实验 #3
曲线拟合的最小二乘法

刘扬 2011012162

实验目标

通过使用最小二乘法对给出的两组数据进行曲线的拟合，其中曲线已经提前给出。

实验原理

对于一个函数f,我们的任务是求出一个g，其中$g \in \phi$,使得$\sum e_i^2$最小,其中$e_i = |f(x_i) - g(x_i)|$,
$\phi = span\{\phi_0, \phi_2 ... \phi_n\}$,于是我们得到了下面的推导过程:
设 $g = \sum a_i \phi_i$,

$$E = \sum_{i=0}^{m} (f(x_i) - g(x_i))^2 \tag{1}$$

$$= \sum_{i=0}^{m} (f(x_i) - \sum_{k=0}^{n} a_k \phi_k(x_i))^2 \tag{2}$$

求偏导数，得:

$$\frac{\partial E}{\partial a_k} = 2 \sum_{i=0}^{m} [\sum_{j=0}^{n} a_j \phi_j(x_i) - f(x_i)] \phi_k(x_i) \tag{3}$$

所以有下面方程成立:

$$\begin{pmatrix} (\phi_0, \phi_0) & (\phi_0, \phi_1) & \cdots & (\phi_0, \phi_n) \\ (\phi_1, \phi_0) & (\phi_1, \phi_1) & \cdots & (\phi_1, \phi_n) \\ \vdots & \vdots & \ddots & \vdots \\ (\phi_n, \phi_0) & (\phi_n, \phi_1) & \cdots & (\phi_n, \phi_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} (f, \phi_0) \\ (f, \phi_1) \\ \vdots \\ (f, \phi_n) \end{pmatrix}$$

其中我们定义$(\phi_j, \phi_k) = \sum_{i=0}^{m} \phi_j(x_i) \phi_k(x_i)$ 关于这个方程有解的条件是(左面的矩阵可逆):

1.$\phi_0(x) ... \phi_n(x)$在[a,b]上线性无关

2.$\{\phi_0 x ... \phi_n(x)\}$的线性组合中的任何一个函数在$\{x_0 ... x_m\}$中至多有n个不同的零点

算法设计与实现

对于具体的问题，我们发现$\phi_i(x) = x^i$,然后根据上面求解的原理使用python实现了OLS函数，
给入两个数据的列表，同时给出拟合函数的最高阶，求出拟合的系数值。
具体在实现OLS函数的过程中调用了写好的equation.solve函数，这个函数的作用是使用高斯消元法解方程。

实验结果

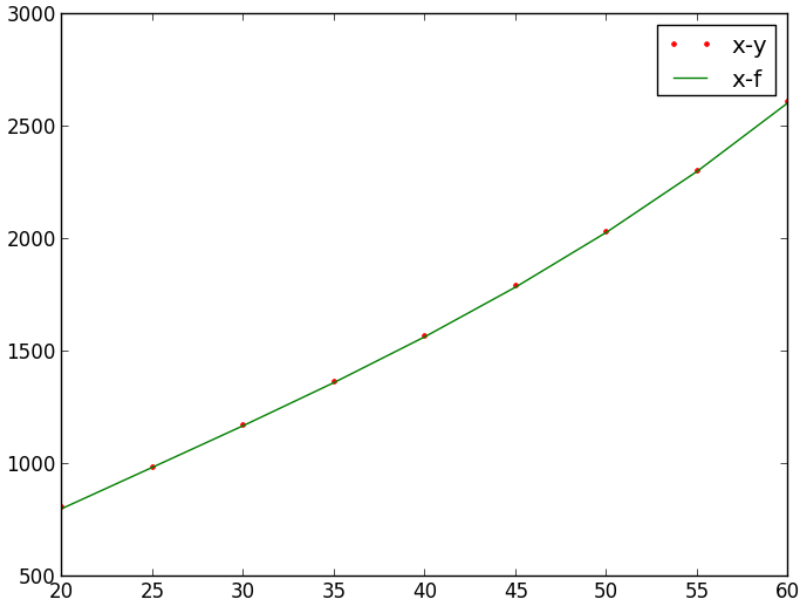
通过运行程序我得到的拟合函数是
$g(x) = -27.1212121212 + 48.8872053872x - 0.511111111111x^2 + 0.00713804713805x^3$

