Assignment 5.2

author: Rachel Nelson

class: DSC650

Assignment 5.2

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

20
```
#3.12 Loading the Reuters dataset
from keras.datasets import reuters
(train_data, train_labels), (test_data, test_labels) = reuters.load_data(num_words=10000)

len(train_data)
len(test_data)

train_data[10]
```

20
```
[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]
```

21
```
#  3.13 Decoding newswires 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]])

train_labels[10]
```

```
3
```

```
# 3.14 Encoding 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)

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

one_hot_train_labels = to_one_hot(train_labels)
one_hot_test_labels = to_one_hot(test_labels)
```

```
# 3.15 Model definition
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'))
```

```
# 3.16 Compiling the modeL
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

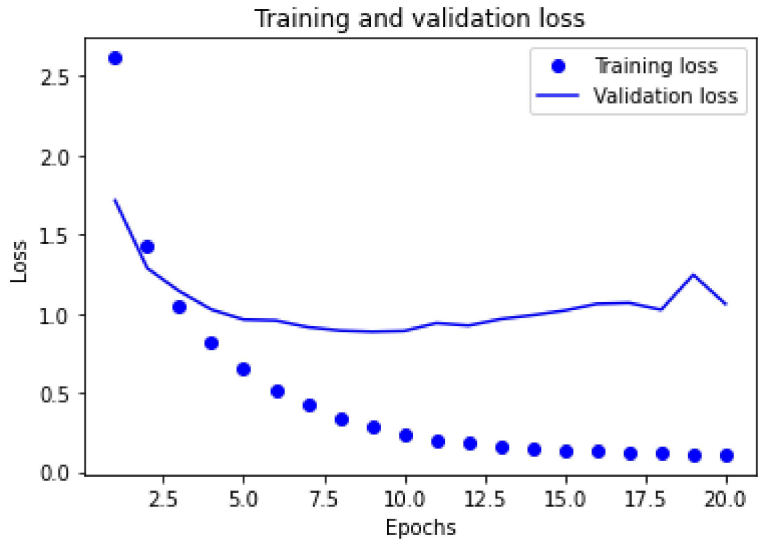
```
#  3.17 Setting aside a validation set
x_val = x_train[:1000]
partial_x_train = x_train[1000:]

y_val = one_hot_train_labels[:1000]
partial_y_train = one_hot_train_labels[1000:]
```

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

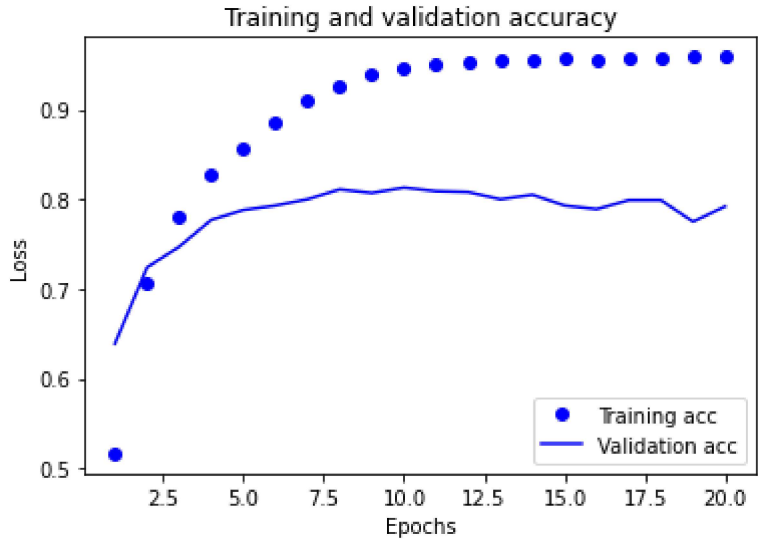
```
Epoch 1/20
16/16 [==============================] - 1s 21ms/step - loss: 2.6140 - accuracy: 0.5167 - val_loss: 1.713
Epoch 2/20
```

