modelCreation

20 juin 2022

1 Model creation

1.1 Libs

[1]:
```
from prepareRSSI import RssiDatas
import pandas as pd
import parameters as param
import numpy as np
import tensorflow as tf
#import intel-tensorflow as tf
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from tensorflow.keras import backend as K # to set the learning rate
```

1.2 Import the RSSI, mac and zones values

[2]:
```
RSSI = RssiDatas()

rssi_tmp = np.genfromtxt(param.rssi_csv_output, delimiter=',')
# deleting index column and row :
rssi_tmp = np.delete(rssi_tmp, 0, 1)
RSSI.rssi = np.delete(rssi_tmp, 0, 0)

mac_tmp = np.genfromtxt(param.mac_csv_output, delimiter=',', dtype=str)
# deleting index column and row :
mac_tmp = mac_tmp[:,1] # keep only the column with the mac adresses
RSSI.mac = np.delete(mac_tmp, 0) # delete the first row because empty

zones_tmp = np.genfromtxt(param.zones_csv_output, delimiter=',')
# deleting index column and row :
zones_tmp = zones_tmp[:,1] # keep only the colomn with the zone ids
RSSI.zones = np.delete(zones_tmp, 0, 0) # delete the first row because empty
```

1.3 Pre load datas into model

[3]:
```
print("RSSI.rssi : ", np.shape(RSSI.rssi))
print("RSSI.zones : ", np.shape(RSSI.zones))
zonesNb = 7 # number of zones

# create the zone output array of vectors :
y_zones = np.zeros([np.size(RSSI.zones), zonesNb])
for i in range(np.size(RSSI.zones)):
    y_zones[i, int(RSSI.zones[i])] = 1

train_data, test_data, train_labels, test_labels = train_test_split(RSSI.
 ↪rssi,y_zones)
print("train_data : ", np.shape(train_data))
print("train_labels : ", np.shape(train_labels))
print("test_data : ", np.shape(test_data))
print("test_labels : ", np.shape(test_labels))
# normalize the RSSI values from 0 to 1
train_data = train_data/(-95)
test_data = test_data/(-95)
```

```
RSSI.rssi :  (265, 128)
RSSI.zones :  (265,)
train_data :  (198, 128)
train_labels :  (198, 7)
test_data :  (67, 128)
test_labels :  (67, 7)
```

1.4 Create the model

[4]:
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(np.size(RSSI.mac), activation='relu'),
  tf.keras.layers.Dropout(param.dropout),
  tf.keras.layers.Dense(np.size(RSSI.mac), activation='relu'),
  tf.keras.layers.Dropout(param.dropout),
  tf.keras.layers.Dense(np.size(RSSI.mac), activation='relu'),
  tf.keras.layers.Dropout(param.dropout),
  tf.keras.layers.Dense(7)
])
```

1.5 Compile the model

[5]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.MeanSquaredError(),
              metrics=['accuracy'])
K.set_value(model.optimizer.learning_rate, param.learningRate)
```

1.6 Train the model

[6]:
```
history = model.fit(train_data, train_labels,
                    epochs=param.epochs,
                    validation_data=(test_data, test_labels))
```

```
Epoch 1/5000
7/7 [==============================] - 0s 17ms/step - loss: 0.5503 - accuracy:
0.0960 - val_loss: 0.2526 - val_accuracy: 0.0896
Epoch 2/5000
7/7 [==============================] - 0s 3ms/step - loss: 0.4314 - accuracy:
0.1010 - val_loss: 0.1921 - val_accuracy: 0.0896

...

Epoch 4999/5000
7/7 [==============================] - 0s 3ms/step - loss: 0.0027 - accuracy:
1.0000 - val_loss: 0.0030 - val_accuracy: 1.0000
Epoch 5000/5000
7/7 [==============================] - 0s 4ms/step - loss: 0.0035 - accuracy:
1.0000 - val_loss: 0.0027 - val_accuracy: 1.0000
```

