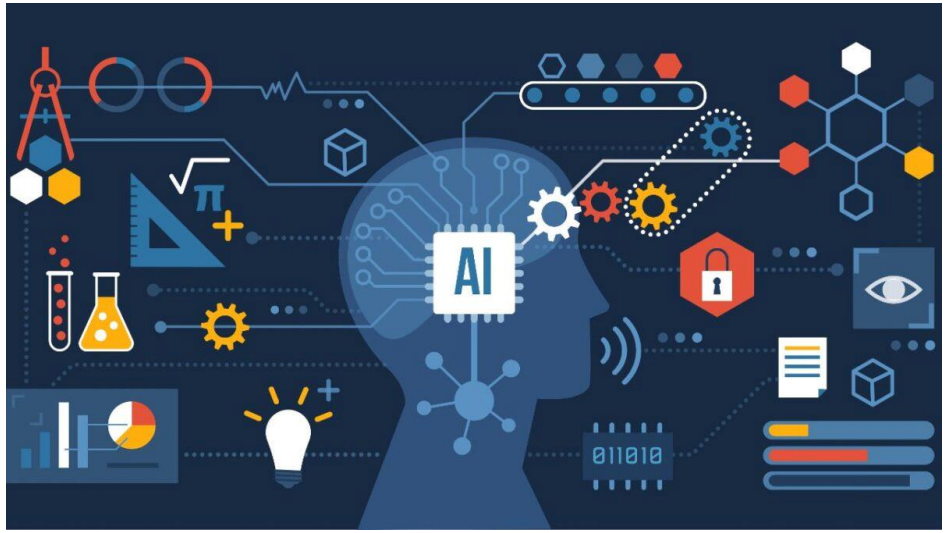
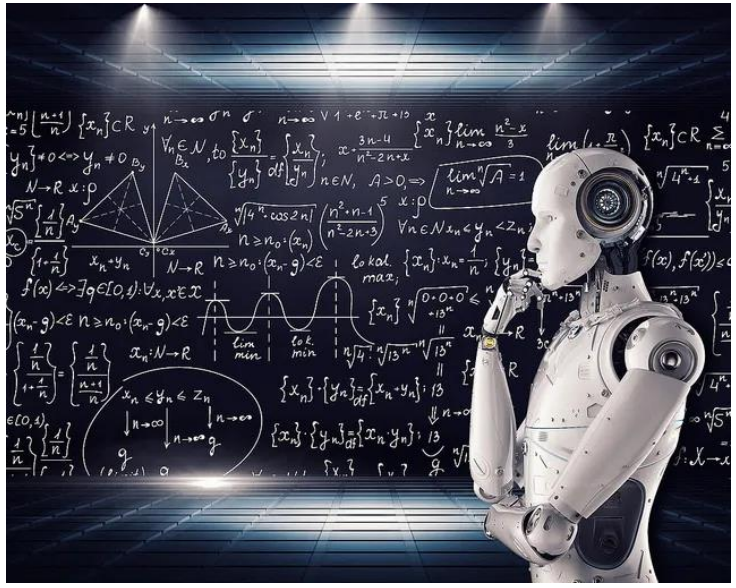
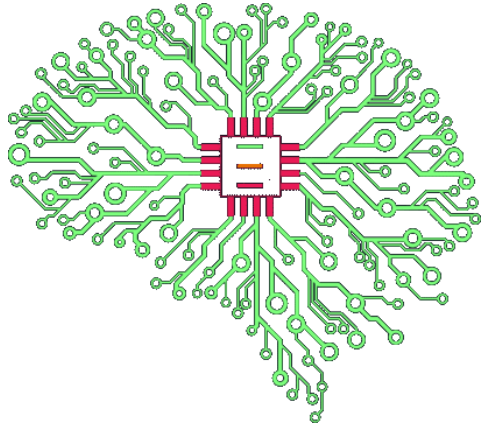
INTELLIGENCE ARTIFICIELLE
Réseaux de neurones







INTELLIGENCE ARTIFICIELLE
Réseaux de neurones

Compétences attendues :

- ✓ Analyser les principes d'intelligence artificielle. ⇆ *I*
- ✓ Choisir une démarche de résolution d'un problème d'ingénierie numérique ou d'intelligence artificielle. ⇆ ***I***
- ✓ Résoudre un problème en utilisant une solution d'intelligence artificielle. ⇆ ***I***

Introduction

Exemples d'applications des réseaux de neurones

- **Cartes de crédit :** détection des fraudes.
- **Finance :** analyse d'investissements et de fluctuations des taux de change.
- **Assurance :** couverture assurantielle et estimation des réserves.
- **Marketing :** ciblage des prospections, mesures et comparaisons des campagnes.
- **Archéologie :** identification et datation de fossiles et d'ossements.
- **Défense :** identification de cibles.
- **Production :** contrôles qualité.
- **Médecine :** diagnostics médicaux.
- **Energies :** estimations des réserves, prévisions de prix.
- **Pharmacie :** efficacité de nouveaux médicaments.
- **Psychologie :** prévisions comportementales.
- **Immobilier :** études de marchés.
- **Recherche scientifique :** identification de spécimens, séquençages de protéines.
- **Télécommunication :** détection des pannes de réseaux.
- **Transport :** maintenance des voies.
- **…**

