

Cours 4 - Introduction aux réseaux de neurones

Neila Mezghani

17 janvier 2022

Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

3 Le perceptron multi-couche (PMC)

- Structure du PMC
- Entrainement du PMC
- Algorithme de rétropropagation du gradient
- Fonction d'activations

Introduction

Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

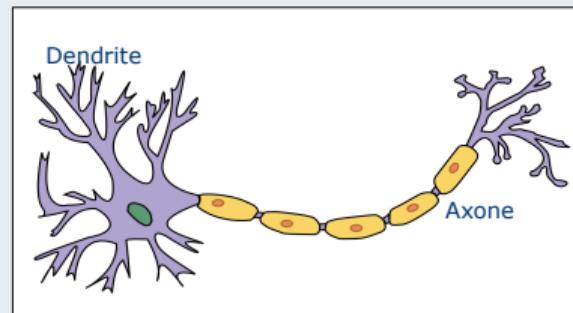
- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

3 Le perceptron multi-couche (PMC)

- Structure du PMC
- Entrainement du PMC
- Algorithme de rétropropagation du gradient
- Fonction d'activations

Le neurone biologique (1/2)

- Dans le cerveau, les neurones sont reliés entre eux par l'intermédiaire d'axones et de dendrites
- Les dendrites représentent les entrées du neurone et son axone sa sortie.
- Le corps cellulaire du neurone est le centre de contrôle.
- Un neurone émet un signal en fonction des signaux qui lui proviennent des autres neurones.



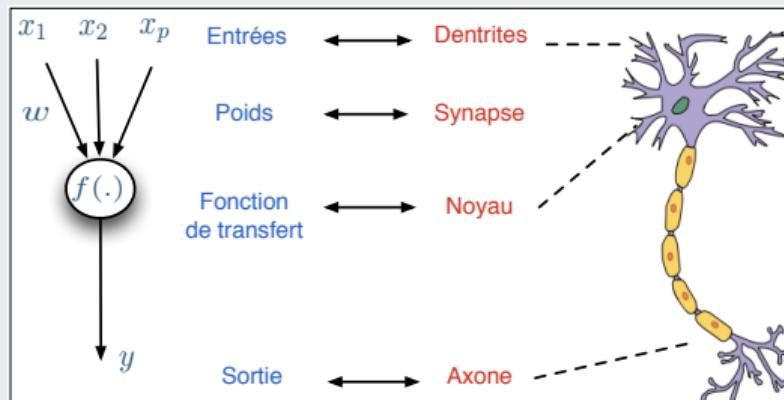
Le neurone biologique (2/2)

- Une intégration des signaux reçus au cours du temps, c'est-à-dire une sommation des signaux, a lieu au niveau du neurone.
- En général, quand la somme dépasse un certain seuil, le neurone émet à son tour un signal électrique.



Le neurone artificiel

- Le neurone artificiel reprend les principes du fonctionnement du neurone biologique, en particulier par la sommation des entrées et la pondération de la somme de ses entrées par des poids synaptiques (aussi appelés coefficients synaptiques)



Le réseau de neurones

- un réseau de neurones est formé par plusieurs neurones interconnectés ensemble.



Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

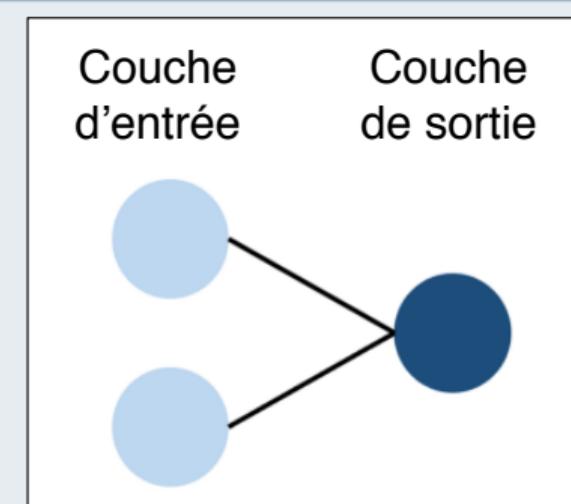
- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

3 Le perceptron multi-couche (PMC)

- Structure du PMC
- Entrainement du PMC
- Algorithme de rétropropagation du gradient
- Fonction d'activations

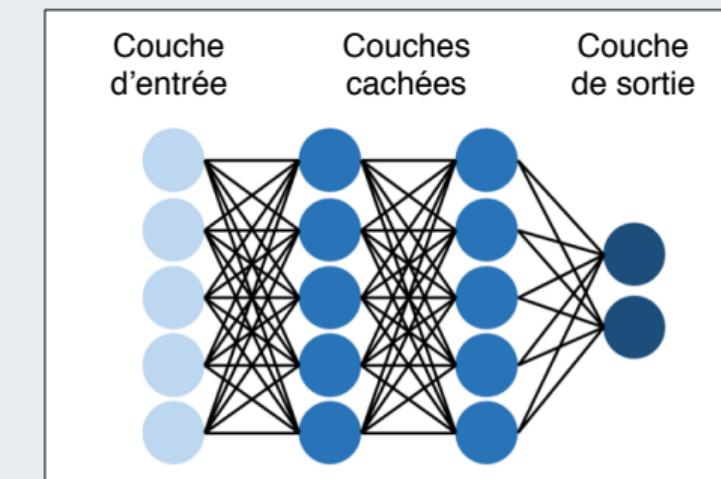
Architecture standard (1/2)

Le perceptron est le plus basique de tous les réseaux neuronaux. C'est un élément fondamental des réseaux neuronaux plus complexes. Il relie simplement une couche d'entrée et une couche de sortie. (Cours 4)



Architecture standard (2/2)

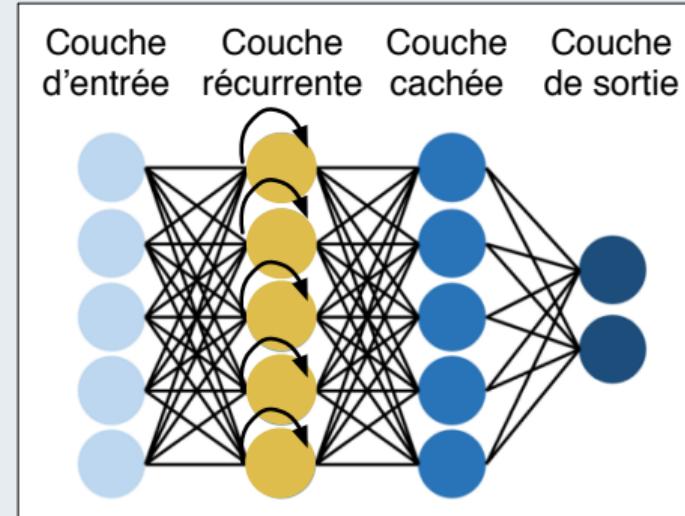
Le réseau **feed-forward** est un ensemble de perceptrons, dans lequel il existe trois types de couches : les couches d'entrée, les couches cachées et les couches de sortie. (Cours 4)



Si le nombre de couche caché est élevé \Rightarrow Réseau de neurone profond (Cours 5)

Architecture récurrente (1/2)

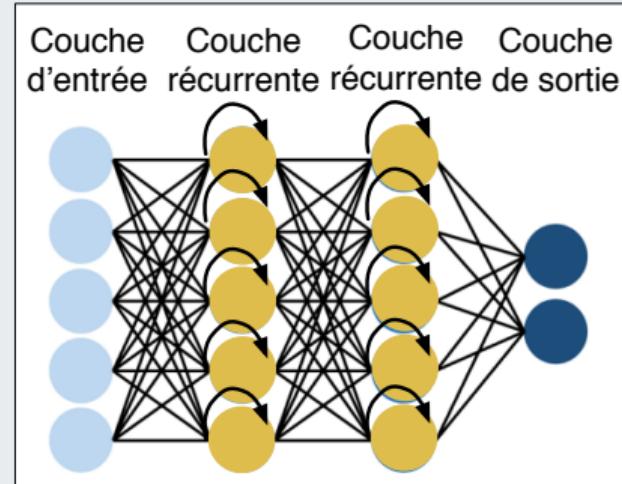
Le réseau de neurone récurrent (RNN) est un type de réseau spécialisé qui contient des boucles et se répète lui-même, d'où le nom "récurrent". Les RNN utilisent le raisonnement de la formation précédente pour prendre des décisions meilleures et prédire les événements à venir. (Cours 7)



Architecture récurrente (2/2)

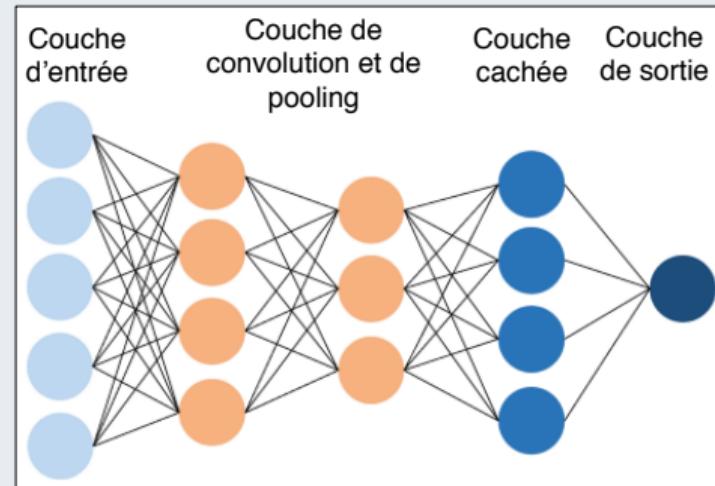
Un réseau Long short-term memory (LSTM) = réseau récurrent à mémoire court et long terme.

