Cycle 00

Sciences Industrielles de l'Ingénieur

L'IA c'est quoi ?

Chapitre 2

Introduction aux réseaux de neurones

Savoirs et compétences :

Cours

- ❒ *Analyser les principes d'intelligence artificielle.*
  - *Phases d'apprentissage et d'inférence.*
  - *Réseaux de neurones (couches d'entrée, cachées et de sortie, neurones, biais, poids et fonction d'activation).*
- ❒ *Résoudre un problème en utilisant une solution d'intelligence artificielle :*
  - *Apprentissage supervisé.*
  - *Choix des données d'apprentissage.*
  - *Mise en œuvre des algorithmes (réseaux de neurones).*
  - *Phases d'apprentissage et d'inférence.*

Xavier Pessoles & Anthony Meurdefroid
PSI⋆ – MP

1 Introduction

1.1 Bref historique

1.2 Exemples d'applications des réseaux de neurones

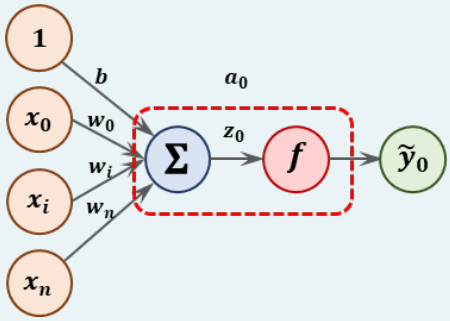
2 Le neurone, les réseaux de neurones

2.1 Modèle de neurone

Définition — Neurone (ou perceptron).

Prenons la représentation suivante pour un neurone. On note :

- $\mathbf{X}$ le vecteur d'entrée et x_i les données de la couche d'entrée ;
- w_i les poids (poids synaptiques) ;
- b le biais ;
- z_0 la somme pondérée des entrée ;
- f une fonction d'activation ;
- $\tilde{y}_0$: la valeur de sortie du neurone.



On a donc, dans un premier temps :

$$z_0 = b + \sum_{i=0}^{n} w_i x_i.$$

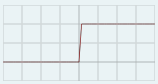
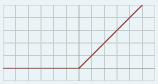
Après la fonction d'activation, on a donc en sortie du neurone :

$$\tilde{y}_0 = f(z_0) = f\left(b + \sum_{i=0}^{n} w_i x_i\right).$$

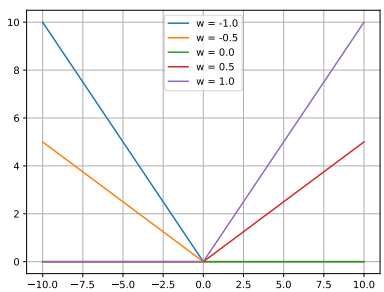
R

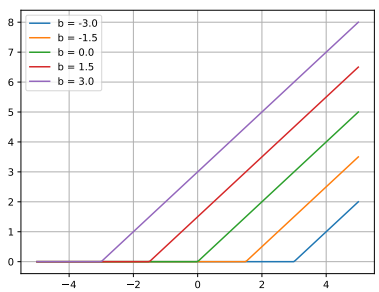
1. La notation tilde ($\tilde{y}_0$) vient du fait que la valeur de sortie d'une neurone est une valeur estimée qu'il faudra comparer à y_0 valeur de l'étiquette utilisée pour l'apprentissage supervisé.
2. Par la suite, dans la représentation graphique on ne fera pas apparaître la somme pondérée et la fonction d'activation, mais seulement la valeur de sortie du neurone (notée par exemple a_0).

Définition — Fonction d'activation. Les fonctions d'activation sont des fonctions mathématiques appliquées au signal de sortie (z). Il est alors possible d'ajouter des non linéarités à la somme pondérée. On donne ci-dessous quelques fonctions usuelles :

Identité	Heaviside	Logistique (sigmoïde)	Unité de rectification linéaire (ReLU)
			
$f(x) = x$	$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ 1 \text{ si } x \geq 0 \end{cases}$	$f(x) = \dfrac{1}{1+e^{-x}}$	$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$

R Influence des poids et des biais sur la sortie du perceptron en utilisant une fonction d'activation ReLU.

