Weighted Least Square Fitting

Prateek Bhardwaj	Monu Chaurasiya
(2020PHY1110)	(2020PHY1102)
(20068567042)	(20068567035)

S.G.T.B. Khalsa College, University of Delhi, Delhi-110007, India.

Practical Report Submitted to

Dr. Mamta

"32221401 - Mathematical Physics III"

Theory

(a) Principle of maximum likelihood is a method of obtaining the optimum values of the parameters that define a model. This is done to increase the likelihood of our model reaching the "True" model.

(b) In weighted least square method, all the y_i don't get equal weightage. Since, for each y_i we have multiple y_{ij}, we can judge the value of error in $\bar{y}_i$, and assign it a weight based on this calculation. In this way, $\bar{y}_i$ with lower error contribute more to our model.

(i) Principle of maximum likelihood is all about obtaining the optimum value of the parameters. In weighted least square fitting, by assigning weights to each y_i, we are optimizing the values of the parameter so that we have a higher likel-ihood of reaching the "True" model.

(ii) Consider a data set $\{(x_i, [y_{ij}]\}$ where x_i are known exactly and σ_i is the value of error in $\bar{y}_i$ and $\bar{y}_i$ is the mean of $[y_{ij}]$ for each i.

Let $y = f(x : m, c)$ are set of parameter to be estimated. Then from central limit theorem, distribution of measured 'y' values about their ideal values is gaussian, and the probability of a particular y_i for a given x_i is

$$P(y_i : m, c) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left\{ -\frac{(y_i - f(x_i : m, c))^2}{2\sigma_i^2} \right\}$$

Then, maximising maximum likelihood function of the estimators $\hat{m}$ and $\hat{c}$ is similar to minimizing.

$$\sum_i \left[\frac{y_i - f(x : m, c)}{\sigma_i} \right]^2$$

This term is called χ^2

$$\therefore \chi^2 = \left[\frac{y_i - f(x : m, c)}{\sigma_i} \right]^2$$

Then, we will minimize χ^2 w.r.t 'm' and 'c'.

So in linear W.L.S where, $y = mx + c$

We will use χ^2 to find estimates for m and c

i.e $\chi^2 = \sum w_i (y_i - m x_i - c)^2$

here, $w_i = \frac{1}{\sigma_i^2}$

Thus, $\frac{d(\chi^2)}{dm} = 0$

$$\Rightarrow \sum \frac{d[w_i (y_i - m x_i - c)^2]}{dm} = 0$$

$$\Rightarrow -2 \sum w_i x_i (y_i - m x_i - c)^2 = 0$$

$$\Rightarrow \sum w_i x_i y_i - m \sum w_i x_i^2 - c \sum w_i x_i = 0 \quad \text{--- (1)}$$

$$\Rightarrow \quad c = \frac{\sum w_i y_i - m \sum w_i x_i}{\sum w_i}$$

$$\Rightarrow c = \bar{y} - m\bar{x}$$

Here, $\bar{y} = \frac{\sum w_i y_i}{\sum w_i}$, $\bar{x} = \frac{\sum w_i x_i}{\sum w_i}$

Putting c in ①

$$\sum w_i x_i y_i - m\sum w_i x_i^2 - (\bar{y} - m\bar{x})\sum w_i x_i = 0$$

$$m = \frac{\sum w_i x_i y_i - \bar{y}\sum w_i x_i}{\sum w_i x_i^2 - \bar{x}\sum w_i x_i}$$

$$\therefore m = \frac{S_{xy} - \bar{y} S_x}{S_{x^2} - \bar{x} S_x} = \frac{\sum w_i (x_i - \bar{x})(y_i - \bar{y})}{\sum w_i (x_i - \bar{x})^2}$$

Where,

$$S_{xy} = \sum w_i x_i y_i$$

$$S_x = \sum w_i x_i$$

$$S_{x^2} = \sum w_i x_i^2$$

So we can finally write

$$m = \frac{\sum w_i \sum w_i x_i y_i - \sum w_i x_i \sum w_i y_i}{\sum w_i \sum w_i x_i^2 - (\sum w_i x_i)^2}$$

and

$$c = \frac{\sum w_i x_i^2 \sum w_i y_i - \sum w_i x_i \sum w_i x_i y_i}{\sum w_i \sum w_i x_i^2 - (\sum w_i x_i)^2}$$

