Herramientas Computacionales para Ciencias
Homework 10

Mauricio Sevilla*

22/04/2019

Rules

Note Read carefully the complete homework before starting, so you will know what is the results you have to get!.

This week we are going to concentrate on regression of *experimental* data.

On this assignment you will have to construct some functions and use the `matplotlib` + `NumPy` +`SciPy` structure to plot the results. There is an additional point that must be developed on class before sending the assignment. This part must be saved on a jupyter Notebook named as your UniAndes username.

Introduction

Science is constructed from experimental data, where once having a model it is tested with them. So nature is who actually tells us if our models are right!!.

The regression problem helps us to verify how good our model *fits* to the experimental data. These concepts are also used with computer generated (Simulations) data, and it can quantify how good our model is (at least for the specific set of data) so that different models can be compared clearly.

Transformation of data! [2.0/5.0]

Sometimes data sets look difficult to model so they have to be transformed to new variables so that it gets easier, we are going to explore an example of that.

- Using `NumPy`, import the data from,
  `https://raw.githubusercontent.com/jmsevillam/Herramientas-Computacionales-UniAndes/master/Data/Regression/spiral.dat`
  The file has 4 columns, x, y, r and θ, you can get and unpack them as

```
x,y,r,theta=np.genfromtxt(url).T
```

  Note: it is important for you to notice that, you have to define the variable `url` before if you want to use that line.

- plot x vs y, you will see a noisy spiral.[1]

  Note: Remember to use labels and titles. With this plot, it is not clear how can we fit this data.

- plot r vs θ. It looks like a straight line!!, so use the parametric equation for a spiral

$$r(\theta) = a \cdot \theta + b \tag{1}$$

  Is the line equation.

  Define a function with your model (Exactly the same case we have on the slides where we wanted to do the fit to a straight line.) and use the `curve_fit` method to find the best pair a and b as we did in class.

```
popt,pcov=curve_fit(model,r,theta)
```

- To see our results, define a new parameter (just to make the plot smooth) I will name r_1, such that

```
r=np.linspace(0,10,1000)
x1,y1=r*np.cos(f(r,*popt)),r*np.sin(f(r,*popt))
```

  Plot x vs y and x_1 vs y_1.

- Comment your results

*email=j.sevillam@uniandes.edu.co

[1] The values of x and y were obtained from the relationship $x = r \cdot \cos(\theta)$ and $y = r \cdot \sin(\theta)$

Overfitting! [2.0/5.0]

- Get the data from

  https://raw.githubusercontent.com/jmsevillam/Herramientas-Computacionales-UniAndes/master/Data/Regression/data.dat

- There are two columns unpack them on the variables x and y.

- Define two more variables x_1 and x_2 such that the last number of x and y are missing, thus

```
x1=x[:-1]
y1=y[:-1]
```

  and we are going to test two different models, a straight line (1st order polynomial) and a 9th order polynomial. define the two models, for the case of the 9th order polynomial, you can use

```
def poly9(x,a0,a1,a2,a3,a4,a5,a6,a7,a8,a9):
        return np.poly1d([a0,a1,a2,a3,a4,a5,a6,a7,a8,a9])(x)
```

- Do the fit of the data x_1 and y_1 with both models and plot them

  Hint: Use different names for the optimal parameters of each model.

```
popt1,pcov1=curve_fit(line,x1,y1)
popt2,pcov2=curve_fit(poly9,x1,y1)
```

- To see the results, we would like to plot the results, but to make smooth the plot of the models define a new variable $x_{\text{plot}}1$ such that

```
xplot1=np.linspace(0,9,1000)
plt.plot(xplot,line(xplot,*popt1))
plt.plot(xplot,poly9(xplot,*popt2))
plt.plot(x1,y1)
```

  comment which model do you think works better.

- Now define a new variable $x_{\text{plot}}2$ such that goes from 0 to 10, and repeat the previous plot but using $x_{\text{plot}}2$ and x, y instead of x_1, y_1. Do you think both models describe the last point?

- Use the two models you just used (The straight line and the 9th order polynomial) to fit x and y (Previously you did it for x_1 and y_1). Use different names for the optimal parameters

```
popt2,pcov2=curve_fit(line,x,y)
popt3,pcov3=curve_fit(poly9,x,y)
```

  Compare you results on both cases, optimal parameters of the pair (x, y) and (x_1, y_1). On the examples I wrote on this document, it should be a comparison between `popt1` and `popt3`, and another comparison between `popt2` and `popt4`. What can you tell from that?[2]

- A better comparison can be done by evaluating the relative difference, it is calculated by taking the difference and dividing by one of them, for example

```
print((popt1-popt3)/popt1)
```

  And like that, you get percentages of changes, repeat the comparison done above and comment.

[2] The comparison can be done just by plotting and then visually see how much the values change. *i.e.* `plot popt1,popt3`