Towards Heuristic Weights for Sequential Monte Carlo by Future Likelihood Estimates

Adam Jozefiak Department of Computer Science
University of British Columbia
Vancouver, BC, Canada
jozefiak@cs.ubc.ca

Abstract

We investigate the hypothesis that likelihood weights in the Sequential Monte Carlo algorithm can undervalue "good" proposals when extreme outliers are present in the observed data. To remedy this and account for long range dependencies, we investigate the likelihood weights of targeting the distributions $\{p(X_{1:n}|Y_{1:N})\}_{n=1}^{N}$. All terms of these likelihood weights are computable given $X_{1:n}$ and $Y_{1:N}$ except for $p(Y_{n+1:N}|X_{1:n}, Y_{1:n})$ and $p(Y_{n:N}|X_{1:n-1}, Y_{1:n-1})$. We then frame these future likelihoods as value functions and derive a recurrence relation that these value functions must satisfy. This recurrence relation gives us an objective for learning approximate value functions by stochastic gradient descent and nested Monte Carlo. However, we do not have any results that these approximate value functions are unbiased estimators, which remains an open question. Another open question is to provide an algorithm that employs these value functions as heuristic weights in the Sequential Monte Carlo algorithm. To test the original hypothesis, we perform inference in a Guassian linear dynamical system with an extreme outlier. However, we do not see a failure of the sequential Monte Carlo algorithm, nor, do the addition of the value functions as a heuristic improve the performance. It remains a question as to whether these value functions have any empirical benefit as heuristics for the Sequential Monte Carlo algorithm.

1 Introduction

In our setting, $X_{1:N}$ are latent random variables and $Y_{1:N}$ are observed random variables of a probabilistic program. For notational convenience, we define $X_{1:0}$ and $Y_{1:0}$ to be the empty sets.

In the Sequential Monte Carlo (SMC) algorithm the sequence of distributions $\{p(X_{1:n}|Y_{1:n})\}_{n=1}^{N}$ are targeted and successively approximated, where at the n^{th} time step the distribution $p(X_{1:n}|Y_{1:n})$ is approximated by a collection of particles. The result of performing inference by the SMC algorithm is an estimate of the distribution $p(X_{1:N}|Y_{1:N})$. At each time step, when going from a $\hat{p}(X_{1:n-1}|Y_{1:n-1})$ to $\hat{p}(X_{1:n}|Y_{1:n})$ we resample particles according to likelihood weights of the form

$$\frac{p(Y_n|X_{1:n}, Y_{1:n-1})p(X_n|X_{1:n-1}, Y_{1:n-1})}{q_\phi(X_n|X_{1:n-1}, Y_{1:N})}$$

where q_ϕ is a proposal distribution. We hypothesize that there may be future dependencies in the data where the importance weights of SMC may undervalue "good" particles in the situation where $p(Y_n|X_{1:n}, Y_{1:n-1})p(X_n|X_{1:n-1}, Y_{1:n-1})$ is extremely small relative to $q_\phi(X_n|X_{1:n-1}, Y_{1:N})$. This motivates us to ask the question, could we have access to, target, or approximate the sequence of distributions $\{p(X_{1:n}|Y_{1:N})\}_{n=1}^{N}$? Thereby, obtaining approximations to the likelihood weights

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

$$\frac{p(X_n|X_{1:n-1}, Y_{1:N})}{q_\phi(X_n|X_{1:n-1}, Y_{1:N})}$$

which no longer collapse for particles with high future likelihoods but low likelihood under the prior. In this paper, we begin by first unpacking the above likelihood weight, obtaining two sets of terms. The first set of terms are those that can be computed exactly given $X_{1:n}$ and $Y_{1:n}$. The second set of terms includes $p(Y_{n+1:N}|X_{1:n}, Y_{1:n})$ and $p(Y_{n:N}|X_{1:n-1}, Y_{1:n-1})$which cannot be computed given $X_{1:n}$ and $Y_{1:N}$ and must therefore be approximated.

