CS 542G: Barnes-Hut

Robert Bridson

November 5, 2008

1 The Gravitational Potential

While it's perfectly possible to derive all the following at the level of forces, it can be both convenient and useful to re-express gravitational forces in terms of a potential energy and then work with the potential.

A bit of differentiation shows that

$$\begin{aligned}\nabla\frac{1}{\|x\|} &= \nabla(x\cdot x)^{-\frac{1}{2}} \\ &= -\frac{1}{2}(x\cdot x)^{-\frac{3}{2}}\nabla(x\cdot x) \\ &= -\frac{1}{2}(x\cdot x)^{-\frac{3}{2}}(2x) \\ &= -\frac{x}{\|x\|^3}\end{aligned}$$

(You can always express this in full coordinate form if you are uncomfortable differentiating $x\cdot x$ with respect to the vector x.)

Therefore, the gravitational force on point i due to j is:

$$f_{ij} = -Gm_im_j\frac{x_j-x_i}{\|x_j-x_i\|^3} = -\nabla\phi_{ij}(x_i)$$

where

$$\phi_{ij}(x) = \frac{-Gm_im_j}{\|x-x_j\|}$$

This describes the force as the negative gradient of a potential energy, making it clear it is a conservative physical system. From this we get the potential energy for the entire n-body problem:

$$\Phi(x) = \sum_{i=1}^{n}\sum_{j>i}\frac{-Gm_im_j}{\|x_i-x_j\|}$$

where here the argument x for $\Phi(x)$ is a $3n$-vector containing the locations of all n mass points. The negative gradient of $\Phi(x)$ gives the total force on each point.

There are two reasons to look at the potential energy instead of the forces directly: it's a scalar quantity that's easier to work with then vector forces, and it makes it simpler to make sure any approximations we make maintain conservation of energy. The idea will be to figure out to approximate $\Phi(x)$, and then take the gradient of that approximation as the approximate forces on the mass points; they won't be the exact gravity forces, but they will be conservative.

Evaluating the derivative of the potential is an interesting matter in its own right, which we probably should have broached even before when talking about optimization (given an objective function f, how do we get ∇f and the Hessian?). There is of course the option of relying on the user to do the derivative by hand and code it up, but this can be pretty arduous in some cases, or the "user" may no longer be around— say you have a gigantic legacy codebase to calculate a complicated objective function, say the drag coefficient of an airfoil, which now must be plugged into a new optimization routine. In some cases, the derivatives themselves can be approximated, noting $f'(x) \approx (f(x+\epsilon) - f(x))/\epsilon$ for example, though this carries many dangers of inaccuracy and in higher dimensions serious inefficiency—but we'll leave that idea for later. Finally, there is a whole topic called **automatic differentiation**: programs which take the source code of a function, parse it, and then use all the usual rules (chain rule, product rule, known derivatives of elementary functions, etc.) to generate new code which evaluates the derivative of the function. This is no walk in the park, but particularly for simpler languages such as FORTRAN 77 there are some great programs out there such as ADIFOR.

2 Error Estimates for the Potential

As mentioned last time for forces our basic approach to speeding up evaluation of the potential, an $O(n^2)$ sum, is to replace a distant cluster of point masses with a single point—one at the centre of mass with the total mass of the cluster. Let's look at the error incurred by this replacement.

We'll identify a point i and a cluster C of other points, where presumably i is much further from the points in C than they are from each other. The exact contribution of this pair (i and C) to the total potential is:

$$-Gm_i \sum_{j \in C} \frac{m_j}{\|x_i - x_j\|}$$

Let the total mass of the cluster be M:

$$M = \sum_{j \in C} m_j$$

and the centre of mass be x_C:

$$x_C = \frac{1}{M}\sum_{j\in C} m_j x_j$$

We will approximate the contribution to the potential as:

$$-Gm_i \frac{M}{\|x_i - x_C\|}$$

How different is this from the exact sum?

Let's start by reinterpreting the approximation as the mathematically equivalent sum:

$$-Gm_i \sum_{j\in C} \frac{m_j}{\|x_i - x_C\|}$$

Obviously we're not going to compute it this way, but it makes it clear that our approximation is equivalent to perturbing the position of each point $j \in C$ from x_j to x_C. So let's look at the error involved in that.

