Subject: Can this be done using array operations instead?
Posted by robintw on Thu, 02 Apr 2009 14:34:58 GMT
View Forum Message <> Reply to Message

Hi,

I'm quite new to IDL and I've written a function to calculate some statistics for an image file in ENVI. The function has to be calculated for the 3x3 square around every pixel in the image. At the moment I'm using a couple of nested for loops to do that - is there a way to do it using array operations instead?

What I suppose I'm asking for is a way to call a particular function for every element of an array - is that possible. I know it's possible to do NewArray = OldArray * 2, which is calling the + function on every element of the array, but is there a way to call my user-defined function on each element?

I can post code if that would help.

Cheers,

Robin

Subject: Re: Can this be done using array operations instead?
Posted by Juggernaut on Thu, 02 Apr 2009 15:58:30 GMT
View Forum Message <> Reply to Message

On Apr 2, 10:34 am, robintw <r.t.wil...@rmplc.co.uk> wrote:
> Hi,
>
> I'm quite new to IDL and I've written a function to calculate some
> statistics for an image file in ENVI. The function has to be
> calculated for the 3x3 square around every pixel in the image. At the
> moment I'm using a couple of nested for loops to do that - is there a
> way to do it using array operations instead?
>
> What I suppose I'm asking for is a way to call a particular function
> for every element of an array - is that possible. I know it's possible
> to do NewArray = OldArray * 2, which is calling the + function on
> every element of the array, but is there a way to call my user-defined
> function on each element?
>
> I can post code if that would help.
>
> Cheers,
>

> Robin

I don't know exactly what "function" you're wanting to perform on
these pixels but CONVOL will operate in such a way.
IDL> x = indgen(3,3)
IDL> print, convol(x, [[1,1,1],[1,0,1],[1,1,1]], /center)
```
      0       0       0
      0      32       0
      0       0       0
```

This will add up the surrounding pixels giving you the value as the
new center pixel...don't know if that's helpful but enjoy.

Subject: Re: Can this be done using array operations instead?
Posted by robintw on Thu, 02 Apr 2009 16:26:18 GMT
View Forum Message <> Reply to Message

Thanks for the response.

I'm trying to perform a function on each 3x3 square which calculates
something called a Getis statistic - which involves calculating a
formula which needs the sum of the values in 3x3 square, their mean,
variance etc.

Is there a way to call a generic user-defined function on each value
in an array - so I can have a function Getis which can be called on
each value to create a new output array?

Cheers,

Robin

Subject: Re: Can this be done using array operations instead?
Posted by Allan Whiteford on Thu, 02 Apr 2009 17:16:30 GMT
View Forum Message <> Reply to Message

Robin,

robintw wrote:
> Hi,
>
> I'm quite new to IDL and I've written a function to calculate some
> statistics for an image file in ENVI. The function has to be
> calculated for the 3x3 square around every pixel in the image. At the
> moment I'm using a couple of nested for loops to do that - is there a

> way to do it using array operations instead?
>

Probably :).

> What I suppose I'm asking for is a way to call a particular function
> for every element of an array - is that possible. I know it's possible
> to do NewArray = OldArray * 2, which is calling the + function on
> every element of the array, but is there a way to call my user-defined
> function on each element?
>

I think you want to do something slightly more complicated, you want to call your user defined function on groups of array elements rather than invididual array elements.

> I can post code if that would help.
>

If you post stand-alone code which does what you want using loops then chances are someone can point you in the right direction of code which does it using array operations.

Particularly useful is the ability to do stuff like this:

NewArray = (OldArray[0:n_elements(OldArray)-1] + OldArray[1:*])/2

which will give you the mean between adjacent elements in oldarray (oldarray assumed to be 1D). Note that you need to worry about what happens at the edges as soon as you start doing stuff like this. Note also that the above example is just to illustrate the type of array techniques you could use.

> Cheers,
>
> Robin

Thanks,

Allan

Subject: Re: Can this be done using array operations instead?
Posted by robintw on Thu, 02 Apr 2009 17:38:46 GMT
View Forum Message <> Reply to Message

Hi Allan,

Thanks for the very informative reply.

The sort of technique that you suggest at the end looks like the sort of thing I want to do - but I'm not entirely sure how to go about it.

