

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

Source: <http://sci.tech-archive.net/Archive/sci.math/2004-08/2491.html>

From: networm (networm8848_at_yahoo.com)

Date: 08/13/04

Date: Fri, 13 Aug 2004 00:17:39 -0700

Dear all,

I am headache on the following problem:

I have a bunch of 2D functions, they are the replication of the same 2D function at different shift (x, y) locations.

They can be viewed as a 2D impulse function replicated at both X and Y dimension, and then convolve with another 2D function. Since impulse function convolve with another function leads to that very function. So now after convolution I have a bunch of 2D functions...

These 2D functions can overlap, if the 2D functions have infinite support, for example, a $Z(x, y) = (\text{sinc}(x, y))^2$, this function has infinite support. Suppose there are 1000x1000 such infinite support 2D functions, shifted at different (x,y) locations...

Now I want to compute their sum.

(In Matlab), the simplest way to do this is to write out the complete expression and then evaluate it at any point (x, y)...

For example, my

$$Z_SUM(x, y) = Z(x, y) + Z(x-x1, y-y1) + Z(x-x2, y-y2) + Z(x-x3, y-y3) + \dots + Z(x-x1000000, y-y1000000)$$

I guess no computing system can handle this expression...

I guess the only way is to go point-oriented:

Given a point (x, y), asking for how much it should be for $Z_SUM(x, y)$, then for that particular point, I do 1000000-1 evaluations and additions to get that one point $Z_SUM(x, y)$ value...

This is still troublesome.

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

Now I approximate the infinite support of each function Z by a finite support — chop the decaying tail of the sinc^2 slope at a certain distance from the center of the function.

This leads to error, but saves time. (storage is not a problem, since I evaluate one point by one point by querying its value...)

However, this leads to a problem I want to ask for help:

How to decide if a point is within a support of a function?

What I meant is that if a point can be decided to only relate to a few functions, then I can evaluate and add those a few functions only.

But how to decide which some functions are relevant to the point?

For rectangular finite support of function Z s, this is a little easier.

But how about for circular-shaped finite support? Doing detecting a point to see if the point is with the circular support needs to take $(x-x_m)^2+(y-y_m)^2$ and then compare with some threshold, doing this 1000000 square and comparison isn't quite faster than doing the 1000000 $(\text{sinc}(x-x_m, y-y_m))^2$ evaluation then add directly...

Any depending on sampling density, the above query for one point's value needs to be millions of times...

Please advise me on how to do this efficiently?

Are there any similar problems and solutions existing in the field?

Thank you very much and I really appreciate your help!

-Net