TENSORFLOW: Combining Everything Together

December 12, 2017

INTRODUCTION :

On va effectuer une classification binaire sur le jeu de données Iris qu'on a importé du site Kaggle. Notre base de données est composée de cinq colonnes qui sont : Identifiant,longueur de petal,largeur de petal,longueur de sepal,largeur de sepal,type d'éspèces.On a 3 types d'iris : setosa ,versicolore,virginica . Le code qu'on va éxécuter est disponible sur le site Github ,il est introduit par le data scientist RichBrosius. On va créer un classificatuer binaire simple en créant une droite et on éxécutant le tout avec une fonction sigmoide pour obtenir un prédicteur binaire . On va se baser sur deux caractéristiques : longuer de petal et largeur de petal puisque l'éspèce setosa est séparable par ces deux caractéristiques.Alors ,nous visons à trouver la droite qui les séparent(setosa ou pas).

On commance par l'installation de tensorflow : !pip install tensorflow

C'EST QUOI TENSORFLOW ? C'est un outil open source d'apprentissage automatique appliqué dans le domaine d intelligence artificielle prenant tout son sens en donnant les capacités evolutives au machine qui s'appuient sur ce domaine afin de remplir des tâches qu'il est généralement difficile aux algorithmes traditionnels de remplir.

CODE :

```
In [13]: #Premièrement, on importe les bibliothèques dont on a besoin lors de
         #l'éxecution de programme
         #et on initialise le graphe de calcul.
         #On importe la bibliothèque matplotlib ici pour avoir le graph résultat.
         import matplotlib.pyplot as plt
         import numpy as np
         from sklearn import datasets
         import tensorflow as tf
         from tensorflow.python.framework import ops
         ops.reset_default_graph()
```

```
In [3]: ## Load the iris data
        ## iris.target = {0, 1, 2}, where '0' is setosa
        ## iris.data ~ [sepal.width, sepal.length, petal.width, petal.length]
        #Maintenant , on va importer la base de données Iris ,composée de 151 observations
        #et 5 colonnes qui sont :
        #ID, sepallength,sepalwidth,petallength,petalwidth,species.
        #On a besoin de transformer la base cible pour devenir 1 ou 0 .
        #Puisque iris marque le type setosa par 0,on va changer les 0 par 1 et les 1 par 0.
        #Et c'est en utilisant seulement les 2 critères la longueur et la largeur de petal .
```

```
        #c'est à dire la 3 ème et la 4 ème entrée
        #pour chaque valeur de x.
        iris = datasets.load_iris()
        binary_target = np.array([1. if x==0 else 0. for x in iris.target])
        iris_2d = np.array([[x[2], x[3]] for x in iris.data])
```

```
In [14]: batch_size = 20 #On définit la taille de l'échantillon par 20 .
```

```
In [4]: ## Create graph
        # On commence une session de graph de calcul
        sess = tf.Session()
```

```
In [ ]: ## Declare placeholders
        #Nous déclarons les espaces réservés pour le modèle.
        #Juste pour illustrer le fait que nous pouvons ajouter plusieurs entités x séparément,
        # nous créons deux espaces réservés distincts pour les deux facteurs de la base Iris.
        x1_data = tf.placeholder(shape=[None, 1], dtype=tf.float32)
        x2_data = tf.placeholder(shape=[None, 1], dtype=tf.float32)
        y_target = tf.placeholder(shape=[None, 1], dtype=tf.float32)
```

```
In [5]: ##Create variables A and b :
        # On va construire un modèle linéaire .Pour cela on a besoin de créer deux variables
        #a(pente) et b.
        A = tf.Variable(tf.random_normal(shape=[1, 1]))
        b = tf.Variable(tf.random_normal(shape=[1, 1]))
        ## Add model to graph:
        # x1 - A*x2 + b
```

```
In [6]: #une droite est définie par x1=A*x2+b , pour créer un separateur linaire on a besoin
        #de savoir du quelle coté de la droite
        #les points vont situés , il y a 3 cas :
        #le point est exactement sur la droite  0=x1-(A.X2+b)
        #le point est au dessus de la droite 0> x1-(A*X2+b)
        #le point est au dessous de la droite 0< x1-(A*X2+b)
        #Donc ,on va dégager l'output de ce modèle : x1-(A*x2+b)
        #Les prévisions vont être le signe de cet output
        #predicition(x1,x2)=signe(x1-(A*x2+b))
        #Alors on ajoute les opérations correspondantes au graphe de calcul.
        # Add model to graph:
        # x1 - A*x2 + b
        bmy_mult = tf.matmul(x2_data, A)
        my_add = tf.add(my_mult, b)
        my_output = tf.subtract(x1_data, my_add)
```

```
In [7]: ## Loss Function
        #Puisque nous faisons une prédiction catégorique (I.setosa ou pas),
        #nous utiliserons la perte d'entropie croisée sigmoïde.
        #C'est une fonction qui  est pour nous fournie par TensorFlow.
        # Add classification loss (cross entropy)
        xentropy = tf.nn.sigmoid_cross_entropy_with_logits(logits=my_output,labels=y_target)
```

In [8]:
```
## Create Optimizer
#Optimisation de la fonction et de l'initialisation des variables
#Nous utilisons la fonction d'optimisation de descente de gradient standard
#avec un taux d'apprentissage de 0,05.
#Nous ajoutons et exécutons ensuite une opération d'initialisation de variable.
my_opt = tf.train.GradientDescentOptimizer(0.05)
train_step = my_opt.minimize(xentropy)
```

In [9]:
```
## Initialize variables
init = tf.global_variables_initializer()
sess.run(init)
```