Les LSTM possèdent un ensemble de portes pour contrôler quand les informations entrent dans la mémoire, quand elles sont sorties et quand elles sont oubliées. (Cours 7)



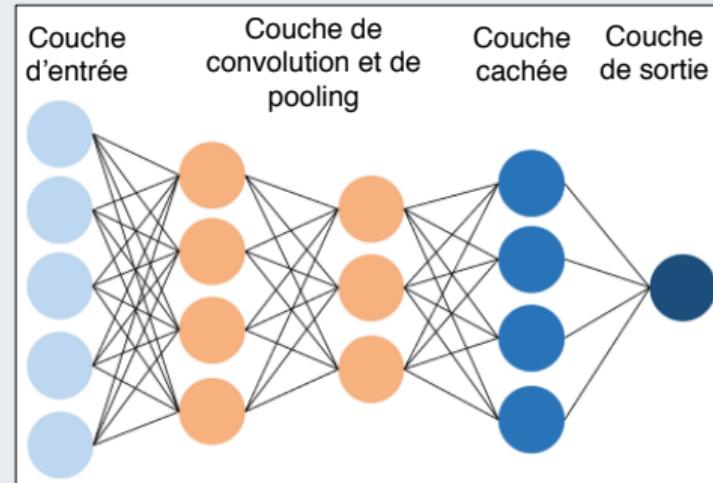
Architecture à convolution (1/2)

Le réseau neuronal convolutif (CNN) apporte une solution à ce problème en utilisant des couches convolutives et de mise en commun pour aider à réduire la dimensionnalité d'une image. (Cours 6)



Architecture à convolution (2/2)

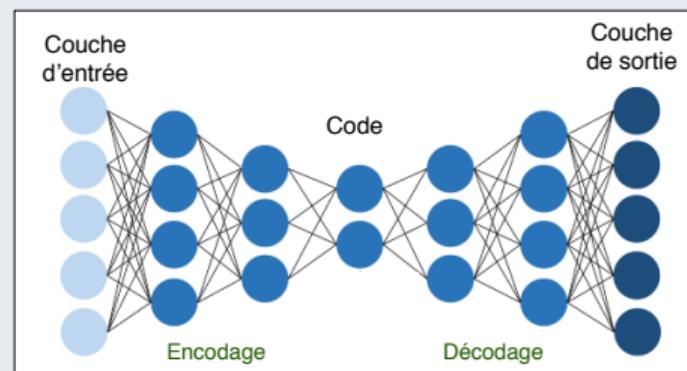
Un réseau adversarial génératif (Generative Adversarial Network, GAN) est un type de réseau spécialisé conçu spécifiquement pour générer des images. Il est composé de deux réseaux : un discriminateur et un générateur. (Cours 8)



Architecture des auto-encodeurs

L'idée fondamentale d'un **auto-codeur** est de prendre des données originales à haute dimensionnalité, de les "comprimer" en données à faible dimensionnalité et à haut niveau d'information, puis de projeter la forme comprimée dans un nouvel espace.

(Cours 8)



Le perceptron

Plan du cours

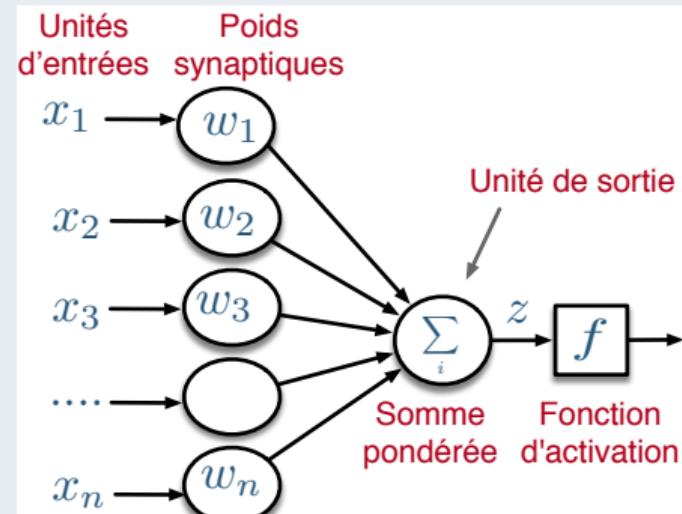
- 1** Introduction
 - Mise en contexte
 - Les différentes architectures des RN
- 2** Le perceptron
 - Structure du perceptron**
 - Fonctions d'activations
 - Entrainement du perceptron
- 3** Le perceptron multi-couche (PMC)
 - Structure du PMC
 - Entrainement du PMC
 - Algorithme de rétropropagation du gradient
 - Fonction d'activations

Le perceptron

- Le perceptron, inventé par Rosenblatt en 1957, est considéré comme le type de réseau de neurones le plus simple.
- Il se fonde sur un neurone artificiel appelé unité logique à seuil (TLU, Threshold Logic Unit) ou encore unité linéaire à seuil (LTU, Linear Threshold Unit).

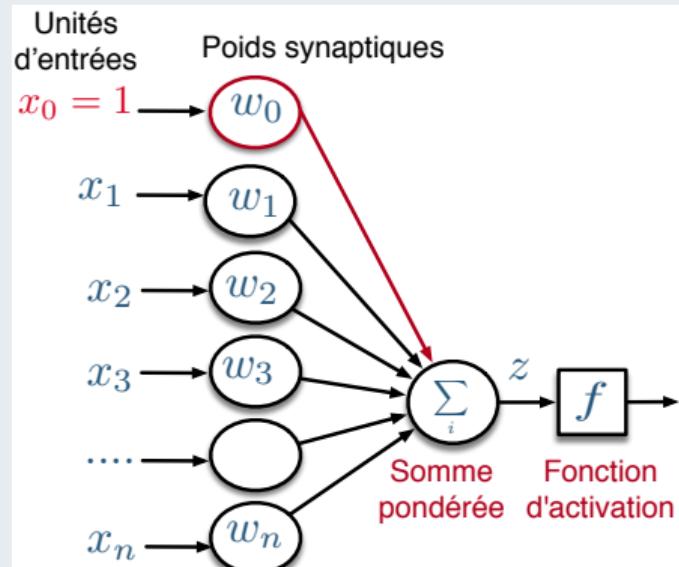
Le perceptron : Structure (1/5)

- Le perceptron est formée d'une première couche d'unités (ou neurones) qui permettent de « lire » les données : chaque unité correspond à une des variables d'entrée.
- Ces unités sont reliées à une seule unité de sortie



Le perceptron : Structure (2/5)

- Comme pour la régression linéaire, le neurone reçoit également une entrée unitaire $x_0 = 1$ dont le poids synaptique est $w_0 = b$.
- Cette entrée, appelée biais b , est toujours activée (elle transmet 1 quelles que soient les données).
- Elle sert à contrôler le seuil d'activation.



Le perceptron : Structure (3/5)

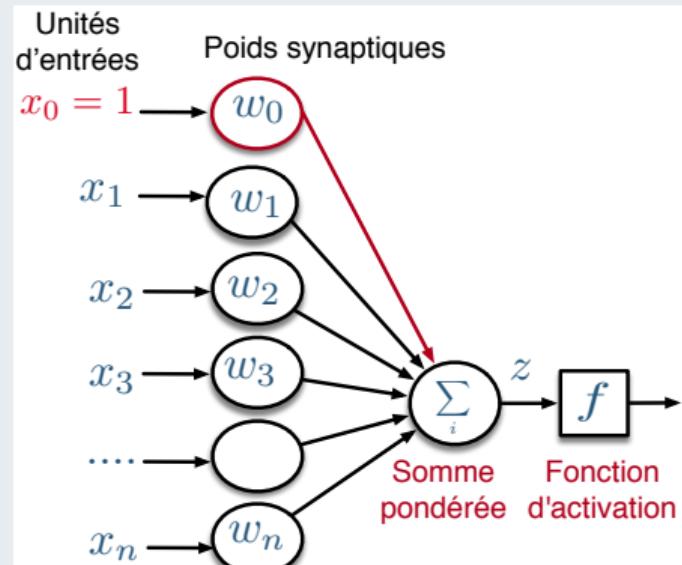
On obtient la somme pondérée des entrées :

$$z = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

$$= w_0 + \sum_{j=1}^n w_j x_j = \mathbf{x}^T \mathbf{w}$$

$$= \mathbf{x}^T \mathbf{w}$$

\mathbf{x} est le vecteur des entrées et \mathbf{w} est le vecteur des poids synaptiques.



