} \sum_{i=1}^n R_n^{(D),i} .$$

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Exemple : couche de sortie III

- ▶ La descente du gradient permet de considérer :

$$\begin{cases} \Delta\omega_0^{(D)} = -\eta \frac{\partial R_n^{(D)}}{\partial \omega_0^{(D)}} = -\frac{\eta}{n} \sum_{i=1}^n \frac{\partial R_n^{(D),i}}{\partial \omega_0^{(D)}} \\ \Delta\omega_\ell^{(D)} = -\eta \frac{\partial R_n^{(D)}}{\partial \omega_\ell^{(D)}} = -\frac{\eta}{n} \sum_{i=1}^n \frac{\partial R_n^{(D),i}}{\partial \omega_\ell^{(D)}} \end{cases}$$

où η est le **taux d'apprentissage**.

- ▶ La modification des poids de la couche de sortie dépend de l'erreur commise en sortie.
- ▶ Le théorème de **dérivation en chaîne** permet d'écrire :

$$\begin{cases} \frac{\partial R_n^{(D),i}}{\partial \omega_0^{(D)}} = \frac{\partial R_n^{(D),i}}{\partial o^{(D),i}} \frac{\partial o^{(D),i}}{\partial a^{(D),i}} \frac{\partial a^{(D),i}}{\partial \omega_0^{(D)}} \\ \frac{\partial R_n^{(D),i}}{\partial \omega_\ell^{(D)}} = \frac{\partial R_n^{(D),i}}{\partial o^{(D),i}} \frac{\partial o^{(D),i}}{\partial a^{(D),i}} \frac{\partial a^{(D),i}}{\partial \omega_\ell^{(D)}} \end{cases}.$$

Correction des poids et biais : couche de sortie IV

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

► On obtient :

$$\frac{\partial R_n^{(D),i}}{\partial o^{(D),i}} = - \left(y_i - o^{(D),i} \right) = -e^{(D),i},$$

$$\frac{\partial o^{(D),i}}{\partial a^{(D),i}} = \frac{\partial \varphi \left(a^{(D),i} \right)}{\partial a^{(D),i}} = 1,$$

$$\frac{\partial a^{(D),i}}{\partial \omega_0^{(D)}} = \frac{\partial}{\partial \omega_0^{(D)}} \left(\omega_0^{(D)} + \sum_{m=1}^{R_{D-1}} \omega_m^{(D)} o_m^{(D-1),i} \right) = 1,$$

$$\frac{\partial a^{(D),i}}{\partial \omega_\ell^{(D)}} = \frac{\partial}{\partial \omega_\ell^{(D)}} \left(\omega_0^{(D)} + \sum_{m=1}^{R_{D-1}} \omega_m^{(D)} o_m^{(D-1),i} \right) = o_\ell^{(D-1),i}.$$

Correction des poids et biais : couche de sortie V

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

► D'où :

$$\begin{cases} \frac{\partial R_n^{(D),i}}{\partial \omega_0^{(D)}} = -e^{(D),i} \\ \frac{\partial R_n^{(D),i}}{\partial \omega_\ell^{(D)}} = -e^{(D),i} o_\ell^{(D-1),i} \end{cases}$$

