Subject: Re: Convolution with non-constant Kernel?
Posted by James[2] on Tue, 16 Nov 2010 22:42:25 GMT
View Forum Message <> Reply to Message

On Nov 14, 2:30 pm, SonicKenking <ywa...@gmail.com> wrote:
> On Nov 12, 11:48 pm, Bennett <juggernau...@gmail.com> wrote:
>
>
>
>> On Nov 11, 7:56 pm, SonicKenking <ywa...@gmail.com> wrote:
>
>>> Hi, I wonder if there is an easy way to perform convolution on an
>>> array with non-constant kernel.
>
>>> The IDL built-in CONVOL function requires the kernel to be a fixed
>>> array, e.g.
>>> [-1,2,-1]. I want to have a dynamic kernel that changes based on the
>>> position of the array. Something like
>
>>> array = [8,6,7,9,1,3,4,5], kernel=[sin(index_i-1), 2, sin(index_i+1)]
>
>>> Is there any other built-in IDL function that can do this or is there
>>> someone who has already coded this up? If the answer is no, I'll go
>>> ahead and code my own program. Just checking it here beforehand to
>>> avoid re-inventing wheels.
>
>>> Thanks!
>
>> Couldn't you just perform multiple convolutions and then combine the
>> result in the end using a position mask?
>
> That probably won't work since the kernel is 2D non-linear. Also doing
> convolutions multiple times may not be efficient for large arrays.

In the most general case, where the convolution kernel is allowed to change for each pixel, I think the best way to do it in IDL would be to supply an array with a kernel for each pixel. This would be a K+I-dimensional array, where K is the number of kernel dimensions and I is the number of image dimensions. So, for example of a 2D image and 2D 3x3 kernel, kernels[*,*,i,j] would be the 3x3 kernel to use at image location [i,j]. Then you would do some dimensional juggling to replicate the image and line it up with the kernels array, multiply them together, and then do some more juggling with summation to add up each individual convolution and get the result. I am interested in figuring this out, but it will have to wait until I'm off the clock. ;)

By the way, as a math nerd I am unable to resist reminding you that

the operation you're describing is technically not convolution at all,
although it may relate to convolution in a higher-dimensional space.