Rapport de Projet

Sujet : Réseau de Neurones Artificiels pour la Classification Binaire

Auteur : Adja Fatou Sagna

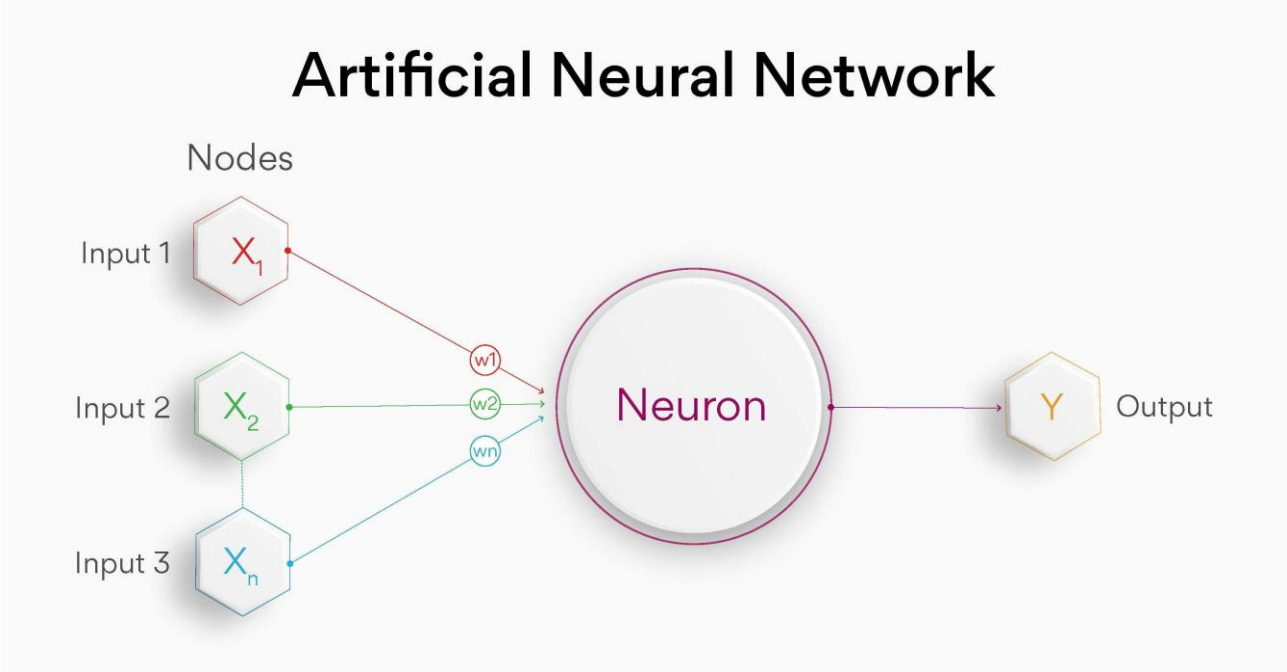
Établissement : IPSSI - Grande École d'Informatique

Cursus : Master en Intelligence Artificielle, Big Data et Développement

Date : 28 janvier 2026

1. Introduction

Dans le paradigme contemporain de l'Intelligence Artificielle, les réseaux de neurones artificiels (ANN) s'imposent comme des outils hégémoniques pour le traitement de relations non linéaires complexes. Ce projet s'inscrit dans une démarche analytique visant à concevoir, entraîner et évaluer un modèle de Deep Learning capable de résoudre une problématique de classification binaire. L'enjeu réside dans la capacité du modèle à établir une frontière de décision pertinente pour discriminer deux classes ("pos" et "neg") au sein d'un espace vectoriel de dimension 2.



2. Écosystème de Données

Le jeu de données, extrait du corpus *artificial_generator.xlsx*, comprend 2 000 observations. Chaque instance est définie par deux prédicteurs numériques (X_1, X_2) et une variable cible catégorielle (Y).

- **Structure :** 2 000 lignes, 2 variables explicatives, 1 variable cible.
- **Nature :** Données synthétiques optimisées pour l'étude de la convergence des algorithmes de rétro propagation et la visualisation des frontières de décision.

3. Méthodologie et Architecture Système

L'implémentation a été réalisée sous l'environnement **TensorFlow 2.20.0**.

