DISTINGUISHING SIGNAL FROM NOISE IN AN SVD OF SIMULATION DATA

Paul G. Constantine

Stanford University
Mechanical Engineering Department
Stanford, CA

David F. Gleich

Purdue University
Computer Science Department
West Lafayette, IN

ABSTRACT

Our goal is to predict the output of a parameterized computer simulation code given a database of outputs at different parameter values. To do so, we investigate a particular model reduction technique that interpolates the right singular vectors in the singular value decomposition of the matrix of outputs. A common observation about these singular vectors is that they become more oscillatory as the index of the singular vectors increases. We use this property to split the singular vectors into "signal" and "noise" regions. The model reduction then interpolates the "signal" and uses the "noise" to estimate the uncertainty in the result. This methodology requires a big-data approach because the simulations we study produce snapshots with hundreds or thousands of timesteps on thousands to millions of nodal values. Each simulation output is then a vector with millions to billions of values. We utilize a MapReduce-based SVD routine to compute the SVD of the snapshot matrix.

1. INTRODUCTION

Simulation is now an established scientific endeavor, providing a straightforward and inexpensive proxy for expensive, impractical, or impossible experiments. As simulation's prominence and impact have grown over the past decades, analyzing the simulation outputs is a new challenge. Each simulation, which often depends on many input parameters, can take hours, days, or weeks on large clusters on computers. Parameter studies, such as uncertainty quantification or sensitivity analysis, are unrealistic because they require an extremely large number of separate simulations.

One possible approach to skirt around this issue is to design a surrogate function for the simulation. The surrogate function should be inexpensive to evaluate at many parameters. Then, we can perform a rigorous study of the easy-to-evaluate surrogate instead of the expensive-to-evaluate simulation. Ideally, the insight suggested by the surrogate can then be checked with a small number of additional simulation runs.

In order to make this idea more concrete, suppose that $\mathbf{f}(s) \in \mathbb{R}^m$ is a function representing the output of a simulation where the input is a vector of input parameters $s \in \mathbb{R}^d$. We are mainly concerned with simulations that can be described as the solution of a time-dependent partial differential equation. These methods typically evolve a set of spatially varying values over in a sequence of discrete timesteps. Let $x_1, \ldots, x_n$ be the spatial points – each x_i could be the coordinates in a three-dimensional space – and $t_1, \ldots, t_k$ be the sequence of time values. Thus, $\mathbf{f}(s)$ is a space-by-time sized vector describing the state of each nodal value at each timestep:

$$\mathbf{f}(s) = \begin{bmatrix} q(x_1, t_1, s) \\ \vdots \\ q(x_n, t_1, s) \\ q(x_1, t_2, s) \\ \vdots \\ q(x_n, t_2, s) \\ \vdots \\ q(x_n, t_k, s) \end{bmatrix},$$

where $q(x_i, t_j, s)$ is the simulation output at the ith position and the jth timestep for input s. Suppose we have evaluated $\mathbf{f}(s_1), \ldots, \mathbf{f}(s_p)$ and stored the results. This data corresponds to an m-by-p matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{f}(s_1) & \mathbf{f}(s_2) & \ldots & \mathbf{f}(s_p) \end{bmatrix}.$$

We refer to this matrix as the *snapshot matrix*. However, it should be noted that this terminology commonly refers to a matrix whose rows correspond to the spatial discretization and columns correspond to temporal sampling for a given parameter value [1]. In our case, rows of the snapshot matrix correspond to coordinates in space/time, and columns correspond to sampling the parameter space.

Inspired by the success of other surrogate models, including methods based on the proper orthogonal decomposition [2] and a residual based surrogate model [3], we investigate a scheme that approximates $\mathbf{f}$ as a linear combination of data-derived basis vectors $\{\mathbf{u}_j : j = 1, \ldots r\}$:

$$\mathbf{f}(s) \approx \sum_{j=1}^{r} \mathbf{u}_j \alpha_j(s).$$

Work from both authors completed at Sandia National Laboratories.

