Self-Learning Emulators and Eigenvector Continuation

Xin Zhang (张馨)

中山大学物理与天文学院

2022年11月14日journal club

Outline

What's the research topic of the paper?

self-learning emulator

A self-learning emulator is an active learning protocol that can be used with any emulator that faithfully reproduces the exact solution at selected training points. The key ingredient is a fast estimate of the emulator error that becomes progressively more accurate as the emulator is improved, and the accuracy of the error estimate can be corrected using machine learning

Three example

1. The first uses cubic spline interpolation to find the solution of a transcendental equation with variable coefficients.
2. The second example compares a spline emulator and a reduced basis method emulator to find solutions of a parameterized differential equation.
3. The third example uses eigenvector continuation to find the eigenvectors and eigenvalues of a large Hamiltonian matrix that depends on several control parameters.

What's the research status of the topic?

Difficult computations

The frontiers of scientific discovery often reside just beyond the limits of computability. This explains the great interest across many scientific disciplines in using machine learning tools to build efficient emulators that predict scientific processes beyond what is possible with direct calculations. However, a problem arises in that large amounts of training data for such an emulator are not possible since the required computations are difficult and expensive.

→

Different from machine learning algorithms using using gradient descent optimization. While these gradient descent optimization methods are highly parallelizable and very fast, they usually suffer from critical slowing down with respect to error and cannot achieve arbitrarily high accuracy in polynomial computing time. Sometimes scientific discovery requires seeing very small but important new phenomena that might otherwise be absent in approximate machine learning models.

Constraint equations and error estimates

Consider a general set of simultaneous constraint equations $G_i(\mathbf{x}, \mathbf{c}) = 0$ that we solve for variables $\mathbf{x} = \{x_j\}$ as a function of control parameters $\mathbf{c} = \{c_k\}$ over some domain $\mathbf{D}$. Denote the exact solutions as $\mathbf{x}(\mathbf{c})$ For some set of training points $\{\mathbf{c}^{(i)}\}$ and construct an approximate solution $\tilde{\mathbf{x}}(\mathbf{c})$ for all $\mathbf{c} \in \mathbf{D}$. Let us define the error or loss function as the norm $\|\Delta\mathbf{x}(\mathbf{c})\|$ of the residual $\Delta\mathbf{x}(\mathbf{c}) = \mathbf{x}(\mathbf{c}) - \tilde{\mathbf{x}}(\mathbf{c})$. Minimize $\Delta\mathbf{x}$ using fewer training points When $\Delta\mathbf{x} \to 0$, we can accurately expand the constraint equations as

$$G_i(\tilde{\mathbf{x}}(\mathbf{c}), \mathbf{c}) + \Delta\mathbf{x}(\mathbf{c}) \cdot \nabla_{\mathbf{x}} G_i(\tilde{\mathbf{x}}(\mathbf{c}), \mathbf{c}) \approx 0 \tag{1}$$

- Number of degrees of freedom is small: use Newton-Raphson method to estimate $\log||\Delta\mathbf{x}||$
- Number of degrees of freedom is large: choose another positive function $F\left[\{G_i(\tilde{\mathbf{x}}(\mathbf{c}), \mathbf{c})\}\right]$ as a surrogate for $\|\Delta\mathbf{x}(\mathbf{c})\|$. The only essential requirement we impose on $F\left[\{G_i(\tilde{\mathbf{x}}(\mathbf{c}), \mathbf{c})\}\right]$ is that it is linearly proportional to $\|\Delta\mathbf{x}(\mathbf{c})\|$ in the limit $\|\Delta\mathbf{x}(\mathbf{c})\| \to 0$. This allows us to write the logarithm of the error as

$$\log \|\Delta\mathbf{x}(\mathbf{c})\| = \log F\left[\{G_i(\tilde{\mathbf{x}}(\mathbf{c}), \mathbf{c})\}\right] + A + B(\mathbf{c}) \tag{2}$$

Solution of the HWG equation

where A is a constant and the average of B(**c**) over the domain **D** is zero. In the limit of large number of training points, we can neglect the much smaller variation of B(**c**) over the domain D. We can therefore approximate the logarithm of the error as $\log F\left[\{G_i(\tilde{\mathbf{x}}(\mathbf{c}), \mathbf{c})\}\right] + A$. The unknown constant A is irrelevant for comparing the logarithm of the error for different points c. Nevertheless, we can also quickly estimate A simply by taking several random samples of **c** and computing the average value of the difference between $\log \|\Delta x(\mathbf{c})\|$ and $\log F\left[\{G_i(\tilde{\mathbf{x}}(\mathbf{c}), \mathbf{c})\}\right]$. We can refine this estimate further using machine learning to approximate the function B(**c**) .

