Assignment 10-4

May 20, 2021

[1]:
```
import tensorflow.compat.v1 as tf
import matplotlib.pyplot as plt
tf.disable_v2_behavior()
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.8/site-
packages/tensorflow/python/compat/v2_compat.py:96: disable_resource_variables
(from tensorflow.python.ops.variable_scope) is deprecated and will be removed in
a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

[2]:
```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop
from keras.datasets import imdb
from keras.preprocessing import sequence
from contextlib import redirect_stdout
from pathlib import Path
import time
start_time = time.time()
```

[3]:
```
results_dir = Path('results').joinpath('model_1')
results_dir.mkdir(parents=True, exist_ok=True)
```

[4]:
```
max_features = 10000
max_len = 500

print('Loading data ...')

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen = max_len)
x_test = sequence.pad_sequences(x_test, maxlen = max_len)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
```

```
Loading data …
```

```
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an ndarray
from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or
ndarrays with different lengths or shapes) is deprecated. If you meant to do
this, you must specify 'dtype=object' when creating the ndarray
/opt/conda/lib/python3.8/site-
packages/tensorflow/python/keras/datasets/imdb.py:159:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
/opt/conda/lib/python3.8/site-
packages/tensorflow/python/keras/datasets/imdb.py:160:
VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences
(which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify
'dtype=object' when creating the ndarray
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])
```

```
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 500)
x_test shape: (25000, 500)
```

[5]:
```
model = Sequential()
model.add(layers.Embedding(max_features, 128, input_length=max_len))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))
model.summary()
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.8/site-
packages/tensorflow/python/keras/initializers/initializers_v1.py:58: calling
RandomUniform.__init__ (from tensorflow.python.ops.init_ops) with dtype is
deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the
constructor
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 500, 128)          1280000
_________________________________________________________________
```

```
conv1d (Conv1D)              (None, 494, 32)           28704
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 98, 32)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 92, 32)            7200
_________________________________________________________________
global_max_pooling1d (Global (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 33
=================================================================
Total params: 1,315,937
Trainable params: 1,315,937
Non-trainable params: 0
_________________________________________________________________
```

[6]:
```
model.compile(optimizer=RMSprop(lr=1e-4), loss='binary_crossentropy',␣
 ↪metrics=['acc'])
history = model.fit(x_train, y_train, epochs=10, batch_size=128,␣
 ↪validation_split=0.2)
```

```
Train on 20000 samples, validate on 5000 samples
Epoch 1/10
19968/20000 [============================>.] - ETA: 0s - loss: 0.7192 - acc:
0.5265
```

```
/opt/conda/lib/python3.8/site-
packages/tensorflow/python/keras/engine/training.py:2325: UserWarning:
`Model.state_updates` will be removed in a future version. This property should
not be used in TensorFlow 2.0, as `updates` are applied automatically.
  warnings.warn('`Model.state_updates` will be removed in a future version. '
```

```
20000/20000 [==============================] - 11s 545us/sample - loss: 0.7191 -
acc: 0.5267 - val_loss: 0.6813 - val_acc: 0.6076
Epoch 2/10
20000/20000 [==============================] - 10s 516us/sample - loss: 0.6571 -
acc: 0.6934 - val_loss: 0.6497 - val_acc: 0.6644
Epoch 3/10
20000/20000 [==============================] - 10s 490us/sample - loss: 0.5922 -
acc: 0.7861 - val_loss: 0.5602 - val_acc: 0.7694
Epoch 4/10
20000/20000 [==============================] - 10s 492us/sample - loss: 0.4638 -
acc: 0.8303 - val_loss: 0.4301 - val_acc: 0.8296
Epoch 5/10
20000/20000 [==============================] - 10s 490us/sample - loss: 0.3657 -
acc: 0.8666 - val_loss: 0.3971 - val_acc: 0.8468
Epoch 6/10
20000/20000 [==============================] - 10s 487us/sample - loss: 0.3158 -
acc: 0.8889 - val_loss: 0.4065 - val_acc: 0.8538
Epoch 7/10
```

