2021-04-01

1 Introduction

Last time, we discussed how to seek low-dimensional structure in the *input* space for functions $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ for large n. In this lecture, we consider how to find low-dimensional structure in the *output* space for $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}^m$ where m is large and n is modest.

Of course, it is also possible to combine the techniques to deal with high-dimensional input *and* output spaces, and this is frequently done in practice.

2 Interpolation and vector-valued functions

If n is not too large and f is smooth, a natural approach to approximating f is interpolation. That is, let $\mu_1, \ldots, \mu_k \in \mathbb{R}^n$ be sample points and let $L_1(\mu), \ldots, L_k(\mu)$ be the Lagrange functions (or cardinal functions) forming a k-dimensional basis for an approximation space $\mathcal{V}$ (with $\dim(\mathcal{V}) = k$).

The Lagrange functions, we might recall, have the property that

$$L_i(\mu_j) = \delta_{ij};$$

we remind ourselves of this now, as it will be important again later. If we start off with an arbitrary basis $v_1, \ldots, v_k$ for $\mathcal{V}$, we can compute the Lagrange functions as

$$L_j(\mu) = \begin{bmatrix} v_1(\mu) & \ldots & v_k(\mu) \end{bmatrix} \begin{bmatrix} v_1(\mu_1) & v_2(\mu_1) & \ldots & v_k(\mu_1) \\ v_1(\mu_2) & v_2(\mu_2) & \ldots & v_k(\mu_2) \\ \vdots & \vdots & \ddots & \vdots \\ v_1(\mu_k) & v_2(\mu_k) & \ldots & v_k(\mu_k) \end{bmatrix}^{-1} e_j,$$

where e_j is the jth column of the identity. Then we can compute our approximation as

$$f(\mu) \approx \hat{f}(\mu) = \sum_{j=1}^{k} f(\mu_j) L_j(\mu).$$

To bound the error in such approximations, we use the same techniques described last time. The error is within a factor of $\max_x \sum_j |L_j(\mu)|$ of the

best possible in the space, and we can bound the latter in terms of Taylor expansions (for example).

The approximations $L_j(\mu)$ depend on the space $\mathcal{V}$ and the points $\{\mu_j\}_{j=1}^k$. It does not depend on the particular basis we choose for $\mathcal{V}$. However, it *does* depend on the space $\mathcal{V}$! With the same set of points, we can get many different interpolants associated with different approximation spaces (polynomial spaces, spline spaces, piecewise linear spaces, or others). Different interpolation schemes are appropriate under different regularity assumptions: high-degree polynomial interpolation might make sense for interpolating very smooth f, but not for a function that is nondifferentiable because of an absolute value function (for example).

Independent of the exact space $\mathcal{V}$, any method based on interpolation through points $\{\mu_1, \ldots, \mu_k\}$ will yield a result in the space $\mathrm{sp}[f(\mu_1), \ldots, f(\mu_k)]$. But we can extract approximations to $f(\mu)$ from this subspace in other ways than interpolating. For example, suppose $f(\mu)$ is defined implicitly by a set of equations, e.g.

$$A(\mu)f(\mu) = b;$$

then we can choose an approximation $f(\mu) = \sum_{j=1}^k c_j f(\mu_j)$ by solving a proxy system with $k \ll n$ unknowns, which we might be able to do much more cheaply than solving a new system from scratch. This suggests that if we have some way to evaluate the quality of a solution without a full evaluation (e.g. by looking at the residual norm for some defining equation), we might be able to use the subspace of approximations accessed by interpolation (which has at most dimension k), but with the coefficients determined by a reduced system of equations rather than by interpolation.

3 Learning from snapshots

If we are going to build a good approximating subspace from evaluation of f at some sample points (sometimes called *snapshots*), we need a good method for choosing those sample points. Some possible methods include:

- Choose k points on some type of regular grid.
- Choosing k points at random according to some distribution (e.g. uniform over the domain Ω).
- Choose k points according to a *low-discrepancy sequence*.

- Choose k points according to an *experimental design* method (e.g. using a Latin hypercube).

Sampling on regular grids must be done with some care to avoid aliasing issues when f has some periodicity to it. And random sampling tends to produce some "clumps" in our domain, and so we may get more evently spread points from a low-discrepancy sequence generator.

