
Subject: correlation question

Posted by [Helder Marchetto](#) on Wed, 04 Nov 2015 09:36:18 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

I have 3d data with n images of xs x xs images (array size xs, ys, n) and I have a cuve (array) with n elements that describes the behavior of a column [x0,y0,*] of the 3d array.

How can I get some statistical information giving me an array of [xs,ys] elements that tells me how close each pixel is to the input array?

It's a correlation problem, but I'm lost in the ?_correlate jungle...

Thanks for any help or tip.

Cheers, Helder

Subject: Re: correlation question

Posted by [Jeremy Bailin](#) on Wed, 04 Nov 2015 16:38:47 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Wednesday, November 4, 2015 at 3:36:21 AM UTC-6, Helder wrote:

> Hi,

> I have 3d data with n images of xs x xs images (array size xs, ys, n) and I have a cuve (array) with n elements that describes the behavior of a column [x0,y0,*] of the 3d array.

> How can I get some statistical information giving me an array of [xs,ys] elements that tells me how close each pixel is to the input array?

> It's a correlation problem, but I'm lost in the ?_correlate jungle...

>

> Thanks for any help or tip.

>

> Cheers, Helder

What exactly do you mean by 'how close each pixel is to the input array'?

-Jeremy.

Subject: Re: correlation question

Posted by [Helder Marchetto](#) on Wed, 04 Nov 2015 17:27:46 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

Sorry I didn't explain myself properly.

I'm looking for a IDL way to do this:

Similarity = fltarr(xs,us)

RefCurve = {array of n elements}

```
data = {array of xs,us,n elements}
for i=0,xs-1 do begin
  for j=0,ys-1 do begin
    Similarity = "correlation between reform(data[i,j,*]) and RefCurve"
  endfor
endfor
```

With correlation I mean a 1 if the two curves (arrays) are similar (e.g.two lines with slope 1), a zero (or -1) if the lines are perpendicular.

[Sorry, but I'm writing from a mobile device...]

By thresholding (e.g. at ≥ 0.9) the resulting Similarity image I can "see" where the evolution of the [xs,ys] slices in data resembles that of RefCurve.

Did I make myself clear? If not, I have to come up with a practical example... Sorry if my explanation is so poor.

Cheers, Helder

Subject: Re: correlation question
Posted by [Helder Marchetto](#) on Thu, 05 Nov 2015 11:27:46 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Wednesday, November 4, 2015 at 6:27:49 PM UTC+1, Helder wrote:

```
> Hi,
> Sorry I didn't explain myself properly.
> I'm looking for a IDL way to do this:
>
> Similarity = fltarr(xs,us)
> RefCurve = {array of n elements}
> data = {array of xs,us,n elements}
> for i=0,xs-1 do begin
>   for j=0,ys-1 do begin
>     Similarity = "correlation between reform(data[i,j,*]) and RefCurve"
>   endfor
> endfor
>
> With correlation I mean a 1 if the two curves (arrays) are similar (e.g.two lines with slope 1), a
> zero (or -1) if the lines are perpendicular.
>
> [Sorry, but I'm writing from a mobile device...]
>
> By thresholding (e.g. at  $\geq 0.9$ ) the resulting Similarity image I can "see" where the evolution of
> the [xs,ys] slices in data resembles that of RefCurve.
>
> Did I make myself clear? If not, I have to come up with a practical example... Sorry if my
```

explanation is so poor.

>

> Cheers, Helder

Hi,

ok, I did my homework and here is what I need: for each column of the 3d array [x0,y0,*] (that means that x0 goes from 0 to xs, and y0 goes from 0 to ys), I want to calculate the Pearson product-moment correlation coefficient with respect to an arbitrary array (of n elements) and therefore obtain a matrix of xs by ys elements.

The Pearson coefficient is the ratio between the covariance and the std-dev and is calculated in IDL by the correlate() function. Now I just have to find a way to avoid the loop... I'll post when I have the solution.

Thanks for making me think (!)

Cheers,

Helder

Subject: Re: correlation question

Posted by [Helder Marchetto](#) on Thu, 05 Nov 2015 12:37:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Thursday, November 5, 2015 at 12:27:49 PM UTC+1, Helder wrote:

> On Wednesday, November 4, 2015 at 6:27:49 PM UTC+1, Helder wrote:

>> Hi,

>> Sorry I didn't explain myself properly.

>> I'm looking for a IDL way to do this:

>>

>> Similarity = fltarr(xs,us)

>> RefCurve = {array of n elements}

>> data = {array of xs,us,n elements}

>> for i=0,xs-1 do begin

>> for j=0,ys-1 do begin

>> Similarity = "correlation between reform(data[i,j,*]) and RefCurve"

>> endfor

>> endfor

>>

>> With correlation I mean a 1 if the two curves (arrays) are similar (e.g.two lines with slope 1), a zero (or -1) if the lines are perpendicular.

