Tuto TensorFlow

- **Installation :**

https://www.tensorflow.org/install/pip

Prendre le package TensorFlow – Current release for CPU-only (recommended for beginners)

!! TensorFlow est installé dans l'environnement virtuel. Il faut donc se placer dans cet environnement lorsque l'on souhaite utiliser la bibliothèque.

Pour se faire il faut tout d'abord utiliser la commande suivante : **source ./venv/bin/activate**

Et pour quitter l'environnement virtuel : **deactivate**

```
laure@laure-VirtualBox ~ $ source ./venv/bin/activate
(venv) laure@laure-VirtualBox ~ $ cd Documents/
(venv) laure@laure-VirtualBox ~/Documents $ python3 hey.py
Hello !
(venv) laure@laure-VirtualBox ~/Documents $ deactivate
laure@laure-VirtualBox ~/Documents $
```

- **Prise en main :**

https://www.tensorflow.org/tutorials

https://intelligence-artificielle.agency/tensorflow/

TensorFlow permet de créer rapidement et facilement des modèles de réseaux de neurones.

Schéma type d'architecture :

1) Importation des données
2) Description des données
3) Sélection du modèle
4) Apprentissage du modèle
   - ➔ Définition de la fonction d'apprentissage
   - ➔ Appel de la fonction d'apprentissage
5) Evaluation du modèle
   - ➔ Définition de la fonction d'évaluation
   - ➔ Appel de la fonction d'évaluation
6) Prédictions

- **Fonctions de TensorFlow :**

https://www.tensorflow.org/api_docs/python

- **Tests :**

Exemple d'un XOR :

```python
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import time

def init_weights(shape):
    return tf.Variable(tf.random_normal(shape, stddev=0.1))

x = np.array([[1, 0], [0, 1], [1, 1], [0, 0]])
y = np.array([[1], [1], [0], [0]])

# define placeholders for input x and output y
tf_features = tf.placeholder(tf.float32, [None, 2])    # x
tf_targets = tf.placeholder(tf.float32, [None, 1])     # y

# init weights
# 5 hidden nodes
w1 = init_weights([2, 3])
b1 = tf.Variable(1.0, [3])
w2 = init_weights([3, 1])
b2 = tf.Variable(1.0, [1])

# first layer
z1 = tf.matmul(tf_features, w1) + b1
a1 = tf.nn.sigmoid(z1)

# output layer
z2 = tf.matmul(a1, w2) + b2
py = tf.nn.sigmoid(z2)

# init learning rate
lr = 0.01
# init epochs
epochs = 50000

# init cost function
cost = tf.reduce_mean(tf.square(py-tf_targets))
```

Importation des données

Description des données

Sélection du modèle

Fonction d'évaluation

Fonction d'apprentissage

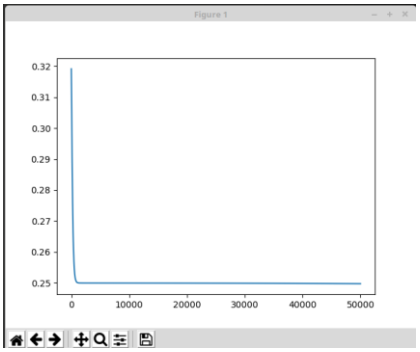
```python
# train function and optimizer
optimizer = tf.train.GradientDescentOptimizer(lr)
train = optimizer.minimize(cost)  #opération d'entrainement qui minimise le cout (avec la méthode de descente de gradient)
#train = tf.train.AdamOptimizer(lr).minimize(cost)

# save costs for plotting
costs = []

# create session and init variables
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

start=time.time()
# start training
for i in range(epochs):
    sess.run(train, feed_dict={tf_features: x, tf_targets: y})

    c = sess.run(cost, feed_dict={tf_features: x, tf_targets: y})
    costs.append(c)

    if i % 5000 == 0:
        print("Epoch: ", i, " Cost: ", c)

print("Training complete.")
end=time.time()
print("Time :", end-start)

# make prediction
correct_prediction = tf.equal(tf.round(py), tf_targets) #définition d'une bonne prédiction
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

print("Theo :\n", sess.run(tf_targets, feed_dict={tf_targets: y}))
print("Reel :\n", sess.run(py, feed_dict={tf_features: x, tf_targets: y}))
print("Accuracy :", sess.run(accuracy, feed_dict={tf_features: x, tf_targets: y}))

# plot cost
plt.plot(costs)
plt.show()
```

Appel des fonctions d'apprentissage et d'évaluation

Prédictions

Résultats :

```
Epoch:  0  Cost:  0.31913754
Epoch:  5000  Cost:  0.24996832
Epoch:  10000  Cost:  0.24995966
Epoch:  15000  Cost:  0.24994963
Epoch:  20000  Cost:  0.24993765
Epoch:  25000  Cost:  0.24992324
Epoch:  30000  Cost:  0.2499056
Epoch:  35000  Cost:  0.24988359
Epoch:  40000  Cost:  0.24985568
Epoch:  45000  Cost:  0.24981974
Training complete.
Time : 53.91649293899536
Theo :
 [[1.]
 [1.]
 [0.]
 [0.]]
Reel :
 [[0.5043603 ]
 [0.49539614]
 [0.4989953 ]
 [0.4998104 ]]
Accuracy : 0.75
```



- **Difficultés rencontrées :**

Il y a beaucoup de fonctions, ce n'est pas facile de s'y retrouver lorsqu'on débute. Il faut s'habituer à l'architecture à utiliser.