Subject: Re: Sorting a matrix
Posted by   on Fri, 01 Mar 2013 16:49:51 GMT
View Forum Message <> Reply to Message

Den fredagen den 1:e mars 2013 kl. 17:28:07 UTC+1 skrev Jeremy Bailin:
> On 3/1/13 7:45 AM, Mats Löfdahl wrote:
>
>> I'd like to sort a matrix M in such a way that the order of the rows is determined by the value in the first column. When the first column values are the same, the second column should be used, etc.
>
>>
>
>> Something like Craig Marqwardt's multisort (http://cow.physics.wisc.edu/~craigm/idl/down/multisort.pro), with M[0,*] used as key1, M[1,*] as key2, etc. But without actually having to specify the keys one by one. (Never mind which index counts as the column index :o)
>
>>
>
>> As the matrices I have in mind right now are integer arrays and do not have that many possible values (just -1,0,1), I thought about turning the matrix into a 1D string array with the length of each string equal to the number of columns and translating the column values into characters in the string, like A for -1, B for 0, C for 1, and then sorting the string array. But I'd prefer a more general program, in case there is one out there.
>
>>
>
>> Pointers, ideas?
>
>>
>
>
>
> I've done something like this before by generating a single unique
>
> index... something like this:
>
>
>
> matrix = [[4,5,6], [4,6,8], [2,3,4], [4,6,7]]
>
>
>
> matrixshape = size(matrix, /dimen)
>
> ; this gives you the range of each column:
>

```
>  matrixord = lonarr(matrixshape)
>
>  for i=0l,matrixshape[0]-1 do matrixord[i,*] = ord(matrix[i,*])
>
>  ordmax = max(matrixord, dimen=2)
>
>
>
>  ; what do you need to multiply by to get a unique range?
>
>  column_multiply = [reverse(product(reverse(ordmax[1:*]+1), /int,
>
>  /cumul)), 1]
>
>
>
>  ; create a unique key and sort on it
>
>  sortkey = total(matrixord * rebin(column_multiply,matrixshape, /sample),
>
>  /int, 1)
>
>  newmatrix = matrix[*, sort(sortkey)]
>
>
>
>  IDL> print, newmatrix
>
>        2      3      4
>
>        4      5      6
>
>        4      6      7
>
>        4      6      8
>
>
>
>  You'll need ORD(), which is part of JBIU which is currently inaccessible
>
>  because I'm in the process of moving domains. But here's a stub:
>
>
>
>  function ord, values
>
>
>
```

```
>  nvalues=n_elements(values)
>
>  sortvalues = sort(values)
>
>  uniqvalues = uniq(values[sortvalues])
>
>
>
>  nuniq = n_elements(uniqvalues)
>
>  ordlist = lindgen(nuniq)
>
>
>
>  ; this is basically the histogram(total(/cumulative)) trick
>
>  h = histogram(uniqvalues,bin=1,min=0,reverse=ri)
>
>  outp = lonarr(size(values, /dimen))
>
>  outp[sortvalues] = ordlist[ri[0:nvalues-1]-ri[0]]
>
>
>
>  return, outp
>
>
>
>  end
```

Oh, this looks pretty clever. I think I understand the idea but I'll have to digest the details. If I'm right, you find out for each column how many different values there are and then multiply the index for that column with the proper number to avoid overlap with the other column indices when adding.

Thanks!