Amortized Rejection Sampling in Universal Probabilistic Programming

THE UNIVERSITY OF BRITISH COLUMBIA

Saeid Naderiparizi, Adam Ścibior, Andreas Munk, Mehrdad Ghadiri, Atılım Güneş Baydin,
Bradley Gram-Hansen, Christian Schroeder de Witt, Robert Zinkov, Philip Torr, Tom Rainforth, Yee Whye Teh, Frank Wood

Georgia Tech

UNIVERSITY OF OXFORD

TL;DR

- Rejection sampling loops are widely used in implementing probabilistic models e.g. sampling from constrained distributions.
- Unbounded loops (including rejection sampling) add major complexities to the inference task.
- We address the problem of efficient amortized importance-sampling-based inference, in particular, Inference Compilation (IC) [4] in such models.
- We show naive application of IC can produce estimators with unbounded variance.
- We propose an unbiased estimator for handling rejection sampling loop, prove its finite variance and implement it in pyprob [1; 2] in a way that requires minimal modifications to user code.

(a) Original program

- $x \sim p(x)$
- **for** $k \in \mathbb{N}^+$ **do**
  - $\boldsymbol{z}^k \sim p(\boldsymbol{z}|x)$
  - **if** $c(x, \boldsymbol{z}^k)$ **then**
    - $\boldsymbol{z} = \boldsymbol{z}^k$
    - **break**
- observe$(y, p(y|\boldsymbol{z}, x))$

(b) Inference compilation

- $x \sim q(x|y)$
- $w \leftarrow \frac{p(x)}{q(x|y)}$
- **for** $k \in \mathbb{N}^+$ **do**
  - $\boldsymbol{z}^k \sim q(\boldsymbol{z}|x, y)$
  - $w^k \leftarrow \frac{p(\boldsymbol{z}^k|x)}{q(\boldsymbol{z}^k|x,y)}$
  - $w \leftarrow w\, w^k$
  - **if** $c(x, \boldsymbol{z}^k)$ **then**
    - $\boldsymbol{z} = \boldsymbol{z}^k$
    - **break**
- $w \leftarrow w p(y|\boldsymbol{z}, x)$

(c) Equivalent to above

- $x \sim p(x)$
- $\boldsymbol{z} \sim p(\boldsymbol{z}|x, c(x, \boldsymbol{z}))$
- observe$(y, p(y|\boldsymbol{z}, x))$

(d) Our IS estimator

- $x \sim q(x|y)$
- $w \leftarrow \frac{p(x)}{q(x|y)}$
- $\boldsymbol{z} \sim q(\boldsymbol{z}|x, y, c(x, \boldsymbol{z}))$
- $w \leftarrow w \frac{p(\boldsymbol{z}|x,c(x,\boldsymbol{z}))}{q(\boldsymbol{z}|x,y,c(x,\boldsymbol{z}))}$
- $w \leftarrow w\, p(y|\boldsymbol{z}, x)$

IC weighting

$$w_{IC} = \frac{p(x)}{q(x|y)} p(y|x, z) \Pi_{k=1}^{L} w^k$$

Theorem: Under some mild conditions if the following holds then the variance of w_{IC} is infinite.

$$\mathbb{E}_{\boldsymbol{z} \sim q(\boldsymbol{z}|x,y)} \left[\frac{p(\boldsymbol{z}|x)^2}{q(\boldsymbol{z}|x, y)^2} (1 - p(A|x, \boldsymbol{z})) \right] \geq 1$$

where A is the event of $c(x, \boldsymbol{z})$ being satisfied.

Collapsed weighting

$$w_C = \frac{p(x)}{q(x|y)} \frac{p(\boldsymbol{z}|x, A)}{q(\boldsymbol{z}|x, y, A)} p(y|x, \boldsymbol{z})$$

- $\mathbb{E}[w_{IC}] = \mathbb{E}[w_C]$ but this weighting scheme does not introduce infinite variance to the estimator.
- Unfortunately, we cannot directly compute w_C

Our method (ARS)

$$w_C = \frac{p(x)}{q(x|y)} \frac{p(\boldsymbol{z}|x)}{q(\boldsymbol{z}|x, y)} p(y|x, \boldsymbol{z}) \frac{q(A|x, y)}{p(A|x)}$$

- $q(A|x, y)$ is a Bernoulli distribution. We get an unbiased estimate of its parameter via Monte Carlo.
- The number of samples $p(\boldsymbol{z}|x)$ until the first acceptance follows a geometric distribution with mean $\frac{1}{p(A|x)}$.
- We get an unbiased Monte Carlo estimates of $q(A|x, y)$ and $\frac{1}{p(A|x)}$.

