Python 基础学习

操作环境：MacOS、Python3.6
配置库：csv，json，numpy，matplotlib，scikit-learn 等，在 terminal 中执行 **pip install 库名** 安装

基本数据结构

numpy

```
>>> import numpy as np
>>> a = np.array([1,2,3])     # 新建一维数组
>>> b = np.array([(1.5,2,3),(4,5,6)], dtype = float)      # 新建二维数组
>>> a
array([1, 2, 3])
>>> b
array([[ 1.5,   2. ,   3. ],
       [ 4. ,   5. ,   6. ]])
>>> # Initial Placeholders
>>> np.zeros((3,4))       # 初始化三行四列数组，值为 0
array([[ 0.,   0.,   0.,   0.],
       [ 0.,   0.,   0.,   0.],
       [ 0.,   0.,   0.,   0.]])
>>> np.ones((2,3,4),dtype = np.int16)    # 初始化 2 个三行四列数组，值为 1
array([[[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]],

       [[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]]], dtype=int16)
>>> c = np.arange(10,25,5)       # 从[10, 25）中以步长为 5 取值
>>> c
array([10, 15, 20])
>>> np.linspace(0,2,9)         # 在[0，2]区间内均匀取 9 个点
```

```
array([ 0.  ,  0.25,  0.5 ,  0.75,  1.  ,  1.25,  1.5 ,  1.75,  2.  ])
>>> np.full((2,2),7)        # 新建二维数组，元素赋值为 7
array([[7, 7],
       [7, 7]])
>>> np.random.random((2,2))  # 随机生成一个二维数组
array([[ 0.36222235,  0.41498699],
       [ 0.87198129,  0.15325147]])
>>> a.shape     # a 的形状
(3,)
>>> len(a)      # a 的长度
3
>>> b.ndim      # b 的维度
2
>>> c.size      # c 的尺寸
3
>>> a-b
array([[-0.5,  0. ,  0. ],
       [-3. , -3. , -3. ]])
>>> np.subtract(a,b)    # 做差
array([[-0.5,  0. ,  0. ],
       [-3. , -3. , -3. ]])
>>> b+a
array([[ 2.5,  4. ,  6. ],
       [ 5. ,  7. ,  9. ]])
>>> np.add(b,a)         # 求和
array([[ 2.5,  4. ,  6. ],
       [ 5. ,  7. ,  9. ]])
>>> a/b
array([[ 0.66666667,  1.        ,  1.        ],
       [ 0.25      ,  0.4       ,  0.5       ]])
>>> np.divide(a,b)      # 除法
array([[ 0.66666667,  1.        ,  1.        ],
       [ 0.25      ,  0.4       ,  0.5       ]])
```

```
>>> a*b
array([[  1.5,   4. ,   9. ],
       [  4. ,  10. ,  18. ]])
>>> np.multiply(a,b)     # 乘法
array([[  1.5,   4. ,   9. ],
       [  4. ,  10. ,  18. ]])
>>> np.exp(a)            # 指数运算
array([  2.71828183,   7.3890561 ,  20.08553692])
>>> np.sqrt(a)           # 平方根
array([ 1.        ,  1.41421356,  1.73205081])
>>> np.sin(a)            # 正弦
array([ 0.84147098,  0.90929743,  0.14112001])
>>> np.cos(a)            # 余弦
array([ 0.54030231, -0.41614684, -0.9899925 ])
>>> np.log(a)            # 对数运算
array([ 0.        ,  0.69314718,  1.09861229])
>>> e = []               # 新建列表
>>> f = [1,2,3]
>>> e.append(f)          # 列表追加
>>> e
[[1, 2, 3]]
```

