Accelerating Metropolis-Hastings with Lightweight Inference Compilation

Feynman Liang, Nim Arora, Nazanin Tehrani, Yucen Li, Michael Tingley, Erik Meijer

November 10, 2020

Facebook Probability, Facebook AI Infrastructure, UC Berkeley

Table of contents

1. Background
   - Probabilistic Programming
   - Bayesian Inference
2. Inference compilation in declarative PPLs
   - SIS in imperative PPLs
   - Lightweight Inference Compilation for MCMC
3. Results
4. Future directions

Background

Two competing philosophies

[van de Meent et al., 2018] To build machines that can reason, random variables and probabilstic calculations are:

Two competing philosophies

[van de Meent et al., 2018] To build machines that can reason, random variables and probabilstic calculations are:

Probabilistic ML

An engineering requirement [Tenenbaum et al., 2011, Ghahramani, 2015]

Two competing philosophies

[van de Meent et al., 2018] To build machines that can reason, random variables and probabilstic calculations are:

Probabilistic ML

An engineering requirement [Tenenbaum et al., 2011, Ghahramani, 2015]

Deep Learning

Irrelevant [LeCun et al., 2015, Goodfellow et al., 2016]

Probabilistic programming languages (PPLs)

> *Just as programming beyond the simplest algorithms requires tools for abstraction and composition, complex probabilistic modeling requires new progress in model representation—probabilistic programming languages.*

[Goodman, 2013]

Abstractions over deterministic computations

Low Level Assembly

```
mov   dx, msg
; ah=9 - "print string" sub-function
mov   ah, 9
int   0x21

"terminate␣program" sub-function
mov   ah, 0x4c
int   0x21

msg
db  'Hello,␣World!', 0x0d, 0x0a, '$'
```

Abstractions over deterministic computations

Low Level Assembly

```
mov   dx, msg
; ah=9 - "print string" sub-function
mov   ah, 9
int   0x21

"terminate␣program" sub-function
mov   ah, 0x4c
int   0x21

msg
db  'Hello,␣World!', 0x0d, 0x0a, '$'
```

High Level Python

```
print("Hello,␣World!")
```

Abstractions over probabilistic computations

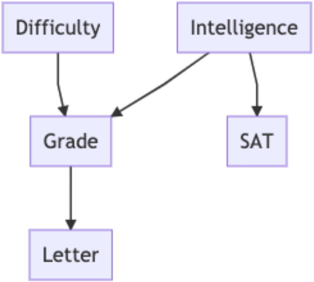


Figure 1:
[Koller and Friedman, 2009]

Abstractions over probabilistic computations

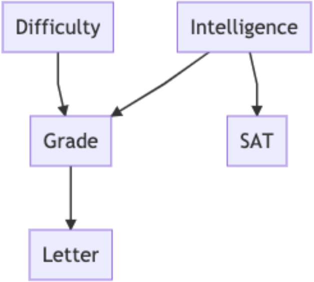
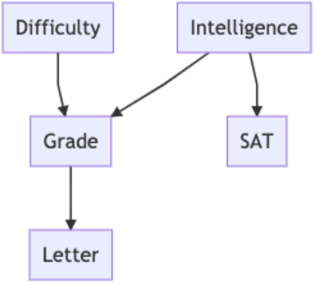


Figure 1:
[Koller and Friedman, 2009]

```
d ~ Bernoulli
i ~ Normal
g ~ g(d, i)
s ~ s(i)
l ~ Multinomial(g)
```

Abstractions over probabilistic computations



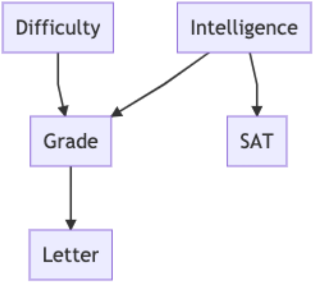

```
d ~ Bernoulli
i ~ Normal
g ~ g(d, i)
s ~ s(i)
l ~ Multinomial(g)
```

Figure 1:
[Koller and Friedman, 2009]

Generative model:

$$P(D, I, G, S, L) = P(D)P(I)P(G \mid D, I)P(S \mid I)P(G \mid L)$$

Abstractions over probabilistic computations

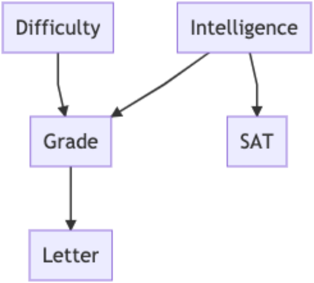



```
d ~ Bernoulli
i ~ Normal
g ~ g(d, i)
s ~ s(i)
l ~ Multinomial(g)
```

Figure 1:
[Koller and Friedman, 2009]

Question : Given a student's recommendation *letter* and *SAT* score, what should I expect their *intelligence* to be?