Motivated by temporal difference techniques from reinforcement learning, we frame the sequence of future likelihoods $\{p(Y_{n+1:N}|X_{1:n}, Y_{1:n})\}_{n=1}^{N-1} \cup p(Y_{1:N})$ as value functions $\{V_n\}_{n=0}^{N-1}$ that satisfy a recurrence relation akin to a Bellman equation. We utilize this recurrence relation to place an objective for learning approximate value functions and show that the approximate value functions can be trained by stochastic gradient descent and nested Monte Carlo. Finally, we investigate whether the SMC algorithm can break down with outliers present in $Y_{1:N}$, and if so, does the addition of the value functions to the "standard" SMC likelihood weights, as a heuristic, improve the performance.

We remark that a similar approach of learning future likelihoods by temporal difference techniques for sequential data has been recently used by (Kim et al., 2020). Specifically, Kim et al. utilize an approximation of the future likelihoods in order to obtain a baseline that achieves a lower variance gradient for the task of variational inference in a Guassian linear state-space model.

2 Future Likelihoods as Value Functions

2.1 Defining the Value Functions

We define

$$V_0 := p(Y_{1:N})$$

and for each $n \in \{1, 2, \ldots, N-1\}$ we define

$$V_n(X_{1:n}) := p(Y_{n+1:N}|X_{1:n}, Y_{1:n})$$

Note, for consistency of notation, we may refer to V_0 as $V_0(X_{1:0})$. Note, $V_n(X_{1:n})$ is really a function of $X_{1:n}$ and $Y_{1:N}$, i.e $V_n(X_{1:n}, Y_{1:N})$, but we drop $Y_{1:N}$ when the observed random variables are apparent and fixed for a given inference task. Inspired by the reinforcement learning context, we can think of $\{V_n\}_{n=0}^{N-1}$ as value functions that capture the future likelihoods given the latent and observed random variables seen so far (Sutton, 1998). In particular $\{V_n\}_{n=0}^{N-1}$ satisfy the following recurrence relation, or rather, a Bellman equation in the reinforcement learning context.

$$\forall n \in \{2, 3, \ldots, N-1\}, \forall X_{1:n-1}, V_{n-1}(X_{1:n-1}) = \mathbb{E}_{X_n \sim p(X_n|X_{1:n-1}, Y_{1:n-1})}[p(Y_n|X_{1:n}, Y_{1:n-1})V_n(X_{1:n})]$$

and the base case

$$V_0 = \mathbb{E}_{X_1 \sim p(X_1)}[p(Y_1|X_1)V_1(X_1)]$$

and the final case

$$V_{N-1}(X_{1:N-1}) = \mathbb{E}_{X_N \sim p(X_N|X_{1:N-1}, Y_{1:N-1})}[p(Y_N|X_{1:N}, Y_{1:N-1})]$$

The proof is as follows for $n \geq 2$

$$
\begin{aligned}
V_{n-1}(X_{1:n-1}) &= p(Y_{n:N}|X_{1:n-1}, Y_{1:n-1}) \\
&= \int p(Y_{n:N}|X_{1:n}, Y_{1:n-1}) p(X_n|X_{1:n-1}, Y_{1:n-1} dX_n \\
&= \int p(Y_{n+1:N}|X_{1:n}, Y_{1:n}) p(Y_n|X_{1:n}, Y_{1:n-1}) p(X_n|X_{1:n-1}, Y_{1:n-1} dX_n \\
&= \mathbb{E}_{X_n \sim p(X_n|X_{1:n-1}, Y_{1:n-1})}[p(Y_n|X_{1:n}, Y_{1:n-1}) V_n(X_{1:n})]
\end{aligned}
$$