每一项的估计值，实际值和误差的结果为：

估计值	实际值	误差
803.282828283	805	1.71717171717
987.146464646	985	2.14646464646
1172.22222222	1170	2.22222222222
1363.86363636	1365	1.13636363636
1567.42424242	1570	2.57575757576
1788.25757576	1790	1.74242424242
2031.71717172	2030	1.71717171717
2303.15656566	2300	3.15656565657
2607.92929293	2610	2.07070707071

Table 1: 实验结果

总的误差的平方和$\sum_{i=0}^{m} e_i^2 = 40.6565656566$
散点图和曲线拟合图为:



讨论与反思

使用曲线拟合的方法我们可以方便地对一种数据进行拟合，拟合出我们想要的模型，这种思想有着很大空间的应用。比如在机器学习领域中我们的核心问题是对于给定的x,y在训练集合上如何拟合得到的结果最为精确。
尽管上面的方法能够保证求得的解为最好解，但是它的局限性还是有的：

它的拟合函数必须是给定的一族函数的线性表示

它运算的时间复杂度比较高

附:实验代码

Listing 1: 曲线拟合代码

```
#-*- coding:utf-8 -*-
#lab 3
from equation import solve
from matplotlib import pyplot as plt

def OLS(n, x, y):
    A = []
    m = len(x)
    if m!=len(y):
        return None
    for i in range(n + 1):
        A.append([])
        for j in range(n + 1):
            if i <= j:
                tmp = 0.0
                for p in range(m):
                    #for q in range(m):
                    tmp += (float(x[p]) ** i) * (float(x[p]) ** j)
                A[i].append(tmp)
            else:
                A[i].append(A[j][i])
    B = []
    for i in range(n+1):
        tmp = 0.0
        for p in range(m):
            #for q in range(m):
            tmp += (float(x[p]) ** i) * y[p]
        B.append(tmp)
    flag, answer = solve(A, B)
    return answer

if __name__=="__main__":
    x = [20, 25, 30, 35, 40, 45, 50, 55, 60]
    y = [805, 985, 1170, 1365, 1570, 1790, 2030, 2300, 2610]
    a = OLS(3, x, y)
    f = []
    for i in range(len(x)):
        tmp = 1.0
        tmp_f = 0.0
        for j in range(len(a)):
            tmp_f += tmp * a[j]
            tmp *= x[i]
        f.append(tmp_f)
    print "Answer : ","\t",
    for i in range(len(a)):
```

```
          print i,"\t",
     print
     print "\t",
     for i in range(len(a)):
          print a[i],"\t",
     print
     print "Estimation\t\t Actual Value\t\t ERROR"
     sum = 0.0
     for i in range(len(y)):
          print f[i],"\t\t",y[i],"\t\t",abs(f[i]-y[i])
          sum += abs(f[i]-y[i])**2
     print "TOTAL ERROR : ",sum

     plt.plot(x, y,'r.',label="x-y")
     plt.plot(x, f, 'g',label="x-f")
     plt.legend()
     plt.show()
```

Listing 2: 高斯消元代码

```
import copy

#A Tool using by experiment
#solve the equation AX=B
def solve(now_A, now_B):
     A = copy.deepcopy(now_A)
     B = copy.deepcopy(now_B)

     m = len(A)
     if m==0 or m!=len(B):
          return False, None
     n = len(A[0])
     if m!=n :
          return True, None
     for i in range(n):
          #change the A(i,i) to 1
          delta = float(A[i][i])
          if delta==0:
               return True, None
          for j in range(i, n):
               A[i][j] /= delta
          B[i] /= delta
          #change A(i+..,i) to 0
          for j in range(i+1, n):
               delta = A[j][i]
               for k in range(i, n):
                    A[j][k] += float(-delta) * A[i][k]
               B[j] += float(-delta) * B[i]

     x = [0.0 for i in range(n)]
     for i in range(n):
          now_line = n - i - 1
          tmp = float(B[now_line])
          for j in range(now_line+1, n):
```

```
                tmp -= x[j] * A[now_line][j]
            x[now_line] = tmp
        return True, x

#check the result of the solution
def check(A, x, B):
        m = len(A)
        n = len(A[0])
        if len(x)!=n or len(B)!=m:
            return False
        for i in range(m):
            tmp = 0.0
            for j in range(n):
                tmp += x[j] * A[i][j]
            if abs(tmp-float(B[i]))>0.001:
                return False
        return True

##check program
if __name__=="__main__":
        A=[[2,3,5.0],[-4.0,5,6],[2,4,6]]
        B=[4, 5,87]
        print solve(A, B)
        print check(A, solve(A, B), B)
```