Accelerating Metropolis-Hastings with Lightweight Inference Compilation

Feynman Liang[1], Nimar Arora[2], Nazanin Tehrani[2], Yucen Li[2], Michael Tingley[2], Erik Meijer[2]

[1]UC Berkeley, [2]Facebook

Summary

Accelerate lightweight Metropolis-Hastings [1] by using neural network approximations to Gibbs sampling distributions. Unlike prior work [2], lightweight inference compilation (LIC) leverages Markov blanket structure provided by its host probabilistic programming language (PPL) to inform its neural network architectures. As a result, LIC's proposers have less parameters, greater robustness to nuisance random variables, and improved posterior sampling in a Bayesian logistic regression and n-schools inference application

Intuition for inference compilation

Leverage generative sampling (i.e. running the probabilistic program) for amortized proposers $q(x; \phi(x, y))$

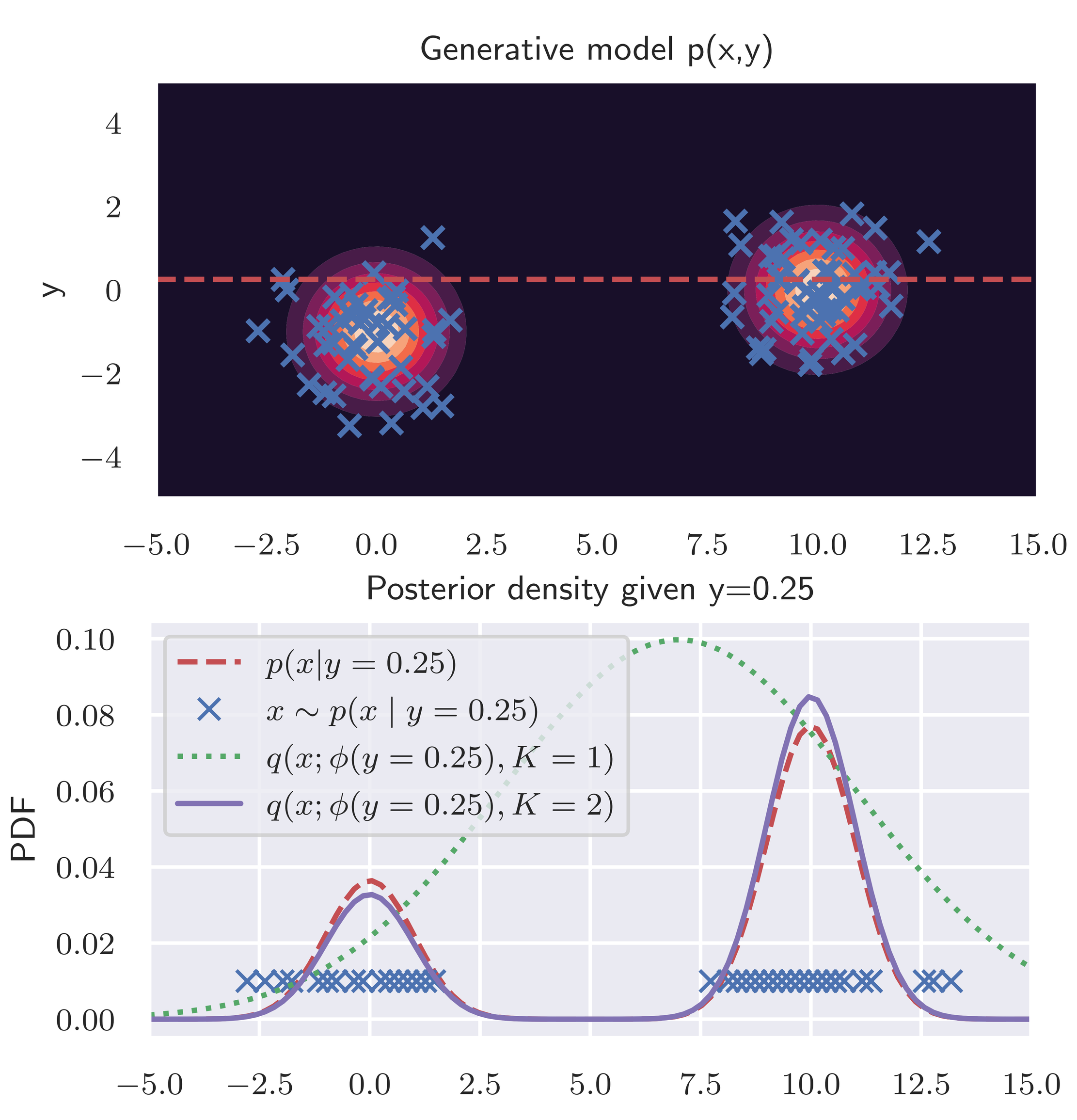

Figure 1: Generative samples $(x, y) \sim p(x, y)$ yield information about the posterior $p(x \mid y)$ for multiple values of y, enabling amortized inference

Presented at the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego.