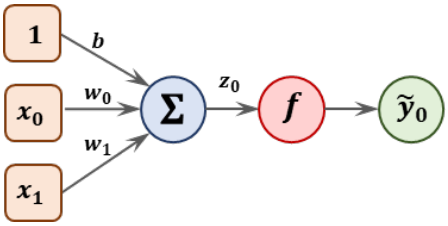




On peut ainsi voir qu'avec la fonction d'activation ReLU, plus le poids sera grand en valeur absolu, plus la neurone amplifiera le signal d'entrée.
Le biais permettra de prendre en compte le « niveau » du signal d'entrée à partir duquel, le signal doit être amplifié, ou non.

▪ Exemple

Prenons un neurone à deux entrées binaires.
Initialisation les poids et le biais avec des valeurs aléatoires : $w_0 = -0,3$, $w_1 = 0,8$ et $b = 0,2$.



On peut donc évaluer l'ensemble des sorties calculable par le neurone.

x_0	x_1	z	Id.	H.	Sig.	ReLu
0	0	0,2	0,2	1	0.549	0,2
0	1	1	1	1	0.731	1
1	0	-0.1	-0.1	0	0.475	0
1	1	0.7	0.7	1	0.668	0.7

▪

Cycle 04

3 Réseaux de neurones

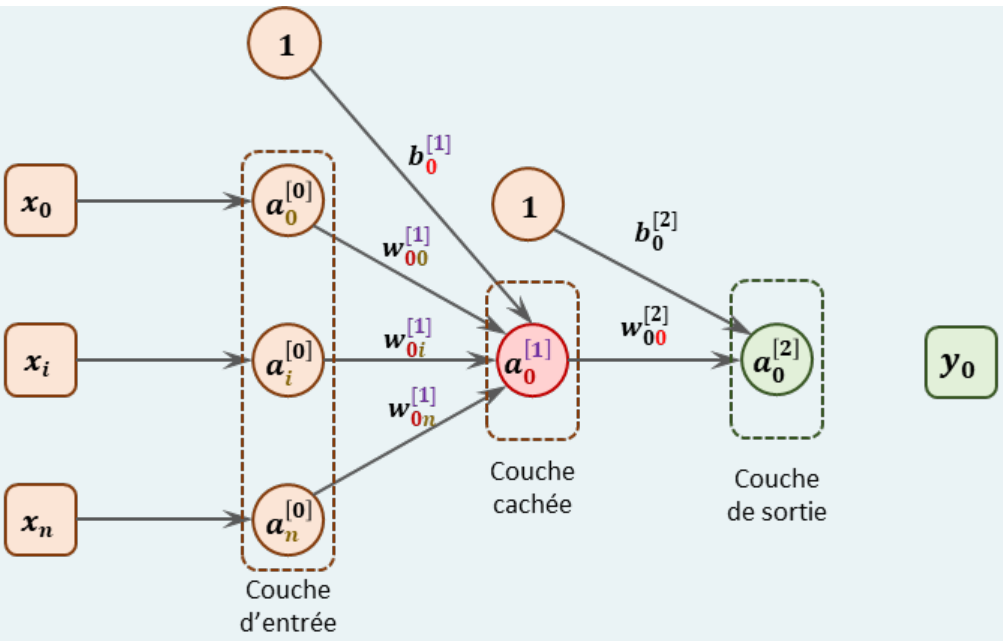
`https://playground.tensorflow.org/`

3.1 Modélisation d'un réseau de neurones

Définition — Couches.

Un réseau de neurones est un ensemble de neurones reliés, par couches, entre eux.
Dans un réseau de neurones **dense** tous les neurones de la couche i seront reliés à tous les neurones de la couche $i+1$.

