Chapter 4.3 - Data Fitting with Python

12 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=_U9C3PBJ6BG
https://www.youtube.com/watch?v=_u9C3PBJ6bg

SUMMARY

This lecture focuses on using Python for data fitting, specifically through polynomial fits, least squares regressions, and interpolation techniques. The instructor demonstrates how to implement these methods using Python's `polyfit` and `polyval` functions, as well as other interpolation methods, to fit data and analyze the quality of the fits.

- *Introduction to polynomial fitting and least squares regression in Python.*
- *Use of `polyfit` to obtain polynomial coefficients for line and parabolic fits.*
- *Explanation of `polyval` to evaluate polynomial functions at specified points.*
- *Demonstration of calculating the least squares error to assess fit quality.*
- *Discussion on the drawbacks of high-degree polynomial fits, such as overfitting and oscillation at data edges.*
- *Introduction to interpolation methods, including linear and cubic splines, to connect data points without fitting a polynomial.*
- *Overview of nonlinear fitting using optimization techniques, specifically fitting Gaussian curves to data.*
- *Emphasis on the simplicity and effectiveness of Python libraries for executing various data fitting techniques.*

we're going to now talk about using python for data fitting we've talked a little bit about polinomial fits to the data or Le Square regressions to the data so line fits parabolic fits splines so what I want to do now is walk you through some of the coding that would happen that in Python that you would need to do all of this so a lot of this will be examples that we walk through remember all the code can be found here

on this GitHub link you can clone this repo everything I demonstrate here right now is directly from that repo so you can execute it all for yourself very simply so let's start off with what is available for python so obviously we're going to try to do some Le Square fits line fits parabolic fits I'm going to just show you how we do this in in Python itself so it's going to go right to code and so here's the code that we

would be using it's the polyfit command and there it is here so the polyfit command you give me data X and Y and then you specify the degree of the polinomial you want to fit this is a first-degree polinomial or a line so it's ax plus b and we already worked through what the equations are so this is all embedded in here so it just takes the data computes the A and B and it gives you back P CF these are the

polinomial coefficients so it doesn't give you back a line it gives you back A and B and so what you can use is this A and B then to do your line fit and so here's how you would do this so I can use I can say I would like to now show the data on the interval from 0 to 7 in steps of 0.1 so this is this new X variable that I Define and I want to do

poly Val so what poly Val does it takes my coefficients in this case it's a and b and it now makes a line that is defined on all these points between 0 to 7 and steps of 0.1 so that's y of P so how that's how the command structure works you first do poly fit to get the coefficients the polinomial fit and then when you want to actually produce the curve polyval is the command structure for you to do this so here's what this

looks like here the blue is the data is an example data set that's again in the notebooks that you have access to and the idea was to say what is the best line fit so this is directly from this code here and that magenta line is the best Le Square line fit available to you when you're using this polyfit command it pulls out the coefficients A and B of this of of this fit you can do more

complex fitting not even that much more complex this is just a parabola so right all I did is increase the degree of the fit to two so now it's ax^2 plus BX plus C same thing poly fit this you have the X Y values I already defined the XP previously so all I have to do is poly valal with these new coefficients to a par parabolic fit and now here's the fit that you would get so it's really

easy to do polinomial fitting it's trivial you just give it the data tell it degree of polinomial do poly fit and then you want to actually plot it poly valve and so here's what it would look like here and again you can start to see the quality of your fit that you would get here by picking out the polinomial one in the next lecture I'm going to show you well what if I don't know what I should pick I have a I have

a method I can show you that will essentially select out the best modeled fit that you would have it might fit for instance the best Parabola or something like this but for now here's what you got all right I could also since this has end points I could say I'm going to do poly Val here oh by the way first before we do that let's just compute The Roots mean square error this is what we're doing

here so I can actually compute yp3 this is using polyal to compute this parabolic fit at the points X where I actually have data so in other words I have a model fit this Parabola and what I want to know is how does the parabola differ from that magenta line at the data points itself at each value of x and so this poly Val is pulling out this line fit at The X

values where I have actual data points and so what I can do here now is walk through and compute the difference between my fit and the data squared sum it all up square root and that's my least square fit error so it's a very simple calculation to get Le Square F error so right because what happens is you know your polyfit command you're typically just getting the A and B out you can actually extract out the fit as

well but here is also a way to compute it and postprocessing all right so now let's go and do a polinomial fit so the polinomial fit now is not going to be a polinomial fit like a line fit or quadratic it's going to be a polinomial that goes through all the data points so this is the second lecture we had in this series on curve fitting so n is the length L of the vector X other was