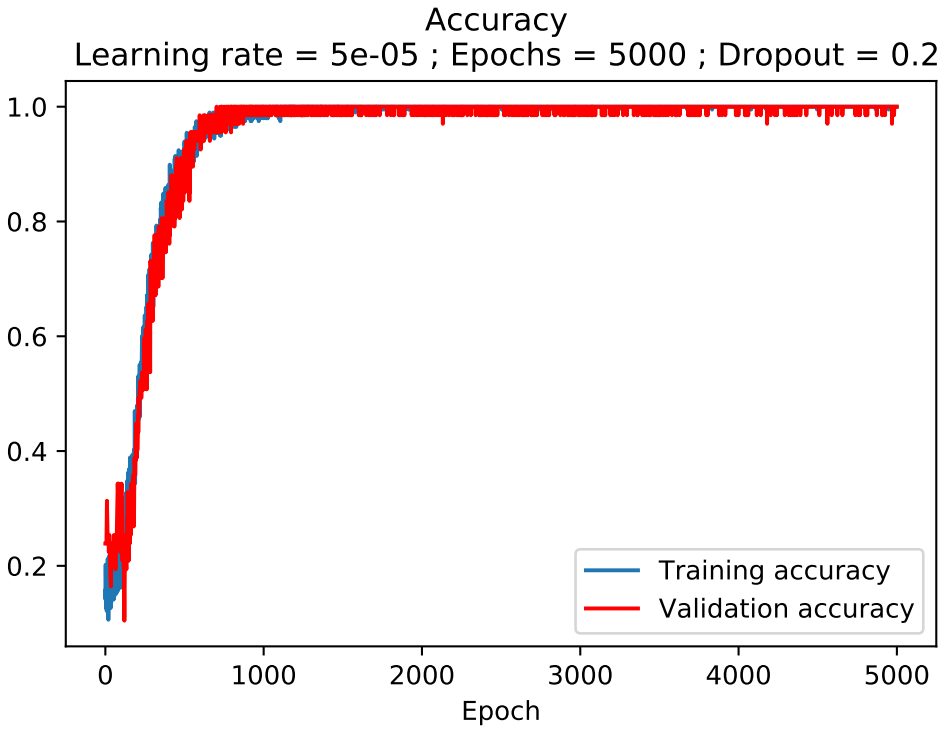
1.7 Ploting learning datas

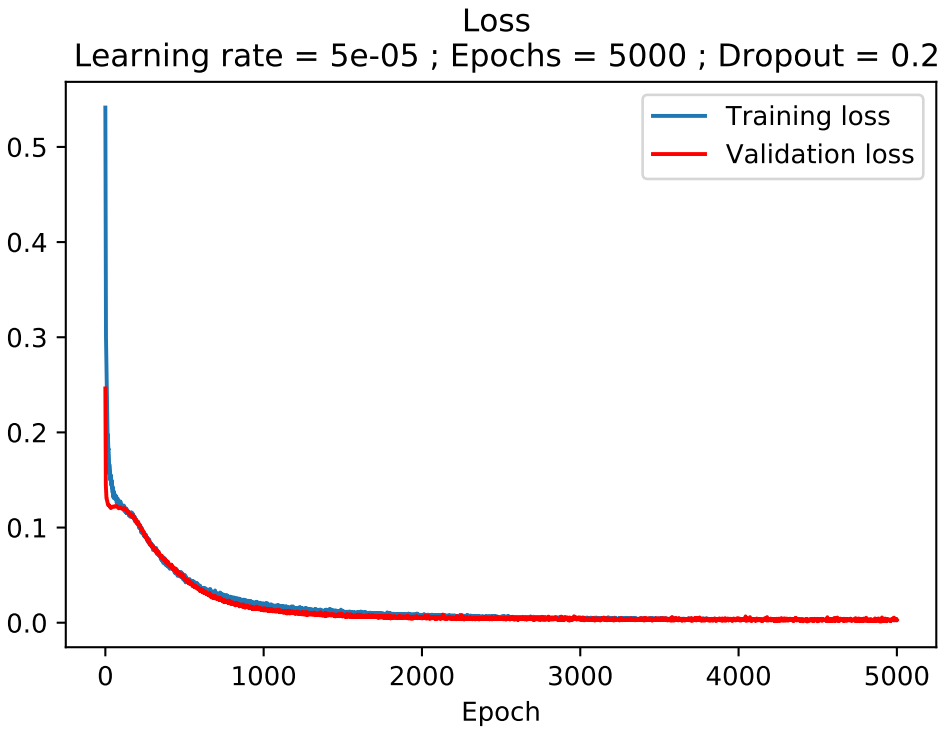
[7]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(1)
plt.plot(acc, label="Training accuracy")
plt.plot(val_acc,color="red", label="Validation accuracy")
plt.legend(loc="lower right")
plt.title("Accuracy \n Learning rate = " + str(param.learningRate) + " ; Epochs␣
 ↪= " + str(param.epochs) + " ; Dropout = " + str(param.dropout))
plt.xlabel("Epoch")
name = param.plot_saving_location + "plot_accuracy_lr_" + str(param.
 ↪learningRate) + "_ep_" + str(param.epochs) + "_dr_" + str(param.dropout) + ".
 ↪pdf"
name = name.replace(".", ",", 2) # replace the two first dots
plt.savefig(name, format="pdf")

plt.figure(2)
plt.plot(loss, label="Training loss")
plt.plot(val_loss,color="red", label="Validation loss")
plt.legend(loc="upper right")
```

```
plt.title("Loss \n Learning rate = " + str(param.learningRate) + " ; Epochs = "␣
 ↪+ str(param.epochs) + " ; Dropout = " + str(param.dropout))
plt.xlabel("Epoch")
name = param.plot_saving_location + "plot_loss_lr_" + str(param.learningRate) +␣
 ↪"_ep_" + str(param.epochs) + "_dr_" + str(param.dropout) + ".pdf"
name = name.replace(".", ",", 2)
plt.savefig(name, format="pdf")
```




Loss
Learning rate = 5e-05 ; Epochs = 5000 ; Dropout = 0.2
Training loss
Validation loss
0.5
0.4
0.3
0.2
0.1
0.0
0
1000
2000
3000
4000
5000
Epoch