Subject: Re: The Behavior of CONVOL
Posted by Ken Kump on Fri, 01 Nov 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Kevin R. Turpie wrote:

> First, CONVOL does not appear to perform a convolution by default;
> rather it seems to do a correlation. They are similar, but give
> different results if the kernel is asymmetric.

No, the values are not normalized in any way. You need to be
aware of truncation and padd accordingly.

> Second, when CENTER is set to 0, CONVOL does a convolution in a
> strict sense *if* the input kernel function, say k(x), is defined
> so that k(x) = 0 for all x < 0. The result is usually shifted to
> the right.
> To do a true convolution with CONVOL for any kernel, it seems that
> CENTER must be set to 1 and REVERSE must be applied to each dimension
> of the kernel prior to input.

Yes, this is true. I like to **always** perform a strict
convolution. I set center=0, reverse my convolution
kernel in each direction, and padd zeros on
either side of my input function, then after the convolution,
I may truncate them to be repositioned correctly. I find this
annoying and computationally inefficient. For a convolution
of some magnitude (I think Numerical Recipies gave a number for
a 1-D case of 60) Fourier convolution is **much** faster.
--

Ken Kump

Department of Biomedical Engineering
Case Western Reserve University
Cleveland, Ohio 44106, USA
E-mail: ksk3@po.cwru.edu

Subject: Re: The Behavior of CONVOL
Posted by Kevin R. Turpie on Fri, 01 Nov 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Ken Kump wrote:
>
>> First, CONVOL does not appear to perform a convolution by default;
>> rather it seems to do a correlation. They are similar, but give
>> different results if the kernel is asymmetric.

>
> No, the values are not normalized in any way. You need to be
> aware of truncation and padd accordingly.
>

I'm afraid you missed the point; normalization doesn't have anything
to do with it. It is the orientation of the kernel. In a convolution
the kernel "flipped" wrt to the function being convolved. In a
correlation is is not. With CENTER=1 (default), CONVOL uses the same
orientation as a correlation. With CENTER=0, it does not.

Consider the following 1-D example:

a = [0., 0., 1., 0., 0.]

b = [.4, .5, .1]

print, convol(a,b,/center),form='(9(F4.2,2x))' ; /center is default

0.00 0.10 0.50 0.40 0.00

print, convol(a,b,center=0),form='(9(F4.2,2x))'

0.00 0.00 0.40 0.50 0.10

Pretend this is an optics problem where _b_ is a point-spread
function and _a_ is a 1-D image. 40% of the energy of each pixel
is dumped into the pixel to the left and 10% to the right. The
result should be

0.00 0.40 0.50 0.10 0.00

That clearly doesn't happen in either application of CONVOL.

>> Second, when CENTER is set to 0, CONVOL does a convolution in a
>> strict sense *if* the input kernel function, say k(x), is defined
>> so that k(x) = 0 for all x < 0. The result is usually shifted to
>> the right.
>> To do a true convolution with CONVOL for any kernel, it seems that
>> CENTER must be set to 1 and REVERSE must be applied to each dimension
>> of the kernel prior to input.
>
> Yes, this is true. I like to **always** perform a strict
> convolution. I set center=0, reverse my convolution
> kernel in each direction, and padd zeros on
> either side of my input function, then after the convolution,
> I may truncate them to be repositioned correctly. I find this
> annoying and computationally inefficient. For a convolution

> of some magnitude (I think Numerical Recipies gave a number for
> a 1-D case of 60) Fourier convolution is **much** faster.

Yes, setting CENTER=0 and shifting "left" for each dimension is also
a valid workaround. Consider also this generalized example:

for i=1,(SIZE(b))(0) do b = REVERSE(b, i)
c = CONVOL(a, b, /EDGE_TRUNCATE).

If we have to workaround then I think this will work for you, and
with less effort. Yes, yes, I also use FFT to convolve things
like images and maps; the implementation is not difficult. But,
along with the benefits, there are caveats and limitations there
too.

But now I don't understand your point. Are you saying CONVOL is fine
as long as you do this, this, and that? My concern is that CONVOL
doesn't seem to behave conventionally and requires workarounds to
do common applications. I believe users should be made fully aware
of this before any serious mistakes are made (or worse, published).

Subject: Re: The Behavior of CONVOL
Posted by agraps on Fri, 01 Nov 1996 08:00:00 GMT
View Forum Message <> Reply to Message

"Kevin R. Turpie" <turpie@seaeagle.gsfc.nasa.gov> writes:

> I've found the behavior of CONVOL to be a bit confusing. Please
> let me know if I'm missing something, but here are my observations:

I've noticed confusing behavior too.

[...]

> Second, when CENTER is set to 0, CONVOL does a convolution in a
> strict sense *if* the input kernel function, say k(x), is defined
> so that k(x) = 0 for all x < 0. The result is usually shifted to
> the right.

So that's what's going on... I've noticed the right shift. I've had to
"fix" (fudge?) the first values to get the result that I knew I should
have gotten.

> To do a true convolution with CONVOL for any kernel, it seems that
> CENTER must be set to 1 and REVERSE must be applied to each dimension
> of the kernel prior to input.

This is really useful, thanks.

I wrote a function a couple of years ago that performed the equivalent
of Matlab's "conv" (convolution) function. (It's crude, but it works.)
That's when I first noticed the different behavior. I'll attach it
below.

> PS - If your interested, I did create a routine to perform
> two dimensional convolutions using a FFT. It is *very* fast
> and behaves like CONVOL with the EDGE_WRAP keyword on and
> the kernel oriented properly.

I'm interested.

Amara

```
;********************************************************** ******************
FUNCTION conv, a, b
;+
; NAME:
;  CONV
;
; PURPOSE:
; This function performs a convolution operation like Matlab's conv.
;
; CATEGORY:
; Signal Processing
;
; CALLING SEQUENCE:
;    y = CONV(a, b)
;
; INPUTS:
;    a:  1st vector.
;    b:  2nd vector, the "kernel"
;
; OUTPUTS:
;    y:  a convolved with b. See description in Matlab Reference Manual
;              for conv.
;
;
; MODIFICATION HISTORY:
;  Written by Amara Graps  November, 1994.
;-

;Pad a with zeros of size of the kernel
p = N_ELEMENTS(b)
new_a = [a,FLTARR(p-1)]
```

```
;Apply IDL's convolution function
out = CONVOL(new_a, b, center = 0)

;Calculate results for 1st p-1 values (another Fudge to equal
;Matlab's conv). i.e. if p=5 then the first p values are:
;out(0) = new_a(0)*b(0)
;out(1) = new_a(0)*b(1) + new_a(1)*b(0)
;out(2) = new_a(0)*b(2) + new_a(1)*b(1) + new_a(2)*b(0)
;out(3) = new_a(0)*b(3) + new_a(1)*b(2) + new_a(2)*b(1) + new_a(3)*b(0)
;out(4) = new_a(0)*b(4) + new_a(1)*b(3) + new_a(2)*b(2) + new_a(3)*b(1) +
;         new_a(4)*b(0)

FOR i = 0, p-1 DO BEGIN
 out(i) = new_a(0) * b(i)
 FOR l = 1, i DO BEGIN
  out(i) = out(i) + new_a(l)*b(i-l)
 END ;l
END ;i

RETURN, OUT
END  ;function conv
 ;************************************************************ ************
```

--

 ** *************
Amara Graps   email: agraps@netcom.com
Computational Physics   vita: finger agraps@best.com
Multiplex Answers   URL: http://www.amara.com/
 ** *************