Autonomous learning of parameters in differential equations

Adel Mezine ADEL.MEZINE@IBISC.FR
Université d'Evry Val d'Essonne, IBISC, Evry, France

Artémis Llamosi ARTEMIS.LLAMOSI@UNIV-PARIS-DIDEROT.FR
Université de Paris Diderot, MSC, CNRS UMR 7057, Paris, France

Veronique Letort VERONIQUE.LETORT@ECP.FR
Ecole Centrale Paris, MAS, Châtenay-Malabry, France

Michèle Sebag MICHELE.SEBAG@LRI.FR
Université de Paris-Sud, LRI, CNRS UMR 8623, Orsay, France

Florence d'Alché-Buc FLORENCE.DALCHE@TELECOM-PARISTECH.FR
Ecole Telecom ParisTech, LTCI, CNRS UMR 5141, Paris, France

Abstract

We propose EDEN+[1] a fully automatic learner of parameters in dynamical systems that selects automatically the next experiment to do in the laboratory to improve its performance. EDEN+ improves upon EDEN, an experimental design algorithm proposed in the context of DREAM 6 and 7 challenges, with several new features: ability to take into account experiments with different costs, Monte-Carlo Tree Search parallelization, global optimization for parameter estimation. An illustration of the behaviour of EDEN+ is given on one of the DREAM7 challenging problems.

Keywords: active learning, system identification, Monte-Carlo Tree Search, multi-armed bandits, gene regulatory networks, differential equations

1. Motivation

Discovery in science relies on the choice of appropriate experiments whose results allow to refute or confirm working hypotheses. Machine learning can provide a precious help in the laboratory not only by automatically analyzing the data at hand but also by making suggestions to the experimenter about which data to acquire as already shown in King et al. (2004). This is especially relevant in a biology lab where the cost of experiments is very often prohibitive. Taking the example of dynamical modeling in systems biology, we address the general problem of parameter estimation of differential equations together with the experimental design in a sequential manner. Starting from an initial system of parametric ordinary differential equations and an initial dataset, the EDEN+ algorithm tackles this active learning problem as one player game where each move is a choice of a single experiment (usually a perturbation experiment). The game is won if the parameters and the hidden states are estimated with sufficient accuracy.

1. Experimental Design for parameter Estimation in a Network

2. Problem description

We consider a dynamical system whose state at time t is the d-dimensional vector $\mathbf{x}(t)^T = [x_1(t) \dots x_d(t)]$ and whose dynamics are modeled by first-order ODE governed by a function f depending on parameters $\boldsymbol{\theta}$ and on an exogenous input $\mathbf{u}(t)$.

We partially observe its behavior given some initial condition $\mathbf{x}(0) = \mathbf{x}_0$ and some neutral input $\mathbf{u}(t) = g_0(t)$, e.g. without any *intervention* (as defined below). Let $\mathbf{H}$ be the observation model, typically a projection of $\mathbb{R}^d$ in a lower dimensional space $\mathbb{R}^p$ $(p < d)$, $\mathbf{Y}_0 = (\mathbf{y}^0_{t_k})_{k=0,\dots,n-1}$, a time series of n p-dimensional observations and $(\boldsymbol{\epsilon}_{t_k})_{k=0,\dots,n-1}$, n i.i.d realizations of a p-dimensional noise. For sake of simplicity, $\mathbf{y}_{t_k}$ (resp. $\boldsymbol{\epsilon}_{t_k}$) will be noted $\mathbf{y}_k$ (resp. $\boldsymbol{\epsilon}_k$). Given these assumptions, the observations and the states of the system in Quach et al. (2007) are expressed as follows: given $k = 0, \dots, n-1$:

$$\begin{aligned} \mathbf{x}(0) &= \mathbf{x}_0 \\ \mathbf{x}(t_{k+1}) &= \mathbf{x}(t_k) + \int_{t_k}^{t_{k+1}} f(\mathbf{x}(\tau), \mathbf{u}(\tau), \boldsymbol{\theta}) d\tau \\ \mathbf{y}_k &= \mathbf{H}(\mathbf{x}(t_k), \mathbf{u}(t), \boldsymbol{\theta}) + \boldsymbol{\epsilon}_k \ . \end{aligned} \tag{1}$$

