Assignment 5.2

July 6, 2021

1 Assignment 5.2

1.0.1 Implement the news classifier found in section 3.5 of Deep Learning with Python.

[15]:
```
import keras
```

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

[2]:
```
(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
```

```
/opt/conda/lib/python3.8/site-
packages/tensorflow/python/keras/datasets/reuters.py:148:
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/reuters.py:149:
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:])
```

[3]:
```
len(train_data)
```

[3]:
```
8982
```

[4]:
```
len(test_data)
```

[4]:
```
2246
```

[5]:

```
train_data[10]
```

[5]:

```
[1,
 245,
 273,
 207,
 156,
 53,
 74,
 160,
 26,
 14,
 46,
 296,
 26,
 39,
 74,
 2979,
 3554,
 14,
 46,
 4689,
 4329,
 86,
 61,
 3499,
 4795,
 14,
 61,
 451,
 4329,
 17,
 12]
```

[6]:

```
# decode back to text
word_index = reuters.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
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]:

```
train_labels[10]
```

[7]:

```
3
```

[8]:
```
decoded_newswire
```

[8]:
```
'? ? ? 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'
```

[27]:
```
# prepare the data
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)
```

[10]:
```
# one hot encode
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

y_train = to_one_hot(train_labels)
y_test = to_one_hot(test_labels)
```

[11]:
```
from keras.utils.np_utils import to_categorical
```

[12]:
```
one_hot_train_labels = to_categorical(train_labels)
one_hot_test_labels = to_categorical(test_labels)
```

[13]:
```
# Build the model
from keras import models, layers
```

[16]:
```
model = keras.Sequential([
    layers.Dense(64, activation="relu", input_shape = (10000,)),
    layers.Dense(64, activation="relu"),
    layers.Dense(46, activation="softmax")
])
```

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

[18]:

```python
# validate approach
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:]
```

[20]:

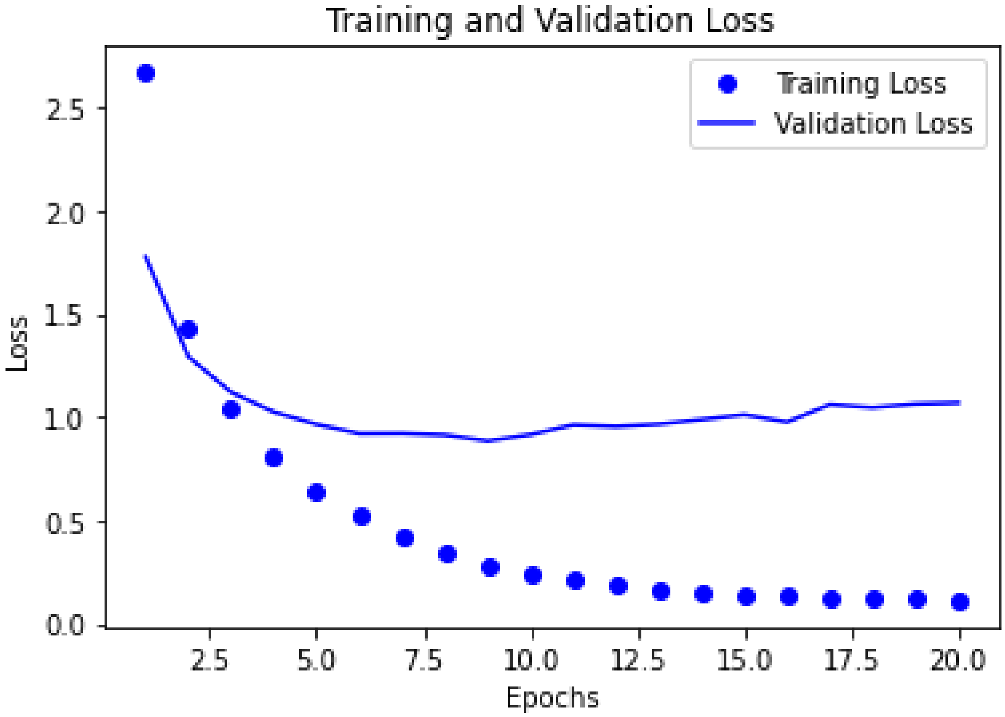
```python
# train the network for 20 epochs
history = model.fit(partial_x_train,
                   partial_y_train,
                    epochs = 20,
                    batch_size = 512,
                    validation_data = (x_val, y_val))
```