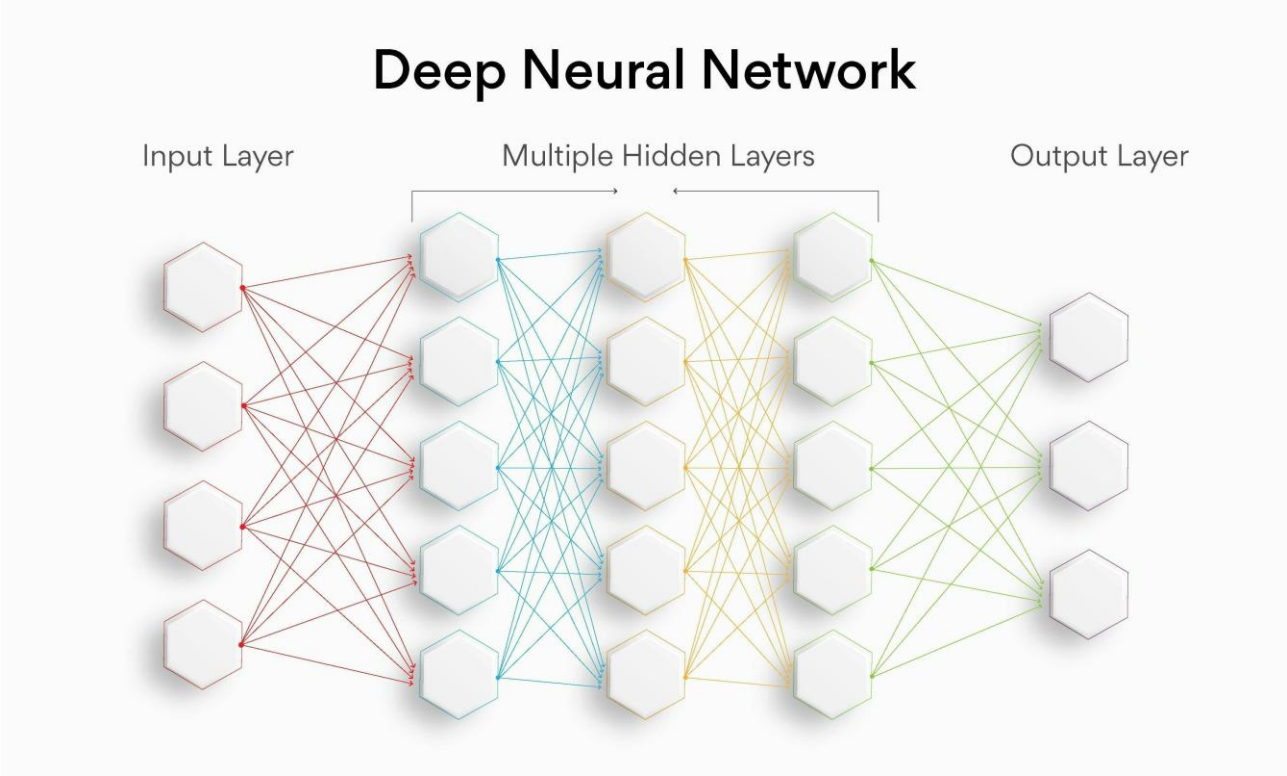
3.1. Prétraitement et Ingénierie des Données

- **Chargement :** Utilisation de la bibliothèque *Pandas* pour la structuration des données.
- **Encodage :** Transformation des étiquettes textuelles en valeurs numériques via *LabelEncoder* de *Scikit-Learn*.
- **Partitionnement :** Division du dataset en sous-ensembles d'entraînement et de test (*train_test_split*) pour garantir l'impartialité de l'évaluation finale.

3.2. Architecture du Modèle

Le réseau repose sur une structure séquentielle (*Keras Sequential API*) intégrant :

- Une couche d'entrée adaptée aux deux caractéristiques (X_1, X_2).
- Des couches denses (Hidden Layers) utilisant des fonctions d'activation non linéaires (notamment **ReLU** pour la parcimonie des activations).
- Une couche de sortie dotée d'une fonction **Sigmoïde**, standard pour la classification binaire.



