CS 542G: Finishing Radial Basis Functions, Solving Linear Systems

Robert Bridson

September 17, 2008

1 Finishing up with Radial Basis Functions

1.1 The Basis Function Viewpoint

Recall from last time the general form of an RBF interpolant:

$$f(x) = \sum_{i=1}^{n} \lambda_i \phi(x - x_i) + P(x)$$

Here $P(x)$ is some low order polynomial (e.g. linear for thin-plate splines). While we derived the thin-plate spline version from a notion of finding an optimally smooth interpolant, we can interpret it in another way: it's just a linear combination of a set of basis functions $\phi_i(x)$. If the polynomial term is just linear, these would be:

$$\begin{aligned}
\phi_1(x) &= \phi(x - x_1) \\
\phi_2(x) &= \phi(x - x_2) \\
&\vdots \\
\phi_n(x) &= \phi(x - x_n) \\
\phi_{n+1}(x) &= 1 \\
\phi_{n+2}(x) &= x^{(1)} \\
&\vdots \\
\phi_{n+k+1}(x) &= x^{(k)}
\end{aligned}$$

The interpolation problem becomes a matter of picking out a function from the space spanned by these basis functions, one that interpolates the data and perhaps satisfies a few extra constraints.

This is a common viewpoint in modern scientific computing: solve a big problem (e.g. finding a function which interpolates data) by reducing it to a small problem via restricting it to a carefully chosen finite dimensional space (e.g. the space spanned by these basis functions). Once the basis has been picked, the rest of the computation is automatic—or at least, reduced to a much simpler "kernel" problem of solving a linear system to figure out the coefficients of the linear combination. Compare this to the "operational" viewpoint, where we take specific data points and a new position x, and try to come up with an algorithm for estimating the value at x.

1.2 The Linear System

Also recall from last time the system of linear equations we derived for the RBF interpolant, which if we are using a linear polynomial term correspond to:

$$\begin{aligned}
f(x_1) = \sum_{i=1}^{n} \lambda_i \phi(x_1 - x_i) + \lambda_{n+1} + \lambda_{n+2} x_1^{(1)} + \cdots + \lambda_{n+k+1} x_1^{(k)} &= f_1 \\
&\vdots \\
f(x_n) = \sum_{i=1}^{n} \lambda_i \phi(x_n - x_i) + \lambda_{n+1} + \lambda_{n+2} x_n^{(1)} + \cdots + \lambda_{n+k+1} x_n^{(k)} &= f_n \\
\sum_{i=1}^{n} \lambda_i &= 0 \\
\sum_{i=1}^{n} \lambda_i x_i^{(1)} &= 0 \\
&\vdots \\
\sum_{i=1}^{n} \lambda_i x_i^{(k)} &= 0
\end{aligned}$$

The first n require $f(x)$ to interpolate the data points, and the last $k+1$ provide extra conditions to close the system. In the thin-plate spline case, the last equations are associated with keeping the function well-behaved out at infinity, but more generally they can also be interpreted or derived from a notion of having the polynomial term $P(x)$ match the data as well as possible. We won't dive into that in this course, but before long you will have the tools to work this out for yourself; for the time being note that if the data points actually did come from a low order polynomial (for example, if all the f_i's were equal to some constant) then we would expect the interpolant to exactly match that polynomial and not involve the radial functions at all, which is what these equations enforce.

So far we've waved our hands, saying we have the same number of unknown coefficients as equa-

tions, and therefore we can hope to have a solution. However, you should remember from linear algebra that this isn't necessarily the case: for example, the linear system $x + y = 0$, $x + y = 1$ (two equations in two unknowns) has no solution. Even if the RBF system does have a solution, we can ask if this is a **well-posed problem**: if the data points were to change slightly, could our interpolant change a lot? If the linear system were perturbed a little by round-off error, would the solution be perturbed a lot?

We'll not go through technical details here on whether or not the RBF interpolation question is well-posed, though your first assignment will investigate this question. However, we will take a deeper look at linear systems: how do you tell if a linear system of equations is well-posed? How do you solve it if it is? What can you do if it isn't?

2 Linear Algebra Infrastructure

Before we can tackle the well-posedness of linear systems, we need some concepts to talk sensibly about it. The first is to write the linear system with matrix-vector notation. The linear system above becomes:

$$\begin{pmatrix} \phi(x_1 - x_1) & \phi(x_1 - x_2) & \cdots & \phi(x_1 - x_n) & 1 & x_1^{(1)} & \cdots & x_1^{(k)} \\ \phi(x_2 - x_1) & \phi(x_2 - x_2) & \cdots & \phi(x_2 - x_n) & 1 & x_2^{(1)} & \cdots & x_2^{(k)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi(x_n - x_1) & \phi(x_n - x_2) & \cdots & \phi(x_n - x_n) & 1 & x_n^{(1)} & \cdots & x_n^{(k)} \\ 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(k)} & x_2^{(k)} & \cdots & x_n^{(k)} & 0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ \lambda_{n+1} \\ \lambda_{n+2} \\ \vdots \\ \lambda_{n+k+1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Letting the matrix be A, the vector of unknown λ coefficients be x (note: this is an abuse of notation, since we're using x for coordinates as well, but it's more consistent with how people usually discuss linear systems), and the known vector on the right-hand-side be b, the linear system can be written as:

$$Ax = b$$

2.1 Vectors

The way we wrote the linear system, $Ax = b$, introduces one convention: vectors are by default thought of as **column vectors**, i.e. vectors in $\mathbb{R}^m$ can be interpreted as matrices of dimension $m \times 1$. The alternative is **row vectors**, i.e. matrices of dimensions $1 \times m$, which is far less common and is generally explicitly noted if needed.