Ignoring the $-Gm_i m_j$ constants, this boils down to estimating the difference

$$\frac{1}{\|x_i - x_j\|} - \frac{1}{\|x_i - x_C\|}$$

Let's label

$$\begin{aligned} x_D &= x_C - x_i \\ \Delta x &= x_j - x_C \end{aligned}$$

so the error expression is equivalent to

$$\frac{1}{\|x_D + \Delta x\|} - \frac{1}{\|x_D\|}$$

Therefore we need to know how much $1/\|x\|$ varies from the value at $x = x_D$ when perturbed by some amount Δx. The obvious tool, as always, is Taylor series!

We already know $\nabla 1/\|x\| = -x/\|x\|^3$ from above. The Hessian matrix, using vector versions of product rule etc. is:

$$\begin{aligned} \nabla\nabla\frac{1}{\|x\|} &= \nabla\left(-\frac{x}{\|x\|^3}\right) \\ &= \nabla\left(-x(x\cdot x)^{-3/2}\right) \\ &= -I(x\cdot x)^{-3/2} - x\left(\frac{-3}{2}\right)(x\cdot x)^{-5/2}(2x) \\ &= -\frac{1}{\|x\|^3}I + \frac{3}{\|x\|^5}x\otimes x \end{aligned}$$

The first term is the 3×3 identity matrix I scaled by $-1/\|x\|^3$, and the second term involves the outer product $x \otimes x$ of x with itself, a 3×3 matrix we sometimes have written as xx^T.

Evaluating these at $x = x_D$, the Taylor series tells us

$$\frac{1}{\|x_D + \Delta x\|} = \frac{1}{\|x_D\|} - \frac{x_D^T}{\|x_D\|^3}\Delta x + \frac{1}{2}\Delta x^T \left[-\frac{1}{\|x_D\|^3} I + \frac{3}{\|x_D\|^5} x_D \otimes x_D\right] \Delta x + O(\Delta x^3)$$

This can be simplified to:

$$\frac{1}{\|x_D + \Delta x\|} = \frac{1}{\|x_D\|} - \frac{x_D \cdot \Delta x}{\|x_D\|^3} - \frac{\|\Delta x\|^2}{2\|x_D\|^3} + \frac{3(x_D \cdot \Delta x)^2}{2\|x_D\|^5} + O(\Delta x^3)$$

Therefore, the absolute error induced by changing x_j to x_C in the potential computation is on the order of

$$\frac{G m_i m_j \|\Delta x\|}{\|x_D\|^2}$$

and the relative error is on the order of

$$\frac{\|\Delta x\|}{\|x_D\|}$$

This gives us a clear route to controlling the error made in the clustering approximation: make sure that the ratio of the radius r of the cluster (the maximum $\|\Delta x\|$) to the distance D to the centre of the cluster $\|x_D\|$ is smaller than a given tolerance.

It turns out life is actually better than this. The fact that x_C is the centre of mass, and not just any old point inside the cluster, implies some extra cancellation in the error when the approximation over the entire cluster is considered. Observe that:

$$\begin{aligned}
\sum_{j \in C} \frac{m_j}{\|x_i - x_j\|} &= \sum_{j \in C} m_j \left(\frac{1}{\|x_C - x_i\|} - \frac{(x_C - x_i) \cdot (x_j - x_C)}{\|x_C - x_i\|^3}\right) + O\left(\frac{\|\Delta x\|^2}{\|x_D\|^3}\right) \\
&= \frac{M}{\|x_C - x_i\|} - \frac{(x_C - x_i)}{\|x_C - x_i\|^3} \cdot \sum_{j \in C} m_j (x_j - x_C) + O\left(\frac{\|\Delta x\|^2}{\|x_D\|^3}\right) \\
&= \frac{M}{\|x_C - x_i\|} - 0 + O\left(\frac{\|\Delta x\|^2}{\|x_D\|^3}\right)
\end{aligned}$$

The middle term vanishes, since x_C as the centre of mass implies $\sum_{j \in C} m_j (x_j - x_C) = 0$. Therefore the error in the clustering is actually quadratic in the ratio r/D of cluster radius to cluster distance, meaning we can be more aggressive in clustering.

I've been a little sloppy in terms of leaving Taylor series errors in $O()$ notation, but some harder work can determine a rigourous bound on the error made; similar analysis also can be carried out at the force level, with similar results.

