Subject: Re: majority voting
Posted by JD Smith on Wed, 11 Feb 2009 23:20:59 GMT
View Forum Message <> Reply to Message

You're trying to compute the n-dimensional mode. For the 1D case, see:

http://www.dfanning.com/code_tips/mode.html

You could of course use SORT_ND and a 2D version of the "find the longest string of the same number" method on this page. This is very safe against sparse input distributions -- notice the HISTOGRAM in SORT_ND is of the sort indices, not the data themselves. It's also easy to recognize a mode length of 1 as "no mode", and (if you like) detect multiple modes in the data (aka "tie votes").

However, your problem is a special case, since (I presume) your vote labels are always low contiguous integers, you don't have to worry about sparseness. In this case, using HIST_ND directly without sorting will likely be much faster:

```
n=n_elements(array)
s=size(array,/DIMENSIONS)
 h=hist_nd([lindgen(1,n)/s[1],reform(transpose(array),1,n)],1 )
m=max(h,DIMENSION=2,mode) gt 1
mode=mode/s[0] * m + m - 1
```

All I'm doing here is forming a 2d space, with the first dimension the integer ID of the vote (aka column number in your example), and the second the actual votes cast.

Had you organized your array with the rows as your "vote items" you could save the transpose. Also note I specifically test for and set to -1 any mode with frequency of 1 (your "don't care"). I do not check for multiple modes, but you could do this as well in a straightforward way. The fact that the dimensions of the 2D histogram are the same as the input array is incidental, by the way.

JD