文件读取

1.TXT 读取

内容：

```
1 23 23
2 56 56
3 56 56
```

代码：

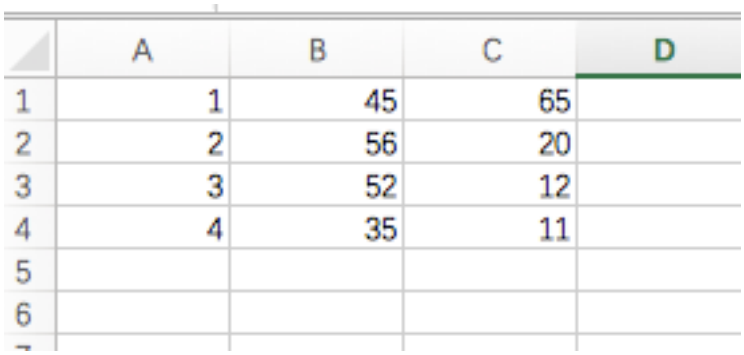
```
1. import numpy as np
2.
```

```
# Method 1
file1 = open("TXT.txt")  # 打开文件
while True:
    line = file1.readline()  # 按照行来读取 txt 文件
    print(line)  # 打印每一行的内容
    if not line:
        break
file1.close()

# Method 2
for line in open("TXT.txt"):
    print(line)

# Method 3
with open("TXT.txt",'r') as f:
    data = f.read()  # 将文本内容读到一个字符串中
    print(data)

'''''
读取文件的 3 种方法：
    read()将文本中所有行读到一个字符串中去
    readline()一行一行读，在读行过程中可以跳过特定行
    readlines()将文本中所有行读到一个 list 中，文本文件每一行是 list 的一个元素
'''
a = np.loadtxt('TXT.txt')
b = a.reshape(3,3)
c = a.reshape(-1,1,3)
print(b,c)

with open('TXT.txt','r') as f:
    data = f.readlines()
    for line in data:
```

```
        item = line.split()
        #item = map(float, item)  # 将 item 中的元素匹配 float 类型
        print(float(item[1])+1)
```

2.Excel 读取

内容：



	A	B	C	D
1	1	45	65	
2	2	56	20	
3	3	52	12	
4	4	35	11	
5				
6				

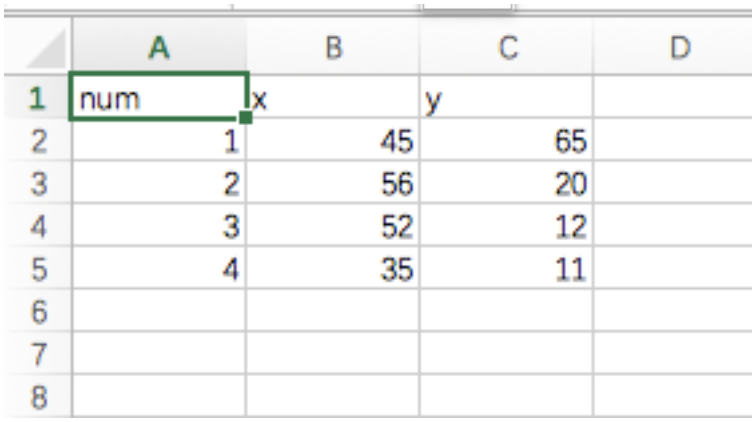
代码：

```
import os
import xlrd

data = xlrd.open_workbook('EXCEL.xlsx')
data = data.sheet_by_index(0)  # 读取索引为 0 的工作表
nrows=data.nrows  # 行数
ncols=data.ncols  # 列数
print(nrows,ncols)
col_score = data.col_values(1)  #获取第一列的值
row_values = data.row_values(0)  # 获取第 0 行的值
print(col_score,row_values)

data=xlrd.open_workbook('EXCEL.xlsx')
table=data.sheets()[0]   #索引为 0 的工作表
data_list=[]      # 新建列表
for i in range(4):
```

```
    data_list.extend(table.row_values(i))  向列表中附加内容
print(data_list)
```

3.CSV 读取

内容：



	A	B	C	D
1	num	x	y	
2	1	45	65	
3	2	56	20	
4	3	52	12	
5	4	35	11	
6				
7				
8				

代码：

```
import csv

csvFile = open("CSV.csv", "r")
reader = csv.reader(csvFile)
data = []
for item in reader:
    if reader.line_num == 1:
        continue  # 跳过第一行（在第一行为属性名时候使用）
    data.extend(item)
csvFile.close()
print(data)

csvFile = open("CSV.csv", "r")
reader = csv.reader(csvFile)
# 建立空字典
result = {}
```

