Madhukar_Ayachit_5.2

April 26, 2022

0.1 Assignment 5.2

[1]:
```
import keras
keras.__version__
```

[1]:
```
'2.4.3'
```

[2]:
```
from keras.datasets import reuters
(train_data, train_labels), (test_data, test_labels) = reuters.
 ↪load_data(num_words=10000)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/reuters.npz
2113536/2110848 [==============================] - 0s 0us/step
```

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

[3]:
```
[1,
 2,
 2,
 8,
 43,
 10,
 447,
 5,
 25,
 207,
 270,
 5,
 3095,
 111,
 16,
 369,
 186,
 90,
 67,
 7,
 89,
```

5,
19,
102,
6,
19,
124,
15,
90,
67,
84,
22,
482,
26,
7,
48,
4,
49,
8,
864,
39,
209,
154,
6,
151,
6,
83,
11,
15,
22,
155,
11,
15,
7,
48,
9,
4579,
1005,
504,
6,
258,
6,
272,
11,
15,
22,
134,
44,

```
 11,
 15,
 16,
 8,
 197,
 1245,
 90,
 67,
 52,
 29,
 209,
 30,
 32,
 132,
 6,
 109,
 15,
 17,
 12]
```

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

[4]:
```
3
```

[5]:
```
word_index = reuters.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.
 ↪items()]) # Note that our indices were offset by 32
# because 0, 1 and 2 are reserved indices for "padding", "start of sequence",␣␣
 ↪ ↪and "unknown".
decoded_newswire = ' '.join([reverse_word_index.get(i - 3, '?') for i in␣
 ↪train_data[0]])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/reuters_word_index.json
557056/550378 [==============================] - 0s 0us/step
```

[7]:
```
decoded_newswire
```

[7]:
```
'? ? ? said as a result of its december acquisition of space co it expects
earnings per share in 1987 of 1 15 to 1 30 dlrs per share up from 70 cts in 1986
the company said pretax net should rise to nine to 10 mln dlrs from six mln dlrs
in 1986 and rental operation revenues to 19 to 22 mln dlrs from 12 5 mln dlrs it
said cash flow per share this year should be 2 50 to three dlrs reuter 3'
```

[8]:
```
train_labels[10]
```

[8]:
```
3
```

[9]:
```
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
# Our vectorized training data
x_train = vectorize_sequences(train_data) # Our vectorized test data
x_test = vectorize_sequences(test_data)
```

[10]:
```
def to_one_hot(labels, dimension=46):
    results = np.zeros((len(labels), dimension))
    for i, label in enumerate(labels):
        results[i, label] = 1.
        return results
# Our vectorized training labels
one_hot_train_labels = to_one_hot(train_labels) # Our vectorized test labels
one_hot_test_labels = to_one_hot(test_labels)
```

[11]:
```
from keras.utils.np_utils import to_categorical
one_hot_train_labels = to_categorical(train_labels)
one_hot_test_labels = to_categorical(test_labels)
```

[12]:
```
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
```

[13]:
```
model.
 ↪compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
```

[14]:
```
x_val = x_train[:1000]
partial_x_train = x_train[1000:]
y_val = one_hot_train_labels[:1000]
partial_y_train = one_hot_train_labels[1000:]
```

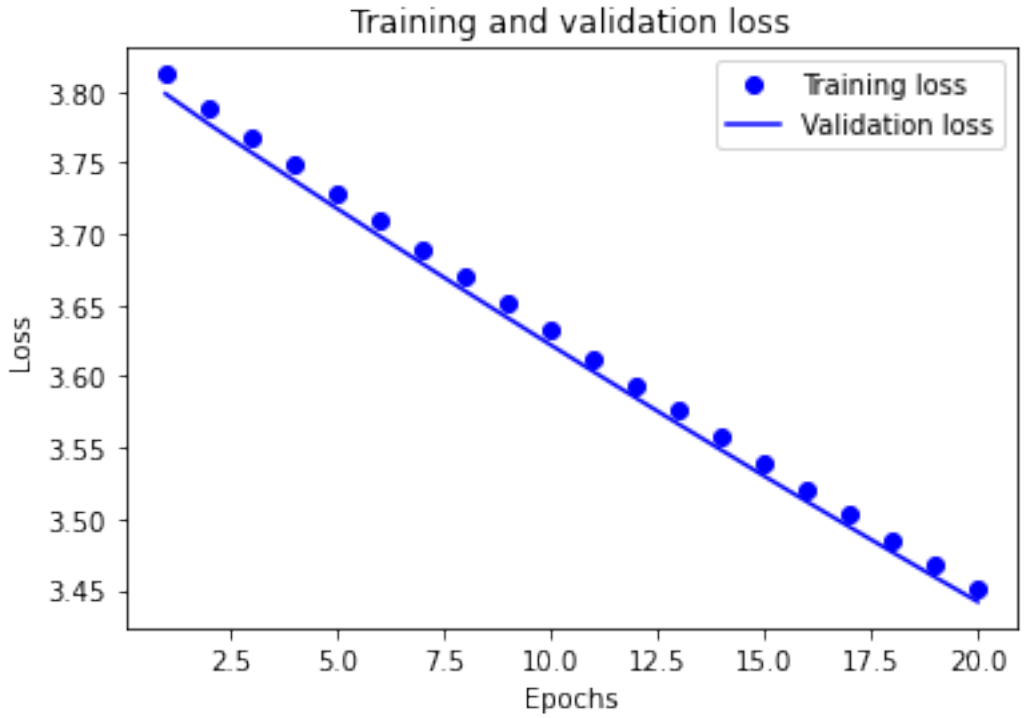
[15]:
```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
16/16 [==============================] - 1s 38ms/step - loss: 3.8120 - accuracy:
0.1750 - val_loss: 3.7981 - val_accuracy: 0.0460
```