As for the base case the proof is

$$
\begin{aligned}
V_0 &= p(Y_{1:N}) \\
&= \int p(Y_{1:N}|X_1) p(X_1) dX_1 \\
&= \int p(Y_{2:N}|X_1, Y_1) p(Y_1|X_1) p(X_1) dX_1 \\
&= \mathbb{E}_{X_1 \sim p(X_1)}[p(Y_1|X_1) V_1(X_1)]
\end{aligned}
$$

As for the final case the proof is

$$
\begin{aligned}
V_{N-1}(X_{1:N-1}) &= p(Y_N|X_{1:N-1}, Y_{1:N-1}) \\
&= \int p(Y_N|X_{1:N}, Y_{1:N-1}) p(X_N|X_{1:N-1}, Y_{1:N-1}) dX_N \\
&= \mathbb{E}_{X_N \sim p(X_N|X_{1:N-1}, Y_{1:N-1})}[p(Y_N|X_{1:N}, Y_{1:N-1})]
\end{aligned}
$$

If $\{V_n\}_{n=0}^{N-1}$ are a family of functions that satisfy the above recurrence, then scaling $\{V_n\}_{n=0}^{N-1}$ by a scalar α will result in $\{\alpha V_n\}_{n=0}^{N-1}$ satisfying all but the final case of the recurrence. Therefore, it is paramount to include the final case in order to "fix" the scale V_{N-1} and ultimately the scale of $\{V_n\}_{n=0}^{N-1}$.

2.2 An Objective for a Value Function Approximator

We utilize the recurrence relation to obtain necessary conditions for a family of approximators $\{\hat{V}_{\psi,n}\}_{n=0}^{N-1}$ for $\{V_n\}_{n=0}^{N-1}$. However, it is intractable to satisfy this collection of equations for all choices of $X_{1:n-1}$ and so we utilize an expectation over choices of $X_{1:n-1}$ when creating an objective function from the recurrence relation. We can then approximate this objective by Monte Carlo, along with approximating its gradient by Monte Carlo. Our choice of objective function is motivated by temporal difference learning from reinforcement learning, in particular the mean square Bellman error. We obtain the following loss function for a family of estimators $\{\hat{V}_{\psi,n}\}_{n=0}^{N-1}$ for $\{V_n\}_{n=0}^{N-1}$:

$$
\begin{aligned}
\mathcal{L}(\psi) &:= \sum_{n=1}^{N-1} \mathbb{E}[(\hat{V}_{\psi,n-1}(X_{1:n-1}) - \mathbb{E}[p(Y_n|X_{1:n}, Y_{1:n-1}) \hat{V}_{\psi,n}(X_{1:n})])^2] \\
&+ \mathbb{E}[(\hat{V}_{\psi,N-1}(X_{1:N-1}) - \mathbb{E}[p(Y_N|X_{1:N}, Y_{1:N-1})])^2]
\end{aligned}
$$

Where the inner expectation is take with respect to $X_n \sim p(X_n|X_{1:n-1}, Y_{1:N})$ and where the outer expectation can be taken with respect to $p(X_{1:n-1})$ the prior, $p(X_{1:n-1}|Y_{1:n-1})$, or $p(X_{1:n-1}|Y_{1:N})$. The choice of distribution for the outer expectation is not unique and its choice is contingent on which distribution is tractable for training. We note that this objective requires nested Monte Carlo estimates and we point to Rainforth (Rainforth, 2018) as a reference.

2.3 Stochastic Gradient Descent on the Value Function Objective

We can learn approximators $\{\hat{V}_{\psi,n}\}_{n=0}^{N-1}$ by performing stochastic gradient descent with respect to $\mathcal{L}(\psi)$. Below, we show the gradient of $\mathcal{L}(\psi)$, omitting the details of the derivation

