Subject: Re: mode of a continuous distribution
Posted by Gray on Thu, 26 Aug 2010 13:15:56 GMT
View Forum Message <> Reply to Message

On Aug 24, 6:42 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> On Aug 24, 6:32 pm, Gray <grayliketheco...@gmail.com> wrote:
>
>> Hi all,
>
>> I have an array of data from a continuous distribution (non-Gaussian),
>> and I'd like to find the mode. I have two ideas:
>
>> 1) an iterative histogram method, where I find the max for smaller &
>> smaller binsizes until it converges;
>> 2) some sort of kernel density estimation method to estimate the
>> distribution, and then find the max from that.
>
>> Anyone implemented this sort of thing before? Any suggestions?
>
>> --Gray
>
> There is a kdf routine at
>
> http://www.faculty.iu-bremen.de/jvogt/cospar/cbw6/ComputerSe ssions/Ba...

Okay, folks, here's my routine for the mode of a continuous
distribution. As far as I can tell, it works (of course, it doesn't
help me solve my problem, which has to do with something else
entirely):

```
;+
; NAME:
;       KDF_MODE
; PURPOSE:
;       Find the mode of a sample from a continuous 1-d distribution
; EXPLANATION:
;       Finds the mode by finding the maximum of the probability
;       distribution, as found by a kernel density estimation with a
;       Gaussian kernel.
; CALLING EXAMPLE:
;       mode = kdf_mode(x)
; INPUTS:
;       x = array of sample points
; KEYWORDS:
;       None.
; OUTPUT:
;       Returns the mode of the distribution
; COMMON BLOCKS:
```

```
;       kdf_x (to pass the data array around)
; PROCEDURE:
;       This function uses MINF_BRACKET and MINF_PARABOLIC from the
;       NASA astronomy IDL library to maximize the KDF without using
PDEs.
; MODIFICATION HISTORY:
;       Written, Gray Kanarek 2010.
;-
FUNCTION kdf, u
  common kdf_x, xx
  nu = n_elements(u)
  case nu of
    0: message, 'U must be a scalar or array of target values'
    1: res = u*0
    else: begin
      s = size(u)
      res = make_array(s[1:s[0]],value=0.,type=size(type,/dim))
          end
  endcase
  lim = [-6,6.]
  for i=0,nu-1 do begin
    arg = u[i]-xx
    nonz = where(arg ge lim[0] and arg le lim[1],nz)
    if (nz eq 0) then continue
    res[i] = total(exp(-arg[nonz]^2/2.)/2.5066283)/n_elements(xx)
  endfor
  return, -res ;we're looking for the max, so return negative prob.
end

FUNCTION kdf_mode, x_array
  common kdf_x
  xx = x_array
  xa = min(xx) & xb = max(xx)
  minf_bracket, xa, xb, xc, func='kdf'
  minf_parabolic, xa, xb, xc, xm, fm, func='kdf'
  return, xm
end
```