RNA

IRSI4 & ROC4

Types de Réseaux de Neurones Artificiels (RNA)

- Il existe plusieurs types de réseaux de neurones en fonction des tâches qu'ils accomplissent :

. Réseaux de Neurones Feedforward (FNN)

- Les plus simples des RNA.
- L'information circule dans un seul sens, de l'entrée à la sortie.
- Utilisé pour la classification et la régression.
- Exemple : **Perceptron Multi-Couches (MLP)**.

. Réseaux de Neurones Convolutifs (CNN)

- Conçu pour traiter les données sous forme de grilles, comme les images.
- Utilise des **couches de convolution** pour extraire des caractéristiques.
- Utilisé en vision par ordinateur (classification d'images, détection d'objets).
- Exemples : **VGGNet, ResNet, EfficientNet**.

. Réseaux de Neurones Récurrents (RNN)

- Adapté aux données séquentielles comme le texte ou les séries temporelles.
- Prend en compte les dépendances entre les éléments de la séquence.
- Variantes avancées : **LSTM (Long Short-Term Memory)**, **GRU (Gated Recurrent Unit)**.
- Applications : NLP, reconnaissance vocale.

Réseaux de Neurones à Transformer

- Remplacent de plus en plus les RNN en NLP.
- Utilisent le **mécanisme d'attention** (Self-Attention).
- Exemples : **BERT, GPT-4, T5**.
- Applications : Chatbots, traduction automatique, génération de texte.

Réseaux de Neurones Généraux (GANs)

- Utilisés pour générer des données réalistes.
- Composés d'un **générateur** et d'un **discriminateur**.
- Applications : génération d'images, deepfake, amélioration de résolution.

Autoencodeurs (AE)

- Utilisés pour la compression et la réduction de dimension.
- Applications : détection d’anomalies, réduction de bruit.

Quand utiliser quoi?

Problème	Type de RNA recommandé
Classification simple (emails, sentiments, images)	**FNN, CNN**
Vision par ordinateur (détection d'objets, segmentation)	**CNN, ViTs (Vision Transformers)**
Séries temporelles (prédictions boursières, météo)	**RNN, LSTM, GRU**
Traduction automatique, chatbots	**Transformers (GPT, BERT, T5)**
Génération d'images ou vidéos	**GANs, VAEs**
Détection d'anomalies (fraude, erreurs)	**Autoencodeurs, CNN, RNN**

Comment utiliser un RNA ?

- **Prétraitement des données** : nettoyage, normalisation, augmentation.
- **Définition de l'architecture** : choix du type de réseau et des hyperparamètres.
- **Choix des fonctions d'activation** (relu, sigmoid, softmax).
- **Optimisation avec un algorithme d'optimisation** (Adam, SGD).
- **Entraînement avec des données et réglage des hyperparamètres**.
- **Évaluation et amélioration** (validation croisée, tuning).

Les fonctions d’activations

- **Introduire la non-linéarité**
  - Permet au réseau d'apprendre des relations complexes entre les variables.
  - Sans fonction d'activation, le modèle ne peut pas capturer des structures complexes.
- **Contrôler la propagation du signal**
  - Certaines activations comme **ReLU** évitent l'explosion des gradients.
  - D'autres comme **sigmoïde** normalisent les sorties entre 0 et 1.
- **Faciliter l’apprentissage**
  - Réduire la saturation des gradients et accélérer la convergence.
  - Éviter le problème du **vanishing gradient** en profondeur.
- **Adapter le réseau à différentes tâches**
  - **Softmax** : Classifier plusieurs classes.
  - **ReLU** : Accélérer l'apprentissage en réseaux profonds.
  - **Sigmoïde/Tanh** : Traiter des sorties entre 0-1 ou -1 à 1.

1. Fonction Sigmoïde

Formule :

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

Pourquoi l'utiliser ?

- Produit une sortie entre **0 et 1**, utile pour interpréter les résultats comme des probabilités.
- Utilisée dans la **classification binaire**.

2. Fonction Tanh

Formule :

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

Pourquoi l'utiliser ?

