Hijacking Simulators with Universal Probabilistic Programming

Paper ID: 1010

Abstract

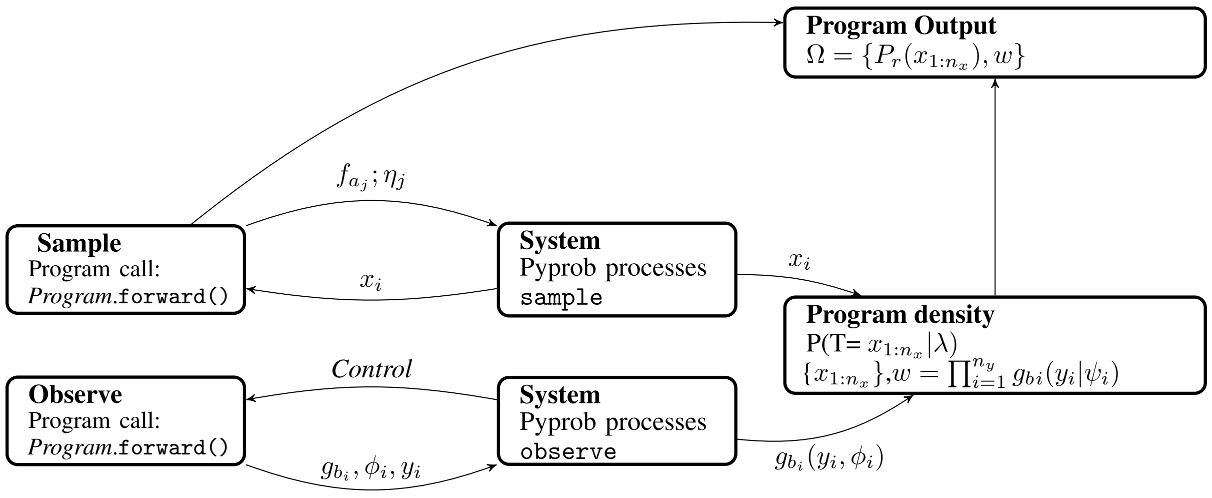
Notes on how to develop a formalism to perform inference in population based simulators.

1 Introduction

One challenge in performing inference in such models is the combinatorial increase in *paths* that can be taken within the given probabilistic model. This is because each member of the population has a bounded, but large number of decisions to make. In addition to this, each member of the population can interact with other members of the population. To model a single population member quantum-based methods leveraging linear superposition and entanglement provide an ability to explore multiple paths simultaneously maintaining the whole program, *state*, in one iteration. Unfortunately, analytically writing the functional form of the state is, in this instance, impossible. The usability of current classical inference methods in this context is also problematic. However, by combining hijacking and amortizing parts of the simulator, in particular rejection sampling loops, we can not only correctly perform inference in such models, as the rejection sampling loops are now replaced with learnt functions, but we can also perform inference much more efficiently. This is because the rejection sampling loops are now replaced with learnt functions which enables us to correctly construct the density and as they are amortized evaluation is cheap and only has to be performed once per call. This both reduces memory consumption and running time.

As updates are made sequentially MCMC methods typically are not-well suited this type of recursive estimation problem. As each time we get a new data point y_{t+1} we have to run new MCMC simulations for $P(x_{0:t_1}|y_{0:t+1})$, we also cannot directly reuse the samples $\{x^i_{0:t}\}$. Likewise, importance sampling is not well suited either, as we cannot reuse the samples and weights for a time t $\{x^i_{0:t}, \tilde{w}^i_t\}^N_{i=1}$ to sample from $(x_{0:t_1}|y_{0:t+1})$. However, as we can only run the simulator `forward()` we cannot return back to a previous state and re-run. So adapting existing inference techniques is the only viable option.

Preprint. Under review.



2 Limitations with different inference engines

2.1 Prior based sampling

In prior based sampling we take our existing program and look directly at the product of all the raw sample calls, denoted $f_{a_i}(x_i|\eta)$ and construct a proposal that is is dependent on those. In this sampling scheme our proposal is defined as:

$$q(x_{1:n_x}) = \begin{cases} \prod_{j=1}^{n_x} f_{a_i}(x_i|\eta) \texttt{ if } \mathcal{B}(x_{1:n_x)=1} \\ 0 \end{cases} \tag{1}$$

However, this makes use of no conditioning and is not sufficient for performing inference in complex, population-based simulations.

3 Non-prior Based Sampling

3.1 Importance Sampling

3.2 Sequential Monte Carlo

3.3 Random-walk Metropolis Hastings