Subject: Re: Polynomial Fitting question
Posted by wita on Fri, 02 Oct 2009 11:47:15 GMT
View Forum Message <> Reply to Message

Dear Rodger,

Are you sure that it is really the double FOR loop that is slowing you down?

I am building a framework for processing satellite time-series data, which is similar to what you are doing here: looping over the rows and columns and fitting a model to the data. I use Craig's MPFitFun to fit a model (usually more complex then a polynomial), but I find that typically 95% of the processing time is spent within the MpFit routines. So trying to optimize on the double FOR loop will not gain anything.

I suggest you run the IDL profiler on your code (maybe you did, but it is not mentioned in your post) to see what is causing the delay.

Allard

Subject: Re: Polynomial Fitting question
Posted by Andrew Rodger on Fri, 02 Oct 2009 13:24:17 GMT
View Forum Message <> Reply to Message

> I suggest you run the IDL profiler on your code (maybe you did, but it
> is not mentioned in your post) to see what is causing the delay.
> Allard

Hi Allard,

shamefully I have not run the profiler yet :(
I will do that first thing monday. I am in fact working with airborne hyperspectral datasets (HyMap) and am fitting polynomials to various surface absorption features (Fe, leaf water, cellulose etc). The polynomial fit is not the ideal and is only the first cut (although it does work well with the broader features) and I will move to gaussians at a later date.

This part of the processing is after I have performed the atmospheric compensation (radiance to reflectance) and a spectral recalibration of the band centres. I have this whole process running from go to whoa in 3-5 minutes. This latter aspect (the fitting of the surface absorptions) which allows me to determine content and composition data

based on the surface reflectance is painfully slow compared to the initial processing and is driving me crazy.

It just seems that there must be others who want/need to fit equations on a per-pixel basis (yourself included by the sound of it) but do not want to have to run through the looping process (seems decidedly like an unIDL way). I am not sure how large your datasets are but mine are 512 pixels by anywhere from 3000-11000 lines. This is probably small to some folks.

I will give the profiler a go and see if I can chase this down a bit better.

Cheers
Andrew

Subject: Re: Polynomial Fitting question
Posted by Andrew Rodger on Sat, 03 Oct 2009 16:15:11 GMT
View Forum Message <> Reply to Message

OK so I am a nerd and worked on this at home, over the weekend :(

But I have my solution now and it does not require an inner FOR loop and could be extended to encompass Gaussian curves as well. I went back to basics and used the tried and true least squares fitting of the following form

coefficients=INVERSE(TRANSPOSE(X)X)TRANSPOSE(X)Y I didn't really know how to write this without using some pseudo syntax.

Anyhoo, the little code snippet below is what I used to test it. I am sure that I have used the # and ## incorrectly and I have also discovered that I could have used matrix_multiply to do the same thing in a lot of cases (I may still do that). The thing I like about this is that the values BIG_X and TRANSPOSE(x) can be calculated outside of any loop and I only need to calculate XtY within the original outer loop (BIG_Y is simply a one liner that is supplied via an ASSOC variable). So I should only need two lines (the last two) within the outer loop to get my coefficients on a line by line and pixel by pixel basis.

```
PRO test,coefficients

;Some X array: These represent my wavelengths which will never change
in a given image
;(12,1) array
 x_sing=[890,900,910,920,930,940,950,960,970,980,990,1000]/10 00.
```

```
;(3,12) array
x=double(transpose([[fltarr(12)+1],[x_sing],[x_sing^2]]))
;(3,3) array
temp_x=x#transpose(x)
;(3,3) array
BIG_X=REAL_PART(lu_complex(temp_x,/inverse))

;make some Y arrays. These represent the surface reflectance for a
given pixel
;in this case 5 pixels. In actuality they will be read in 512 pixels
at a time from my image data
;(12,1) array
y1=0.5+0.3*x_sing+0.77*x_sing^2
;(12,1) array
y2=2.0+0.5*x_sing+1.2*x_sing^2
;(12,1) array
y3=4.0+1.5*x_sing+0.2*x_sing^2
;(12,1) array
y4=10.0+0.6*x_sing+3.0*x_sing^2
;(12,1) array
y5=1.0+2.0*x_sing+3.0*x_sing^2
;(5,12) array
BIG_Y=TRANSPOSE([[y1],[y2],[y3],[y4],[y5]])

;(5,3) array
XtY=TRANSPOSE(x)##BIG_Y
;These are the coefficients that I seek
coefficients=BIG_X##xty
END
```

Cheers
Andrew
