Projet – Réseau de neurones intelligent

Introduction

Dans ce projet, vous devez créer un réseau de neurones dans le but de pouvoir jouer au jeu du Tic-Tac-Toe. Vous devrez dans un premier temps créer un réseau de neurones simple basé sur les données fournies. Ensuite, vous devrez créer de nouvelles données d'entraînement en bataillant votre modèle contre lui-même et enfin en entrainant un nouveau modèle sur ces données.

Règles générales

- Le projet s'effectue en équipe de 2 ou individuellement.
- La date de soumission est indiquée sur eCité, vous devrez soumettre un fichier *.ipnyb* contenant votre projet complet.
- L'usage de code copier/coller d'internet, d'IAG (ChatGPT, etc.) ou de membres qui ne font pas parti du groupe est interdit et entraînera une note de 0.
- Assurez-vous de bien soumettre tous les livrables demandés.

Contexte

Vous connaissez certainement le jeu du Tic-Tac-Toe qui consiste simplement à placer un X ou O dans l'une des 9 cases possibles à tour de rôle et former une ligne horizontale, verticale ou diagonale pour gagner. Nous pouvons représenter cette grille dans un vecteur en associant simplement un numéro de 0 à 8 pour chacune des cases. De plus, nous pouvons associer **une case vide** (0), le **X** (1) et le **O** (-1) à des chiffres également.

Case 0	Case 1	Case 2
Case 3	Case 4	Case 5
Case 6	Case 7	Case 8

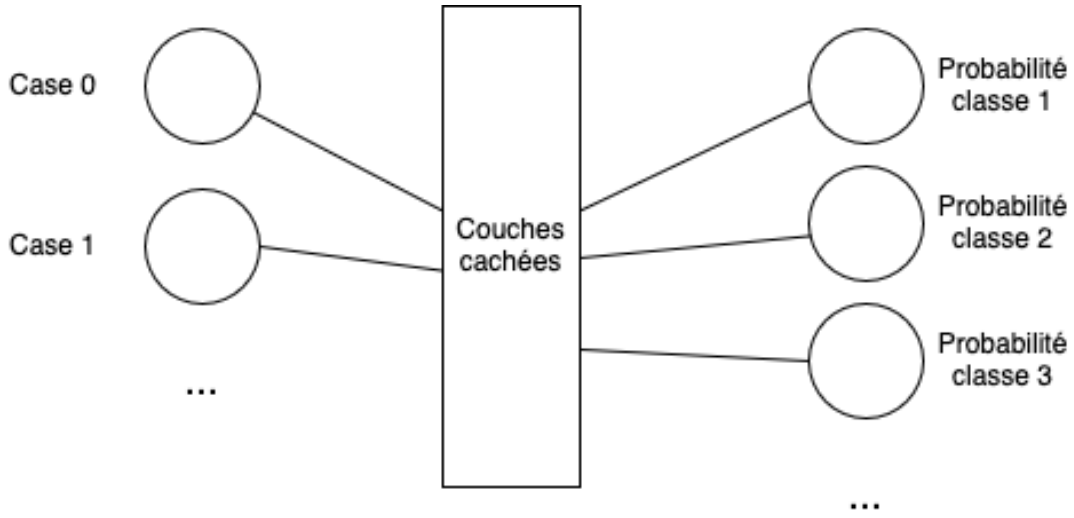
X		
O	O	
X		

⇒

1	0	0
-1	-1	0
1	0	0

En utilisant cette représentation, nous pouvons créer un réseau de neurones prenant l'état (-1, 0, 1) de chacune des cases en entrées et prédisant la case avec le meilleur prochain mouvement. Il s'agit d'un problème de classification **multi-classe**. Voici quelques éléments à considérer pour les classifications multi-classe avec les réseaux de neurones :

- Le nombre de couche de neurones de notre couche de sortie sera égale au nombre de classes à prédire.
- Notre étiquette (y) doit être convertie en un vecteur (voir *tf.keras.utils.to_categorical()*)
- La fonction d'activation *softmax* est utilisée pour les neurones de la couche de sortie. Celles-ci prédiront un nombre entre 0 et 1 représentant la probabilité de chaque classe. Le neurone ayant la plus haute probabilité est utilisée.
- La fonction de perte utilisée est habituellement *categorical_crossentropy*



Dans le jeu de données fourni, chaque ligne contient l'état d'un jeu (le vecteur de 9 éléments) et l'index (0-8) de la case où le prochain coup a été joué par un joueur expérimenté.

