Iris

February 12, 2021

[ ]:
```
#Tutorial for Iris Dataset
#code based on the tutorial @ https://www.kdnuggets.com/2020/07/
↪getting-started-tensorflow2.html
```

[ ]:
```
#load data
from sklearn.datasets import load_iris
iris = load_iris()
```

[ ]:
```
#load other needed libraries
from sklearn.model_selection import train_test_split #to split data

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
```

[ ]:
```
#convert into data frame
X = pd.DataFrame(data = iris.data, columns = iris.feature_names)
y = pd.DataFrame(data=iris.target, columns = ['irisType'])
```

[ ]:
```
#check if data is complete
X.info()
```

[ ]:
```
#Split data into training and test set
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.1)
```

[ ]:
```
#check variance
X_train.var(), X_test.var()
```

[ ]:
```
#Convert into OneHotVector
y_train = tf.keras.utils.to_categorical(y_train)
y_test  = tf.keras.utils.to_categorical(y_test)
```

[ ]:
```
#convert data back to numpy arrays
X_train = X_train.values
```

```
X_test  = X_test.values
```

[ ]:
```
#set up model 1
model1 = Sequential()
model1.add(Dense(64,activation='relu', input_shape= X_train[0].shape))
model1.add(Dense(128,activation='relu'))
model1.add(Dense(128,activation='relu'))
model1.add(Dense(128,activation='relu'))
model1.add(Dense(128,activation='relu'))
model1.add(Dense(64,activation='relu'))
model1.add(Dense(64,activation='relu'))
model1.add(Dense(64,activation='relu'))
model1.add(Dense(64,activation='relu'))
model1.add(Dense(3,activation='softmax'))
```

[ ]:
```
model1.compile(optimizer='adam', loss='categorical_crossentropy',␣
↪metrics=['acc'])
```

[ ]:
```
history = model1.fit(X_train, y_train, batch_size = 40,␣
↪epochs=800,validation_split = 0.1)
```

[ ]:
```
#plot accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.xlabel('Epochs')
plt.ylabel('Acc')
plt.legend(['Training', 'Validation'], loc='upper right')
```

[ ]:
```
#plot loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Training', 'Validation'], loc='upper left')
```

[ ]:
```
#evaluation with test data
model1.evaluate(X_test, y_test)
```

[ ]:
```
#set up model 2 with regularization and dropout
model2 = Sequential()
model2.add(Dense(64, activation = 'relu', input_shape= X_train[0].shape))
model2.add(Dense(128, activation = 'relu', kernel_regularizer=tf.keras.
↪regularizers.l2(0.001)))
model2.add(Dense (128, activation = 'relu',kernel_regularizer=tf.keras.
↪regularizers.l2(0.001)))
model2.add(tf.keras.layers.Dropout(0.5))
```

```
model2.add(Dense (128, activation = 'relu', kernel_regularizer=tf.keras.
 ↪regularizers.l2(0.001)))
model2.add(Dense(128, activation = 'relu', kernel_regularizer = tf.keras.
 ↪regularizers.l2(0.001)))
model2.add(Dense (64, activation = 'relu', kernel_regularizer=tf.keras.
 ↪regularizers.l2(0.001)))
model2.add(Dense (64, activation = 'relu', kernel_regularizer=tf.keras.
 ↪regularizers.l2(0.001)))
model2.add(tf.keras.layers.Dropout(0.5))
model2.add(Dense (64, activation = 'relu', kernel_regularizer=tf.keras.
 ↪regularizers.l2(0.001)))
model2.add(Dense (64, activation = 'relu', kernel_regularizer=tf.keras.
 ↪regularizers.l2(0.001)))
model2.add(Dense (3, activation = 'softmax', kernel_regularizer=tf.keras.
 ↪regularizers.l2(0.001)))
```

[ ]:
```
model2.compile(optimizer='adam',␣
 ↪loss='categorical_crossentropy',metrics=['acc'])
history2 = model2.fit(X_train, y_train, epochs=800, validation_split=0.1,␣
 ↪batch_size=40)
```

[ ]:
```
#plot accuracy model 2
plt.plot(history2.history['acc'])
plt.plot(history2.history['val_acc'])
plt.title('Accuracy vs. epochs')
plt.ylabel('Acc')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='lower right')
plt.show()
```