

Re: how to evaluate the addition of millions of functions dynamically and efficiently?

Source: <http://sci.tech-archive.net/Archive/sci.math/2004-08/2939.html>

From: networm (networm8848_at_yahoo.com)

Date: 08/16/04

Date: Sun, 15 Aug 2004 18:38:37 -0700

"jim green" <jjg@withheld.org.de> wrote in message
news:874qn4wmo3.fsf@turbot.fishnet...

> "networm" <networm8848@yahoo.com> writes:

>

>> *The shape-defining PSF can be radially symmetric, for a round shaped dot,
>> which is the first case you've mentioned, in need of a RBF monopole
method;*

>> *or it can be other strange shape, e.g. rectangular, which is the second
case*

>> *you've mentioned.*

>>

>> *For the first case, I did search google, did find 3590 hits, but did not
see*

>> *clearly how they are related to my worry: how to evaluate the value of
the*

>> *sum of these shifted functions(need to evaluate millions of time) fast;*

Can

>> *you give me some more clearer pointers?*

>>

>> *For the second case, though the rectangles are finitesupported, but they
can*

>> *be overlapped, how to smartly decide the overlapped neighbors which have*

>> *impact on the point I am evaluating?*

>

> *OK, since your functions have compact support and are on a regular
> grid things get rather easy without resorting to mutipole methods.*

>

> *We'll assume that the distance between adjacent grid nodes is d , and
> that your functions have support in the square of side length x .*

>

> *Now, suppose we are looking at the (i,j) -th grid point, then the only*

> *non-zero contributions are from neighbours with centres a distance*

> *at most $x/2$ away. Since the grid node separation is d , we take*

> *k to be the smallest integer such that $kd > x/2$, then the indicies of*

> *the neighbouring non-zero nodes are $(i-k, j-k), \dots, (i, j-k), \dots, (i+k, j-k)$,*

sci.math: Re: how to evaluate the addition of millions of functions dynamically and efficiently?

- > ... $(i-k, j+k), \dots, (i, j+k), \dots, (i+k, j+k)$. In other words you only need
- > to perform a **local** sum of the $(2k)^2$ nodes in the square centred on
- > the (i, j) node. Now do the same for every node.
- >
- > If N is the number of grid nodes then this trick reduces the complexity
- > of your evaluation from $O(N^2)$ to $O(N(2k)^2) = O(N)$, which should be
- > feasible for $N \sim 10^6$.
- >
- > This can be easily adapted to the case when the is grid rectangular,
- > rather than square (or if your functions has support in a rectangle
- > rather than a square) -- just treat the x - and y -coordinates
- > seperately. For functions with a circular support, just take the a
- > square containing the circle.
- >
- > Hope this helps!
- >
- > -j

Hi Jim,

Thank you very much for your help. Yeah, my first version program is completed, using Matlab,

for adding $66 \times 60 = 3960$ such shifted functions, my program needs 51 seconds to finish.

This is really embarrassing, since the program is supposed to work on sum of millions of such functions in near-realtime.

Anyway, now I want to adopt your method in this post, do "local neighbor summation"... this is attractive to me!

However, after digesting your algorithm, I do have questions:

Is your algorithm supposed only to work only for regularly laid-out functions? I mean for the support of functions, I can make them truncated into rectangular-shaped finite support, even for infinite support function. These functions are nicely behaved. So that's ok. But for the layout of the center of these functions, for this function array, your dynamic local summation only work for horizontally and vertically alligned function-center array layouts?

How to decide if a point is covered by which overlapped neighboring functions, for those irregular layout? I guess there should be some fast algorithm and fast data structure for handling this in the literature?

Thank you very much for your help!

-N