Subject: Re: Transpose(A)*P*A
Posted by Vince Hradil on Fri, 10 Oct 2008 15:03:05 GMT
View Forum Message <> Reply to Message

On Oct 10, 9:25 am, "mapper...@gmail.com" <zjwan...@gmail.com> wrote:
> hello,
>
> I have a question about how to improve the computation speed when deal
> with non-linear equation
> A x = l, P is the weight for each
> row, P is M*M
> M*N N*1 M*1
>
> then I have to build normal matrix which is
> Transpose(A)*P*A x = Transpose(A)*P*l
> N*N N*1 N*1
>
> then x can be solved.
>
> When M is bigger as 20000, N as 3000, the time to build AtPA is almost
> one hour, that is too long. the code is:
>
> PRO NormalMatrix, A, l, ATPA, ATPL, Weight = P
> szA = SIZE(A, /DIMENSIONS)
> if N_ELEMENTS(szA) ne 2 then begin
> mess = dialog_message('Input matrix must be M*N format!', /
> information)
> return
> endif
> szL = SIZE(L, /DIMENSIONS)
> if (N_ELEMENTS(szL) ne 2 and szL[0] ne 1) then begin
> mess = dialog_message('Input l must be 1*M format!', /information)
> return
> endif
> if (szA[1] ne szL[1]) then begin
> mess = dialog_message('Input A and l must be same rows!', /
> information)
> return
> endif
> if not keyword_set(P) then P = fltarr(szA[1])*0.0+1.0
> ATPA = fltarr([szA[0], szA[0]])
> ATPL = fltarr(1, szA[0])
>
> for i=0L, szA[0]-1 do begin
> t1 = systime(1)
> ATPA[i, i:szA[0]-1] = (p*A[i, *])##A[i:szA[0]-1, *]
> ATPA[i:szA[0]-1, i] = ATPA[i, i:szA[0]-1]
> ATPL[0, i] = (p*A[i, *])##l

```
>               t2 = systime(1)
>               print, sza[0], i, t2-t1
>         endfor
>
>  END
>
>  I like to know how I can improve it to few seconds.
```

I think I'm confused - won't be the first time - why not just do the multiplication?
atpa = matrix_multiply(a,p##a,/atranspose)
or maybe it's
atpa = matrix_multiply(p##a,a,/btranspose)

Subject: Re: Transpose(A)*P*A
Posted by Craig Markwardt on Sat, 11 Oct 2008 19:42:15 GMT
View Forum Message <> Reply to Message

"mapper4u6@gmail.com" <zjwang2u@gmail.com> writes:

```
>  hello,
>
>  I have a question about how to improve the computation speed when deal
>  with non-linear equation
>    A      x      =    l,                  P  is the weight for each
>  row, P is M*M
>  M*N    N*1        M*1
>
>  then I have to build normal matrix which is
>  Transpose(A)*P*A          x    =          Transpose(A)*P*l
>  N*N                       N*1             N*1
>
>  then x can be solved.
```

I'm going to channel our vice presidential candidate and answer a different question.

It looks like you are trying to solve a least squares problem. It's well documented that the normal equation method suffers from accuracy problems, basically because you are squaring the A matrix, and thus squaring the errors. Have you tried SVD or QR factorization? Implemented correctly, the execution times should scale as,

```
  Normal equation ~ N^2 * (  M +   N/3)
  QR              ~ N^2 * (2*M - 2*N/3)
  SVD             ~ N^2 * (2*M +11*N/3)
```

On its face, QR factorization will take longer (not more than double the time though), but it is known to be more stable.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: cbmarkwardt+usenet@gmail.com
---------- ----------

Subject: Re: Transpose(A)*P*A
Posted by Karlo Janos on Sun, 12 Oct 2008 11:24:33 GMT
View Forum Message <> Reply to Message

> It looks like you are trying to solve a least squares problem. It's
> well documented that the normal equation method suffers from accuracy
> problems, basically because you are squaring the A matrix, and thus
> squaring the errors. Have you tried SVD or QR factorization?

That sounds interesting. Do you know an algorithm for SVD or QR factorization, which is capable of handling large sparse matrices (10E6 rows, 10E4 columns, 10e7 non-zeros)? Or do I have to use the IMSL extension for IDL?
Greets

Karlo