Subject: Re: How to speed up kernel density smoothing for many data points
Posted by Moritz Fischer on Thu, 10 Oct 2013 13:48:41 GMT
View Forum Message <> Reply to Message

as allways: time vs memory. try below.
I guess one could also specialize matrix_eucl... for this very case,
where n=1 and k=2, especially removing the make_array part and the loop.

```
PRO test_kernel

N = 1000000
r = 2*randomn(seed,N)
v = 2*randomu(seed,N)
x = [[r],[v]]

; Just a smoothing parameter, unimportant...
hopt =  6.24/(N^(1./6.))*sqrt((stddev(x(*,0))^2+stddev(x(*,1))^2)/2. )

; Slow loop, where I need help
  f1 = fltarr(N)
  for i=0L,N-1 do begin
     mat = matrix_euclidean_distance( x[i,*], x) ; line by line...
     f1(i) = 1./float(N) * total(1./(hopt^2)*K(mat,hopt))
  endfor

END

FUNCTION K, t ,h
  aa  = where(t ge 1., n0, COMPLEMENT = bb, nCOMPLEMENT=nt)
  if n0 ne 0 then t(aa) = 0.
  if nt ne 0 then t(bb) = 4./!Pi*(1.-t(bb)^2)^3
  RETURN,t
END
```

Am 10.10.2013 15:29, schrieb jacobsvensmark@gmail.com:
> On Thursday, October 10, 2013 3:03:15 PM UTC+2, Moritz Fischer
> wrote:
>> ...and taking a look at K:
>>
>> check out the COMPLEMENT keyword to WHERE!
>>
>> Note that RES[aa] = 0. is redundant (it gets initialized with
>> zeros)!
>>
>> --m
>>
>>

>>
>>
>>
>> Am 10.10.2013 14:22, schrieb :
>>
>>> Okay so I have a long array of N 2D points:
>
> Hey,
>
> Thanks, I removed the RES[aa] = 0, good point. That will give some
> speed. And thanks for your help with the matrix_euclidean_distance
> program - I tested it out, and for N=1000 it runs instantly, but for
> N=10000, its very slow and becomes unresponsive, and for N=100000 it
> just spits out "% Array has too many elements". Makes sense because I
> guess your program effectively makes a NxN matrix from the N
> points...
>