Modèle de neurone

Définition d'un neurone (ou perceptron)

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

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

Modèle de neurone

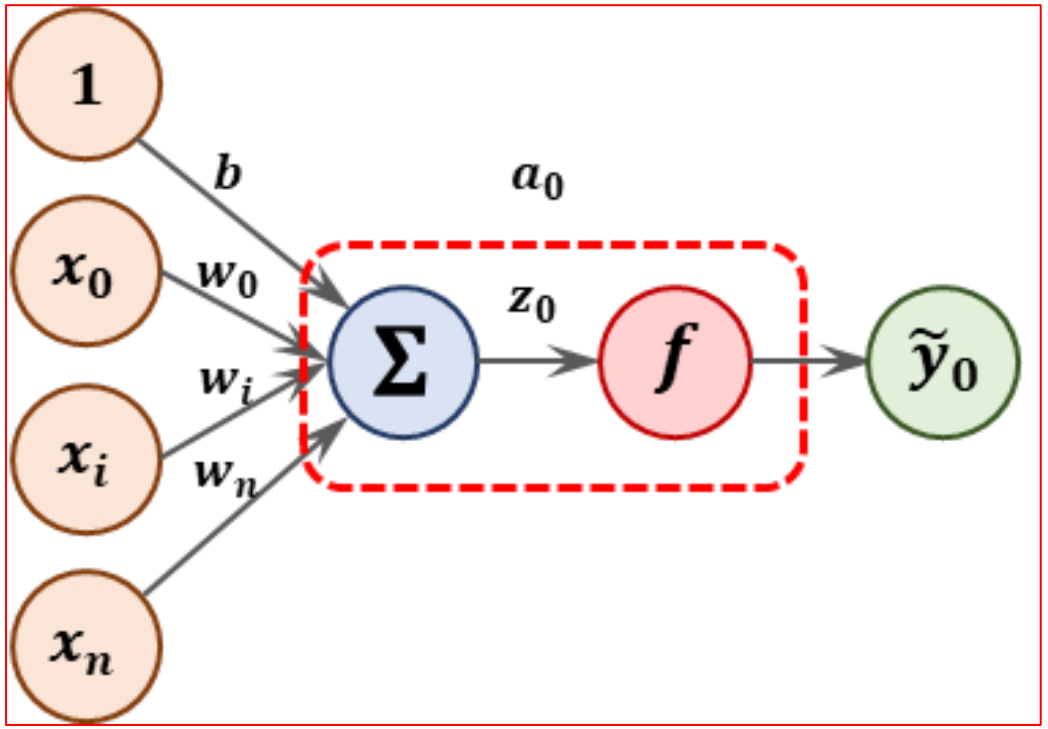
Définition d'un neurone (ou perceptron)

Sortie du neurone :

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

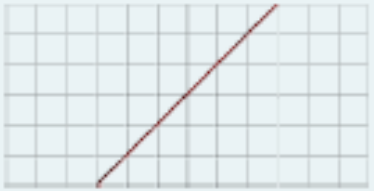
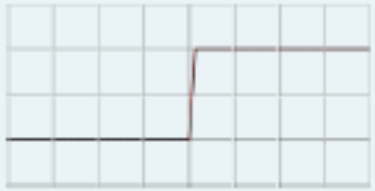
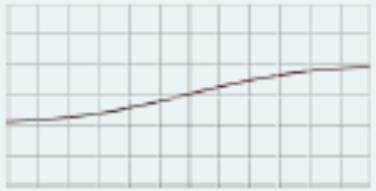
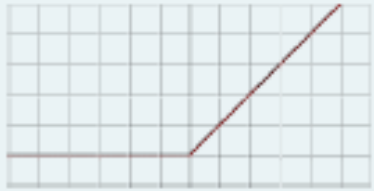
- **X** : vecteur d'entrée et x_i les données de la couche d'entrée.
- ω_i : poids (poids synaptiques).
- b : biais.
- z_0 : somme pondérée des entrées. $z_0 = b + \sum_{i=0}^{n} \omega_i x_i$
- f : fonction d'activation.
- $\tilde{y}_0$: valeur de sortie du neurone.