- Couche d'entrée : cette couche est une copie de l'ensemble des données d'entrées. Le nombre de neurones de cette couche correspond donc aux nombre de données d'entrées. On note $\mathbf{X} = (x_0, ..., x_n)$ le vecteur d'entrées.
- Couche cachée (ou couche intermédiaire) : il s'agit d'une couche qui a une utilité intrinsèque au réseau de neurones. Ajouter des neurone dans cette couche (ou ces couches) permet donc d'ajouter de nouveaux paramètres. Pour une couche, la même fonction d'activation est utilisée pour tous les neurones. En revanche la fonction d'activation utilisée peut être différente pour deux couches différentes. Les fonctions d'activations des couches intermédiaires sont souvent non linéaires.
- Couche de sortie : le nombre de neurones de cette couche correspond au nombre de sorties attendues. La fonction d'activation de la couche de sortie est souvent linéaire. On note $\mathbf{Y} = \left(y_0, ..., y_y\right)$ le vecteur des sorties.



En utilisant la loi de comportement du modèle de perceptron, on peut donc exprimer $\mathbf{Y} = \mathscr{F}(\mathbf{X})$ où $\mathscr{F}$ est une fonction dépendant des entrées, des poids et des biais.

Notations :

- on note $w_{jk}^{[\ell]}$ les poids permettant d'aller vers la couche ℓ depuis le neurone k vers le neurone j ;
- $b_j^{[\ell]}$ le biais permettant d'aller sur le neurone j de la couche ℓ ;
- $f^{[\ell]}$ la fonction d'activation de la couche ℓ ;
- $n^{[\ell]}$ le nombre de neurones de la couche ℓ.

Définition — Équation de propagation. Pour chacun des neurones $a_j^{[\ell]}$ on peut donc écrire l'équation de propagation qui lui est associé :

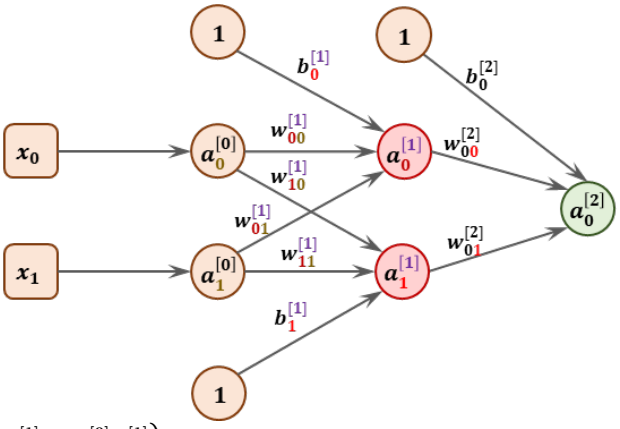
$$a_j^{[\ell]} = f^{[\ell]}\left(\sum_{k=0}^{n^{[\ell-1]}}\left(w_{jk}^{[\ell]} a_k^{[\ell-1]}\right) + b_j^{[\ell]}\right) = f^{[\ell]}\left(z_j^{[\ell]}\right).$$

▪ Exemple

Prenons un réseau de neurones à 3 couches :

- 1 couche d'entrée à 2 neurones ;
- 1 couche cachée à 2 neurones, de fonction d'activation f_1 ;
- 1 couche de sortie à 1 neurone, de fonction d'activation f_2 ;

Initialisation les poids et le biais avec des valeurs aléatoires : $w_0 = -0{,}3$, $w_1 = 0{,}8$ et $b = 0{,}2$.



Il est possible d'écrire que $y_0 = a_0^{[1]} = f_2\left(b_0^{[2]} + w_{00}^{[2]} a_0^{[1]} + w_{01}^{[2]} a_1^{[1]}\right)$.

Par ailleurs : $a_0^{[1]} = f_1\left(b_0^{[1]} + w_{00}^{[1]} a_0^{[0]} + w_{01}^{[1]} a_1^{[0]}\right)$ et $a_1^{[1]} = f_1\left(b_1^{[1]} + w_{10}^{[1]} a_0^{[0]} + w_{11}^{[1]} a_1^{[0]}\right)$.

