The Journal of Open Source Education

An open source crash course on parameter estimation of computational models using a Bayesian optimization approach

Mojtaba Barzegari[1] **and Liesbet Geris**[1, 2]

1 Biomechanics Section, Department of Mechanical Engineering, KU Leuven, Belgium **2** Biomechanics Research Unit, GIGA in Silico Medicine, University of Liège, Belgium

DOI: 10.21105/jose.00089

Software

- Review
- Repository
- Archive

Submitted: 07 April 2020
Published: 22 June 2021

Summary

Parameter estimation is a crucial aspect of computational modeling projects, especially the ones that deal with ordinary differential equations (ODE) or partial differential equation (PDE) models. Well-known examples in this regard are models derived from a basic balance or conservation law, such as mass balance or heat transfer problems. For real-world applications, these equations contain some coefficients that cannot be obtained directly from published scientific materials or experimental studies (Dehghan, 2001). One of the best solutions to this challenge is constructing an inverse problem.

According to (Tarantola, 1987), inverse modeling is the use of the actual results of some measurements of observable parameters to infer the actual values of the model parameters. Put differently, what we want to do is estimating parameters that cannot be directly measured for our computational model. This is also called parameter estimation or model calibration (Magni & Sparacino, 2014). Indeed, we calibrate our model to act similarly to available experimental data, and then this calibrated model can be used to simulate other scenarios that haven't been tested yet in the experiments. This is a common process in a lot of modeling problems in science and engineering.

Take a simple reaction-diffusion equation as an example, in which the change of the concentration of a sample chemical component C is studied over time. By assuming that the correlated chemical reaction is $A + 2B \rightleftharpoons C$, occurring in a diffusible medium (such as a chemical solution), the PDE to describe the mass transfer phenomenon over time can be written as (Grindrod, 1996):

$$\frac{\partial [C]}{\partial t} = \nabla . (D_C \nabla [C]) + k_1 [A][B]^2 - k_2 [C]$$

in which $[X]$ denotes the concentration of the chemical component X, the D_C is the diffusion coefficient of C in the medium, and k_1 and k_2 are the rates of the forward and backward reactions, respectively. To solve this PDE numerically and get quantitative data (which is the goal of most of the scientific computing projects), we need to know the value of D_C, k_1, and k_2, which is usually hard-to-find in the literature.

As mentioned above, one solution is to to solve the inverse problem, in which we can use optimization techniques to minimize the difference between the model output and experimental data. Bayesian optimization is one of the most efficient approaches in this regard (Mockus, 1989). `HyperOpt` (Bergstra et al., 2013) is a Python package that provides easy-to-use interfaces to implement a Bayesian optimization problem, making it a proper choice for both educational and practical purposes. In this educational module, we used this package to teach the principles of an efficient parameter estimation pipeline.

For demonstration purposes, an interpolation problem is solved by using the parameter estimation techniques that a computational modeling researcher employs for model calibration. Indeed, the computational intensive code is replaced with a simple function evaluator, which helps students to learn the core concepts without waiting too much for the process to finish. Students will be guided through several steps of refining the results inside the notebook, in which the interactive computing environment of Jupyter facilitates exploring the implementation more efficiently in comparison to traditional educational materials.

Statement of Need

Despite its simplicity, building an inverse problem is hard for many students. The problem is although it is relatively simple to describe the process visually, implementing it for a practical application becomes challenging in its early stages. In this educational module, a simple optimization problem is implemented in a Jupyter notebook to teach students how to construct an inverse problem and tune it to get better results in such a problem. In this way, students can work on a real-world optimization problem in an interactive environment and learn the concepts behind taking advantage of a modern optimization method (Bayesian approach) for parameter estimation of a computational model.

This notebook is indeed a modern learning module for relatively old and frequently-used concepts (global optimization, Bayesian techniques, inverse problems). It has been designed to be useful for both teachers and students. Students can use it as a self-study guide for parameter estimation and inverse problem construction, while teachers can change the underlying problem to any other desired one easily and make the learning module compatible with their own teaching requirements.