Remarque : Notation tilde ($\tilde{y}_0$) → valeur de sortie d'un neurone → valeur estimée



Modèle de neurone

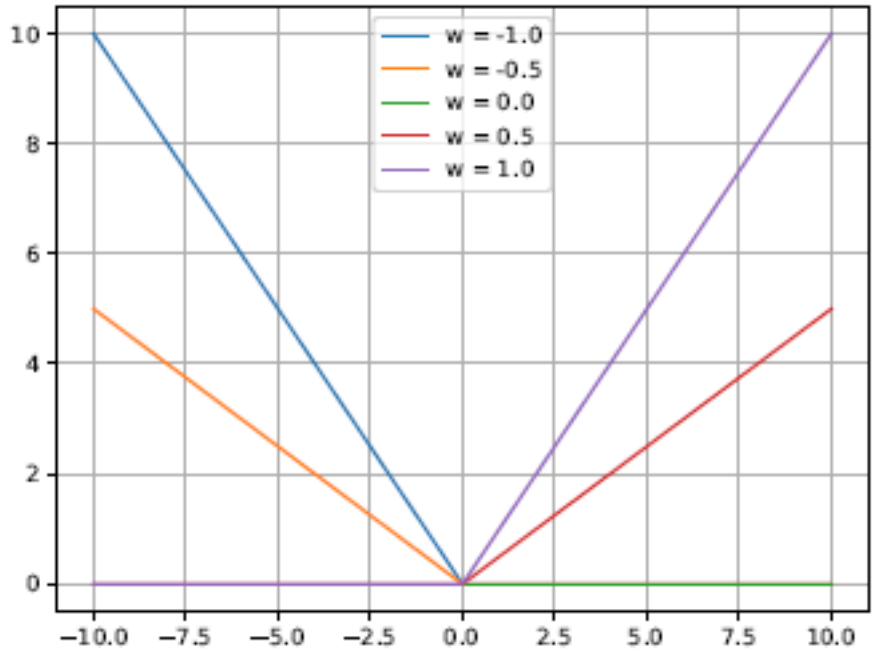
Définition d'une fonction d'activation :

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

Modèle de neurone

Définition d'une fonction d'activation :

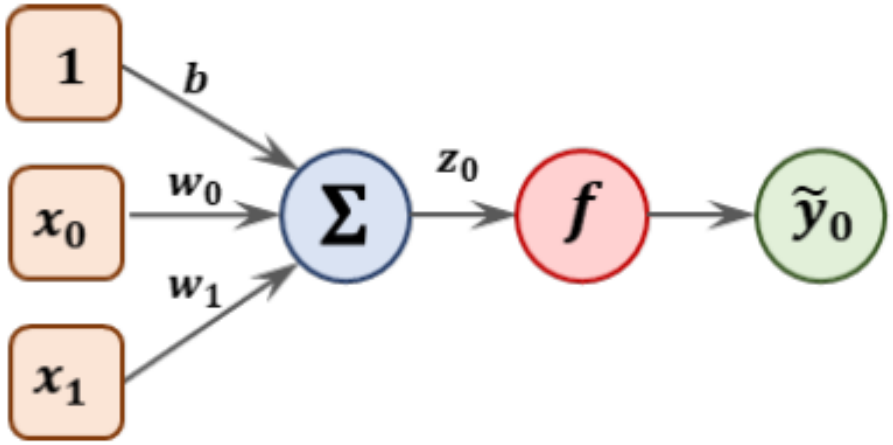
Remarque : Influence des poids et des biais sur la sortie du neurone en utilisant une fonction d'activation *ReLU*.



Modèle de neurone

Exemple de calcul avec un réseau de neurones

Initialisation les poids et le biais avec des valeurs aléatoires : $\omega_0 = -0{,}3 \quad \omega_1 = 0{,}8 \;\; et \; b = 0{,}2$.



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

Réseaux de neurones

Modélisation d'un réseau de neurones

Définition des 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 $i+1$.

Réseaux de neurones

Modélisation d'un réseau de neurones

Définition des couches

- **Couche d'entrée** → copie de l'ensemble des données d'entrées.

Nombre de neurones de cette couche correspond aux nombres de données d'entrées.

- **Couche cachée (ou couche intermédiaire)** → utilité intrinsèque au réseau de neurones.

Ajouter des neurones dans cette couche (ou ces couches) → ajouter de nouveaux paramètres.
Pour une couche → même fonction d'activation pour tous les neurones.
Fonction d'activation utilisée → peut être différente pour deux couches différentes.