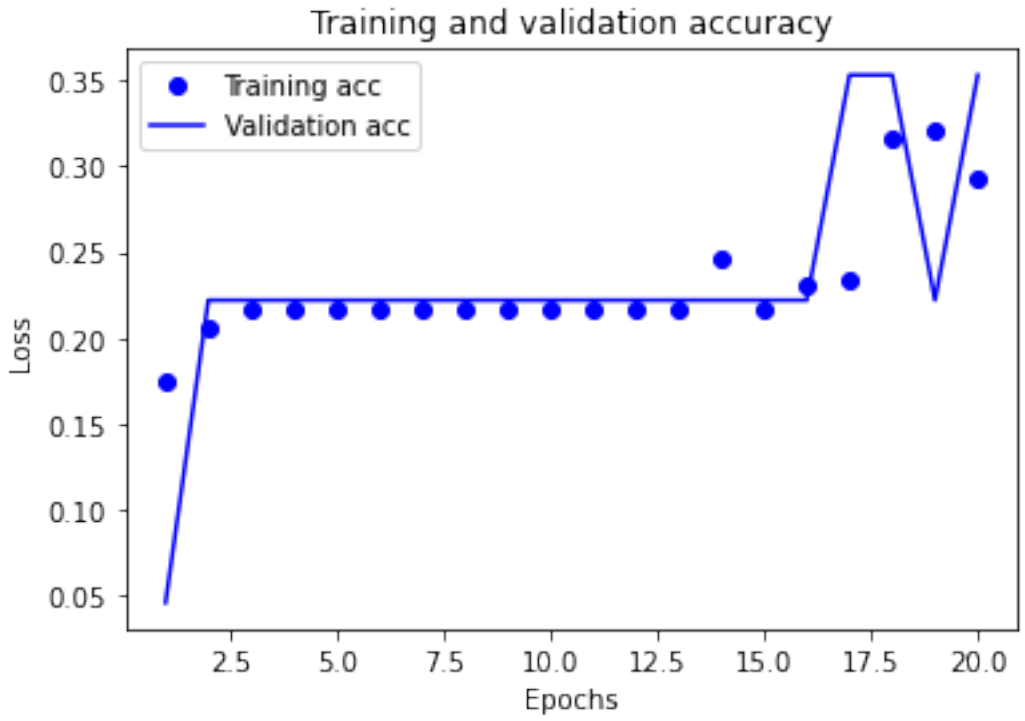
```
Epoch 2/20
16/16 [==============================] - 0s 20ms/step - loss: 3.7883 - accuracy:
0.2057 - val_loss: 3.7772 - val_accuracy: 0.2220
Epoch 3/20
16/16 [==============================] - 0s 18ms/step - loss: 3.7678 - accuracy:
0.2164 - val_loss: 3.7572 - val_accuracy: 0.2220
Epoch 4/20
16/16 [==============================] - 0s 17ms/step - loss: 3.7479 - accuracy:
0.2164 - val_loss: 3.7375 - val_accuracy: 0.2220
Epoch 5/20
16/16 [==============================] - 0s 17ms/step - loss: 3.7282 - accuracy:
0.2164 - val_loss: 3.7179 - val_accuracy: 0.2220
Epoch 6/20
16/16 [==============================] - 0s 16ms/step - loss: 3.7085 - accuracy:
0.2164 - val_loss: 3.6985 - val_accuracy: 0.2220
Epoch 7/20
16/16 [==============================] - 0s 20ms/step - loss: 3.6891 - accuracy:
0.2164 - val_loss: 3.6792 - val_accuracy: 0.2220
Epoch 8/20
16/16 [==============================] - 0s 16ms/step - loss: 3.6698 - accuracy:
0.2164 - val_loss: 3.6599 - val_accuracy: 0.2220
Epoch 9/20
16/16 [==============================] - 0s 19ms/step - loss: 3.6506 - accuracy:
0.2164 - val_loss: 3.6410 - val_accuracy: 0.2220
Epoch 10/20
16/16 [==============================] - 0s 16ms/step - loss: 3.6317 - accuracy:
0.2164 - val_loss: 3.6222 - val_accuracy: 0.2220
Epoch 11/20
16/16 [==============================] - 0s 16ms/step - loss: 3.6129 - accuracy:
0.2164 - val_loss: 3.6034 - val_accuracy: 0.2220
Epoch 12/20
16/16 [==============================] - 0s 16ms/step - loss: 3.5942 - accuracy:
0.2164 - val_loss: 3.5849 - val_accuracy: 0.2220
Epoch 13/20
16/16 [==============================] - 0s 17ms/step - loss: 3.5756 - accuracy:
0.2164 - val_loss: 3.5664 - val_accuracy: 0.2220
Epoch 14/20
16/16 [==============================] - 0s 15ms/step - loss: 3.5572 - accuracy:
0.2464 - val_loss: 3.5482 - val_accuracy: 0.2220
Epoch 15/20
16/16 [==============================] - 0s 20ms/step - loss: 3.5390 - accuracy:
0.2164 - val_loss: 3.5300 - val_accuracy: 0.2220
Epoch 16/20
16/16 [==============================] - 1s 39ms/step - loss: 3.5209 - accuracy:
0.2308 - val_loss: 3.5121 - val_accuracy: 0.2220
Epoch 17/20
16/16 [==============================] - 0s 27ms/step - loss: 3.5030 - accuracy:
0.2343 - val_loss: 3.4942 - val_accuracy: 0.3530
```

```
Epoch 18/20
16/16 [==============================] - 0s 21ms/step - loss: 3.4852 - accuracy:
0.3155 - val_loss: 3.4766 - val_accuracy: 0.3530
Epoch 19/20
16/16 [==============================] - 0s 16ms/step - loss: 3.4676 - accuracy:
0.3205 - val_loss: 3.4590 - val_accuracy: 0.2220
Epoch 20/20
16/16 [==============================] - 0s 14ms/step - loss: 3.4502 - accuracy:
0.2933 - val_loss: 3.4417 - val_accuracy: 0.3530
```

