Deep Learning Example

Prof. Gheith Abandah

Reference: François Chollet, *Deep Learning with Python*, Manning Pub. 2018

Outline

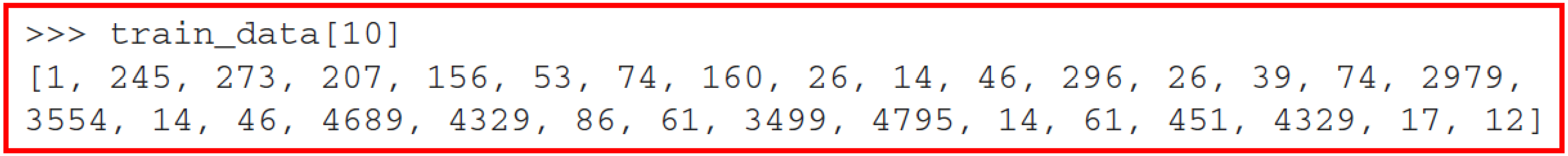
1. Problem Definition
2. Data Loading and Preparation
3. Building the Deep Model
4. Training and Evaluation
5. Callbacks to Control Training

1. Problem Definition

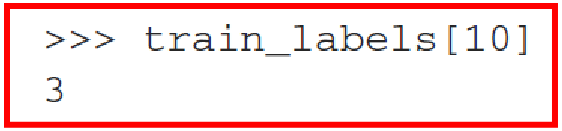
- *Reuters dataset*, a set of short newswires and their topics, published by Reuters in 1986.
- Each story is restricted to the 10,000 most frequently occurring words found in the data (`num_words=10000`) ; words are replaced with numbers.
- 46 mutually exclusive topics: This problem is *single-label, multiclass classification*
- 8,982 training examples
- 2,246 test examples

2. Data Loading and Preparation

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

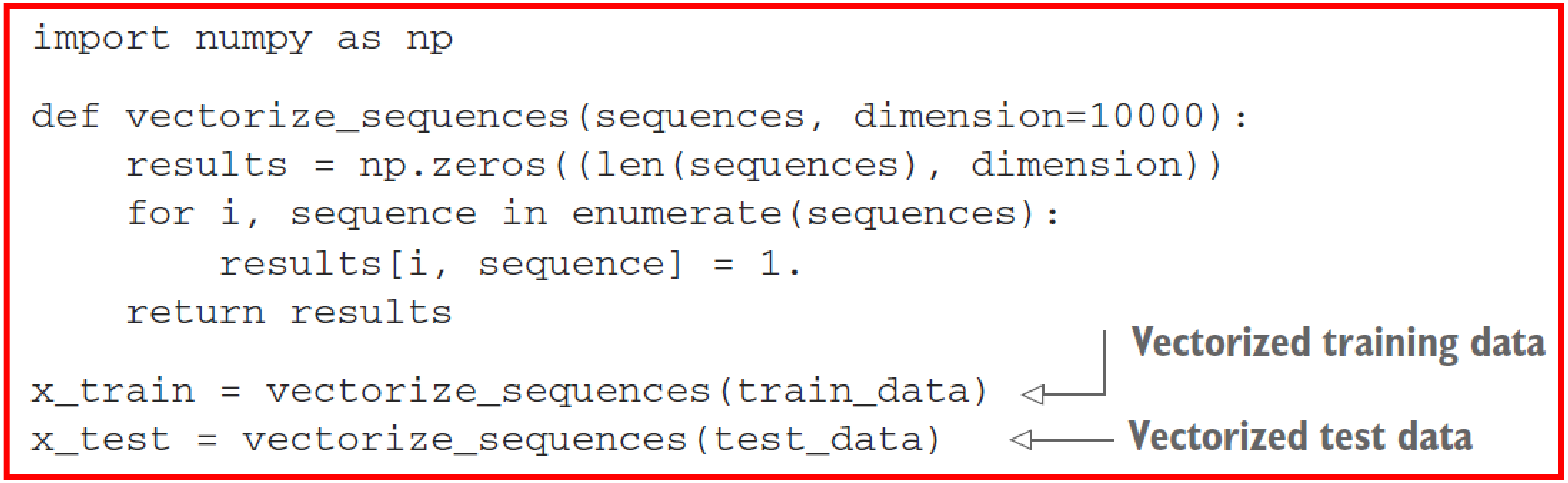


```
>>> train_data[10]
[1, 245, 273, 207, 156, 53, 74, 160, 26, 14, 46, 296, 26, 39, 74, 2979,
3554, 14, 46, 4689, 4329, 86, 61, 3499, 4795, 14, 61, 451, 4329, 17, 12]
```



```
>>> train_labels[10]
3
```