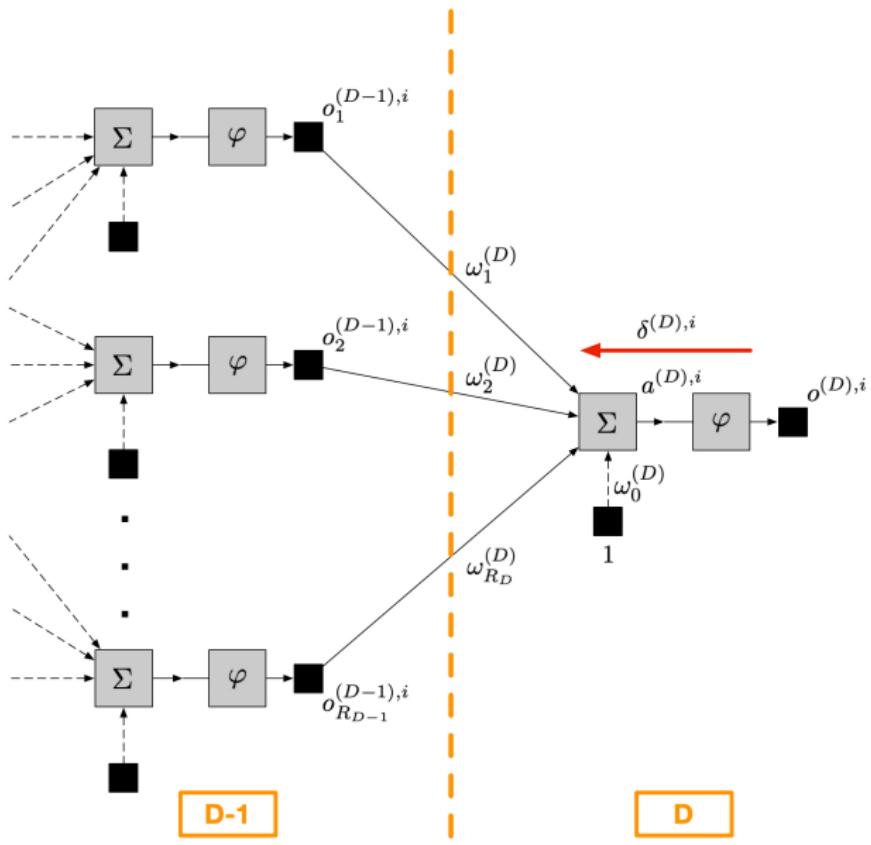
► La correction des poids et biais apportée sur la (dernière) couche D est :

$$\begin{cases} \Delta \omega_0^{(D)} = \frac{\eta}{n} \sum_{i=1}^n \delta^{(D),i} \\ \Delta \omega_\ell^{(D)} = \frac{\eta}{n} \sum_{i=1}^n \delta^{(D),i} o_\ell^{(D-1),i} \end{cases}$$

où :