[16]:
```
import matplotlib.pyplot as plt
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



[17]:
```
plt.clf() # clear figure
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
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('Loss')
plt.legend()
plt.show()
```



[18]:
```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(partial_x_train,
              partial_y_train,
              epochs=8,
              batch_size=512,
              validation_data=(x_val, y_val))
```

```
results = model.evaluate(x_test, one_hot_test_labels)
```

```
Epoch 1/8
16/16 [==============================] - 0s 27ms/step - loss: 3.8123 - accuracy:
0.0791 - val_loss: 3.7983 - val_accuracy: 0.0590
Epoch 2/8
16/16 [==============================] - 0s 17ms/step - loss: 3.7885 - accuracy:
0.0995 - val_loss: 3.7773 - val_accuracy: 0.0590
Epoch 3/8
16/16 [==============================] - 0s 17ms/step - loss: 3.7680 - accuracy:
0.3140 - val_loss: 3.7573 - val_accuracy: 0.3530
Epoch 4/8
16/16 [==============================] - 0s 15ms/step - loss: 3.7481 - accuracy:
0.3413 - val_loss: 3.7377 - val_accuracy: 0.3530
Epoch 5/8
16/16 [==============================] - 0s 17ms/step - loss: 3.7284 - accuracy:
0.3314 - val_loss: 3.7181 - val_accuracy: 0.3530
Epoch 6/8
16/16 [==============================] - 0s 15ms/step - loss: 3.7088 - accuracy:
0.3435 - val_loss: 3.6986 - val_accuracy: 0.3530
Epoch 7/8
16/16 [==============================] - 0s 16ms/step - loss: 3.6894 - accuracy:
0.3514 - val_loss: 3.6793 - val_accuracy: 0.3530
Epoch 8/8
16/16 [==============================] - 0s 15ms/step - loss: 3.6701 - accuracy:
0.3514 - val_loss: 3.6603 - val_accuracy: 0.3530
71/71 [==============================] - 0s 2ms/step - loss: 3.6623 - accuracy:
0.3620
```

[19]:
```
results
```

[19]:
```
[3.6623315811157227, 0.36197686195373535]
```

[21]:
```
import copy
test_labels_copy = copy.copy(test_labels)
np.random.shuffle(test_labels_copy)
float(np.sum(np.array(test_labels) == np.array(test_labels_copy))) /␣
 ↪len(test_labels)
```

[21]:
```
0.18432769367764915
```

[22]:
```
predictions = model.predict(x_test)
```

[23]:
```
predictions[0].shape
```

[23]:
```
(46,)
```

[24]:
```
np.sum(predictions[0])
```

[24]:
```
0.99999994
```

[25]:
```
np.argmax(predictions[0])
```

[25]:
```
3
```

[26]:
```
y_train = np.array(train_labels)
y_test = np.array(test_labels)
```

[27]:
```
model.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy',␣
↪metrics=['acc'])
```

[28]:
```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(4, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(partial_x_train,
          partial_y_train,
          epochs=20,
          batch_size=128,
          validation_data=(x_val, y_val))
```

