Subject: Re: poly_fit for less number of points
Posted by Heinz Stege on Sat, 04 May 2013 12:00:09 GMT
View Forum Message <> Reply to Message

On Sat, 4 May 2013 03:44:28 -0700 (PDT), sid wrote:

> Hello everyone,
> I have only 6 x values and y values, for example
> my x = [3932.9321,3933.0452,3933.1162,3933.2514,3933.3517,3933.4271]
> y = [0.090313701,0.084354165,0.080794218,0.075102273,0.074025219 ,0.082037902]
> I have to fit this with polynomial, I tried svdfit and poly_fit. But are not giving good fit. Please suggest me a way to fit these data points.
>
> thanking you in advance
> sid

You could try the following: Calculate the average xc of the x values.
Then transform the x values to xt=x-xc and do the fitting for the
tranformed values.

Heinz

Subject: Re: poly_fit for less number of points
Posted by sid on Sat, 04 May 2013 14:02:57 GMT
View Forum Message <> Reply to Message

On Saturday, May 4, 2013 4:14:28 PM UTC+5:30, sid wrote:
> Hello everyone,
>
> I have only 6 x values and y values, for example
>
> my x = [3932.9321,3933.0452,3933.1162,3933.2514,3933.3517,3933.4271]
>
> y = [0.090313701,0.084354165,0.080794218,0.075102273,0.074025219 ,0.082037902]
>
> I have to fit this with polynomial, I tried svdfit and poly_fit. But are not giving good fit. Please suggest me a way to fit these data points.
>
>
>
> thanking you in advance
>
> sid

Thank you very much sir it works. But can you please suggest me is there any other fit available to fit this kind of dataset. Because I want to get a smoothed fit.
thanking you in advance

sid

Subject: Re: poly_fit for less number of points
Posted by Craig Markwardt on Sun, 05 May 2013 19:55:01 GMT
View Forum Message <> Reply to Message

On Saturday, May 4, 2013 10:02:57 AM UTC-4, sid wrote:
> On Saturday, May 4, 2013 4:14:28 PM UTC+5:30, sid wrote:
>
>> Hello everyone,
>
>>
>
>> I have only 6 x values and y values, for example
>
>>
>
>> my x = [3932.9321,3933.0452,3933.1162,3933.2514,3933.3517,3933.4271]
>
>>
>
>> y = [0.090313701,0.084354165,0.080794218,0.075102273,0.074025219 ,0.082037902]
>
>>
>
>> I have to fit this with polynomial, I tried svdfit and poly_fit. But are not giving good fit. Please suggest me a way to fit these data points.
>
>>
>
>>
>
>>
>
>> thanking you in advance
>
>>
>
>> sid
>
>
>
> Thank you very much sir it works. But can you please suggest me is there any other fit available to fit this kind of dataset. Because I want to get a smoothed fit.
>

This fit,

res = poly_fit(x, y, 2)
gives a smooth fit.

Craig

Subject: Re: poly_fit for less number of points
Posted by Craig Markwardt on Sun, 05 May 2013 19:59:13 GMT
View Forum Message <> Reply to Message

On Saturday, May 4, 2013 6:44:28 AM UTC-4, sid wrote:
> Hello everyone,
>
> I have only 6 x values and y values, for example
>
> my x = [3932.9321,3933.0452,3933.1162,3933.2514,3933.3517,3933.4271]
>
> y = [0.090313701,0.084354165,0.080794218,0.075102273,0.074025219 ,0.082037902]
>
> I have to fit this with polynomial, I tried svdfit and poly_fit. But are not giving good fit. Please suggest me a way to fit these data points.

This fit,
res = poly_fit(x-mean(x), y, 2)
gives a smooth fit.

Craig

Subject: Re: poly_fit for less number of points
Posted by David Fanning on Sun, 05 May 2013 22:03:25 GMT
View Forum Message <> Reply to Message

Craig Markwardt writes:

> This fit,
> res = poly_fit(x-mean(x), y, 2)
> gives a smooth fit.

What is the principle here that made you think of this solution and
caused it to work?

Cheers,

David
--
David Fanning, Ph.D.

Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: poly_fit for less number of points
Posted by Jeremy Bailin on Mon, 06 May 2013 13:48:57 GMT
View Forum Message <> Reply to Message

On 5/5/13 6:03 PM, David Fanning wrote:
> Craig Markwardt writes:
>
>> This fit,
>> res = poly_fit(x-mean(x), y, 2)
>> gives a smooth fit.
>
> What is the principle here that made you think of this solution and
> caused it to work?
>
> Cheers,
>
> David
>

Roundoff error in the typical fitting routines. Look at the OP's
original x values:

x = [3932.9321,3933.0452,3933.1162,3933.2514,3933.3517,3933.4271]

By switching to x-mean(x):

IDL> print, x-Mean(x)
  -0.255371 -0.142334 -0.0712891 0.0639648 0.164307
0.239502

The effective precision of the values is 4 extra digits. Within the
polynomial fitting code, they're going to be squared, so roundoff error
in the original data is going to get bad, but the extra 4 digits keep it
under control.