This model is a state-space model with the particularity that the hidden process is deterministic and computed using a numerical integration. Although nonlinear filtering approaches such as Unscented Kalman Filtering (UKF) in Quach et al. (2007) and extended Kalman Filtering (EKF) in Wang et al. (2009) can be applied, we will use here global maximization algorithm of the log-likelihood. A major difficulty in parameter and hidden state estimation is the practical non-identifiability of parameters. Namely, two different parameter solutions can provide the same likelihood value. A well-known way to address this issue is to intervene on the dynamical system to perform additional experiments producing observations that exhibit different kinetics. It can consist either in perturbing the system, e.g. forcing the level of a state variable to be zero, or in changing the observation model by allowing to observe different state variables. To benefit from these new data during the estimation phase, the ODE model must account for all the available experiments defined by a finite set of size E: $\mathcal{E} = \mathcal{E}_0 = \{e_1, \dots, e_E\}$. This can be done by defining adequately the exogenous input $\mathbf{u}(t)$ among a set of intervention functions $\mathbf{g}_e(t), e \in \mathcal{E}$ as shown in the application section.

Automatic selection of the appropriate interventions (experiments) to apply to the system is the purpose of this work. We are especially interested in an *active learning algorithm* that sequentially selects at each step ℓ, the next experiment e^*_ℓ among the candidate experiments of the set $\mathcal{E}_\ell = \mathcal{E}_{\ell-1} - \{e^*_{\ell-1}\}$, that will produce the most useful dataset for the estimation task. Contrary to the purely statistical approaches of experimental design, ours aims at offering the possibility to anticipate on the fact that one given experiment will be followed by others. The search for an optimal $e^*_\ell \in \mathcal{E}_\ell$ thus depends on the potential subsequent sequences of experiments, their total number being limited by a finite affordable *budget* to account for the cost of real experiments. In this work, we improve a previous version of the active learning algorithm EDEN presented in Llamosi et al. (2014) by increasing considerably its autonomy and its scope of application.

3. Algorithm

As described in Llamosi et al. (2014) the EDEN algorithm consists mainly in three parts: the estimation, the design of experiment (DOE) and the experimentation. The estimation step aims at exploiting the current available data to learn the parameters of the model. The design of experiment is automatically performed by an algorithm of the Monte-Carlo Tree Search (MCTS) family that suggests a sequence of one or more experiments. The experimentations are done with respect to the recommendations of the DOE procedure. Algorithm 1 presents the improved version of EDEN.

Data: Budget: B; Initial data: $\mathcal{D}_0$; Design space: $\mathcal{E}$; Initial candidate parameters: Θ_0.
Algo: $\mathcal{L}earner$; $\mathcal{R}eward$.
Input: Intermediate budget: b; Initial model: $\mathcal{M}_0$; Cost function: $error()$;
Tree policy: ϕ; Recommendation: ψ; Number of tree walks: N; Maximum tree depth: T;
Size of the version space: N_{VS}; Rejection threshold: $\lambda > 1$.
Output: $\hat{\Theta}$.
$budget \leftarrow B$
$\mathcal{D} \leftarrow \mathcal{D}_0$
$\mathcal{M} \leftarrow \mathcal{M}_0$
while $\inf_{e \in \mathcal{E}} price(e) \leq budget$ **do**
  $VS = \emptyset$
  `/* Building an empirical version space */`
  while $card(VS) < N_{VS}$ **do**
    $\hat{\Theta} \leftarrow \mathcal{L}earner(\Theta_0, \mathcal{M}, \mathcal{D}, error())$
    $VS \leftarrow VS \cup \left\{ \boldsymbol{\theta} \in \hat{\Theta} \mid \forall (\mathcal{M}_i, \mathcal{D}_i) \in \mathcal{M} \times \mathcal{D}, error(\mathcal{M}_i, \mathcal{D}_i, \boldsymbol{\theta}) < \lambda \cdot \min_\theta(error(\mathcal{M}_i, \mathcal{D}_i, \boldsymbol{\theta})) \right\}$
    $\Theta_0 \leftarrow \hat{\Theta}$
  end
  `/* Design of experiment based on Monte-Carlo Tree Search */`
  $e \leftarrow MCTS(\phi, \psi, N, \mathcal{E}, VS, \mathcal{L}earner, min(b, B), horizon, reward())$
  $\mathcal{D}_e \leftarrow Experimentation(e)$; `// In silico or in vivo experiment`
  $\mathcal{M}_e \leftarrow Modeling(e)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_e$
  $\mathcal{M} \leftarrow Fusion(\mathcal{M}, \mathcal{M}_e)$
  $\mathcal{E} \leftarrow \mathcal{E} \backslash \{e\}$; `// Replicated experiment are not considered`
  $budget \leftarrow budget - price(e)$
end
$\hat{\Theta} \leftarrow \mathcal{L}earner(\Theta_0, \mathcal{M}, \mathcal{D}, error())$
return $\hat{\Theta}$