$$\delta^{(D),i} = e^{(D),i}$$

Exemple : couche de sortie VI



Introduction

Neurone formel

Fonctions
d'activation

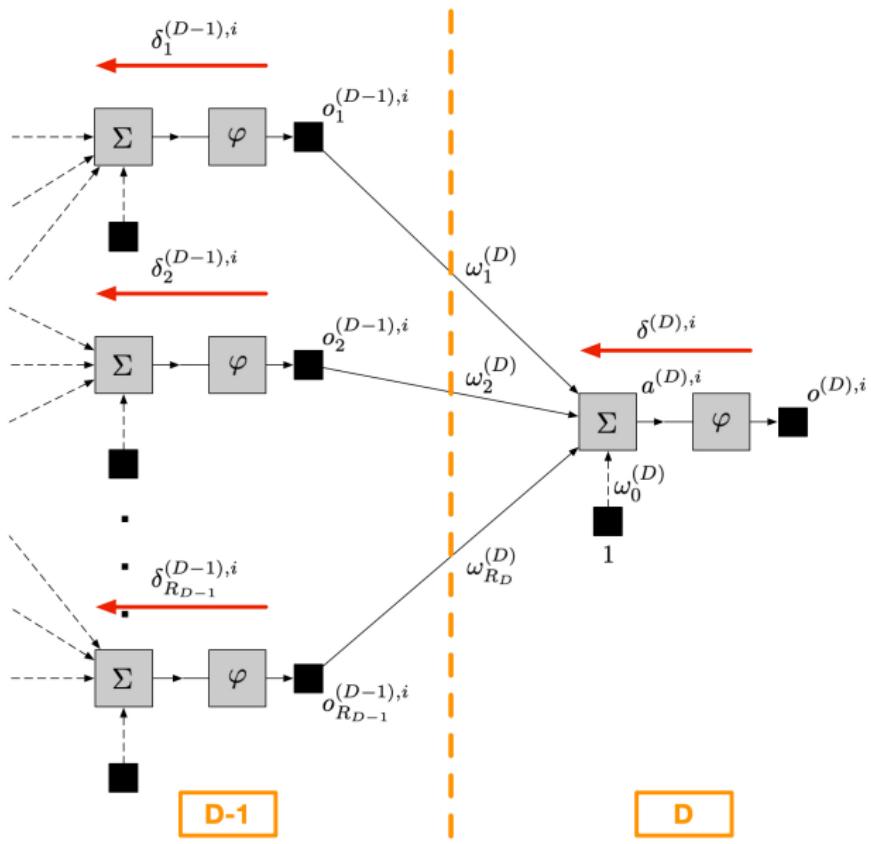
Perceptron
multicouche

Réto-
propagation

Pratique du
perceptron
multicouche

Références

Exemple : couche de sortie VII



Introduction

Neurone formel

Fonctions d'activation

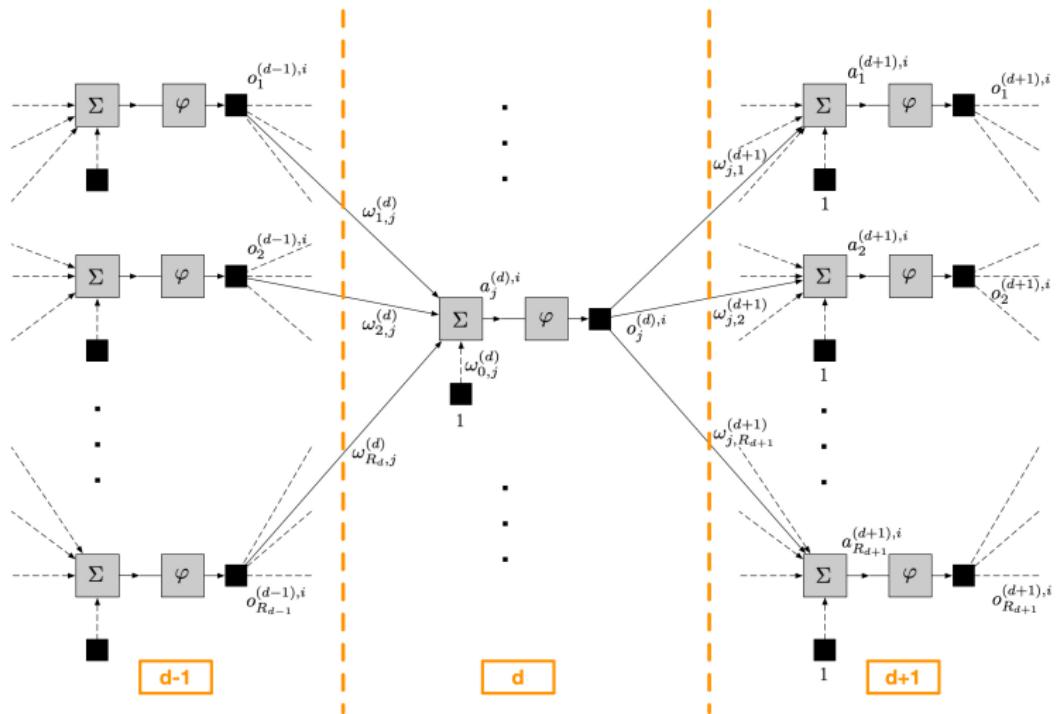
Perceptron multicouche

Réto-propagation

Pratique du perceptron multicouche

Références

Exemple : couche cachée I



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Réto-
propagation

Pratique du
perceptron
multicouche

Références

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Exemple : couche cachée II

- ▶ L'erreur sur un neurone $j \in \{1, \dots, d\}$ de la couche d dépend de l'erreur des R_{d+1} neurones de la couche $d + 1$.
- ▶ Pour l'observation $i \in \{1, \dots, n\}$, l'erreur en sortie du neurone m de la couche $d + 1$ vaut :

$$e_m^{(d+1),i} = t_m^{(d+1),i} - o_m^{(d+1),i}$$

où $t_m^{(d+1),i}$ est la valeur « objectif » (*target*) à la sortie du neurone m , et le risque empirique (erreur quadratique) vaut :

$$R_n^{(d+1),i} = \sum_{m=1}^{R_{d+1}} \frac{1}{2} \left(e_m^{(d+1),i} \right)^2 = \sum_{m=1}^{R_{d+1}} \frac{1}{2} \left(t_m^{(d+1),i} - o_m^{(d+1),i} \right)^2.$$

