Applied Interpretable Inference: Connecting Probabilistic Programming with Epidemiology Simulators

Paper ID:

Abstract

A goal of probabilistic programming is to couple simulators, with inference. This is because stochastic simulators are used prominently in many industrial settings, do not require one to construct hand-crafted joint distributions as they implicitly define a joint distribution of the program and encode learnt structures directly. This makes simulators powerful tools and much of machine learning (ML) and Artificial Intelligence (AI) can be seen as trying to emulate such simulators from a purely data-driven approach. However, in the ML/AI setting, although we can often infer outcomes, we have little understanding about what in the data led to the outputted inferences. This makes it challenging to deploy ML/AI systems into the wild, especially in health-related and safety-critical domains, such as epidemiology, as we lose *interpretability*. In this work, we explain how to design ML/AI systems that combine probabilistic programming systems (PPSs) and epidemiology simulators, to extract fully interpretable posterior structures, enabling policy makers and practitioners to make interpretable inferences. In particular, we demonstrate this for the Malaria disease and show how we can perform interpretable inference in such settings.

1 Introduction

To do.

- Outline the objectives
- State the importance of interpretability in the inference of the simulator
- Our system enables one to understand and interpret the most common paths in a program (simulation).
- In highlighting this a practitioner can then go back to the field and explore that parameter/s in more depth.
- This facilitates understanding and the construction of more detailed models and data collection strategies

Stochastic simulators are common across a number of domains, statistical physics [6], financial modeling [5], weather prediction [2], epidemiology [9] and many others. An advantage of using stochastic simulators is that they provide a level of interpretability not found in modern deep learning settings, as they directly incorporate model structure from carefully reasoned observations and experiments. Probabilistic programming systems (PPSs) are purpose built systems for making inferences and probabilistic modeling accessible. In particular, PPSs allow probabilistic models to be represented in the form of a generative model through the probabilistic programming language (PPL), which enables one to write statements that enable conditioning on data [4, 3]. Thus, it seems only natural to connect simulators with PPSs, as simulators explicitly define a generative model, in the language in which they are written, and the nature of the PPSs enables us to perform inference in these simulators, by conditioning on observations, which are fed in as input to the simulators. In doing this we can infer things about stochastic input parameters and other variables sampled during the program's forward execution, whilst also providing full interpretability in the inference results, which is absolutely necessary in safety-critical domains. By exploiting the tools that we develop in this paper, one is able to connect epidemiology simulators with `Pyprob` [7] to extract fully interpretable posterior structures allowing practitioners to interpret which input parameters have the largest affect on the epidemic, in our case malaria. Over five-hundred-thousand people die from malaria each year, mostly children under five years of age, with 90 per cent of malaria cases occurring in Sub-Saharan Africa. An estimated 100-300 million people suffer from malaria each year [8]. Thus, by understanding the prominence of input parameters to epidemiology simulators, guides decision making processes relating to effective treatment and prevention strategies.

2 Epidemiology Simulators and Probabilistic Programming

This will kind of be like a background section.

3 Methodology

TODO

Simulators encode several years, if not decades, of research and development and so by their very nature are structurally complicated. Since the development happens over several decades, usually without documentation, understanding a code base written with legacy libraries and software is incredibly difficult and an almost impossible task for a non-expert, or expert who has not had access to the previous years of development. Our method of hijacking simulators builds on the work of [1] and extends the framework to encapsulate a more diverse range of simulators, in particular epidemiology simulators. Our framework provides a simple solution and makes it easy to transform arbitrary stochastic simulators into probabilistic programs, regardless of the complexity of the simulator and the language that the simulator is written in[1]. In addition to this, we can understand the structure of programs in simulators for both decoding *black-box structures* and for posterior inference. This enables two things. 1) It enables software developers and researchers to understand complicated code bases. 2) It provides interpretable inference results that provide policy makers with predictions that are truly interpretable, in the sense that the end-user of the inference results understands what physical events led to inference outcome. This is critical in several domains, and is particularly important in the medical domain. We provide examples of both in Section **TO ADD REF**.

