Subject: Re: curve fitting: works badly?
Posted by rivers on Tue, 07 Mar 1995 03:52:11 GMT
View Forum Message <> Reply to Message

>> PDER(*,1) = (x-a(2))^2
>> PDER(*,2) = 2*a(1)*(x-a(2))
> ^^^^^^^^^^^^^^^^
>> end
>
>> RETURN
>
>> end
>
> I think that your problem is here. I believe that the sign of this partial
> derivative should be negative.
>
> I've often had this problem with this technique. If the partial derivatives
> are calculated wrong, you can compute till doomsday and it'll never converge.
> I often just don't bother to try to figure it out anymore, and just do the
> derivatives numerically.
>

I have written a new version of CURVEFIT which will take care of doing the derivatives for you numerically. It fixes a number of other problems, but remains backward compatible with the old version. I sent it to RSI and David Stern tells me it will be in the next release.

Here is the letter I sent them describing the new version:

****

Folks,

I am appending an improved version of CURVEFIT, the non-linear least-squares routine from the MATH library. The improvements I have made include the following:

- Fixed a serious bug. The existing version passes PDER to the user's procedure on the first call, but does not define PDER first. Thus, if the user's routine tests whether PDER is defined (n_elements(PDER ne 0)), it is not defined, and the user's routine is thus not supposed to calculate the partial derivatives. However, CURVEFIT attempts to use PDER on return, and generates an error.

- Added the NODERIVATIVE keyword. This allows the user to tell CURVEFIT that the user's procedure will NOT calculate the partial derivatives. If this keyword is set, then CURVEFIT will estimate the partial derivative array using a forward-difference approximation. This saves users the constant hassle of having to do forward-difference approximation of the derivatives

in their own procedure. The user is still free to compute derivatives, either from analytical equations or finite-differences, if desired.

- Added the ITMAX keyword, to allow the user to specify the maximum number of iterations. This was previously hardcoded at 20.

- Added the TOL keyword, to allow the user to specify the relative improvement in chi-squared for convergence. This was previously hardcoded at .001.

- Added the CHI2 keyword to allow the user to retrieve the final value of chi-squared.

- I removed the test for convergence prior to the first iteration, since this used a different test, and prevented the user from getting back error estimates.

My changes should all be upward compatible with the existing version of CURVEFIT, so existing code should not break.

Please feel free to use this in your next distribution if you would like.

Mark Rivers

****

Here is the new version of CRUVEFIT. This version is slightly modified from the one I sent RSI - it now correctly handles double precision.