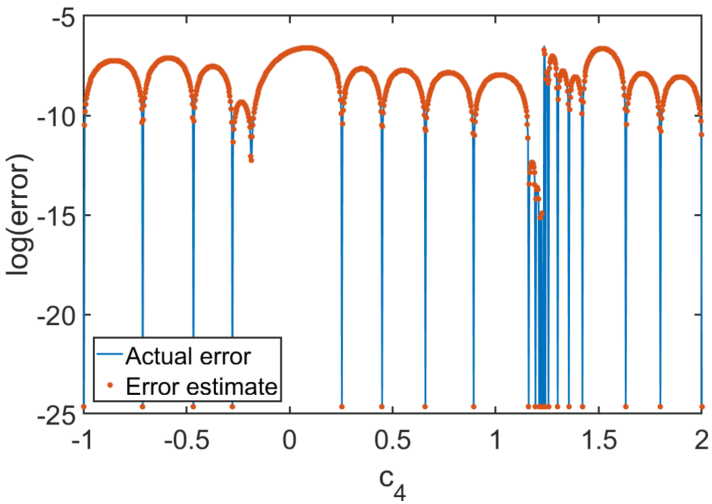

Figure: self-learning Emulator

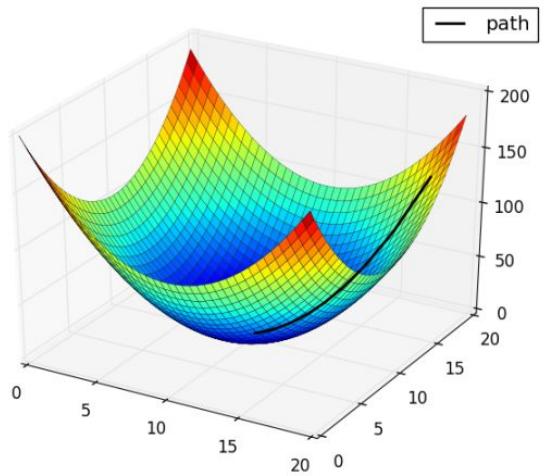

Figure: Machine learning

What has been done in this paper?

Model 1

transcendental equation: $x^5 + c_4 x^4 \sin(10x) + x^3 + x^2 + x + 1 = 0$,

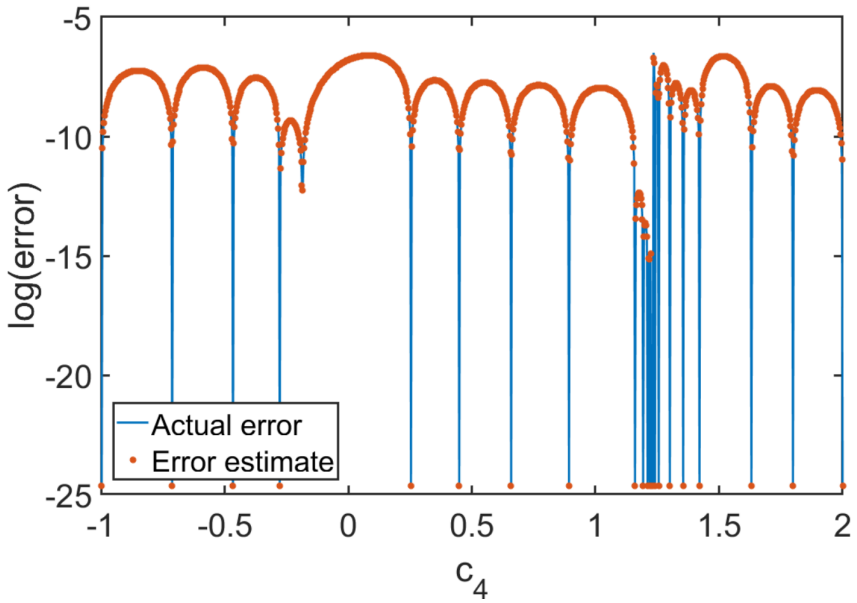

Figure: Logarithm of the actual error and error estimate for the cubic spline self-learning emulator after 20 iterations

We start with three training points for c_4 , two on the boundary and one in the interior, and use natural cubic splines to define the cubic spline approximation $\tilde{x}(c_4)$ for all values of c_4 . The logarithm of the error function is then $\log|\Delta x(c_4)|$ where $\Delta x(c_4) = x(c_4) - \tilde{x}(c_4)$. We can estimate $|\Delta x(c_4)|$ using the Newton-Raphson method,

$$|\Delta x(c_4)| \approx \frac{|p(\tilde{x}(c_4))|}{\sqrt{|p'(\tilde{x}(c_4))|^2 + \epsilon^2}} \quad (3)$$

What has been done in this paper?