Abstractions over probabilistic computations




```
d ~ Bernoulli
i ~ Normal
g ~ g(d, i)
s ~ s(i)
l ~ Multinomial(g)
```

Figure 1:
[Koller and Friedman, 2009]

Question : Given a student's recommendation *letter* and *SAT* score, what should I expect their *intelligence* to be?

PPL Query : `infer(i, {l=Good, s=800})`

Bayesian Inference Basics

Latent Variables X

Observed Variables Y

Prior $P(X)$

Likelihood $P(Y \mid X)$

Bayesian Inference Basics

Latent Variables X

Observed Variables Y

Prior $P(X)$

Likelihood $P(Y \mid X)$

Goal: Approximate the posterior $P(X \mid Y)$

Bayesian Inference Basics

Goal: Approximate the posterior $P(X \mid Y)$

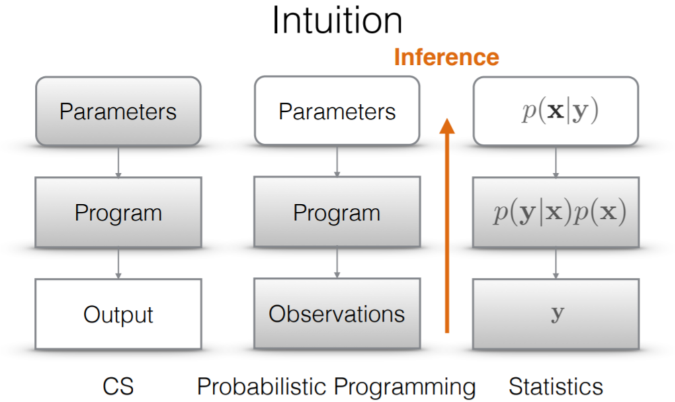


Figure 2: [van de Meent et al., 2018]

Bayesian Inference Basics

Goal: Approximate the posterior $P(X \mid Y)$

X	Y
intelligence	letter and grade
scene description	image
simulation	simulator output
program source code	program return value
policy prior and world simulator	rewards
cognitive decision making process	observed behavior

Table 1: [van de Meent et al., 2018]

Why only approximate?

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)} = \frac{P(Y \mid X)P(X)}{\underbrace{\int_X P(Y \mid X)P(X)dX}_{=:Z(Y)}}$$

Why only approximate?

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)} = \frac{P(Y \mid X)P(X)}{\underbrace{\int_X P(Y \mid X)P(X)dX}_{=:Z(Y)}}$$

Denominator $Z(Y)$ (i.e. partition function, marginal likelihood $P(Y)$) high-dimensional integral, known only for a small family of *conjugate* prior/likelihood pairs

How to approximate?

Variational Inference

Monte Carlo

How to approximate?

Variational Inference

Let q_ϕ be a tractable parametric family (e.g. Gaussian mean-field $q_\phi(X) = \prod_{i=1}^{d} N(X_i \mid \phi_{1,i}, \phi_{2,i})$)

Monte Carlo

How to approximate?

Variational Inference

Let q_ϕ be a tractable parametric family (e.g. Gaussian mean-field $q_\phi(X) = \prod_{i=1}^{d} N(X_i \mid \phi_{1,i}, \phi_{2,i})$)

$$\arg\min_\phi KL(q_\phi(X) \mid P(X \mid Y))$$

$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(X \mid Y)}\right]$$

$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(Y \mid X) P(X)}\right]$$

Monte Carlo

How to approximate?

Variational Inference

Let q_ϕ be a tractable parametric family (e.g. Gaussian mean-field $q_\phi(X) = \prod_{i=1}^{d} N(X_i \mid \phi_{1,i}, \phi_{2,i})$)

$$\arg\min_\phi KL(q_\phi(X) \mid P(X \mid Y))$$

$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(X \mid Y)}\right]$$

$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(Y \mid X)P(X)}\right]$$

Monte Carlo

Sample $X_i \overset{\text{iid}}{\sim} P(X \mid Y)$. Then

How to approximate?

