HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Laboratoire 3 : Learning with Artificial Neural Networks

Départements : TIC

Unité d'enseignement ARN

Auteurs : **Emily Baquerizo**
Rafael Dousse

Professeur : **Andres Perez - Uribe**
Assistants : **Simon Walther**
Shabnam Ataee

Classe : **ARN - B**
Salle de labo : **C41**

Date : **21 avril 2024**

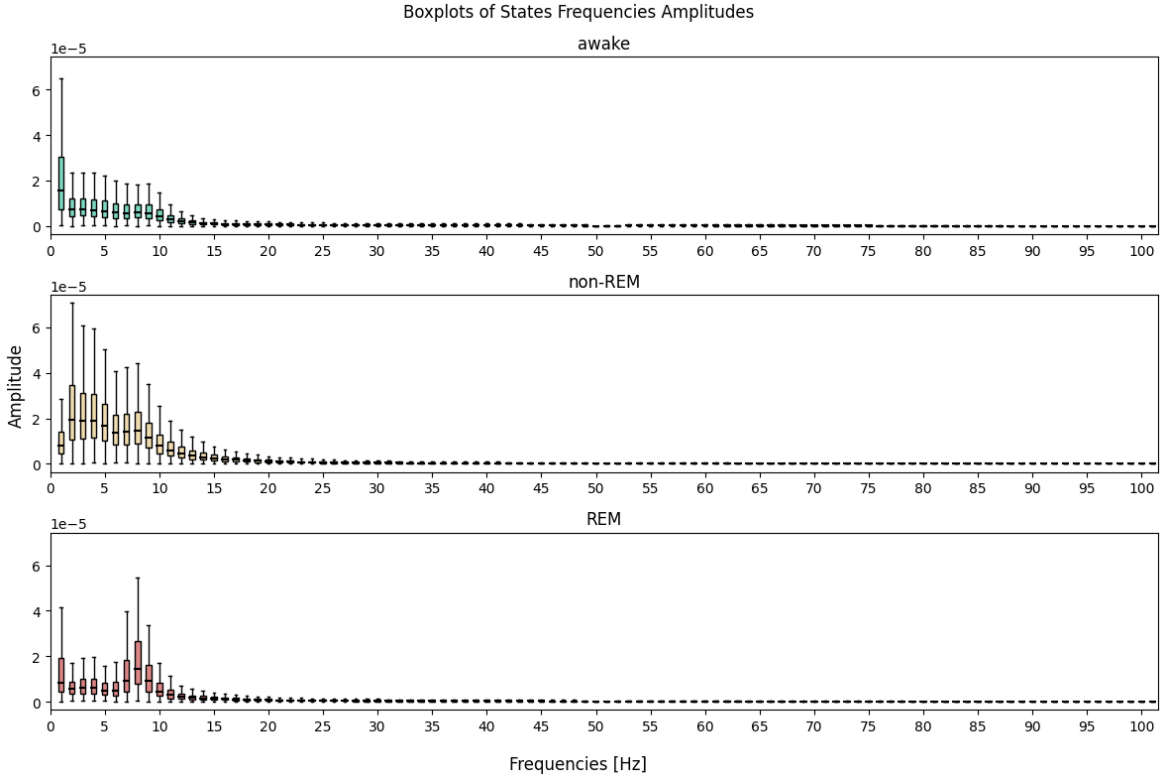
1 Introduction

Ce travail pratique a pour but de mettre en pratique les connaissances acquises durant nos précédents laboratoires afin d'implémenter des MLPs (Multi-Layer Perceptrons) afin de résoudre des problèmes de classification. Dans notre cas, nous utilisons la base de données EEG_mouse_data qui contient des données d'EEG de souris, c'est-à-dire des données prises sur le cerveau de souris qui représentent le cycle de sommeil de ces dernières.

Pour pouvoir réaliser ce travail, nous avons utilisé la librairie keras qui est une librairie open-source qui permet de créer des réseaux de neurones de manière simple et rapide. Finalement, nous vérifions les données de notre modèle en faisant une validation croisée et en affichant les matrices de confusions et les F1-scores.