-Jeremy.

Subject: Re: poly_fit for less number of points
Posted by Heinz Stege on Mon, 06 May 2013 18:02:07 GMT
View Forum Message <> Reply to Message

On Sat, 4 May 2013 07:02:57 -0700 (PDT), sid wrote:

> ... But can you please suggest me is there any other fit available to fit this kind of dataset. Because I want to get a smoothed fit.

I'm not sure, what you mean with "smoothed fit". I fully agree with Craig, that a polynomial fit of degree 2 is very smooth.

Putting a more detailed look onto your data, I see that the function has to be vary asymmetric to match the data. I guess you tried degree 4 or higher and don't like that wiggles, which are typical for polynomal fits of higher degree.

It would be very helpful to know, if you are looking for a least square fit (which compensates uncertainties within the y values) or a function which interpolates the given points (and goes exactly through the given points).

In any case I think, that it is very risky to draw an arbitrary curve through the points on the right side of the diagram. It would be very helpful, to have a model explaining y vs. x. Can you tell something about the origin of the data?

A "quick and dirty" solution may be a hyperbolic function like
  y(t) = (p0 + p1*t + p2*t^2) / (t - q0)
where p0, p1, p2 and q0 are the (fit-)constants and t=x-mean(x).

Such a non-linear function can be fitted by IDL's curvefit() or Craig Markwardt's mpfit(). If you want to see the result for your example before doing the fitting stuff, here are my results:
  q0=0.311793, p0=-0.0241749, p1=0.0906013, p2=-0.0602339

But again, this is very speculative. This is one function, that fits your data. And there may be other functions with other results! Furthermore I would expect, that you have other data examples, where the function above does not work!

Heinz

Subject: Re: poly_fit for less number of points
Posted by Craig Markwardt on Mon, 06 May 2013 20:26:51 GMT
View Forum Message <> Reply to Message

On Sunday, May 5, 2013 6:03:25 PM UTC-4, David Fanning wrote:
> Craig Markwardt writes:
>
>> This fit,

>> res = poly_fit(x-mean(x), y, 2)
>> gives a smooth fit.
>
>
>
> What is the principle here that made you think of this solution and
> caused it to work?

Well, not much. He asked for a smooth function and I gave it to him. Jeremy is right, there is a round-off problem that makes the original POLY_FIT(X,Y,2) not work. Subtracting the mean of X (and/or Y) is a classic solution to the problem of round-off.

As HeinzS intimates, there's really a lot more to the question, like, how smooth should the function be? what degree of polynomial is permissible? how close of an approximation should the function be? what is the tolerance for outliers? why are there no error bars on the data? what is the definition of a "good fit?"

Without any of that information, I felt it worth to just provide *an* answer, even if it wasn't *the* answer to the original implicit question.

Craig

Subject: Re: poly_fit for less number of points
Posted by Yngvar Larsen on Tue, 07 May 2013 08:20:56 GMT
View Forum Message <> Reply to Message

On Monday, 6 May 2013 22:26:51 UTC+2, Craig Markwardt wrote:
> On Sunday, May 5, 2013 6:03:25 PM UTC-4, David Fanning wrote:
>
>> Craig Markwardt writes:
>
>>> This fit,
>>> res = poly_fit(x-mean(x), y, 2)
>>> gives a smooth fit.
>
>> What is the principle here that made you think of this solution and
>> caused it to work?
>
> Well, not much. He asked for a smooth function and I gave it to him. Jeremy is right, there is a round-off problem that makes the original POLY_FIT(X,Y,2) not work. Subtracting the mean of X (and/or Y) is a classic solution to the problem of round-off.

Often you also want to _normalize_ the variation of X (and possibly also Y), such that X is bounded within [-1,1], e.g.

Xmid = 0.5*(min(X)+max(X))
Xinterval = max(X)-min(X)

Xnorm = 2*(X-Xmid)/Xinterval

This makes all powers of Xnorm to also be bounded within [-1,1], which is good for preservation of precision. In the particular example given by OP, all X values are bounded in an interval with a width of around 0.5, so normalization is not really important, just centering.

[Another option is the statistical normalization: Xnorm = (X-mean(X))/stddev(X)]

> As HeinzS intimates, there's really a lot more to the question, like, how smooth should the function be? what degree of polynomial is permissible? how close of an approximation should the function be? what is the tolerance for outliers? why are there no error bars on the data? what is the definition of a "good fit?"
>
> Without any of that information, I felt it worth to just provide *an* answer, even if it wasn't *the* answer to the original implicit question.

... but of course, what Heinz&Craig write here is the real issue.

--
Yngvar