>>

>> [Sorry, but I'm writing from a mobile device...]

>>

>> By thresholding (e.g. at gt 0.9) the resulting Similarity image I can "see" where the evolution of the [xs,ys] slices in data resembles that of RefCurve.

>>

>> Did I make myself clear? If not, I have to come up with a practical example... Sorry if my explanation is so poor.

```

>>
>> Cheers, Helder
>
> Hi,
> ok, I did my homework and here is what I need: for each column of the 3d array [x0,y0,*] (that
> means that x0 goes from 0 to xs, and y0 goes from 0 to ys), I want to calculate the Pearson
> product-moment correlation coefficient with respect to an arbitrary array (of n elements) and
> therefore obtain a matrix of xs by ys elements.
>
> The Pearson coefficient is the ratio between the covariance and the std-dev and is calculated in
> IDL by the correlate() function. Now I just have to find a way to avoid the loop... I'll post when I
> have the solution.
>
> Thanks for making me think (!)
> Cheers,
> Helder

```

Ok, so I keep replying to myself. Here is the solution that I got. It's based on the IDL correlate() function and I build up from there.

I tested it on a 512x509x6 array using the function below and going through every index in the "rough" (non-IDL) way.

Using correlatelmage it took 0.063 seconds and going through every "pixel" column using this:

```

for i=0,sData[0]-1 do begin
  for j=0,sData[1]-1 do begin
    corrlmg[i,j] = correlate(reform(subCube[i,j,*]), arr)
  endfor
endfor
it took 1.7 seconds.

```

So it was worth the effort :-)

I hope I'm not the only one finding this useful. If you find it useful and have some improvements suggestions, plz let me know... I'll be posting this here: <http://idl.marchetto.de/correlation-image/>

Cheers,
Helder

```

function correlatelmage, img, arr, double=doubleIn
on_error, 2
slmg = size(img, /dimensions)
if n_elements(slmg) ne 3 then begin
  message, 'Input matrix must have 3 dimensions', /continue
  return, -1
endif
sArr = size(arr, /dimensions)
if n_elements(sArr) ne 1 then begin
  message, 'Input array must have 1 dimension', /continue

```

```

    return, -1
endif
sArr = sArr[0]
if sArr ne slmg[2] then begin
    message, 'Input array must have same number of elements of the third input matrix dimension',
/continue
    return, -1
endif

typelmng = size(img, /type)
typeArr = size(arr, /type)
dbl = (n_elements(doubleIn) gt 0) ? keyword_set(doubleIn) : (typelmng eq 5 || typeArr eq 5)
cplx = typelmng eq 6 || typelmng eq 9 || typeArr eq 6 || typeArr eq 9

imgMean = total(img, 3, double = dbl) / sArr
imgDev = img
for i=0,slmg[2]-1 do imgDev[*,* ,i] = imgDev[*,* ,i] - imgMean

arrMean = mean(arr, double = dbl)
arrDev = arr - arrMean

nan = dbl ? !VALUES.D_NAN : !VALUES.F_NAN

dlmg2 = total(abs(imgDev)^2,3, double = dbl)
dArr2 = total(abs(arrDev)^2, double = dbl)
if dArr2 eq 0 then return, nan
void = where(dlmg2 eq 0,cntBadVals)
if cntBadVals gt 0 then return, nan
arrDevCube = imgDev
for i=0,slmg[2]-1 do arrDevCube[*,* ,i] = arrDev[i]
if cplx then return, -1 > total(imgDev * conj(arrDevCube), double=dbl) / (sqrt(dlmg2)*sqrt(dArr2)) <
1 $
    else return, total(imgDev * arrDevCube,3, double=dbl) / (sqrt(dlmg2)*sqrt(dArr2))
end

```

Subject: Re: correlation question
Posted by [lecacheux.alain](#) on Thu, 05 Nov 2015 15:05:23 GMT
[View Forum Message](#) <> [Reply to Message](#)