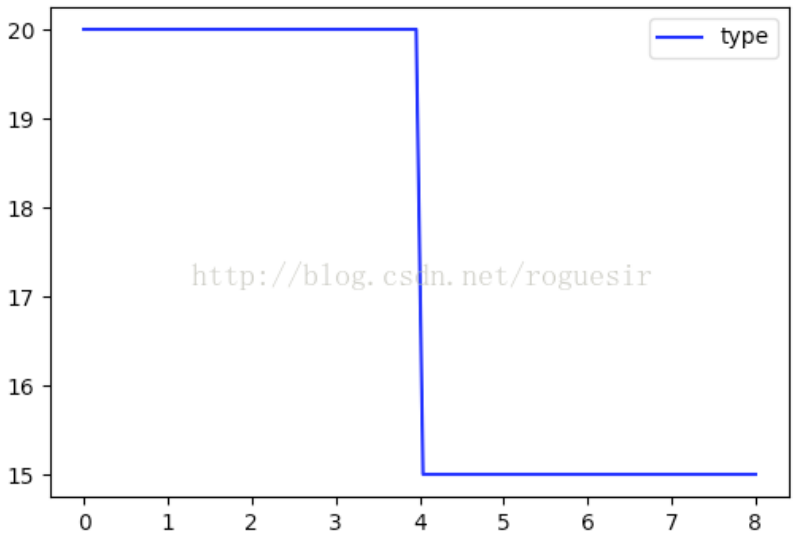
```
for item in reader:
    if reader.line_num == 1:
        continue
    result[item[1]] = item[2]

csvFile.close()
print(result)
```

Matplotlib 作图

1.散点图

2.分段函数

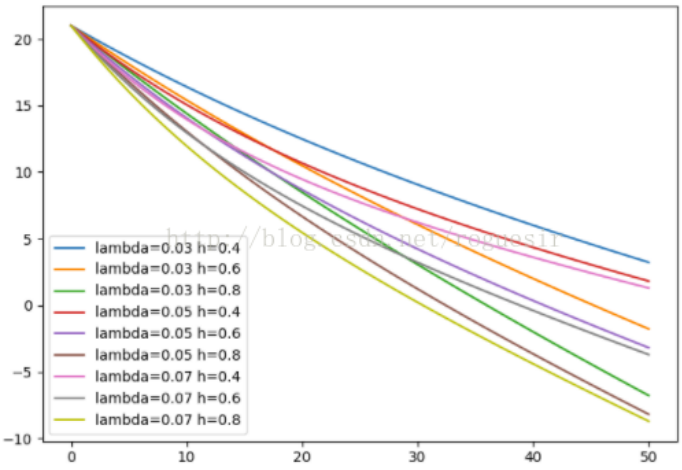
```
import numpy as np
import matplotlib.pyplot as plt  # 加载相关库

def sgn(value):  #定义分段函数
    if value < 4:
        return 20
    else:
        return 15
plt.figure(figsize=(6,4))  # 定义图框
x = np.linspace(0, 8, 100)  #定义 x 值：在[0,8 ]区间取 100 点
y = np.array([])  # 定义空数组 y
for v in x:
    y = np.append(y,np.linspace(sgn(v),sgn(v),1))
l=plt.plot(x,y,'b',label='type')
plt.legend()  # 显示图例
plt.show() # 显示图片
```



3.根据公式作图

代码：

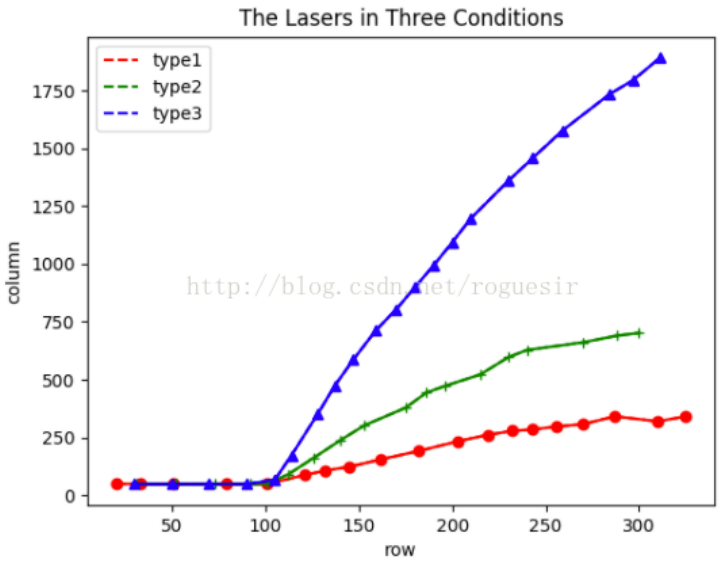
```
1. import numpy as np
2. import matplotlib.pyplot as plt
3.
4. h0 = [0.4,.6,.8] # h0 取值
5. lambda0 = [0.03,0.05,0.07] # λ 取值
6. plt.figure(figsize=(6,4)) # 设置图片大小
7. x = np.linspace(0, 50, 50) # 在[0, 50)中取 50 个点
8.
9. for i in lambda0:
10.     for h in h0:
11.         plt.plot(x, 11+(20*np.exp(-i*x)-
   h*x)/2,label='lambda='+str(i)+' h='+str(h))
12. # plt.plot(x,y,图例内容)
13. plt.legend() # 显示图例
14. plt.show() # 显示画图
```