- Similaire à la sigmoïde mais **centrée autour de 0** (-1 à 1), ce qui améliore la convergence.
- Évite le problème des petits gradients mieux que la sigmoïde.

3. Fonction ReLU (Rectified Linear Unit)

Formule :

$$ReLU(x) = \max(0, x)$$

Pourquoi l'utiliser ?

- Simple et efficace, elle accélère la convergence des réseaux profonds.
- Évite le problème de saturation des gradients rencontré avec Sigmoïde et Tanh.

4. Fonction Softmax

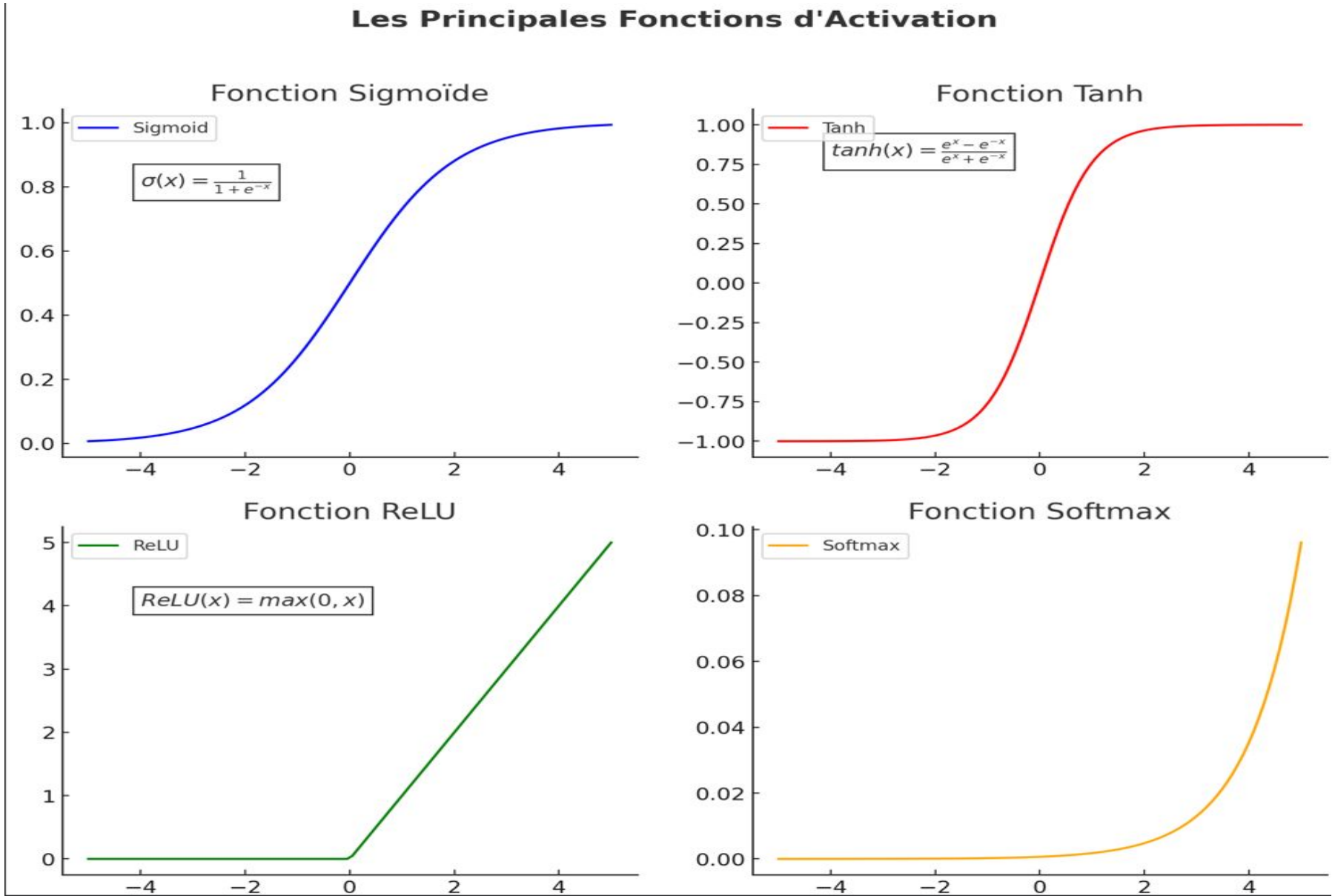
Formule :

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Pourquoi l'utiliser ?