```
20000/20000 [==============================] - 10s 484us/sample - loss: 0.2804 -
acc: 0.9029 - val_loss: 0.4135 - val_acc: 0.8638
Epoch 8/10
20000/20000 [==============================] - 10s 494us/sample - loss: 0.2506 -
acc: 0.9148 - val_loss: 0.4193 - val_acc: 0.8674
Epoch 9/10
20000/20000 [==============================] - 10s 491us/sample - loss: 0.2280 -
acc: 0.9243 - val_loss: 0.4467 - val_acc: 0.8728
Epoch 10/10
20000/20000 [==============================] - 10s 485us/sample - loss: 0.2051 -
acc: 0.9350 - val_loss: 0.4527 - val_acc: 0.8744
```

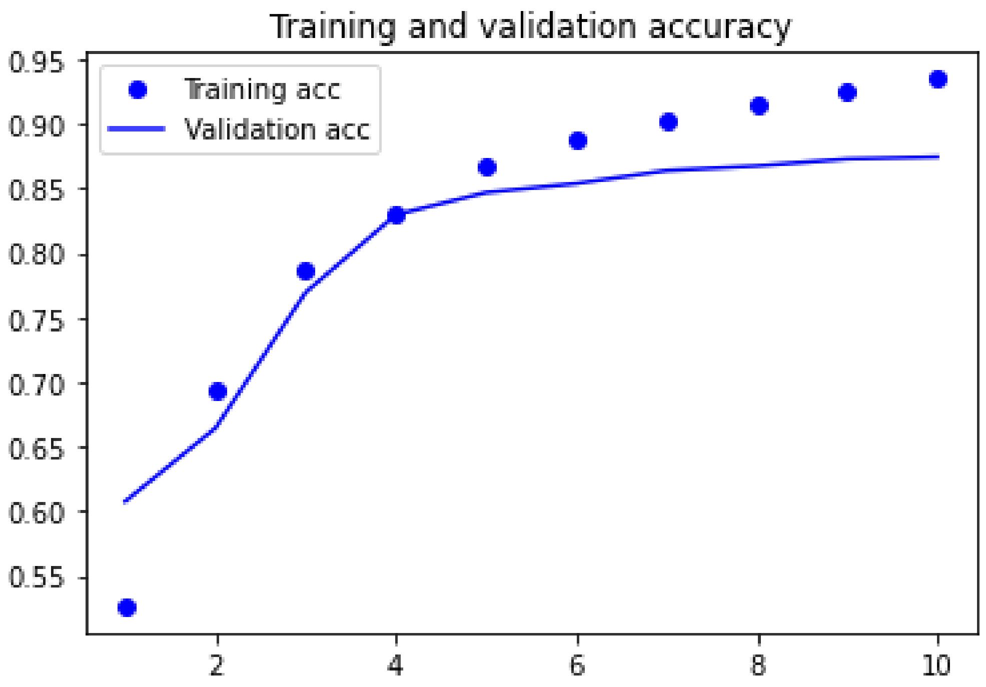
[7]:
```python
# Save the summary to file
summary_file = results_dir.joinpath('Assignment_10.4_ModelSummary.txt')
with open(summary_file, 'w') as f:
    with redirect_stdout(f):
        model.summary()
```

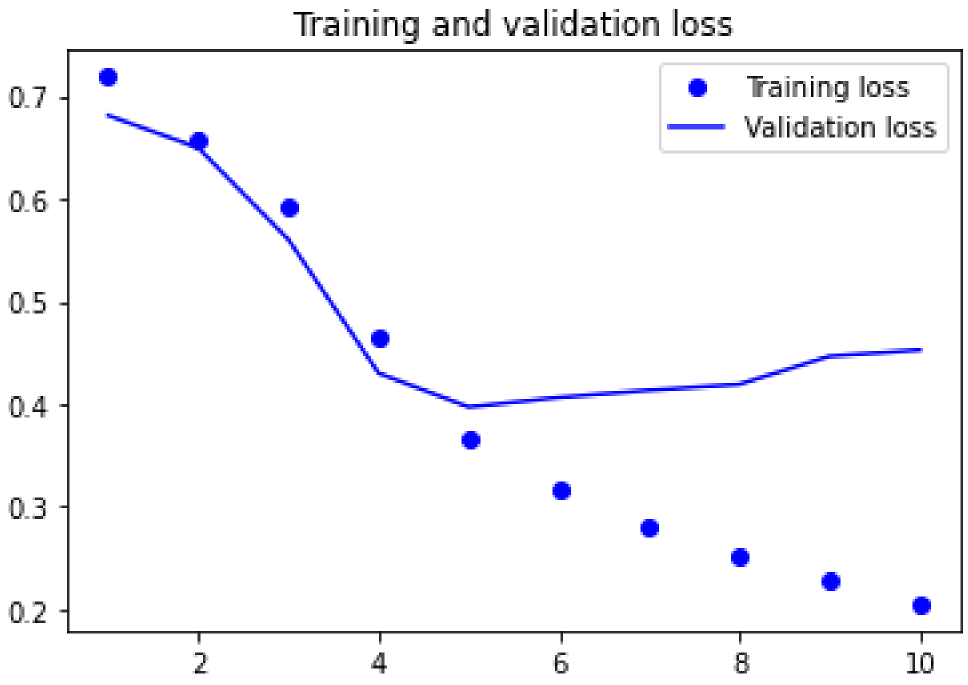
[8]:
```python
result_model_file = results_dir.joinpath('pre_trained_glove_model_1D_Convnet.
 ↪h5')
model.save_weights(result_model_file)
```

[9]:
```python
# Place plot here
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
img_file = results_dir.joinpath('Assignment_10.4_Model Accuracy Validation.png')
plt.savefig(img_file)
plt.show()
```





[10]:
```
#save the model performance metrics and training and validation accuracy curves␣
↪in the results/model_2 direc
```

```
model.load_weights(result_model_file)
eval = model.evaluate(x_test, y_test)
print("")
print(eval)
```

```
[0.4816769612121582, 0.8662]
```

```
[ ]:
```