4.折线图

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. x1=[20,33,51,79,101,121,132,145,162,182,203,219,232,243,256,270,287,310,325]
4. y1=[49,48,48,48,48,87,106,123,155,191,233,261,278,284,297,307,341,319,341]
5. x2=[31,52,73,92,101,112,126,140,153,175,186,196,215,230,240,270,288,300]
6. y2=[48,48,48,48,49,89,162,237,302,378,443,472,522,597,628,661,690,702]
7. x3=[30,50,70,90,105,114,128,137,147,159,170,180,190,200,210,230,243,259,284,297,311]
8. y3=[48,48,48,48,66,173,351,472,586,712,804,899,994,1094,1198,1360,1458,1578,1734,1797,1892]
9. x=np.arange(20,350)
10. l1=plt.plot(x1,y1,'r--',label='type1')  # 设置红色线’r--’
11. l2=plt.plot(x2,y2,'g--',label='type2')
12. l3=plt.plot(x3,y3,'b--',label='type3')
13. plt.plot(x1,y1,'ro-',x2,y2,'g+-',x3,y3,'b^-')
14. plt.title('The Lasers in Three Conditions')  # 设置标题
15. plt.xlabel('row')  # 设置横、纵坐标
```

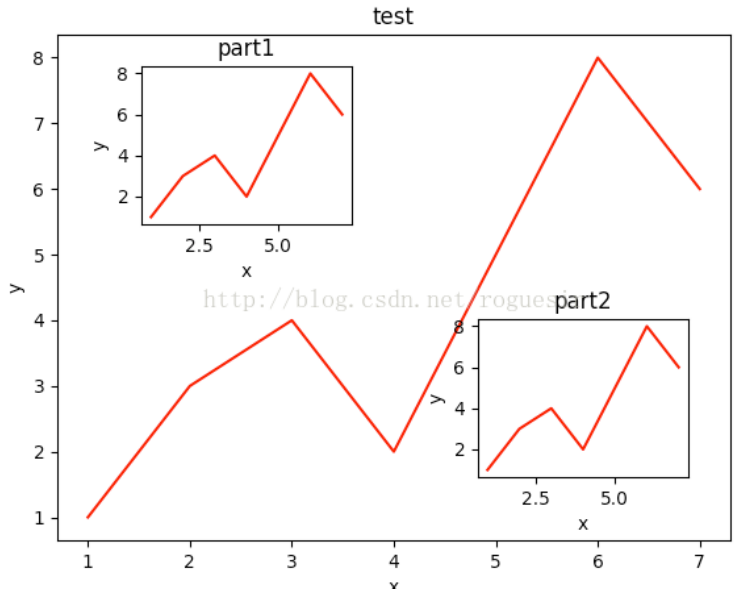
```
16. plt.ylabel('column')
17. plt.legend()
18. plt.show()
```



5.图中图