With this convention, the dot product or inner product of two vectors—the sum of the pairwise products of elements—can be written with a transpose and matrix multiplication:

$$x \cdot y = x^T y$$

(Remember the transpose flips a matrix over, in this case taking the column vector x and producing the row vector version x^T.) Another useful construct is the **outer product**, which is what you get if you switch the transpose:

$$x \otimes y = xy^T$$

Note that the outer product is actually a matrix, and is well defined even if x and y have different lengths: if x is m-dimensional and y is n-dimensional, $x \otimes y$ is an $m \times n$ matrix with (i, j)'th entry equal to $x_i y_j$. You can get the dot product of two vectors from the outer product by taking the **trace**, the sum of the diagonal entries of a matrix:

$$x^T y = tr(xy^T)$$

A vector **norm** allows us to measure the magnitude of a vector. Recall from linear algebra that a norm $\|\cdot\|$ must satisfy the following properties:

- $\|x\| \geq 0$ for all vectors x, and $\|x\| = 0$ if and only if $x = 0$
- $\|\alpha x\| = |\alpha|\|x\|$ for any vector x and scalar α
- $\|x + y\| \leq \|x\| + \|y\|$ for any vectors x and y, sometimes known as the triangle inequality

There are several widely used norms. The most important is the **2-norm**, also called the Euclidean norm or l^2 norm, defined for an n-dimensional vector as:

$$\|x\|_2 = \sqrt{x_1^2 + \ldots + x_n^2}$$

Note the subscript 2 after the $\|$ indicating this is not just any norm, but the 2-norm. Equivalently this can be written as

$$\|x\|_2 = \sqrt{x \cdot x} = \sqrt{x^T x}$$

This is often the most convenient norm to deal with when looking at theoretical analysis of algorithms.

The **max norm**, also called sup norm, inf norm, ∞-norm, or l^∞ norm, is instead defined as the maximum magnitude of any entry in the vector:

$$\|x\|_\infty = \max_i |x_i|$$

When talking about errors, the max norm is the one people usually want to know about: it gives the worst case.

Finally, the **1-norm**, also called l^1 norm, occasionally is useful, defined as the sum of the absolute values of the entries of the vector:

$$\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n|$$

(As an aside, the l^1, l^2 and l^∞ monikers do actually reflect these are drawn from a big family of l^p norms where p can be any real number from 1 to (and including) ∞, with p essentially replacing the power 2 in the 2-norm. However, outside of $p = 1, 2, \infty$ these norms are rarely of any practical importance.)

You may remember from linear algebra that in finite dimensional spaces, any two norms are equivalent: if $\|\cdot\|_a$ and $\|\cdot\|_b$ are norms, there must be positive finite constants c and C that satisfy

$$c\|x\|_b \leq \|x\|_a \leq C\|x\|_b$$

for every vector x. Thus if one norm shows the error in a calculation is dropping to zero, it must be dropping to zero in every other norm at essentially the same rate (up to constant factors). It's not hard to derive the equivalence constants for comparing these three norms; it might make a fun exercise if you're bored one day.

Finally, with a vector norm in hand, we can define the notions of absolute error and relative error from before: if x is a computed vector and $\hat{x}$ is the exact vector, the absolute error is

$$\|\hat{x} - x\|$$

and the relative error, assuming $\hat{x} \neq 0$, is

$$\frac{\|\hat{x} - x\|}{\|\hat{x}\|}$$

Usually the relative error is the more important quantity, as before.

2.2 Matrices

We'll focus for now just on bringing the notion of norm to matrices. There are two common ways to do this, both tied to vector norms.

The first is to simply treat a matrix as a big vector that happens to be oddly laid out in a grid, and then use any vector norm. Indeed, the set of all $m \times n$ matrices is an mn-dimensional vector space

equivalent to $\mathbb{R}^{mn}$. The only really important example of this class of matrix norms is the **Frobenius norm**, which is what you get with the 2-norm applied to the matrix-as-a-big-vector:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$$

This is equivalently defined as

$$\|A\|_F = \sqrt{tr(A^T A)} = \sqrt{tr(AA^T)}$$

where tr again indicates the **trace** of a matrix, the sum of its diagonal entries. You might also remember from linear algebra that the trace happens to be the sum of the eigenvalues.