Variational Inference

Let q_ϕ be a tractable parametric family (e.g. Gaussian mean-field $q_\phi(X) = \prod_{i=1}^d N(X_i \mid \phi_{1,i}, \phi_{2,i})$)

$$\arg\min_\phi KL(q_\phi(X) \mid P(X \mid Y))$$

$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(X \mid Y)}\right]$$

$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(Y \mid X)P(X)}\right]$$

Monte Carlo

Sample $X_i \overset{\text{iid}}{\sim} P(X \mid Y)$. Then

$$\mathbb{E}[g(X)|Y]$$

$$= \int g(X) \cdot P(X \mid Y) dX$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} g(X_i)$$

How to approximate?

Variational Inference

Let q_ϕ be a tractable parametric family (e.g. Gaussian mean-field $q_\phi(X) = \prod_{i=1}^{d} N(X_i \mid \phi_{1,i}, \phi_{2,i}))$

$$\arg\min_\phi KL(q_\phi(X) \mid P(X \mid Y))$$
$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(X \mid Y)}\right]$$
$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(Y \mid X)P(X)}\right]$$

Monte Carlo

Sample $X_i \overset{\text{iid}}{\sim} P(X \mid Y)$. Then

$$\mathbb{E}[g(X)|Y]$$
$$= \int g(X) \cdot P(X \mid Y) dX$$
$$\approx \frac{1}{N} \sum_{n=1}^{N} g(X_i)$$

How to approximate?

Variational Inference

Let q_ϕ be a tractable parametric family (e.g. Gaussian mean-field $q_\phi(X) = \prod_{i=1}^{d} N(X_i \mid \phi_{1,i}, \phi_{2,i})$)

$$\arg\min_\phi KL(q_\phi(X) \mid P(X \mid Y))$$

$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(X \mid Y)}\right]$$

$$= \arg\max_\phi E_{q_\phi}\left[\frac{\log q_\phi(X)}{\log P(Y \mid X)P(X)}\right]$$

Monte Carlo

Sample $X_i \overset{\text{iid}}{\sim} P(X \mid Y)$. Then

$$\mathbb{E}[g(X)|Y]$$

$$= \int g(X) \cdot P(X \mid Y)dX$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} g(X_i)$$

Inference compilation in declarative PPLs

Imperative vs Declarative PPLs

Imperative: Evaluation-based, samples (linear) execution traces (Pyro, Church, WebPPL)

Imperative vs Declarative PPLs

Imperative: Evaluation-based, samples (linear) execution traces (Pyro, Church, WebPPL)

```
(begin
    (define geometric
        (lambda (p)
        (if (flip p)
        1
        (+ 1 (geometric p)))))
    (geometric .7))
```

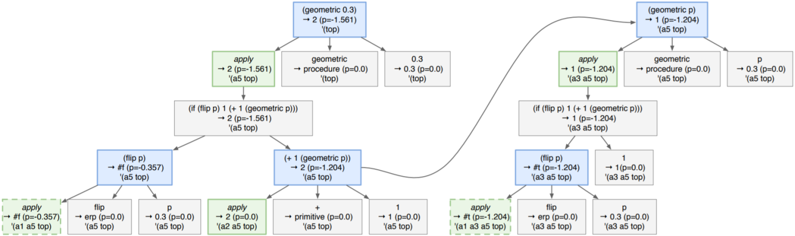


Figure 2: [Wingate et al., 2011]

Sequential importance sampling (SIS)

If we can sample $X_i \sim q$ from proposal distribution q,

Sequential importance sampling (SIS)

If we can sample $X_i \sim q$ from proposal distribution q,

$$\mathbb{E}_{P(X|Y)}[g(X)] = \mathbb{E}_{P(X|Y)}\left[\frac{q(X)}{q(X)}g(X)\right] = \frac{\mathbb{E}_q\left[\frac{P(X,Y)}{q(X)}g(X)\right]}{P(Y)}$$

$$\approx \frac{1}{N}\sum_i^N \frac{\frac{P(X_i,Y)}{q(X_i)}}{P(Y)}g(X_i) \approx \sum_i^N \frac{\frac{P(X_i,Y)}{q(X_i)}}{\sum_i^N \frac{P(X_i,Y)}{q(X_i)}}g(X_i)$$

Sequential importance sampling (SIS)

If we can sample $X_i \sim q$ from proposal distribution q,

$$\mathbb{E}_{P(X|Y)}[g(X)] = \mathbb{E}_{P(X|Y)}\left[\frac{q(X)}{q(X)}g(X)\right] = \frac{\mathbb{E}_q\left[\frac{P(X,Y)}{q(X)}g(X)\right]}{P(Y)}$$

$$\approx \frac{1}{N}\sum_i^N \frac{\frac{P(X_i,Y)}{q(X_i)}}{P(Y)}g(X_i) \approx \sum_i^N \frac{\frac{P(X_i,Y)}{q(X_i)}}{\sum_i^N \frac{P(X_i,Y)}{q(X_i)}}g(X_i)$$