I've looked at my code again and realised that many values in my formula are constant (yes I know some of them can be taken out of the loop in the following code - but once I realised I might be able to do it with array functions I decided not to try and improve that code any more until I'd converted it to use array functions), and in fact only a few things vary. What I need to do is step through each 3x3 square in the array (not sure exactly what to do about the edges) and get the total for each of those squares, as well as the number of values in the square (normally 9 obviously, but if it's at the edge then there might be less). I then need to run a calculation which includes these total and number, as well as some other constant values.

I've posted my code below, but I'd prefer it if you could describe in general how to do these kind of things with some useful examples, rather than just taking my code and writing the new version for me. This code is part of a project I'm doing for university, and, although the code is not the main part of the project (the code is just to help me implement a new technique for image processing) I'd rather write the code mostly myself.

Here is the code:

```
; Creates a Getis image given a FID, the dimensions of the file, a
distance to use for the getis routine
; and a base window to send progress updates to
PRO CREATE_GETIS_IMAGE, file, dims, distance, report_base

  ; Print debugging info
  print, "Distance", distance

  ; TODO: Get this to loop through bands
  ; Get the data for the first band of the file (ignores pos from
earlier)
  WholeBand = ENVI_GET_DATA(fid=file, dims=dims, pos=0)

  ; Calculate the dimensions of WholeBand
  SizeInfo = SIZE(WholeBand, /DIMENSIONS)
  NumRows = SizeInfo[0]
  NumCols = SizeInfo[1]

  ; Let the progress bar know what the denominator of the fraction is
- the max
  ENVI_REPORT_INC, report_base, NumRows
```

```
  ; --- Calculate variable values for the WholeBand

  ; Get the global mean
  GlobMean = MEAN(WholeBand)

  ; Get the global variance
  GlobVariance = VARIANCE(WholeBand)

  ; Get the number of values in the whole image
  GlobNumber = NumRows * NumCols

  ; Create the output array to store the Getis values in - NB: Must be
an array of floats
  OutputArray = FLTARR(NumRows, NumCols)

  ; For each pixel in the image
  FOR Rows = 0, NumRows - 1 DO BEGIN

    ; Send an update to the progress window telling it to let us know
if cancel has been pressed
    ENVI_REPORT_STAT, report_base, Rows, NumRows - 1, cancel=cancelled

    ; If cancel has been pressed then...
    IF cancelled EQ 1 THEN BEGIN
      ; Close the progress window
      ENVI_REPORT_INIT,base=report_base, /FINISH
      ; Exit the function
      RETURN
    ENDIF
    FOR Cols = 0, NumCols - 1 DO BEGIN
      ; Make sure RowBottom doesn't go below 0
      RowBottom = Rows - Distance
      IF RowBottom LT 0 THEN RowBottom = 0

      ; Make sure RowTop doesn't go above NumRows
      RowTop = Rows + Distance
      IF RowTop GE NumRows THEN RowTop = NumRows - 1

      ColBottom = Cols - Distance
      IF ColBottom LT 0 THEN ColBottom = 0

      ColTop = Cols + Distance
      IF ColTop GE NumCols THEN ColTop = (NumCols - 1)

      ; Get the subset of the image corresponding to the Area of
Interest (AOI)
      AOI = WholeBand[RowBottom:RowTop, ColBottom:ColTop]
```

```
      ; Calculate the getis value for this AOI
      getis = CALCULATE_GETIS(GlobMean, GlobVariance, GlobNumber, AOI)

      ; Set the pixel in the output image equal to the getis value
      OutputArray[Rows, Cols] = getis
    ENDFOR
  ENDFOR

  ; Code to scale 0-255 - not used at the moment

  ;MaxOutputArray = MAX(OutputArray)
  ;MinOutputArray = MIN(OutputArray)
  ;RangeOutputArray = MaxOutputArray - MinOutputArray

  ;print, MaxOutputArray
  ;print, MinOutputArray
  ;print, RangeOutputArray

  ;ScaledArray = OutputArray - MinOutputArray
  ;ScaledArray = ScaledArray * (255 / MaxOutputArray - MinOutputArray)

  ; Write the data to an image in ENVI memory
  ENVI_ENTER_DATA, OutputArray

  ; Close the progress window
  ENVI_REPORT_INIT,base=report_base, /FINISH
END

; Calculates the getis value for an AOI given the AOI as an array, and
various
; values of global constants - Mean, Variance and Number of pixels
FUNCTION CALCULATE_GETIS, GlobMean, GlobVariance, GlobNumber, AOI
  ; --- Calculate variable values for the AOI

  ; Get the Sum of the values in the AOI
  AOISum = TOTAL(aoi)

  ; Get number of values in AOI
  SizeInfo = SIZE(aoi, /DIMENSIONS)
  SizeOfSize = SIZE(SizeInfo, /DIMENSIONS)
  IF SizeOfSize EQ 2 THEN AOINumber = SizeInfo[0] * SizeInfo[1]
  IF SizeOfSize EQ 1 THEN AOINumber = SizeInfo[0]

  ; --- Start Calculating Getis Statistic

  ; Calculate the top of the fraction
  TopFraction = AOISum - (FLOAT(AOINumber) * GlobMean)
```

```
  ; Calculate the square root
  SquareRootAnswer = SQRT((FLOAT(AOINumber) * (GlobNumber -
AOINumber))/(GlobNumber - 1))

  ; Calculate bottom of fraction
  BottomFraction = GlobVariance * SquareRootAnswer

  ; Calculate Getis Statistic
  Getis = TopFraction / BottomFraction

  RETURN, Getis
END
```

