Assignment 3: Fitting Data to Models

Manognya Param Koolath

February 12, 2019

The task

A dataset consisting of differently noisy data, each consisting of 101 data points is to be fitted in models minimizing the mean squared errors. The results are to be analyzed by plotting different plots for visualisation.

Dataset format

The dataset has 101 rows and 10 columns. The first column is the independent variable, which will be referred as x hereafter. The next 9 columns are the data corresponding to x, with different amount of noise (hereafter called as σ). We'll call them as y_1, y_2, ..., y_9. We are given the amount of noise in each column of the data, which we have to use to interpret the results.

Procedure

We use a linear model to fit the given data. The following is our linear function.

$$y = C_1 * Jn_2(x) + C_2 * x$$

Since we have x, we can get the predicted values of y for each C_1 and C_2. We have to optimize the values of C_1 and C_2 such that the mean squared errors for each y_i is minimum.

Python code

The code is properly commented. All explanations are commented out normally and statements used for debugging purposes are commented out using

```
##
```

Assignment 3 Code

```
##########################################
#                                        #
# Assignment 3                           #
#                                        #
# Author : Manognya Param Koolath        #
# Date completed : 10 th Feb 2019        #
#                                        #
##########################################


import numpy as np
import matplotlib.pyplot as plt
import scipy.special as sp
from scipy.linalg import lstsq
file_data = np.loadtxt("fitting.dat")
## print(file_data)

# Our funtion approximation:
def fitting_func_g(t,C1,C2):
        res=C1*sp.jn(2,t)+C2*t
        return res

correct_C1 = 1.05
correct_C2 = (-0.105)
x_real = np.linspace(0,10,101)
y_real = fitting_func_g(x_real,correct_C1,correct_C2)


# Plotting data that has to be fitted:
plt.plot(x_real,file_data[:,1],label='standard_deviation = 0.100')
plt.plot(x_real,file_data[:,2],label='standard_deviation = 0.056')
plt.plot(x_real,file_data[:,3],label='standard_deviation = 0.032')
plt.plot(x_real,file_data[:,4],label='standard_deviation = 0.018')
plt.plot(x_real,file_data[:,5],label='standard_deviation = 0.010')
plt.plot(x_real,file_data[:,6],label='standard_deviation = 0.006')
plt.plot(x_real,file_data[:,7],label='standard_deviation = 0.003')
plt.plot(x_real,file_data[:,8],label='standard_deviation = 0.002')
plt.plot(x_real,file_data[:,9],label='standard_deviation = 0.001')
plt.plot(x_real,y_real,label='true value')
plt.legend(loc='upper right')
plt.xlabel(r'$t$',size=20)
plt.ylabel(r'$g(t)+noise$',size=20)
```

```
plt.title(r'The true value and the data to be fitted')
plt.show()

y_first_column = file_data[:,1]
## std_dev=np.std(y_first_column) #--> gives the standard deviation of y_
# We need the standard deviation of the noise, which we know is 0.1
## print(x_real.shape, y_first_column.shape)

plt.errorbar(x_real[::5], y_first_column[::5],0.1,fmt='ro',markersize=5, l
# Used the std_dev of noise. Idk if the plot is right.
## plt.errorbar(x_real[::5], y_first_column[::5], std_dev,fmt='ro--',marke
plt.plot(x_real, y_real, label="Real function")
plt.legend(loc='upper right')
plt.xlabel(r'$t$',size=20)
plt.ylabel(r'$g(t)$',size=20)
plt.title(r'Real function and the first noisy data (standard deviation =
plt.show()

J2_x_real = sp.jn(2,x_real)
## print(J2_x_real.shape)
M = np.c_[J2_x_real, x_real]
# We know A_0=1.05, B_0=(-0.105)
C_0 = np.array([correct_C1, correct_C2])
M_p = np.dot(M, C_0)
## print(M.shape, M_p.shape)
## print(M_p.shape, y_real.shape)
# This should be compared with y_real.
if np.array_equal(M_p, y_real):
        print("Verified that they are both same.")
else:
        print("Debug the code. :P ")

A_values = np.arange(0,2.01,0.1)
B_values = np.arange(-0.2,0.01,0.01)
A_val , B_val = np.meshgrid(A_values, B_values)
## print(A_val.shape)
## print(B_val.shape)
A_size = A_values.size
B_size = B_values.size
A_val_3d = A_val[:,:,np.newaxis]
