Assignment 5.1

October 5, 2020

1 Assignment 5.1

Brandon Sams 30Sep2020

1.1 Loading the IMDB dataset

[1]:
```
from keras.datasets import imdb
```

[2]:
```
(train_data, train_labels), (test_data, test_labels) = imdb.
↪load_data(num_words=10000)
```

[3]:
```
train_data[0]
```

[3]:
```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
 25,
 100,
 43,
 838,
 112,
 50,
```

670,
2,
9,
35,
480,
284,
5,
150,
4,
172,
112,
167,
2,
336,
385,
39,
4,
172,
4536,
1111,
17,
546,
38,
13,
447,
4,
192,
50,
16,
6,
147,
2025,
19,
14,
22,
4,
1920,
4613,
469,
4,
22,
71,
87,
12,
16,
43,
530,

```
38,
76,
15,
13,
1247,
4,
22,
17,
515,
17,
12,
16,
626,
18,
2,
5,
62,
386,
12,
8,
316,
8,
106,
5,
4,
2223,
5244,
16,
480,
66,
3785,
33,
4,
130,
12,
16,
38,
619,
5,
25,
124,
51,
36,
135,
48,
25,
1415,
```

```
33,
6,
22,
12,
215,
28,
77,
52,
5,
14,
407,
16,
82,
2,
8,
4,
107,
117,
5952,
15,
256,
4,
2,
7,
3766,
5,
723,
36,
71,
43,
530,
476,
26,
400,
317,
46,
7,
4,
2,
1029,
13,
104,
88,
4,
381,
15,
297,
```

```
98,
32,
2071,
56,
26,
141,
6,
194,
7486,
18,
4,
226,
22,
21,
134,
476,
26,
480,
5,
144,
30,
5535,
18,
51,
36,
28,
224,
92,
25,
104,
4,
226,
65,
16,
38,
1334,
88,
12,
16,
283,
5,
16,
4472,
113,
103,
32,
15,
```

```
 16,
 5345,
 19,
 178,
 32]
```

[4]:
```
train_labels[0]
```

[4]:
```
1
```

[5]:
```
max([max(sequence) for sequence in train_data])
```

[5]:
```
9999
```

[6]:
```
word_index = imdb.get_word_index()
```

[7]:
```
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
```

[8]:
```
decoded_review = ' '.join([reverse_word_index.get(i - 3, '?') for i in␣
↪train_data[0]])
```

[9]:
```
decoded_review
```

[9]:
```
"? this film was just brilliant casting location scenery story direction
everyone's really suited the part they played and you could just imagine being
there robert ? is an amazing actor and now the same being director ? father came
from the same scottish island as myself so i loved the fact there was a real
connection with this film the witty remarks throughout the film were great it
was just brilliant so much that i bought the film as soon as it was released for
? and would recommend it to everyone to watch and the fly fishing was amazing
really cried at the end it was so sad and you know what they say if you cry at a
film it must have been good and this definitely was also ? to the two little
boy's that played the ? of norman and paul they were just brilliant children are
often left out of the ? list i think because the stars that play them all grown
up are such a big profile for the whole film but these children are amazing and
should be praised for what they have done don't you think the whole story was so
lovely because it was true and was someone's life after all that was shared with
us all"
```

1.2 Encoding the integer sequences into a binary matrix

[10]:
```
import numpy as np
```

[11]:
```
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
```

```
    return(results)
```

[12]:
```
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

[13]:
```
x_train[0]
```

[13]:
```
array([0., 1., 1., …, 0., 0., 0.])
```

[14]:
```
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

1.3 The model definition

[15]:
```
from keras import models
from keras import layers
```

[16]:
```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu',input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

1.4 Compiling the model

[17]:
```
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

1.5 Configuring the optimizer

[18]:
```
from keras import optimizers
```

[19]:
```
model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

1.6 Using custom losses and metrics

[20]:
```
from keras import losses
from keras import metrics
```

[21]:
```
model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss=losses.binary_crossentropy,
              metrics=[metrics.binary_accuracy])
```

1.7 Setting aside a validation set

[22]:
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

1.8 Training your model

[23]:
```
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
30/30 [==============================] - 1s 40ms/step - loss: 0.5050 - acc:
0.7810 - val_loss: 0.3721 - val_acc: 0.8702
Epoch 2/20
30/30 [==============================] - 1s 33ms/step - loss: 0.2911 - acc:
0.9088 - val_loss: 0.3203 - val_acc: 0.8732
Epoch 3/20
30/30 [==============================] - 1s 32ms/step - loss: 0.2154 - acc:
0.9283 - val_loss: 0.3325 - val_acc: 0.8629
Epoch 4/20
30/30 [==============================] - 1s 31ms/step - loss: 0.1694 - acc:
0.9441 - val_loss: 0.2781 - val_acc: 0.8891
Epoch 5/20
30/30 [==============================] - 1s 31ms/step - loss: 0.1366 - acc:
0.9573 - val_loss: 0.2950 - val_acc: 0.8859
Epoch 6/20
30/30 [==============================] - 1s 33ms/step - loss: 0.1155 - acc:
0.9624 - val_loss: 0.3003 - val_acc: 0.8858
Epoch 7/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0950 - acc:
0.9721 - val_loss: 0.3289 - val_acc: 0.8831
Epoch 8/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0801 - acc:
0.9771 - val_loss: 0.3582 - val_acc: 0.8730
Epoch 9/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0649 - acc:
0.9836 - val_loss: 0.3752 - val_acc: 0.8730
Epoch 10/20
```