Model 1

transcendental equation: $x^5 + c_4 x^4 \sin(10x) + x^3 + x^2 + x + 1 = 0$,

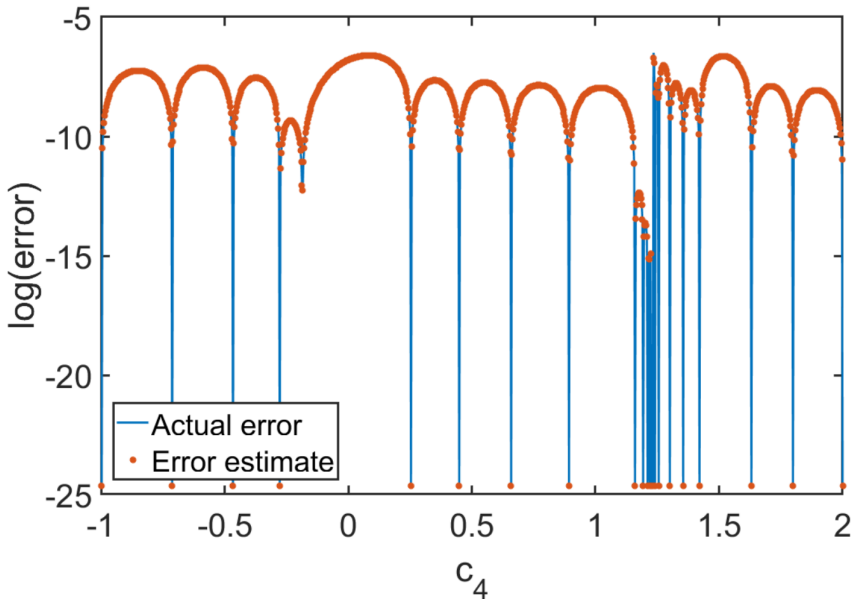

Figure: Logarithm of the actual error and error estimate for the cubic spline self-learning emulator after 20 iterations

We start with three training points for c_4 , two on the boundary and one in the interior, and use natural cubic splines to define the cubic spline approximation $\tilde{x}(c_4)$ for all values of c_4 . The logarithm of the error function is then $\log|\Delta x(c_4)|$ where $\Delta x(c_4) = x(c_4) - \tilde{x}(c_4)$. We can estimate $|\Delta x(c_4)|$ using the Newton-Raphson method,

$$|\Delta x(c_4)| \approx \frac{|p(\tilde{x}(c_4))|}{\sqrt{|p'(\tilde{x}(c_4))|^2 + \epsilon^2}} \quad (4)$$

Model 2

transcendental equation: $L = \frac{1}{(1+2z)^2}\frac{d^2}{dz^2} - \frac{2}{(1+2z)^3}\frac{d}{dz} + c^2 e^{2c} L = \frac{1}{(1+2z)^2}\frac{d^2}{dz^2} - \frac{2}{(1+2z)^3}\frac{d}{dz} + c^2 e^{2c}$

Instruction

It's a family of differential equations $Lx(z) = 0$, c is a real parameter. Our boundary conditions are $x(z = 0, c) = 0$ and $\partial_z x(z = 0, c) = 1$ for all c . We consider the region $0 \leq z \leq 1$, and $0 \leq c \leq 1$. The exact solution is $x(z, c) = \frac{1}{ce^c} \sin\left[ce^c\left(z + z^2\right)\right]$. We consider twoand c is a real parameter. Our boundary conditions are $x(z = 0, c) = 0$ and $\partial_z x(z = 0, c) = 1$ for all c . We consider the region $0 \leq z \leq 1$, and $0 \leq c \leq 1$. The exact solution is $x(z, c) = \frac{1}{ce^c} \sin\left[ce^c\left(z + z^2\right)\right]$. We consider two different emulators. natural spline emulator and a reduced basis emulator.For our fast error estimate F[x(z, c), c] , we need some function that is linearly proportional to the actual error $\|\Delta x(z, c)\|$ in the limit $\|\Delta x(z, c)\| \to 0$. we choose

$$F[\tilde{x}(z, c), c] = \left\| \frac{L\tilde{x}(z, c)}{\sqrt{\left(\frac{d}{dz} L\tilde{x}(z, c)\right)^2 + \epsilon^2}} \right\|_1 \tag{5}$$

