# TD - TP 5 Réseau de neurones profonds pour la régression

Ce TP s'intéresse à la prédiction des prix des appartements à partir d'un ensemble d'attributs. Vous serez amené à construire votre premier réseau de neurones profonds. Un compte rendu est demandé en fin de séance.

## A) Prise en main de l'environnement de travail

Vous avez la possibilité d'utiliser 2 environnements de travail décrits ci-dessus.

### Google Colab

Pour entrainer un réseau de neurones profond, exigeant en termes de ressources de calculs, il est possible d'entrainer son architecture sur le cloud de Google en exploitant ses GPU. Il suffit de créer un compte utilisateur sur Google Colab. 12 heures d'entrainement (machine virtuelle), avec une vacance de 90 minutes. L'accès à Colab est possible via le lien suivant :
https:\\colab.research.google.com/notebooks/welcome.ipynb
Colab propose de travailler avec l'environnement Jupyter notebook. Les librairies pour l'apprentissage profond sont déjà installées et sont disponibles. Colab utilise google drive pour la sauvegarde Vous devez donc connecter votre Google drive à votre notebook afin de pouvoir accéder au jeu de données d'entrainement et sauvegarder vos programmes.

Ci-dessous, les différentes instructions permettant de vérifier votre environnement de travail :

```
from psutil import *
cpu_count() # indique le nombre de CPU
!lscpu |grep 'Model name' # CPU mode and speed
!df –h / | awk '{print $4}' # available Hard disk space
!free –h –si | awk '/Mem:/{print $2}' # Usable memory
!nvidia-smi –L # GPU specifications.
```

1) Vérifiez l'environnement puis allez dans « runtime », puis dans Notebook settings, sélectionnez GPU
2) Sauvegardez votre fichier : exploring colab.
3) Vérifiez que le fichier est dans votre drive.

### Anaconda

Vous pouvez également travailler sous l'environnement Anaconda en utilisant votre propre machine. TensorFlow est une bibliothèque open source de Machine Learning, créée par Google, permettant de développer et d'exécuter des applications de Machine Learning et de Deep Learning. Il est nécessaire de configurer Anaconda comme suit :

1) Installez tensorflow : pip install tensorflow
2) Redémarrez le noyau
3) Vérifiez que vous pouvez importer tensorflow et keras :

```
import tensorflow as tf
from tensorflow import keras
```

## B) Réseau de neurones profonds pour la régression

Importez tout d'abord les libraies ci-dessous :

```
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn
import KerasRegressor
from pandas import read_csv
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
```

## Exercice 1 : Lecture et labellisation du jeu de données

1) Téléchargez les données et réorganisez les comme suit :

```
df = read_csv("housing_data_for_regression.csv", delim_whitespace=True, eader=None)
```

Analysez le contenu de df.

2) On décide de donner des noms aux attributs comme suit :

```
feature_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
'PTRATIO', 'B', 'LSTAT', 'MEDV']
df.columns = feature_names
```

Vérifiez en affichant :

```
print(df.head())
```

3) Renommez 'MEDV' par 'PRICE' :

```
df = df.rename(columns={'MEDV': 'PRICE'})
print(df.describe())
```

4) Dissociez les attributs à sauveagder dasn X des labels à sauvegarder dans y :

```
X = df.drop('PRICE', axis = 1)
y = df['PRICE']
```

## Exercice 2 : Construction et entrainement du modèle

1) Préparez le jeu de données d'entrainement et de test comme suit :

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 20)
```

2) Avant d'entrainer le modèle, il est nécessaire de normaliser les données. Expliquez chacune des lignes de code ci-dessous :

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3) Commentez chacune des lignes de code :

```
model = Sequential()
model.add(Dense(128, input_dim=13, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='linear'))
```

4) Compilez le modèle :

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae'])
model.summary()
```

5) Entrainez le modèle

```
history = model.fit(X_train_scaled, y_train, validation_split=0.2, epochs =100)
```

## Exercice 3 : Analyse des performances du modèle

1) Affichez les performances obtenus sur les données d'entrainement et de validation à chaque épisode (en changeant la valeur de epochs = 20 puis epochs = 30) :

```
from matplotlib import pyplot as plt
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Commentez les courbes affichées.

```
acc = history.history['mean_absolute_error']
val_acc = history.history['val_mean_absolute_error']
plt.plot(epochs, acc, 'y', label='Training MAE')
plt.plot(epochs, val_acc, 'r', label='Validation MAE')
plt.title('Training and validation MAE')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Commentez les courbes affichées.

2) Testez les performances du modèle entrainé :

```
predictions = model.predict(X_test_scaled[:5])
print("Predicted values are: ", predictions)
print("Real values are: ", y_test[:5])
```

3) Affichez l'erreur quadratique moyenne(mse) et l'erreur moyenne au sens de la valeur absolue (mae) :

```
mse_neural, mae_neural = model.evaluate(X_test_scaled, y_test)
print('Mean squared error from neural net: ', mse_neural)
print('Mean absolute error from neural net: ', mae_neural)
```

## Exercice 4 : Comparaison des performances

Les performances de ce réseau de neurones profonds sont comparées à des algorithmes classiques d'apprentissage statistique (régréssion linéaire, arbre de decision, random Forest).

## Régression linéaire

1) Importez tout d'abord les librairies :

```
from sklearn import linear_model
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

2) Commentez chacune des lignes du code :

```
lr_model = linear_model.LinearRegression()
lr_model.fit(X_train_scaled, y_train)
y_pred_lr = lr_model.predict(X_test_scaled)
```

2) Analysez et comparez les performances de ce modèle au réseau de neurones profonds :

```
mse_lr = mean_squared_error(y_test, y_pred_lr)
mae_lr = mean_absolute_error(y_test, y_pred_lr)
print('Mean squared error from linear regression: ', mse_lr)
print('Mean absolute error from linear regression: ', mae_lr)
```

## Arbres de décision

1) Commentez chacune des lignes du code :

```
tree = DecisionTreeRegressor()
tree.fit(X_train_scaled, y_train)
y_pred_tree = tree.predict(X_test_scaled)
```

2) Analysez et comparez les performances de ce modèle au réseau de neurones profonds :

```
mse_dt = mean_squared_error(y_test, y_pred_tree)
mae_dt = mean_absolute_error(y_test, y_pred_tree)
print('Mean squared error using decision tree: ', mse_dt)
print('Mean absolute error using decision tree: ', mae_dt)
```

## Random Forest

1) Commentez chacune des lignes du code :

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators = 30, random_state=30)
model.fit(X_train_scaled, y_train)
y_pred_RF = model.predict(X_test_scaled)
```

2) Analysez et comparez les performances de ce modèle au réseau de neurones profonds :

```
mse_RF = mean_squared_error(y_test, y_pred_RF)
mae_RF = mean_absolute_error(y_test, y_pred_RF)
print('Mean squared error using Random Forest: ', mse_RF)
print('Mean absolute error Using Random Forest: ', mae_RF)
```