2 Exercice 1 : Classification de 2 classes

Pour ce travail nous avons eu accès a une base de données EEG_mouse_data qui contient des données d'EEG de souris. Cette base contient plus d'une centaine de colonnes sur l'amplitude des fréquences de l'EEG de souris. Pour avoir des résultats plus pertinents et précis, nous avons décidé de ne prendre que les colonnes dont l'amplitude des fréquences sont les plus significatives et pour faire cela nous avons pu constater, grâce à un tableau de boxplot pris sur le laboratoire 1, que les colonnes les plus utiles se trouvent être dans les 20 premières.



2.1 Préparation des données

Pour la première partie de ce travail, nous avons commencé par importer les données de la base de données EEG_mouse_data et nous avons effectué un prétraitement des données en utilisant la colonne des états et en la transformant en deux classes. Nous avons regroupé les états n-rem et rem ensemble en un état asleep à qui nous avons donné la valeur 1 et l'état awake à qui nous avons attribué la valeur -1. Ensuite, nous avons normalisé les données grâce à la fonction StandardScaler de sklearn.

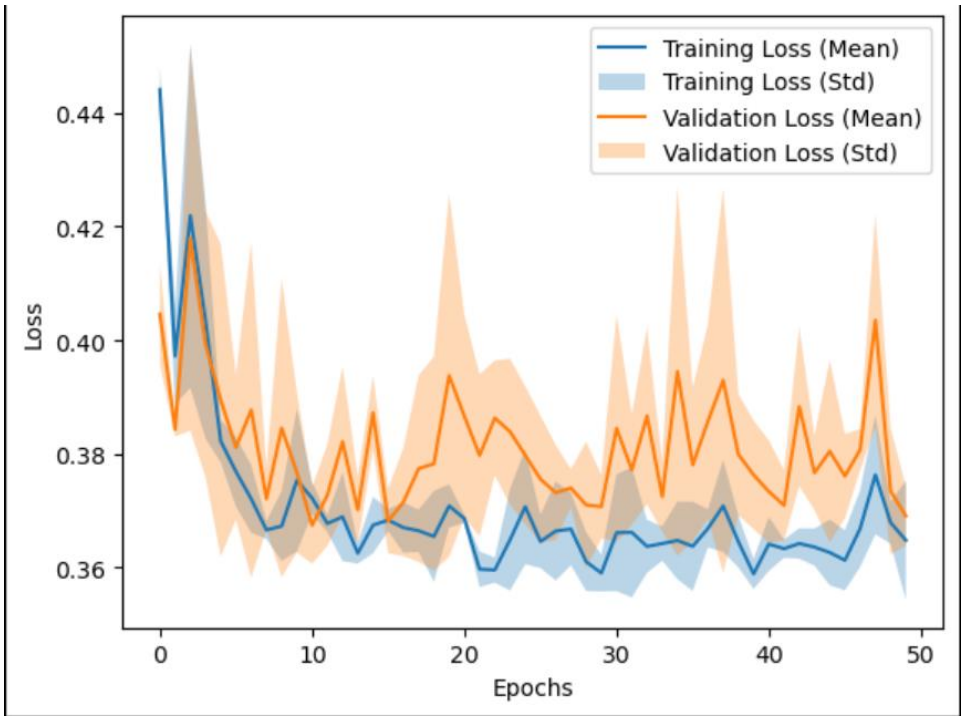
2.2 Création de modèle

Pour le modèle nous avons utilisé un MLP avec 1 couche cachée de 4 neurones et une sortie de 1 neurone. Nous avons utilisé la fonction d'activation tanh comme pour la couche cachée ainsi que pour la couche de sortie. Ce n'est pas un choix original mais elle a été motivée par le fait que nous avons uniquement 2 classes comme résultat possible. Enfin, nous avons utilisé comme learning rate 0.01 et comme momentum 0.9.