Model 2

transcendental equation: $L = \frac{1}{(1+2z)^2}\frac{d^2}{dz^2} - \frac{2}{(1+2z)^3}\frac{d}{dz} + c^2 e^{2c} L = \frac{1}{(1+2z)^2}\frac{d^2}{dz^2} - \frac{2}{(1+2z)^3}\frac{d}{dz} + c^2 e^{2c}$

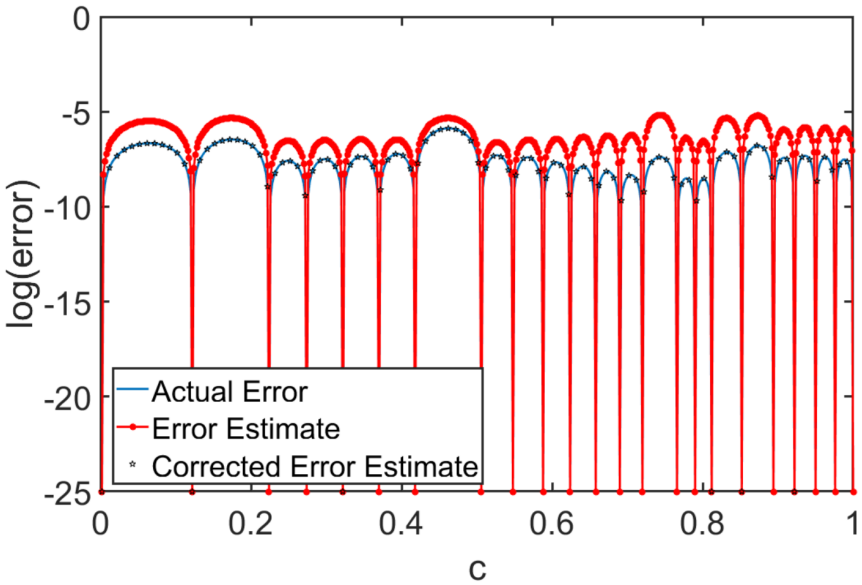

Figure: Logarithm of the actual error, error estimate, and corrected error estimate for the natural spline emulator with self-learning in Model 2 after 20 iterations

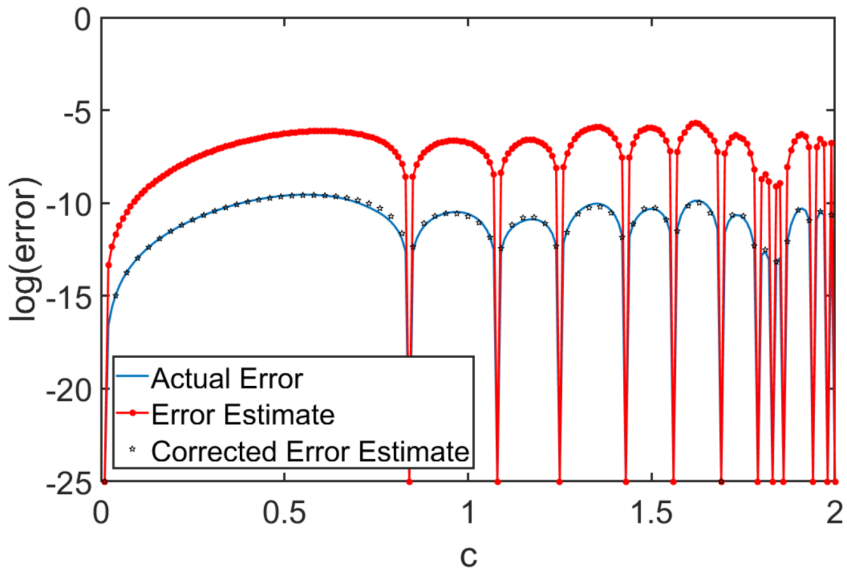

Figure: Logarithm of the actual error, error estimate, and corrected error estimate for the reduced basis emulator with self-learning in Model 2 after 10 iterations

Model 3 (eigenvector continuationeigenvector continuation)

$H(c)$ is a manifold of Hamiltonians where the dependence on the control parameters c (Here is the six possible pairwise interactions, c_{ij} , with i < j for four distinguishable particles under ground state) is smooth, and its eigenvector $|\Delta v(\mathbf{c})\rangle$

