Rapport : Implémentation et Tests des Modèles Machine Learning : Etape 2

Timothée M'bassidje, Ivo Talvet

October 27, 2025

1 Introduction

Ce rapport présente l'implémentation et les tests de modèles de Machine Learning dans la librairie `MachineLib`. L'objectif est de préparer une structure exploitable ultérieurement dans Unity.

2 Langage utilisé

La librairie a été implémentée en C pour des performances optimales et une intégration facile avec d'autres systèmes.

3 Modèles Implémentés

- **Linear model** : modèle linéaire $y = w \cdot x + b$ pour classification binaire.
- **Perceptron Rosenblatt** : classification binaire avec mise à jour itérative des poids. Régression linéaire : modèle pour prédire des valeurs continues, non testé ici.

Les fonctions de prédiction n'ont pas été implémentées dans la librairie C.

4 Structure de la Librairie

4.1 Fichiers principaux

- `Machinelib.h` : définition de la structure `Model` et des prototypes.
- `Machinelib.c` : implémentation de l'initialisation et de l'entraînement pour le linear model et le perceptron.
- `train.c` : programme principal pour lire les fichiers binaires, entraîner le modèle et générer un fichier de poids.

4.2 Format des fichiers binaires

1. `model_type` (int32) : 0 = linear, 2 = perceptron.
2. `n_samples` (int32)

3. `input_size` (int32)
4. Données X (float64)
5. Labels Y (int32)

Le fichier de sortie contient les poids (`w`) et le biais (`b`).

5 Compilation et Exécution

5.1 Compilation

Sous Windows avec LLVM MinGW :

```
gcc -o train.exe train.c Machinelib.c
```

5.2 Exécution des tests

```
.\train.exe MachineLib/CasTestBinary/[NOM TEST].bin PoidBinary/[NOM TEST]_w.bins
```

6 Procédure de Test Python

Les tests ont été réalisés dans un Jupyter Notebook :

1. Création de datasets synthétiques.
2. Conversion en float64 pour X et int32 pour Y.
3. Écriture des fichiers binaires dans `MachineLib/CasTestBinary`.
4. Appel du programme C (`train.exe`) pour entraîner les modèles.
5. Lecture des fichiers de poids dans Python pour visualiser les plans de décision.

6.1 Cas de tests

- **Linear simple et multiple** : séparation linéaire des points, fonctionne avec linear model et perceptron.
- **Perceptron XOR** : sans couche cachée, pas de prédiction correcte.
- **X3 multi-classe** : réalisé en Python avec trois modèles linéaires (un contre tous) pour visualiser trois classes.

6.2 Visualisation

Les poids et biais générés par le C sont utilisés pour tracer les plans de décision avec `matplotlib`, chaque classe étant représentée par une couleur distincte et une zone semi-transparente.

7 Limitations et Perspectives

- Les tests concernent uniquement la classification linéaire et le perceptron Rosenblatt.
- Les fonctions de prédiction ne sont pas implémentées dans C.
- La régression linéaire n'a pas été testée.
- Ces tests valident uniquement les fonctions d'entraînement de `Machinelib.c`.
- L'objectif futur est d'intégrer ces modèles dans Unity pour des simulations et environnements complexes.

8 Conclusion

Cette première phase a permis d'implémenter les modèles Rosenblatt de base, de tester leur entraînement via des fichiers binaires et de visualiser les résultats dans Python. Le système est prêt pour intégrer d'autres modèles et une interopérabilité avec Unity.