- Convertit un vecteur de valeurs en **probabilités** (somme des sorties = 1).
- Permet de comparer directement les sorties comme des scores de classification.

Courbes mathématiques des FA



Résumé : Quand Utiliser Quelle Fonction ?

Fonction	Utilisation	Exemples
Sigmoïde	Classification binaire	Détection de spam, reconnaissance de chiffres (0/1)
Tanh	NLP, RNN	Traitement du langage naturel, analyse de sentiments
ReLU	Réseaux profonds (CNN, MLP)	Vision par ordinateur, classification d'images
Softmax	Classification multi-classes	Reconnaissance d'objets, NLP (analyse de texte)

Introduction aux Optimisateurs pour l'Apprentissage des Réseaux de Neurones

- Les **optimisateurs** sont des algorithmes qui ajustent les **poids** d'un réseau de neurones pour **minimiser l'erreur** et améliorer la convergence.
- Ils utilisent la **descente de gradient**, qui met à jour les poids en suivant la direction qui réduit la fonction de coût.
- Différentes variantes existent pour améliorer la **vitesse**, **stabilité**, et **efficacité** de l'apprentissage.

Pourquoi Utiliser un Optimiseur ?

✔ **Accélérer la convergence** → Trouver une solution optimale plus rapidement.
✔ **Améliorer la stabilité** → Empêcher les mises à jour brutales des poids.
✔ **Éviter les minima locaux** → Assurer une meilleure généralisation du modèle.
✔ **Adapter dynamiquement le taux d'apprentissage** → Optimisation plus fine selon le type de données.
✔ **Gérer de grands ensembles de données** → Optimisation efficace sans exploser la mémoire.

💡 **Choisir le bon optimiseur** est essentiel pour obtenir un **modèle performant et bien entraîné.** 🚀

Les principales méthodes d'optimisation

Optimiseur	Description	Avantages	Inconvénients	Utilisation recommandée
Gradient Descent (GD)	Met à jour les poids en calculant le gradient sur **l'ensemble des données.**	Stable, trouve un bon minimum global.	Très lent pour les **grandes bases de données.**	Petites bases de données, problèmes convexes.
Stochastic Gradient Descent (SGD)	Met à jour les poids après **chaque échantillon**, sans attendre l'ensemble des données.	Très rapide, efficace sur grands jeux de données.	Très bruité, peut converger vers un minimum local sous-optimal.	Réseaux profonds, grands ensembles de données.
Mini-batch SGD	Compromis entre GD et SGD : met à jour les poids après **un sous-ensemble (batch) de données.**	Plus rapide que GD, plus stable que SGD.	Nécessite un **choix optimal de la taille du batch.**	Apprentissage profond, NLP, vision par ordinateur.
Adam (Adaptive Moment Estimation)	Combine **SGD avec un taux d'apprentissage adaptatif**, basé sur l'historique des gradients.	Très efficace pour les **réseaux profonds** et les **CNN/RNN.**	Peut parfois sur-ajuster les poids, instable sur certains problèmes.	**CNN, RNN, NLP, classification d'images.**
RMSprop (Root Mean Square Propagation)	Ajuste dynamiquement le taux d'apprentissage selon la variance du gradient.	Très efficace pour **les séries temporelles et RNN.**	Nécessite un **réglage précis** de son paramètre de décroissance.	Séries temporelles, NLP, reconnaissance vocale.

Quel Optimiseur Choisir en Fonction du Contexte ?

Contexte	Optimiseur Recommandé
Petite base de données	GD (Gradient Descent)
Grand jeu de données	Mini-batch SGD
Réseaux convolutifs (CNN - Vision par ordinateur)	Adam
Réseaux récurrents (RNN, séries temporelles, NLP)	RMSprop, Adam
Données bruitées nécessitant de la stabilité	Mini-batch SGD
Optimisation rapide avec adaptabilité	Adam