********** **********

```
; $Id: curvefit.pro,v 1.2 1993/10/26 23:44:03 doug Exp $

function curvefit, x, y, w, a, sigmaa, Function_Name = Function_Name, $
                   itmax=itmax, iter=iter, tol=tol, chi2=chi2, $
                   noderivative=noderivative
;+
; NAME:
;       CURVEFIT
;
; PURPOSE:
;       Non-linear least squares fit to a function of an arbitrary
;       number of parameters.  The function may be any non-linear
;       function.  If available, partial derivatives can be calculated by
;       the user function, else this routine will estimate partial derivatives
;       with a forward difference approximation.
;
; CATEGORY:
;       E2 - Curve and Surface Fitting.
;
```

```
; CALLING SEQUENCE:
;       Result = CURVEFIT(X, Y, W, A, SIGMAA, FUNCTION_NAME = name, $
;                         ITMAX=ITMAX, ITER=ITER, TOL=TOL, /NODERIVATIVE)
;
; INPUTS:
;       X:  A row vector of independent variables.
;
;       Y:  A row vector of dependent variable, the same length as x.
;
;       W:  A row vector of weights, the same length as x and y.
;             For no weighting,
;             w(i) = 1.0.
;             For instrumental weighting,
;             w(i) = 1.0/y(i), etc.
;
;       A:  A vector, with as many elements as the number of terms, that
;           contains the initial estimate for each parameter.  If A is double-
;           precision, calculations are performed in double precision,
;           otherwise they are performed in single precision.
;
; KEYWORDS:
;       FUNCTION_NAME:  The name of the function (actually, a procedure) to
;       fit.  If omitted, "FUNCT" is used. The procedure must be written as
;       described under RESTRICTIONS, below.
;
;       ITMAX:  Maximum number of iterations. Default = 20.
;       ITER:   The actual number of iterations which were performed
;       TOL:    The convergence tolerance. The routine returns when the
;               relative decrease in chi-squared is less than TOL in an
;               interation. Default = 1.e-3.
;       CHI2:   The value of chi-squared on exit
;       NODERIVATIVE:   If this keyword is set then the user procedure will not
;               be requested to provide partial derivatives. The partial
;               derivatives will be estimated in CURVEFIT using forward
;               differences. If analytical derivatives are available they
;               should always be used.
;
; OUTPUTS:
;       Returns a vector of calculated values.
;       A:  A vector of parameters containing fit.
;
; OPTIONAL OUTPUT PARAMETERS:
;       Sigmaa:  A vector of standard deviations for the parameters in A.
;
; COMMON BLOCKS:
;       NONE.
;
; SIDE EFFECTS:
```

```
;       None.
;
; RESTRICTIONS:
;       The function to be fit must be defined and called FUNCT,
;       unless the FUNCTION_NAME keyword is supplied.  This function,
;       (actually written as a procedure) must accept values of
;       X (the independent variable), and A (the fitted function's
;       parameter values), and return F (the function's value at
;       X), and PDER (a 2D array of partial derivatives).
;       For an example, see FUNCT in the IDL User's Libaray.
;       A call to FUNCT is entered as:
;       FUNCT, X, A, F, PDER
; where:
;       X = Vector of NPOINT independent variables, input.
;       A = Vector of NTERMS function parameters, input.
;       F = Vector of NPOINT values of function, y(i) = funct(x(i)), output.
;       PDER = Array, (NPOINT, NTERMS), of partial derivatives of funct.
;               PDER(I,J) = DErivative of function at ith point with
;               respect to jth parameter.  Optional output parameter.
;               PDER should not be calculated if the parameter is not
;               supplied in call. If the /NODERIVATIVE keyword is set in the
;               call to CURVEFIT then the user routine will never need to
;               calculate PDER.
;
; PROCEDURE:
;       Copied from "CURFIT", least squares fit to a non-linear
;       function, pages 237-239, Bevington, Data Reduction and Error
;       Analysis for the Physical Sciences.
;
;       "This method is the Gradient-expansion algorithm which
;       combines the best features of the gradient search with
;       the method of linearizing the fitting function."
;
;       Iterations are performed until the chi square changes by
;       only TOL or until ITMAX iterations have been performed.
;
;       The initial guess of the parameter values should be
;       as close to the actual values as possible or the solution
;       may not converge.
;
; EXAMPLE:
;       pro gfunct, x, a, f, pder
;         f=a(0)*exp(a(1)*x)+a(2)
;         pder=findgen(10, 3)
;       end
;
;       pro fit_curve
;         x=float(indgen(10))
```

```
;         y=[12.0, 11.0,10.2,9.4,8.7,8.1,7.5,6.9,6.5,6.1]
;         w=1.0/y
;         a=[10.0,-0.1,2.0]
;         yfit=curvefit(x,y,w,a,sigmaa,function_name='gfunct')
;         print, yfit
;       end
;
; MODIFICATION HISTORY:
;       Written, DMS, RSI, September, 1982.
;       Does not iterate if the first guess is good.  DMS, Oct, 1990.
;       Added CALL_PROCEDURE to make the function's name a parameter.
;             (Nov 1990)
;       12/14/92 - modified to reflect the changes in the 1991
;           edition of Bevington (eq. II-27) (jiy-suggested by CreaSo)
;       Mark Rivers, Feb. 12, 1995
;          - Added following keywords: ITMAX, ITER, TOL, CHI2, NODERIVATIVE
;            These make the routine much more generally useful.
;          - Removed Oct. 1990 modification so the routine does one iteration
;            even if first guess is good. Required to get meaningful output
;            for errors.
;          - Added forward difference derivative calculations required for
;            NODERIVATIVE keyword.
;          - Fixed a bug: PDER was passed to user's procedure on first call,
;            but was not defined. Thus, user's procedure might not calculate
;            it, but the result was then used.
;
;-
      on_error,2              ;Return to caller if error

      ;Name of function to fit
      if n_elements(function_name) le 0 then function_name = "FUNCT"
      ;Convergence tolerance
      if n_elements(tol) eq 0 then tol = 1.e-3
      ;Maximum number of iterations
      if n_elements(itmax) eq 0 then itmax = 20
      a = 1.*a                ; Make params floating or double
      ; Set flag if calculations are to be done in double precision
      t = size(a)
      if (t(n_elements(t)-2) eq 5) then double=1 else double=0
      ; If we will be estimating partial derivatives then compute machine
      ; precision
      if keyword_set(NODERIVATIVE) then begin
        if (double) then res = nr_machar(/DOUBLE) else res=nr_machar()
        eps = sqrt(res.eps)
      endif

      nterms = n_elements(a)   ; # of parameters
      nfree = (n_elements(y)<n_elements(x))-nterms ; Degrees of freedom
```

```
      if nfree le 0 then message, 'Curvefit - not enough data points.'
      flambda = 0.001         ;Initial lambda
      diag = indgen(nterms)*(nterms+1) ; Subscripts of diagonal elements

;     Define the partial derivative array
      pder = fltarr(n_elements(x), nterms)
      if (double) then pder=double(pder)

      for iter = 1, itmax do begin   ; Iteration loop

;       Evaluate alpha and beta matricies.
        if keyword_set(NODERIVATIVE) then begin
;         Evaluate function and estimate partial derivatives
          call_procedure, Function_name, x, a, yfit
          for term=0, nterms-1 do begin
            p = a       ; Copy current parameters
            ; Increment size for forward difference derivative
            inc = eps * abs(p(term))
            if (inc eq 0.) then inc = eps
            p(term) = p(term) + inc
            call_procedure, function_name, x, p, yfit1
            pder(0,term) = (yfit1-yfit)/inc
          endfor
        endif else begin
          ; The user's procedure will return partial derivatives
          call_procedure, function_name, x, a, yfit, pder
        endelse

        beta = (y-yfit)*w # pder
        alpha = transpose(pder) # (w # (fltarr(nterms)+1)*pder)
        chisq1 = total(w*(y-yfit)^2)/nfree ; Present chi squared.

;       Invert modified curvature matrix to find new parameters.
        repeat begin
          c = sqrt(alpha(diag) # alpha(diag))
          array = alpha/c
          array(diag) = array(diag)*(1.+flambda)
          array = invert(array)
          b = a+ array/c # transpose(beta) ; New params
          call_procedure, function_name, x, b, yfit  ; Evaluate function
          chisqr = total(w*(y-yfit)^2)/nfree        ; New chisqr
          flambda = flambda*10.                     ; Assume fit got worse
        endrep until chisqr le chisq1
;
        flambda = flambda/100.  ; Decrease flambda by factor of 10
        a=b                     ; Save new parameter estimate.
        if ((chisq1-chisqr)/chisq1) le tol then goto,done  ; Finished?
      endfor                    ;iteration loop
```

```
;
      message, 'Failed to converge', /INFORMATIONAL
;
done:  sigmaa = sqrt(array(diag)/alpha(diag)) ; Return sigma's
      chi2 = chisqr                    ; Return chi-squared
      return,yfit            ;return result
END
```


