Série d'exercices
Réseaux de neurones récurrents

9 février 2026

Dans cette série d'exercices nous utilisons Python puis Python/Keras pour créer, valider et analyser des réseaux de neurones récurrents.

---

1 Composantes et fonction d'un RNN

Dans cet exercice, le but est d'écrire un script en Python (standard) permettant d'illustrer le fonctionnement d'un réseau de neurones récurrent. Pour ce faire :

1. On considèrera un RNN ayant une couche cachée.
2. On demandera à l'utilisateur de choisir la taille de la couche d'entrée, celle de la couche cachée et celle de la couche de sortie, ainsi que la taille de la séquence d'entrée.
3. On utilisera tangente hyperbolique comme fonction d'activation de la couche cachée et softmax comme fonction d'activation de la couche de sortie.
4. Les poids et les biais, ainsi que le contenu de la séquence d'entrée, seront générés aléatoirement.

2 Une application des RNNs

Dans cet exercice, nous allons utiliser Python/Keras pour créer, entraîner et tester un RNN. Notre problème est la **prédiction du cours d'une action** en utilisant son **historique**. Pour ce faire, nous procéderons selon les étapes suivantes :

1. Lecture des données (fichier *MSFT.csv* contenant l'historique du cours de l'action de Microsoft entre juin 2017 et juin 2022).

2. Création des données d'apprentissage et de tests. On utilisera exclusivement les prix à la fermeture. Une attention particulière sera accordée à la forme de ces deux ensembles.
3. Création du réseau. Une attention particulière sera accordée (bis) à la fonction d'activation de la couche de sortie et à la fonction de perte.
4. Entraînement puis test du réseau. Une attention particulière sera accordée (ter) à la métrique utilisée dans le test.
5. On terminera par un graphique comparant les prédictions du réseau aux valeurs réelles du cours de l'action.

3 Un LSTM pour générer du texte

Dans cet exercice, notre objectif est de comprendre les différentes étapes d'un processus de **génération de texte**. Pour cela, nous analyserons et compléterons un code illustrant l'utilisation d'un modèle LSTM pour générer un texte caractère par caractère. Une fois entraîné, ce réseau de neurones fonctionnera comme suit :

1. Il reçoit en entrée une séquence *seq* de caractères.
2. Il génère un seul caractère : *ch*.
3. *ch* est ajouté à *seq* et la nouvelle séquence est renvoyée à l'entrée du réseau de neurones.

Pour compléter et utiliser le code Python *forTextGen.py*, vous procéderez comme suit :

1. Affichez les principales caractéristiques des données (le document *texte*).
2. Créez deux dictionnaires : *char_indices* dans lequel la clé est un caractère et la valeur son index, et *indices_char* dans lequel la clé est l'index et la valeur le caractère correspondant.
3. Divisez le texte en exemples qui se chevauchent. Chaque exemple est une paire (séquence de caractères, caractère suivant). Nous appelons la longueur des séquences *seqlen*.
4. Représentez nos exemples en utilisant l'encodage one-hot.
5. Créez un réseau de neurones comportant une couche **LSTM** et une couche *entièrement connectée*.
6. Entraînez le réseau de neurones.
7. Utilisez le réseau de neurones pour prédire une nouvelle séquence suivant une séquence donnée.