Livrable – Jupyter Notebook

Vous devez remettre un Jupyter Notebook sur eCité comprenant minimalement les éléments suivants :

1. **Création d'un réseau de neurones de référence** : En utilisant les données fournies. Incluant tous les éléments si nécessaire, par exemple :
    - Pré-traitement des données
    - Création de l'architecture
    - Entraînement et évaluation en utilisant le(s) métrique(s) appropriée(s)
    - Ajustement des hyperparamètres et/ou de l'architecture afin d'améliorer les performances
    - Usage des bonnes techniques et ajustement des paramètres (ex. fonctions d'activations, arrêt prématuré, nombre epochs, visualisation des courbes d'apprentissage, …)
    - …

2. **Entraîner le modèle à jouer contre lui-même** : En utilisant le modèle créer en premier, créer un nouveau jeu de données (fichier *.csv*) contenant les données de votre modèle jouant contre lui-même :
    - Créer une fonction permettant d'obtenir la case prédite en fonction de l'état du jeu. Dans 20% des cas on prend une case vide aléatoire et dans l'autre 80%, on prend la case prédite par le modèle avec la plus haute probabilité, si elle est déjà occupée on prend la seconde si elle est déjà occupée la troisième, …
    - Créer une fonction permettant de déterminer s'il y a un gagnant ou non dépendent de l'état du jeu.
    - Créer une boucle jouant le jeu à tour de rôle prenant la sortie du modèle et mettant à jour l'état du jeu la case correspondante arrêtant s'il y a un gagnant et stockant les états de jeu et les coups effectués par le gagnant.
    - Exécuter plusieurs fois (ex. 200) afin de collecter un grand nombre de données, si vous en faites plus c'est encore mieux, mais ça peut prendre du temps. Stocker tous les coups des modèles gagnants dans un nouveau fichier *.csv*

3. **Création d'un nouveau modèle** : Créer un nouveau modèle sur les données générées et le comparer au modèle original.
    - Performance du modèle comparé au précédent
    - Discussion et analyse des résultats

Grille d'évaluation

1. **Création du réseau de neurones de référence /35**
    a. Chargement et préparation adéquats des données fournies (e.g., conversion des étiquettes). /5
    b. Conception d'une architecture de réseau de neurones appropriée pour la tâche (nombre de couches, nombre de neurones, fonctions d'activation). Justification des choix. /10
    c. Choix de la fonction de perte et optimiseur. Entraînement du modèle sur les données. Évaluation du modèle à l'aide de la métrique appropriée (e.g., exactitude). /10
    d. Tentatives d'amélioration des performances par l'ajustement des hyperparamètres (taux d'apprentissage, nombre d'époques, taille du lot) et/ou de l'architecture. Utilisation de techniques appropriées (e.g., fonctions d'activation, arrêt prématuré, visualisation des courbes d'apprentissage). Documentation des essais. /10

2. **Entraîner le modèle à jouer contre lui-même /30**
    a. Création des fonctions qui utilise le modèle pour prédire le coup suivant en tenant compte de l'état du jeu et en s'assurant de la validité et qui vérifie le gagnant. /15
    b. Implémentation d'une boucle qui simule des parties complètes (jusqu'à la victoire, la défaite ou le match nul) où le modèle joue contre lui-même. Collecte d'un nombre significatif de données (états du jeu et coups joués) et stockage dans un fichier CSV. /15

3. **Création d'un nouveau modèle /25**

a. Chargement et prétraitement des données générées. Création et entraînement d'un nouveau modèle. /10
b. Évaluation du nouveau modèle (en utilisant la même métrique que pour le modèle original). Comparaison des performances des deux modèles. Discussion et analyse des résultats : expliquer si l'entraînement par auto-jeu a amélioré ou non les performances et proposer des hypothèses. /15

4. **Qualité générale du code et Notebook /10**
    a. Code clair, bien structuré, avec des commentaires pertinents. /5
    b. Le Notebook est bien organisé avec des titres clairs et des explications logiques des étapes suivies, des choix effectués et des résultats obtenus. L'analyse est pertinente et bien argumentée. /5