Frequently, when we look at a collection of snapshots, we will discover that they have some redundancy to them. In fact, we hope they do! The *POD* (or *principle orthogonal directions*) approach to reducing a snapshot collection basically involves an SVD:

$$\begin{bmatrix} f(\mu_1) & \dots & f(\mu_k) \end{bmatrix} = U \Sigma V^T.$$

If there is an r-dimensional subspace of the k-dimensional snapshot space that does a good job of capturing all the snapshots, we will have a small value for σ_{r+1}, and the truncated factor U_r provides an orthonormal basis for the relevant space.

Taking the SVD of a snapshot matrix can be seen as a discrete approximation of computing the eigendecomposition of

$$C = \int_\Omega f(\mu) f(\mu)^T \rho(\mu) \, d\mu$$

for which the dominant r eigenvectors span the "optimal" r-dimensional invariant subspace for approximating general $f(\mu)$ in the domain. In principle, we could try to approximate the difference between the integral and the finite sum (based on the smoothness of f), but we usually don't bother in practice. If we have enough samples to reasonably cover the parameter space, we typically assume a small value σ_{r+1} is good evidence that the whole image $f(\Omega)$ is well approximated by U_r.

4 Empirical interpolation method (EIM)

For constructing a low-dimensional space from which to draw approximation, an alternative to principle orthogonal decomposition is the *empirical interpolation method* (EIM). The EIM algorithm is a greedy method that simultaneously constructs a set of interpolation points (the so-called "magic

points") and a basis for a subspace spanned by associated samples of the function.

For this case, consider $f : \mathcal{X} \times \otimes \to \mathbb{R}$, where $\mathcal{X}$ is essentially an index set. For fixed $\mu \in \Omega$, we will write $f(x, \mu) = f_\mu(x)$. At each step, the EIM algorithm solves the problem

$$x_{j+1}, \mu_{j+1} = \operatorname{argmax} |f(x, \mu) - \hat{f}_j(x, \mu)|$$

where $\hat{f}_j(x, \mu)$ is the interpolant for f based on $\{(x_i, \mu_i)\}_{i=1}^j$. Along the way, one builds up a basis of functions for the nested interpolation spaces based on *scaled error functions*, i.e.

$$h_j(x, \mu) = \frac{f(x, \mu_{j+1}) - \hat{f}_j(x, \mu_{j+1})}{f(x_{j+1} j, \mu_{j+1}) - \hat{f}_j(x_{j+1}, \mu_{j+1})}.$$

Readers familiar with univariate interpolation may recognize this as similar to the construction of the Newton basis for polynomial interpolation. The algorithm terminates when the maximum error falls below some tolerance.

Those readers who are afficianados of numerical linear algebra may recognize another algorithm lurking in the shadows. To make this more explicit, note that the interpolant at step j is

$$\hat{f}_j(x, \mu) = \begin{bmatrix} h_1(x) & h_2(x) & \dots & h_j(x) \end{bmatrix} \begin{bmatrix} c_1(\mu) \\ c_2(\mu) \\ \vdots \\ c_j(\mu) \end{bmatrix},$$

where

$$\begin{bmatrix} 1 & & & & \\ h_1(x_2) & 1 & & & \\ h_1(x_3) & h_2(x_3) & 1 & & \\ \vdots & & & \ddots & \\ h_1(x_j) & h_2(x_j) & \dots & h_{j-1}(x_j) & 1 \end{bmatrix} \begin{bmatrix} c_1(\mu) \\ c_2(\mu) \\ \vdots \\ c_j(\mu) \end{bmatrix} = \begin{bmatrix} f(x_1, \mu) \\ f(x_2, \mu) \\ \vdots \\ f(x_j, \mu) \end{bmatrix},$$

where we are using the fact (which holds by construction) that $h_i(x_i) = 1$ and $h_i(x_\ell) = 0$ for $\ell < i$. We also observe that $c_i(\mu_j) = 0$ for $i > j$, again by construction.

If the sets $\mathcal{X}$ and Ω are discrete, what we have done is to write

$$PFQ \approx \begin{bmatrix} H_{11} \\ H_{21} \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} \end{bmatrix}$$

where H_{11} is unit lower triangular and C_{11} is upper triangular, and the permutation matrices P and Q reorder $\mathcal{X}$ so that $x_1, \ldots, x_j$ and $\mu_1, \ldots, \mu_j$ appear first in the ordering of variables. Then we seek the largest magnitude element of

$$PFQ - \begin{bmatrix} H_{11} \\ H_{21} \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & (PFQ)_{22} - H_{21}C_{12} \end{bmatrix}$$

in order to obtain x_{j+1} and μ_{j+1} and extend the factorization. This is exactly the procedure for *Gaussian elimination with complete pivoting*.

