Marc BOURGEOIS - Matthieu LINEL - Mathieu CHATEIGNER
5A RVJV ESGI
07/01/2024

Machine Learning

Introduction

Outils :

- Unity 2021.3.10f1 (pour le client)
- Visual Studio (scripts C# et DLL C++)
- Git (https://github.com/marc-brgs/Machine-Learning)

Emplacement des fichiers sources de la DLL C++ dans le projet Unity :
./Assets/MLLibrary/

Seulement le perceptron multi couche a été implémenté pour le moment.

Cas de tests implémentés :

- **AND gate** (console Unity)
- **XOR gate** (console Unity)
- **Cross** (visualisable dans la scène Unity)

Les autres cas de test n'ont pas encore été implémentés par manque de temps mais le seront dans le rendu final.

Script C# permettant de changer le cas de test :
./Assets/TestML.cs (dans la méthode Start)

Perceptron multi couche (MLP)

Les méthodes de notre MLP sont passés de la DLL à Unity et sont utilisées par une classe PerceptronWrapper côté client.

Méthodes implémentées :

- Initialisation (biais et poids) (OK)
- Predict (feed forward) (OK)
- Train (back propagation) (OK)

Les méthodes pour sérialiser notre MLP seront implémentées dans le rendu final.

AND (linéairement séparable)

Couches : (2, 1, 1)
Nombre de cas différents : 4
Nombre d'entrainement : 100000 * 4

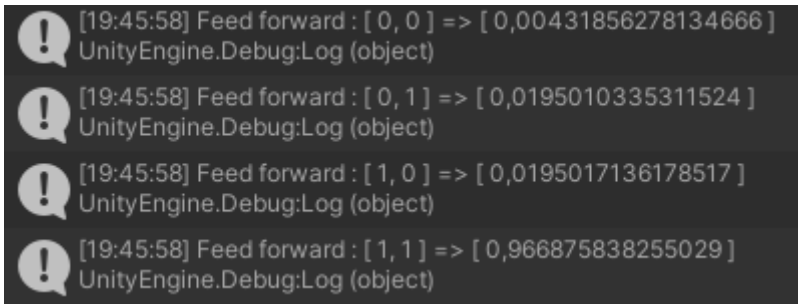

Figure 1 : Résultat obtenu pour le test AND dans Unity en utilisant notre DLL MLP

Ce test est concluant car le résultat correspond à nos attentes.

XOR (non linéairement séparable)

Couches : (2, 2, 1)
Nombre de cas différents : 4
Nombre d'entrainement : 100000 * 4

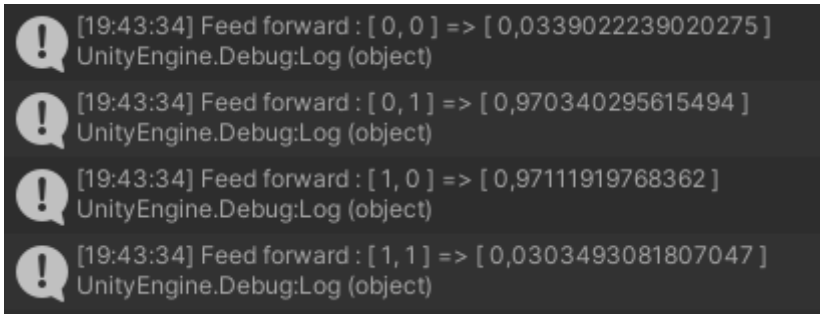

Figure 2 : Résultat obtenu pour le test XOR dans Unity en utilisant notre DLL MLP

Ce test est concluant car le résultat correspond à nos attentes.

Cross (non linéairement séparable)

Couches : (2, 4, 1)
Nombre de points : 500
Nombre d'entrainement : 3000 * 500

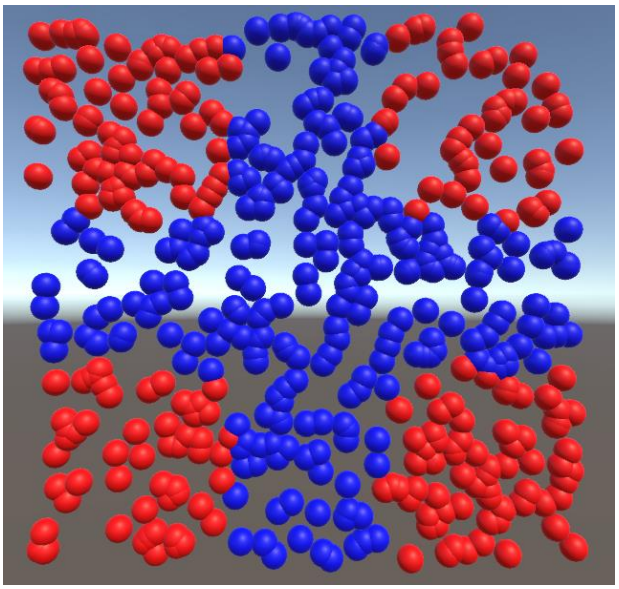

Figure 3 : Résultat obtenu pour le test Cross dans Unity en utilisant notre DLL MLP

Ce test est concluant car le résultat correspond à nos attentes.