**Python神经网络对语料库进行文本分类**

**相关知识**

以监督学习为例，假设我们有训练样本集

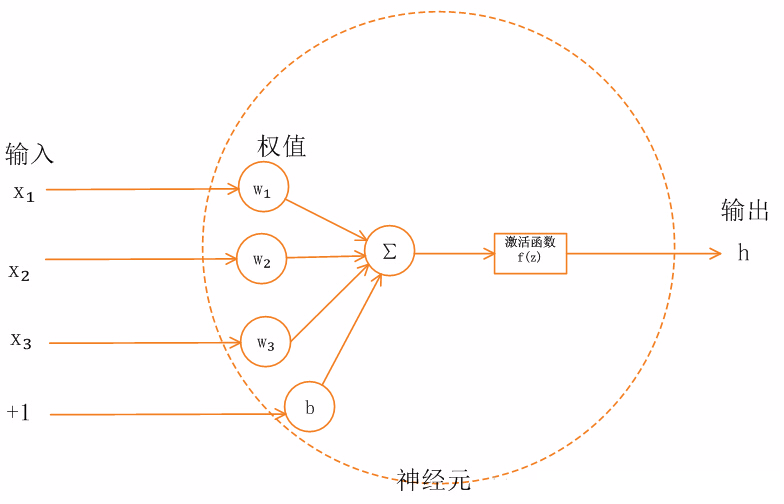
[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/01-1.png)

那么神经网络算法能够提供一种复杂且非线性的假设模型

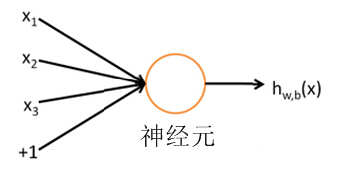
[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/01-2.png)

它具有参数W,b，可以以此参数来拟合我们的数据。

为了描述神经网络，我们先从最简单的神经网络讲起，这个神经网络仅由一个“神经元”构成，以下即是这个“神经元”的图示

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/01-3.png)

后文我们会介绍有多个神经元的神经网络，因此单个神经元模型我们后面会简化成如下图：

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/01-4.png)

这个“神经元”是一个以

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/01-5.png)

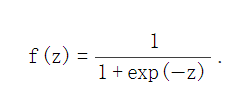
及截距 +1 为输入值的运算单元，其输出为

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/01-6.png)

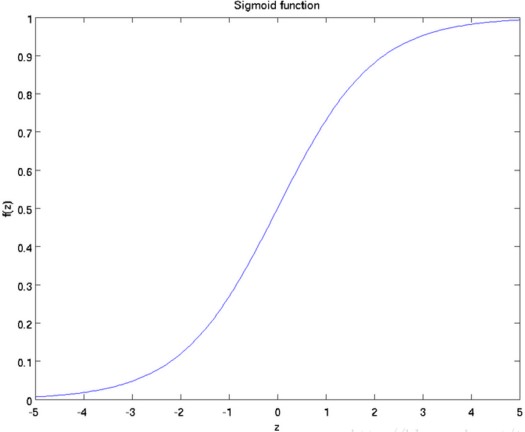
[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/01-7.png)

其中函数 f:R↦R被称为“激活函数”。在本教程中，我们选用sigmoid函数作为”激活函数” f(⋅)

sigmoid函数:

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/01-8.png)

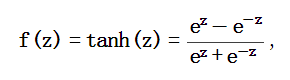
**sigmoid函数图像如下：**



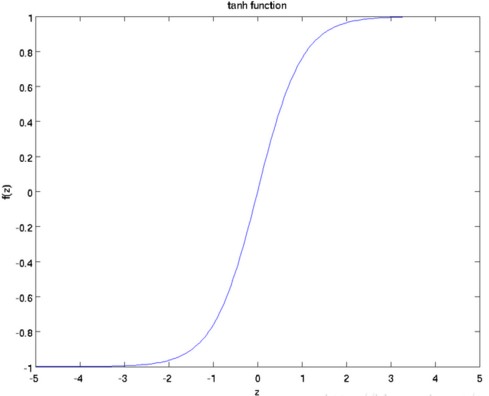
可以看出，这个单一“神经元”的输入－输出映射关系其实就是一个逻辑回归（logistic regression）。

此外，你也可以选择双曲正切函数（tanh）.

**tanh函数:**

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/02.png)

**tanh函数的图像如下:**



tanh(z) 函数是sigmoid函数的一种变体，它的取值范围为 [−1,1] ，而不是sigmoid函数的 [0,1]。

**注意:**这里我们不再令 x0=1 。取而代之，我们用单独的参数 b 来表示截距。

最后要说明的是，有一个等式我们以后会经常用到：如果选择

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/02-1.png)

也就是sigmoid函数，那么它的导数就是

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/02-2.png)

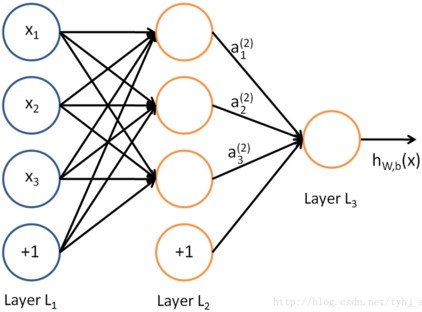
（如果选择tanh函数，那它的导数就是

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/02-3.png)