- ▶ Pour l'échantillon d_n , le risque empirique en sortie de la couche $d + 1$ vaut :

$$R_n^{(d+1)} = \frac{1}{n} \sum_{i=1}^n R_n^{(d+1),i}.$$

Exemple : couche cachée III

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

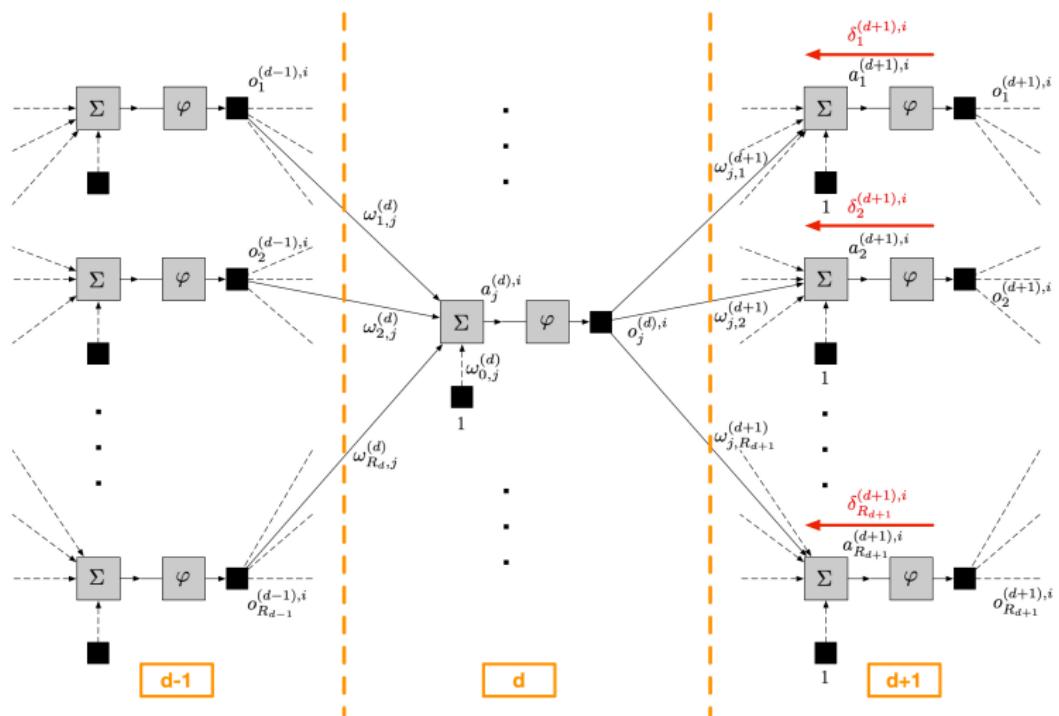
- ▶ La correction des poids et biais apportée à la couche d est :

$$\begin{cases} \Delta\omega_{0,j}^{(d)} = \frac{\eta}{n} \sum_{i=1}^n \delta_j^{(d),i} \\ \Delta\omega_{\ell,j}^{(d)} = \frac{\eta}{n} \sum_{i=1}^n \delta_j^{(d),i} o_{\ell,j}^{(d-1),i} \end{cases}$$

où :

$$\delta_j^{(d),i} = \left(\sum_{m=1}^{R_{d+1}} \delta_m^{(d+1),i} \omega_{j,m}^{(d+1)} \right) o_j^{(d),i} \left[1 - o_j^{(d),i} \right].$$