B_val_3d = B_val[:,:,np.newaxis]
## print(A_val_3d.shape, B_val_3d.shape)
## print(A_size, B_size)
## print((A_values*B_values).shape)
```

```
## g_first_column = fitting_func_g(x_real, A_values, B_values)
## print(g_first_column.shape)

# Vectorised implementation:
errors = (np.square(y_first_column - fitting_func_g(x_real, A_val_3d, B_v
errors = np.transpose(errors)
## print(errors.shape)

# Not vectorised:
# error = np.full((A_size,B_size),0.0)
# for a in range (A_size):
#       for b in range (B_size):
#               error[a][b]=(np.square(y_first_column - fitting_func_g(x_
##              # print(error[a][b])
## print(error.shape)

#Contour plot:
plt.figure()
cp = plt.contour(A_val,B_val,errors)
# plt.legend(loc='center right')
plt.clabel(cp,cp.levels[0:5],inline=True)
plt.title(r'Contour plot of mean squared error of first column of data')
# plt.plot(correct_C1,correct_C2,'bo',label = 'minimum')
plt.scatter(correct_C1,correct_C2,color='r')
plt.xlabel('A')
plt.ylabel('B')
plt.annotate("True value",xy=(correct_C1,correct_C2))
plt.show()

sigma=np.logspace(-1,-3,9) # noise stdev
## print(sigma)

#Solution for all columns:
sol = np.full((0,2),0.0)
for i in range(1,10):
        y_column = file_data[:,i]
        sol_i = lstsq(M,y_column)[0]
##      print(sol_i.shape)
        sol=np.concatenate((sol,sol_i[None,:]),axis=0)
## print(sol)
real_ans = np.array([correct_C1,correct_C2])
## print(real_ans)
observed_error = abs(sol - real_ans)
## print(observed_error)
```

```
error_A = observed_error[:,0]
error_B = observed_error[:,1]
## print(error_A)
## print(error_B)

#Plotting errors with stdev of noise:
plt.plot(sigma,error_A,'ro--',label='Error in A',linewidth=1.0)
plt.plot(sigma,error_B,'go--',label='Error in B',linewidth=1.0)
plt.legend(loc='upper left')
plt.xlabel(r'Standard Deviation of the noise',size=20)
plt.ylabel(r'Error',size=20)
plt.title(r'Variation of errors with noise')
plt.show()

# Log log plot:
plt.loglog(sigma,error_A,'ro--',label='Error in A')
plt.loglog(sigma,error_B,'go--',label='Error in B')
plt.legend(loc='upper left')
plt.xlabel(r'Standard Deviation of the noise',size=15)
plt.ylabel(r'Error',size=15)
plt.title(r'Error in the least squares estimates (log log plot)')
plt.show()

# Plots for understanding and exploring further patterns:
# plt.plot(sigma,error_A)
# plt.xlabel(r'Standard Deviation of the noise',size=20)
# plt.ylabel(r'Error in estimation of A',size=20)
# plt.title(r'Error in the least squares estimate of A')
# plt.show()

# plt.plot(sigma,error_B)
# plt.xlabel(r'Standard Deviation of the noise',size=20)
# plt.ylabel(r'Error in estimation of B',size=20)
# plt.title(r'Error in the least squares estimate of B')
# plt.show()

# plt.loglog(sigma,error_A)
# plt.xlabel(r'Standard Deviation of the noise',size=20)
# plt.ylabel(r'Error in estimation of A',size=20)
# plt.title(r'Error in the least squares estimate of A (log log plot)')
# plt.show()

# plt.loglog(sigma,error_B)
# plt.xlabel(r'Standard Deviation of the noise',size=20)
```

```
# plt.ylabel(r'Error in estimation of B',size=20)
# plt.title(r'Error in the least squares estimate of B (log log plot)')
# plt.show()

# plt.semilogx(sigma,error_A)
# plt.xlabel(r'Standard Deviation of the noise',size=20)
# plt.ylabel(r'Error in estimation of A',size=20)
# plt.title(r'Error in the least squares estimate of A (semi log plot)')
# plt.show()

# plt.semilogx(sigma,error_B)
# plt.xlabel(r'Standard Deviation of the noise',size=20)
# plt.ylabel(r'Error in estimation of B',size=20)
# plt.title(r'Error in the least squares estimate of B (semi log plot)')
# plt.show()

# plt.semilogy(sigma,error_A)
# plt.xlabel(r'Standard Deviation of the noise',size=20)
# plt.ylabel(r'Error in estimation of A',size=20)
# plt.title(r'Error in the least squares estimate of A (semi log plot)')
# plt.show()

# plt.semilogy(sigma,error_B)
# plt.xlabel(r'Standard Deviation of the noise',size=20)
# plt.ylabel(r'Error in estimation of B',size=20)
# plt.title(r'Error in the least squares estimate of B (semi log plot)')
# plt.show()

####################
#     END OF FILE    #
####################
```