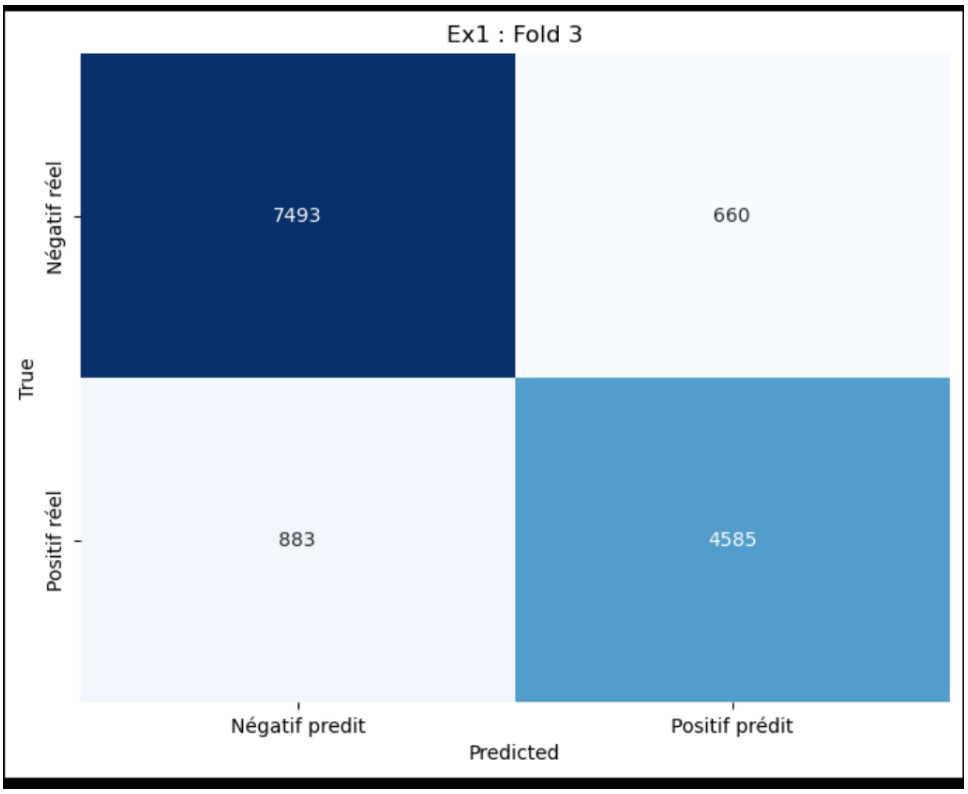
2.3 Entraînement et résultats

Nous avons effectué un entrainement sur 50 epochs et avons décider de split nos données en 3 k-folds pour la validation croisée. Le graphique qui suit nous montre que la perte de nos modèles diminue au fur et à mesure des epochs et ne semble pas diverger en remontant ce qui montre que nous n'avons pas d'overfitting de présent. Néanmoins, le modèle ne semble pas converger non plus. Plus d'epochs ou un changement de learning rate pourraient améliorer les résultats. Un learning rate plus petit pourrait permettre de converger plus rapidement et d'avoir moins de fluctuation dans la perte.

Nous n'avons pas une perte très élevée cependant elle reste assez haute avec un minimum à 0.36. Dans un cas idéal, nous aurions eu l'occasion de diminuer cette erreur.



La matrice de confusion et la moyenne des F1-scores, cette dernière étant de 0.855, que nous obtenons confirme ce que nous avons pu observer avec le graphique de la perte. Un score de 0.855 n'est pas mauvais mais il pourrait être amélioré.



3 Exercice 2 : Classification de 3 classes

Pour ce 2ème modèle, nous avons effectué la même chose que pour le premier modèle mais en ajoutant une classe supplémentaire.

