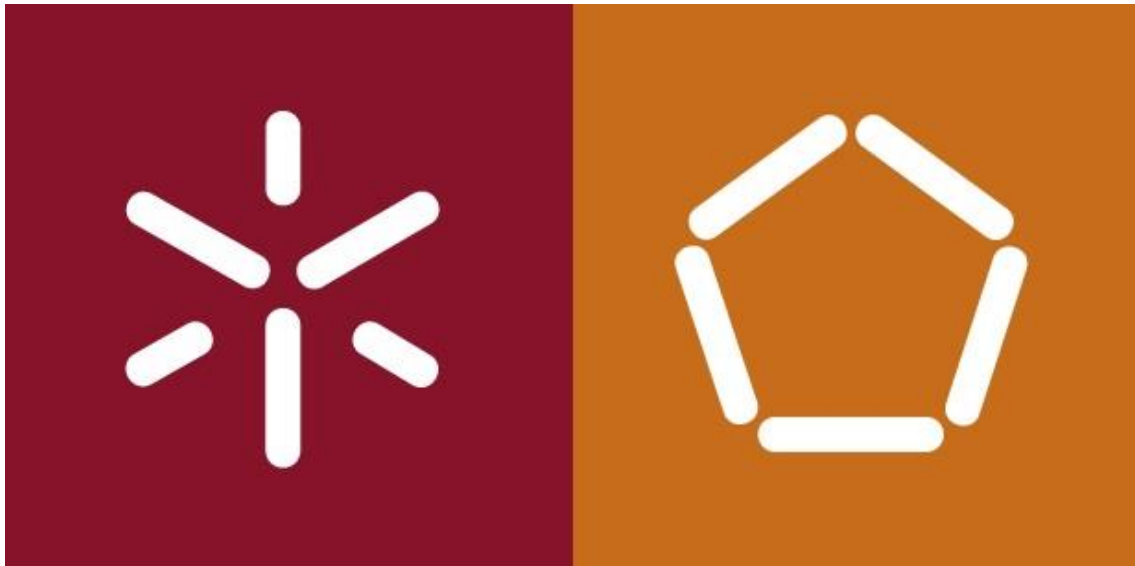


Universidade do Minho

Departamento de Informática
Mestrado integrado em Engenharia Informática



Perfil de Machine Learning: Fundamentos e Aplicações
Classificadores e Sistemas Conexionistas
Trabalho Prático nº 4

Ricardo Pereira (A77045)
12 e março de 2020


Parte 1


Nesta parte da ficha era pretendido que fosse adicionado ao script já realizado no trabalho prático anterior, aquando da execução de alto-nível, guardar ficheiros checkpoints a cada 5 épocas. A tarefa foi devidamente executada através da inclusão de callbacks. Pode ser verificada na imagem abaixo a sua utilização:


```
callbacks= [
    tf.keras.callbacks.ModelCheckpoint(
        filepath='./my_model_{epoch}_{val_loss:.3f}.hdf5', monitor='val_loss', verbose=1, save_best_only=True,
        save_weights_only=False, period=5
    )
]
history = mlp.fit(train_dataset, validation_data=validation_dataset, epochs=epochs, callbacks=callbacks)
```