```
Epoch 1/20
16/16 [==============================] - 1s 35ms/step - loss: 3.1693 - accuracy:
0.3622 - val_loss: 1.7754 - val_accuracy: 0.6490
Epoch 2/20
16/16 [==============================] - 0s 18ms/step - loss: 1.5401 - accuracy:
0.6906 - val_loss: 1.2919 - val_accuracy: 0.7180
Epoch 3/20
16/16 [==============================] - 0s 16ms/step - loss: 1.0713 - accuracy:
0.7721 - val_loss: 1.1199 - val_accuracy: 0.7550
Epoch 4/20
16/16 [==============================] - 0s 15ms/step - loss: 0.8409 - accuracy:
0.8236 - val_loss: 1.0236 - val_accuracy: 0.7770
Epoch 5/20
16/16 [==============================] - 0s 15ms/step - loss: 0.6288 - accuracy:
0.8680 - val_loss: 0.9641 - val_accuracy: 0.8030
Epoch 6/20
16/16 [==============================] - 0s 16ms/step - loss: 0.5210 - accuracy:
0.8918 - val_loss: 0.9193 - val_accuracy: 0.8080
Epoch 7/20
16/16 [==============================] - 0s 15ms/step - loss: 0.4124 - accuracy:
0.9143 - val_loss: 0.9202 - val_accuracy: 0.8070
Epoch 8/20
16/16 [==============================] - 0s 15ms/step - loss: 0.3408 - accuracy:
0.9296 - val_loss: 0.9125 - val_accuracy: 0.8130
Epoch 9/20
16/16 [==============================] - 0s 18ms/step - loss: 0.2712 - accuracy:
0.9397 - val_loss: 0.8837 - val_accuracy: 0.8110
Epoch 10/20
16/16 [==============================] - 0s 22ms/step - loss: 0.2266 - accuracy:
0.9486 - val_loss: 0.9144 - val_accuracy: 0.8190
Epoch 11/20
16/16 [==============================] - 0s 24ms/step - loss: 0.1914 - accuracy:
0.9532 - val_loss: 0.9616 - val_accuracy: 0.8120
```

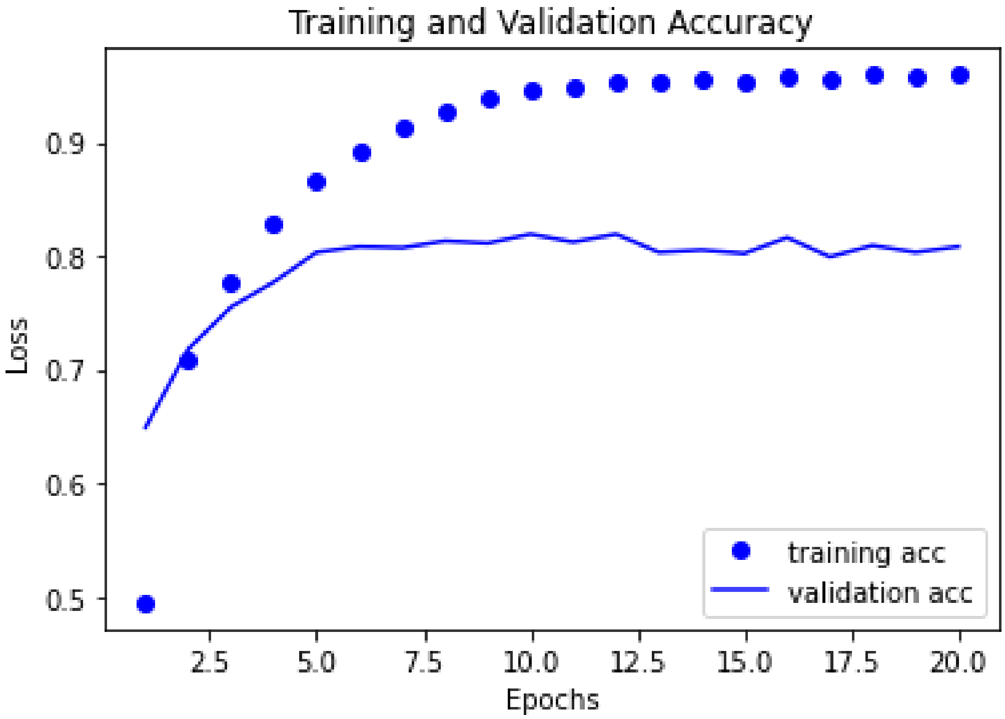
```
Epoch 12/20
16/16 [==============================] - 0s 15ms/step - loss: 0.1853 - accuracy:
0.9519 - val_loss: 0.9536 - val_accuracy: 0.8190
Epoch 13/20
16/16 [==============================] - 0s 15ms/step - loss: 0.1561 - accuracy:
0.9570 - val_loss: 0.9649 - val_accuracy: 0.8030
Epoch 14/20
16/16 [==============================] - 0s 15ms/step - loss: 0.1389 - accuracy:
0.9585 - val_loss: 0.9873 - val_accuracy: 0.8050
Epoch 15/20
16/16 [==============================] - 0s 15ms/step - loss: 0.1255 - accuracy:
0.9587 - val_loss: 1.0108 - val_accuracy: 0.8020
Epoch 16/20
16/16 [==============================] - 0s 16ms/step - loss: 0.1193 - accuracy:
0.9595 - val_loss: 0.9751 - val_accuracy: 0.8160
Epoch 17/20
16/16 [==============================] - 0s 15ms/step - loss: 0.1178 - accuracy:
0.9609 - val_loss: 1.0601 - val_accuracy: 0.7990
Epoch 18/20
16/16 [==============================] - 0s 19ms/step - loss: 0.1118 - accuracy:
0.9618 - val_loss: 1.0452 - val_accuracy: 0.8090
Epoch 19/20
16/16 [==============================] - 0s 32ms/step - loss: 0.1171 - accuracy:
0.9608 - val_loss: 1.0625 - val_accuracy: 0.8030
Epoch 20/20
16/16 [==============================] - 0s 27ms/step - loss: 0.0998 - accuracy:
0.9637 - val_loss: 1.0682 - val_accuracy: 0.8080
```

[22]:

