MM Algorithms to Estimate Parameters in Continuous-time Markov Chains

Giovanni Bacci Anna Ingólfsdóttir Kim G. Larsen
Raphaël Reynouard

Continuous-time Markov chains (CTMCs) are a model of a dynamical system that constitute the underlying semantics for real-time probabilistic systems such as queuing networks [13], stochastic process algebras [8], and calculi for systems biology [5, 9]. Model checking tools such as Prism [10] and Storm [6] provide access to a number of powerful analysis techniques for CTMCs. Both tools accept models written in the Prism language, an expressive state-based language based on [1] that represents synchronous and asynchronous components in a uniform framework that supports compositional design.

The outcome of the analysis of a Prism model is strongly dependent on the parameter values used in each module, as they govern the timing and probability of events of the CTMC describing its semantics. Consider for instance a variant of the Susceptible-Infected-Recovered (SIR) model proposed in [15] to describe the spread of COVID-19 in presence of lockdown restrictions (*cf.* Fig. 1). The model is parametric in the variables `beta`, `gamma`, and `plock`. Their values have to be empirically evaluated from a number of partially-observable executions, typically in the form of a time series that plots the number of infected individuals day-by-day.

Neither Prism nor Storm provide integrated support for this kind of task, leaving the burden of estimating parameter values to the modeler.

A paradigmatic example is the modeling pipeline described in [15], where the parameters of the SIR model in Fig. 1 are estimated based on a definition of the model as ODEs, and later used in an approximation of the original SIR model designed to reduce its state space. Such modeling pipelines require high technical skills, are error-prone, and are time-consuming, thus limiting the applicability and the user base of model checking tools.

In recent work [3], we address the problem of estimating parameter values of CTMCs expressed as Prism models from a number of partially-observable

```
ctmc
 // SIR model paramaters
const double beta; const double gamma;
const double plock;
const int SIZE = 100000;  // population size

module SIR
s : [0..SIZE] init 99936;
i : [0..SIZE] init 48;
r : [0..SIZE] init 16;

[] i>0 & i<SIZE & s>0 →
   beta * s * i * plock/SIZE : (s'=s − 1)&(i'=i + 1);
[] i>0 & r<SIZE →
    gamma * i * plock : (i'=i − 1)&(r'=r + 1);
endmodule
```

Figure 1: SIR model with lockdown from [15]

executions. The expressive power of the PRISM language brings two technical challenges: (i) the classic state-space explosion problem due to modular specification, and (ii) the fact that the transition rates of the CTMCs result from the algebraic composition of the rates of different (parallel) modules which are themselves defined as arithmetic expressions over the parameters. We address the second aspect of the problem by considering the class of *parametric* CTMCs, which are CTMCs where transition rates are polynomial functions over a fixed set of parameters. In this respect, parametric CTMCs have the advantage to cover a rich subclass of PRISM models and to be closed under the operation of parallel composition.

Following the standard approach, we pursue the maximum likelihood estimate (MLE), i.e., we look for the parameter values that achieve the maximum joint likelihood of the observed execution sequences $\mathcal{O}$. For a systematic treatment of the non-convex surface described from the likelihood function, we employ a theoretical iterative optimization principle known as MM algorithm [12, 11]. This optimization technique generalizes the well-known Expectation-Maximization (EM) algorithm [7] by employing an elegant theory of inequalities that allows one to derive simple, yet effective, optimization procedures.

As the EM algorithm, our algorithm starts with an initial valuation of the parameters $\mathbf{x}_0$ which is iteratively updated in a way that the likelihood is nondecreasing at each step, that is $\mathcal{L}(\mathcal{O}|\mathbf{x}_m) \leq \mathcal{L}(\mathcal{O}|\mathbf{x}_{m+1})$, until the likelihood difference between the current and the previous hypothesis goes below a fixed threshold ϵ. The update of each parameter x_{mi} revolves around finding a root of a univariate polynomial function $P_i(y)$. The coefficients of $P_i(y)$

are obtained by performing a forward-backward procedure in the same fashion as the Baum-Welch algorithm [16]. Notably, under mild assumptions on the format of the (multivariate) polynomial functions describing the transition rates of the parametric CTMC, the equation $P_i(y) = 0$ admits a simple closed-form solution. A detailed description of the parameter estimation procedures here mentioned can be found in [3] and have been implemented in the JAJAPY python library [17].