Au final, on a donc

$$y_0 = a_0^{[1]} = f_2\left(b_0^{[2]} + w_{00}^{[2]}\left(f_1\left(b_0^{[1]} + w_{00}^{[1]} a_0^{[0]} + w_{01}^{[1]} a_1^{[0]}\right)\right) + w_{01}^{[2]}\left(f_1\left(b_1^{[1]} + w_{10}^{[1]} a_0^{[0]} + w_{11}^{[1]} a_1^{[0]}\right)\right)\right)$$

▪

Définition — Paramètres. Les paramètres du réseau de neurones sont les poids et les biais, autant de valeurs que l'entraînement devra déterminer.

Méthode — Calcul du nombre de paramètres – à vérifier.

Soit un jeu de données étiquetées avec n entrées et p sorties.

On construit un réseau possédant ℓ couches et a_ℓ le nombre de neurones de la couche ℓ. Dans ce cas, la première couche est la couche d'entrée ($a_1 = n$) et la dernière couche et la couche de sortie ($a_\ell = p$).

Nombre de poids : $n_w = \sum_{i=1}^{\ell-1} (a_i \times a_{i+1})$.

Nombre de bais : $n_b = \sum_{i=2}^{\ell} (a_i)$.

Au final, le nombre total de paramètre à calculer est donné par $N = n_w + n_b$.

Objectif Soit un jeu de données étiquetées. On note $\mathbf{X}$ le vecteur des données d'entrées. On note $\mathbf{Y}$ le vecteur des données de sorties. On note $\tilde{\mathbf{Y}}$ le vecteur de sortie calculé par le réseau de neurones.

L'objectif de la phase d'apprentissage du réseau de neurones est de déterminer les valeurs de l'ensemble des poids et des biais de telle sorte que l'écart entre $\mathbf{Y}$ et $\tilde{\mathbf{Y}}$ soit minimale.

3.2 Fonction de coût

Dans le but de minimiser l'écart entre la sortie du réseau de neurones et la valeur réelle de la sortie, on utilise une fonction coût (ou fonction de perte). Il est possible de définir plusieurs types de fonctions, notamment en fonction du type de problème à traiter (classification ou régression par exemple).

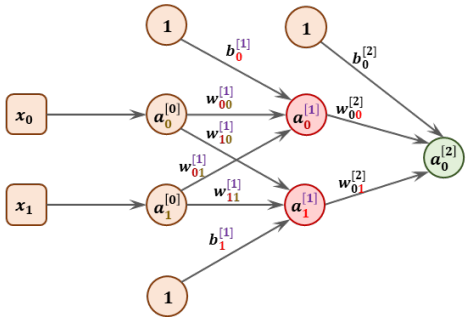
Définition — Fonction coût régression. Notons nb le nombre de données dans la base d'entraînement. Dans le cadre d'un problème de régression, on peut définir la fonction coût comme la moyenne des erreurs quadratique entre la valeur donnée par l'équation de propagation et la valeur de l'étiquette :

$$C = \frac{1}{nb} \sum_{i=1}^{nb} \left(\tilde{\mathbf{Y}}_\mathbf{i} - \mathbf{Y}_\mathbf{i}\right)^2$$

Objectif L'objectif est dès lors de déterminer les poids et les biais qui minimisent la fonction coût.

3.3 Notion de rétropropagation – Descente de gradient

En réutilisant l'exemple ci-contre, nous allons présenter succinctement comment est minimisée la fonction coût. Pour cela, il va falloir dériver la fonction coût par rapport à chacune des variables. Cherchons uniquement à déterminer le coût que par rapport à un seul vecteur d'entrée du jeu d'entraînement. On a alors :



- $C = \left(\tilde{\mathbf{Y}}_\mathbf{i} - \mathbf{Y}_\mathbf{i}\right)^2 = \left(a_0^{[2]} - y\right)^2$;
- $a_0^{[2]} = f^{[2]}\left(z_0^{[2]}\right)$;
- $z_0^{[2]} = \sum_{k=0}^{1} \left(w_{0k}^{[2]} a_k^{[1]}\right) + b_0^{[2]}$.