Let $\Delta = \sum w_i \sum w_i x_i^2 - (\sum w_i x_i)^2$

$\therefore$ This m depends on x_i and y_i but only y_i have error, by propagation of error.

$$\sigma_m^2 = \sum \left(\frac{dm}{dy_i}\sigma_i\right)^2$$

i.e
$$\frac{dm}{dy_i} = \frac{(\sum w_i) w_i x_i - (\sum w_i x_i) w_i}{\Delta}$$

then, $\left(\frac{dm}{dy_i}\right)\sigma_i = \frac{(\Sigma w_i) x_i/\sigma_i - \Sigma w_i x_i/\sigma_i}{\Delta}$

$$\sigma_m^2 = \frac{\Sigma\left(\frac{(\Sigma w_i) x_i}{\sigma_i} - \frac{\Sigma w_i x_i}{\sigma_i}\right)^2}{\Delta^2}$$

$$\sigma_m^2 = \frac{\Sigma\left[\frac{(\Sigma w_i)^2 x_i^2}{\sigma_i^2} + \frac{(\Sigma w_i x_i)^2}{\sigma_i^2} + \frac{2(\Sigma w_i \Sigma w_i x_i) x_i}{\sigma_i^2}\right]}{\Delta^2}$$

$$\sigma_m^2 = \frac{\Sigma w_i\left[(\Sigma w_i)^2 x_i^2 + (\Sigma w_i x_i)^2 + 2(\Sigma w_i \Sigma w_i x_i) x_i\right]}{\Delta^2}$$

$$\sigma_m^2 = \frac{\Sigma w_i\left[(\Sigma w_i)^2 x_i^2 + \left[(\Sigma w_i)\bar{x}\right]^2 - 2\left[(\Sigma w_i)^2 x_i\right]\right]}{\Delta^2}$$

$$\sigma_m^2 = \Sigma w_i\left[(\Sigma w_i)^2 (x_i - \bar{x})^2\right]/\Delta^2$$

$$\sigma_m^2 = \frac{(\Sigma w_i)\Sigma w_i (x_i - \bar{x})^2}{\Delta^2}$$

$$\sigma_m^2 = \frac{(\Sigma w_i)\Sigma w_i (x_i - \bar{x})^2}{\Delta\left[\Sigma w_i(\Sigma w_i x_i^2 - \bar{x}\Sigma w_i x_i)\right]}$$

$$\sigma_m^2 = \frac{\Sigma w_i}{\Delta} \Rightarrow \sigma_m = \sqrt{\frac{\Sigma w_i}{\Delta}}$$

Similarly, $\sigma_i^2 = \Sigma\left(\frac{dc}{dy_i}\sigma_i\right)^2$

Hence,

$$\frac{dc}{dy} = \frac{(\Sigma w_i x_i^2) w_i - (\Sigma w_i x_i) w_i x_i}{\Delta}$$

Using similar steps,

$$\sigma_c^2 = \frac{\sum w_i x_i^2}{\Delta} = \frac{S_{x}^2}{\Delta}$$

$$\sigma_c = \sqrt{\frac{S_x^2}{\Delta}}$$

(iii)

When our data points are equally distributed along their best estimates, i.e,

$$\sigma_i = \sigma \quad \forall i$$

Then our data points provides equally precise information about deterministic part of the process.
i.e for our all the values of explanatory variables standard deviation is constant then,

$$w_i = 1 \quad \forall i$$

Being Constant can be written out of summation

$$m = \frac{w \sum (x_i - \bar{x})(y_i - \bar{y})}{w \sum (x_i - \bar{x})^2}$$

$$\boxed{m = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}}$$

$$\boxed{c = \bar{y} - m\bar{x}}$$

(c) The correlation coefficient is the specific measure that quantifies the strength of the linear relationship between two variables in a correlation analysis.
The adjusted correlation coefficient is an adjustment & for the correlation coefficient that takes into account the no. of variables in a data set, and is optimized for getting closer to the "true" model by weighing the variables according to error in them.