Results and explanations

The noisy dataset was plotted after loading into the program. The plot is as shown in Figure 1. We can observe the deviations of noisy curves from the curve which denotes the true function values. We take the following function to fit the noisy data:

$$g(t; A, B) = AJ_2(x) + Bx$$

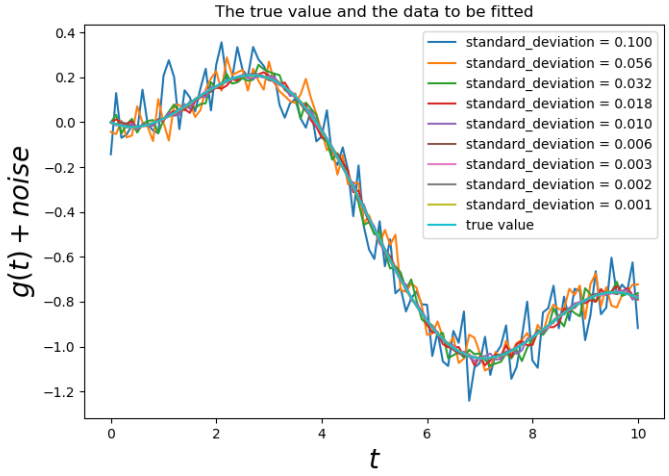

Figure 1: The smooth curve is the true function to which different noises are added and the other curves are the ones to be fitted

We have used a function in our code to evaluate this at any A and B. The plot with the first column of data ($\sigma = 0.10$) with error bars is shown in Figure 2. This is plotted with every 5^{th} data item to make the plot readable. Next we constructed the matrix M as verified that this matrix multiplied

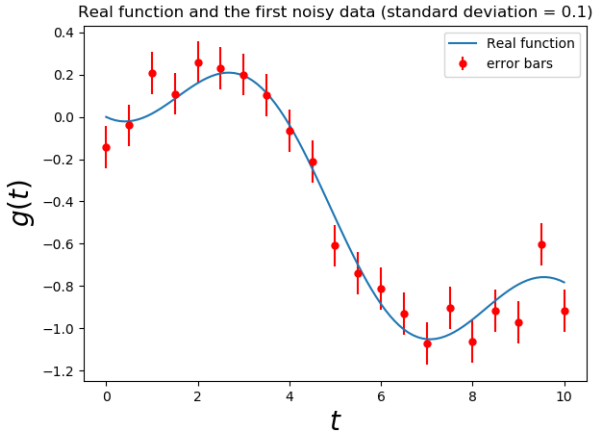

Figure 2: The original curve with errorbounds for every fifth data point of the first noisy data with $\sigma = 0.1$

with the vector (A_0, B_0) is equivalent to the function `fitting_func_g()` evaluated at (A_0, B_0). The mean squared error for various values was calculated in a completely vectorized way and the contour plot is plotted as shown in Figure 3. From this plot, we understand that there is only one

minimum for this error function and it is also shown in this plot.

