INSAT	Module : Deep Learning
Département Mathématiques et Informatique	Sections : GL4, RT4
Année Universitaire : 2023 - 2024	Enseignante : Sana Hamdi

TP N°2 : Classification avec des réseaux de neurones entièrement connectés

Objectif :

Dans ce TP, nous allons résoudre quelques problèmes de classification avec des réseaux de neurones entièrement connectés (Fully Connected Neural Networks, FCNN).

I. Classification binaire :

1. Dataset :

Nous allons créer un dataset pour simuler un problème de classification binaire. La bibliothèque Scikit-Learn nous permet de créer des données binaire à l'aide de la méthode make circles().

a. Créer des données avec make circles(). Nous fixons les paramètres comme suit :
   • **Nombre d'échantillons dans le dataset :** 1000
   • **Bruit** : 0.03 (Ecart type du bruit gaussien ajouté aux données)
b. Afficher les dimensions du dataset. Quelles sont les dimensions de l'entrée et de la sortie du classifieur?
c. Visualiser les données.
d. Ce dataset nécessite une fonction linéaire ou non linéaire pour séparer ses classes ? Justifier.
e. Diviser les données en ensembles d'apprentissage et de test à l'aide de **train test split().** Fixer la valeur test size a 0.2, c'est à dire 80% données d'entrainement, et 20% données de test et comme la répartition se produit de manière aléatoire entre les données, fixer la valeur de random state a 42 afin que la répartition soit reproductible.

2. Définition du modèle

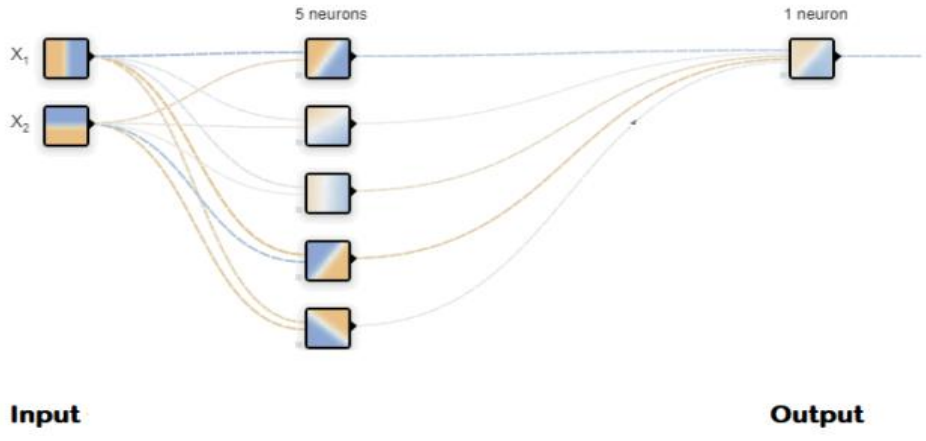
2.1 Cycle de vie d'un modèle :

Un modèle a un cycle de vie. Les cinq étapes du cycle de vie sont les suivantes :
1. Définir le modèle
2. Compiler le modèle

3. Entrainer le modèle
4. Évaluer le modèle
5. Faire des prédictions

2.2 Définition du modèle FCNN :

L'architecture du modèle de classification est illustrée dans la figure suivante :



a. Créer un nouveau modèle model = Sequential() et le compléter en lui ajoutant :
   - une couche entièrement connectée avec 100 neurones cachés,
   - une activation sigmoide,

```
model = ... # À compléter
model.add(...) # À compléter
```

b. Afficher un résumé concis de l'architecture du modèle, y compris le nombre de paramètres et la forme de la sortie de chaque couche.

2.3 Compilation du modèle :

Compiler le modèle avec la fonction de perte (entropie croisée binaire) et l'optimiseur (SGD - Stochastic Gradient Descent)

2.4 Entrainement du modèle :

Entrainer le réseau sur 100 époques et afficher la progression du modèle toutes les 10 époques (Loss, Accuracy, Test loss et Test Accuracy).

2.5 Evaluation du modèle

1. Tracer la (les) frontière(s) de décision pour les ensembles d'entrainement et de test.
2. Quel est le problème de ce modèle ?
3. Comment améliorer les performances du modèle ? Proposer 3 techniques d'amélioration et implémenter une (ou la combinaison) des solutions proposées.

II. La multi-Classification :

La bibliothèque Scikit-Learn nous permet de créer des données multi-classes à l'aide de la méthode make-blobs(). Cette méthode créera le nombre de classes que nous voulons.

1. Créer des données multi-classes avec make-blobs(). Nous fixons les paramètres comme suit :
   • Nombre de classes : 4
   • Nombre de features : 2
   • Nombre d'échantillons dans le dataset : 1000
2. Diviser les données en ensembles d'apprentissage et de test
3. Visualiser les données.
4. Ce dataset nécessite une fonction linéaire ou non linéaire pour séparer ses classes ? Justifier.
5. Proposer une architecture basée sur FCNN pour la classification de ces données et la développer.
6. Entrainer le modèle et afficher les mesures de progression toutes les 10 époques, y compris la perte, la précision, la perte de test et la précision de test.
7. Afficher toutes les métriques de classification vues en cours (précision, rappel, score F1, matrice de confusion, ...) et expliquer l'utilité de chacune d'elles en interprétant les résultats obtenus.
8. Tracer la (les) frontières(s) de décision pour les ensembles d'entrainement et de test.
9. Etudier l'impact de l'augmentation du nombre d'échantillons dans le dataset sur l'exactitude de classification, et afficher la courbe de la progression de l'exactitude de classification en fonction du nombre d'échantillons dans le dataset.