```
30/30 [==============================] - 1s 32ms/step - loss: 0.0535 - acc:
0.9867 - val_loss: 0.3879 - val_acc: 0.8764
Epoch 11/20
30/30 [==============================] - 1s 32ms/step - loss: 0.0443 - acc:
0.9895 - val_loss: 0.4144 - val_acc: 0.8755
Epoch 12/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0344 - acc:
0.9929 - val_loss: 0.4489 - val_acc: 0.8754
Epoch 13/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0289 - acc:
0.9938 - val_loss: 0.4786 - val_acc: 0.8719
Epoch 14/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0231 - acc:
0.9955 - val_loss: 0.5092 - val_acc: 0.8705
Epoch 15/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0194 - acc:
0.9957 - val_loss: 0.5397 - val_acc: 0.8684
Epoch 16/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0122 - acc:
0.9987 - val_loss: 0.5831 - val_acc: 0.8698
Epoch 17/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0146 - acc:
0.9969 - val_loss: 0.6049 - val_acc: 0.8673
Epoch 18/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0066 - acc:
0.9996 - val_loss: 0.6371 - val_acc: 0.8664
Epoch 19/20
30/30 [==============================] - 1s 28ms/step - loss: 0.0075 - acc:
0.9988 - val_loss: 0.6773 - val_acc: 0.8660
Epoch 20/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0059 - acc:
0.9995 - val_loss: 0.7344 - val_acc: 0.8596
```

[24]:
```
history_dict = history.history
```

[25]:
```
history_dict.keys()
```

[25]:
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

1.9 Plotting the training and validation loss

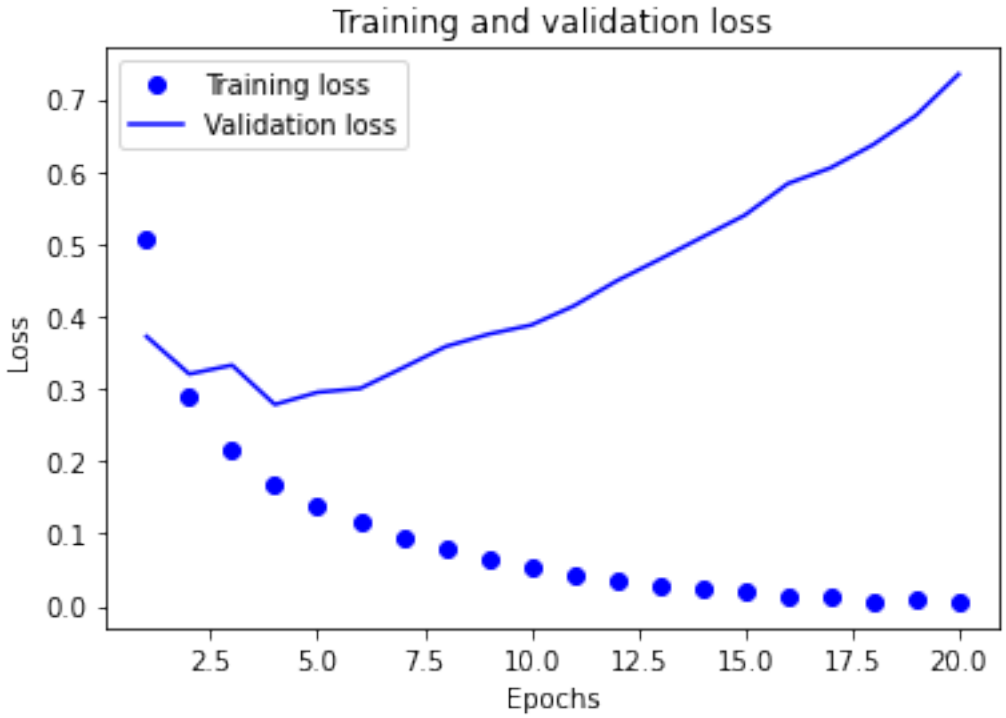
[26]:
```
import matplotlib.pyplot as plt
```

[27]:
```
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

epochs = range(1, len(loss_values) + 1)
```

```
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



1.10 Retraining a model from scratch

[28]:
```
model  = models.Sequential()
model.add(layers.Dense(16, activation='relu',input_shape=(10000,)))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=4, batch_size=512)
results = model.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 [==============================] - 0s 10ms/step - loss: 0.4354 - accuracy:
0.8252
Epoch 2/4
49/49 [==============================] - 0s 9ms/step - loss: 0.2660 - accuracy:
0.9101
Epoch 3/4
49/49 [==============================] - 0s 8ms/step - loss: 0.2117 - accuracy:
0.9273
Epoch 4/4
49/49 [==============================] - 0s 7ms/step - loss: 0.1801 - accuracy:
0.9392
782/782 [==============================] - 1s 2ms/step - loss: 0.2807 -
accuracy: 0.8876
```

[29]:
```
results
```

[29]:
```
[0.2807336449623108, 0.8875600099563599]
```

[30]:
```
model.predict(x_test)
```

[30]:
```
array([[0.25121224],
       [0.9999241 ],
       [0.7814768 ],
       …,
       [0.14182904],
       [0.09279433],
       [0.594135  ]], dtype=float32)
```