```
1. import matplotlib.pyplot as plt
2.
3. # define figure
4. fig = plt.figure()
5.
6. # define data
7. x = [1, 2, 3, 4, 5, 6, 7]
8. y = [1, 3, 4, 2, 5, 8, 6]
9.
10. left, bottom, width, height = 0.1, 0.1, 0.8, 0.8
11.   # 设置上下左右的位置比例
12. ax1 = fig.add_axes([left, bottom, width, height])
13. ax1.plot(x, y, 'r')
14. ax1.set_xlabel('x')
15. ax1.set_ylabel('y')
16. ax1.set_title('test')
```

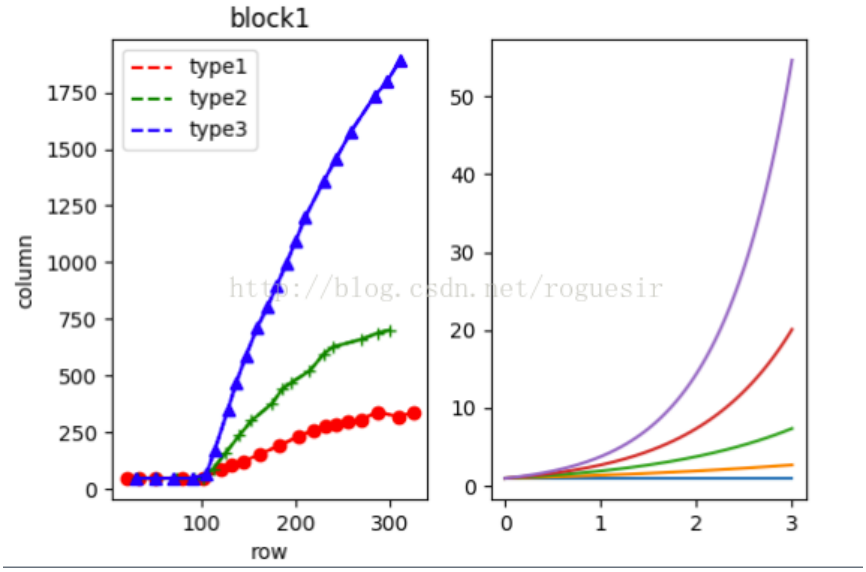
```
17.
18. # Method 1
19. left, bottom, width, height = 0.2, 0.6, 0.25, 0.25
20. ax2 = fig.add_axes([left, bottom, width, height])
21. ax2.plot(x, y, 'r')
22. ax2.set_xlabel('x')
23. ax2.set_ylabel('y')
24. ax2.set_title('part1')
25.
26. # Method 2
27. plt.axes([bottom, left, width, height])
28. plt.plot(x, y, 'r')
29. plt.xlabel('x')
30. plt.ylabel('y')
31. plt.title('part2')
32.
33. plt.show()
```



6.多图共框

```
1. import numpy as np
2. import matplotlib.pyplot as plt
```

```
3.
4. x1=[20,33,51,79,101,121,132,145,162,182,203,219,232,243,2
   56,270,287,310,325]
5. y1=[49,48,48,48,48,87,106,123,155,191,233,261,278,284,297
   ,307,341,319,341]
6. x2=[31,52,73,92,101,112,126,140,153,175,186,196,215,230,2
   40,270,288,300]
7. y2=[48,48,48,48,49,89,162,237,302,378,443,472,522,597,628
   ,661,690,702]
8. x3=[30,50,70,90,105,114,128,137,147,159,170,180,190,200,2
   10,230,243,259,284,297,311]
9. y3=[48,48,48,48,66,173,351,472,586,712,804,899,994,1094,1
   198,1360,1458,1578,1734,1797,1892]
10.
11. plt.figure(figsize=(6, 4))
12. plt.subplot(121)  # 设置框的位置，表示一行两列的第一个位置
13. l1=plt.plot(x1,y1,'r--',label='type1')
14. l2=plt.plot(x2,y2,'g--',label='type2')
15. l3=plt.plot(x3,y3,'b--',label='type3')
16. plt.plot(x1,y1,'ro-',x2,y2,'g+-',x3,y3,'b^-')
17. plt.title('block1')
18. plt.xlabel('row')
19. plt.ylabel('column')
20. plt.legend()
21.
22. # plt.subplot(n_rows, n_cols, plot_num)
23. plt.subplot(1, 2, 2)
24. x = np.linspace(0, 3, 100)
25. for i in xrange(5):
26.     plt.plot(x, np.exp(i*x/3))
27. plt.show()
```

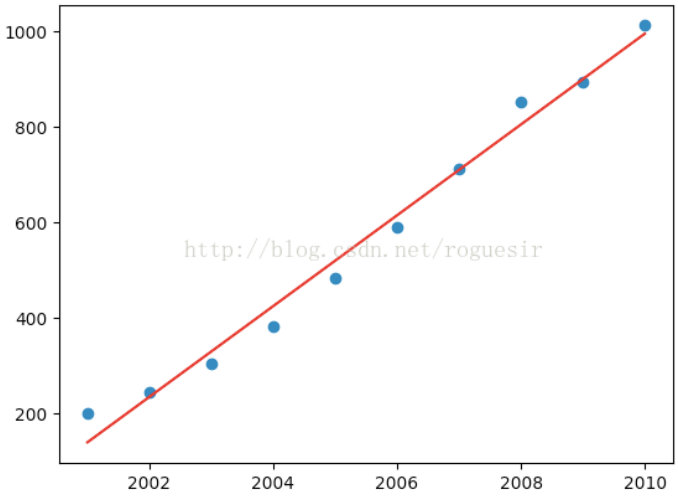


机器学习算法实现

一元回归预测