```python
# display loss & accuracy plots
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()
```



[24]:
```
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()
```



[29]:

```
# retrain model from scratch
model = models.Sequential([
  layers.Dense(64, activation="relu", input_shape = (10000,)),
  layers.Dense(64, activation="relu"),
  layers.Dense(46, activation="softmax")
])

model.compile(optimizer="rmsprop",
              loss="categorical_crossentropy",
              metrics=["accuracy"])

model.fit(partial_x_train,
          partial_y_train,
          epochs=9,
          batch_size=512,
         validation_data = (x_val, y_val))

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

```
Epoch 1/9
16/16 [==============================] - 1s 24ms/step - loss: 3.1262 - accuracy:
0.4261 - val_loss: 1.7757 - val_accuracy: 0.6290
Epoch 2/9
16/16 [==============================] - 0s 15ms/step - loss: 1.5662 - accuracy:
```

```
0.6823 - val_loss: 1.3151 - val_accuracy: 0.7050
Epoch 3/9
16/16 [==============================] - 0s 14ms/step - loss: 1.0756 - accuracy:
0.7678 - val_loss: 1.1274 - val_accuracy: 0.7580
Epoch 4/9
16/16 [==============================] - 0s 16ms/step - loss: 0.8390 - accuracy:
0.8255 - val_loss: 1.0172 - val_accuracy: 0.7880
Epoch 5/9
16/16 [==============================] - 0s 16ms/step - loss: 0.6591 - accuracy:
0.8658 - val_loss: 0.9544 - val_accuracy: 0.7980
Epoch 6/9
16/16 [==============================] - 0s 16ms/step - loss: 0.5198 - accuracy:
0.8980 - val_loss: 0.9076 - val_accuracy: 0.8120
Epoch 7/9
16/16 [==============================] - 0s 16ms/step - loss: 0.4058 - accuracy:
0.9187 - val_loss: 0.9236 - val_accuracy: 0.8040
Epoch 8/9
16/16 [==============================] - 0s 16ms/step - loss: 0.3406 - accuracy:
0.9304 - val_loss: 0.9041 - val_accuracy: 0.8110
Epoch 9/9
16/16 [==============================] - 0s 17ms/step - loss: 0.2787 - accuracy:
0.9411 - val_loss: 0.8847 - val_accuracy: 0.8200
71/71 [==============================] - 0s 2ms/step - loss: 0.9742 - accuracy:
0.7912
```

[30]:
```
results
```

[30]:
```
[0.9741821885108948, 0.7911843061447144]
```

[31]:
```
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)
```

[31]:
```
0.18967052537845058
```

[32]:
```
predictions = model.predict(x_test)
```

[33]:
```
predictions[0].shape
```

[33]:
```
(46,)
```

[37]:
```
# each entry in predictions is a vector of length 46
```

[35]:
```
np.sum(predictions[0])
```

[35]:
```
1.0000002
```

[38]:
```python
# the coefficients in the vector sum to 1
```

[36]:
```python
np.argmax(predictions[0])
```

[36]:
```
3
```

[39]:
```python
# different way to handle labels & the loss
y_train = np.array(train_labels)
y_test = np.array(test_labels)
```

