Ejercicio 2

January 31, 2021

1 Importamos las librerias necesarias.

[1]:
```
from scipy import stats
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from random import seed
```

[2]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
```

[3]:
```
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.models import Model
from tensorflow.keras.models import load_model
from tensorflow.keras.callbacks import ModelCheckpoint
```

2 Definimos lo modelos 'model_et' y model_peso correspondientes a los operadores '&' y ' $'.

[4]:
```
model_et = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=[2]),
    keras.layers.Dense(1)
  ])
model_et.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae',␣
 ↪'mse'])
model_et.summary()

model_peso = keras.Sequential([
```

```
    keras.layers.Dense(64, activation='relu', input_shape=[2]),
    keras.layers.Dense(1)
  ])
model_peso.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae',␣
 ↪'mse'])
model_peso.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                192
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 257
Trainable params: 257
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 64)                192
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 257
Trainable params: 257
Non-trainable params: 0
_________________________________________________________________
```

3 Definimos nuestros datos correspondientes a los operadores '&' y ' $'.

[5]:
```
x_train = np.array([
    [0,0],
    [1,0],
    [0,1],
    [1,1]
])

y_train_et = np.array([1,1,1,0])

y_train_peso = np.array([0,0,0,1])
```

4 Realizamos el entrenamiento de los modelos con los datos correspondietes.

[6]:
```
model_et.fit(x=x_train, y=y_train_et, batch_size=5, epochs=5000, verbose=0)
```

[6]:
```
<tensorflow.python.keras.callbacks.History at 0x1ae2fee3dc8>
```

[7]:
```
model_peso.fit(x=x_train, y=y_train_peso, batch_size=5, epochs=5000, verbose=0)
```

[7]:
```
<tensorflow.python.keras.callbacks.History at 0x1ae311a1408>
```

5 Verificamos que que el modelo funcione bien en los datos originales.

[8]:
```
y_pred_et = model_et.predict(np.array([[0,0],[1,0],[0,1],[1,1]]))
y_pred_et
```

[8]:
```
array([[1.0000001e+00],
       [1.0000000e+00],
       [9.9999994e-01],
       [1.4901161e-08]], dtype=float32)
```

[9]:
```
y_pred_peso = model_peso.predict(np.array([[0,0],[1,0],[0,1],[1,1]]))
y_pred_peso
```

[9]:
```
array([[ 0.0000000e+00],
       [-1.8626451e-09],
       [-5.5879354e-09],
       [ 1.0000001e+00]], dtype=float32)
```

6 Finalmente, realizamos las operaciones que se piden en el problema.

[10]:
```
a = [1.001,0,0.001,1]
b = [0,1,0,1]
c = [0,1,1,0]

a_et_b = []
for i in range(4):
    a_et_b.append([a[i],b[i]])
a_et_b
```

[10]:
```
[[1.001, 0], [0, 1], [0.001, 0], [1, 1]]
```

[11]:
```
a_et_b = model_et.predict(np.array(a_et_b))
a_et_b
```

[11]:
```
array([[1.0000385e+00],
       [9.9999994e-01],
       [1.0001057e+00],
       [1.4901161e-08]], dtype=float32)
```

[12]:
```
a_et_b_peso_c = []
for i in range(4):
    a_et_b_peso_c.append([a_et_b.tolist()[i][0],c[i]])
a_et_b_peso_c
```

[12]:
```
[[1.000038504600525, 0],
 [0.9999999403953552, 1],
 [1.0001057386398315, 1],
 [1.4901161193847656e-08, 0]]
```

[13]:
```
a_et_b_peso_c = model_peso.predict(np.array(a_et_b_peso_c))
a_et_b_peso_c
```

[13]:
```
array([[1.16135925e-05],
       [1.00000000e+00],
       [1.00003064e+00],
       [3.72529030e-09]], dtype=float32)
```

7 Observemos que el resultado es muy cercano al [0,1,1,0].