1 Programming

```


'''
NAME - Prateek Bhardwaj
ROLL NO - 2020PHY1110

PARTNER:
NAME - Monu Chaurasiya
ROLL NO - 2020PHY1102
'''

import numpy as np
import matplotlib.pyplot as plt
import cmath
import csv
from scipy.stats import linregress
from statistics import stdev,variance
import pandas as pd
from prettytable import PrettyTable

print("Name - Prateek Bhardwaj \n Roll No. 2020PHY1110")
#2 PROGRAMMING
#(a)
#Ordinary Least squares fitting
def lsqf(x,y):
    n1=len(x)
    n2=len(y)
    slope=0
    intercept=0
    if n1==n2:
        sigma_xi=0
        sigma_yi=0
        sigma_xi_yi=0
        sigma_xisq=0
        sigma_yisq=0
        sse=0
        Rs=0
        count=0
        SSxx=0
        while count<n1:
            sigma_xi=sigma_xi+x[count]                              #SUM OF ALL X ELEMENTS
            sigma_yi=sigma_yi+y[count]                              #SUM OF ALL Y ELEMENTS
            sigma_xi_yi=sigma_xi_yi+x[count]*y[count]               #SUM OF X*Y ELEMENTS
            sigma_xisq=sigma_xisq+x[count]**2                       #SUM OF X**2
            sigma_yisq=sigma_yisq+y[count]**2                       #SUM OF Y**2
            count=count+1
        slope=(sigma_xi*sigma_yi-n1*sigma_xi_yi)/(sigma_xi**2-n1*
```

```
sigma_xisq)                    #CALCULATES SLOPE
        intercept=(sigma_xisq*sigma_yi-sigma_xi*sigma_xi_yi)/(n1*sigma_xisq-sigma_xi**2)    #CALCULATESINTERCEPT
        r=cmath.sqrt(((sigma_xi_yi)**2)/(sigma_xisq*sigma_yisq)) #COEFFICIENT OF CORRELATION
        c=np.array([intercept]*n1)
        xm=np.dot(slope,x)
        y_calc=xm+c
        x_bar=np.mean(x)        #MEAN OF X
        y_bar=np.mean(y)        #MEAN OF Y
        err_y=y-y_calc
        for i in range(len(y)):
            sse+=(y[i]-y_calc[i])**2  #SUM OF RESIDUAL SQUARES
            Rs+=(y[i]-y_calc[i])        #SUM OF RESIDUALS
        SSxx = sigma_xisq - ((sigma_xi)**2/n1)
        std_slope = cmath.sqrt((sse)/(SSxx * (n1-2)))     #standard deviation of slope
        std_intercept = np.sqrt(((std_slope)**2)*(sigma_xisq/n1))     #standard deviation of intercept
        return y_calc,[slope,intercept,std_slope,std_intercept,r,Rs,sse]
        # y_calc,slope,intercept,std_slope,std_intercept,r=#COEFFICIENT OF CORRELATION,Rs=SUM OF RESIDUALS,sse=SUM OF RESIDUAL SQUARES

#Weighted Least Squares Fitting
def wlsf(x,y,w):
    n1=len(x)
    n2=len(y)
    slope=0                        #slope
    intercept=0                #intercept
    e_s=0                          #error in slope
    e_i=0                          #error in intercept
    i=0
    r = 0                          # correlation  coefficient
    sse=0     #SUM OF RESIDUAL SQUARES
    Rs=0      #SUM OF RESIDUALS
    if n1==n2 and n1>3:
        S_wi_xi=0