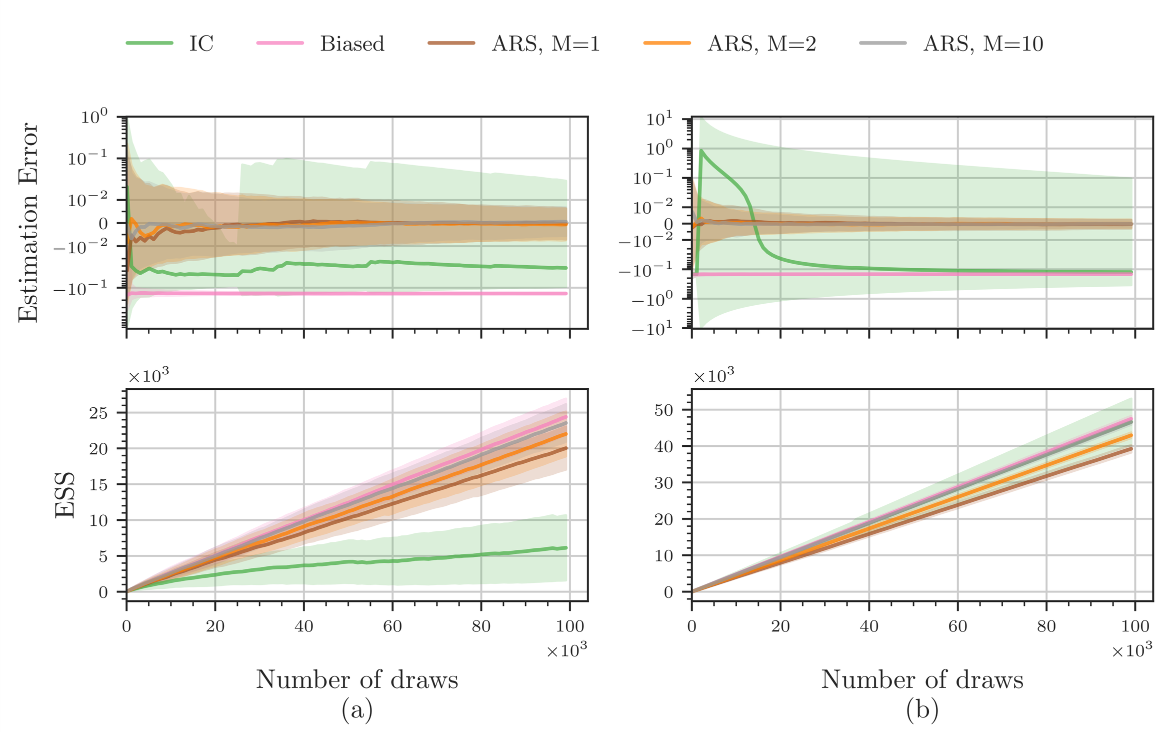
Marsaglia Experiment

Marsaglia polar method [5] is a pseudo-random number sampling method for generating samples from a Normal distribution.

$$\left.\begin{array}{l}(a, b) \sim \text{Unit circle} \\ \mu = a\sqrt{\frac{-2\log(a^2+b^2)}{a^2+b^2}}\end{array}\right\} \equiv \mu \sim \mathcal{N}(0, 1)$$

$$y_i \sim \mathcal{N}(mu, \sigma^2), \; i \in \{1, 2\}$$

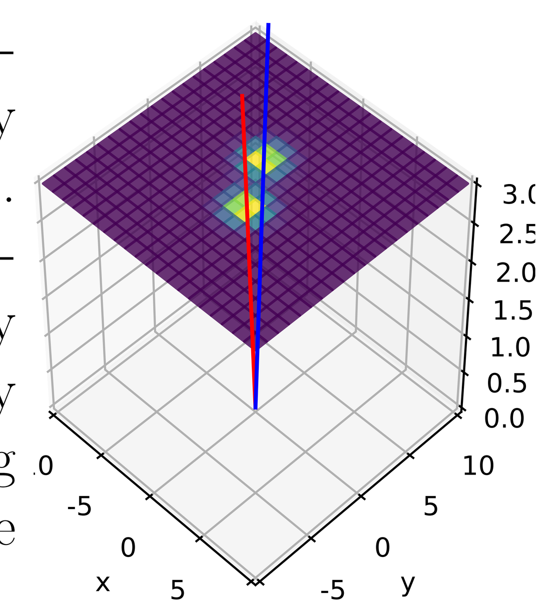
We target estimating $p(y_1, y_2)$.