Algorithm 1: EDEN+

The EDEN+ algorithm requires a finite set of experiments $\mathcal{E}$ that are feasible in real world and can be simulated by an appropriate model. A cost is associated to each experiment, and a finite budget $B > 0$ is available. A prior is required to initialize the EDEN+ algorithm. Depending on the choice of the estimation algorithm, the prior can be formulated either as a hypercube, that delimits the region of parameters to explore, a probability distribution or a population of parameters. At each step the $\mathcal{L}earner$ updates this prior Θ_0.
The estimation phase aims at exploring the parameter space in order to extract a finite subset of plausible parameter vectors called *Version Space*. It is essential that this set highlights any non-identifiability issues that may exist. Furthermore, it should exhibit a wide spectrum of behaviours with respect to the next experiments that can be considered. In

order to solve the parameter estimation problem, we propose to use the global optimization algorithm Cooperative Enhanced Scatter Search (CESS) introduced in Villaverde et al. (2012). CESS is a metaheuristic approach that has shown good performances in various problems as mentioned in Villaverde et al. (2015). The term *Cooperative* indicates that we refer to the parallel implementation in which several threads run ESS in parallel and synchronize regularly.

During the step of experimental design, the MCTS algorithm is applied to explore efficiently the set of sequences of experiments from $\mathcal{E}$ thanks to a game tree. An intermediate budget b and a horizon h are required to limit the depth of the tree search. A policy ϕ is used to determine which experiment to select. In Kocsis and Szepesvári (2006), the Upper Confident bound for Tree (UCT) uses the Upper Confident Bound (UCB) algorithm as selection policy to deal with the exploration-exploitation trade-off. The version of MCTS

Input: Policy: ϕ; Recommendation: ψ; Number of tree walks: N; Design space: $\mathcal{E}$;
Version Space: VS; $\mathcal{L}earner$; Intermediate budget b; $horizon$; $\mathcal{R}eward$
Output: Recommended experiment: e
$tree = \{root\}$
$treewalk = 1$
while $treewalk < N$ **do**
  $\mathcal{A} \leftarrow \mathcal{E}$
  $path = \{root\}$
  $node = root$
  while $(price(path) < b) \wedge (length(path) < horizon)$ **do**
    $a \leftarrow \phi(tree, path, \mathcal{A})$
    $node \leftarrow getNode(tree, path, a)$; `// Void if the action has not associated node`
    if $node = \emptyset$ **then**
      $node \leftarrow tree.addNode(path, a)$; `// Add a child associated to the action`
    end
    $\mathcal{A} \leftarrow \mathcal{A} \backslash \{a\}$; `// Replicated experiment are not considered`
    $path \leftarrow path \cup node$
  end
  $E \leftarrow getAction(tree, path, \mathcal{E})$
  $r \leftarrow \mathcal{R}eward(E, VS, \mathcal{E}, \mathcal{L}earner)$
  `/* Back-propagation of the reward through the visited path */`
  $tree.backPropagate(path, r)$
end
return $e \leftarrow \psi(tree, \mathcal{E})$

Algorithm 2: MCTS: Monte-Carlo Tree Search

presented in Algorithm 2 has been adapted to allow considering that different experiments can have different costs and thus stop the expansion of the tree search once the budget is exhausted.

