Universidade do Minho
Departamento de Informática
Mestrado [integrado] em Engenharia Informática

Perfil de Machine Learning: Fundamentos e Aplicações
Classificadores e Sistemas Conexionistas
4º Ano, 2º Semestre
Ano letivo 2019/2020

Enunciado Prático nº 2
20 de fevereiro de 2020

Tema

Introdução ao *TensorFlow* e Treino de uma Rede Neuronal

Enunciado

Pretende-se, com esta ficha, que seja realizado um conjunto de tarefas que permitam uma maior compreensão da API do *TensorFlow*. Pretende-se também promover um espírito crítico e de investigação na implementação, pela primeira vez, do treino de uma rede neuronal.

Tarefas

Esta ficha encontra-se dívida em duas partes distintas.

1. Na primeira parte desta ficha prática pretende-se que sejam resolvidos os seguintes exercícios:
   - Criar dois *tensors* de *rank* 0, a e b, de qualquer valor. Retornar a+b se a>b senão a-b;
   - Criar dois *tensors* de *rank* 0, a e b, de qualquer valor aleatório entre -1 e 1. Retornar a+b se a<b; a-b se a>b; e 0 como *default*;
   - Criar um *tensor* do tipo variável, a, com o valor [[1, 2, 0], [3, 0, 2]], e um *tensor* de zeros, b, com o mesmo *shape* de a (*shape*=(2, 3)). Retornar um *tensor* booleano com o valor *True* para cada elemento de a igual a b;
   - Criar um *tensor* 1d, a, com 20 elementos compreendidos entre 1 e 10. Retornar um *tensor* com os elementos de a cujo valor é superior a 7.
2. Na segunda parte deste enunciado pretende-se que seja efectuado o treino de uma rede neuronal. Devem, para esse efeito, utilizar e completar o seguinte excerto de código (procurar pela tag ***TODO*** - 10 no total):

```
import tensorflow as tf
import matplotlib.pyplot as plt

#tensorflow version being used
print(tf.__version__)
#is tf executing eagerly?
print(TODO)

#load mnist training and test data
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
#data shape and cardinality
print('Train data shape', TODO)
print('Test data shape', TODO)
print('Number of training samples', TODO)
print('Number of testing samples', TODO)
#plotting some numbers!
for i in range(25):
   plt.subplot(5,5,i+1)   #Add a subplot as 5 x 5
   plt.xticks([])         #get rid of labels
   plt.yticks([])         #get rid of labels
   plt.imshow(x_test[i], cmap="gray")
plt.show()

#reshape the input to have a list of 784 (28*28) and normalize it (/255)
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1]*x_train.shape[2])
x_train = x_train.astype('float32')/255
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1]*x_test.shape[2])
x_test = x_test.astype('float32')/255

#building a three-layer sequential model
model = tf.keras.Sequential([
   tf.keras.layers.Dense(64, activation='relu'),
   tf.keras.layers.Dense(TODO),
   tf.keras.layers.Dense(10, activation='softmax')
])
#compiling the model
model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
#training it
model.fit(TODO)
#evaluating it
_, test_acc = model.evaluate(TODO)
print('\nTest accuracy:', test_acc)

#finally, generating predictions (the output of the last layer)
print('\nGenerating predictions for the first fifteen samples...')
predictions = model.predict(TODO)
print('Predictions shape:', predictions.shape)
for i, prediction in enumerate(predictions):
   #tf.argmax returns the INDEX with the largest value across axes of a tensor
   predicted_value = tf.argmax(prediction)
   label = TODO
   print('Predicted a %d. Real value is %d.' %(predicted_value, label))
```