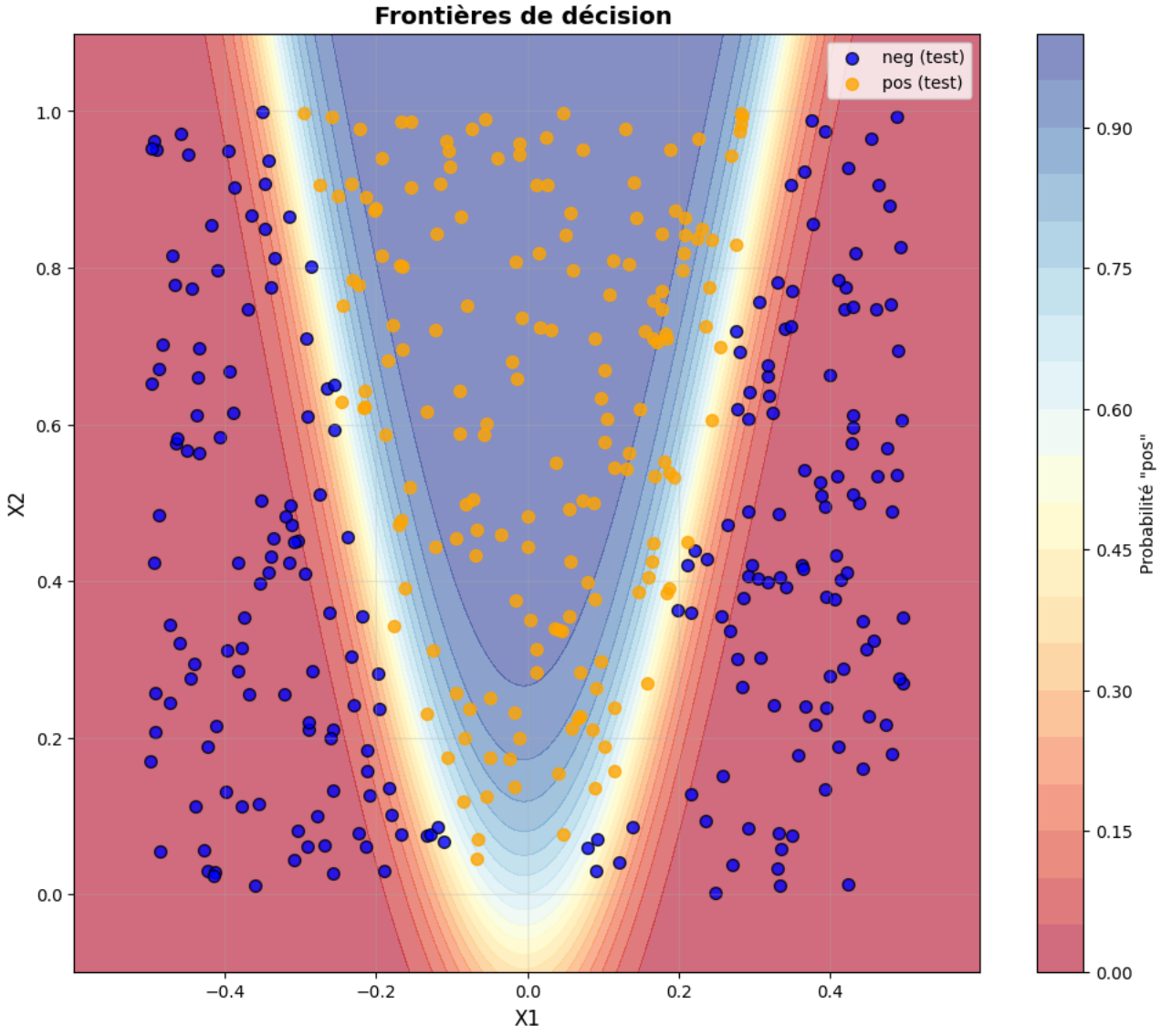
3.3. Stratégie d'Optimisation

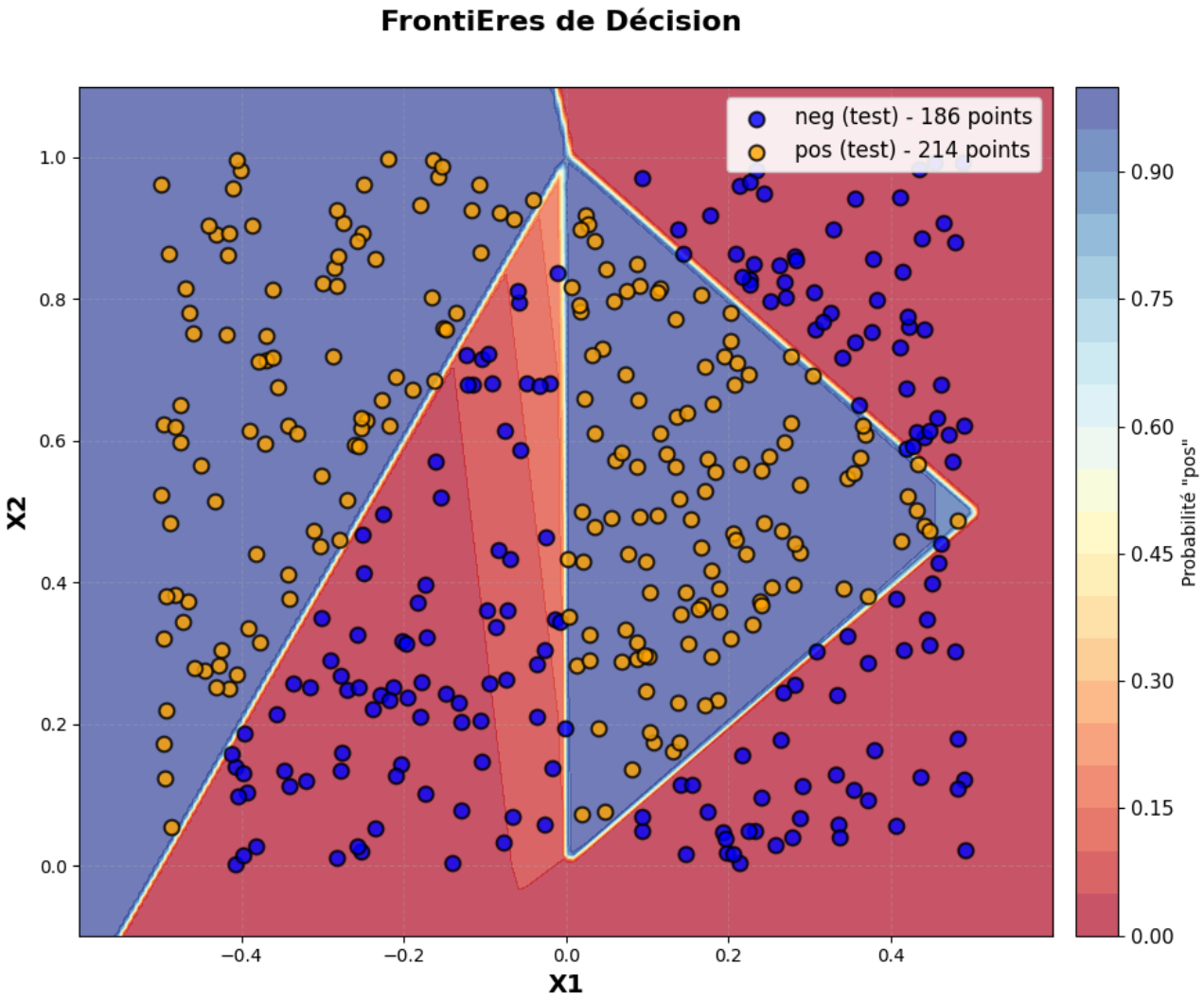
La compilation du modèle a mobilisé la fonction de perte *binary_crossentropy*. L'ajustement des poids synaptiques a été confié à un optimiseur de pointe (type Adam), visant la minimisation du gradient sur 250 époques d'apprentissage (data4).

4. Analyse des Résultats et Performances

L'évaluation rigoureuse du modèle révèle une convergence remarquable.

- **Précision (Accuracy) :** Le modèle atteint des scores d'une grande finesse, oscillant autour de **98% à 99%** sur les données de validation lors des dernières itérations.
- **Convergence :** La fonction de perte (Loss) montre une décroissance asymptotique stable, témoignant d'une absence d'overfitting (surapprentissage) majeur.
- **Visualisation :** Grâce à mlxtend, les frontières de décision ont été projetée, confirmant que les réseaux ont parfaitement capturé les topologies des données, même dans les zones de forte densité de classes imbriquées pour second dataset.





5. Synthèse et Perspectives

Ce projet démontre l'efficience des réseaux de neurones pour la séparation de données non linéairement séparables. La méthodologie employée, de l'ingénierie des données à la visualisation des régions de décision, constitue un socle robuste pour des applications industrielles plus vastes (score de crédit, diagnostic médical, etc.).

Pour transcender ces résultats, nous envisageons :

1. **L'optimisation des hyperparamètres** (Grid Search) pour affiner la structure du réseau.
2. **L'intégration de la régularisation Dropout** pour accroître la résilience du modèle face au bruit.
3. **L'analyse comparative** avec des algorithmes de type XGBoost pour évaluer le rapport performance/coût computationnel.

6. Références Techniques

- **Calcul Numérique :** NumPy & Pandas.
- **Moteur d'Apprentissage :** TensorFlow 2.20.0 & Keras.
- **Métriques & Visualisation :** Scikit-learn, Matplotlib & Mlxtend