The parallelization is an efficient way to increase the power of the MCTS algorithm. Several parallelization methods have been investigated in Gelly et al. (2008) and Chaslot et al. (2008). There are mainly two approaches to parallelize MCTS. The first method is to use a shared memory to store one version of the tree, and several threads explore the tree to update it. The second method that has been chosen here (due to the way Matlab implements multi-threaded applications) consists in building a tree per thread and synchronizing them via messages. In our case, the time to compute the reward is larger than the time to update

the tree, thus the synchronization can be done frequently. In this way, the decisions taken by the algorithm are always based on the most recent information.

4. Experimental result

EDEN+ has been confronted with the best competitor of the sub-challenge of parameter inference in Dream7 Challenge, namely the *orangeballs* team in Meyer et al. (2014). Given the topology and the underlying mechanistic model of an *in silico* 9-gene regulatory network, the participants are asked to estimate the parameter vector and to predict the concentrations of some proteins in specific conditions. The full model is given under the form of an ordinary differential equation system that describes the evolution of the protein and mRNA concentration in time associated to each gene.
The performance of the participants is evaluated using two scores D_{param} and D_{pred} that are respectively based on the parameter vector estimated and the protein timecourse prediction. D_{param} represents the error in order of magnitude between the true vector parameter $\boldsymbol{\theta}^*$ and the estimated $\hat{\boldsymbol{\theta}}$.

$$D_{param}(\boldsymbol{\theta}, \boldsymbol{\theta}^*) = \frac{1}{N_\theta} \sum_{i=1}^{N_\theta} \left[log \left(\frac{\hat{\theta}_i}{\theta_i^*} \right) \right]^2 . \tag{2}$$

D_{pred} is the normalized mean squared error between the prediction of three timecourse protein and the trajectories simulated with the true parameters for a given set of perturbations.

$$D_{pred}(\boldsymbol{\theta}, \boldsymbol{\theta}^*) = \frac{1}{90} \sum_{k=3,5,8} \sum_{i=1}^{40} \frac{\left[p_k(\hat{\boldsymbol{\theta}}, t_i) - p_k(\boldsymbol{\theta}^*, t_i) \right]^2}{0.01 + 0.04 \cdot p_k(\boldsymbol{\theta}^*, t_i)^2}. \tag{3}$$

In order to apply MCTS, a reward function is designed from the two scores to evaluate the utility of the sequence of experiments explored. The reward consists in multiplying the mean of reduction of each component of D_{param} and D_{pred} computed between a surrogate oracle $\boldsymbol{\theta}^* \in VS$ and $\boldsymbol{\theta}^{opt}(Y) = \text{argmin}_{\boldsymbol{\theta} \in R^d}\, error(\mathcal{M}, Y)$ the optimum outcome of the model with respect to the available data Y.

$$reward(\boldsymbol{\theta}^*, Y_0, Y_1) = \left(1 - \frac{D_{param}(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{opt}(Y_0 \cup Y_1))}{D_{param}(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{opt}(Y_0))} \right) \cdot \left(1 - \frac{D_{pred}(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{opt}(Y_0 \cup Y_1))}{D_{pred}(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{opt}(Y_0))} \right). \tag{4}$$

Figure 1 shows the evolution of the mean of the scores in the version space. The version space in updated sequentially by adding one dataset at a time to set of observations. The tree is explored with an horizon 3 that makes the algorithm looks ahead to reach a better score in long term.

Table 1 shows the final scores obtained by EDEN+ and DREAM competitors whose methods are described in Meyer et al. (2014). EDEN+ performance is comparable to the first-rank team, with a better prediction score and no human intervention.

5. Implementation

We develop an implementation with the standard version of Matlab. The computation time needed to play the full challenge is one week, using 12 cores of a 4 processors Intel

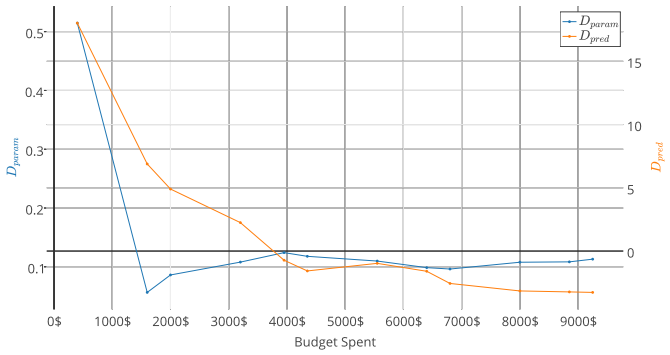

Figure 1: Evolution of the scores with $h = 3$, $N = 20000$, $N_{VS} = 5000$.