Thanks,

Robin

Subject: Re: Can this be done using array operations instead?
Posted by Jean H. on Thu, 02 Apr 2009 18:39:57 GMT
View Forum Message <> Reply to Message

Hi Robin,

ok, so basically you are computing the sum and the square root of your 3*3 images. One option can be, if your computer memory permits it, to create copies of your image, and to shift them...

```
data
data1 = shift(data,1)
data2 = shift(data,1,1)
data3 = shift(data,1,-1)
data4 = shift(data,0,1)
data5 = shift(data,0,-1)
data6 = shift(data,-1)
data7 = shift(data,-1,1)
data8 = shift(data,-1,-1)
```

then you can do whatever you want...

```
sum = data + data1 + ...+ data8
sqrt = sqrt(sum)
```

You get the result for every pixel.

Be careful on the edges, again.

Now this will be a problem if your image is a big one.

Jean

robintw wrote:
> Hi Allan,
>
> Thanks for the very informative reply.
>
> The sort of technique that you suggest at the end looks like the sort
> of thing I want to do - but I'm not entirely sure how to go about it.
>
> I've looked at my code again and realised that many values in my
> formula are constant (yes I know some of them can be taken out of the
> loop in the following code - but once I realised I might be able to do
> it with array functions I decided not to try and improve that code any
> more until I'd converted it to use array functions), and in fact only
> a few things vary. What I need to do is step through each 3x3 square
> in the array (not sure exactly what to do about the edges) and get the
> total for each of those squares, as well as the number of values in
> the square (normally 9 obviously, but if it's at the edge then there
> might be less). I then need to run a calculation which includes these
> total and number, as well as some other constant values.
>
> I've posted my code below, but I'd prefer it if you could describe in
> general how to do these kind of things with some useful examples,
> rather than just taking my code and writing the new version for me.
> This code is part of a project I'm doing for university, and, although
> the code is not the main part of the project (the code is just to help
> me implement a new technique for image processing) I'd rather write
> the code mostly myself.
>
> Here is the code:
>
> ; Creates a Getis image given a FID, the dimensions of the file, a
> distance to use for the getis routine
> ; and a base window to send progress updates to
> PRO CREATE_GETIS_IMAGE, file, dims, distance, report_base
>
> ; Print debugging info
> print, "Distance", distance
>
> ; TODO: Get this to loop through bands
> ; Get the data for the first band of the file (ignores pos from
> earlier)
> WholeBand = ENVI_GET_DATA(fid=file, dims=dims, pos=0)
>
> ; Calculate the dimensions of WholeBand

```
>    SizeInfo = SIZE(WholeBand, /DIMENSIONS)
>    NumRows = SizeInfo[0]
>    NumCols = SizeInfo[1]
>
>    ; Let the progress bar know what the denominator of the fraction is
>  - the max
>    ENVI_REPORT_INC, report_base, NumRows
>
>    ; --- Calculate variable values for the WholeBand
>
>    ; Get the global mean
>    GlobMean = MEAN(WholeBand)
>
>    ; Get the global variance
>    GlobVariance = VARIANCE(WholeBand)
>
>    ; Get the number of values in the whole image
>    GlobNumber = NumRows * NumCols
>
>    ; Create the output array to store the Getis values in - NB: Must be
>  an array of floats
>    OutputArray = FLTARR(NumRows, NumCols)
>
>    ; For each pixel in the image
>    FOR Rows = 0, NumRows - 1 DO BEGIN
>
>      ; Send an update to the progress window telling it to let us know
>  if cancel has been pressed
>      ENVI_REPORT_STAT, report_base, Rows, NumRows - 1, cancel=cancelled