        S_wi_yi=0
        S_wi_xi_yi=0
        S_wi_xisq=0
        S_wi=0
        x_mean = 0
        y_mean = 0
        Sxx =0
        Syy=0
        Sw=sum(w)
        while i < n1:
                   x_mean += x[i]*w[i]/Sw
                   y_mean += y[i]*w[i]/Sw
                   i = i+1
        count=0
        while count<n1:
            S_wi_xi=S_wi_xi+ x[count]*w[count]
```

```
            S_wi_yi= S_wi_yi+y[count]*w[count]
            S_wi_xi_yi=S_wi_xi_yi+ x[count]*y[count]*w[count]
            S_wi_xisq=S_wi_xisq+(x[count]**2)*w[count]
            S_wi=S_wi+w[count]
            Sxx = Sxx + w[count]*(x[count]- x_mean)**2
            Syy =  Syy +w[count]*(y[count] - y_mean)**2
            count+=1


        # SLOPE , INTERCEPT AND CORRELATION COEFFICIENT
        intercept=(S_wi_xisq*S_wi_yi-S_wi_xi*S_wi_xi_yi)/(S_wi*
    S_wi_xisq-S_wi_xi**2)
        slope=(S_wi*S_wi_xi_yi-S_wi_xi*S_wi_yi)/(S_wi*S_wi_xisq-
    S_wi_xi**2)
        r = (slope*np.sqrt(Sxx))/(np.sqrt(Syy))


        #DEFINING CORRESPONDING BEST FITTED Y VALUE FOR X

        c=np.array([intercept]*n1)
        xm=np.dot(slope,x)
        y_calc=xm+c                     # Best Fitted y

        # ERROR IN SLOPE , INTERCEPT

        e_s=((S_wi)/(S_wi*S_wi_xisq - S_wi_xi**2))**(1/2)
        e_i=((S_wi_xisq)/(S_wi*S_wi_xisq-S_wi_xi**2))**(1/2)
        for i in range(len(y)):
            sse+=(y[i]-y_calc[i])**2  #SUM OF RESIDUAL SQUARES
            Rs+=(y[i]-y_calc[i])       #SUM OF RESIDUALS


        return y_calc,[slope,intercept,e_s,e_i,r,Rs,sse]
        #y_calc,slope,intercept,e_s=error in slope,e_i=error in
    intercept,r=correlation coefficient,Rs=sum of residuals,sse=sum of
    residual squares

#d
# (i)
m=[];t1=[];t2=[];t3=[];t4=[];t5=[];t6=[];t7=[];t8=[];t9=[];t10=[]
with open('1110.csv','r') as csv_file:
    csv_reader = csv.reader(csv_file)       #reading the values from
    csv file and making list of them.
    next(csv_reader)
    for line in csv_reader:
        m.append(line[1])
        t1.append(line[2])
        t2.append(line[3])
        t3.append(line[4])
        t4.append(line[5])
        t5.append(line[6])
        t6.append(line[7])
        t7.append(line[8])
        t8.append(line[9])
        t9.append(line[10])
```

```
        t10.append(line[11])

M=[];T1=[];T2=[];T3=[];T4=[];T5=[];T6=[];T7=[];T8=[];T9=[];T10=[]
for i in range(len(m)):                 #to convert elements of list from
    str into float
     M.append(float(m[i]))
     T1.append((float(t1[i])**2))
     T2.append((float(t2[i])**2))
     T3.append((float(t3[i])**2))
     T4.append((float(t4[i])**2))
     T5.append((float(t5[i])**2))
     T6.append((float(t6[i])**2))
     T7.append((float(t7[i])**2))
     T8.append((float(t8[i])**2))