Le perceptron : Structure (4/5)

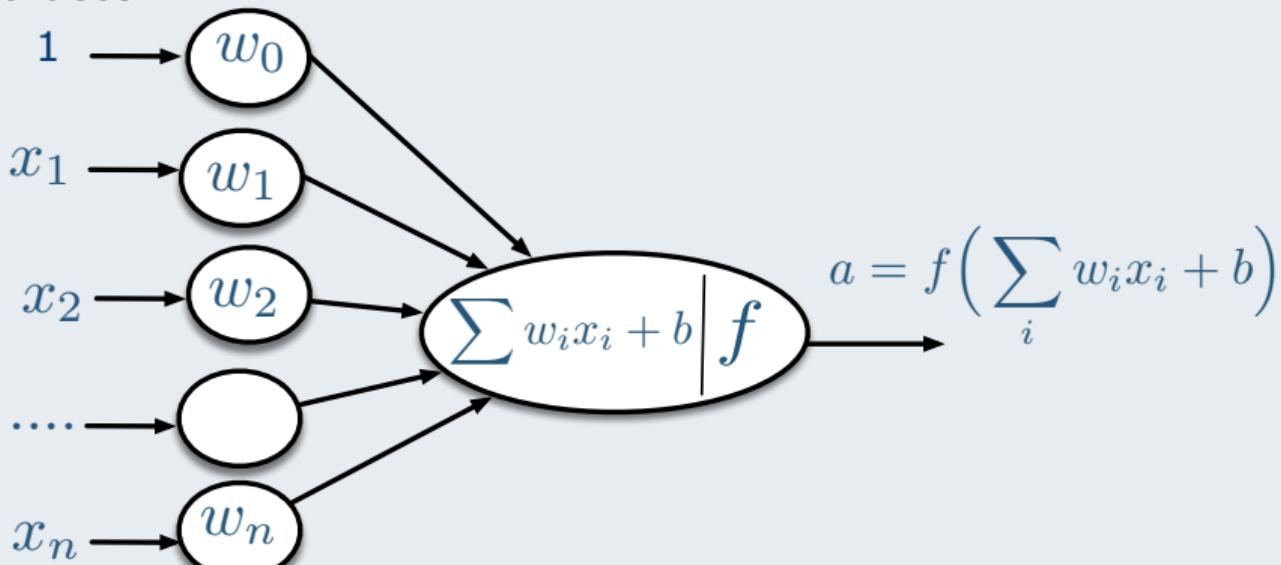
- À l'unité de sortie, est appliquée une fonction d'activation (aussi appelée *fonction de transfert*)

$$a(z) = h_w(\mathbf{x}) = f\left(w_0 + \sum_{i=1}^n w_i x_i\right) \quad (1)$$

- Les fonctions d'activations f les plus connues pour le perceptron sont la fonction à seuil et la fonction signe

Le perceptron : Structure (5/5)

Unités
d'entrées



Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

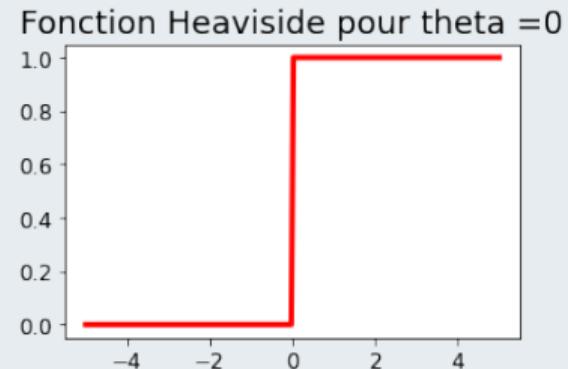
3 Le perceptron multi-couche (PMC)

- Structure du PMC
- Entrainement du PMC
- Algorithme de rétropagation du gradient
- Fonction d'activations

Fonction d'activation : Fonction à seuil (1/2)

- La fonction à seuil aussi appelée *Heaviside* de seuil $\theta \in \mathbb{R}$ est définie par :

$$\begin{cases} f(x) = 0 & \text{si } x < \theta \\ f(x) = 1 & \text{si } x \geq \theta \end{cases} \quad (2)$$

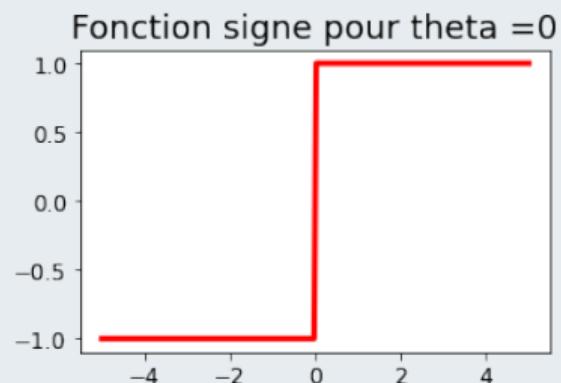


Autrement dit, la fonction *Heaviside* $f(x)$ prend la valeur 0 si $x < \theta$ sinon elle prend la valeur 1.

Fonction d'activation : Fonction signe (2/2)

- La fonction signe $\theta \in \mathbb{R}$ est définie par :

$$\begin{cases} f(x) = -1 & \text{si } x < \theta \\ f(x) = 1 & \text{si } x \geq \theta \end{cases} \quad (3)$$



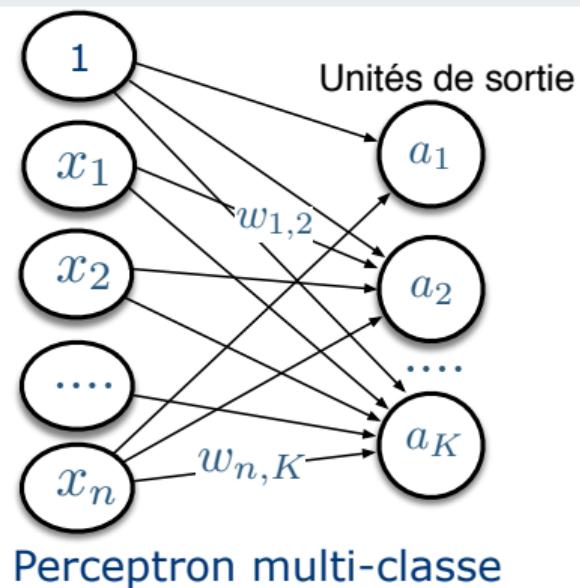
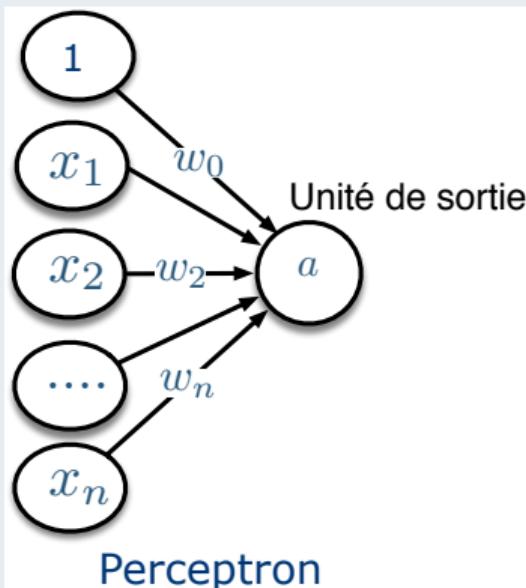
Autrement dit, la fonction Signe $f(x)$ prend la valeur -1 si $x < \theta$ sinon elle prend la valeur 1.

Le perceptron

- En utilisant ces fonctions d'activations, un neurone artificiel (un TLU) peut être employé pour une classification binaire linéaire simple.
- Il calcule une combinaison linéaire des entrées.
- Si le résultat dépasse un seuil, il présente en sortie la classe positive. Sinon, il présente la classe négative.
- Par exemple, nous pouvons utiliser un seul TLU pour classer les iris en fonction de la longueur et de la largeur des pétales \Rightarrow Trouver les valeurs appropriées pour les poids synaptiques.