$$
\begin{aligned}
& \nabla_\psi \mathcal{L}(\psi) \\
= \; & \sum_{n=1}^{N-1} \mathbb{E}\big[(\hat{V}_{\psi,n-1}(X_{1:n-1}) - \mathbb{E}[p(Y_n|X_{1:n}, Y_{1:n-1})\hat{V}_{\psi,n}(X_{1:n})]) \\
& \qquad (\nabla_\psi \hat{V}_{\psi,n-1}(X_{1:n-1}) - \mathbb{E}[p(Y_n|X_{1:n}, Y_{1:n-1})\nabla_\psi \hat{V}_{\psi,n}(X_{1:n})])\big] \\
+ \; & \mathbb{E}[(\hat{V}_{\psi,N-1}(X_{1:N-1}) - \mathbb{E}[p(Y_N|X_{1:N}, Y_{1:N-1})])\nabla_\psi \hat{V}_{\psi,N-1}(X_{1:N-1}]
\end{aligned}
$$

Again, nested Monte Carlo estimation is required to approximate this gradient and, again, we point to Rainforth (Rainforth, 2018) as a reference for nested for Monte Carlo.

3 Derivation of Generalized Weights in SMC

Here we derive the importance weights for targeting $\{p(X_{1:n}|Y_{1:N})\}_{n=1}^N$. We assume that we have a proposal distribution, parameterized by ϕ, $q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})$ for each n.

3.1 General Time Step

We first consider the n^{th} time step, where $n \neq 1, N$, of our modified SMC-target algorithm at which point we are attempting to approximate $p(X_{1:n}|Y_{1:N})$ and we assume that we can sample from $p(X_{1:n-1}|Y_{1:N})$. Then we observe that:

$$
\begin{aligned}
p(X_{1:n}|Y_{1:N}) &= \frac{p(X_{1:n}|Y_{1:N})}{q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})} q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N}) \\
&= \frac{p(X_n|X_{1:n-1}, Y_{1:N})}{q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})} q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N}) p(X_{1:n-1}|Y_{1:N})
\end{aligned}
$$

Then, we can perform self-normalized importance sampling by sampling from the distribution $q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})p(X_{1:n-1}|Y_{1:N})$ and using the importance weight $W_n := \frac{p(X_n|X_{1:n-1}, Y_{1:N})}{q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})}$. We expand W_n further, breaking this weight into two types of terms: those that can be computed at the n^{th} time step of SMC and those that will need to be approximated by heuristic weights which will be learned before inference can be performed.

$$
\begin{aligned}
W_n &= \frac{p(X_n|X_{1:n-1}Y_{1:N})}{q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})} \\
&= \frac{p(X_{1:n}, Y_{1:N})}{p(X_{1:n-1}, Y_{1:N}) q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})} \\
&= \frac{p(Y_{1:N}|X_{1:n})p(X_{1:n})}{p(Y_{1:N}|X_{1:n-1})p(X_{1:n-1}) q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})} \\
&= \frac{p(Y_{1:N}|X_{1:n})p(X_n|X_{1:n-1})}{p(Y_{1:N}|X_{1:n-1}) q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})} \\
&= \frac{p(Y_{n+1:N}|X_{1:n}, Y_{1:n})p(Y_{1:n}|X_{1:n})p(X_n|X_{1:n-1})}{p(Y_{n:N}|X_{1:n-1}, Y_{1:n-1})p(Y_{1:n-1}|X_{1:n-1}) q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})}
\end{aligned}
$$

Then at the n^{th} time step of SMC we can compute the following terms of W_n:

$$
\frac{p(Y_{1:n}|X_{1:n})p(X_n|X_{1:n-1})}{p(Y_{1:n-1}|X_{1:n-1}) q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})}
$$

while the following terms must be approximated:

$$
\frac{p(Y_{n+1:N}|X_{1:n}, Y_{1:n})}{p(Y_{n:N}|X_{1:n-1}, Y_{1:n-1})}
$$

Therefore, using the value functions $\{V_n\}_{n=0}^{N-1}$, for $n = 2, \ldots, N-1$, the likelihood weights for SMC are

$$W_n = \frac{V_n(X_{1:n})p(Y_{1:n}|X_{1:n})p(X_n|X_{1:n-1})}{V_{n-1}(X_{1:n-1})p(Y_{1:n-1}|X_{1:n-1})q_\phi(X_{1:n}|X_{1:n-1}, Y_{1:N})}$$

When performing inference, we can replace $V_n(X_{1:n})$ and $V_{n-1}(X_{1:n-1})$ with learned estimators $\hat{V}_n(X_{1:n})$ and $\hat{V}_{n-1}(X_{1:n-1})$.