For reasons that I don't entirely understand, the interpretation of the EIM basis construction as Gaussian elimination with complete pivoting (GCP) doesn't seem to appear in the literature[1], though a number of authors have come close. The closest I have found to making this explicit is the idea of a "continuous matrix" version of GECP was described in a 2014 SIREV paper by Townsend and Trefethen, who comment that "we are not aware of explicit previous discussions of LU factorization of a cmatrix." A 2016 SISC paper by Drmač and Gugercin suggests the use of GECP in a similar context, but applied to the POD basis, and with an eye only to selecting interpolation points (rather than selecting a subspace basis); for this purpose, the pivoted QR approach that they analyze in detail (leading to the Q-DEIM method) seems preferable.

5 Extracting solutions

The POD method and the EIM method give us a subspace from which to draw approximations to $f(\mu)$. However, they do *not* tell us how to actually choose the approximations. Of course, we can use interpolation; but, as discussed earlier, we might prefer to use a method that consults with some defining set of equations.

For the sake of concreteness, let's again consider the problem of approximating the function $f(\mu)$ implicitly defined by the equation

$$A(\mu)f(\mu) = b.$$

[1] At least, two days of searching didn't turn it up

We will seek $\hat{f}(\mu) \in \mathcal{V}$ to approximately satisfy this equation. But what does it mean to "approximately" satisfy the equation? There are several possible ways we might approach this:

- *Least squares*: We might choose $\hat{f}(\mu) = \operatorname{argmin}_{y \in \mathcal{V}} \|A(\mu)y - b\|^2$. Provided we have a basis V for our space, this requires forming $A(\mu)V \in \mathbb{R}^{m \times k}$ (with k matrix-vector products), solving a least squares problem (in $O(mk^2)$ time), and then performing a matrix-vector product to reconstruct the solution (in $O(mk)$ time). If A is dense, the dominant cost is the time to compute $A(\mu)V$; this may or may not be true if A is sparse.

- *Galerkin*: We might choose a space $\mathcal{W}$ and enforce the condition $A(\mu)y - b \perp \mathcal{W}$. This is known as a *Bubnov-Galerkin* condition when $\mathcal{W} = \mathcal{V}$; otherwise, it is a *Petrov-Galerkin* condition. Unless A has some special structure, the costs are similar to the least squares approach, with the greatest cost being the formation of $W^T A(\mu)V$ (the "projected system matrix"), involving k matrix-vector products and an $O(mk^2)$ cost for a matrix-matrix multiplication. Note that if A is a linear function of μ (or a linear function of a few nonlinear functions of μ), we might be able to do some pre-computation to get the online cost down to $O(k^3)$.

- *Interpolation*: We might choose k equations to enforce; the indices of these equations are our "interpolation points" in the discrete setting. In this case, we might be able to avoid the cost of computing $A(\mu)V$, since we only need a few rows. Note that this is a special case of Galerkin where the test basis W consists of columns of the identity associated with the indices for the enforced equations. The challenge for this problem is choosing a good set of interpolation points (in the EIM setting, these are the points selected during the subspace construction).

The error analysis of each of these methods follows the same general quasi-optimality recipe: we establish a bound on the minimum distance between $f(\mu)$ and $\mathcal{V}$, then show that $\hat{f}(\mu)$ is within some constant factor of that best distance. The quasi-optimality factor depends on the stability of the projected problem, and can often be computed explicitly.

6 Addendum: Low-discrepancy sequences

Consider the problem of sampling on the unit hypercube, whether for quadrature or interpolation. One approach to distributing sample points is to take independent draws from the uniform distribution; however, random draws tend to include "clumps" (and leave some parts of the domain empty). What we would really like is a sampling scheme that ensures that points are reasonably spread out (like what happens with grids) and yet is "random-seeming" and not prone to aliasing issues (which is an advantage of random draws).

Discrepancy is a measure of the "clumpiness" of a distribution of sample points[2]. A number of low-discrepancy sequences are available; these include Halton sequences, Sobol sequences, and van der Corput sequences, for example. These sequences can be generated rapidly (they mostly involve simple recurrences), and there is good software available for them. Sampling via low-discrepance sequences (vs independent random draws) is the key distinction between quasi-Monte Carlo methods and Monte Carlo methods (and the reason that QMC tends to converge faster than standard MC).

[2]Actually, there are several measures that go under the common heading of "discrepancy," but all are getting at the same thing