Group Homework 2: Deep Learning Project

108034534 王禹晴 108034573 蔡和諺 108034542 溫致淵 108034549 蘇怡安

主題 ： 新聞多元分類分析- Reuters 新聞

詳述 ： 許多民眾只對某一特定主題的新聞感興趣，而這需要將新聞條加以分類才能做到。我們以 Reuters 新聞為例，將新聞條分類標籤成 46 大類，使民眾可以自由選擇自己喜愛的主題，並且我們加以分析。

Reuters 資料集 ： 釋出在 1986 年，為一系列短新聞及對應話題的資料集；是文字分類問題最常用的小資料集。和 IMDB、MNIST 資料集類似，Reuters 資料集也可以通過 Keras 直接下載。

我們構建神經網路將 Reuters 新聞分類。因為有多個類別，屬於多分類問題，而每條資料只屬於一個類別，所以是單標籤多分類問題；如果每條資料可以被分到多個類別中，那問題則屬於多標籤多分類問題。

路透社資料集

首先我們會先載入資料集

其中有 8982 條訓練集，2246 條測試集。每個樣本表示成整數列表 ：

```
>>> 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]
```

也可以將整數列表轉換成原始資料[英文句子]，如下 ：

```
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, '?') f
or i in train_data[0]])
```

同時，我們也將編碼對應到的單字 print 出來。(雙向)

準備資料(數據預處理)

整數資料向量化，與 IMDB 資料集處理方法相同。

```
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)
```

標籤的向量化有兩種方法：將標籤列表轉換成整數張量；使用 one-hot 編碼。

One-hot 編碼方式是類別資料常用的一種資料格式，也稱為 categorical encoding。

```
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)
```

Keras 中有一個內建的 One-hot 編碼轉換函式：

```
from keras.utils.np_utils import to_categorical

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

模型搭建

使用 Dense 線性連線堆疊結構，每層網路只能處理上層網路的輸出結果。如果網路層丟失了一些關於分類問題的資訊，那麼下一層網路並不能恢復這些資訊：每個網路層潛在地成為一個資訊處理瓶頸。

網路定義 ：

