Environmental Modelling & Software 92 (2017) 17–27

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Appraisal of data-driven and mechanistic emulators of nonlinear simulators: The case of hydrodynamic urban drainage models

Juan Pablo Carbajal*, João Paulo Leitão, Carlo Albert, Jörg Rieckermann

Swiss Federal Institute of Aquatic Science and Technology, Eawag, Überlandstrasse 133, 8600 Dübendorf, Switzerland

ARTICLE INFO

Article history:
Received 10 October 2016
Received in revised form
1 February 2017
Accepted 2 February 2017

Keywords:
Gaussian Processes
Mechanistic emulation
Model order reduction
Hydrology
Principal components
Nonnegative matrix factorization

ABSTRACT

System identification, sensitivity analysis, optimization and control, require a large number of model evaluations. Accurate simulators are too slow for these applications. Fast emulators provide a solution to this efficiency demand, sacrificing unneeded accuracy for speed. There are many strategies for developing emulators but selecting one remains subjective. Herein we compare the performance of two kinds of emulators: *mechanistic emulators* that use knowledge of the simulator's equations, and purely *data-driven emulators* using matrix factorization. We borrow simulators from urban water management, because more stringent performance criteria on water utilities have made emulation a crucial tool within this field. Results suggest that naive data-driven emulation outperforms mechanistic emulation. We discuss scenarios in which mechanistic emulation seems favorable for extrapolation in time and dealing with sparse and unevenly sampled data. We also point to advances in Machine Learning that have not permeated yet into the environmental science community.

1. Introduction

For many real-world systems with a nonlinear response, model based tasks such as sensitivity analysis, learning model parameters from data (i.e. system identification or model calibration), and real-time control, are hampered by the long runtime of the employed numerical simulators. Even if runtimes are short, these methods require a large number of model runs, which can take a prohibitive long time. One way of speeding up these tasks is to build fast surrogate models, so called *emulators*, to replace the computationally expensive simulators. An emulator is a numerical model that is tailored to approximate the results of a computationally expensive simulator with a huge reduction in the time needed to run a simulation (O Hagan, 2006), i.e. it is a metamodel. These ideas also belong to the technique of Reduced-Order Models (ROM), specially for models based on Partial Differential Equations (PDE) (Baur et al., 2014; Quarteroni et al., 2016), and emulation as described below.

To ground ideas, imagine that the flow at the outlet of a drainage network is limited using a flow limiting gate or by activating water storage systems (Fig. 1). The position of the gate and the activation of storage is controlled using a model predictive controller (Xi et al., 2013). Such a scenario is relevant in performance optimization of water treatment plants (Fu et al., 2008). The signals used to control the flows could be the current intensity and duration of rain events from several rain gauges within the catchment. The controller needs to estimate an optimal course of action by predicting the flows induced by the rain and many possible actuations. This optimization generally requires thousands of model runs, which can take a prohibitive long time when running a physically detailed simulator of the sewer network, such as a EPA Storm Water Management Model (SWMM) model (Rossman, 2010). However, the simulator is just used to estimate the relation between the rain, the actuation, and the flow. The full details of the simulator might not be required to obtain an accurate estimation of this relation. Feedback control might further reduce the required accuracy of the estimated relation.

Emulation and interpolation are equivalent problems. An emulator is built using the best available simulator to sample the space of actuations and/or parameters (henceforth the latter will include actuations). The training data is then used to build an interpolation function which should predict values at unseen parameters with an acceptable degree of accuracy, which is case dependent. That is, we reconstruct an unknown function $F : \mathbb{R}^{|\vec{\gamma}|+1} \to \mathbb{R}$, that takes a parameter vector of size $|\vec{\gamma}|$ and a time instant, and generates the value of the magnitude of interest. When this function is evaluated at the inputs used for training, the results

* Corresponding author.
E-mail address: juanpablo.carbajal@eawag.ch (J.P. Carbajal).

http://dx.doi.org/10.1016/j.envsoft.2017.02.006
1364-8152/

Fig. 1. An imaginary drainage network with flow limiting gate and/or water storage systems. The interesting singal is the level/flow at the outlet of the network, marked with a circle. Isometric tiles from www.kenney.nl.

are the same as the training outputs,[1] i.e. this is the meaning of interpolation of the training data adopted herein. Stated in this way, no distinction is needed between the parameters and the time components in the input. However, knowing that the data is generated by a dynamical system, we separate time from the other parameter components. Thus, we can find one interpolant in time and one in parameter space, which might be coupled to each other. This parameter-time coupling emerges naturally in mechanistic emulation, as will be shown in Sec. 2.2.

When the simulator is based on differential equations, the link between Gaussian Processes (GP) and linear stochastic differential equations (SDE) (Albert, 2012; González et al., 2014; Poggio and Girosi, 1990; Solin, 2014; Särkkä et al., 2013; Steinke and Schölkopf, 2008), permits the creation of GP based emulators that include knowledge about the simulator dynamics; these are called *mechanistic emulators* (MEMs). Conceptually, mechanistic emulation seeks a function that interpolates the training data whithin a class of functions defined by an SDE. The importance of GPs for MEMs stems from the fact that they are the formal solution of this SDE. Hence when the simulator is linear the emulator gives exact results; while for nonlinear simulators, the MEM will provide only an approximation. Increasing the accuracy of this approximation and the efficiency of the methods are two fundamental challenges in GP based emulation (O Hagan, 2006; Rasmussen and Williams, 2006, Ch. 8).

Reichert et al. (2011). enumerated four overlapping approaches for developing emulators of dynamic simulators:

i) Gaussian Processes
ii) Basis function decompositions
iii) State space transition function approximation
iv) Stochastic linear model conditioned on data using Kalman smoothing.

In particular approaches i) and iv) are two different implementations of the same problem (Steinke and Schölkopf, 2008). Roughly speaking the Kalman smoothing algorithm used in iv) is an iterative implementation of the conditioning of the GP in i). The iteration in iv) avoids the ill-conditioned covariance matrices (Hansen, 1998) involved in GP when sampling rates are high (Reichert et al., 2011; Steinke and Schölkopf, 2008) and it is faster than direct matrix inversion in a serial implementation. The GP approach i) is better suited for parallelization, speedups and energy saving via approximated computing (Angerer et al., 2015).

Approaches i) (or iv)) and ii) are similar with respect to their implementation. That is, approach ii) can be implemented using GP regression (Rasmussen and Williams, 2006, sec. 2.7). Therefore, the essential difference between i) and ii) is that the former explicitly introduces mechanistic knowledge. It will be shown here that approach i) is currently constrained to linear mechanistic knowledge, while popular methods based on maximum entropy (Christakos, 1998; Harte, 2011; Victor and Johannesma, 1986) can handle nonlinear knowledge. The difference between GP based mechanistic emulation and maximum entropy methods is that in the latter, the mechanistic knowledge is added as constraints on the moments of the predictive or posterior distribution, while in the former its is added as the dynamics of the prior model. Adding constraints to predictive distributions require expert knowledge available at the level of the emerging behavior of the simulator, while adding dynamic information requires knowledge about the constitutive elements parts of the simulator. The latter is likely to be readily available from the development of the simulator itself.

Herein we compare the performance of GP emulators built using approaches i), which we call *mechanistic emulation*, and ii) which we call *data-driven emulation*. The basis function that will be used for data-driven emulation will be derived solely from the data using

[1] Herein considered noiseless since they are generated by a deterministic simulator.