Le perceptron multi-classes (1/3)

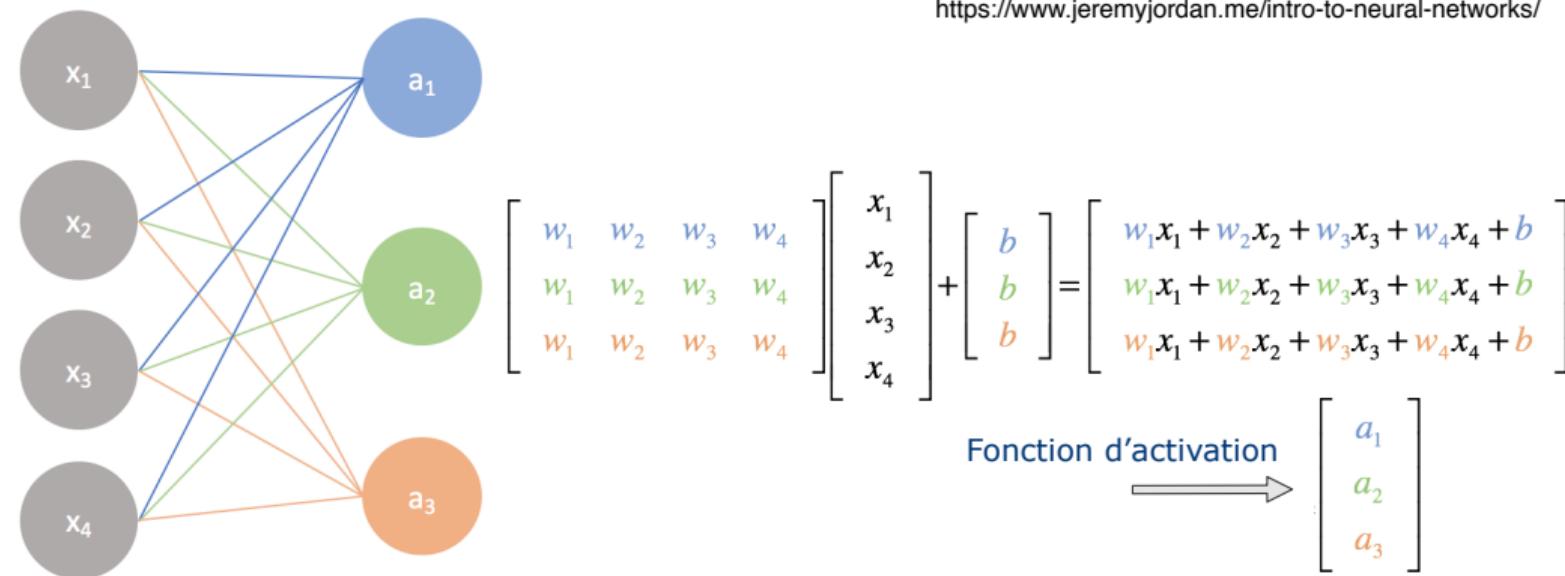
Lors que nous avons plusieurs classes binaires en sortie, nous avons besoin d'un perceptron multi-classes.



Le perceptron multi-classes (2/3)

On obtient la représentation matricielle suivante :

<https://www.jeremyjordan.me/intro-to-neural-networks/>



Le perceptron multi-classes (3/3)

- Lorsque nous avons plusieurs observations, ces dernières sont structurées dans une matrice \mathbf{X} qui comprend une observation dans chaque ligne et une variable (caractéristiques ou features) dans chaque colonne.
- Nous pouvons donc calculer les sorties d'une couche de neurones artificiels pour plusieurs observations à la fois selon :

$$h_{\mathbf{W}, \mathbf{b}}(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b}) \quad (4)$$

Le neurone artificiel

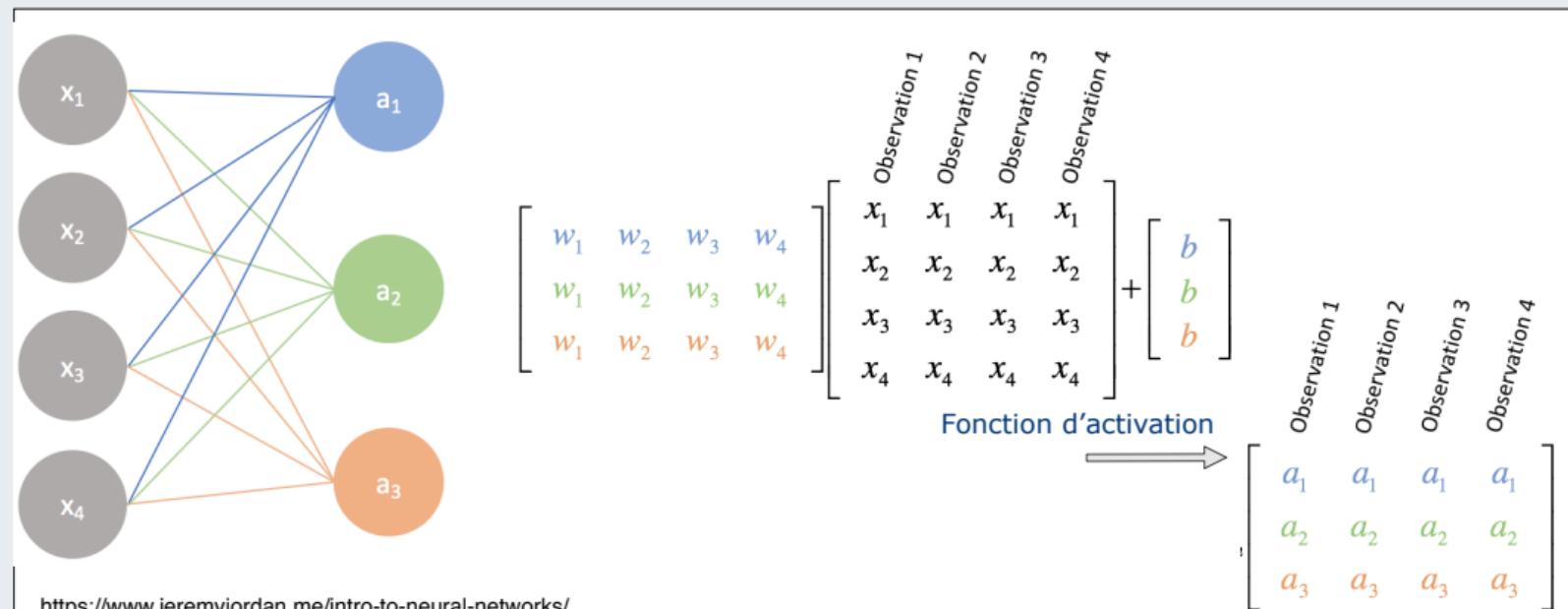
$$h_{\mathbf{W}, \mathbf{b}}(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b}) \quad (5)$$

avec

- **W** la matrice des poids des connexions. Elle comprend une ligne par neurone d'entrée et une colonne par neurone artificiel de la couche.
- **b** est le vecteur de termes constants b contient tous les poids des connexions entre le neurone de terme constant et les neurones artificiels. Il comprend un terme constant par neurone artificiel.
- La fonction ϕ est la fonction d'activation (Fonction seuil ou fonction signe par exemple)

Le perceptron multi-classes

On obtient la représentation matricielle suivante :



Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

3 Le perceptron multi-couche (PMC)

- Structure du PMC
- Entrainement du PMC
- Algorithme de rétropropagation du gradient
- Fonction d'activations

Entrainement du perceptron (1/5)

- L'apprentissage du perceptron simple a pour objectif la détermination des valeurs des poids synaptiques qui permettent de fournir la sortie (classe) y_i à partir des données d'apprentissage \mathbf{x}_i .
- Autrement dit, il s'agit de déterminer les valeurs des poids synaptiques à partir des paires $(\mathbf{x}_i, y_i) \in D$ qui minimisent l'erreur entre les sortie obtenue et souhaitée
- D étant l'ensemble des données d'apprentissage.

Entrainement du perceptron (2/5)

- L'algorithme opère de façon itérative pendant n_{max} itérations.
- Il permet de calculer la prédiction $a(\mathbf{x}_i)$ de l'entrée \mathbf{x}_i et de la comparer à la classe réelle y_i .
- La mise à jour des poids est appelée règle d'apprentissage du perceptron.

Entrainement du perceptron (3/5)

La règle d'apprentissage du perceptron à l'itération $t + 1$ est donnée par :

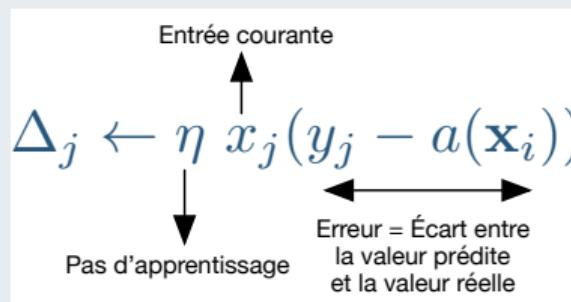
$$w_{i,j}^{(t+1)} = w_{i,j}^{(t)} + \eta(y_j - \hat{y}_j)x_i \quad (6)$$

- $w_{i,j}$ le poids de la connexion entre le neurone d'entrée i et le neurone de sortie j
- x_i est i ème valeur d'entrée de l'observation d'entraînement
- \hat{y}_j et y_j sont respectivement les sortie obtenue et souhaitée du neurone de sortie j pour l'observation d'entraînement
- η le taux d'apprentissage

Entrainement du perceptron (4/5)

$$w_{i,j}^{(t+1)} = w_{i,j}^{(t)} + \eta(y_j - \hat{y}_j)x_i$$

- Si $y_i \neq a(\mathbf{x}_i)$, les éléments du vecteur poids \mathbf{w} sont mis à jour en fonction de la différence entre les valeurs de y_i et de $a(\mathbf{x}_i)$.
- Cette règle de mise à jour est appelé règle Delta (Δ).