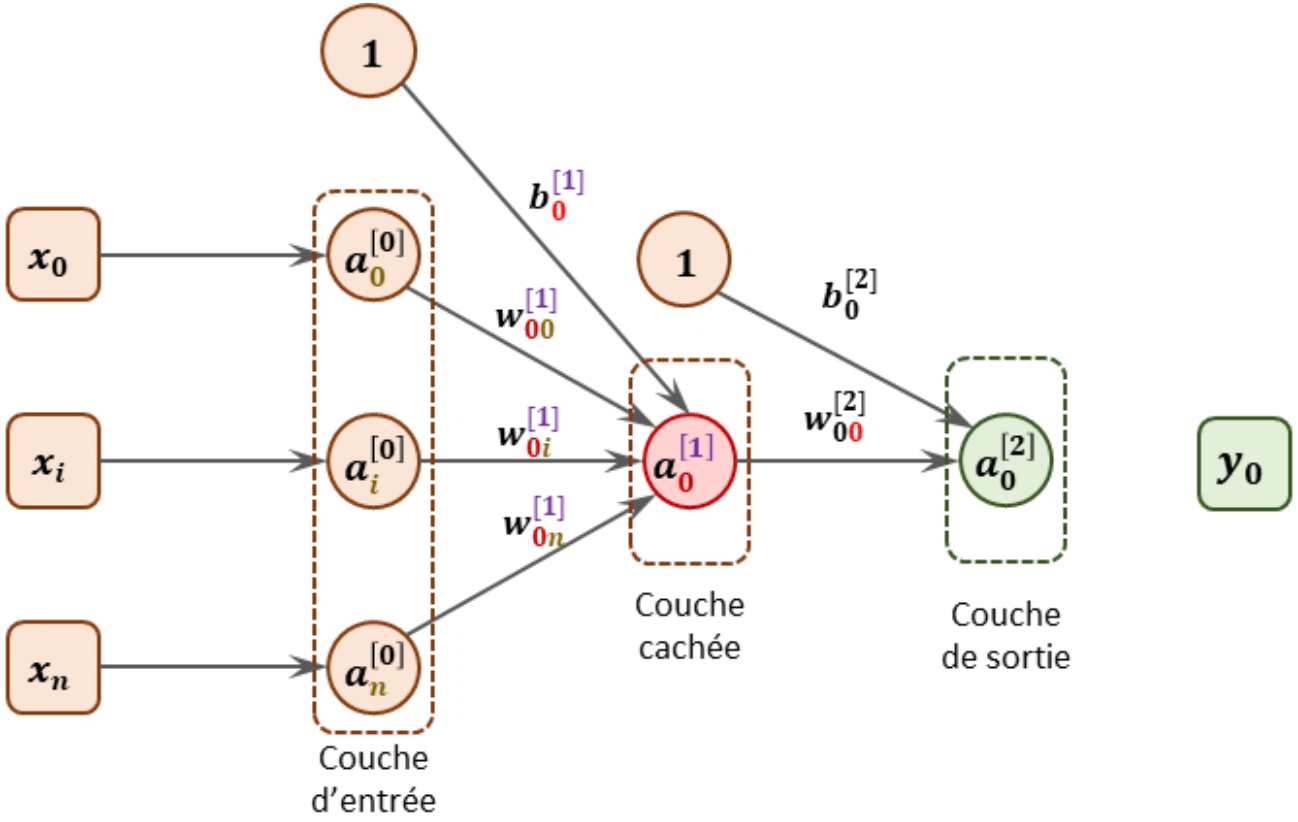
- **Couche de sortie :** Nombre de neurones → nombre de sorties attendues.

Fonction d'activation de la couche de sortie → souvent linéaire.

Réseaux de neurones

Modélisation d'un réseau de neurones

Définition des couches



Remarque : Chaque couche a sa propre matrice de poids, son propre vecteur de biais, un vecteur d'entrée et un vecteur de sortie.

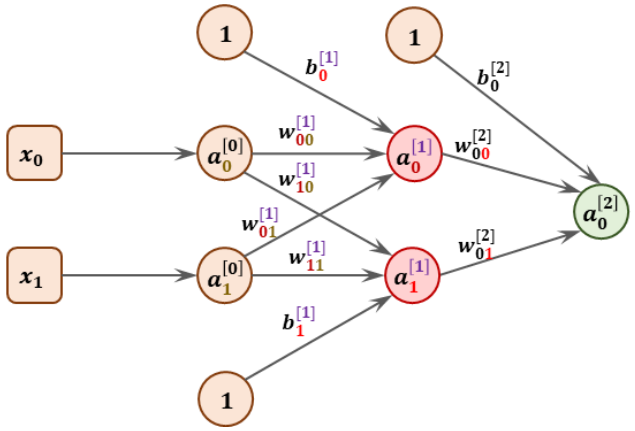
Réseaux de neurones

Modélisation d'un réseau de neurones

Définition de l'équation de propagation

Pour chacun des neurones $a_j^{[l]}$ → équation de propagation :