3.2 Initial Time Step

For the initial time step of our SMC algorithm we are attempting to approximate $p(X_1|Y_{1:N})$ and we first note that:

$$p(X_1|Y_{1:N}) \quad = \quad \frac{p(X_1|Y_{1:N})}{q_\phi(X_1|Y_{1:N})} q_\phi(X_1|Y_{1:N})$$

Then our likelihood weight is defined as $W_1 := \frac{p(X_1|Y_{1:N})}{q_\phi(X_1|Y_{1:N})}$ and we have that:

$$\begin{aligned} W_1 &= \frac{p(X_1|Y_{1:N})}{q_\phi(X_1|Y_{1:N})} \\ &= \frac{p(X_1, Y_{1:N})}{p(Y_{1:N})q_\phi(X_1|Y_{1:N})} \\ &= \frac{p(Y_{1:N}|X_1)p(X_1)}{p(Y_{1:N})q_\phi(X_1|Y_{1:N})} \\ &= \frac{p(Y_{2:N}|X_1, Y_1)p(Y_1|X_1)p(X_1)}{p(Y_{1:N})q_\phi(X_1|Y_{1:N})} \end{aligned}$$

Then at the initial time step of SMC we can compute the following terms of W_1:

$$\frac{p(Y_1|X_1)p(X_1)}{q_\phi(X_1|Y_{1:N})}$$

while the following terms must be approximated:

$$\frac{p(Y_{2:N}|X_1, Y_1)}{p(Y_{1:N})}$$

Therefore, using the value functions V_0 and V_1, the likelihood weight W_1 for SMC is

$$W_1 = \frac{V_1(X_1)p(Y_1|X_1)p(X_1)}{V_0 q_\phi(X_1|Y_{1:N})}$$

When performing inference, we can replace $V_1(X_1)$ and V_0 with learned estimators $\hat{V}_1(X_1)$ and $\hat{V}_0$.

3.3 Final Time Step

For the final time step of our SMC algorithm we are attempting to approximate $p(X_{1:N}|Y_{1:N})$. We continue from the second last line of the simplification of the likelihood weights of the general time step:

$$\begin{aligned} W_N &= \frac{p(Y_{1:N}|X_{1:N})p(X_N|X_{1:N-1})}{p(Y_{1:N}|X_{1:N-1})q_\phi(X_N|X_{1:N-1}, Y_{1:N})} \\ &= \frac{p(Y_{1:N}|X_{1:N})p(X_N|X_{1:N-1})}{p(Y_N|X_{1:N-1}, Y_{1:N-1})p(Y_{1:N-1}|X_{1:N-1})q_\phi(X_N|X_{1:N-1}, Y_{1:N})} \end{aligned}$$

Then at the final time step of SMC we can compute the following terms of W_N:

$$\frac{p(Y_{1:N}|X_{1:N})p(X_N|X_{1:N-1})}{p(Y_{1:N-1}|X_{1:N-1})q_\phi(X_N|X_{1:N-1}, Y_{1:N})}$$

while the following term must be approximated:

$$\frac{1}{p(Y_N|X_{1:N-1}, Y_{1:N-1})}$$

Therefore, using the value function V_{N-1}, the likelihood weight W_N for SMC is

$$W_N = \frac{p(Y_{1:N}|X_{1:N})p(X_N|X_{1:N-1})}{V_{N-1}(X_{1:N-1})p(Y_{1:N-1}|X_{1:N-1})q_\phi(X_N|X_{1:N-1}, Y_{1:N})}$$

When performing inference, we can replace $V_{N-1}(X_{1:N-1})$ with a learned estimator $\hat{V}_{N-1}(X_{1:N-1})$.