>
>      ; If cancel has been pressed then...
>      IF cancelled EQ 1 THEN BEGIN
>        ; Close the progress window
>        ENVI_REPORT_INIT,base=report_base, /FINISH
>        ; Exit the function
>        RETURN
>      ENDIF
>      FOR Cols = 0, NumCols - 1 DO BEGIN
>        ; Make sure RowBottom doesn't go below 0
>        RowBottom = Rows - Distance
>        IF RowBottom LT 0 THEN RowBottom = 0
>
>        ; Make sure RowTop doesn't go above NumRows
>        RowTop = Rows + Distance
>        IF RowTop GE NumRows THEN RowTop = NumRows - 1
>
>        ColBottom = Cols - Distance
>        IF ColBottom LT 0 THEN ColBottom = 0
```

```
>
>       ColTop = Cols + Distance
>       IF ColTop GE NumCols THEN ColTop = (NumCols - 1)
>
>       ; Get the subset of the image corresponding to the Area of
> Interest (AOI)
>       AOI = WholeBand[RowBottom:RowTop, ColBottom:ColTop]
>
>       ; Calculate the getis value for this AOI
>       getis = CALCULATE_GETIS(GlobMean, GlobVariance, GlobNumber, AOI)
>
>       ; Set the pixel in the output image equal to the getis value
>       OutputArray[Rows, Cols] = getis
>     ENDFOR
>   ENDFOR
>
>   ; Code to scale 0-255 - not used at the moment
>
>   ;MaxOutputArray = MAX(OutputArray)
>   ;MinOutputArray = MIN(OutputArray)
>   ;RangeOutputArray = MaxOutputArray - MinOutputArray
>
>   ;print, MaxOutputArray
>   ;print, MinOutputArray
>   ;print, RangeOutputArray
>
>   ;ScaledArray = OutputArray - MinOutputArray
>   ;ScaledArray = ScaledArray * (255 / MaxOutputArray - MinOutputArray)
>
>   ; Write the data to an image in ENVI memory
>   ENVI_ENTER_DATA, OutputArray
>
>   ; Close the progress window
>   ENVI_REPORT_INIT,base=report_base, /FINISH
> END
>
> ; Calculates the getis value for an AOI given the AOI as an array, and
> various
> ; values of global constants - Mean, Variance and Number of pixels
> FUNCTION CALCULATE_GETIS, GlobMean, GlobVariance, GlobNumber, AOI
>   ; --- Calculate variable values for the AOI
>
>   ; Get the Sum of the values in the AOI
>   AOISum = TOTAL(aoi)
>
>   ; Get number of values in AOI
>   SizeInfo = SIZE(aoi, /DIMENSIONS)
>   SizeOfSize = SIZE(SizeInfo, /DIMENSIONS)
```

```
>    IF SizeOfSize EQ 2 THEN AOINumber = SizeInfo[0] * SizeInfo[1]
>    IF SizeOfSize EQ 1 THEN AOINumber = SizeInfo[0]
>
>    ; --- Start Calculating Getis Statistic
>
>    ; Calculate the top of the fraction
>    TopFraction = AOISum - (FLOAT(AOINumber) * GlobMean)
>
>    ; Calculate the square root
>    SquareRootAnswer = SQRT((FLOAT(AOINumber) * (GlobNumber -
>  AOINumber))/(GlobNumber - 1))
>
>    ; Calculate bottom of fraction
>    BottomFraction = GlobVariance * SquareRootAnswer
>
>    ; Calculate Getis Statistic
>    Getis = TopFraction / BottomFraction
>
>    RETURN, Getis
>  END
>
>  Thanks,
>
>  Robin
```

Subject: Re: Can this be done using array operations instead?
Posted by robintw on Thu, 02 Apr 2009 18:58:35 GMT
View Forum Message <> Reply to Message

Hi,

Thanks for the response, but I think that will take too much memory. The system will have to be able to deal with very large images from satellite remote sensing systems, so I don't think copying will be viable.

Thanks for the suggestion though,

Robin

Subject: Re: Can this be done using array operations instead?
Posted by robintw on Thu, 02 Apr 2009 19:43:05 GMT
View Forum Message <> Reply to Message

Thank you everyone for your help. I've now managed to write a function

to do this with array functions. I've put the code below if anyone is interested. I realised that all I really needed was the sum of the values, and I could do all the rest by arithmetic operations on the array itself.