[43]:
```python
model.compile(optimizer="rmsprop",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
```

[46]:
```python
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 9ms/step - loss: 3.2004 - accuracy:
0.3201 - val_loss: 2.0835 - val_accuracy: 0.4140
Epoch 2/20
63/63 [==============================] - 0s 6ms/step - loss: 1.8746 - accuracy:
0.5051 - val_loss: 1.5848 - val_accuracy: 0.6760
Epoch 3/20
63/63 [==============================] - 0s 6ms/step - loss: 1.4093 - accuracy:
0.7058 - val_loss: 1.4028 - val_accuracy: 0.6960
Epoch 4/20
63/63 [==============================] - 0s 6ms/step - loss: 1.1917 - accuracy:
0.7246 - val_loss: 1.3505 - val_accuracy: 0.7030
Epoch 5/20
63/63 [==============================] - 0s 6ms/step - loss: 1.0440 - accuracy:
0.7449 - val_loss: 1.3314 - val_accuracy: 0.6950
Epoch 6/20
63/63 [==============================] - 0s 6ms/step - loss: 0.9217 - accuracy:
0.7734 - val_loss: 1.3311 - val_accuracy: 0.7060
```

```
Epoch 7/20
63/63 [==============================] - 0s 7ms/step - loss: 0.8468 - accuracy:
0.7838 - val_loss: 1.3553 - val_accuracy: 0.7070
Epoch 8/20
63/63 [==============================] - 0s 6ms/step - loss: 0.8083 - accuracy:
0.7849 - val_loss: 1.3692 - val_accuracy: 0.7070
Epoch 9/20
63/63 [==============================] - 0s 7ms/step - loss: 0.7558 - accuracy:
0.7906 - val_loss: 1.4055 - val_accuracy: 0.7080
Epoch 10/20
63/63 [==============================] - 0s 6ms/step - loss: 0.7119 - accuracy:
0.8094 - val_loss: 1.4118 - val_accuracy: 0.7050
Epoch 11/20
63/63 [==============================] - 0s 7ms/step - loss: 0.6569 - accuracy:
0.8271 - val_loss: 1.4345 - val_accuracy: 0.7070
Epoch 12/20
63/63 [==============================] - 0s 6ms/step - loss: 0.6485 - accuracy:
0.8293 - val_loss: 1.4658 - val_accuracy: 0.7020
Epoch 13/20
63/63 [==============================] - 0s 7ms/step - loss: 0.5917 - accuracy:
0.8398 - val_loss: 1.5320 - val_accuracy: 0.7010
Epoch 14/20
63/63 [==============================] - 0s 6ms/step - loss: 0.5843 - accuracy:
0.8401 - val_loss: 1.5475 - val_accuracy: 0.7080
Epoch 15/20
63/63 [==============================] - 0s 6ms/step - loss: 0.5401 - accuracy:
0.8508 - val_loss: 1.6211 - val_accuracy: 0.7040
Epoch 16/20
63/63 [==============================] - 0s 6ms/step - loss: 0.5032 - accuracy:
0.8618 - val_loss: 1.6301 - val_accuracy: 0.7080
Epoch 17/20
63/63 [==============================] - 0s 6ms/step - loss: 0.5021 - accuracy:
0.8561 - val_loss: 1.7094 - val_accuracy: 0.6960
Epoch 18/20
63/63 [==============================] - 0s 6ms/step - loss: 0.4704 - accuracy:
0.8695 - val_loss: 1.7055 - val_accuracy: 0.7010
Epoch 19/20
63/63 [==============================] - 0s 6ms/step - loss: 0.4409 - accuracy:
0.8718 - val_loss: 1.7482 - val_accuracy: 0.7080
Epoch 20/20
63/63 [==============================] - 0s 6ms/step - loss: 0.4338 - accuracy:
0.8739 - val_loss: 1.8433 - val_accuracy: 0.6970
```

```
[46]: <tensorflow.python.keras.callbacks.History at 0x7fd66c0b4370>
```

[48]:

```
# ~70% accuracy
# ~ 9% drop due to compressing a lot of info
```

Reference: https://github.com/fchollet/deep-learning-with-python-notebooks

page xviii from book

[ ]:

```
```