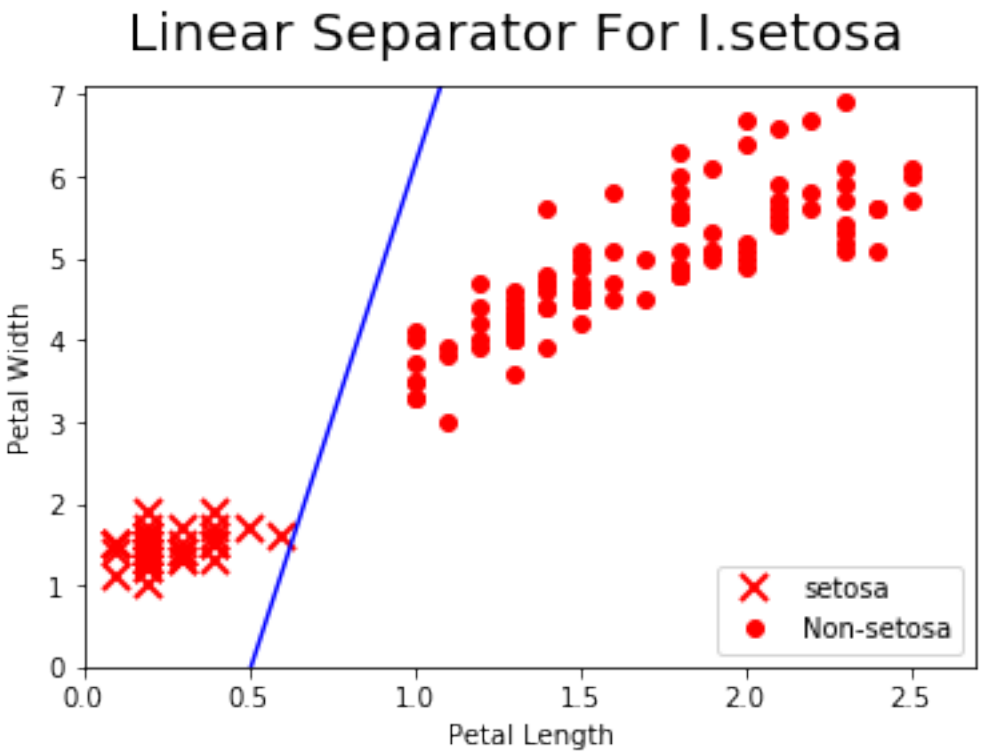
In [10]:
```
### Run Classification :
 #Nous exécutons la classification pour 1000 itérations
 #et fournissons les valeurs de A, b et de perte toutes les 200 itérations.
 ## Run Loop
 for i in range(1000):
     rand_index = np.random.choice(len(iris_2d), size=batch_size)
     #rand_x = np.transpose([iris_2d[rand_index]])
     rand_x = iris_2d[rand_index]
     rand_x1 = np.array([[x[0]] for x in rand_x])
     rand_x2 = np.array([[x[1]] for x in rand_x])
     #rand_y = np.transpose([binary_target[rand_index]])
     rand_y = np.array([[y] for y in binary_target[rand_index]])
     sess.run(train_step, feed_dict={x1_data: rand_x1, x2_data:
                                     rand_x2, y_target: rand_y})
     if (i+1)%200==0:
     print('Step #' + str(i+1) + ' A = ' + str(sess.run(A)) + '
           , b = ' + str(sess.run(b)))
```

```
Step #200 A = [[ 8.66421032]], b = [[-3.50700784]]
Step #400 A = [[ 10.26213455]], b = [[-4.56434965]]
Step #600 A = [[ 11.10057354]], b = [[-5.38356638]]
Step #800 A = [[ 11.86108303]], b = [[-5.83125925]]
Step #1000 A = [[ 12.38840485]], b = [[-6.27848339]]
```

In [11]:
```
## Visualize Results:
 ## Nous tirons la pente et l'ordonné à l'origone et on trace les prédictions :
 # Pull out slope/intercept
 [[slope]] = sess.run(A)
 [[intercept]] = sess.run(b)

 # Create fitted line
 x = np.linspace(0, 3, num=50)
 ablineValues = []
 for i in x:
   ablineValues.append(slope*i+intercept)
```

```
# Plot the fitted line over the data
setosa_x = [a[1] for i,a in enumerate(iris_2d) if binary_target[i]==1]
setosa_y = [a[0] for i,a in enumerate(iris_2d) if binary_target[i]==1]
non_setosa_x = [a[1] for i,a in enumerate(iris_2d) if binary_target[i]==0]
non_setosa_y = [a[0] for i,a in enumerate(iris_2d) if binary_target[i]==0]
plt.plot(setosa_x, setosa_y, 'rx', ms=10, mew=2, label='setosa')
plt.plot(non_setosa_x, non_setosa_y, 'ro', label='Non-setosa')
plt.plot(x, ablineValues, 'b-')
plt.xlim([0.0, 2.7])
plt.ylim([0.0, 7.1])
plt.suptitle('Linear Separator For I.setosa', fontsize=20)
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.legend(loc='lower right')
plt.show()
```



CONCLUSION : On a pu visualiser l'Output qui est un graphe composé de la droite qu'on a estimé précedemmant et de deux classes d'Iris représentées sous formes de points : Les points qui se situent au dessous de la droite représente bien la classe setosa et les points qui se situent au dessus de la droite traduit bien les autres classes(versicolores,virginica). On mentionne que la précision de notre modèle linèaire est de 95% puisque on a introduit un taux d'apprentissage de 0,05.

TRAVAIL REALISE PAR : Bahria Hela et Becheikh Asma