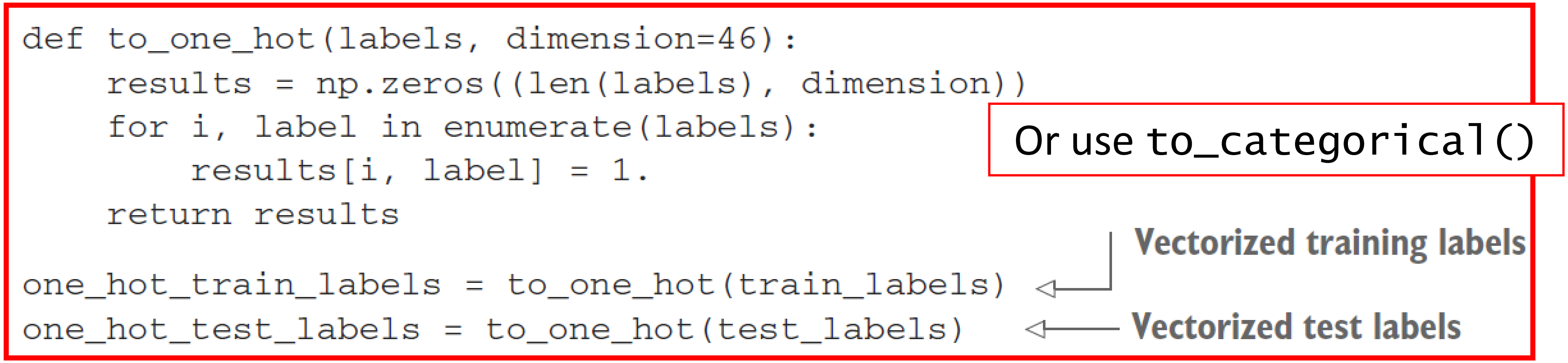
2. Data Loading and Preparation



```
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)
```

Vectorized training data

Vectorized 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)
```

Vectorized training labels

Vectorized test labels

Or use `to_categorical()`

2. Data Loading and Preparation

- Set apart 1,000 samples of the train set to use as 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. Building the Deep Model

```
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'))

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

4. Training and Evaluation

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

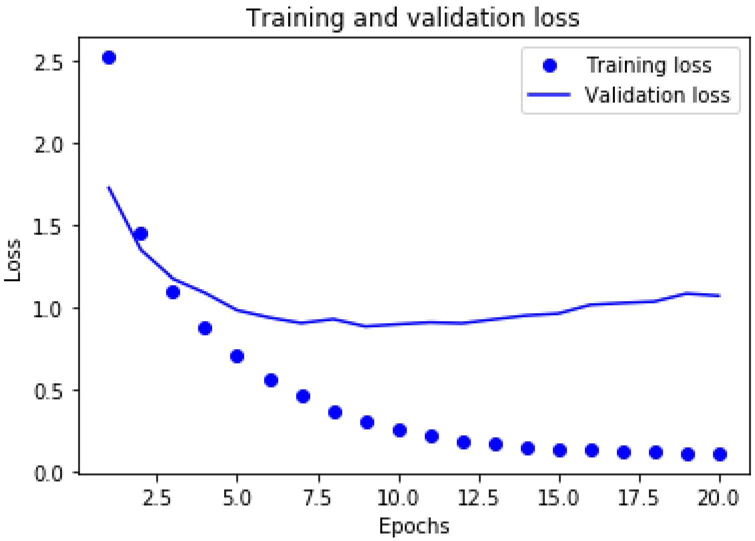
```
Train on 7982 samples, validate on 1000 samples
Epoch 1/20
7982/7982 [=====] - 1s - loss: 2.5241 - acc: 0.4952 -
      val_loss: 1.7263 - val_acc: 0.6100
```

…

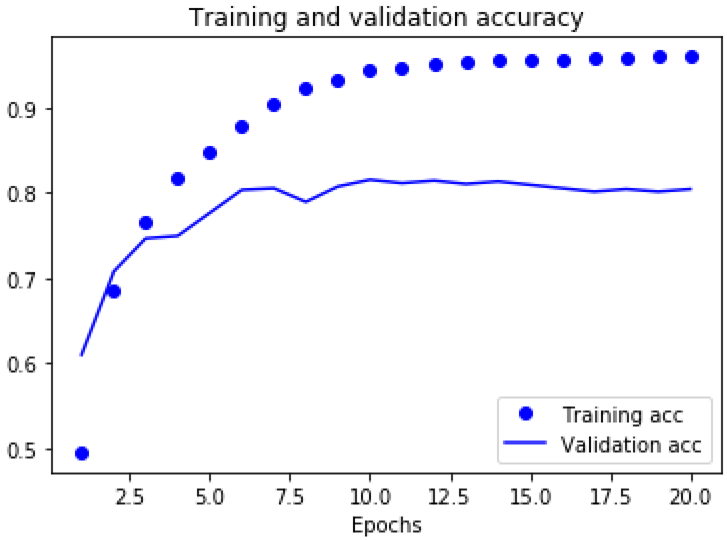
4. Training and Evaluation

```
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='Val. loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

4. Training and Evaluation



Training and validation loss



Training and validation accuracy

```
results = model.evaluate(x_test, one_hot_test_labels)
>>> results
[0.9876462854876257, 0.7769367765180786]
```

5. Callbacks to Control Training

```
callbacks_list = [
       keras.callbacks.EarlyStopping(
              monitor='accuracy', patience=2,),
       keras.callbacks.ModelCheckpoint(
              filepath='my_model.h5',
              monitor='accuracy',
              save_best_only=True,)]
model.compile(optimizer='rmsprop',
       loss='categorical_crossentropy',
       metrics=['accuracy'])
model.fit(partial_x_train, partial_y_train,
       epochs=20, batch_size=512,
       callbacks=callbacks_list,
       validation_data=(x_val, y_val))
```

Summary

1. Problem Definition
2. Data Loading and Preparation
3. Building the Deep Model
4. Training and Evaluation
5. Callbacks to Control Training