3.1 Préparation des données

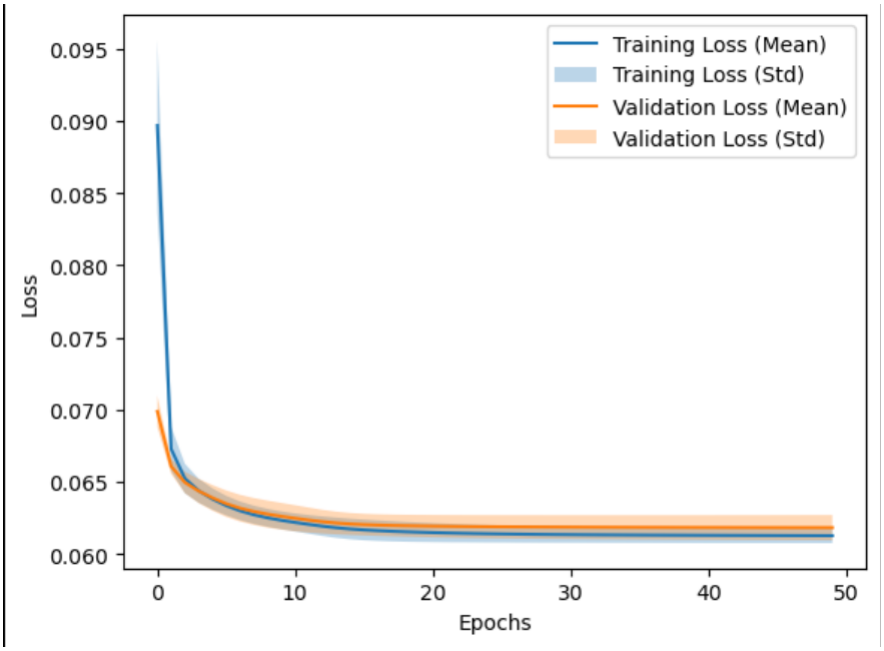
Nous avons à nouveau importé les données de la base de données EEG_mouse_data et effectué un prétraitement des données en utilisant la colonne des états et en la transformant en trois classes. Nous avons regroupé les états n-rem, rem et awake en trois classes différentes. Ensuite, nous avons normalisé les données grâce à la fonction StandardScaler de sklearn. Les valeurs que nous avons attribué à nos classes sont les suivantes : n-rem = 0, rem = 1 et awake = 2.

3.2 Création de modèle

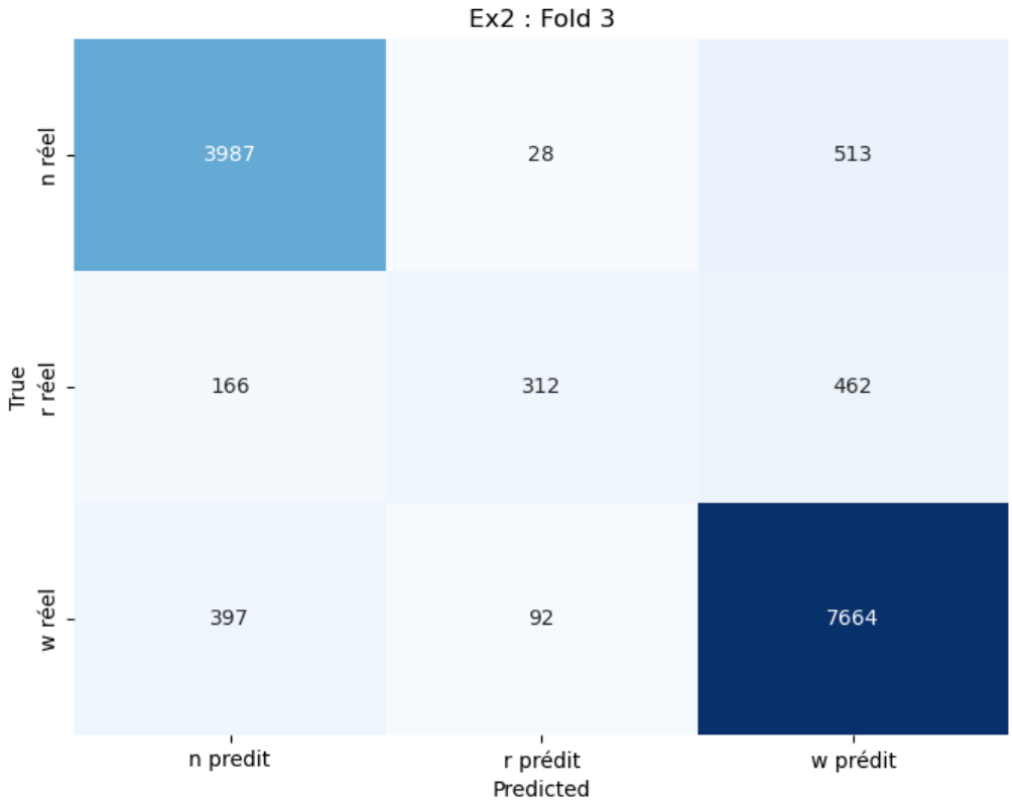
Cette fois-ci, nous avons utilisé un MLP avec 1 couche cachée de 8 neurones et une sortie de 3 neurones. Nous avons utilisé la fonction d'activation sigmoid pour la couche cachée et softmax pour la couche de sortie car cette dernière est plus adaptée pour la classification multi-classes. Après plusieurs essais, c'est finalement avec 4 neurones dans la couche cachée, un learning rate à 0.01 et un momentum de 0.99 que nous avons obtenu les meilleurs résultats.