4 Limitations of our Estimators for Generalized-Target SMC

We do not have guarantees that our trained $\{\hat{V}_{\psi,n}\}_{n=0}^{N-1}$ are unbiased estimators of the value functions. Therefore, we cannot simply plug them into the weights for inferring the long-range distributions $\{p(X_{1:n}|Y_{1:N})\}_{n=1}^{N}$ without forfeiting the unbiasedness of the resulting estimates. We have two possible future directions: The first is to provide a tractable algorithm that learns unbiased estimates of the value functions. The second is to utilize the value function approximators as heuristic weights within SMC. A heuristic weight is a weight introduced in a program in order to help guide incremental inference algorithms, like SMC, while an equivalent cancelling weight is introduced at a later point of inference in order to keep the program's distribution invariant (Stuhlmüller et al., 2015).

5 Experiment

We tested the original hypothesis that the likelihood weights of SMC could result in poor weighting of particles, causing the SMC algorithm to break down. Our experiment was the inference task of a single dimension Gaussian linear dynamical system (LDS). The observed random variables were chosen so that all but a single outlier were a sensible observation from the prior distribution. This choice of a reasonable observation sequence with an extreme outlier was so that we check whether the likelihood weights at the time step of this observation would cause the SMC algorithm to break down.

Due to the model being a Gaussian LDS, we were able to compute the posterior exactly which we used as our proposal distribution q_ϕ (Bishop, 2006). Although we were able to train value function approximators, we were likewise able to compute exact value functions due to the model being a Gaussian LDS. In figure 1 we show the result of sampling 1000 particle sequences from the posterior using SMC. We observe that the SMC algorithm is not breaking down at the point of the outlier. Additionally, making the outlier more extreme did not produce differing outcomes.

Furthermore, we attempted to see if using value functions, both approximated and exact, incorrectly as value heuristic weights would result in any benefit to the SMC algorithm for this sequence of observed random variables. We repeated the previous experiment, except this time including the term $\frac{V_n(X_{1:n})}{V_{n-1}(X_{1:n-1})}$ to the likelihood weights of the SMC algorithm. However, this did not seem to improve the results, giving us an almost identical sample. In figure 2 we show the results of this second experiment.

6 Conclusion

In terms of the initial hypothesis, the likelihood weights of SMC can cause the SMC algorithm to fail when $p(Y_n|X_{1:n}, Y_{1:n-1})p(X_n|X_{1:n-1}, Y_{1:n-1})$ is relatively small compared to