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

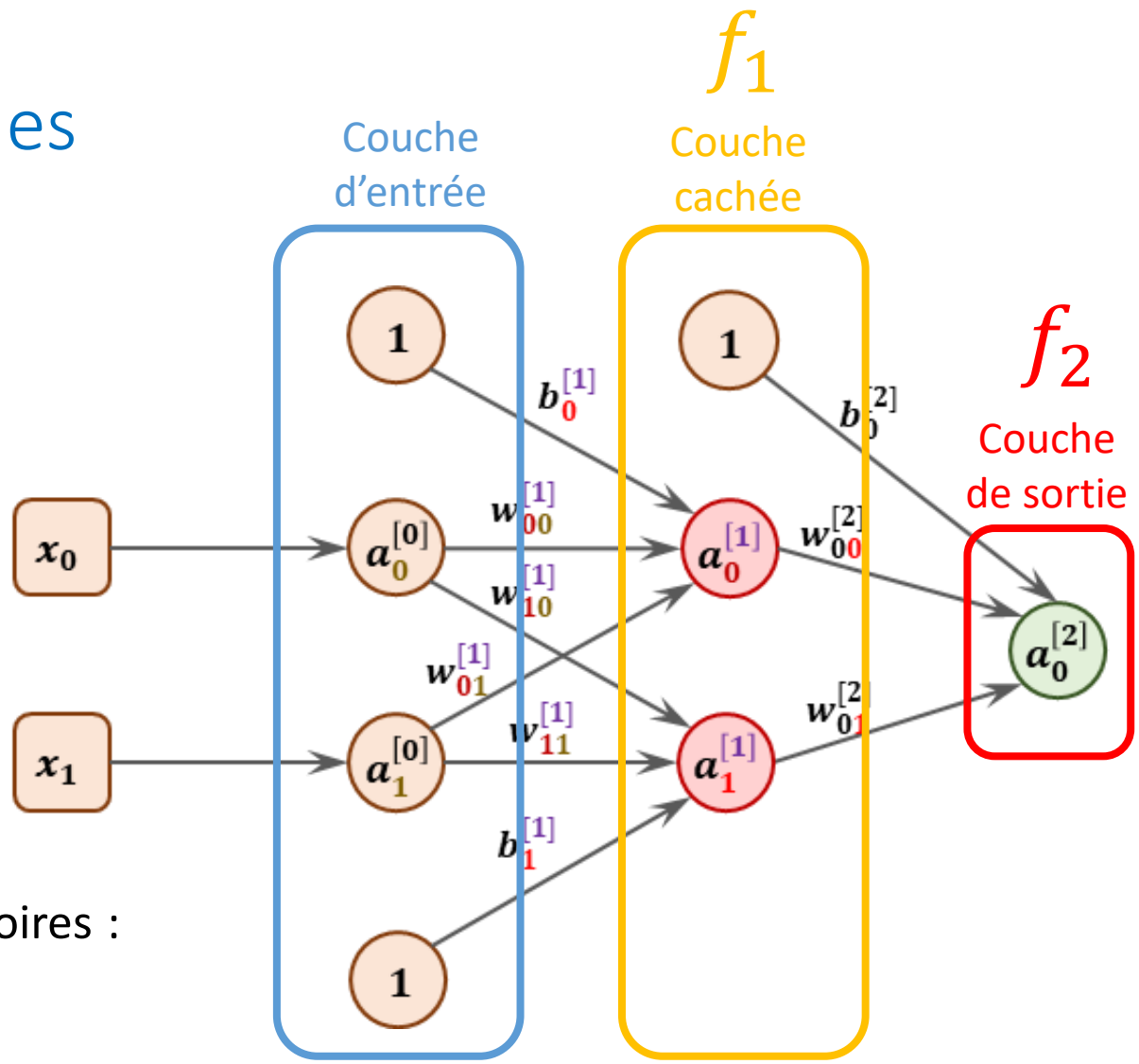


Réseaux de neurones

Modélisation d'un réseau de neurones

Exemple (réseau de neurone 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 :
$\omega_0 = -0{,}3 \quad \omega_1 = 0{,}8 \;\; et \; b = 0{,}2.$