Challenger	D_{param}	D_{pred}
orangeballs	**0.0229**	0.0024
Team #341	0.8404	0.0160
Team #92	0.1592	0.0354
EDEN+ ($h = 3$)	0.0371	**0.0016**

Table 1: Comparison of the final scores on Dream 7 Model 1. Only the name of winner *orangeballs* has been revealed. The results are shown for an horizon of 3 experiments.

Xeon X5670 2.93GHz. The whole EDEN+ is entirely automated for *in silico* problem. The simulations are performed using a numerical integration algorithm provided by the *sundials* suite detailed in Hindmarsh et al. (2005). The optimization part of parameter inference problem is performed by the MEIGO toolbox by Egea et al. (2014).

6. Conclusion

A very general algorithm has been developed for dynamical systems identification and experimental design. One of the most important features of this algorithm is its ability to take into account the fact that an experiment is followed by other experiments and thus to anticipate. Allowing for different experiment costs opens the door to real applications and parallelism is helpful to reduce the computational load of exploration and estimation. Working perspectives concern the extension of the approach to other learning problems related to system biology.

References

Guillaume MJ-B Chaslot, Mark HM Winands, and H Jaap van Den Herik. Parallel monte-carlo tree search. In *Computers and Games*, pages 60–71. Springer, 2008.

Jose A Egea, David Henriques, Thomas Cokelaer, Alejandro F Villaverde, Aidan MacNamara, Diana-Patricia Danciu, Julio R Banga, and Julio Saez-Rodriguez. Meigo: an open-source software suite based on metaheuristics for global optimization in systems biology and bioinformatics. *BMC bioinformatics*, 15(1):136, 2014.

Sylvain Gelly, Jean-Baptiste Hoock, Arpad Rimmel, Olivier Teytaud, and Yann Kalemkarian. On the parallelization of monte-carlo planning. In *ICINCO*, 2008.

Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM TOMS*, 31(3):363–396, 2005.

Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

Artémis Llamosi, Adel Mezine, Florence d'Alché Buc, Véronique Letort, and Michele Sebag. Experimental design in dynamical system identification: A bandit-based active learning approach. In *ECML/PKDD 2014*, pages 306–321. Springer, 2014.

Pablo Meyer, Thomas Cokelaer, Deepak Chandran, Kyung H Kim, Po-Ru Loh, George Tucker, Mark Lipson, Bonnie Berger, Clemens Kreutz, Andreas Raue, et al. Network topology and parameter estimation: from experimental design methods to gene regulatory network kinetics using a community based approach. *BMC systems biology*, 8(1):13, 2014.

Minh Quach, Nicolas Brunel, and Florence d'Alché Buc. Estimating parameters and hidden variables in non-linear state-space models based on odes for biological networks inference. *Bioinformatics*, 23(23):3209–3216, 2007.

Alejandro F Villaverde, Jose A Egea, and Julio R Banga. A cooperative strategy for parameter estimation in large scale systems biology models. *BMC systems biology*, 6(1): 75, 2012.

Alejandro F Villaverde, David Henriques, Kieran Smallbone, Sophia Bongard, Joachim Schmid, Damjan Cicin-Sain, Anton Crombach, Julio Saez-Rodriguez, Klaus Mauch, Eva Balsa-Canto, et al. Biopredyn-bench: a suite of benchmark problems for dynamic modelling in systems biology. *BMC systems biology*, 9(1):8, 2015.

Zidong Wang, Xiaohui Liu, Yurong Liu, Jinling Liang, and Veronica Vinciotti. An extended kalman filtering approach to modeling nonlinear dynamic gene regulatory networks via short gene expression time series. *IEEE/ACM TCBB*, 6(3):410–419, 2009.