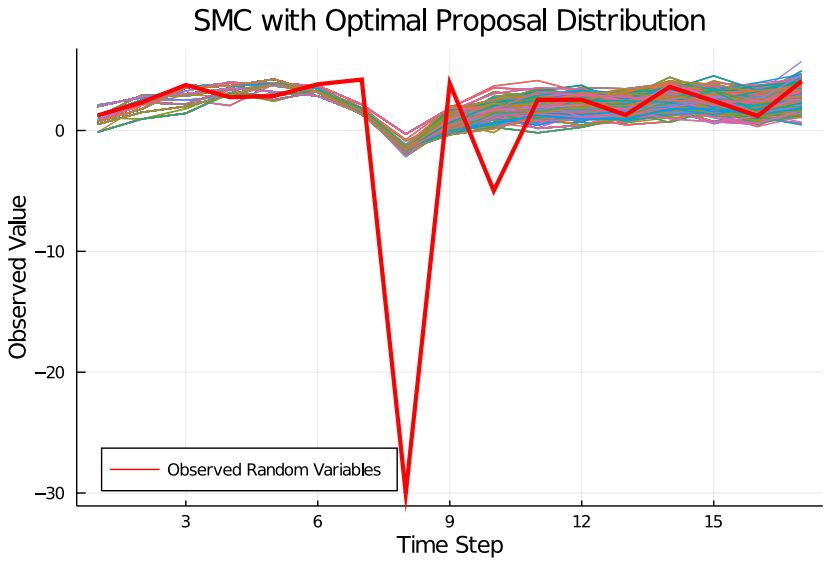

Figure 1: SMC Experiment Results

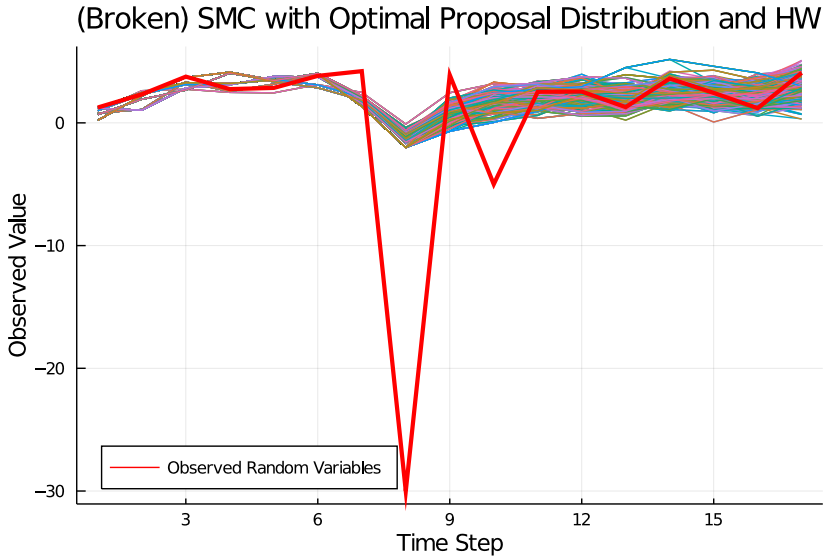

Figure 2: (Broken) Heuristic Weight SMC Experiment Results

$q_\phi(X_n|X_{1:n-1}, Y_{1:N})$, we did not find any evidence to support this hypothesis in our limited experiments. In fact, the addition of the exact and approximated value functions as (incorrect) heuristic weights did not improve the performance of the SMC algorithm. Further investigation is necessary to determine if the value functions form any useful heuristics.

On a more positive note, we showed that the likelihood weights of the "target" distributions $p(X_{1:n}|Y_{1:N})\}_{n=1}^N$ differ from the weights of $\{p(X_{1:n}|Y_{1:n})\}_{n=1}^N$ by a ratio of future likelihoods or value functions $\frac{V_n(X_1:n)}{V_{n-1}(X_{1:n-1})}$. Furthermore, we derived a necessary recurrence relation that these value functions must satisfy and from which we were able to place an objective on learning approximate value functions. We showed how to use this objective in order to train value function approximators with SGD and nested Monte Carlo estimation and in particular we were able to compute approximate value functions in our experiment.

On this front, further work remains. One open problem is to provide a tractable algorithm for training unbiased value function estimators with asymptotic guarantees. The other problem is to productively utilize the value function approximators as heuristic weights within the SMC algorithm.

References

[1] Bishop, Christopher M. Pattern recognition and machine learning. springer, 2006.

[2] Kim, Geon-Hyeong, et al. "Variational Inference for Sequential Data with Future Likelihood Estimates." International Conference on Machine Learning. PMLR, 2020.

[3] Rainforth, Tom. "Nesting probabilistic programs." arXiv preprint arXiv:1803.06328 (2018).

[4] Stuhlmüller, Andreas, et al. "Coarse-to-fine sequential monte carlo for probabilistic programs." arXiv preprint arXiv:1509.02962 (2015).

[5] Sutton, Richard S., and Andrew G. Barto. Introduction to reinforcement learning. Vol. 135. Cambridge: MIT press, 1998.