The major computational element is computing an SVD of the matrix $\mathbf{X}$ once to get the data-derived basis $\{\mathbf{u}_j\}$. Because of the massive scale of the data involved (thousands to millions of spatial points and hundreds to thousands of time steps), we employ a MapReduce architecture for this task. Each evaluation of the surrogate will involve computing the coefficients $\alpha_j(s)$ – which will be quite easy as we'll see in the next section – and multiplying them by the matrix of basis vectors in another MapReduce computation. We can easily evaluate this surrogate function for thousands of points simultaneously and without significant overhead.

2. SVD BASED MODEL REDUCTION

To explain the intuition for our method, consider a bivariate scalar function $g(x,s)$. Let

$$\mathbf{X} = \begin{bmatrix} g(x_1,s_1) & g(x_1,s_2) & \cdots & g(x_1,s_p) \\ g(x_2,s_1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & g(x_{m-1},s_p) \\ g(x_m,s_1) & \cdots & g(x_m,s_{p-1}) & g(x_m,s_p). \end{bmatrix}$$
$$= \mathbf{U\Sigma V}^T,$$

where the final form is given by the SVD of the matrix $\mathbf{X}$. (In terms of what was written in the introduction, note that when $\begin{bmatrix} g(x_1,s) & \cdots & g(x_m,s)\end{bmatrix}^T = \mathbf{f}(s)$, then we arrive at the snapshot matrix.) Our idea follows from the decoupling that arises in a functional interpretation of the SVD:

$$g(x_i,s_j) = \sum_{\ell=1}^{r} U_{i,\ell}\sigma_\ell V_{j,\ell} = \sum_{\ell=1}^{r} u_\ell(x_i)\sigma_\ell v_\ell(s_j),$$

where r is the rank of the matrix $\mathbf{X}$. For these samples, the SVD produces a decoupled-sum-of-products expression. Now, suppose we wish to evaluate $g(x_i,s)$ for a value $s \neq s_j$. Ideally, we'd have:

$$g(x_i,s) = \sum_{\ell=1}^{r} u_\ell(x_i)\sigma_\ell v_\ell(s).$$

Put another way, in a perfect world, there would be a *known* functional basis $v_\ell(s)$ that would determine our evaluations at a new point s. Reality isn't quite so nice, and our idea is to approximate $v_\ell(s)$ via an interpolation:

$$v_\ell(s) \approx \sum_{j=1}^{p} v_\ell(s_j)\beta_j^{(\ell)}(s).$$

That is, we interpret the right-singular vectors $\mathbf{V}$ as samples from an unknown functional basis $\{v_\ell(s)\}$ and interpolate each function $v_\ell(s)$ separately. Hence,

$$g(x_i,s) \approx \sum_{\ell=1}^{r} u_\ell(x_i)\sigma_\ell \sum_{j=1}^{p} v_\ell(s_j)\beta_j^{(\ell)}(s).$$

This can be evaluated for all values x_i simultaneously:

$$g(\cdot,s) \approx \mathbf{U\Sigma}\operatorname{diag}(\mathbf{V}^T\mathbf{B}(s)) = \mathbf{Ua}(s) = \sum_{\ell=1}^{T} \mathbf{u}_\ell \alpha_\ell(s),$$

where $\alpha_\ell = \sigma_\ell \sum_{j=1}^{p} \beta_j^{(\ell)}(s) v_\ell(s_j)$ and $B_{j,\ell} = \beta_j^{(\ell)}(s)$. Note that this is exactly the form we prescribed at the end of the introduction. Because we assume that p, the number of samples, is small (hundreds), performing the interpolation is a tractable problem. We can use any linear interpolation scheme – e.g., polynomial interpolation, radial basis functions, regression splines – for this task.

3. DISTINGUISHING SIGNAL FROM NOISE

While performing a linear interpolation is a well-understood problem, it is not always advisable. Let $g(x,s) = \frac{-1}{8s}\log\bigl(1 + 4s(x^2-x)\bigr)$. (See [4] for more about the origin of this function.) We take x_i to be 500 equally spaced points in $[0,1]$ and s_j to be 10 equally spaced points in $[-1,1]$. We have plotted the resulting singular vectors, interpreted as functions $v_\ell(s)$, in Figure 1.