你可以根据sigmoid（或tanh）函数的定义自行推导这个等式。

**神经网络模型**

所谓神经网络就是将许多个单一“神经元”联结在一起，这样，一个“神经元”的输出就可以是另一个“神经元”的输入。例如，下图就是一个简单的神经网络：



我们使用蓝色圆圈来表示神经网络的输入，标上“+1”的圆圈被称为”’偏置节点”’，也就是截距项。神经网络最左边的一层叫做”’输入层”’，最右的一层叫做”’输出层”’（本例中，输出层只有一个节点）。中间所有节点组成的一层叫做”’隐藏层”’，因为我们不能在训练样本集中观测到它们的值。同时可以看到，以上神经网络的例子中有3个”’输入单元”’（偏置单元不计在内），3个”’隐藏单元”’及一个”’输出单元”’。

**系统环境**

Linux Ubuntu 16.04

Python3.6

**任务内容**

利用神经网络进行对reuters路透社语料库进行文本分类。

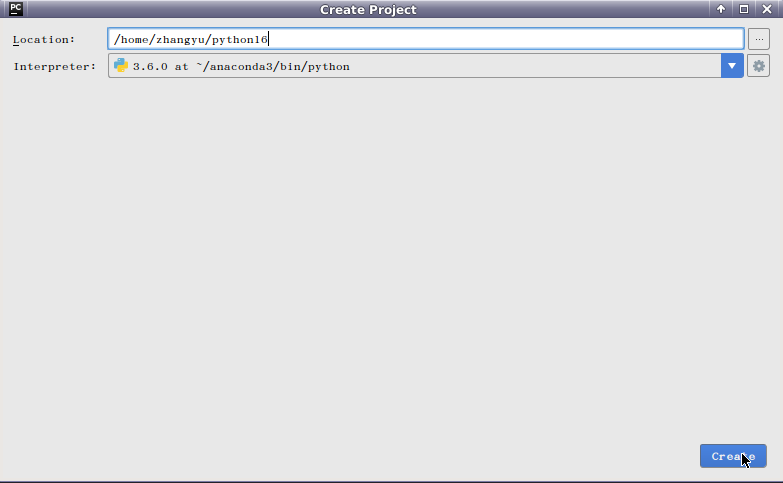
**任务步骤**

1. 准备实验所需要的数据

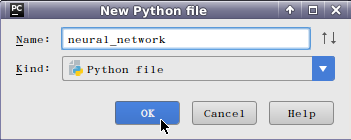
[view plain](https://www.ipieuvre.com/e/185/311/9539) [copy](https://www.ipieuvre.com/e/185/311/9539)

1. mkdir -p /home/zhangyu/.keras/datasets
2. cd /home/zhangyu/.keras/datasets
3. sudo wget http://192.168.1.100:60000/allfiles/python16/reuters.npz

2. 新建Python Project ，名为：python16



在python16项目下，新建Python file ，名为：neural\_network



3.利用神经网络进行对reuters路透社语料库进行文本分类，完整代码如下：

[view plain](https://www.ipieuvre.com/e/185/311/9539) [copy](https://www.ipieuvre.com/e/185/311/9539)

1. from \_\_future\_\_ **import** print\_function
2. **import** numpy as np
3. np.random.seed(1337)  # **for** reproducibility
4. from keras.datasets **import** reuters
5. from keras.models **import** Sequential
6. from keras.layers **import** Dense, Dropout, Activation
7. from keras.utils **import** np\_utils
8. from keras.preprocessing.text **import** Tokenizer
9. max\_words = 1000  # vocab大小
10. batch\_size = 32  # mini\_batch\_size
11. nb\_epoch = 5  # 大循环次数
12. print('Loading data...')
13. (X\_train, y\_train), (X\_test, y\_test) = reuters.load\_data(num\_words=max\_words, test\_split=0.2)  # 载入路透社语料 #打印
14. print(len(X\_train), 'train sequences')
15. print(len(X\_test), 'test sequences')
16. # 分类数目--原版路透社我记着是10来着，应该是语料用的是大的那个
17. nb\_classes = np.max(y\_train) + 1
18. print(nb\_classes, 'classes')
19. print('Vectorizing sequence data...')   # tokenize
20. tokenizer = Tokenizer(
21. nb\_words=max\_words)  # 序列化，取df前1000大 #这里有个非常好玩的事， X\_train 里面初始存的是wordindex，wordindex是按照词大小来的（应该是，因为直接就给撇了）#所以这个效率上还是很高的<br>#转化的还是binary，默认不是用tfidf
22. X\_train = tokenizer.sequences\_to\_matrix(X\_train, mode='binary')
23. X\_test = tokenizer.sequences\_to\_matrix(X\_test, mode='binary')
24. print('X\_train shape:', X\_train.shape)
25. print('X\_test shape:', X\_test.shape)
26. print('Convert class vector to binary class matrix (for use with categorical\_crossentropy)')
27. Y\_train = np\_utils.to\_categorical(y\_train, nb\_classes)
28. Y\_test = np\_utils.to\_categorical(y\_test, nb\_classes)
29. print('Y\_train shape:', Y\_train.shape)
30. print('Y\_test shape:', Y\_test.shape)
31. print('Building model...')
32. model = Sequential()   # 第一层 #Dense就是全连接层
33. model.add(Dense(512, input\_shape=(max\_words,)))  # 输入维度, 512==输出维度
34. model.add(Activation('relu'))  # 激活函数
35. model.add(Dropout(0.5))  # dropout  #第二层
36. model.add(Dense(nb\_classes))
37. model.add(Activation('softmax'))
38. # 损失函数设置、优化函数，衡量标准
39. model.compile(loss='categorical\_crossentropy',
40. optimizer='adam',
41. metrics=['accuracy'])
42. # 训练，交叉验证
43. history = model.fit(X\_train, Y\_train,
44. epochs=nb\_epoch, batch\_size=batch\_size,
45. verbose=1, validation\_split=0.1)
46. score = model.evaluate(X\_test, Y\_test,
47. batch\_size=batch\_size, verbose=1)
48. print('\nTest score:', score[0])
49. print('Test accuracy:', score[1])