Le jeudi 5 novembre 2015 13:38:00 UTC+1, Helder a écrit :
 > On Thursday, November 5, 2015 at 12:27:49 PM UTC+1, Helder wrote:
 >> On Wednesday, November 4, 2015 at 6:27:49 PM UTC+1, Helder wrote:
 >>> Hi,
 >>> Sorry I didn't explain myself properly.
 >>> I'm looking for a IDL way to do this:
 >>>
 >>> Similarity = fltarr(xs,us)

```

>>> RefCurve = {array of n elements}
>>> data = {array of xs,us,n elements}
>>> for i=0,xs-1 do begin
>>>   for j=0,ys-1 do begin
>>>     Similarity = "correlation between reform(data[i,j,*]) and RefCurve"
>>>   endfor
>>> endfor
>>>
>>> With correlation I mean a 1 if the two curves (arrays) are similar (e.g.two lines with slope 1),
a zero (or -1) if the lines are perpendicular.
>>>
>>> [Sorry, but I'm writing from a mobile device...]
>>>
>>> By thresholding (e.g. at gt 0.9) the resulting Similarity image I can "see" where the evolution
of the [xs,ys] slices in data resembles that of RefCurve.
>>>
>>> Did I make myself clear? If not, I have to come up with a practical example... Sorry if my
explanation is so poor.
>>>
>>> Cheers, Helder
>>
>> Hi,
>> ok, I did my homework and here is what I need: for each column of the 3d array [x0,y0,*] (that
means that x0 goes from 0 to xs, and y0 goes from 0 to ys), I want to calculate the Pearson
product-moment correlation coefficient with respect to an arbitrary array (of n elements) and
therefore obtain a matrix of xs by ys elements.
>>
>> The Pearson coefficient is the ratio between the covariance and the std-dev and is calculated
in IDL by the correlate() function. Now I just have to find a way to avoid the loop... I'll post when I
have the solution.
>>
>> Thanks for making me think (!)
>> Cheers,
>> Helder
>
> Ok, so I keep replying to myself. Here is the solution that I got. It's based on the IDL correlate()
function and I build up from there.
> I tested it on a 512x509x6 array using the function below and going through every index in the
"rough" (non-IDL) way.
> Using correlateImage it took 0.063 seconds and going through every "pixel" column using this:
> for i=0,sData[0]-1 do begin
>   for j=0,sData[1]-1 do begin
>     corrImg[i,j] = correlate(reform(subCube[i,j,*]), arr)
>   endfor
> endfor
> it took 1.7 seconds.
>
> So it was worth the effort :-)
```

```

>
> I hope I'm not the only one finding this useful. If you find it useful and have some improvements
> suggestions, plz let me know... I'll be posting this here: http://idl.marchetto.de/correlation-image/
>
> Cheers,
> Helder
>
>
> function correlatelmage, img, arr, double=doubleIn
> on_error, 2
> slmg = size(img, /dimensions)
> if n_elements(slmg) ne 3 then begin
>   message, 'Input matrix must have 3 dimensions', /continue
>   return, -1
> endif
> sArr = size(arr, /dimensions)
> if n_elements(sArr) ne 1 then begin
>   message, 'Input array must have 1 dimension', /continue
>   return, -1
> endif
> sArr = sArr[0]
> if sArr ne slmg[2] then begin
>   message, 'Input array must have same number of elements of the third input matrix
> dimension', /continue
>   return, -1
> endif
>
> typelm = size(img, /type)
> typeArr = size(arr, /type)
> dbl = (n_elements(doubleIn) gt 0) ? keyword_set(doubleIn) : (typelm eq 5 || typeArr eq 5)
> cplx = typelm eq 6 || typelm eq 9 || typeArr eq 6 || typeArr eq 9
>
> imgMean = total(img, 3, double = dbl) / sArr
> imgDev = img
> for i=0,slmg[2]-1 do imgDev[*,*,i] = imgDev[*,*,i] - imgMean
>
> arrMean = mean(arr, double = dbl)
> arrDev = arr - arrMean
>
> nan = dbl ? !VALUES.D_NAN : !VALUES.F_NAN
>
> dImg2 = total(abs(imgDev)^2,3, double = dbl)
> dArr2 = total(abs(arrDev)^2, double = dbl)
> if dArr2 eq 0 then return, nan
> void = where(dImg2 eq 0,cntBadVals)
> if cntBadVals gt 0 then return, nan
> arrDevCube = imgDev
> for i=0,slmg[2]-1 do arrDevCube[*,*,i] = arrDev[i]

```

```
> if cplx then return, -1 > total(imgDev * conj(arrDevCube), double=dbl)/ (sqrt(dImg2)*sqrt(dArr2))  
< 1 $  
>      else return, total(imgDev * arrDevCube,3, double=dbl) / (sqrt(dImg2)*sqrt(dArr2))  
> end
```

I guess that you might try FFT by doing:

```
dims = Size(images, /DIM) ;images is your 3D array  
ft = fft(images, DIM=3)  
ft0 = conj(ft[x0,y0,*])  
ft0 = ft0[intarr(dims[0]),intarr(dims[1]),*] ;REBIN not allowed in complex  
corimg = real_part(fft(ft*ft0, DIM=3, /INVERSE)
```

alx.