Recall that our goal is to *interpolate* each of these functions in the parameter s. For a few of these functions, this looks like a good idea because the functions are relatively smooth and *resolved*, by which we mean that any oscillations are captured by the samples. However as the index ℓ increases, the functions become more oscillatory and it is not as clear that we can interpolate such functions. For reference, we also show (Figure 2) more highly resolved versions of these functions that result from taking s_j as 21 equally spaced points in $[-1,1]$.

Increased oscillatory behavior in the singular vectors is a common observation [5, Heuristic 2.1]. Based on this observation, we develop a simple heuristic to determine the reliability of an interpolant.

The essential idea is to estimate the gradient of the function $v_\ell(s)$ with respect to s for $\ell = 1,\ldots,r$, which we do by computing the gradient of the interpolant. The increasing oscillations of the $v_\ell(s)$ implies that the norm of the gradient $\|\partial v_{\ell^\star}/\partial s\|$ will increase as the index ℓ increases. Let $\ell^\star$ be the first index such that $\|\partial v_{\ell^\star}/\partial s\|$ is larger than a chosen threshold. Then we split the bases into two groups: *predictable* and *unpredictable*, where the predictable bases are well-suited for interpolation at the point s. Specifically, the basis functions $v_\ell(s)$ with $\ell = 1,\ldots,\ell^\star$ are deemed predictable while $v_\ell(s)$ for $\ell = \ell^\star+1,\ldots,r$ are deemed unpredictable.

In practice, we have found that cumulative summation is a more stable metric than the norm of the gradient. For a given threshold γ, we set $\ell^\star$ to be the largest τ such that

$$\sum_{\ell=1}^{\tau} \sigma_\ell \|\partial v_{\ell^\star}/\partial s\| \;<\; \gamma,$$

where σ_ℓ are the singular values of the snapshot matrix.

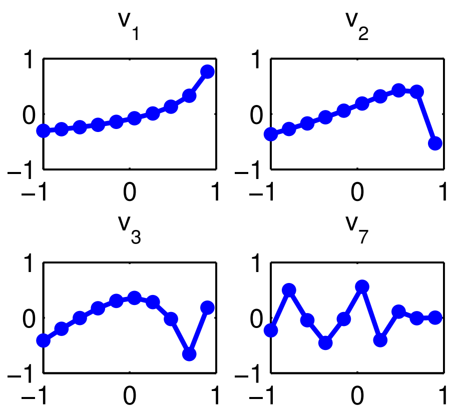

Fig. 1. An example of when the functions v_ℓ become difficult to interpolate. Each plot shows a singular-vector from the example in Section 3, which we interpret as a function $v_\ell(s)$. While we might have some confidence in an interpolation of $v_1(s)$ and $v_2(s)$, interpolating $v_3(s)$ for s nearby 1 is problematic, and interpolating $v_7(s)$ anywhere is dubious.

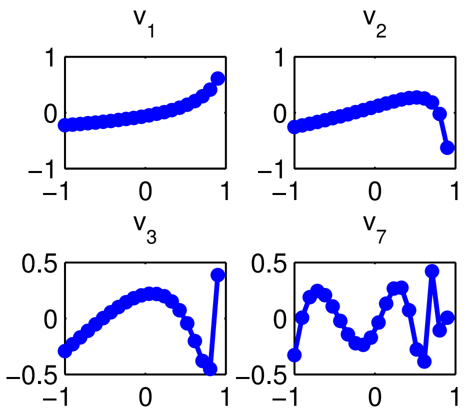

Fig. 2. For reference, we show a finer discretization of the functions above, which shows that interpolating $v_7(s)$ nearby 1 is difficult.