Exemple : couche cachée IV



Introduction

Neurone formel

Fonctions d'activation

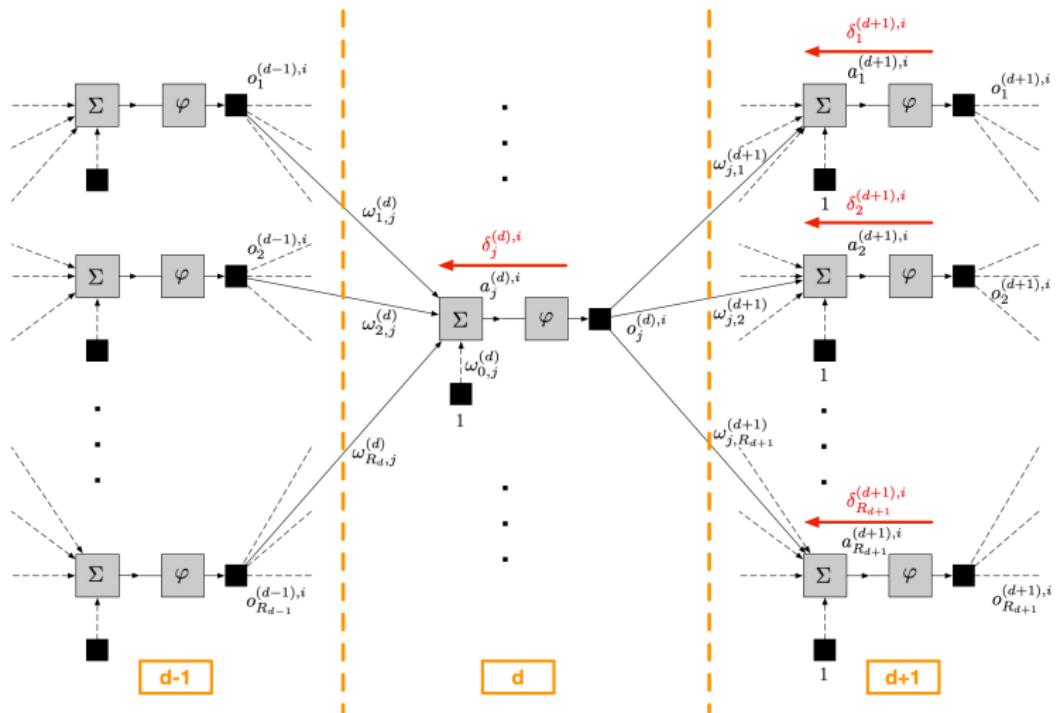
Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Exemple : couche cachée V



Introduction

Neurone formel

Fonctions d'activation

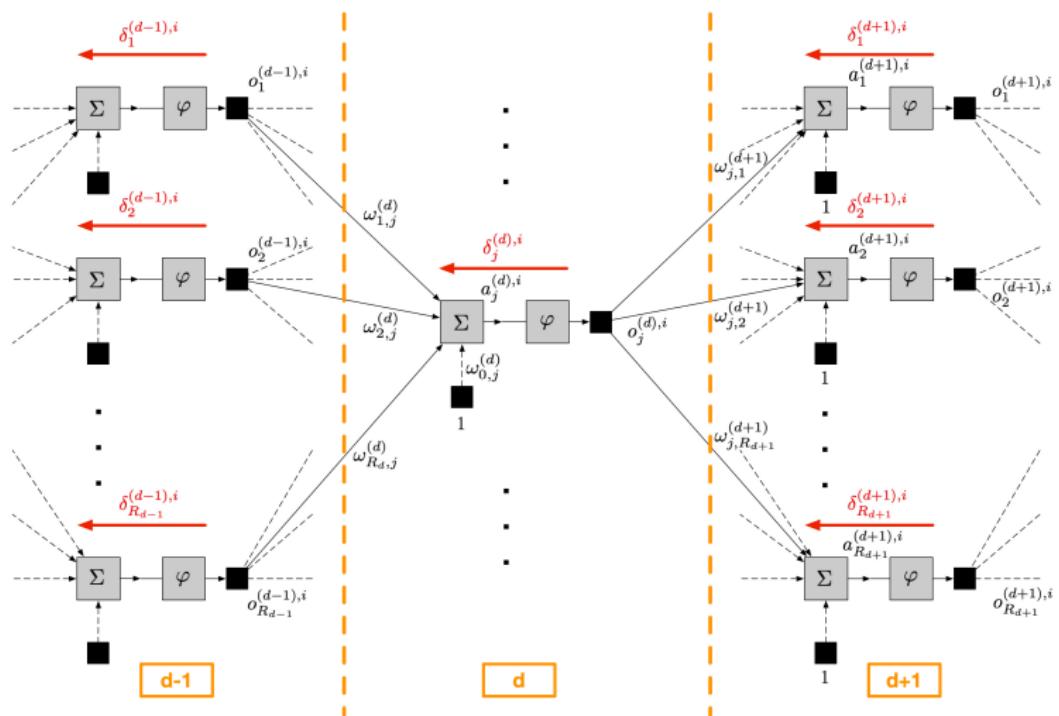
Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Exemple : couche cachée VI



Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Exemple : couche cachée VII

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

- ▶ La descente du gradient permet de considérer pour $\ell \in \{1, \dots, R_{d-1}\}$:

$$\begin{cases} \Delta\omega_{0,j}^{(d)} = -\eta \frac{\partial R_n^{(d+1)}}{\partial \omega_{0,j}^{(d)}} = -\frac{\eta}{n} \sum_{i=1}^n \frac{\partial R_n^{(d+1),i}}{\partial \omega_{0,j}^{(d)}} \\ \Delta\omega_{\ell,j}^{(d)} = -\eta \frac{\partial R_n^{(d+1)}}{\partial \omega_{\ell,j}^{(d)}} = -\frac{\eta}{n} \sum_{i=1}^n \frac{\partial R_n^{(d+1),i}}{\partial \omega_{\ell,j}^{(d)}} \end{cases}.$$

- ▶ Le théorème de dérivation en chaîne permet d'écrire :

$$\begin{cases} \frac{\partial R_n^{(d+1),i}}{\partial \omega_{0,j}^{(d)}} = \frac{\partial R_n^{(d+1),i}}{\partial o_j^{(d),i}} \frac{\partial o_j^{(d),i}}{\partial a^{(D),i}} \frac{\partial a^{(D),i}}{\partial \omega_{0,j}^{(d)}} \\ \frac{\partial R_n^{(d+1),i}}{\partial \omega_{\ell,j}^{(d)}} = \frac{\partial R_n^{(d+1),i}}{\partial o_j^{(d),i}} \frac{\partial o_j^{(d),i}}{\partial a^{(D),i}} \frac{\partial a^{(D),i}}{\partial \omega_{\ell,j}^{(d)}} \end{cases}.$$

Exemple : couche cachée VIII

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

► On obtient :

$$\frac{\partial o_j^{(d),i}}{\partial a_j^{(d),i}} = \frac{\partial \varphi(a_j^{(d),i})}{\partial a_j^{(d),i}} = \varphi(a_j^{(d),i}) [1 - \varphi(a_j^{(d),i})] = o_j^{(d),i} (1 - o_j^{(d),i}) ,$$

$$\frac{\partial a_j^{(d),i}}{\partial \omega_{0,j}^{(d)}} = \frac{\partial}{\partial \omega_{0,j}^{(d)}} \left(\omega_{0,j}^{(d)} + \sum_{\ell=1}^{R_{d-1}} \omega_{\ell,j}^{(d)} o_{\ell,j}^{(d-1),i} \right) = 1 ,$$

$$\frac{\partial a_j^{(d),i}}{\partial \omega_{\ell,j}^{(d)}} = \frac{\partial}{\partial \omega_{\ell,j}^{(d)}} \left(\omega_{0,j}^{(d)} + \sum_{m=1}^{R_{d-1}} \omega_{m,j}^{(d)} o_{m,j}^{(d-1),i} \right) = o_{\ell,j}^{(d-1),i} .$$