Entrainement du perceptron (5/5)

$$w_{i,j}^{(t+1)} = w_{i,j}^{(t)} + \eta(y_j - \hat{y}_j)x_i \quad (7)$$

- La constante η est appelée pas d'apprentissage ou taux d'apprentissage
- Le critère d'arrêt peut être le nombre maximal d'itérations ou bien une valeur d'erreur fixée au préalable.

Exemple sur le perceptron simple : Données Iris (1/5)

- Soit l'ensemble des données Iris.
- Pour illustrer le perceptron simple, nous allons considérer la classe d'espèce Setosa
- nous allons considérer les deux caractéristiques Petal length et Petal width

Exemple sur le perceptron simple : Données Iris (2/5)

Pour l'entraînement du modèle, nous utilisons la fonction
`sklearn.linear_model.Perceptron`

```
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron
# instantiate the model
per_clf = Perceptron(max_iter=100, tol=None, random_state=42)
# fit the model with data
per_clf.fit(X, y)
```

Exemple sur le perceptron simple : Données Iris (3/5)

Dans notre cas, on obtient :

```
print(per_clf.intercept_)
[4.]
print(per_clf.coef_)
[[-1.4 -2.2]]
```

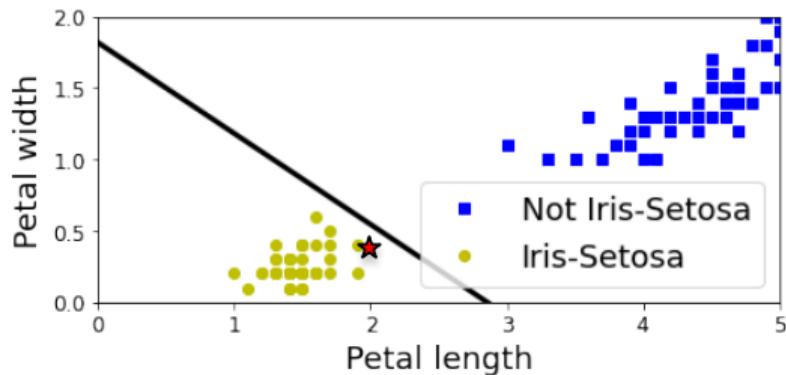
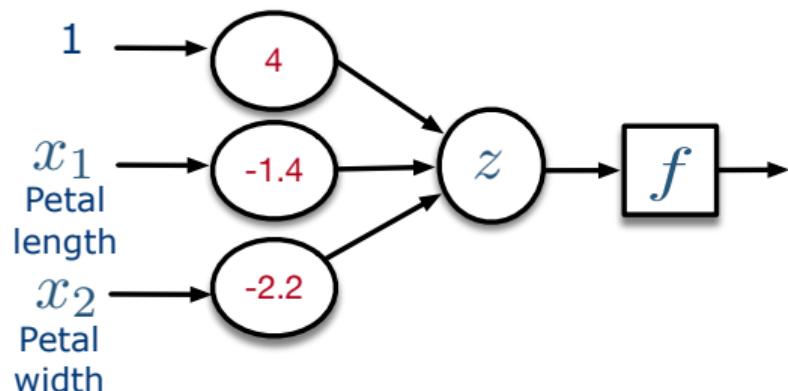
Ce qui veut dire que : $w_0 = 4$

$w_1 = -1.4$

$w_2 = -2.2$

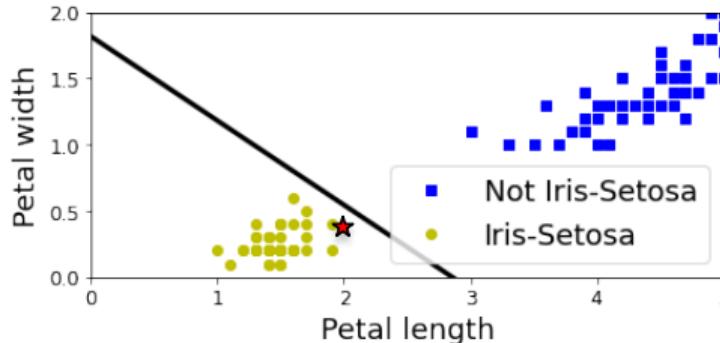
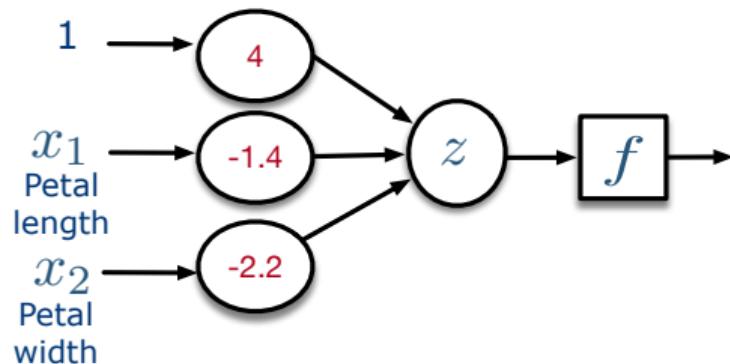
Exemple sur le perceptron simple : Données Iris (4/5)

On peut ainsi schématiser cela par :



$z = 4 - 1.4 \times x_1 - 2.2 \times x_2$. Si $x_1 = 2$ et $x_2 = 0.5$ alors $z = 0.1 > 0$
 $\Rightarrow \hat{y} = 1$ et donc la classe est Iris-Setosa

Exemple sur le perceptron simple : Données Iris (5/5)



Évidemment les valeurs peuvent être prédites directement :

```
print(per_clf.predict([[2, 0.5]]))
```

```
[1]
```

```
print(per_clf.predict([[4, 1.5]]))
```

```
[0]
```

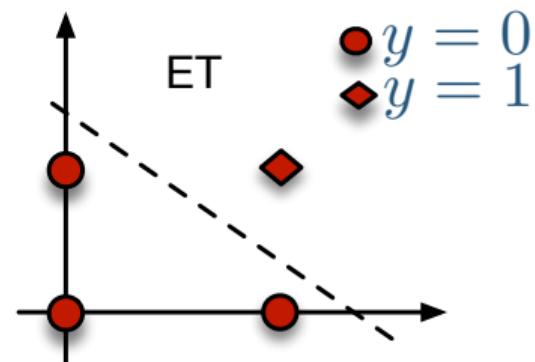
Exemple sur le perceptron simple : Fonction ET (1/9)

- L'objectif de cet exemple est de simuler à l'aide d'un perceptron la fonction logique ET (AND) qui prend en entrée deux valeurs binaires (0 ou 1) et qui retourne 1 si les deux entrées valent 1, sinon elle retourne 0.
- La fonction d'activation considérée est une fonction à seuil avec $\theta = 0$
- L'algorithme d'apprentissage sera exécuté pour $t_{max} = 4$, c'est-à-dire pour 4 itérations.
- La constante d'apprentissage η étant fixée à 0.1.

Exemple sur le perceptron simple : Fonction ET (2/9)

$$\left\{ \begin{array}{l} w_0 = b = 0.1 \\ w_1 = 0.2 \\ w_2 = 0.05 \end{array} \right.$$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



La fonction d'activation considérée est une fonction à seuil avec $\theta = 0$
Le taux d'apprentissage η étant fixée à 0.1. $t_{max} = 4$

Exemple sur le perceptron simple : Fonction ET (3/9)

- Nous commençons par initialiser aléatoirement les poids synaptiques. Soit :

$$\begin{cases} w_0 = b = 0.1 \\ w_1 = 0.2 \\ w_2 = 0.05 \end{cases}$$

Ceci veut dire qu'initialement la frontière de décision est décrite par l'équation :

$$0.1 + 0.2x_1 + 0.05x_2 = 0 \iff x_2 = -4x_1 - 2$$

Exemple sur le perceptron simple : Fonction ET (4/9)

- Nous considérons la première observation $\mathbf{x}_1 = [0, 0]$ dont la sortie $y_1 = 0$,

$$\begin{cases} x_1 = 0 \\ x_2 = 0 \\ y_1 = 0 \end{cases}$$

Selon l'équation 1, $a(\mathbf{x}_1) = f(0.1 + 0.2 \times 0 + 0.05 \times 0) = f(0.1) = 1$

- Nous procédons, ensuite, à une mise à jour des poids selon :

$$\begin{cases} w_0 = 0.1 + 0.1 \times 1 \times (0 - 1) = 0 \\ w_1 = 0.2 + 0.1 \times 0 \times (0 - 1) = 0.2 \\ w_2 = 0.05 + 0.1 \times 0 \times (0 - 1) = 0.05 \end{cases}$$

