
Subject: Can these nested loops be made to run faster?

Posted by [humphreymurray](#) on Wed, 16 Aug 2006 09:39:36 GMT

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

Hi,

I am trying to optimize this code. Can anybody suggest how I can make it run faster? This code is already much more efficient than when I first wrote it. A description of what I am trying to do can be found on my previous post:

http://groups.google.com/group/comp.lang.idl-pvwave/browse_thread/thread/fa5c9a7dca215392/188ba05a3a095f6c#188ba05a3a095f6c

In short, training_data is a 2d array, which contains sample of pixel values, and training_classes contains an integer representing what classes these pixels belong to. testing_data is a 3d array, of which I am trying to classify. This is similar to my previous post, except I am now working with pixels in a 2d image.

Before you ask why I have nested loops, the other loop is processing each row of the image at a time, so that I don't run out of memory. If I do the operations of all at once, it crashes with a memory error. If the inner loop can be removed, or some operations removed from it, I believe this function will speed up immensely.

Cheers

Humphrey Murray
Student of Bachelor of Computing Honours
University of Tasmania, Australia

```
; Loop through for each row of the image
for y=0, y_size-1 do begin
```

```
    training_duplicates = REBIN(TRANPOSE(training_data), num_attributes,
    num_training_elements, x_size)
    testing_duplicates =
    rebin(transpose(testing_data[* , y, *]), num_attributes, num_training_elements, x_size)
    euclidean = sqrt(TOTAL((training_duplicates-testing_duplicates)^2, 1))
```

```
; calculate the distances for each training item
for x = long(0), x_size - 1 do begin
```

```
; Calculate the distances and sort the indexes of these
    sorted_indexes = sort(euclidean[* , x])
```

```

; Create an array that contains the classes of the items with the k
k_closest_classes = training_classes[sorted_indexes[0:k-1]]

; Calculate the Mode
distfreq = Histogram(k_closest_classes, MIN=Min(k_closest_classes)) ;
Calculate the distribution frequency
maxfreq = Max(distfreq) ; Find the maximum of the
frequency distribution.
mode = Where(distfreq EQ maxfreq, count) + Min(k_closest_classes) ;
Find the mode.

; Store the mode (classes with the highest frequency)
result[x,y] = mode[0]

endfor

endfor

```