     T9.append((float(t9[i])**2))
     T10.append((float(t10[i])**2))

T=[]

for i in range(0,10):
     s1=np.mean([T1[i],T2[i],T3[i],T4[i],T5[i],T6[i],T7[i],T8[i],T9[i],
    T10[i]])
     T.append(s1)

T=np.array(T)

hh=np.array([np.array(T1),np.array(T2),np.array(T3),np.array(T4),np.
    array(T5),np.array(T6),np.array(T7),np.array(T8),np.array(T9),np.
    array(T10)]).reshape(10,10)
dd=hh.T
w = []
err = []
for i in range(0,10):
  w.append(1/stdev(dd[i])**2)
  err.append(stdev(dd[i]))
s_s=np.column_stack((M,T,w))
np.savetxt("1110.txt", s_s,header = "xi, yi, wi")

#(ii)
pr1=lsqf(M,T)
pr=wlsf(M,T,w)

#(iii)
plt.scatter(M,T,label="Actual",c="magenta")
plt.plot(M,pr1[0],label="Best Fitted")
plt.xlabel("Mass (g)")
plt.ylabel("$T^2$ ($s^2$)")
plt.title("Ordinary Least Square Fitting")
plt.legend()
plt.grid()
plt.savefig("1110_OLSF.pdf")
plt.show()


plt.errorbar(M,T,yerr=err,xerr=None,fmt='o',ecolor = 'red',color='
```

```
    black')
plt.plot(M,pr[0],label="Best Fitted")
plt.xlabel("Mass (g)")
plt.ylabel("$T^2$ ($s^2$)")
plt.title("Weighted Least Square Fitting")
plt.legend()
plt.grid()
plt.savefig("1110_WLSF.pdf")
plt.show()

plt.scatter(M,T,label="Actual",c="magenta")
plt.plot(M,pr1[0],label="OLS",c="orange")
plt.plot(M,pr[0],label="WLS",linestyle="dashed",c="green")
plt.xlabel("Mass (g)")
plt.ylabel("$T^2$ ($s^2$)")
plt.title("OLSF vs WLSF")
plt.legend()
plt.grid()
plt.savefig("1110_WLSF_vs_OLSF.pdf")
plt.show()
sl, intec, r, p, se =linregress(M, T)

#(e)
k_ols = 4*np.pi*np.pi/pr1[1][0]
k_wls = 4*np.pi*np.pi/pr[1][0]
k_linregress = 4*np.pi*np.pi/sl
m_ols = (k_ols*pr1[1][1])/(4*np.pi*np.pi)
m_wls = (k_wls*pr1[1][1])/(4*np.pi*np.pi)
m_linregress = (k_linregress*intec)/(4*np.pi*np.pi)
err_k_ols=(pr1[1][2]/pr1[1][0])*k_ols
err_k_wls=(pr[1][2]/pr[1][0])*k_wls
err_m_ols=((pr1[1][3]/pr1[1][1])+(err_k_ols/k_ols))*m_ols
err_m_wls=((pr[1][3]/pr[1][1])+(err_k_wls/k_wls))*m_wls
header = ["Parameter", "OLSF", "WLSF", "linregress fn"]
myTable = PrettyTable(header)
myTable.add_row(["k",k_ols,k_wls,k_linregress ])
myTable.add_row(["Error in k",err_k_ols,err_k_wls, ""])
myTable.add_row(["m",m_ols,m_wls,m_linregress])
myTable.add_row(["Error in m",err_m_ols,err_m_wls,""])
print(myTable)

#d(iv),(f)

y_calc_inbuilt=[]
for i in range(0,10):
    y_calc_inbuilt.append(sl*M[i] +intec)
S_RS=0;S_R=0
for i in range(10):
    S_RS+=(T[i]-y_calc_inbuilt[i])**2  #SUM OF RESIDUAL SQUARES
    S_R+=(T[i]-y_calc_inbuilt[i])        #SUM OF RESIDUALS
array_3=[sl,intec,"","",r,S_R,S_RS]


para=["SLOPE","INTERCEPT","ERROR IN SLOPE","ERROR IN INTERCEPT","
    CORRELATION COEFFICIENT","SUM OF RESIDUALS","SUM OF RESIDUAL
```

```
    SQUARES"]
DATA={"PARAMTER":para,"OLSF":pr1[1],"WLSF":pr[1],"lingress fn":array_3
    }
print(pd.DataFrame(DATA))
```

