Assignment 5.1

October 3, 2022

[1]:
```
# Load IMDB dataset

from keras.datasets import imdb
(train_data, train_labels), (test_data, test_labels) = imdb.
 ↪load_data(num_words=10000)
```

[2]:
```
train_data[0]
```

[2]:
```
[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]
```

[3]:
```
train_labels[0]
```

[3]:
```
1
```

[4]:
```
# Encoding the integer sequences into a binary matrix

import numpy as np
def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
        return results
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

[5]:
```
# Sample Data
x_train[0]
```

[5]:
```
array([0., 1., 1., …, 0., 0., 0.])
```

[6]:
```
# vectorize your labels
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

[7]:
```
# The model definition
from keras import models
from keras import layers
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'))
```

[8]:
```
# Compiling the model

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

[9]:
```
# Configuring the optimizer

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

[10]:
```
# Using custom losses and metrics

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

[11]:
```
# Setting aside a validation set

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

[12]:
```
# Training your model

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: 0.6932 - acc:
0.4965 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 2/20
30/30 [==============================] - 1s 29ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 3/20
30/30 [==============================] - 1s 28ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 4/20
30/30 [==============================] - 1s 34ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 5/20
30/30 [==============================] - 1s 26ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 6/20
30/30 [==============================] - 1s 26ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 7/20
30/30 [==============================] - 1s 27ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 8/20
30/30 [==============================] - 1s 27ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 9/20
```

```
30/30 [==============================] - 1s 26ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 10/20
30/30 [==============================] - 1s 27ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 11/20
30/30 [==============================] - 1s 27ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 12/20
30/30 [==============================] - 1s 29ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 13/20
30/30 [==============================] - 1s 26ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 14/20
30/30 [==============================] - 1s 27ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 15/20
30/30 [==============================] - 1s 31ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 16/20
30/30 [==============================] - 1s 33ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 17/20
30/30 [==============================] - 1s 28ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6932 - val_acc: 0.4947
Epoch 18/20
30/30 [==============================] - 1s 29ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 19/20
30/30 [==============================] - 1s 39ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
Epoch 20/20
30/30 [==============================] - 1s 33ms/step - loss: 0.6931 - acc:
0.5035 - val_loss: 0.6933 - val_acc: 0.4947
```

[13]:
```
history_dict = history.history
history_dict.keys()
```

[13]:
```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

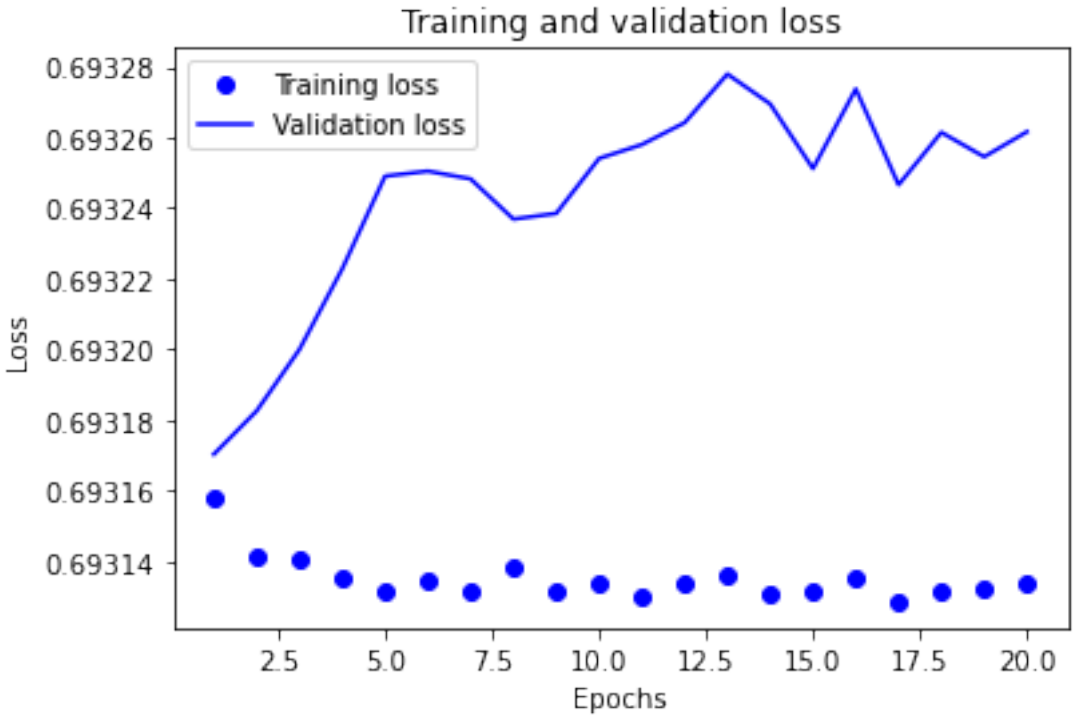
[14]:
```
# Plot the training and validation LOSS