Mark Rivers (312) 702-2279 (office)
CARS (312) 702-9951 (secretary)
Univ. of Chicago (312) 702-5454 (FAX)
5640 S. Ellis Ave. (708) 922-0499 (home)
Chicago, IL 60637 rivers@cars3.uchicago.edu (Internet)

Subject: Re: curve fitting: works badly?
Posted by vek on Wed, 08 Mar 1995 05:43:44 GMT

View Forum Message <> Reply to Message

Mark Rivers (rivers@cars3.uchicago.edu) wrote:
: I have written a new version of CURVEFIT which will take care of doing the
: derivatives for you numerically.

This is great (really). But has anyone done a (linear) curvefit that includes both X and Y errors? A much harder problem, but much more realistic (c'mon, the Real World's *fun*)! NUMERICAL RECIPES has one for linear fitting (ch. 15.3), but not for non-linear (no surprise, since it's probably nigh impossible). I don't suppose anyone has translated the NR routine into IDL?

Also, why does CURVEFIT want weights instead of errors? W = 1/(sig_y)^2. Whose datasets give them weights? Mine all give me errors! :-)

--

Vincent Kargatis|Rice U.|Houston TX|[http://spacsun.rice.edu/~vek/vek.html]

"It's in the tradition of parties in darkly lit rooms where men wrestled you up against walls and grinded you to death until the break of dawn."
-- Cassandra Wilson, about her album BLUE LIGHT TILL DAWN

Subject: Re: curve fitting: works badly?
Posted by agraps on Wed, 08 Mar 1995 20:44:15 GMT

View Forum Message <> Reply to Message

vek@spacsun.rice.edu (Vincent E. Kargatis) writes:

> This is great (really). But has anyone done a (linear) curvefit that
> includes both X and Y errors? A much harder problem, but much more
> realistic (c'mon, the Real World's *fun*)! NUMERICAL RECIPES has one for
> linear fitting (ch. 15.3), but not for non-linear (no surprise, since it's
> probably nigh impossible). I don't suppose anyone has translated the NR
> routine into IDL?

I think the IDL Goddard Astronomy Library has IDL code to do linear
curvefit with x and y errors. See the IDL FAQ for how to get to the
library. I don't have the library information handy (but Bill Thompson
might :)).

> Also, why does CURVEFIT want weights instead of errors? W = 1/(sig_y)^2.
> Whose datasets give them weights? Mine all give me errors! :-)

Mine too!

Seriously, my understanding of why weights instead of errors
is because:

1) Bevington did it that way.
2) It has a precise mathematical definition that can be implemented
easily in least-squares algorithms.
2) There are several different kinds of errors (i.e. instrumental,
statistical, etc. See Bevington for a good discussion of error analysis).

Speaking of errors, if you have criteria for checking whether a fit is
good, looking at the sigma of the output fit parameters is sometimes
not enough because the parameters are correlated to each other. That's
why I added a covariance output matrix to my version of CURVEFIT that
quantitatively describes the correlated variances between the fit
parameters. See my previous post on the Marquardt-Levenberg thread
here on how to do this.

Amara

--

********** **********
Amara Graps email: agraps@netcom.com

Computational Physicist vita: finger agraps@sunshine.arc.nasa.gov
Intergalactic Reality bio: finger -lm agraps@netcom.com
********** **********

"Awaken the mind without fixing it anywhere." --Kungo Kyo