```
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(128, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
```

1. 關於這個網路架構有兩點需要注意：

   最後一層網路神經元數目為 46.意味著每個輸入樣本最終變成 46 維的向量。輸出向量的每個數表示不同的類別；

2. 最後一層網路使用 softmax 啟用函式–網路會輸出一個 46 類的概率分佈。每個輸入最終都會產生一個 46 維的向量，每個數表示屬於該類別的概率，46 個數加起來等於 1

最好的損失函式為 categorical_crossentropy—衡量兩個概率分佈之間的距離：網路的輸出向量和標籤的真實分佈向量。通過最小化兩個分佈之間的距離，訓練網路模型，使得輸出向量儘可能與真實分佈相似。

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

模型驗證

在訓練資料中分出 1000 條樣本做為驗證集 ：

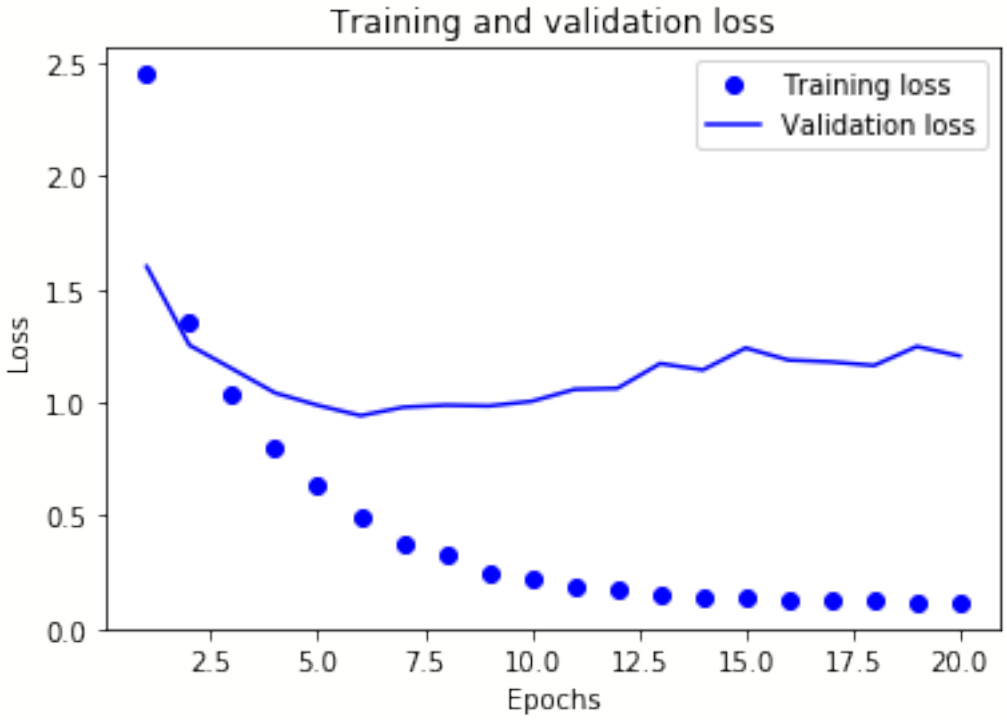
```
x_val = x_train[:1000]
partial_x_train = x_train[1000:]
y_val = one_hot_train_labels[:1000]
partial_y_train = one_hot_train_labels[1000:]
```

訓練 20 個 epochs：

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

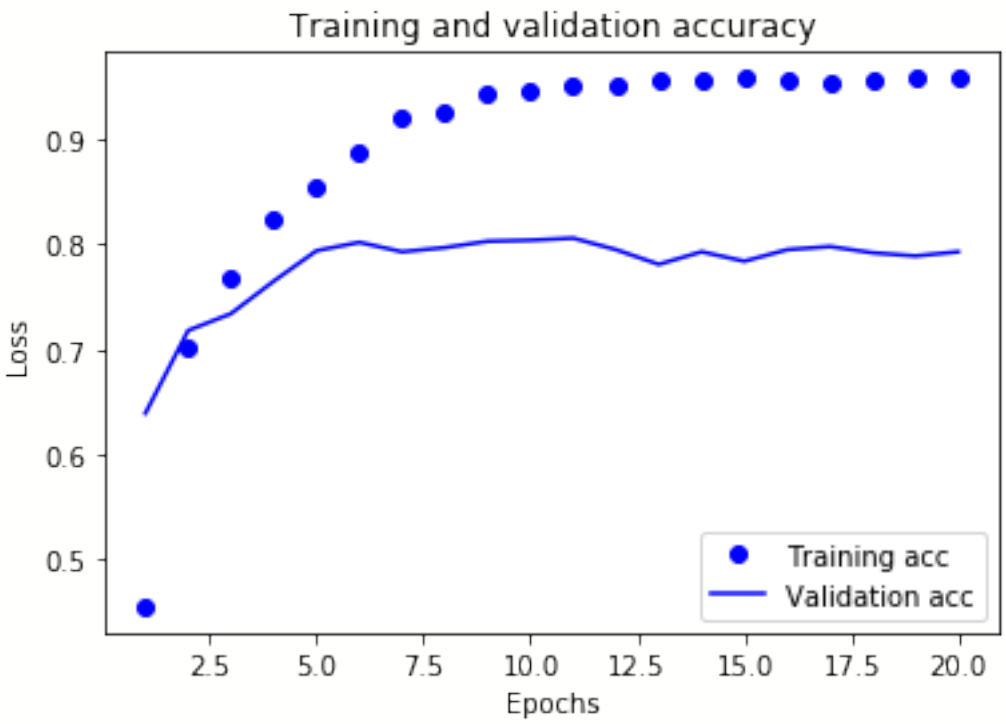
訓練集和驗證集的損失值變化 ：

```
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()
```



訓練集和驗證集的準確率變化：

```
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()
```



模型在第 9 次 epochs 之後開始過擬合。我們將 epochs 設定為 8 重新訓練，

同時在測試集上測試。

```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,
)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
model.compile(optimizer=RMSprop(lr=0.001),
              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)
[loss, acc]
```

```
[1.0594612212448595, 0.7702582478523254]
```

預測新資料

使用 predict 函式，產生一個 46 維的概率分佈。在測試資料上進行預測：

```
predictions_1 = model.predict(x_test)
predictions_2 = model.predict_classes(x_test)
```

在預測結果中概率最大的類別就是預測類：

```
np.argmax(predictions_1[0]) #第一條新聞預測值為 3
```

小結

- N 分類問題，網路最後 Dense 層神經元數目為 N
- 單標籤多分類問題中，最後一層的啟用函式為 softmax，產生一個包含 N 類的概率分佈
- categorical crossentropy 是處理單標籤多分類問題最常用的損失函式；
- 在多分類問題中有兩種標籤處理方式：

  1.使用 categorical encoding(one-hot)編碼，將標籤 one-hot 化，同時使用 categorical_crossentropy 作為損失函式；

  2.編碼成整數向量，使用 sparse_categorical_crossentropy 作為損失函式；

模型的通用性

能夠將資料集進行多元分類，對未來新的資料集能夠有所幫助，此模型的搭建適用各類資料集，其準確性之穩定可幫助避免出差錯。

結果的討論

如果分類數目過大，應該避免網路中間層數目過小(比分類數目小–資訊壓縮)，

產生資訊瓶頸。此模型之完整度還能再利用其他調整參數來提升。