Subject: Singular Value Decomposition in 3 Dimensions
Posted by tomandwilltamu08 on Tue, 02 Sep 2008 16:33:36 GMT
View Forum Message <> Reply to Message

I am wondering how to do Singular Value Decomposition in 3 Dimensions
in IDL. All of the canned routines seem to work only on 2D arrays.

Specifically, I am trying to preform Principle Component Analysis on
stacks of 2D images.

For example, how can one preform an SVD on a 2048x2048xn array to get
2048x2048 principle components?

Thanks much,
-Will

Subject: Re: Singular Value Decomposition in 3 Dimensions
Posted by Mort Canty on Wed, 03 Sep 2008 11:46:29 GMT
View Forum Message <> Reply to Message

tomandwilltamu08@gmail.com schrieb:
> I am wondering how to do Singular Value Decomposition in 3 Dimensions
> in IDL. All of the canned routines seem to work only on 2D arrays.
>
> Specifically, I am trying to preform Principle Component Analysis on
> stacks of 2D images.
>
> For example, how can one preform an SVD on a 2048x2048xn array to get
> 2048x2048 principle components?
>
> Thanks much,
> -Will

SVD is a decomposition theorem for matrices (2D arrays). I think you may
have an incorrect understanding of what principal (not principle)
component analysis means. If you apply PCA to a stack of n 2048x2048
images, you will get n 2048x2048 principal component images. The n
images will be uncorrelated and stacked in the order of decreasing
variance.

Mort

Subject: Re: Singular Value Decomposition in 3 Dimensions
Posted by Juggernaut on Wed, 03 Sep 2008 11:52:00 GMT
View Forum Message <> Reply to Message

On Sep 2, 12:33 pm, tomandwilltam...@gmail.com wrote:
> I am wondering how to do Singular Value Decomposition in 3 Dimensions
> in IDL. All of the canned routines seem to work only on 2D arrays.
>
> Specifically, I am trying to preform Principle Component Analysis on
> stacks of 2D images.
>
> For example, how can one preform an SVD on a 2048x2048xn array to get
> 2048x2048 principle components?
>
> Thanks much,
> -Will

If you want the principal components for the
3D array you can do something like this
sz = size(array, /dimensions)
newArray = fltarr(sz[2], sz[1]*sz[0])
FOR i=0, sz[2]-1 DO BEGIN
  newArray[i,*] = transpose(reform(array[*,*,i], sz[0]*sz[1]))
ENDFOR
result = pcomp(newArray, eigenvalues=evals, /standardize)

pcomp() is IDLs built in for doing PCA and result will be
an array of I believe the same dimensions of newArray which to
get back into viewing form you could just reform it back like

tv, reform(result[0,*],sz[0],sz[1])

There may be better ways of doing it but I may as well give
you a point to jump off of

Subject: Re: Singular Value Decomposition in 3 Dimensions
Posted by Juggernaut on Wed, 03 Sep 2008 11:56:07 GMT
View Forum Message <> Reply to Message

On Sep 3, 7:52 am, Bennett <juggernau...@gmail.com> wrote:
> On Sep 2, 12:33 pm, tomandwilltam...@gmail.com wrote:
>
>> I am wondering how to do Singular Value Decomposition in 3 Dimensions
>> in IDL. All of the canned routines seem to work only on 2D arrays.
>
>> Specifically, I am trying to preform Principle Component Analysis on
>> stacks of 2D images.
>
>> For example, how can one preform an SVD on a 2048x2048xn array to get
>> 2048x2048 principle components?
>

>> Thanks much,
>> -Will
>
> If you want the principal components for the
> 3D array you can do something like this
> sz = size(array, /dimensions)
> newArray = fltarr(sz[2], sz[1]*sz[0])
> FOR i=0, sz[2]-1 DO BEGIN
> newArray[i,*] = transpose(reform(array[*,*,i], sz[0]*sz[1]))
> ENDFOR
> result = pcomp(newArray, eigenvalues=evals, /standardize)
>
> pcomp() is IDLs built in for doing PCA and result will be
> an array of I believe the same dimensions of newArray which to
> get back into viewing form you could just reform it back like
>
> tv, reform(result[0,*],sz[0],sz[1])
>
> There may be better ways of doing it but I may as well give
> you a point to jump off of

By the way the for loop can be eliminated by just putting
transpose(reform(array, sz[0]*sz[1], sz[2])) into pcomp

> FOR i=0, sz[2]-1 DO BEGIN
> newArray[i,*] = transpose(reform(array[*,*,i], sz[0]*sz[1]))
> ENDFOR
> result = pcomp(newArray, eigenvalues=evals, /standardize)

becomes

result = pcomp(transpose(reform(array, sz[0]*sz[1], sz[2])), ...)