Instruction

The logarithm of the error is $\log \||\Delta v(\mathbf{c})\rangle\|$, where $|\Delta v(\mathbf{c})\rangle = |v(\mathbf{c})\rangle - |\tilde{v}(\mathbf{c})\rangle$. Computing the error directly will be computationally too expensive for large systems, and so we will instead work with $\log F[\tilde{v}(\mathbf{c}), H(\mathbf{c})]$, where $F[\tilde{v}(\mathbf{c}), H(\mathbf{c})]$ is proportional to the square root of the variance of the Hamiltonian,

$$F[\tilde{v}(\mathbf{c}), H(\mathbf{c})] = \sqrt{\frac{\left\langle \tilde{v}(\mathbf{c}) \left| [H(\mathbf{c}) - \tilde{E}(\mathbf{c})]^2 \right| \tilde{v}(\mathbf{c}) \right\rangle}{\left\langle \tilde{v}(\mathbf{c}) \left| [H(\mathbf{c})]^2 \right| \tilde{v}(\mathbf{c}) \right\rangle}}. \tag{6}$$

We note that $F[\tilde{v}(\mathbf{c}), H(\mathbf{c})]$will be linearly proportional to $\||\Delta v(\mathbf{c})\rangle\|$ in the limit $\||\Delta v(\mathbf{c})\rangle\| \to 0$. Therefore $\log F[\tilde{v}(\mathbf{c}), H(\mathbf{c})]$ can be used as a surrogate for the logarithm.$|\tilde{v}(\mathbf{c})\rangle$ is the eigenvector of $H(c)$ onto the subspace spanned by the training eigenvectors $\{|\tilde{v}(\mathbf{c}^i)\rangle\}$

Model 3

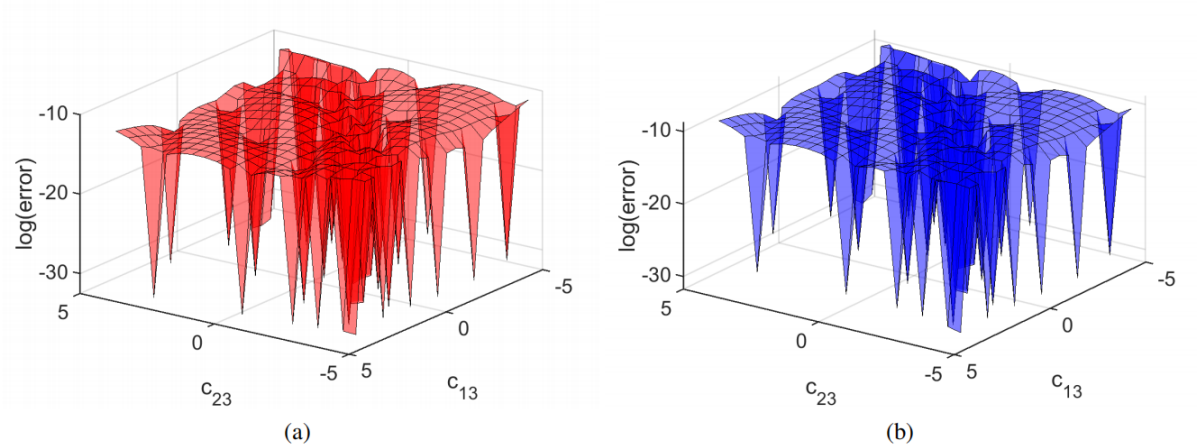

Figure: Logarithm of the error in Model 3 after 40 iterations using self-learning EC. In panel (a) we show the logarithm of the actual error (red), and in panel (b) we show the logarithm of the estimated error (blue)

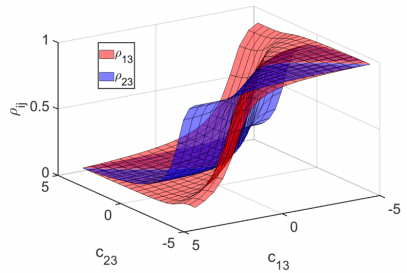

Figure: Plot of the two-particle clustering and short-range correlations in Model 3. ρ 13 (red) measures the probability that particles 1 and 3 occupy the same lattice site, and the correlation function ρ 23 (blue) measures the probability that particles 2 and 3 occupy the same lattice site

What's the conclusion?

Conclusion

Self-learning emulation is a general approach that can be implemented with any emulator that faithfully reproduces the exact solution at selected training points. They use a fast estimate for the error in the training process and perform full calculations only for the chosen new training points. If needed, the difference between the estimated error and exact error can be corrected using machine learning. If many evaluations are required, the computational advantage can grow as large as the raw speedup factor of the emulator, S_{raw}, which can be several orders of magnitude or more. Self-learning emulators are a highly efficient class of algorithms that offer both high speed and accuracy as well as a reliable estimate of the error.

What's the novelty?

novelty

- Model 1 : $S_{raw} < 10^5$
- Model 2 : $S_{raw} < 130$
- Model 3 : $S_{raw} < 150$