3.3 Entraînement et résultats

Ici, nous voyons que nous possédons une perte qui diminue au fur et à mesure des epochs et qui ne diverge pas. Nous n'avons toujours pas d'overfitting et le train ainsi que la validation semblent converger. Nous avons, cette fois-ci, une perte minimale qui est tourne autour de 0.060. C'est une amélioration par rapport au premier modèle. Le fait d'avoir plusieurs classes et d'avoir une fonction d'activation plus adaptée pour la classification multi-classes a permis d'améliorer les résultats.



Finalement, nous pouvons observer avec notre matrice de confusion ainsi que le F1-score que les résultats se sont sensiblement amélioré mais pas au point de dépasser les 0.9 d'accuracy car nous possédons une moyenne de 0.879 ce qui est déjà un bon score. Pour améliorer ce score nous pourrions toujours essayer de changer le learning rate le momentum ou encore le nombres d'epoch pour voir si nous pouvons obtenir de meilleurs résultats. À garder en mémoire que plus le nombre d'epoch augmente plus les risques d'overfitting et le temps nécessaire augmentent.



4 Compétition

Pour cette compétition, nous avons repris le même code que l'exercice précèdent avec quelques modifications. Nous avons ajouté une couche cachée de 6 neurones et changé le learning rate pour 0.001. Nous avons aussi décider d'utiliser comme optimiseur ADAM.

Adam signifie "Adaptive Moment Estimation" et c'est un optimiseur qui ajuste le taux d'apprentissage de chaque paramètre individuellement en fonction des estimations des premiers et seconds moments des gradients. Contrairement à SGD qui utilise un taux d'apprentissage fixe, Adam va adapter son taux d'apprentissage tout au long de l'entraînement. Nous avons choisi Adam avec un taux d'apprentissage initial de 0.001.

La fonction de perte Categorical Crossentropy est spécifiquement conçue pour les problèmes de classification multi-classe où l'objectif est de prédire une probabilité pour chaque classe. Cette fonction de perte compare la distribution des probabilités prédites par le modèle avec la véritable distribution des étiquettes et pénalise les différences entre ces distributions.

Ces changements nous ont permis d'augmenter notre moyenne de F1-score à 0.88, ce qui est une légère amélioration par rapport à notre F1-score de l'exercice 2. Malheureusement, nous n'avons pas eu plus de temps pour essayer d'autres combinaisons de paramètre dans l'optique d'obtenir un meilleur F1-score.

Nous avons pu faire la prédiction sur les données de test et les résultats sont dans le fichier test_pred.npy.

5 Conclusion

Ce travail pratique nous a permis d'appliquer nos connaissances en mettant en œuvre des MLPs pour classer des données EEG de souris. Nous avons élaboré des modèles pour la classification en deux et trois classes, tout en utilisant diverses architectures de réseaux de neurones ainsi que des méthodes d'optimisation et des fonctions d'activation adaptées à chaque cas.

En conclusion, cette expérience nous a éclairés sur l'importance cruciale du choix des hyperparamètres, de la structure du modèle et des techniques d'optimisation pour créer des réseaux de neurones performants dans la classification de données complexes telles que celles de l'EEG de souris. Malgré cela, des opportunités d'amélioration subsistent, notamment en continuant à explorer ces aspects et en perfectionnant nos modèles.

6 Note

Le fichier old_version et la première ébauche de notre code. Nous avons décidé de le garder pour montrer notre processus de travail. De plus, il manque le fichier EGG_mouse_data_test dans ce github étant trop lourd pour être uploadé.