Réseaux de neurones

Modélisation d'un réseau de neurones

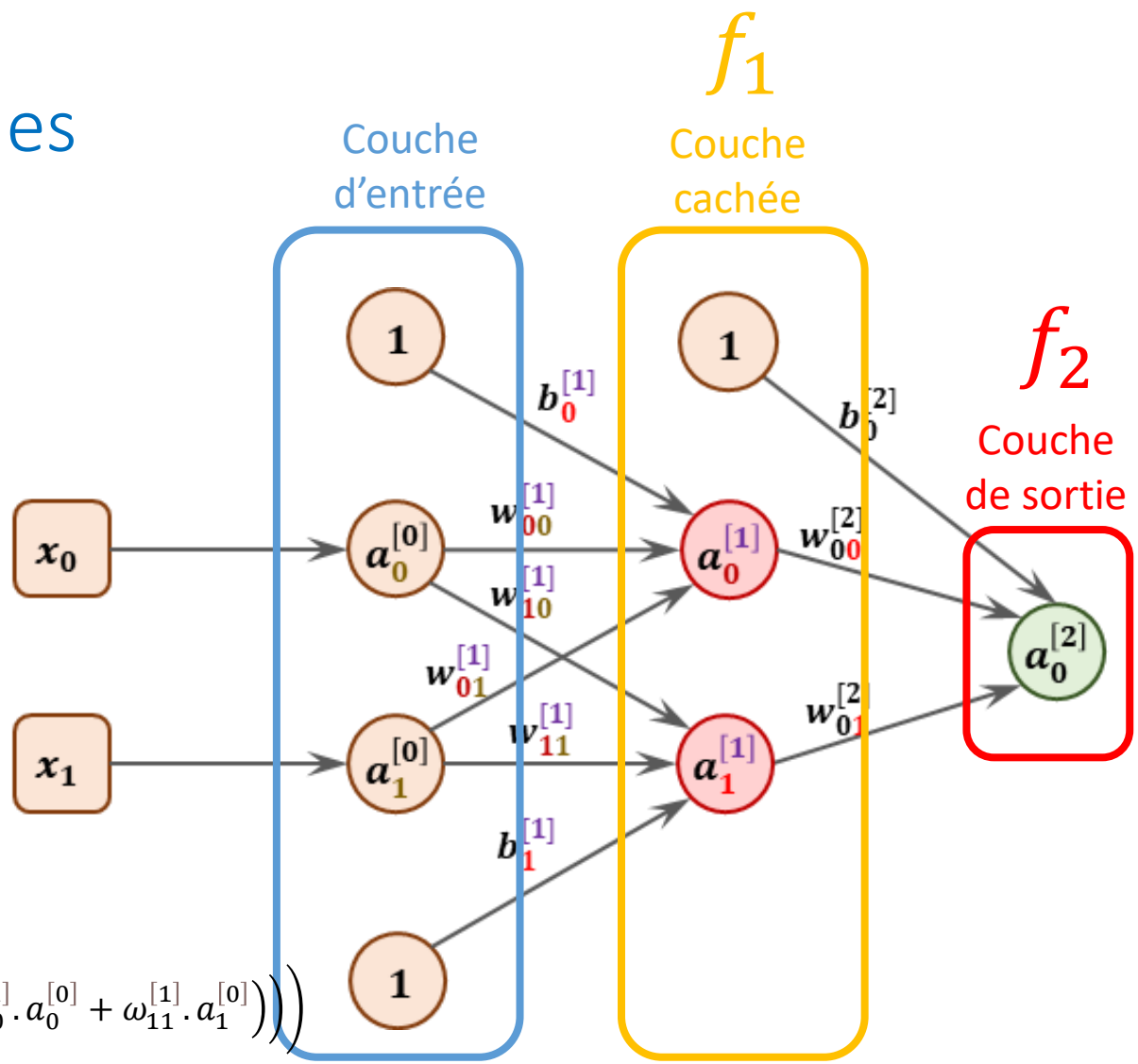
Exemple (réseau de neurone à 3 couches)

$$y_0 = a_0^{[2]}$$

$$y_0 = a_0^{[2]} = f_2\left(b_0^{[2]} + \omega_{00}^{[2]}.\boldsymbol{a_0^{[1]}} + \omega_{01}^{[2]}.\boldsymbol{a_1^{[1]}}\right)$$

$$a_0^{[1]} = f_1(b_0^{[1]} + \omega_{00}^{[1]}.a_0^{[0]} + \omega_{01}^{[1]}.a_1^{[0]})$$

$$a_1^{[1]} = f_1(b_1^{[1]} + \omega_{10}^{[1]}.a_0^{[0]} + \omega_{11}^{[1]}.a_1^{[0]})$$



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

Réseaux de neurones

Modélisation d'un réseau de neurones

Définition des paramètres

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

Réseaux de neurones

Modélisation d'un réseau de neurones

Méthode : Calcul du nombre de paramètres

n entrées *p* sorties l couches a_l nombre de neurones de la couche l.

Nombre de poids : $n_\omega = \sum_{i=1}^{l-1}(a_i . a_{i+1})$ **Nombre de bais :** $n_b = \sum_{i=2}^{l}(a_i)$

Nombre total de paramètre à calculer : $N = n_\omega + n_b$.

Objectif de la phase d'apprentissage : Déterminer les valeurs de l'ensemble des poids et des biais de telle sorte que l'écart entre les entrées et le résultat prédit soit minimale.

Réseaux de neurones

Etapes préliminaires à l'entrainement

- Sélection des données.
- Prétraitement des données.
- Choix du type de réseau et de son architecture.

Réseaux de neurones

Etapes préliminaires à l'entrainement

Sélection des données

Qualité du réseau ←→ Qualité des données utilisées pour l'entraîner.

Les données relatives à l'entrainement doivent couvrir l'ensemble de l'espace des entrées d'utilisation du réseau.

« *Avons-nous suffisamment de données ?* »

→ Quantité de données requises ←→ Complexité de la fonction que nous essayons d'approximer.

→ Processus d'entrainement du réseau neuronal → itératif (analyse après chaque entrainement des performances du réseau)

Réseaux de neurones

Etapes préliminaires à l'entrainement

Prétraitement des données

Prétraitement des données → faciliter l'entrainement au réseau.

Prétraitement des données → normalisation / transformations non linéaires / extraction de caractéristiques / traitement des données manquantes / …

Réseaux de neurones

Etapes préliminaires à l'entrainement

Choix du réseau

Type de base de l'architecture de réseau ←→ Type de problème que nous souhaitons résoudre.

Une fois que l'architecture de base est choisie → décider → nombre de neurones / nombre de couches / nombre de sorties / fonction de performance pour l'entrainement / …

Réseaux de neurones

Entrainement

Après la préparation des données et la mise en place de l'architecture du réseau sélectionnés, nous sommes prêts à entrainer le réseau.