Numa execução de 10 épocas foram guardados como seria pretendido dois checkpoints:

```
Epoch 1/10
2020-03-11 22:28:30.094371: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cublas64_10.dll
1719/1719 [=====] - 5s 3ms/step - loss: 0.5222 - sparse_categorical_accuracy: 0.8452 - val_loss: 0.2559 - val_sparse_categorical_a
ccuracy: 0.9290
Epoch 2/10
1719/1719 [=====] - 4s 2ms/step - loss: 0.2896 - sparse_categorical_accuracy: 0.9164 - val_loss: 0.2062 - val_sparse_categorical_a
ccuracy: 0.9408
Epoch 3/10
1719/1719 [=====] - 4s 2ms/step - loss: 0.2369 - sparse_categorical_accuracy: 0.9306 - val_loss: 0.1746 - val_sparse_categorical_a
ccuracy: 0.9516
Epoch 4/10
1719/1719 [=====] - 4s 2ms/step - loss: 0.2023 - sparse_categorical_accuracy: 0.9399 - val_loss: 0.1512 - val_sparse_categorical_a
ccuracy: 0.9580
Epoch 5/10
1708/1719 [=====>.] - ETA: 0s - loss: 0.1768 - sparse_categorical_accuracy: 0.9483
Epoch 00005: val_loss improved from inf to 0.15324, saving model to ./my_model_5_0.153.hdf5
1719/1719 [=====] - 4s 2ms/step - loss: 0.1765 - sparse_categorical_accuracy: 0.9484 - val_loss: 0.1532 - val_sparse_categorical_a
ccuracy: 0.9550
Epoch 6/10
1719/1719 [=====] - 4s 2ms/step - loss: 0.1604 - sparse_categorical_accuracy: 0.9526 - val_loss: 0.1436 - val_sparse_categorical_a
ccuracy: 0.9572
Epoch 7/10
1719/1719 [=====] - 4s 2ms/step - loss: 0.1487 - sparse_categorical_accuracy: 0.9555 - val_loss: 0.1357 - val_sparse_categorical_a
ccuracy: 0.9588
Epoch 8/10
1719/1719 [=====] - 4s 2ms/step - loss: 0.1375 - sparse_categorical_accuracy: 0.9596 - val_loss: 0.1325 - val_sparse_categorical_a
ccuracy: 0.9600
Epoch 9/10
1719/1719 [=====] - 4s 2ms/step - loss: 0.1293 - sparse_categorical_accuracy: 0.9618 - val_loss: 0.1302 - val_sparse_categorical_a
ccuracy: 0.9594
Epoch 10/10
1713/1719 [=====>.] - ETA: 0s - loss: 0.1219 - sparse_categorical_accuracy: 0.9640
Epoch 00010: val_loss improved from 0.15324 to 0.12671, saving model to ./my_model_10_0.127.hdf5
1719/1719 [=====] - 4s 2ms/step - loss: 0.1219 - sparse_categorical_accuracy: 0.9639 - val_loss: 0.1267 - val_sparse_categorical_a
ccuracy: 0.9632
```

 ex1.py

 my_model_5_0.153.hdf5

 my_model_10_0.127.hdf5

Parte 2

Nesta segunda parte era pretendido que fosse criado um agente capaz de jogar no ambiente CartPole-v1, disponibilizado pelo *Gym* da *OpenAI*. Não apenas jogar, mas sim criar um modelo que aprendesse com as observações que o ambiente retorna de forma a conseguir ultrapassar o objetivo de conseguir jogar 100 jogos e obter uma pontuação entre 195 e 200 pontos. Esta tarefa foi concluída com sucesso da seguinte forma:

Primeiramente, foi construído um dataset com o histórico de jogos feitos de forma aleatória em que só foram guardados scores acima de 120, com o objetivo de ter o melhor dataset possível. Podemos ver na primeira imagem abaixo os jogos que foram guardados no dataset.

De seguida, foi então construído um modelo, através da API Keras, sequencial com duas *hidden layers*, aplicando a função *relu* como função de activação. Para a otimização do modelo e sucessiva aproximação de melhores declives na procura por mínimos foi utilizada a função *Adam*. O modelo foi treinado durante 20 épocas.

Por fim, foi construído o agente que vai tomar cada ação com base nas probabilidades dadas pelo modelo em cada jogada.

[illegible]

Quanto ao exercício extra de criar um agente para jogar no ambiente Acrobot-v1, também foi realizado. Neste caso foi também construída uma MLP para prever as ações a tomar e um agente para jogar.

Todas as resoluções podem ser observadas nos ficheiros enviados com este relatório.