Exemple : couche cachée IX

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

$$\begin{aligned}\frac{\partial R_n^{(d+1),i}}{\partial o_j^{(d),i}} &= \frac{\partial}{\partial o_j^{(d),i}} \left(\sum_{m=1}^{R_{d+1}} \frac{1}{2} \left(e_m^{(d+1),i} \right)^2 \right) \\ &= \sum_{m=1}^{R_{d+1}} \frac{\partial}{\partial o_j^{(d),i}} \left(\frac{1}{2} \left(e_m^{(d+1),i} \right)^2 \right) \\ &= \sum_{m=1}^{R_{d+1}} \frac{\partial}{\partial e_m^{(d+1),i}} \left(\frac{1}{2} \left(e_m^{(d+1),i} \right)^2 \right) \frac{\partial e_m^{(d+1),i}}{\partial o_j^{(d),i}} \\ &= \sum_{m=1}^{R_{d+1}} e_m^{(d+1),i} \frac{\partial e_m^{(d+1),i}}{\partial o_j^{(d),i}} \\ &= \sum_{m=1}^{R_{d+1}} e_m^{(d+1),i} \frac{\partial e_m^{(d+1),i}}{\partial a_m^{(d+1),i}} \frac{\partial a_m^{(d+1),i}}{\partial o_j^{(d),i}} .\end{aligned}$$

Exemple : couche cachée X

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

$$\begin{aligned}\frac{\partial e_m^{(d+1),i}}{\partial a_m^{(d+1),i}} &= \frac{\partial}{\partial a_m^{(d+1),i}} \left(t_m^{(d+1),i} - o_m^{(d+1),i} \right) \\ &= -\frac{\partial}{\partial a_m^{(d+1),i}} \left(o_m^{(d+1),i} \right) \\ &= -\frac{\partial}{\partial a_m^{(d+1),i}} \left(\varphi \left(a_m^{(d+1),i} \right) \right) \\ &= -\varphi \left(a_m^{(d+1),i} \right) \left[1 - \varphi \left(a_m^{(d+1),i} \right) \right] \\ &= -o_m^{(d+1),i} \left(1 - o_m^{(d+1),i} \right) ,\end{aligned}$$

$$\begin{aligned}\frac{\partial a_m^{(d+1),i}}{\partial o_j^{(d),i}} &= \frac{\partial}{\partial o_j^{(d),i}} \left(\omega_{0,m}^{(d+1),i} + \sum_{\ell=1}^{R_d} \omega_{\ell,m}^{(d+1)} o_{\ell}^{(d),i} \right) \\ &= \omega_{j,m}^{(d+1)} .\end{aligned}$$

Exemple : couche cachée XI

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

► Donc :

$$\begin{aligned}\frac{\partial R_n^{(d+1),i}}{\partial o_j^{(d),i}} &= - \sum_{m=1}^{R_{d+1}} e_m^{(d+1),i} o_m^{(d+1),i} \left(1 - o_m^{(d+1),i}\right) \omega_{j,m}^{(d+1)} \\ &= - \sum_{m=1}^{R_{d+1}} \delta_m^{(d+1),i} \omega_{j,m}^{(d+1)}\end{aligned}$$

où :

$$\delta_m^{(d+1),i} = e_m^{(d+1),i} o_m^{(d+1),i} \left(1 - o_m^{(d+1),i}\right) .$$

Exemple : couche cachée XII

► D'où :

$$\begin{cases} \frac{\partial R_n^{(d+1),i}}{\partial \omega_{0,j}^{(d)}} = - \left(\sum_{m=1}^{R_{d+1}} \delta_m^{(d+1),i} \omega_{j,m}^{(d+1)} \right) o_j^{(d),i} \left(1 - o_j^{(d),i} \right) \\ \frac{\partial R_n^{(d+1),i}}{\partial \omega_{\ell,j}^{(d)}} = - \left(\sum_{m=1}^{R_{d+1}} \delta_m^{(d+1),i} \omega_{j,m}^{(d+1)} \right) o_j^{(d),i} \left(1 - o_j^{(d),i} \right) o_{\ell,j}^{(d-1),i} \end{cases}$$