JAJAPY also implements a number of methods to learn different types of Markov models from partially-observable executions. These algorithms include variants of the Baum-Welch algorithm [16] as well as ALERGIA [4, 14] for learning discrete-time Markov chains (MCs) and Markov decision processes (MDPs). For the case of MDPs, which typically demand more observations, JAJAPY offers a model-based active learning sampling strategy that chooses examples that are most informative w.r.t. the current model hypothesis [2].

References

[1] Rajeev Alur and Thomas A. Henzinger. Reactive modules. *Formal Methods Syst. Des.*, 15(1):7–48, 1999. doi:10.1023/A:1008739929481.

[2] Giovanni Bacci, Anna Ingólfsdóttir, Kim G. Larsen, and Raphaël Reynouard. Active Learning of Markov Decision Processes using Baum-Welch algorithm. In M. Arif Wani, Ishwar K. Sethi, Weisong Shi, Guangzhi Qu, Daniela Stan Raicu, and Ruoming Jin, editors, *20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021*, pages 1203–1208. IEEE, 2021. doi:10.1109/ICMLA52953.2021.00195.

[3] Giovanni Bacci, Anna Ingólfsdóttir, Kim G. Larsen, and Raphaël Reynouard. MM Algorithms to Estimate Parameters in Continuous-time Markov Chains. *CoRR*, abs/2302.08588, 2023. URL: https://arxiv.org/abs/2302.08588, arXiv:2302.08588.

[4] Rafael C. Carrasco and José Oncina. Learning Stochastic Regular Grammars by Means of a State Merging Method. In *ICGI-94*, volume 862 of *LNCS*, pages 139–152. Springer, 1994. doi:10.1007/3-540-58473-0_144.

[5] Federica Ciocchetta and Jane Hillston. Bio-pepa: A framework for the modelling and analysis of biological systems. *Theor. Comput. Sci.*, 410(33-34):3065–3084, 2009. doi:10.1016/j.tcs.2009.02.037.

[6] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A Storm is Coming: A Modern Probabilistic Model Checker. In *CAV 2017*, volume 10427 of *LNCS*, pages 592–600. Springer, 2017. doi:10.1007/978-3-319-63390-9_31.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[8] Jane Hillston. *A compositional approach to performance modelling*. PhD thesis, University of Edinburgh, UK, 1994. URL: http://hdl.handle.net/1842/15027.

[9] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Using probabilistic model checking in systems biology. *SIGMETRICS Perform. Evaluation Rev.*, 35(4):14–21, 2008. doi:10.1145/1364644.1364651.

[10] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *CAV 2011*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011. doi:10.1007/978-3-642-22110-1_47.

[11] Kenneth Lange. *Optimization*. Springer New York, NY, 2 edition, 2013.

[12] Kenneth Lange. *MM Optimization Algorithms*. SIAM, 2016. URL: http://bookstore.siam.org/ot147/.

[13] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative system performance - computer system analysis using queueing network models*. Prentice Hall, 1984. URL: https://homes.cs.washington.edu/~lazowska/qsp/.

[14] Hua Mao, Yingke Chen, Manfred Jaeger, Thomas D. Nielsen, Kim Guldstrand Larsen, and Brian Nielsen. Learning Deterministic Probabilistic Automata from a Model Checking Perspective. *Machine Learning*, 105(2):255–299, 2016. doi:10.1007/s10994-016-5565-9.

[15] Paolo Milazzo. Analysis of covid-19 data with prism: Parameter estimation and sir modelling. In Juliana Bowles, Giovanna Broccia, and

Mirco Nanni, editors, *From Data to Models and Back*, pages 123–133, Cham, 2021. Springer International Publishing.

[16] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi:10.1109/5.18626.

[17] Raphaël Reynouard. Jajapy (v 0.10), 2022. URL: https://github.com/Rapfff/jajapy.