Réseaux de neurones avec Keras

TP 3– Outils pour l'apprentissage automatique

Tensorboard : outil de visualisation

Il existe des outils de visualisation très utiles pour analyser un réseau de neurones et surveiller son apprentissage.
Un des outils les plus avancés est Tensorboard, qui permet, par exemple, de suivre l'évolution de la fonction de coût sur le corpus d'apprentissage mais aussi sur un corpus de validation :
https://www.tensorflow.org/guide/summaries_and_tensorboard

Tensorboard permet également de visualiser le graphe que représente le réseau de neurones

I. Les callbacks

Afin de pouvoir réaliser certaines tâches lors de l'apprentissage, Keras propose un mécanisme nommé *callback* : https://keras.io/callbacks/

Un callback doit être initialisé, par exemple pour Tensorbard dans le code Python :
```
tensorboard = TensorBoard(log_dir="logs/{}".format(time()), histogram_freq=0, write_graph=True, write_images=True)
```

Ne pas oublié d'importer les callbacks visés. Par exemple :
```
from keras.callbacks import TensorBoard
```

Puis le callback (ou les callbacks) sont activés lors de l'appel de la méthode *fit* :
```
model.fit(x_train, y_train, nb_epoch=20, verbose=1, validation_data=(x_test, y_test), callbacks=[checkpoint])
```

Pour TensorBoard, il faudra installer le package avec dans le terminal :
```
pip install Tensorboard
```

Puis l'exécuter (serveur web) :
```
tensorboard –logdir=logs/
```

(logdir doit correspondre à la valeur logdir indiquée dans le code Python)

Question 1. En vous aidant de la documentation disponible sur internet :
Afficher les histogrammes des gradients des différentes couches dans Tensorboard
Afficher l'évaluation de la loss et de la métrique choisie dans de le corpus d'apprentissage et celui de validation

Vous avez probablement remarqué durant la séance de TP précédente que par défaut les paramètres du meilleur modèle ne sont pas conservés, mais écrasés par le modèle construit par l'itération suivante.
Un *callback* permet de résoudre ce problème.

Question 2. *En vous aidant encore de la documentation, mettez en place un mécanisme de sauvegarde d'un modèle lorsque celui-ci améliore la métrique choisie sur le corpus d'évaluation. Pour cela, utiliser le callback ModelCheckpoint : https://keras.io/callbacks/#modelcheckpoint*
Après avoir sauvegardé la meilleure version de votre réseau de neurones, écrivez un programme qui charge ce réseau et traite le corpus de test sans réaliser un nouvel apprentissage.

II. Préparation des données et construction d'un réseau de neurones

Question 3. *Construire un réseau de neurones qui traite les données téléchargeables sur l'ENT :*
https://e-uapv2019.univ-avignon.fr/mod/resource/view.php?id=25155

Il s'agit de reconnaître des voyelles à partir de données calculées sur du signal audio.

Consignes :
utiliser les colonnes 'F1', 'F2', 'F3', 'F4', 'Z1', 'Z2', 'f0' comme caractéristiques qui serviront à prédire les valeurs de la colonne 'voyelle'. Il s'agit de différentes informations calculées après analyse du signal audio correspondant à de la parole https://fr.wikipedia.org/wiki/Formant .

Scinder le corpus en trois : le corpus d'apprentissage (80%), un corpus de validation (10%) et un corpus de test (10%).

Aide. Les lignes suivantes peuvent vous être utiles.

```
import pandas as pnd

dataset =
pnd.read_csv('acoustique_voy_orales_20loc_ESTER_NCCFr_contexte_fre
qLex_distCentroide.csv',
             sep='\t', usecols=['voyelle', 'F1', 'F2', 'F3', 'F4',
'Z1', 'Z2', 'f0'])
```

Question 4. *Quel est le meilleur taux de précision que vous obtenez sur le corpus de test, et comment ?*