Initialiser les poids et les biais (généralement fixés à de petites valeurs aléatoires, par exemple, répartis uniformément entre -0,5 et 0,5).

Entrainement des réseaux de neurones → Processus itératif

Même après convergence de l'algorithme → plusieurs cycles d'entrainement

Réseaux de neurones

Fonction de coût (cost function ou de perte, loss function)

Définition de la fonction de coût

Objectif : Minimiser l'écart entre la sortie du réseau de neurones et la valeur réelle de la sortie

n_b : nombre de données dans la base d'entraînement

Fonction de coût (moyenne des erreurs quadratique) :

$$C = \frac{1}{n_b} \sum_{i=1}^{n_b} (\tilde{Y}_i - Y_i)^2$$

Objectif : Déterminer les poids et les biais qui minimisent la fonction coût.

Réseaux de neurones

Fin d'apprentissage

Définition de l'epoch

Cycle d'apprentissage où tous les poids et tous les biais ont été mis à jour en faisant passer toutes les données du jeu d'entraînement dans les algorithmes de propagation et de rétropropagation.

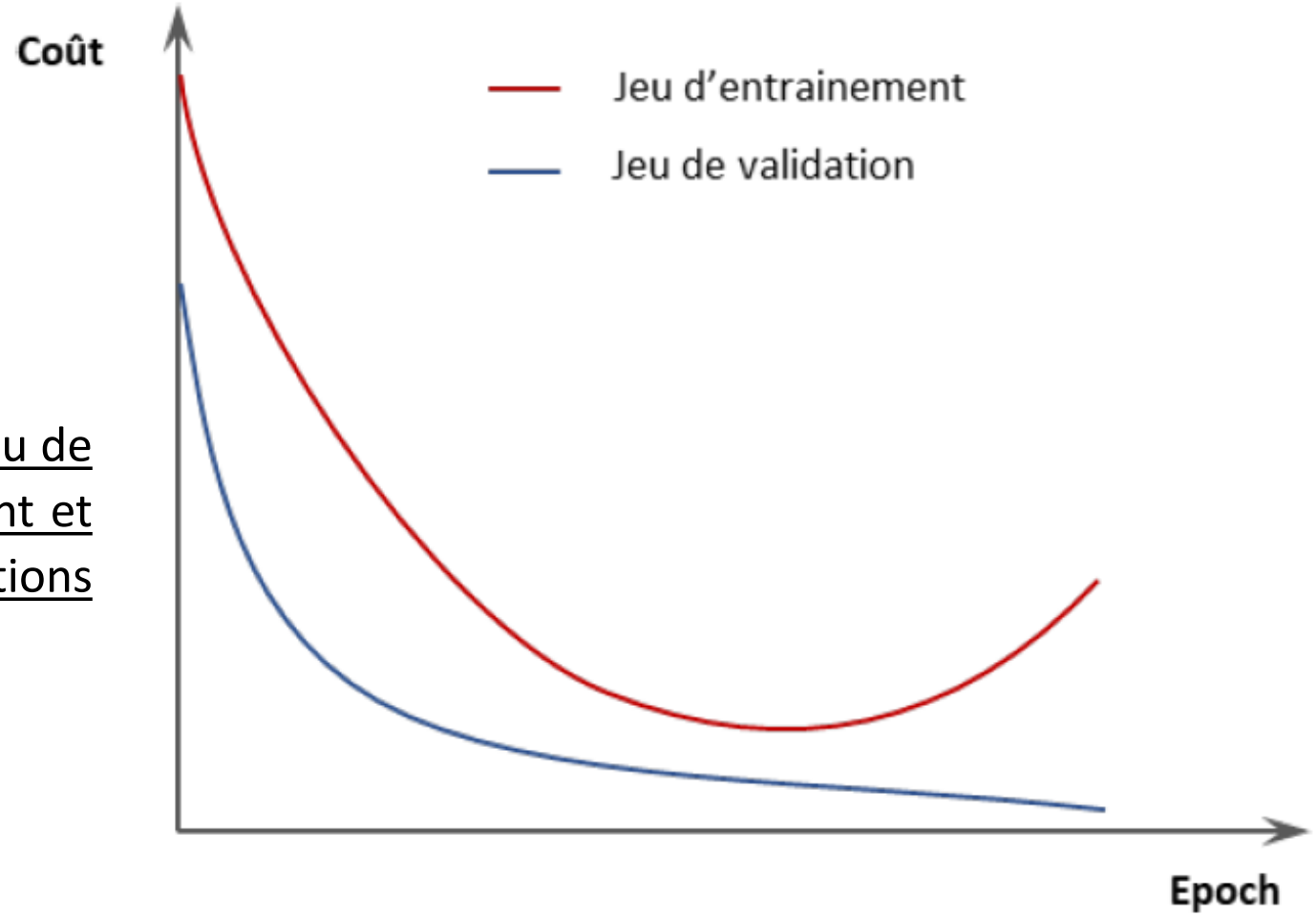
Réseaux de neurones

Fin d'apprentissage

Définition de l'epoch

Il existe en fait un stade à partir duquel le réseau de neurones se spécialise sur le jeu d'entraînement et devient donc incapable de réaliser des prédictions fiables sur un nouveau jeu de données.