LIC's architecture and training

- Run probabilistic program forwards to generate n joint samples $(x, y) \sim p(x, y)$
- Minimize Monte-Carlo estimate of inclusive KL-divergence

$$\mathbb{E}_{y \sim p(y)} KL(p(x \mid y), q(x)) \approx \frac{1}{n} \sum_{i=1}^{n} \log \frac{p(x_i \mid y_i)}{q(x_i; \phi)} \propto \frac{1}{n} \sum_{i=1}^{n} \log \frac{p(x_i, y_i)}{q(x_i; \phi)}$$

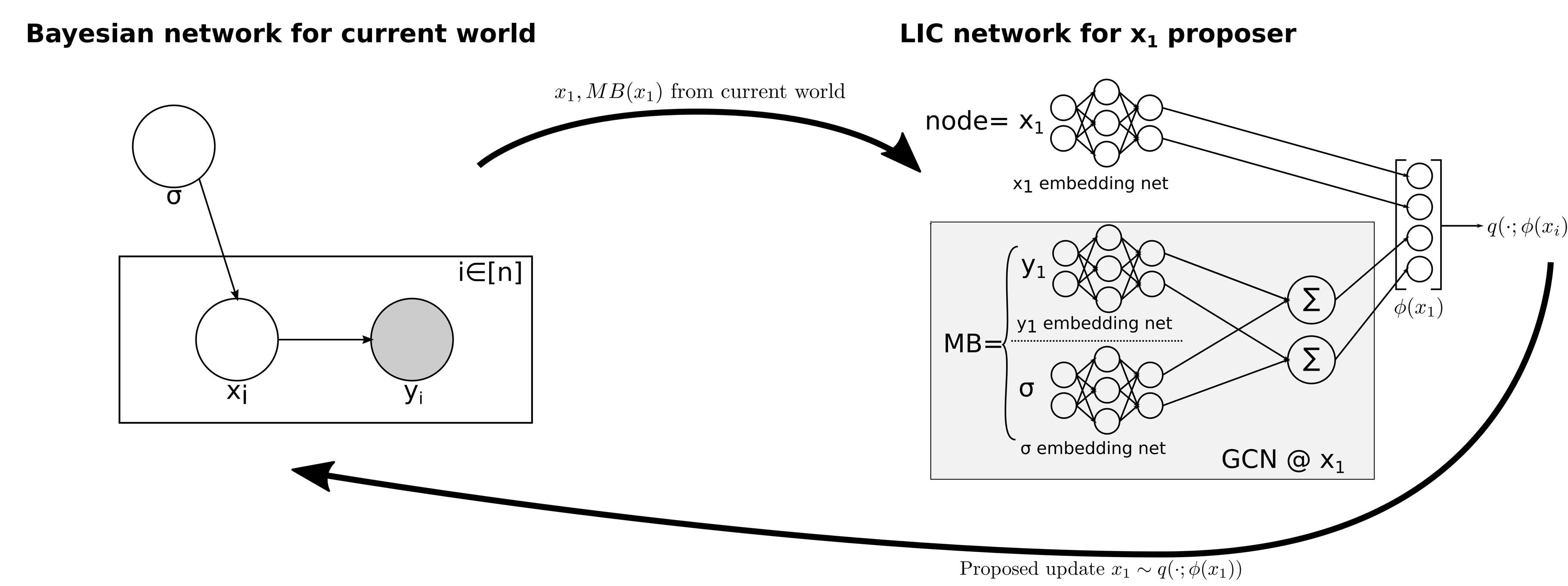

Figure 2: LIC's architecture for a proposal distribution over latent x_i conditioned on y_i and with parent σ

Bayesian logistic regression and n-schools

Bayesian logistic regression (BLR)

$$\vec{\beta} \sim \mathcal{N}_{d+1}(\vec{0}_{d+1}, \text{diag}(10, 2.5\vec{1}_d))$$
$$y_i \mid \vec{x}_i \overset{\text{iid}}{\sim} \text{Bernoulli}(\sigma(\vec{\beta}^\top \vec{x}_i))$$
$$\sigma(t) = (1 + e^{-t})^{-1}$$

n-schools, a generalization of 8-schools [3] used for Bayesian meta-analysis at a large internet company

$$\beta_0 \sim \text{StudentT}(3, 0, 10)$$
$$\tau_i \sim \text{HalfCauchy}(\sigma_i) \qquad \text{for } i \in [\text{district}, \text{state}, \text{type}]$$
$$\beta_{i,j} \sim \mathcal{N}(0, \tau_i) \qquad \text{for } i \in [\text{district}, \text{state}, \text{type}], j \in [n_i]$$
$$y_k \sim \mathcal{N}(\beta_0 + \sum_i \beta_{i,j_k}, \sigma_k)$$