► Au final on obtient :

$$\begin{cases} \Delta \omega_{0,j}^{(d)} = \frac{\eta}{n} \sum_{i=1}^n \delta_j^{(d),i} \\ \Delta \omega_{\ell,j}^{(d)} = \frac{\eta}{n} \sum_{i=1}^n \delta_j^{(d),i} o_{\ell,j}^{(d-1),i} \end{cases}$$

où :

$$\delta_j^{(d),i} = \left(\sum_{m=1}^{R_{d+1}} \delta_m^{(d+1),i} \omega_{j,m}^{(d+1)} \right) o_j^{(d),i} \left[1 - o_j^{(d),i} \right] .$$

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

Plan

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Apprentissage en ligne et par lot I

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
Références

- ▶ L'**apprentissage par lot** (*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 en ligne et par lot II

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

- ▶ L'apprentissage par lot 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.

Apprentissage en ligne et par lot III

Introduction
Neurone formel
Fonctions d'activation
Perceptron multicouche
Rétro-propagation
Pratique du perceptron multicouche
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,
 - ▶ 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

Pratique du
perceptron
multicouche

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 :
 - ▶ $\mathcal{U}(-[\frac{1}{2}, \frac{1}{2}])$,
 - ▶ $\mathcal{N}(0, 1)$,
 - ▶ $\mathcal{N}\left(0, \frac{1}{R_{d-1}}\right)$ pour les neurones de la couche $d \in \{2, \dots, D\}$.

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

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 :

$$X^{j*} = \frac{X^j - \min(X^j)}{\max(X^j) - \min(X^j)} .$$

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

Normalisation des données II

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

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.

Quelques problèmes liés à l'optimisation

- ▶ 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

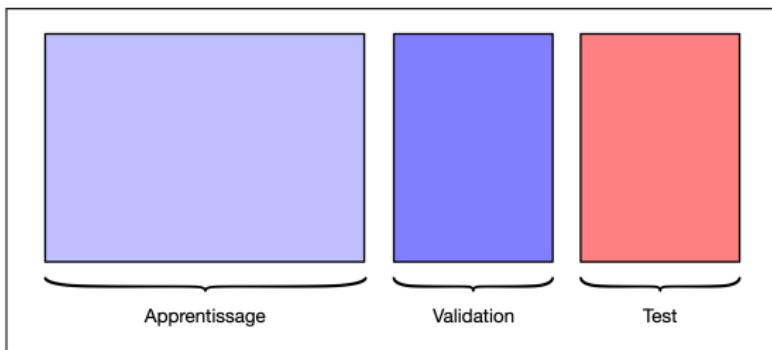
Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Eviter le sur-apprentissage

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

Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

Le coin Python

- ▶ Google



- ▶ Meta



Introduction

Neurone formel

Fonctions
d'activation

Perceptron
multicouche

Rétro-
propagation

Pratique du
perceptron
multicouche

Références

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.
- Cardon, D., J.-P. Cointet et A. Mazieres. 2018, «La revanche des neurones : l'invention des machines inductives et la controverse de l'intelligence artificielle», *Réseaux*, vol. 5, n° 211, p. 173–220.
- Chollet, F. 2020, *L'apprentissage profond avec Python*, Machinelearning.fr.

Introduction

Neurone formel

Fonctions d'activation

Perceptron multicouche

Rétro-propagation

Pratique du perceptron multicouche

Références

Références II

- Cybenko, G. 1989, «Approximation by superpositions of a sigmoidal function», *Mathematics of Control, Signals and Systems*, vol. 2, n° 4, p. 303–314.
- Gagné, C. 2022, «Introduction à l'apprentissage automatique», URL
<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.
- 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

Pratique du perceptron multicouche

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

Pratique du perceptron multicouche

Références