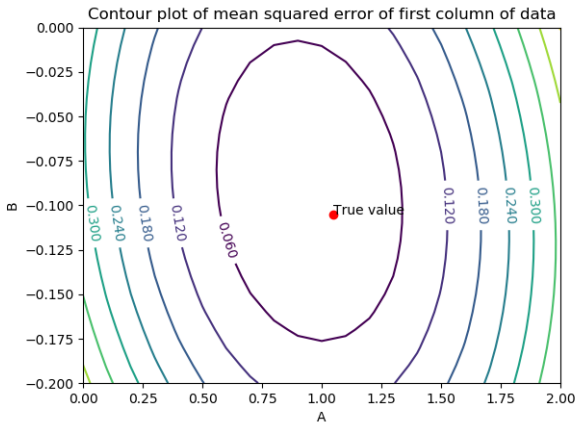

Figure 3: The contour plot obtained by calculating the mean squared error of $21 * 21 = 441$ ponts in the A-B plane. The blue dot represents the minimum which is the expected solution of the optimization problem

We used the python function `lstsq` from `scipy.linalg` to obtain the best estimate of A and B, minimizing the mean squared error. Similarly, optimum values are found for other y_is as well. We know the optimum value of A and B for the actual data withou noise. We take the difference between this value and the estimated A and B as the error. This error for each data files is plotted against the standard deviation of the noise σ in Figure 4. As it is evident from Figure 4, the error in the estimate is not growing linearly with the noise.

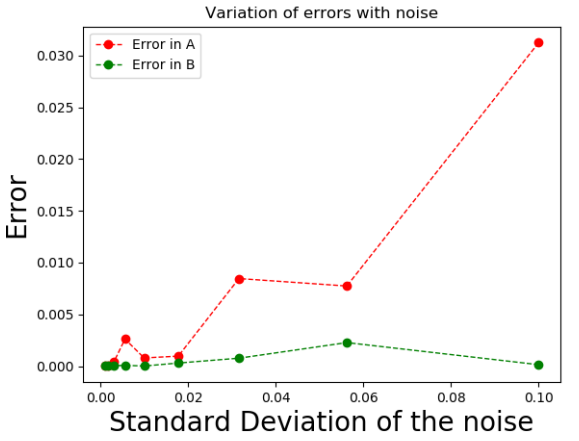

Figure 4: The plot shows how the error in A and B vary with σ

We replot Figure 4 using `loglog` in Figure 5. Evidently, this plot is not linear. It was also observed that for different dataset with the same noise σ, the error in the least squares estimate need not be the same. Hence the error plot not being linear is expected.

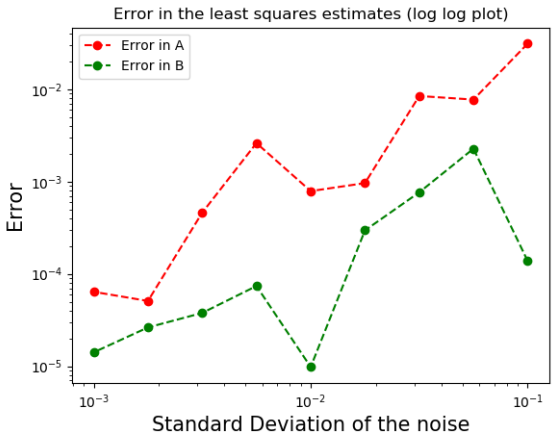

Figure 5: This is Figure 4 plotted in log log scale