2 Result

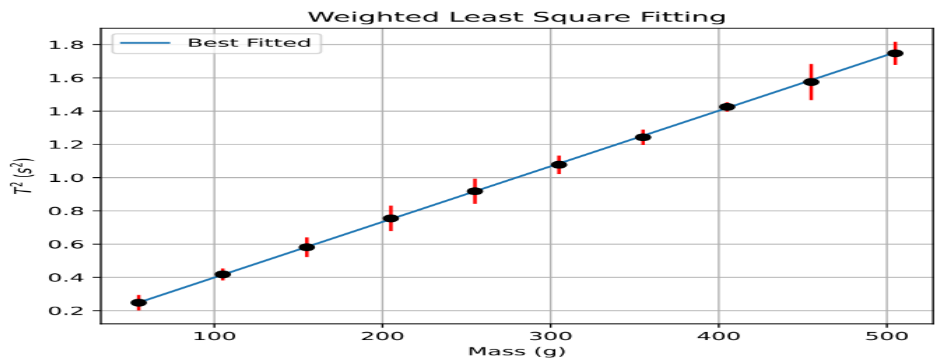
Name -

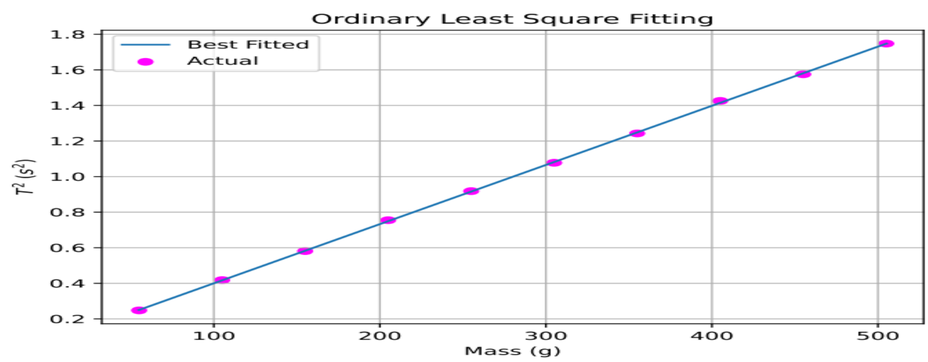
Monu Chaurasiya
Roll No. 2020PHY1102

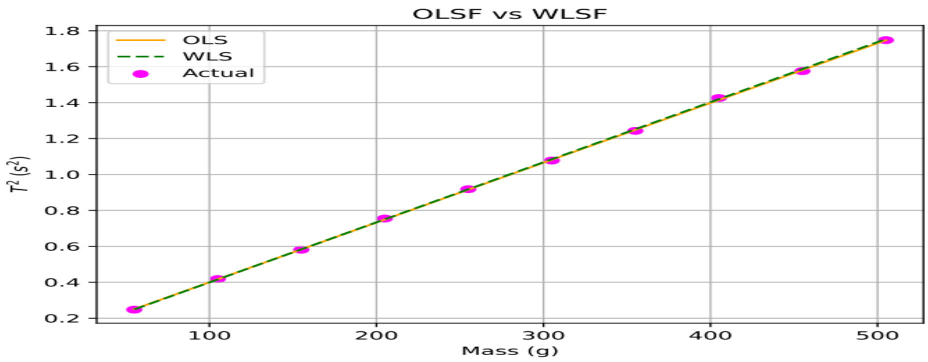
Prateek Bhardwaj
Roll No. 2020PHY1110

Parameter	OLSF	WLSF	linregress fn
k	11867.309570533085	11803.930677874625	11867.309570533078
Error in k	(44.00708444701446+0j)	376.01594233547087	
m	20.26838853676536	20.160142601678164	20.268388536765094
Error in m	(1.2420849603046908+0j)	10.868822258767166	

	PARAMTER	OLSF	WLSF	lingress fn
0	SLOPE	3.326653e-03+0.000000e+00j	0.003345	0.003327
1	INTERCEPT	6.742589e-02+0.000000e+00j	0.064898	0.067426
2	ERROR IN SLOPE	1.233610e-05+0.000000e+00j	0.000107	
3	ERROR IN INTERCEPT	3.881952e-03+0.000000e+00j	0.032921	
4	CORRELATION COEFFICIENT	9.996035e-01+0.000000e+00j	0.999933	0.999945
5	SUM OF RESIDUALS	-2.164935e-15+0.000000e+00j	-0.024732	0.0
6	SUM OF RESIDUAL SQUARES	2.510958e-04+0.000000e+00j	0.000378	0.000251






OLSF vs WLSF
OLS
WLS
Actual
$T^2\,(s^2)$
Mass (g)
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
100
200
300
400
500