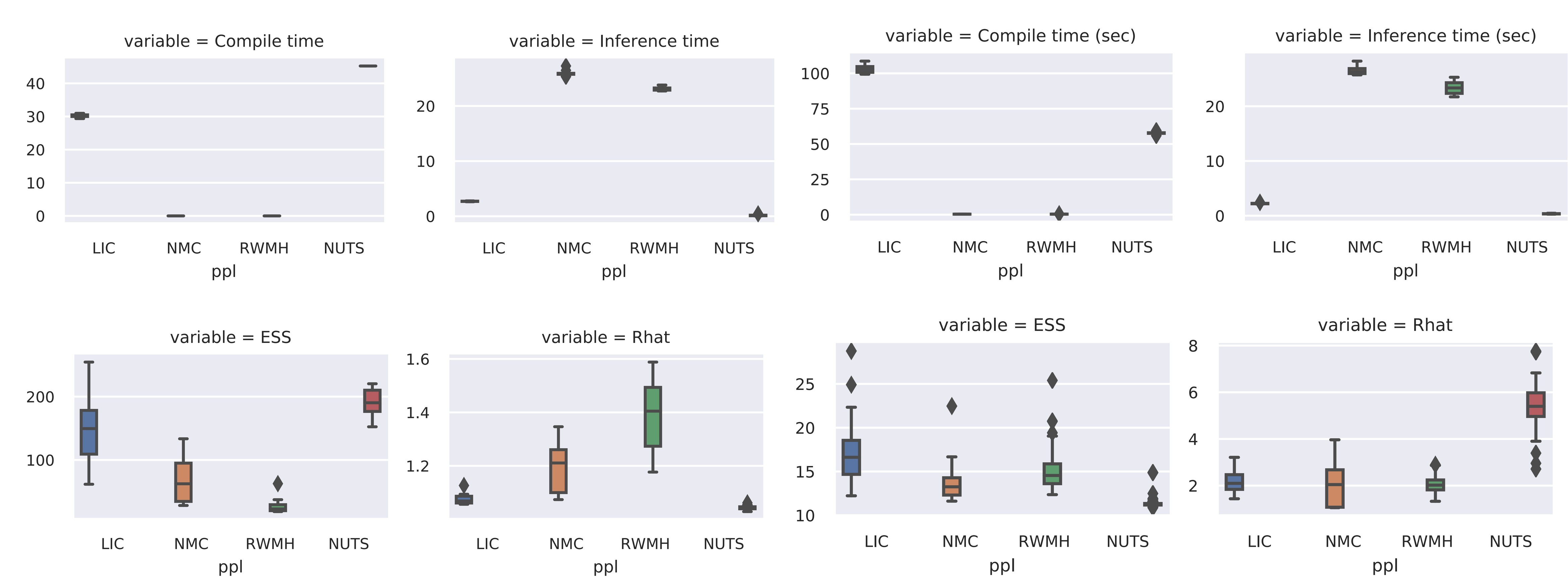

Figure 3: Empirical results for Bayesian logistic regression (left) and n-schools (right), compile time (seconds, lower is better, only applicable to LIC and NUTS), inference time (seconds, lower is better), effective sample size (higher is better), and $\hat{R}$ [4] (lower is better)

Robustness to nuisance parameters

A version of Program 1 from [5] illustrates LIC's improved robustness to nuisance parameters compared to [6]:

```
x = sample(Normal(0, 10))
for _ in range(100):
  nuisance = sample(Normal(0, 10))
y = sample(Normal(0, 10))
observe(obs**2,
  likelihood=Normal( x**2 + y**2, 0.1))
```

	# params	compile time	ESS
LIC	3,358	44 sec.	49.75
PyProb	21,952	472 sec.	10.99

References

[1] Wingate et. al. Lightweight implementations of probabilistic programming languages via transformational compilation. In *AISTATS*, 2011.

[2] Le et. al. Inference compilation and universal probabilistic programming. In *AISTATS*, 2017.

[3] Rubin. Estimation in parallel randomized experiments. *Journal of Educational Statistics*, 1981.

[4] Brooks et. al. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 1998.

[5] Harvey et. al. Attention for inference compilation. *arXiv preprint arXiv:1910.11961*, 2019.

[6] PyProb. `https://github.com/pyprob/pyprob`, 2020.

Acknowledgements

Feynman Liang is supported by a NPSC Graduate Fellowship. This work was done during an internship at Facebook.

Contact Information

- github.com/feynmanliang/lightweight-inference-compilation
- feynman@berkeley.edu