Once we have determined the predictable bases, we interpolate them using procedures discussed above to create the $\alpha_\ell(s)$. From the singular values and left singular vectors corresponding to the unpredictable bases, we can statistically characterize the noise in the surrogate function. This statistical characterization provides a time/space-varying prediction variance, which is related to the errors in the surrogate.

4. COMPUTING AN SVD WITH MAPREDUCE

Recall that $\mathbf{X}$ is m-by-p, where m is the product of the number of timesteps and spatial points, and p is the number of samples, and the biggest computational bottleneck in this algorithm is computing the SVD of this matrix. The matrix is extremely tall-and-skinny because there usually be millions to billions or rows and around 1000 columns. Consequently, we can use an R-SVD procedure [6] to compute the truncated-SVD of the matrix $\mathbf{X}$ by first doing a QR factorization of $\mathbf{X}$, then an SVD on the small matrix $\mathbf{R}$ that results. Let

$$\mathbf{X} = \mathbf{QR}$$

be a QR-factorization, then $\mathbf{R} = \mathbf{U}_R\mathbf{\Sigma V}^T$, and

$$\mathbf{X} = \underbrace{\mathbf{QU}_R}_{\mathbf{U}}\mathbf{\Sigma V}^T$$

is the SVD.

In practice, we use an approach in the MapReduce paradigm [7], which first computes the $\mathbf{R}$ in the QR factorization, and then computes $\mathbf{U} = \mathbf{XV\Sigma}^+$. This approach, although economical, may result in low accuracy if Σ is highly ill-conditioned and we continue to seek alternatives although we do not seem to observe the worst case loss-of-accuracy. For the QR factorization, we use a MapReduce implementation [8] of the communication-avoiding QR scheme [9].

Initially, each row of the matrix $\mathbf{X}$ is a record in the MapReduce paradigm, as is each record of the left singular vectors $\mathbf{U}$. Thus, after the SVD, the interpolation just involves distributing the coefficients $\mathbf{a}$ via the distributed cache and performing the inner-products. Moreover, we can compute the result for many interpolants simultaneously – a computational blocking technique that can amortize the effects of system overhead.

5. RESULTS

We now briefly present some results from a thermal-heating simulation of a complex geometry to illustrate the performance of this method on a real-world problem. There are three parameters s for this simulation, each of which controls a material property. The simulation is done with the Aria package in the SIERRA mechanics toolkit, both developed by Sandia National Laboratories for their simulations. An individual simulation has 240 time steps and 32768 spatial points and takes about 30 minutes to complete on a 32-core machine. Our database contained the output of 1000 simulations.

The SVD of this data took 30 minutes using the Dumbo python wrapper [10] with Hadoop 0.21 [11]. In Figure 4, we show a singular vector as a function. Subsequently, computing the data $\mathbf{a}$ for a single interpolant took about 4 seconds on a laptop. To evaluate 1000 separate interpolants took 8 minutes using a C++ code to do the matrix-vector products in a Hadoop streaming code.

The Hadoop cluster had 62 nodes, with 4 cores on each node. Thus, neglecting the cost of the SVD, the model reduction procedure takes $8\text{ minutes} \cdot (62\text{ nodes} \cdot 4\text{ cores/node})/1000\text{ simulations} = 1.98\text{ core-minutes}$ per simulation; whereas the original simulation took $32\text{ cores} \cdot 30\text{ minutes} = 960\text{ core-minutes}$, for a speedup of around 450.

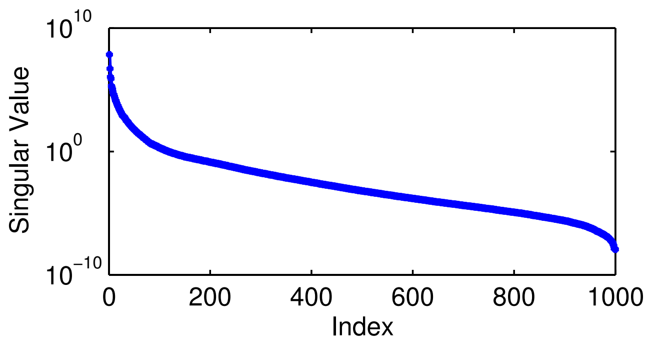

Fig. 3. The singular values of the thermal-heating simulation. We find that the singular vectors for the first 781 singular values are resolved for our heuristic. This is equivalent to the numerical rank.