```
16/16 [==============================] - 0s 13ms/step - loss: 1.4213 - accuracy: 0.7068 - val_loss: 1.285
Epoch 3/20
16/16 [==============================] - 0s 13ms/step - loss: 1.0481 - accuracy: 0.7799 - val_loss: 1.140
Epoch 4/20
16/16 [==============================] - 0s 13ms/step - loss: 0.8231 - accuracy: 0.8280 - val_loss: 1.024
Epoch 5/20
16/16 [==============================] - 0s 13ms/step - loss: 0.6545 - accuracy: 0.8566 - val_loss: 0.961
Epoch 6/20
16/16 [==============================] - 0s 14ms/step - loss: 0.5209 - accuracy: 0.8855 - val_loss: 0.956
Epoch 7/20
16/16 [==============================] - 0s 13ms/step - loss: 0.4208 - accuracy: 0.9094 - val_loss: 0.913
Epoch 8/20
16/16 [==============================] - 0s 13ms/step - loss: 0.3428 - accuracy: 0.9262 - val_loss: 0.892
Epoch 9/20
16/16 [==============================] - 0s 13ms/step - loss: 0.2825 - accuracy: 0.9399 - val_loss: 0.884
Epoch 10/20
16/16 [==============================] - 0s 13ms/step - loss: 0.2331 - accuracy: 0.9470 - val_loss: 0.889
Epoch 11/20
16/16 [==============================] - 0s 14ms/step - loss: 0.2002 - accuracy: 0.9494 - val_loss: 0.939
Epoch 12/20
16/16 [==============================] - 0s 13ms/step - loss: 0.1809 - accuracy: 0.9516 - val_loss: 0.924
Epoch 13/20
16/16 [==============================] - 0s 13ms/step - loss: 0.1585 - accuracy: 0.9549 - val_loss: 0.964
Epoch 14/20
16/16 [==============================] - 0s 13ms/step - loss: 0.1466 - accuracy: 0.9554 - val_loss: 0.989
Epoch 15/20
16/16 [==============================] - 0s 14ms/step - loss: 0.1368 - accuracy: 0.9560 - val_loss: 1.018
Epoch 16/20
16/16 [==============================] - 0s 13ms/step - loss: 0.1315 - accuracy: 0.9558 - val_loss: 1.061
Epoch 17/20
16/16 [==============================] - 0s 12ms/step - loss: 0.1235 - accuracy: 0.9569 - val_loss: 1.067
Epoch 18/20
16/16 [==============================] - 0s 13ms/step - loss: 0.1178 - accuracy: 0.9564 - val_loss: 1.023
Epoch 19/20
16/16 [==============================] - 0s 13ms/step - loss: 0.1124 - accuracy: 0.9588 - val_loss: 1.244
Epoch 20/20
16/16 [==============================] - 0s 13ms/step - loss: 0.1105 - accuracy: 0.9584 - val_loss: 1.060
```

```
27  #  3.19 Plotting the training and validation loss
    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()
```



28
```
#  3.20 Plotting the training and validation accuracy
plt.clf()