3.1 Connecting a Stochastic Simulator to a Probabilistic Programming System

In order to hijack the stochastic simulator we are only required to override the location of the stochastic primitives, i.e. the operations that generate the stochasticity and randomness within the simulator. We will walk through how this is done for EMOD and OpenMalaria, you will see that it is procedurally identical, however, given the sheer number of simulators it maybe slightly different procedurally. To our knowledge it shouldn't be, but we have not been able view of all stochastic simulators. The first step is to build a containerized environment, such as Docker**ADD citation** and Singularity, which enables the simulators to be run on any device, independent of different hardware architectures removing the requirement for hardware specific machines. We provide examples of building a containerized environment in the Supplement **ADD ref**. This environment only has to be built once, which then enables the simulator to be run simultaneously on multiple machines. The second step

3.2 Extending Probabilistic Programming Protocols

TODO

[1]The framework currently supports 9 popular languages, but there is nothing to stop this being extended to any language of interest.

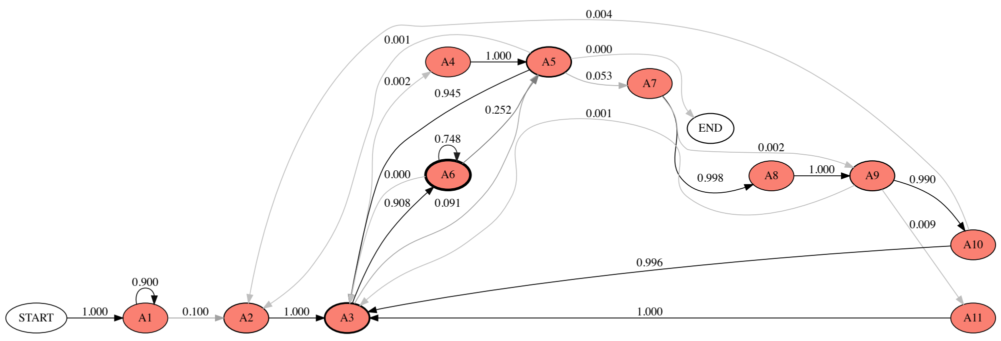

Figure 1: Add something. Population 10

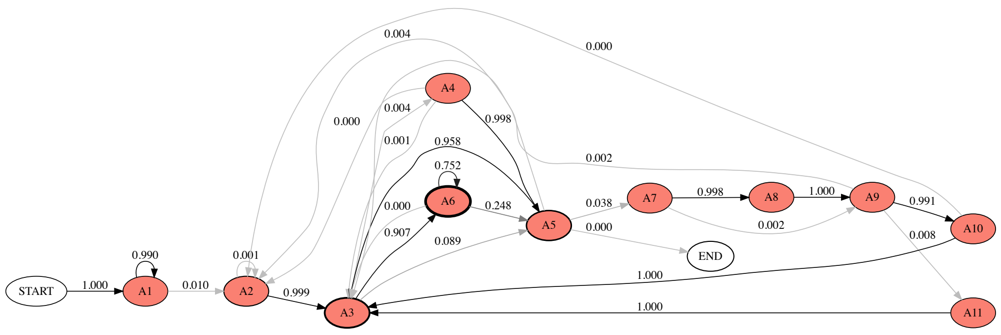

Figure 2: Add something. Population 100

4 Results

TO ADD

References

References

[1] Atilim Gunes Baydin, Lukas Heinrich, Wahid Bhimji, Bradley Gram-Hansen, Gilles Louppe, Lei Shao, Kyle Cranmer, Frank Wood, et al. Efficient probabilistic inference in the quest for physics beyond the standard model. *arXiv preprint arXiv:1807.07706*, 2018.

[2] Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

[3] Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: A language for generative models. In *In UAI*, pages 220–229, 2008.

[4] Andrew D Gordon, Thomas A Henzinger, Aditya V Nori, and Sriram K Rajamani. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*, pages 167–181. ACM, 2014.

[5] P. Jäckel. *Monte Carlo Methods in Finance*. The Wiley Finance Series. Wiley, 2002.

[6] David P. Landau and Kurt Binder. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, 4 edition, 2014.

[7] Tuan Anh Le, Atılım Güneş Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pages 1338–1348, Fort Lauderdale, FL, USA, 2017. PMLR.

[8] World Health Organization. *World malaria report 2015*. World Health Organization, 2016.

[9] T Smith, N Maire, A Ross, M Penny, N Chitnis, A Schapira, A Studer, B Genton, C Lengeler, Fabrizio Tediosi, et al. Towards a comprehensive simulation model of malaria epidemiology and control. *Parasitology*, 135(13):1507–1516, 2008.