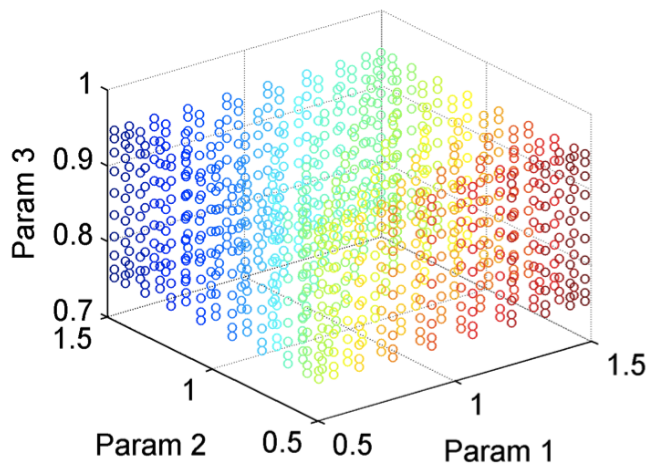

Fig. 4. The first right-singular vector of **X** from our simulation test-case, plotted as a three-dimensional function. Overall, this function is quite smooth, as are most of the functions for this problem.

Including the cost of the SVD, would could be amortized over many more simulations, reduces the advantage to 100-fold speedup. Here, we've still neglected the cost of computing the interpolant, which is small compared to the other times involved and easy to parallelize. This makes it feasible to use our SVD based surrogate function for the uncertainty quantification studies discussed in the introduction.

6. FUTURE DIRECTIONS

Given the complexities in altering an industrial simulation package like Aria, this work highlights an interesting way that *data* from a simulation can help build a surrogate for a simulation. In the future, we hope to develop additional theory to support our heuristic truncation of the singular values, as well as to explore additional applications.

7. REFERENCES

[1] L. Sirovich, "Turbulence and the dynamics of coherent structures. I - Coherent structures. II - Symmetries and transformations. III - Dynamics and scaling," *Quarterly of Applied Mathematics*, vol. 45, pp. 561–571, October 1987.

[2] G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual Review of Fluid Mechanics*, vol. 25, pp. 539–575, 1993.

[3] Paul G. Constantine and Qiqi Wang, "Residual minimizing model reduction for parameterized nonlinear dynamical systems," *arXiv*, vol. math.NA, pp. 1012.0351, 2010.

[4] Paul G. Constantine, David F. Gleich, and Gianluca Iaccarino, "Spectral methods for parameterized matrix equations," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 5, pp. 2681–2699, 2010.

[5] Per Christian Hansen, "Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank," *SIAM J. Sci. and Stat. Comput.*, vol. 11, no. 3, pp. 503–518, 1990.

[6] Gene H. Golub and Charles F. van Loan, *Matrix Computations*, Johns Hopkins Studies in Mathematical Sciences. The Johns Hopkins University Press, third edition, October 1996.

[7] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplied data processing on large clusters," in *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI2004)*, 2004, pp. 137–150.

[8] Paul G. Constantine and David F. Gleich, "Tall and skinny qr factorizations in mapreduce architectures," in *Proceedings of the second international workshop on MapReduce and its applications*, New York, NY, USA, 2011, MapReduce '11, pp. 43–50, ACM.

[9] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, "Communication-avoiding parallel and sequential QR factorizations," *arXiv*, vol. cs.NA, pp. 0806.2159, 2008.

[10] Klaas Bosteels, *Fuzzy techniques in the usage and construction of comparison measures for music objects*, Ph.D. thesis, Ghent University, 2009.

[11] Various, "Hadoop version 0.21," `http://hadoop.apache.org`, 2010.

Our thanks to Jeremy Templeton and Joe Ruthruff for providing the data in our simulation. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.