```
Epoch 1/20
63/63 [==============================] - 1s 12ms/step - loss: 3.7821 - accuracy:
0.1845 - val_loss: 3.7420 - val_accuracy: 0.2230
Epoch 2/20
63/63 [==============================] - 1s 9ms/step - loss: 3.7060 - accuracy:
0.2539 - val_loss: 3.6682 - val_accuracy: 0.3540
Epoch 3/20
63/63 [==============================] - 0s 7ms/step - loss: 3.6329 - accuracy:
0.3457 - val_loss: 3.5962 - val_accuracy: 0.3540
Epoch 4/20
63/63 [==============================] - 0s 8ms/step - loss: 3.5624 - accuracy:
0.3443 - val_loss: 3.5268 - val_accuracy: 0.3540
Epoch 5/20
63/63 [==============================] - 0s 8ms/step - loss: 3.4938 - accuracy:
0.3514 - val_loss: 3.4592 - val_accuracy: 0.3540
Epoch 6/20
63/63 [==============================] - 1s 8ms/step - loss: 3.4275 - accuracy:
0.3514 - val_loss: 3.3941 - val_accuracy: 0.3540
Epoch 7/20
63/63 [==============================] - 0s 8ms/step - loss: 3.3636 - accuracy:
```

```
0.3514 - val_loss: 3.3311 - val_accuracy: 0.3540
Epoch 8/20
63/63 [==============================] - 0s 8ms/step - loss: 3.3020 - accuracy:
0.3514 - val_loss: 3.2704 - val_accuracy: 0.3540
Epoch 9/20
63/63 [==============================] - 0s 8ms/step - loss: 3.2421 - accuracy:
0.3514 - val_loss: 3.2112 - val_accuracy: 0.3540
Epoch 10/20
63/63 [==============================] - 1s 8ms/step - loss: 3.1845 - accuracy:
0.3514 - val_loss: 3.1543 - val_accuracy: 0.3540
Epoch 11/20
63/63 [==============================] - 1s 9ms/step - loss: 3.1291 - accuracy:
0.3514 - val_loss: 3.0997 - val_accuracy: 0.3540
Epoch 12/20
63/63 [==============================] - 1s 10ms/step - loss: 3.0759 - accuracy:
0.3514 - val_loss: 3.0474 - val_accuracy: 0.3540
Epoch 13/20
63/63 [==============================] - 0s 7ms/step - loss: 3.0248 - accuracy:
0.3514 - val_loss: 2.9966 - val_accuracy: 0.3540
Epoch 14/20
63/63 [==============================] - 0s 7ms/step - loss: 2.9759 - accuracy:
0.3514 - val_loss: 2.9484 - val_accuracy: 0.3540
Epoch 15/20
63/63 [==============================] - 0s 7ms/step - loss: 2.9288 - accuracy:
0.3514 - val_loss: 2.9017 - val_accuracy: 0.3540
Epoch 16/20
63/63 [==============================] - 0s 7ms/step - loss: 2.8838 - accuracy:
0.3514 - val_loss: 2.8574 - val_accuracy: 0.3540
Epoch 17/20
63/63 [==============================] - 1s 8ms/step - loss: 2.8408 - accuracy:
0.3514 - val_loss: 2.8148 - val_accuracy: 0.3540
Epoch 18/20
63/63 [==============================] - 0s 7ms/step - loss: 2.8001 - accuracy:
0.3514 - val_loss: 2.7746 - val_accuracy: 0.3540
Epoch 19/20
63/63 [==============================] - 1s 10ms/step - loss: 2.7617 - accuracy:
0.3514 - val_loss: 2.7368 - val_accuracy: 0.3540
Epoch 20/20
63/63 [==============================] - 1s 8ms/step - loss: 2.7254 - accuracy:
0.3514 - val_loss: 2.7010 - val_accuracy: 0.3540
```

[28]: `<tensorflow.python.keras.callbacks.History at 0x7fc7ff7a4b50>`

[ ]: