Subject: Re: optimization using IDL
Posted by Amara Graps on Tue, 17 Sep 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Karl Young wrote:
>
> I know this has been discussed before and I apologize for any
> redunancy. I am finding IDL's curvefit inadequate for the types
> of optimization I need to do. For one thing I need to do
> constrained optimization. I hacked curvefit to do a kludgy
> version of constrained optimization and a coleague put
> together an even better version but I'm at the point at which
> I really need an "industrial strength" constrained optimizer.
> (I went through the section in Bevington's book from which
> IDL's curvefit was culled and there didn't seem to be any really
> obvious ways to extend that algorithm to do the kind of
> constrained optimization that you find in modern textbooks on
> optimization, i.e. I generally couldn't find any way of
> extending standard Marquardt-Levinson type optimization, but
> maybe I'm reading the wrong books)

Hi Karl,

Do you want to stick with the Marquardt algorithm or try another algorithm? I'm aware of more sophisticated optimization techniques (but have never needed them, and would only be able to give references).

If you want to stick with the Marquardt algorithm, another thing to try is to work with the lambda parameter. The Bevington/Numerical Recipes/IDL routine uses a lambda that is the same value for all of the free parameters. Make lambda an _array_ instead, with a slightly different value for each of the free parameters.

If you are navigating a tricky Chi-square "terrain" where particular parameters give really flat chi-squares and therefore, the iterations go on forever, this technique helps one get out of that problem. One would have to have a pretty good understanding of their function that they are trying to fit, however.

For example, one could do something like the following:

```
;percentage of lambda to increase/decrease
del_flambda = [30.d0,30.d0,30.d0,30.d0,25.d0,25.d0]
```

And so inside of the CURVFIT routine, the flambda looks like:

```
flambda = flambda/del_flambda
```

You can also impose your your own min and max inside the CURVFIT routine.

For example, let's say you have 6 free parameters v(0..5). You could initially set up the minimums/maximums for them:

```
;Min/max values allowed for v()
minmax(0,0)=[v(0)/3,v(1)-3*cf,v(2)/3,.0015,v(4),0]  ;min
minmax(0,1)=[v(0)*3,v(1)+3*cf,v(2)*3,.0027,v(4),0]  ;max

mmn=2   ;Least #iterations to impose temporary minmax values
mmx=100 ;Max #iterations to impose temporary minmax values
```

(This is just an example of min/max's I've used for a curvefitting problem a few years ago.)

Then use these inside of CURVFIT in the iterations loop, checking to see if the value of the parameters have exceeded the min/max's.

The Marquardt curvefitting is also extremely sensitive to initial guesses. That's another place where one's knowledge of their curvefitting function is necessary. If one doesn't have good initial guesses, they're in trouble. I've recently thought that a genetic algorithm method might work well for coming up with good initial guesses, but have never implemented anything like that.

Are your derivatives accurate? That is a really easy place for one to make a mistake, and will make the Marquardt algorithm completely useless.

So I don't know if the above techniques are is the "industrial strength" Marquardt curvefitting, that you were looking for, (and I'm not even sure if these tricks make sense mathematically) but these tricks worked pretty well for me fitting some complicated functions that consisted of integrals, numerical derivatives, and with up to 7 free parameters.

Amara

--

********** *****
Amara Graps amara@quake.stanford.edu

Solar Oscillation Investigations Stanford University
http://quake.stanford.edu/~amara/amara.html
********** *****

"Never fight an inanimate object." - P. J. O'Rourke