```
1. import numpy as np
2. from sklearn import linear_model
3. import matplotlib.pyplot as plt
4.
5. x_train = [[2001],[2002],[2003],[2004],[2005],[2006],[200
   7],[2008],[2009],[2010]]
6. y_train = [[202],[244],[305],[382],[484],[591],[712],[853
   ],[894],[1013]]
7.
8. x_test = [[2011]]
9.
10. linear = linear_model.LinearRegression()
11. linear.fit(x_train,y_train)
12. linear.score(x_train,y_train)
13.
14. print('Coefficient: ',linear.coef_)
15. print('Intercept: ',linear.intercept_)
```

```
16.
17. predicted = linear.predict(x_test)
18. print(predicted)
19.
20. plt.figure()
21. plt.scatter(x_train,y_train)
22. plt.plot(x_train,linear.coef_*x_train+linear.intercept_,
    'r')
23. plt.show()
```



BP 神经网络

用 BP 神经网络模型拟合方程 y=x^2

```
1. # coding:utf-8
2.
3. """
4. Author: roguesir
5. Date: 2017/8/30
6. GitHub: https://roguesir.github.com
7. Blog: http://blog.csdn.net/roguesir
8. """
9.
```

```
10. from __future__ import print_function
11. import tensorflow as tf
12. import numpy as np
13. 
14. def add_layer(inputs, in_size, out_size, activation_function=None):
15.     # add one more layer and return the output of this layer
16.     Weights = tf.Variable(tf.random_normal([in_size, out_size]))
17.     biases = tf.Variable(tf.zeros([1, out_size]) + 0.1)
18.     Wx_plus_b = tf.matmul(inputs, Weights) + biases
19.     if activation_function is None:
20.         outputs = Wx_plus_b
21.     else:
22.         outputs = activation_function(Wx_plus_b)
23.     return outputs
24. 
25. # Make up some real data
26. x_data = np.linspace(-1,1,300)[:, np.newaxis]
27. noise = np.random.normal(0, 0.05, x_data.shape)
28. y_data = np.square(x_data) - 0.5 + noise
29. 
30. # define placeholder for inputs to network
31. xs = tf.placeholder(tf.float32, [None, 1])
32. ys = tf.placeholder(tf.float32, [None, 1])
33. # add hidden layer
34. l1 = add_layer(xs, 1, 10, activation_function=tf.nn.relu)
35. # add output layer
36. prediction = add_layer(l1, 10, 1, activation_function=None)
```

```
37.
38. # the error between prediction and real data
39. loss = tf.reduce_mean(tf.reduce_sum(tf.square(ys - predi
    ction),
40.                         reduction_indices=[1]))
41. train_step = tf.train.GradientDescentOptimizer(0.1).mini
    mize(loss)
42.
43. # important step
44. if int((tf.__version__).split('.')[1]) < 12:
45.     init = tf.initialize_all_variables()
46. else:
47.     init = tf.global_variables_initializer()
48.
49. sess = tf.Session()
50. sess.run(init)
51.
52. for i in range(1000):
53.     # training
54.     sess.run(train_step, feed_dict={xs: x_data, ys: y_da
    ta})
55.     if i % 50 == 0:
56.         # to see the step improvement
57.         print(sess.run(loss, feed_dict={xs: x_data, ys:
    y_data}))
```