最终运行结果：

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/09.png)

4.对完整代码进行分部描述，导入外包

[view plain](https://www.ipieuvre.com/e/185/311/9539) [copy](https://www.ipieuvre.com/e/185/311/9539)

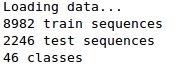
1. from \_\_future\_\_ **import** print\_function
2. **import** numpy as np
3. from keras.datasets **import** reuters
4. from keras.models **import** Sequential
5. from keras.layers **import** Dense, Dropout, Activation
6. from keras.utils **import** np\_utils
7. from keras.preprocessing.text **import** Tokenizer

5.载入数据

[view plain](https://www.ipieuvre.com/e/185/311/9539) [copy](https://www.ipieuvre.com/e/185/311/9539)

1. (X\_train, y\_train), (X\_test, y\_test) = reuters.load\_data(num\_words=max\_words, test\_split=0.2)  # 载入路透社语料 #打印
2. print(len(X\_train), 'train sequences')
3. print(len(X\_test), 'test sequences')

运行结果

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/05.png)

6.数据预处理

[view plain](https://www.ipieuvre.com/e/185/311/9539) [copy](https://www.ipieuvre.com/e/185/311/9539)

1. tokenizer = Tokenizer(
2. nb\_words=max\_words)  # 序列化，取df前1000大 #这里有个非常好玩的事， X\_train 里面初始存的是wordindex，wordindex是按照词大小来的（应该是，因为直接就给撇了） #所以这个效率上还是很高的 #转化的还是binary，默认不是用tfidf
3. X\_train = tokenizer.sequences\_to\_matrix(X\_train, mode='binary')
4. X\_test = tokenizer.sequences\_to\_matrix(X\_test, mode='binary')
5. print('X\_train shape:', X\_train.shape)
6. print('X\_test shape:', X\_test.shape)

运行结果

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/06.png)

7.特征提取

[view plain](https://www.ipieuvre.com/e/185/311/9539) [copy](https://www.ipieuvre.com/e/185/311/9539)

1. Y\_train = np\_utils.to\_categorical(y\_train, nb\_classes)
2. Y\_test = np\_utils.to\_categorical(y\_test, nb\_classes)
3. print('Y\_train shape:', Y\_train.shape)
4. print('Y\_test shape:', Y\_test.shape)

运行结果

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/07.png)

8.建立模型

[view plain](https://www.ipieuvre.com/e/185/311/9539) [copy](https://www.ipieuvre.com/e/185/311/9539)

1. model = Sequential()   # 第一层 #Dense就是全连接层
2. model.add(Dense(512, input\_shape=(max\_words,)))  # 输入维度, 512==输出维度
3. model.add(Activation('relu'))  # 激活函数
4. model.add(Dropout(0.5))  # dropout  #第二层
5. model.add(Dense(nb\_classes))
6. model.add(Activation('softmax'))
7. # 损失函数设置、优化函数，衡量标准
8. model.compile(loss='categorical\_crossentropy',
9. optimizer='adam',
10. metrics=['accuracy'])

运行结果：

[](https://www.ipieuvre.com/doc/exper/28ab0ee0-91ad-11e9-beeb-00215ec892f4/img/08.png)

9.训练，交叉验证

[view plain](https://www.ipieuvre.com/e/185/311/9539) [copy](https://www.ipieuvre.com/e/185/311/9539)

1. history = model.fit(X\_train, Y\_train,
2. epochs=nb\_epoch, batch\_size=batch\_size,
3. verbose=1, validation\_split=0.1)
4. score = model.evaluate(X\_test, Y\_test,
5. batch\_size=batch\_size, verbose=1)
6. print('\nTest score:', score[0])
7. print('Test accuracy:', score[1])