number of data minus one so this gives us remember we start counting at zero and python so I'm going to go ahead and take the length of this data I'm going to go ahead and do a polyfit to n so xyn so this is going to give me essentially a polom that goes through all the data so if I have end points it's right that I would have or I would have n an N degree polinomial to go through

those points so I do a poly valon on it I just put in the points I want to compute out again put in the coefficients and then I fit this line here and here's what it looks like so you can see now the error is zero the least square error in other words if I look at my prediction of where what the data says versus the actual data I'm actually sitting on I'm going through every single data point so

technically speaking my error is zero but you can see what ends up happening if you do a poly fit you get this polinomial wiggle at the edges and so you could always ask the question how would you compare this curve fit to this Parabola or to this line so the line you know it's not a bad fit but I would say the parabola is better and this of course looks fine in the middle but at the edges it's

actually a very bad fit so these are the kind of things you should be aware of when you're using a polyfit if you go up in degrees of Pol you can get things like this happening that may be not something that you desire and actually in showing we can also do standard interpolations in other words we take the data and we simply do interpolate between the data points so we're not doing a curve fit or a polinomial fit

we're just doing interpolate we're doing an interpolation between the data points so here's how you do this in turp 1D you give me the XY data and it produces interpolation now normally when you do this what what it just simply does is exactly you did when you're kid it's connect the dot so between any two points it just draws a line you can specify this explicitly here in this command so interp one you give me the

data kind linear so what it does is just fit you know it just connects the dots so two points draws a line next two points draws a line so in between each data point is a a new line that is interpolated line you could also instead do nearest so it's going to take on the value of the nearest data point or a cubic which is now going to basically create a spline so this cubic inter Turan D is equivalent to a

spline so here's what these look like here so you're looking at is the data and the points against the par parabolic fit and we're looking at nearest neighbor which is this you can see the nearest neighbor is takes this stair step type structure because it's taking on the value of the nearest data point it has versus a linear interpolation which is just a line fit you can see it here connect the dots you can also put a spline in and

here's the spline command you bring in a spline you tell the data points you want it and it basically creates a spline using your XY data remember creates a cubic the cubic has first and second deriva that are smooth and so this is a spline which is equivalent to a TP 1D with the cubic command in it okay but you can also call splines directly and again this is what this would look like the blue are the data that black line is

your spline fit and you can see the spline is in some sense a very nice fit to the data it's very smooth and it's much better than a polinomial fit going through the data because it doesn't wiggle it doesn't have the polinomial wiggle issue what about more sophisticated fitting well we can do something like like fitting something like this here which is a gaussian to data and so now this is going to be a nonlinear system

of equations we're going to fit it to where I have to determine A and B so let's talk about how we might do this so we're going to use some of the optimization routines in Python and what we're going to do is minimize the least squit error okay and so putting this in here this would generically give you a nonlinear system of equations that you'd have to solve to get a and b and this is exact exactly what this optimization

algorithm is going to do we're going to guess what A and B is and then it's going to do something like gradient descent actually something a little more sophisticated probably a ner me algorithm that's going to walk downhill and get us a and b so here's how You' first do it first I Define this gaussian I come in here gafit I give it a guess so in all these algorithms it's going to be based upon an iteration if you don't

give it a guess it will typically get S 0 0 you give it a guess we're going to pick one in a minute and then what I'm going to fit is some data so I actually loaded some data the XY values I really should do this outside the loop and basically send it in but here it's fine it's inside of this right hand side and here is the error this is the least square root me square error right here

and so the idea here is whatever I put here this is what it's going to minimize and this is what I'm return is the minimization of this error giv the guesses which is two components of X notot which is the first component which is a or sorry the amplitude right a and the second component which is B okay so that's it so you send in initial values of A and B it's going to iterate On A

and B to minimize this error how do I do it with this fmin command the fmin command calls the function that you're going to fit which is right here with a guess one one and what's going to come out you guess you send in guesses of one and one for the A and the B and it's going to come out with the actual values A and B that minimize the error for your guess remember this is a nonlinear system of

equations so there's no guarantee that you have don't have multiple Minima in this or some local Minima but it's based largely upon what you guess so the idea is to be a reasonable guesser you can always plot the data and always plot your guess and then let this thing work away to give you back the coefficients you want so here's what it might look like so the blue is the data the magenta is the least square fit galum so I'm now

doing nonlinear curve fitting with this using the fmins command which has an iterative structure inside of it to minimize the error so these are the kind of options you have available to you for curve fitting right all in Python very simple to execute very simple to use you know know you should use them whenever you can because it's all built in there so just just plot your data do a curve fit then this gives you the

first go at having a model that can extrapolate or interpolate on your data that's interpretable so that's what we want to do and this is all the some of the command structures in Python to do it