Commençons par déterminer la dérivée partielle par rapport à un poids de la couche de sortie :

$$\frac{\partial C}{\partial w_{00}^{[2]}} = \frac{\partial C}{\partial a_0^{[2]}} \frac{\partial a_0^{[2]}}{\partial z_0^{[2]}} \frac{\partial z_0^{[2]}}{\partial w_{00}^{[2]}}.$$

De même, on peut calculer la dérivée partielle du coût par rapport au biais : $\frac{\partial C}{\partial b_0^{[2]}} = \frac{\partial C}{\partial a_0^{[2]}} \frac{\partial a_0^{[2]}}{\partial z_0^{[2]}} \frac{\partial z_0^{[2]}}{\partial b_0^{[2]}}$.

Calculons les dérivées nécessaires :

- $\frac{\partial C}{\partial a_0^{[2]}} = 2\left(a_0^{[2]} - y\right)$;
- $\frac{\partial a_0^{[2]}}{\partial z_0^{[2]}} = f'^{[2]}\left(z_0^{[2]}\right)$;
- $\frac{\partial z_0^{[2]}}{\partial w_{00}^{[2]}} = a_0^{[1]}$.
- $\frac{\partial z_0^{[2]}}{\partial b_0^{[2]}} = 1$.

On a donc, $\frac{\partial C}{\partial w_{00}^{[2]}} = 2\left(a_0^{[2]} - y\right) f'^{[2]}\left(z_0^{[2]}\right) a_0^{[1]}$ et $\frac{\partial C}{\partial b_0^{[2]}} = 2\left(a_0^{[2]} - y\right) f'^{[2]}\left(z_0^{[2]}\right)$.

Prenons le cas ou la fonction d'activation est la fonction identité. On a alors $\frac{\partial C}{\partial w_{00}^{[2]}} = 2\left(a_0^{[2]} - y\right) a_0^{[1]}$ et $\frac{\partial C}{\partial b_0^{[2]}} = 2\left(a_0^{[2]} - y\right)$...

On va ainsi pouvoir exprimer $\frac{\partial C}{\partial w_{00}^{[2]}}$, $\frac{\partial C}{\partial w_{01}^{[2]}}$, $\frac{\partial C}{\partial b_{0}^{[2]}}$, ... On pourrait ici écrire 9 équations en fonction des différents poids, des biais et des entrées x_0 et x_1.

À partir de cela, on va modifier les poids comme suit :

- $w_{00,i+1}^{[2]} = w_{00,i}^{[2]} + \eta \frac{\partial C}{\partial w_{00,i}^{[2]}}$;
- $b_{0,i+1}^{[2]} = b_{0,i}^{[2]} + \eta \frac{\partial C}{\partial b_{0,i}^{[2]}}$.

On réitère ensuite les opérations précédentes jusqu'à ce que la fonction coût ait été suffisamment réduite.

Définition — Taux d'apprentissage. On définit l'hyperparamètre $\eta \in [0,1[$ comme étant le taux d'apprentissage. Si ce taux d'apprentissage est très grand, l'algorithme d'apprentissage mettra beaucoup de temps à trouver le minimum. S'il est trop grand, le minimum peut ne jamais être trouvé.

Définition — Gradient.

3.4 Suraprentissage

3.5 Définitions supplémentaires

Définition — Epoch.

4 Pour aller plus loin...

Analyse d'image et réseaux convolutifs. Times series...
Exemples : `https://makina-corpus.com/blog/metier/2017/initiation-au-machine-learning-avec-python-p`

Références

[1] Éric Biernat et Michel Lutz. *Data science : fondamentaux et études de cas.* Eyrolles.