Learning objectives

Upon completion, students will be able to:

- Understand the concept and necessity of parameter estimation in science and engineering
- Describe what the whole process of Bayesian optimization is all about
- Define and implement a Bayesian optimization workflow for parameter estimation of common use-cases
- Critically evaluate the output of the process and fine-tune the setup of the Bayesian optimization
- Apply the obtained knowledge to any kind of models that are commonly used in science and engineering

Prerequisites

In order to go through the learning module, the students should have a working knowledge of programming in Python. Additionally, a basic understanding of mathematics is required to get the concept of models in science and engineering. The given example is a mathematical model derived from differential equations, so knowledge of differential equations can help to understand the importance of parameter estimation in these widely-used models. However, in case of necessity, the example can be replaced by any other relevant one for the target learners.

Pedagogy and instructional design

The provided material is the format of a crash course, which is suitable for being taught in one session of undergraduate or graduate courses for science and engineering students. Courses to which this material is relevant can be “optimization,” “scientific computing,” or “parametric design.” The material may also be useful for relevant educational projects for the target students, in which they can employ the learned techniques to construct efficient inverse problems for parameter estimation.

The teaching strategy is based on the worked-example effect (Chen et al., 2015), in which an example of parameter estimation is fully implemented to allow students to play with and modify the code to have their own reflection in class discussions. Basic prior knowledge of Python suffices as the problem doesn’t involve students with complicated programming stuff. The student-centric characteristic of this crash course helps teachers to adopt the material easily and integrate it into an existing syllabus of relevant courses in science and engineering.

Getting started

The learning material is provided as a single Jupyter notebook, in which all the steps of constructing an inverse problem is described in detail with accompanying Python codes. A very simple simulation code (in the context of an interpolation problem) is also provided and can be found in the repository. The code inside the notebook calls this external program at certain points to mimic the interaction of the parameter estimation routine and the main computational code that contains the unknown parameters.

To get started with the module, the user should set up the environment first. The setup instruction is provided in the `README.md` file of the repository. After setting up the Jupyter notebook and installing the required packages, the user can navigate to the `src` folder and run the `CrashCourseOnParameterEstimation.ipynb` notebook. No further action is needed as the content of the notebook is self-explanatory and easy-to-follow.

Acknowledgements

This research is financially supported by the Prosperos project, funded by the Interreg VA Flanders – The Netherlands program, CCI grant no. 2014TC16RFCB046 and by the Fund for Scientific Research Flanders (FWO), grant G085018N. We acknowledge support from the European Research Council under the European Union’s Horizon 2020 research and innovation programmen, ERC CoG 772418.

References

Bergstra, J., Yamins, D., & Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *Proceedings of the 30th International Conference on Machine Learning - Volume 28*, I-115-I-123.

Chen, O., Kalyuga, S., & Sweller, J. (2015). The worked example effect, the generation effect, and element interactivity. *Journal of Educational Psychology*, *107*(3), 689–704. https://doi.org/10.1037/edu0000018

Dehghan, M. (2001). An inverse problem of finding a source parameter in a semilinear parabolic equation. *Applied Mathematical Modelling*, *25*(9), 743–754. https://doi.org/10.1016/S0307-904X(01)00010-5

Grindrod, P. (1996). *The theory and applications of reaction-diffusion equations : Patterns and waves*. Oxford University Press.

Magni, P., & Sparacino, G. (2014). 5 - parameter estimation. In E. Carson & C. Cobelli (Eds.), *Modelling methodology for physiology and medicine (second edition)* (Second Edition, pp. 83–110). Elsevier. https://doi.org/10.1016/B978-0-12-411557-6.00005-7

Mockus, J. (1989). *Bayesian approach to global optimization*. Springer Netherlands. https://doi.org/10.1007/978-94-009-0909-0

Tarantola, A. (1987). *Inverse problem theory: Methods for data fitting and model parameter estimation*. Elsevier Science Ltd. https://doi.org/10.1137/1.9780898717921