Subject: Re: majority voting
Posted by ben.bighair on Wed, 11 Feb 2009 18:13:20 GMT
View Forum Message <> Reply to Message

On Feb 11, 11:14 am, mort canty <m.ca...@fz-juelich.de> wrote:
> Hi all,
>
> Given a 2-D array such as
>
> 0 1 1 2 1
> 0 2 1 1 1
> 1 0 2 2 1
>
> where the entries are labels, the columns represent items and the rows
> are voters, I want a IDL function that returns the majority vote labels.
> So here I should get
>
> 0 ? 1 2 1
>
> as output, where ? = "don't care". There must _not_ be a loop over
> columns. I've got a clumsy solution, but I'm sure there's an elegant one
> somewhere?

Hi,

This is incomplete as it doesn't flag the "don't care" crowd. I can't noodle that part out without column looping. Looping would make it easy to use something like...

```
for i = 0, ncol-1 do dontCare[i] = ARRAY_EQUAL(votes[i,*],votes[i,0])
```

but by your rules, that is out of bounds.

```
***BEGIN
x=[[0,1,1,2,1],$
[0,2,1,1,1],$
[1,0,2,2,1]]
sz = SIZE(x, /DIM)
votes = [[TOTAL(x EQ 0, 2)],$
  [TOTAL(x EQ 1,  2)], $
  [TOTAL(x EQ 2, 2)]]
mx = MAX(votes, mxIdx,dim = 2)
majority = (array_indices(sz, mxIdx, /dim))[1,*]
print, majority
***END
```

Cheers,

Ben

Subject: Re: majority voting
Posted by Jean H. on Wed, 11 Feb 2009 19:52:47 GMT
View Forum Message <> Reply to Message

ben.bighair wrote:
> On Feb 11, 11:14 am, mort canty <m.ca...@fz-juelich.de> wrote:
>> Hi all,
>>
>> Given a 2-D array such as
>>
>> 0 1 1 2 1
>> 0 2 1 1 1
>> 1 0 2 2 1
>>
>> where the entries are labels, the columns represent items and the rows
>> are voters, I want a IDL function that returns the majority vote labels.
>> So here I should get
>>
>> 0 ? 1 2 1
>>
>> as output, where ? = "don't care". There must _not_ be a loop over
>> columns. I've got a clumsy solution, but I'm sure there's an elegant one
>> somewhere?
>
> Hi,
>
> This is incomplete as it doesn't flag the "don't care" crowd. I can't
> noodle that part out without column looping. Looping would make it
> easy to use something like...
>
> for i = 0, ncol-1 do dontCare[i] = ARRAY_EQUAL(votes[i,*],votes[i,0])
>
> but by your rules, that is out of bounds.
>
> ***BEGIN
> x=[[0,1,1,2,1],$
> [0,2,1,1,1],$
> [1,0,2,2,1]]
> sz = SIZE(x, /DIM)
> votes = [[TOTAL(x EQ 0, 2)],$
> [TOTAL(x EQ 1, 2)], $
> [TOTAL(x EQ 2, 2)]]
> mx = MAX(votes, mxIdx,dim = 2)
> majority = (array_indices(sz, mxIdx, /dim))[1,*]
> print, majority

> ***END
>
>
> Cheers,
> Ben
>

Ah ah, got it!

so, you get the votes, then:

sortID = sort_nd(votes,2)
mostVotes = votes[sortID]
mostvotes = mostVotes[*,2] ;only the highest votes
shiftVotes = shift(votes[sortID],0,1) ; shift by 1, so that the 2nd
highest vote is now on the last line of the array
SecondMostVotes = shiftVotes[*,2] ;only the highest votes
ToLabel = where(secondMostVotes - mostVotes eq 0) ;When is the 2nd most
common vote the same as the 1st one

majority[toLabel] = 999

Jean

Subject: Re: majority voting
Posted by Mort Canty on Wed, 11 Feb 2009 21:53:02 GMT
View Forum Message <> Reply to Message

ben.bighair schrieb:
> On Feb 11, 11:14 am, mort canty <m.ca...@fz-juelich.de> wrote:
>> Hi all,
>>
>> Given a 2-D array such as
>>
>> 0 1 1 2 1
>> 0 2 1 1 1
>> 1 0 2 2 1
>>
>> where the entries are labels, the columns represent items and the rows
>> are voters, I want a IDL function that returns the majority vote labels.
>> So here I should get
>>
>> 0 ? 1 2 1
>>
>> as output, where ? = "don't care". There must _not_ be a loop over
>> columns. I've got a clumsy solution, but I'm sure there's an elegant one
>> somewhere?

>
> Hi,
>
> This is incomplete as it doesn't flag the "don't care" crowd. I can't
> noodle that part out without column looping. Looping would make it
> easy to use something like...
>
> for i = 0, ncol-1 do dontCare[i] = ARRAY_EQUAL(votes[i,*],votes[i,0])
>
> but by your rules, that is out of bounds.
>
> ***BEGIN
> x=[[0,1,1,2,1],$
> [0,2,1,1,1],$
> [1,0,2,2,1]]
> sz = SIZE(x, /DIM)
> votes = [[TOTAL(x EQ 0, 2)],$
> [TOTAL(x EQ 1, 2)], $
> [TOTAL(x EQ 2, 2)]]
> mx = MAX(votes, mxIdx,dim = 2)
> majority = (array_indices(sz, mxIdx, /dim))[1,*]
> print, majority
> ***END
>
>
> Cheers,
> Ben
>

Thanks Ben. What I meant by "don't care" is that I don't care which of the labels that got equal votes is output. I think my solution is essentially the same as yours, certainly not more elegant:

```
function majority_vote, A, num_labels
   n = n_elements(A[*,0])
   B = intarr(n,num_labels)
   for i=0,num_labels-1 do begin
      C = A*0
      idx = where(A eq i,count)
      if count gt 0 then C[idx] = 1
      B[*,i] = total(C,2)
   endfor
   void = max(B,labels,dimension=2)
   return, labels/n
end
```

I was probably hoping for some HISTOGRAM magic :-)
Mort