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
```
#  3.21 Retraining a model from scratch
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)

results

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)


Epoch 1/8
16/16 [==============================] - 1s 21ms/step - loss: 2.6448 - accuracy: 0.5233 - val_loss: 1.754
Epoch 2/8
16/16 [==============================] - 0s 13ms/step - loss: 1.4492 - accuracy: 0.6954 - val_loss: 1.322
Epoch 3/8
16/16 [==============================] - 0s 14ms/step - loss: 1.0646 - accuracy: 0.7734 - val_loss: 1.135
Epoch 4/8
16/16 [==============================] - 0s 13ms/step - loss: 0.8405 - accuracy: 0.8242 - val_loss: 1.020
Epoch 5/8
16/16 [==============================] - 0s 13ms/step - loss: 0.6664 - accuracy: 0.8632 - val_loss: 0.985
Epoch 6/8
16/16 [==============================] - 0s 13ms/step - loss: 0.5377 - accuracy: 0.8890 - val_loss: 0.910
Epoch 7/8
16/16 [==============================] - 0s 13ms/step - loss: 0.4344 - accuracy: 0.9103 - val_loss: 0.881
Epoch 8/8
16/16 [==============================] - 0s 12ms/step - loss: 0.3539 - accuracy: 0.9276 - val_loss: 0.885
71/71 [==============================] - 0s 1ms/step - loss: 0.9911 - accuracy: 0.7769□□□□□□□□□□□□□□□□□□□□□□□□

29  0.195013357079252


30  #  3.22 Retraining a model from scratch
    predictions = model.predict(x_test)

    predictions[0].shape
    np.sum(predictions[0])
    np.argmax(predictions[0])

    y_train = np.array(train_labels)
    y_test = np.array(test_labels)

    model.compile(optimizer='rmsprop',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])



31  # 3.23 A model with an information bottleneck
    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 8ms/step - loss: 3.2372 - accuracy: 0.3355 - val_loss: 2.6886
Epoch 2/20
63/63 [==============================] - 1s 7ms/step - loss: 2.3230 - accuracy: 0.4213 - val_loss: 2.0128
Epoch 3/20
63/63 [==============================] - 0s 7ms/step - loss: 1.8037 - accuracy: 0.5471 - val_loss: 1.7330
Epoch 4/20
63/63 [==============================] - 0s 7ms/step - loss: 1.6005 - accuracy: 0.5531 - val_loss: 1.6413
Epoch 5/20
63/63 [==============================] - 0s 7ms/step - loss: 1.4930 - accuracy: 0.5565 - val_loss: 1.6067
Epoch 6/20
63/63 [==============================] - 0s 7ms/step - loss: 1.4118 - accuracy: 0.5877 - val_loss: 1.5854
Epoch 7/20
63/63 [==============================] - 0s 7ms/step - loss: 1.3511 - accuracy: 0.5927 - val_loss: 1.5585
Epoch 8/20
63/63 [==============================] - 0s 7ms/step - loss: 1.3042 - accuracy: 0.5943 - val_loss: 1.5675
Epoch 9/20
63/63 [==============================] - 0s 7ms/step - loss: 1.2641 - accuracy: 0.5957 - val_loss: 1.5886
Epoch 10/20
63/63 [==============================] - 0s 7ms/step - loss: 1.2291 - accuracy: 0.5972 - val_loss: 1.5863
Epoch 11/20
63/63 [==============================] - 0s 7ms/step - loss: 1.2003 - accuracy: 0.6015 - val_loss: 1.5952
Epoch 12/20
63/63 [==============================] - 0s 7ms/step - loss: 1.1731 - accuracy: 0.6061 - val_loss: 1.6302
Epoch 13/20
63/63 [==============================] - 0s 7ms/step - loss: 1.1490 - accuracy: 0.6131 - val_loss: 1.6503
Epoch 14/20
63/63 [==============================] - 0s 6ms/step - loss: 1.1309 - accuracy: 0.6275 - val_loss: 1.6671
Epoch 15/20
63/63 [==============================] - 0s 7ms/step - loss: 1.1094 - accuracy: 0.6353 - val_loss: 1.6717
Epoch 16/20
63/63 [==============================] - 0s 7ms/step - loss: 1.0948 - accuracy: 0.6397 - val_loss: 1.7378
Epoch 17/20
63/63 [==============================] - 0s 7ms/step - loss: 1.0769 - accuracy: 0.6462 - val_loss: 1.7676
Epoch 18/20
63/63 [==============================] - 0s 7ms/step - loss: 1.0639 - accuracy: 0.6646 - val_loss: 1.7743
Epoch 19/20
63/63 [==============================] - 0s 7ms/step - loss: 1.0543 - accuracy: 0.6630 - val_loss: 1.8196
Epoch 20/20
63/63 [==============================] - 0s 7ms/step - loss: 1.0419 - accuracy: 0.6783 - val_loss: 1.8339
```

```
<keras.callbacks.History at 0x254ffc1c588>
```