3 The Barnes-Hut Tree Algorithm

The central idea of the **Barnes-Hut** algorithm for efficiently approximating the forces in an n-body problem is this: replace distant clusters with their centres of mass, when the ratio r/D is small enough to keep the relative error below a given tolerance. The tricky part is determining those clusters: our input data is just a scattered set of points.

This is where trees enter. As a prelude to computing forces, a tree is constructed around the points, each node identified as a bounded region of space containing a subset of the points. The root contains all the points, and the children of a node split the node's points into smaller subsets in more tightly bounded regions. Each node is a potential cluster—and if a node is too large, its children can be considered instead.

One common tree example is the **octree**: a cube containing all the points is the root of the tree; each non-leaf node of the tree is split into eight children (in a $2 \times 2 \times 2$ arrangement, halving the length along each axis); the leaves are nodes containing one or zero points. Another popular alternative is the kd-tree, where instead of splitting a node eight ways, each node is just split in half along the longest axis. Either tree can be made more efficient if every node is shrunk to only just contain the points inside.

The total mass and centre of mass of each node (or rather, of all the points contained within a node of the tree) can be computed efficiently from the leaves up: a node's total mass is the sum of its children's masses, and its centre of mass is the centre of mass of its children's centres. Added to this data we include at each node a bounding radius r, an upper bound on the maximum distance of any of its points to the node's centre of mass.

Then, when it comes time to evaluate the potential or force on a particular i, the tree is traversed from the root down. If the ratio r/D of the node's bounding radius to the distance from x_i to the node's centre of mass is small enough to satisfy an error tolerance, we use the approximation; otherwise, we recursively examine the children of the node. Obviously if we hit a leaf, we can use the exact formula (or ignore it if the leaf happens to be the point i itself).

With a little more effort we can do even a bit better than this: instead of just considering a single point versus a cluster, we can compare clusters to clusters, simultaneously approximating all pairs of forces between the points inside the clusters with a single calculation between their centres of mass. The error analysis is only a little more complicated.

Constructing the tree takes $O(n \log n)$ time and $O(n)$ space; with a constant that depends on the distribution of the points and the error tolerance, the time to compute each force is expected to take $O(\log n)$ time. Thus Barnes-Hut is an $O(n \log n)$ algorithm, a huge improvement over the naïve $O(n^2)$

exact approach.

There's a lot more to say about Barnes-Hut and related methods, that I'll leave it to you to explore on your own if interested. For example, instead of controlling the error only by way of controlling the size of clusters (according to the second order error term in the Taylor series), we can also include more terms in the Taylor series (bringing the error up to a higher power of the r/D ratio). This is the basis of the **Fast Multipole Method**, which in principle can evaluate the potential or forces to full floating-point accuracy still in $O(n \log n)$ time. Another generalization is to problems other than gravity—electrostatics, for example, uses the same $1/\|x\|$ potential function but with charge instead of mass, and many other problems (such as RBF interpolation) use different radial functions that can nevertheless still be approximated with Taylor series in much the same way.

4 Introducing Partial Differential Equations

The potential function $1/\|x\|$ in 3D happens to satsify a very special partial differential equation. We saw above the gradient $\nabla 1/\|x\|$ is $-x/\|x\|^3$. If we then further take the divergence of that gradient, or equivalently the trace of the Hessian which we also worked out above, we get:

$$\begin{aligned}
\operatorname{tr}\left(-\frac{1}{\|x\|^3}I + \frac{3}{\|x\|^5}x \otimes x\right) &= \frac{-\operatorname{tr}(I)}{\|x\|^3} + \frac{3}{\|x\|^5}\operatorname{tr}(x \otimes x) \\
&= \frac{-3}{\|x\|^3} + \frac{3}{\|x\|^5}\|x\|^2 \\
&= \frac{-3}{\|x\|^3} + \frac{3}{\|x\|^3} \\
&= 0 \quad \text{when } x \neq 0
\end{aligned}$$

That is, the Laplacian of the potential is zero except at $x = 0$ where it is undefined:

$$\nabla \cdot \nabla \frac{1}{\|x\|} = 0 \quad \text{when } x \neq 0$$

This leads to the observation that calculating the gravitational potential is equivalent to solving a particular partial differential equation, which we will pursue next time.