The other main class of matrix norms are the **induced norms**, derived from a related vector norm in the following way:

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

or equivalently:

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

As is apparent in the first form, this is how much multiplication by A can possibly amplify the size of a vector, measured in some chosen norm: how big $\|Ax\|$ can get relative to $\|x\|$. The three important examples are, of course, the matrix 2-norm $\|A\|_2$, the matrix max norm $\|A\|_\infty$, and the matrix 1-norm $\|A\|_1$. We won't derive this here, but these matrix norms can be expressed somewhat more explicitly. The matrix 2-norm $\|A\|_2$ is the maximum eigenvalue of $A^T A$ (or equivalently AA^T), which should set off some alarm bells since eigenvalues aren't the easiest thing in the world to compute. (Also you might think of how this connects to the Frobenius norm, which is the sum of all eigenvalues of $A^T A$.) The matrix max norm and 1-norm are intimately related by these formulas:

$$\begin{aligned} \|A\|_\infty &= \max_i \sum_{j=1}^{n} |a_{ij}| \\ \|A\|_1 &= \max_j \sum_{i=1}^{m} |a_{ij}| \end{aligned}$$

The matrix max norm is the maximum 1-norm of any row, and the matrix 1-norm is the maximum 1-norm of any column; $\|A\|_\infty = \|A^T\|_1$.

The chief great feature of the induced matrix norms is the bound they give on matrix-vector multiplication:

$$\|Ax\| \leq \|A\|\|x\|$$

for any vector x (assuming the norm of the matrix is the induced norm from the norm of the vectors).

Another very useful property of all four matrix norms introduced here (including the Frobenius norm, though not necessarily other matrix norms I haven't mentioned) is how they interact with matrix multiplication: $\|AB\| \leq \|A\|\|B\|$ for any two matrices A and B which can be multiplied together.

3 Well-Posed Linear Systems

Recall the definition of a well-posed problem: it has a solution, it's unique, and it doesn't change too much if the data are perturbed slightly. You should recall from linear algebra that a linear system $Ax = b$ has a unique solution if and only if A is nonsingular (equivalently: invertible, has nonzero determinant, has no zero eigenvalues). However, you probably didn't check on the last condition: how much can the solution x change if the data are perturbed?

For the purposes of this course, we'll only look at perturbing b, not A as well, though that would be a better model of what actually happens when A is stored and operated on in floating point arithmetic. Including perturbations in A make the analysis a lot harder, but don't appreciably change the results.

Assuming A is nonsingular, then the exact answer to the linear system is $x = A^{-1}b$. If we perturb the right hand side b to $b + \Delta b$, we get the following linear system instead, written with the perturbed solution $x + \Delta x$:

$$A(x + \Delta x) = b + \Delta b$$

Subtracting off $Ax = b$ gives $A\Delta x = \Delta b$, which implies $\Delta x = A^{-1}\Delta b$.

Using an induced matrix norm, this gives us a nice bound on the absolute error:

$$\begin{aligned} \|\Delta x\| &= \|A^{-1}\Delta b\| \\ &\leq \|A^{-1}\|\|\Delta b\| \end{aligned}$$

In other words, as long as $\|A^{-1}\|$ isn't too large, we can be confident a small change in b won't result in a big change in the solution.

However, as before absolute error isn't always very meaningful. The linear system $Ax = b$ has exactly the same solution as the rescaled $(10^{-6}A)x = 10^{-6}b$, and in some fundamental sense really should be treated as equivalent, yet the absolute error bound is a million times worse: $\|(10^{-6}A)^{-1}\| = 10^6\|A^{-1}\|$.

Turning to relative error instead, we first also note that $Ax = b$ implies

$$\begin{aligned} \|b\| &= \|Ax\| \\ &\leq \|A\|\|x\| \\ \Rightarrow \quad \|x\| &\geq \frac{\|b\|}{\|A\|} \end{aligned}$$

We can now combine the absolute error estimate with this inequality to bound the relative error:

$$\begin{aligned} \frac{\|\Delta x\|}{\|x\|} &\leq \frac{\|A^{-1}\|\|\Delta b\|}{\|b\|/\|A\|} \\ &= \left(\|A\|\|A^{-1}\|\right) \frac{\|\Delta b\|}{\|b\|} \end{aligned}$$

This bound is now invariant under rescaling the linear system, and thus more meaningfully captures the idea of how well-posed a linear system is. The constant $\|A\|\|A^{-1}\|$ has a special name, the condition number of A denoted as $\kappa(A)$. Taking a limit, if A is singular then we define $||A^{-1}|| = \infty$ and $\kappa(A) = \infty$, so the condition number fully captures the notion of whether or not a linear system is well-posed. A linear system is well-posed if the condition number is not too large; we also say the matrix is **well-conditioned**. A nonsingular matrix with a big condition number is said to be **ill-conditioned**.

Note that the condition number depends critically on which induced norm is used, though thanks to the equivalence of norms it doesn't usually matter exactly which is taken. If needed, a subscript can make it clear: $\kappa_2(A)$, $\kappa_\infty(A)$ or $\kappa_1(A)$.

The notion of condition number, which essentially quantifies how much a perturbation in the data for a problem can be amplified into a perturbation in the solution, is more general and can be defined for many problems. However, in this course we'll always take it to refer to this specific case of a linear system.