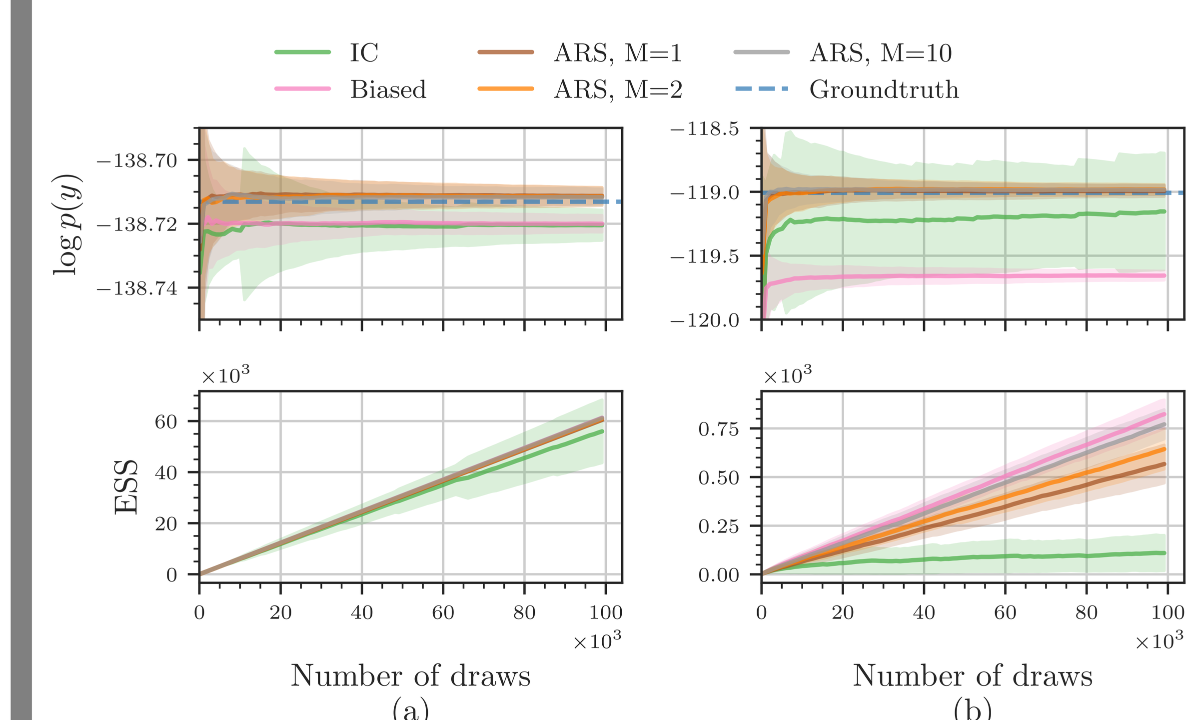


(a) $y_1 = y_2 = 0$ (b) $y_1 = y_2 = -1$

Mini-SHERPA Experiment

Mini-SHERPA is a simplified model of high-energy reactions of particles [3]. It simulates a particle decay event and returns noisy measurement of the energy dispatched by the resulting particles on the 2D surface of the detector.





Algorithm

- $x \sim q(x|y)$
- $w \leftarrow \frac{p(x)}{q(x|y)}$
- **for** $k \in \mathbb{N}^+$ **do**
  - $\boldsymbol{z}^k \sim q(\boldsymbol{z}|x, y)$
  - **if** $c(x, \boldsymbol{z}^k)$ **then**
    - $\boldsymbol{z} = \boldsymbol{z}^k$
    - **break**
- $w \leftarrow w \frac{p(\boldsymbol{z}|x)}{q(\boldsymbol{z}|x,y)}$
- $K \leftarrow 0$
- **for** $i \in 1, \ldots N$ **do**
  - $\boldsymbol{z}'_i \leftarrow q(\boldsymbol{z}|x, y)$
  - $K \leftarrow K + c(\boldsymbol{z}, x)$
- **for** $j \in 1, \ldots M$ **do**
  - **for** $l \in \mathbb{N}^+$ **do**
    - $\boldsymbol{z}''_{j,l} \leftarrow q(\boldsymbol{z}|x, y)$
    - **if** $c(x, \boldsymbol{z}''_{j,l})$ **then**
      - $T_j \leftarrow l$
      - **break**
- $T \leftarrow \frac{1}{M}\sum_{j=1}^{M} T_j$
- $w \leftarrow w \frac{KT}{N}$
- $w \leftarrow w\, p(y|\boldsymbol{z}, x)$

Implementation

We introduce two new functions to tag the beginning and end of rejection sampling loops.

Original

```
x = sample(P_x)
while True:

    z = sample(P_z(x))
    if c(x, z):

        break
observe(P_y(x,z), y)
return x, z
```

Annotated

```
x = sample(P_x)
while True:
    rs_start()
    z = sample(P_z(x))
    if c(x, z):
        rs_end()
        break
observe(P_y(x,z), y)
return x, z
```

References

[1] A. G. Baydin, L. Heinrich, W. Bhimji, B. Gram-Hansen, G. Louppe, L. Shao, K. Cranmer, F. Wood, et al. Efficient probabilistic inference in the quest for physics beyond the standard model. In *Thirty-second Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[2] A. G. Baydin, L. Shao, W. Bhimji, L. Heinrich, L. Meadows, J. Liu, A. Munk, S. Naderiparizi, B. Gram-Hansen, G. Louppe, et al. Etalumis: Bringing probabilistic programming to scientific simulators at scale. In *the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '19)*, 2019.

[3] T. Gleisberg, S. Höche, F. Krauss, M. Schönherr, S. Schumann, F. Siegert, and J. Winter. Event generation with sherpa 1.1. *Journal of High Energy Physics*, 2009(02):007, 2009.

[4] T. A. Le, A. G. Baydin, and F. Wood. Inference compilation and universal probabilistic programming. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1338–1348, Fort Lauderdale, FL, USA, 2017. PMLR.

[5] G. Marsaglia and T. A. Bray. A convenient method for generating normal variables. *SIAM review*, 6(3):260–264, 1964.