Exemple sur le perceptron simple : Fonction ET (5/9)

- Nous considérons, ensuite, la deuxième observation $\mathbf{x}_2 = [0, 1]$ dont la sortie $y_2 = 0$,

$$\begin{cases} x_1 = 0 \\ x_2 = 1 \\ y_2 = 0 \end{cases}$$

On a,

$$a(\mathbf{x}_2) = f(0 + 0.2 \times 0 + 0.05 \times 1) = f(0.05) = 1$$

- Nous procédons à une mise à jour. Nous obtenons :

$$\begin{cases} w_0 = 0 + 0.1 \times 1 \times (0 - 1) = -0.1 \\ w_1 = 0.2 + 0.1 \times 0 \times (0 - 1) = 0.2 \\ w_2 = 0.05 + 0.1 \times 1 \times (0 - 1) = -0.05 \end{cases}$$

Exemple sur le perceptron simple : Fonction ET (6/9)

- Nous considérons, ensuite, la deuxième observation $\mathbf{x}_2 = [0, 1]$ dont la sortie $y_2 = 0$,

$$\begin{cases} x_1 = 0 \\ x_2 = 1 \\ y_2 = 0 \end{cases}$$

On a,

$$a(\mathbf{x}_2) = f(0 + 0.2 \times 0 + 0.05 \times 1) = f(0.05) = 1$$

- Nous procédons à une mise à jour. Nous obtenons :

$$\begin{cases} w_0 = 0 + 0.1 \times 1 \times (0 - 1) = -0.1 \\ w_1 = 0.2 + 0.1 \times 0 \times (0 - 1) = 0.2 \\ w_2 = 0.05 + 0.1 \times 1 \times (0 - 1) = -0.05 \end{cases}$$

Exemple sur le perceptron simple : Fonction ET (7/9)

- Nous considérons, la troisième observation à traiter $\mathbf{x}_3 = [1, 0]$ dont la sortie $y_3 = 0$,

$$\begin{cases} x_1 = 1 \\ x_2 = 0 \\ y_3 = 0 \end{cases}$$

$$a(\mathbf{x}_3) = f(-0, 1 + 0.2 \times 1 - 0.05 \times 0) = f(0.1) = 1$$

- Nous procédons, ensuite, à une mise à jour selon :

$$\begin{cases} w_0 = -0.1 + 0.1 \times 1 \times (0 - 1) = -0.2 \\ w_1 = 0.2 + 0.1 \times 1 \times (0 - 1) = 0.1 \\ w_2 = -0.05 + 0.1 \times 0 \times (0 - 1) = -0.05 \end{cases}$$

Exemple sur le perceptron simple : Fonction ET (8/9)

- Nous considérons, la quatrième observation à traiter $\mathbf{x}_4 = [1, 1]$ dont la sortie $y_4 = 1$,

$$\begin{cases} x_1 = 1 \\ x_2 = 1 \\ y_4 = 1 \end{cases}$$

$$a(\mathbf{x}_4) = f(-0.2 + 0.1 \times 1 - 0.05 \times 1) = f(-0.15) = 0$$

- Nous procédons à une mise à jour selon :

$$\begin{cases} w_0 = -0.2 + 0.1 \times 1 \times (1 - 0) = -0.1 \\ w_1 = 0.1 + 0.1 \times 1 \times (1 - 0) = 0.2 \\ w_2 = -0.05 + 0.1 \times 1 \times (1 - 0) = 0.05 \end{cases}$$

Exemple sur le perceptron simple : Fonction ET (9/9)

$$\begin{cases} w_0 = -0.2 + 0.1 \times 1 \times (1 - 0) = -0.1 \\ w_1 = 0.1 + 0.1 \times 1 \times (1 - 0) = 0.2 \\ w_2 = -0.05 + 0.1 \times 1 \times (1 - 0) = 0.05 \end{cases}$$

- À la quatrième itérations nous obtenons la frontière de décision décrite par l'équation : $-0.1 + 0.2x_1 + 0.05x_2 = 0$

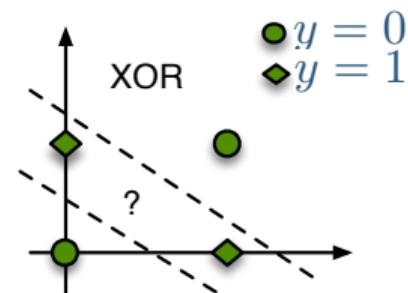
Limite du perceptron (1/2)

- Le perceptron est incapable de distinguer les données non séparables linéairement.
- Il ne peut construire qu'une frontière linéaire comme pour la fonction ET et OU logique
- Or, la majorité des problèmes de classification ne sont pas linéairement séparables, d'où le besoin de déterminer des modèles plus complexes qui permettent de modéliser des frontières plus complexes.

Limite du perceptron (2/2)

- C'est le cas de la fonction XOR logique qui en dépit de sa simplicité ne peut être modélisée par une frontière linéaire
- Solution : Empiler plusieurs perceptrons pour contourner certaines de leurs limites
⇒ Le perceptron multicouche

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



Le perceptron multi-couche (PMC)

Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

3 Le perceptron multi-couche (PMC)

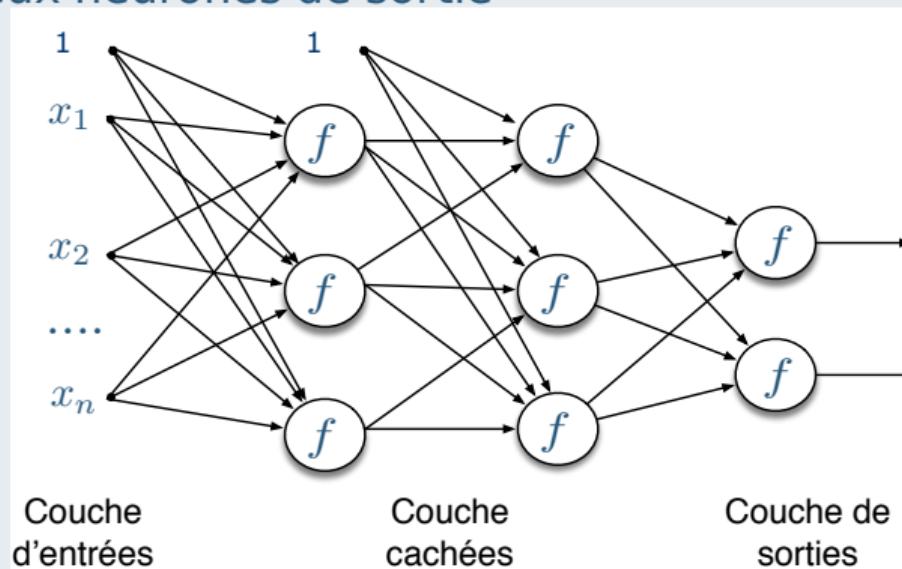
- Structure du PMC
- Entrainement du PMC
- Algorithme de rétropropagation du gradient
- Fonction d'activations

Le perceptron multi-couche (PMC) : structure (1/2)

- Comme son nom l'indique, le PMC est une succession de neurones organisés en plusieurs couches.
- La première couche est reliée aux entrées.
- Ensuite, on retrouve un ensemble de couches intermédiaires qui ne sont pas visibles et qui sont appelées des couches cachées.
- Chaque couche est reliée à la couche précédente.
- La dernière couche du PMC définit la réponse des neurones et produit les différentes sorties

Le perceptron multi-couche (PMC) : structure (2/2)

Perceptron multicouche avec n entrées, deux couches cachées de trois neurones et deux neurones de sortie



Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

3 Le perceptron multi-couche (PMC)

- Structure du PMC
- Entrainement du PMC**
- Algorithme de rétropropagation du gradient
- Fonction d'activations

Entrainement du PMC (1/3)

- L'apprentissage du perceptron se fait selon la règle Delta.
- Cette règle a été généralisée en 1986 aux réseaux multicouches et formalisée sous l'appellation de la règle de la rétropropagation du gradient de l'erreur.
- Cette dernière consiste à propager l'erreur obtenue à un neurone de sortie d'un réseau à couches à travers le réseau par descente du gradient dans le sens inverse de la propagation des activations.

Entrainement du PMC (2/3)

- ➊ Pour chaque observation d'entraînement, l'algorithme de rétropropagation commence par effectuer une prédiction (passe vers l'avant)
- ➋ Mesure l'erreur
- ➌ Traverse chaque couche en arrière pour mesurer la contribution à l'erreur de chaque connexion (passe vers l'arrière)
- ➍ Termine en ajustant légèrement les poids des connexions de manière à réduire l'erreur (étape de descente de gradient).

Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

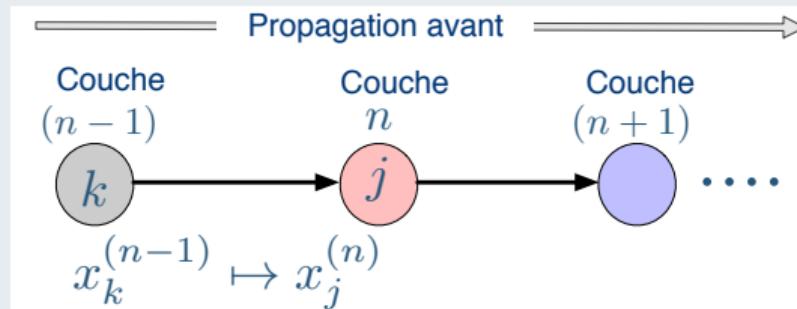
- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

3 Le perceptron multi-couche (PMC)

- Structure du PMC
- Entrainement du PMC
- **Algorithme de rétropropagation du gradient**
- Fonction d'activations

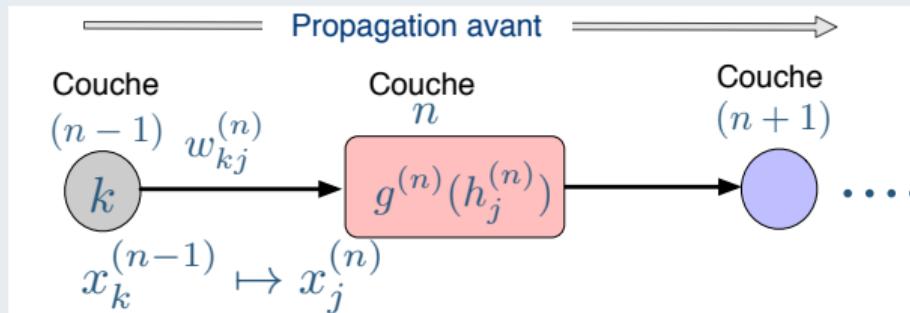
Algorithme de rétropropagation du gradient (1/8)

- Soit un échantillon d'entrée \mathbf{x} du MLP et t l'étiquette cible de cet échantillon
- On propage le signal en avant dans les couches du réseau de neurones : $x_k^{(n-1)} \mapsto x_j^{(n)}$, avec n le numéro de la couche.



Algorithme de rétropropagation du gradient (2/8)

- La propagation vers l'avant se calcule à l'aide la fonction d'activation g , de la fonction d'agrégation $h = \text{produit scalaire}$ entre les poids et les entrées du neurone et des poids synaptiques w_{jk} entre le neurone $x_k^{(n-1)}$ et le neurone $x_j^{(n)}$.



Algorithme de rétropropagation du gradient (3/8)

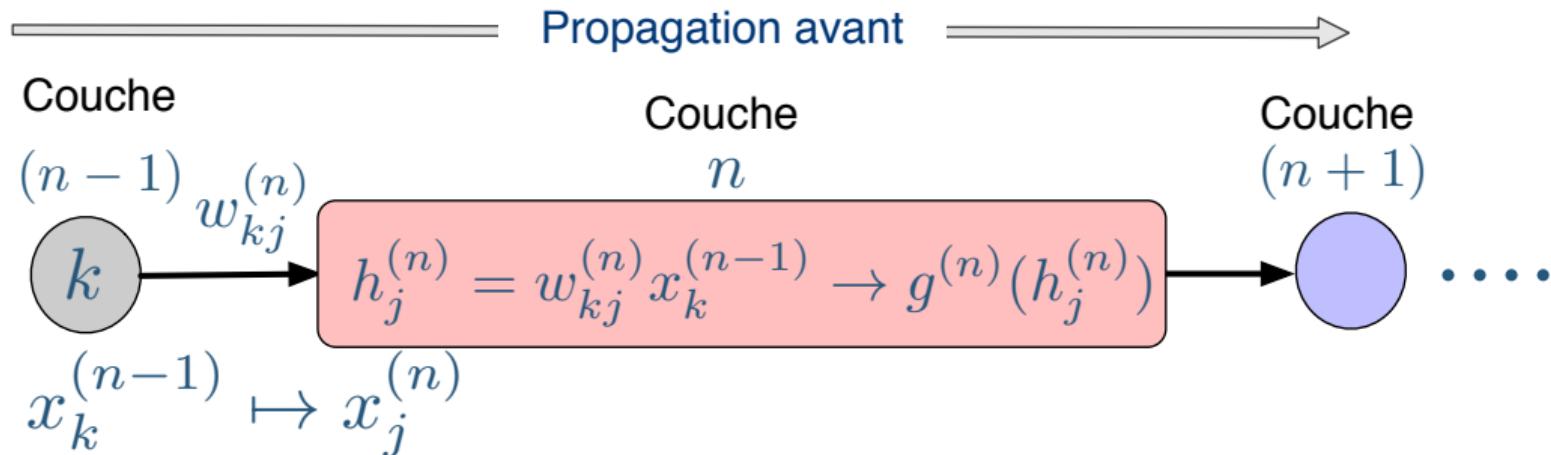
- On a alors la fonction d'agrégation $h_j^{(n)}$ à la couche n :

$$h_j^{(n)} = \sum_k w_{jk}^{(n)} x_k^{(n-1)}$$

- et la fonction d'activation $g^{(n)}$ lui correspondant

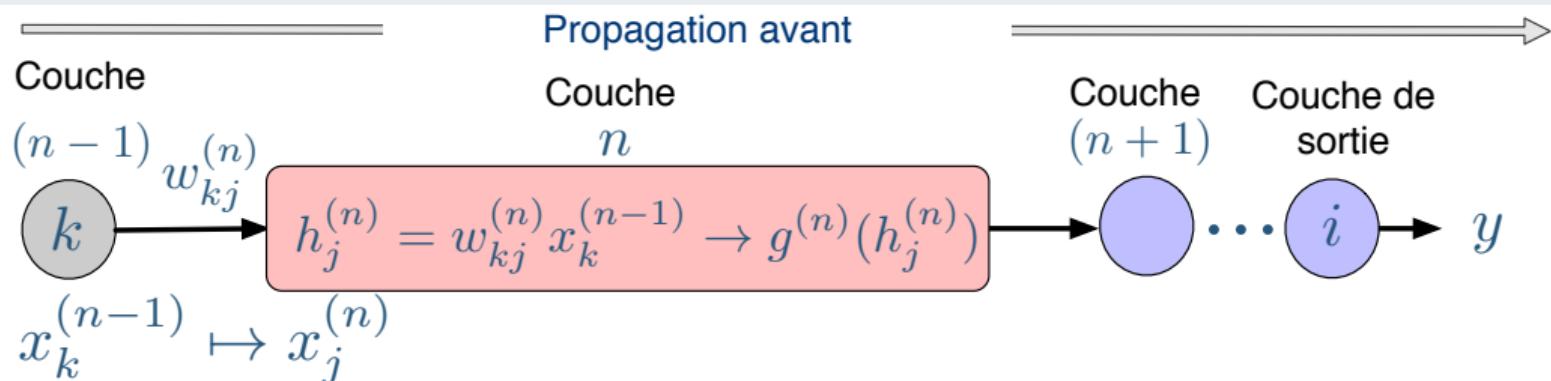
$$x_j^{(n)} = g^{(n)}(h_j^{(n)}) = g^{(n)}\left(\sum_k w_{jk}^{(n)} x_k^{(n-1)}\right)$$

Algorithme de rétropropagation du gradient (4/8)



Algorithme de rétropropagation du gradient (5/8)

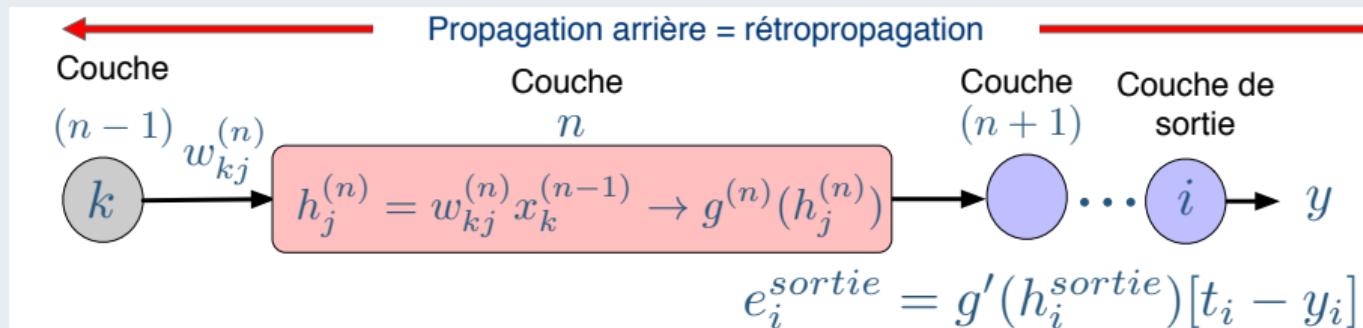
- Lorsque la propagation vers l'avant est terminée, on obtient à la sortie le résultat y



