Subject: Convolving speed issue
Posted by rogass on Thu, 17 Apr 2008 16:29:50 GMT
View Forum Message <> Reply to Message

Hi there,
I have a strange problem with the IDL convolving possibilities. I'd like to make a script which convolves 2 matrices - e.g. a and b - in the same behavior conv2(a,b,'same') Matlab does. The Problem is not to make such a script, the problem is that IDL takes too long or hangs when i try to convolve larger matrices. I tried certainly all kinds of using the built-in IDL-convol method, but convolving large arrays ends always in different results compared to the very fast Matlab conv2.
Maybe someone could help me. Here's the sample code:

```
function size_dim, in, direction
dims = size(in, /dimensions)
return, dims[direction]
end

function zeropadding, in,xsize,ysize
;only for 2D-arrays
xsize_in=size_dim(in,0)
ysize_in=size_dim(in,1)
shiftx  =  ceil((xsize-xsize_in)/2)
shifty  =  ceil((ysize-ysize_in)/2)
temp    =  fltarr(xsize,ysize)
temp[0:xsize_in-1,0:ysize_in-1]  = in
temp  =  shift(temp,shiftx,shifty)
return, temp
end

function conv2, a,b
size_a=[size_dim(a,0),size_dim(a,1)]
size_b=[size_dim(b,0),size_dim(b,1)]
a=zeropadding(a,size_a[0]+size_b[0], size_a[1]+size_b[1])
b=zeropadding(b,size_a[0]+size_b[0], size_a[1]+size_b[1])
c=fltarr(size_a[0]+size_b[0], size_a[1]+size_b[1], /nozero)
addx  =  floor(total(size_a)/4)
endx  =  ceil(double(total(size_a)/4))
addy  =  floor(total(size_b)/4)
endy  =  ceil(double(total(size_b)/4))

for n1=0,size_a[0]+size_b[0]-1 do begin
  for n2=0,size_a[1]+size_b[1]-1 do begin
    temp=0
    temp2=0
    for k1=0+addx,size_a[0]+size_b[0]-1-endx do begin
      for k2=0+addx,size_a[1]+size_b[1]-1-endy do begin
    if n1-k1 gt-1 and n2-k2 gt-1 then begin
     temp=a[k1,k2]*b[n1-k1+addx,n2-k2+addy]
     temp2=temp2+temp
    endif
   endfor
  endfor
  c[n1,n2]=temp2
 endfor
endfor
temp  =  shift(c,-2*addx,-2*addy)
return, temp[0:size_a[0]-1,0:size_b[0]-1]
end


pro conv
; sample matrix -> magic(5) in Matlab
a=    [[17,    24,     1,     8,    15],$
       [23,     5,     7,    14,    16],$
        [4,     6,    13,    20,    22],$
       [10,    12,    19,    21,     3],$
       [11,    18,    25,     2,     9]]
b=2*a
c=a
d=b
print, 'Trying own convolution...',string(10b),conv2(a,b), string(10B)
print, 'Trying built in convolution...',string(10b),$
shift(convol(zeropadding(c,10,10),zeropadding(d,10,10), center=0,/
edge_wrap),-4,-4),string(10b)
end
```

Maybe there is a solution for using reform in some way? It seems to be quicker as for-loops. But I can't imagine how it could work when the indices of the multiplying matrices are varying.

Hope on help

Thank you and best regards

Chris

Subject: Re: Convolving speed issue
Posted by pgrigis on Mon, 21 Apr 2008 15:24:59 GMT
View Forum Message <> Reply to Message

How large are your matrices?
You really need to use the FFT method.
The FFT in IDL should not be significantly slower then in MATLAB,

right?

Ciao,
Paolo

rog...@googlemail.com wrote:
> Dear Paolo,
> it's unfortunately just too slow, so I'm just trying to enhance the
> speed of your well working method. Maybe you have further ideas?
>
> Thanks and best regards
>
> Christian

Subject: Re: Convolving speed issue
Posted by rogass on Mon, 21 Apr 2008 18:25:49 GMT
View Forum Message <> Reply to Message

Dear Paolo,
yes, it isnt so much slower. I could also use fftw, which is available for matlab and also for IDL. But nevertheless, I had to convolve it directly and not in frequency domain. The problem is that the matrices could be very large (their size is changing dynamically) and there are also some iterations (nearly 200) for each convolve. So I always try to use as often as possible reform, replicate and rebin, because those are very fast for manipulating or computing arrays.

But, unfortnately I'm not able to exchange the for-to loops completely. I think, it's quite difficult to do this in the way I mentioned above.

But thanks again for answer, Paolo

Any other ideas?

Best regards

Chris

Subject: Re: Convolving speed issue
Posted by pgrigis on Mon, 21 Apr 2008 18:39:03 GMT
View Forum Message <> Reply to Message

rog...@googlemail.com wrote:
> Dear Paolo,

> yes, it isnt so much slower. I could also use fftw, which is available
> for matlab and also for IDL. But nevertheless, I had to convolve it
> directly and not in frequency domain.

OK, here's where I cannot follow you: why do you care how the
convolution is preformed? If you get the same answer by the
direct (slow) and FFT (fast) method, why would you not want to use
the latter one?

Paolo

> The problem is that the matrices
> could be very large (their size is changing dynamically) and there are
> also some iterations (nearly 200) for each convolve. So I always try
> to use as often as possible reform, replicate and rebin, because those
> are very fast for manipulating or computing arrays.
>
> But, unfortnately I'm not able to exchange the for-to loops
> completely. I think, it's quite difficult to do this in the way I
> mentioned above.
>
> But thanks again for answer, Paolo
>
> Any other ideas?
>
> Best regards
>
> Chris

Subject: Re: Convolving speed issue
Posted by Paul Van Delst[1] on Mon, 21 Apr 2008 19:21:30 GMT

View Forum Message <> Reply to Message

pgrigis@gmail.com wrote:
>
> rog...@googlemail.com wrote:
>> Dear Paolo,
>> yes, it isnt so much slower. I could also use fftw, which is available
>> for matlab and also for IDL. But nevertheless, I had to convolve it
>> directly and not in frequency domain.
>
> OK, here's where I cannot follow you: why do you care how the
> convolution is preformed? If you get the same answer by the
> direct (slow) and FFT (fast) method, why would you not want to use
> the latter one?

I've been lurking about in this thread.

I have to agree with Paolo. If you do convolutions "explicitly" [my terminology] in the original data domain, it's nearly always going to be much slower that doing multiplications in the frequency domain.

MAthematically, the operators should be the same (with due respect paid to numerical precision issues, i.e. always FFT in double precision in IDL! :o)

>> The problem is that the matrices
>> could be very large (their size is changing dynamically) and there are
>> also some iterations (nearly 200) for each convolve. So I always try
>> to use as often as possible reform, replicate and rebin, because those
>> are very fast for manipulating or computing arrays.

Probably more details are required about the iterations for each convolve. What is changing? The data itself? The convolution function? Both?

Without knowing some more of the nitty gritty details of your problem it's hard to comment usefully in a generic manner.

In my experience, however, I've found that doing something like

FFT^-1(FFT(data) * FFT(convfn))

is nearly always much much speedier[*] than doing

Data (x) convfn

where (x) is my ASCII Art representing the convolution operator. :o)

cheers,

paulv

[*] Where the data and convfn have been specified (or zeropadded) to an efficient number of points, FFT-wise.