import matplotlib.pyplot as plt

history_dict = history.history
loss_values = history_dict["loss"]
```

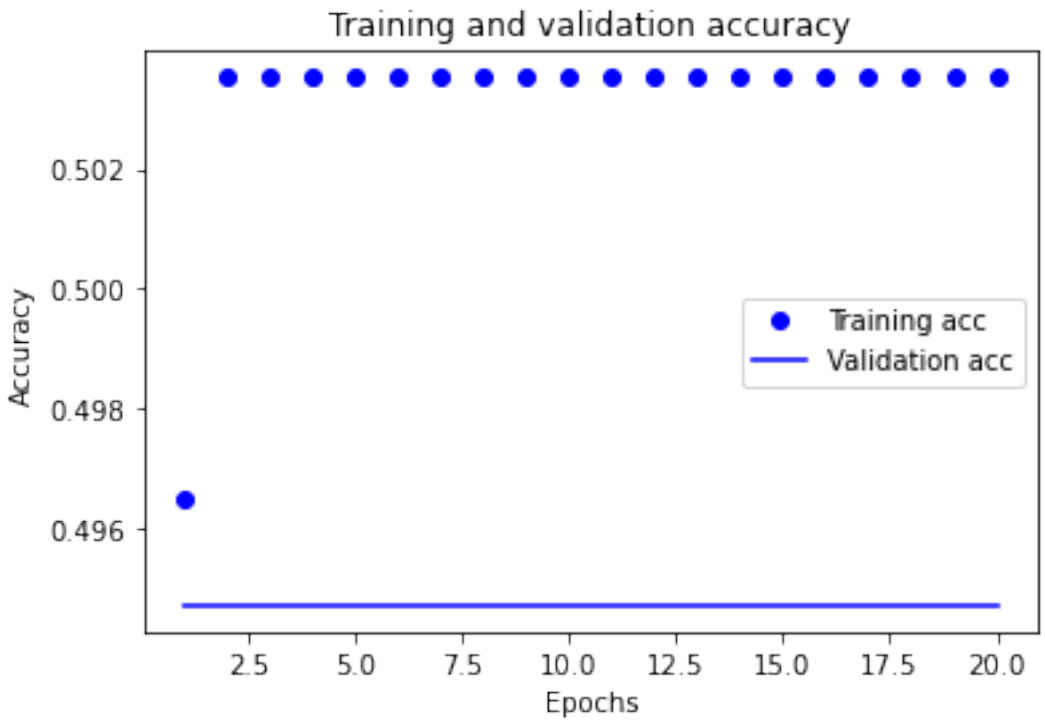
```
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)

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



[15]:
```
# Plot the training and validation ACCURACY

plt.clf()  # clear the figure
acc = history_dict["acc"]
val_acc = history_dict["val_acc"]
plt.plot(epochs, acc, "bo", label="Training acc")
plt.plot(epochs, val_acc, "b", label="Validation acc")
plt.title("Training and validation accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



[16]:
```
# Retraining a model from scratch

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'))
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: 0.6932 - accuracy:
0.4977
Epoch 2/4
49/49 [==============================] - 1s 10ms/step - loss: 0.6932 - accuracy:
0.4970
Epoch 3/4
49/49 [==============================] - 0s 9ms/step - loss: 0.6931 - accuracy:
0.4976
Epoch 4/4
49/49 [==============================] - 0s 8ms/step - loss: 0.6932 - accuracy:
0.4940
```

```
782/782 [==============================] - 2s 2ms/step - loss: 0.6932 -
accuracy: 0.5000
```

[17]:
```
results
```

[17]:
```
[0.6931501626968384, 0.4999600052833557]
```

[18]:
```
model.predict(x_test)
```

[18]:
```
array([[0.54178935],
       [0.4997027 ],
       [0.4997027 ],
       …,
       [0.4997027 ],
       [0.4997027 ],
       [0.4997027 ]], dtype=float32)
```

[ ]:
```

```