Algorithme de rétropropagation du gradient (6/8)

- On calcule alors l'erreur entre la sortie donnée par le réseau y et la valeur cible t désiré à la sortie pour cet échantillon.
- Pour chaque neurone i dans la couche de sortie, on calcule :

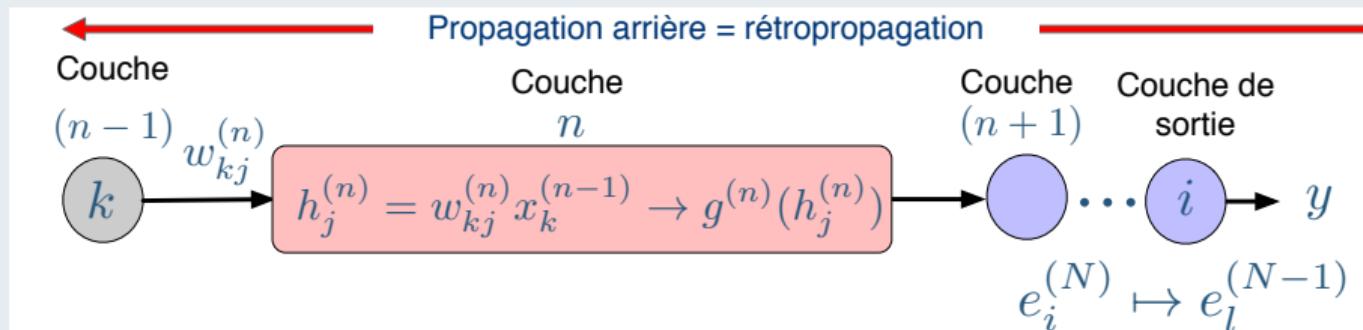
$$e_i^{\text{sortie}} = g'(h_i^{\text{sortie}})[t_i - y_i]$$



Algorithme de rétropropagation du gradient (7/8)

- On propage l'erreur vers l'arrière $e_i^{(n)} \mapsto e_j^{(n-1)}$ grâce à la formule suivante :

$$e_j^{(n-1)} = g'^{(n-1)}(h^{(n-1)})$$

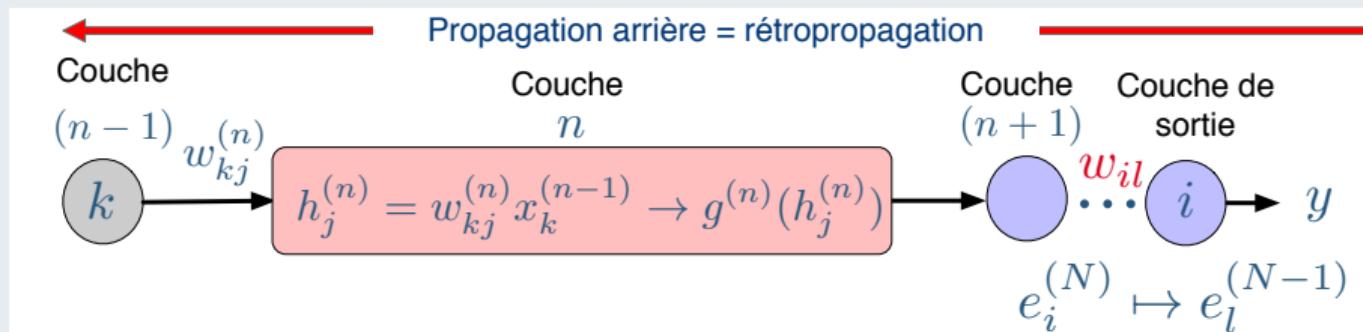


Algorithme de rétropropagation du gradient (8/8)

- Et on met à jour les poids dans toutes les couches :

$$\Delta w_{ij}^{(n)} = \eta e_i^{(n)} x_j^{(n-1)}$$

où η représente le taux d'apprentissage



Plan du cours

1 Introduction

- Mise en contexte
- Les différentes architectures des RN

2 Le perceptron

- Structure du perceptron
- Fonctions d'activations
- Entrainement du perceptron

3 Le perceptron multi-couche (PMC)

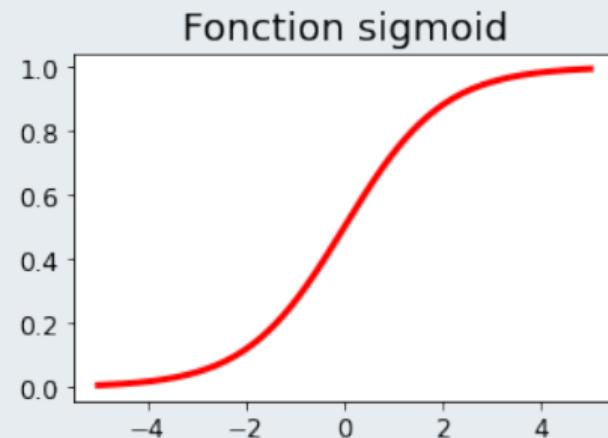
- Structure du PMC
- Entrainement du PMC
- Algorithme de rétropropagation du gradient
- Fonction d'activations

Fonction d'activation : logistique

- L'entraînement du PMC nécessite le choix d'une fonction d'activation.
- La fonction logistique est définie par :

$$f(x) = \frac{1}{1 + e^{-x}} \quad (8)$$

- La fonction logistique possède une dérivée non nulle en tout point, ce qui permet à la descente de gradient de progresser à chaque étape.

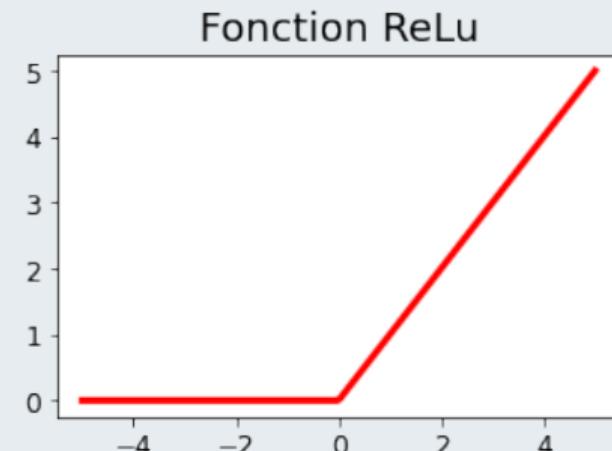


Fonction d'activation : ReLu

- La fonction ReLU (Rectified Linear Unit) est définie par :

$$\begin{cases} f(x) = 0 & \text{si } x < \theta \\ f(x) = x & \text{si } x \geq \theta \end{cases} \quad (9)$$

- Dans la pratique la fonction ReLU fonctionne très bien et a l'avantage d'être rapide à calculer.



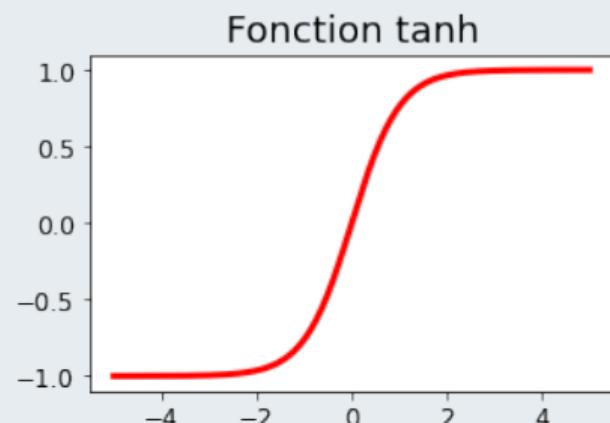
- ReLu n'a pas de valeur de sortie maximale ce qui aide à diminuer certains problèmes au cours de la descente de gradient

Fonction d'activation : La tangente hyperbolique

- La tangente hyperbolique tanh est définie par :

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (10)$$

- tanh a une forme similaire à la fonction logistique. Elle est continue et dérivable, mais ses valeurs de sortie se situent dans l'intervalle -1 à 1.



- La convergence en utilisant tanh est souvent accélérée

Exemple sur le perceptron multicouche (1/2)

- Soit l'ensemble des données Iris.
- Pour illustrer le perceptron multicouche, nous allons considérer les deux caractéristiques Petal length et Petal width
- Pour l'entraînement du modèle, nous utilisons la fonction **MLPClassifier**

Exemple sur le perceptron multicouche (2/2)

```
from sklearn.datasets import load_iris
from sklearn.neural_network import MLPClassifier
# instantiate the model
per_clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(5, 2), random_state=1)
# fit the model with data
per_clf.fit(X, y)
```

- `hidden_layer_sizes` comprend le nombre de couches cachées et le nombre de neurones par couche cachée.
- `alpha` est le pas d'apprentissage
- `activation` permet de choisir la fonction d'activation