On parle de surentraînement (ou *overfitting*).



Réseaux de neurones

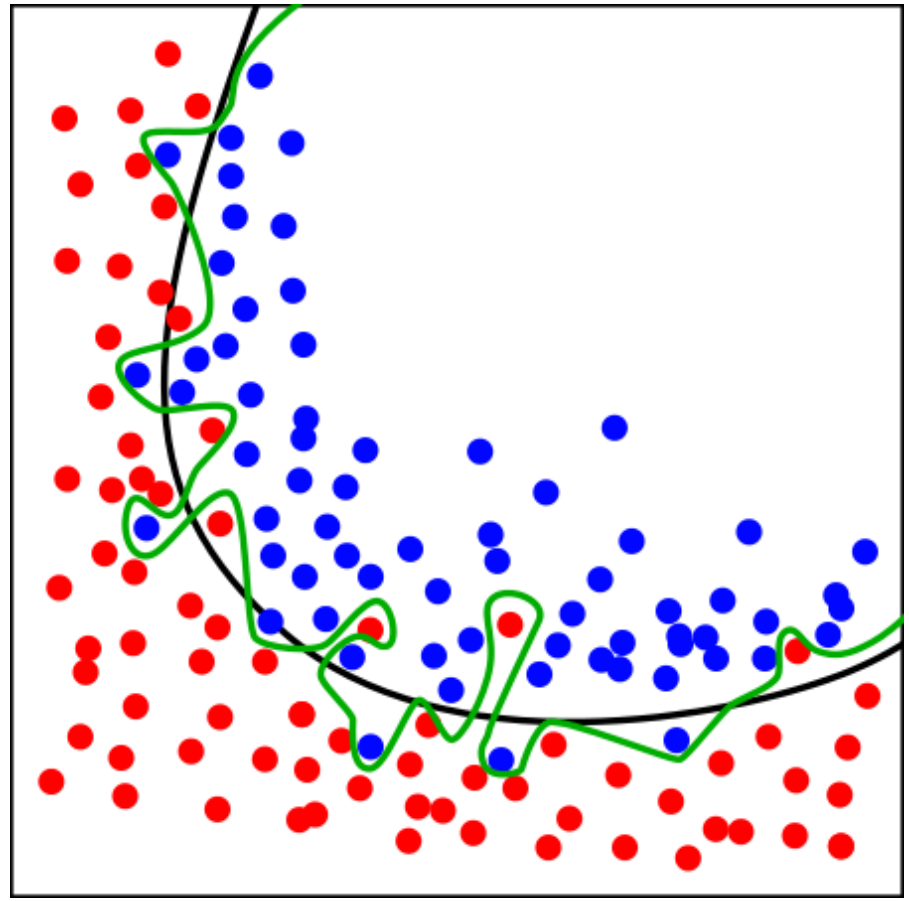
Fin d'apprentissage

Définition de l'epoch

Exemple :

Ligne verte → modèle sur-appris

Ligne noire → modèle régulier



Ligne verte classifie trop parfaitement les données d'entraînement, elle généralise mal et donnera de mauvaises prévisions futures avec de nouvelles données.

Modèle vert → moins bon → Modèle noir

Choix d'une solution pour un problème de prédiction

Entrées $x \in \mathbb{R}^d$

Sorties $y \in \mathbb{R}^p$

Base de données

d'entraînement de n exemples

	Modèle linéaire (régression ou classification)	k plus proches voisins	Réseaux de neurones (éventuellement profonds)
Temps de calcul à l'apprentissage	+ Inversion de matrice et calcul de $X^T X$ $\mathcal{O}(d^3 + nd^2)$ mais existence de méthodes numériques plus efficaces	++ Stockage des données de la base de donnée d'entraînement	- - Procédure longue et coûteuse notamment sur de grandes bases de données
Temps de calcul à la prédiction	++ 1 Produit matrice vecteur $\mathcal{O}(d \times p)$	- Tri d'un tableau de distances $\mathcal{O}(n \log(n))$	+ Séquence de produits matrice vecteur
Performances avec peu de données d'entrainement	++ Peu de paramètres à apprendre → bonnes performances avec peu de données	+ Performances correctes avec peu de voisins	+ Peu de paramètres par couche conduisent à un modèle proche du linéaire
Performances avec données abondantes	- Peu de paramètres apprenables	+ Réglage du nombre de voisins	++ Grande flexibilité de la fonction de prédiction
Performance avec données complexes (image / texte)	- Représentation des entrées inadaptable efficacement	- Difficile de choisir une distance correcte	++ Possibilité d'adapter les couches cachées aux propriétés des entrées