However, the next part of this project will require using subarrays as I will need to get the Standard Deviation of each 3x3 square. Is there a way to do this?

Cheers,

Robin

Code Here:

```
; Creates a Getis image given a FID, the dimensions of the file, a
distance to use for the getis routine
; and a base window to send progress updates to
PRO CREATE_GETIS_IMAGE, file, dims, distance, report_base
  ; TODO: Get this to loop through bands
  ; Get the data for the first band of the file (ignores pos from
earlier)
  WholeBand = ENVI_GET_DATA(fid=file, dims=dims, pos=0)

  ; Calculate the dimensions of WholeBand
  SizeInfo = SIZE(WholeBand, /DIMENSIONS)
  NumRows = SizeInfo[0]
  NumCols = SizeInfo[1]

  ; Get the global mean
  GlobMean = MEAN(WholeBand)

  ; Get the global variance
  GlobVariance = VARIANCE(WholeBand)

  ; Get the number of values in the whole image
  GlobNumber = NumRows * NumCols

  DimOfArray = (distance * 2) + 1
  Kernel = FLTARR(DimOfArray, DimOfArray)
  Kernel = Kernel + 1
  print, Kernel
  SummedImage = CONVOL(FLOAT(WholeBand), Kernel, /CENTER, /EDGE_ZERO)

  TopFraction = SummedImage - (FLOAT(9) * GlobMean)

  SquareRootAnswer = SQRT((FLOAT(9) * (GlobNumber - 9))/(GlobNumber -
1))
```

```
  BottomFraction = GlobVariance * SquareRootAnswer

  Getis = FLOAT(TopFraction) / BottomFraction

  ENVI_ENTER_DATA, Getis

  print, "Program finished."
END
```

Subject: Re: Can this be done using array operations instead?
Posted by Jeremy Bailin on Thu, 02 Apr 2009 21:11:22 GMT
View Forum Message <> Reply to Message

On Apr 2, 3:43 pm, robintw <r.t.wil...@rmplc.co.uk> wrote:
> Thank you everyone for your help. I've now managed to write a function
> to do this with array functions. I've put the code below if anyone is
> interested. I realised that all I really needed was the sum of the
> values, and I could do all the rest by arithmetic operations on the
> array itself.
>
> However, the next part of this project will require using subarrays as
> I will need to get the Standard Deviation of each 3x3 square. Is there
> a way to do this?
>
> Cheers,
>
> Robin
>
> Code Here:
>
> ; Creates a Getis image given a FID, the dimensions of the file, a
> distance to use for the getis routine
> ; and a base window to send progress updates to
> PRO CREATE_GETIS_IMAGE, file, dims, distance, report_base
> ; TODO: Get this to loop through bands
> ; Get the data for the first band of the file (ignores pos from
> earlier)
> WholeBand = ENVI_GET_DATA(fid=file, dims=dims, pos=0)
>
> ; Calculate the dimensions of WholeBand
> SizeInfo = SIZE(WholeBand, /DIMENSIONS)
> NumRows = SizeInfo[0]
> NumCols = SizeInfo[1]
>
> ; Get the global mean
> GlobMean = MEAN(WholeBand)
>

```
>    ; Get the global variance
>    GlobVariance = VARIANCE(WholeBand)
>
>    ; Get the number of values in the whole image
>    GlobNumber = NumRows * NumCols
>
>    DimOfArray = (distance * 2) + 1
>    Kernel = FLTARR(DimOfArray, DimOfArray)
>    Kernel = Kernel + 1
>    print, Kernel
>    SummedImage = CONVOL(FLOAT(WholeBand), Kernel, /CENTER, /EDGE_ZERO)
>
>    TopFraction = SummedImage - (FLOAT(9) * GlobMean)
>
>    SquareRootAnswer = SQRT((FLOAT(9) * (GlobNumber - 9))/(GlobNumber -
>  1))
>    BottomFraction = GlobVariance * SquareRootAnswer
>
>    Getis = FLOAT(TopFraction) / BottomFraction
>
>    ENVI_ENTER_DATA, Getis
>
>    print, "Program finished."
>  END
```

You may find the built-in SMOOTH function useful... although it's billed as a smoothing filter, it's sometimes convenient to think of it as a way of calculating a sum over the neighbours of each pixel. For example, for a variance you can do something like:

```
smooth(image,3)^2 - smooth(image^2,3)
```

...with factors of 9 and/or 81 thrown in appropriate places that I haven't bothered to figure out. ;-)

-Jeremy.