Convergence rate $\mathrm{Var}_q\left[\frac{P(X,Y)}{q(X)}g(X)\right]^{-1/2}$ [Yuan and Druzdzel, 2007]

SIS of execution traces

1. Execute the probabilistic program forwards
2. At each stochastic variable (i.e. `sample` statement), sample from proposer $q(\cdot)$ and assign value
3. At each observed random variable (i.e. `observe` statement), multiply likelihood into trace's importance weight

SIS of execution traces

1. Execute the probabilistic program forwards
2. At each stochastic variable (i.e. `sample` statement), sample from proposer $q(\cdot)$ and assign value
3. At each observed random variable (i.e. `observe` statement), multiply likelihood into trace's importance weight

Problem: Myopic choices from sampling q early in the trace may result in low importance weights later. Would like q to account for observations Y.

Constructing a proposal distribution

How to choose q?

Constructing a proposal distribution

How to choose q?

Likelihood-Weighting [Norvig and Intelligence, 2002]:
$q(X) = P(X)$

Constructing a proposal distribution

How to choose q?

Likelihood-Weighting [Norvig and Intelligence, 2002]: $q(X) = P(X)$

Direct sampling: $q(X) = P(X \mid Y)$, optimal

Constructing a proposal distribution

How to choose q?

Likelihood-Weighting [Norvig and Intelligence, 2002]: $q(X) = P(X)$

Direct sampling: $q(X) = P(X \mid Y)$, optimal

Key Idea: Exploit access to $P(X, Y)$ to build a proposer q "close" to $P(X \mid Y)$?

Trace-based inference compilation (IC)

- Construct DNN with parameters ϕ mapping observations Y (amortized inference, [Goodman, 2013]) and execution prefix to proposal distribution $q_\phi(\cdot \mid Y)$
- Train q_ϕ against forward samples from the probabilistic program $p(x, y)$ (inference compilation)

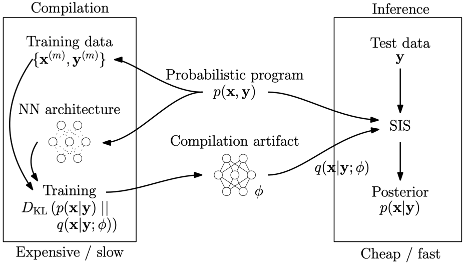
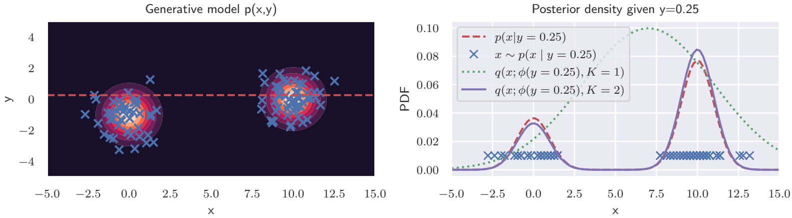


Figure 3: [Le et al., 2017]

Intuition for inference compilation



Trace-based inference compilation (IC)

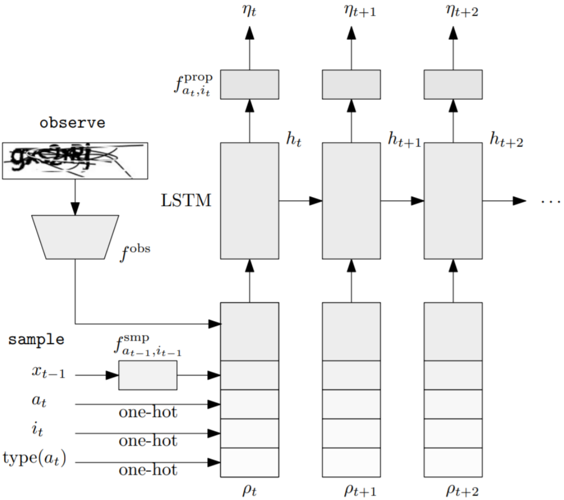


Figure 4: [Le et al., 2017]

Sensitivity to nuisance random variables

```
def magnitude(obs, M):
    x = sample(Normal (0, 10))
    [sample(Normal (0 ,10)) for _ in range(M)] # extend trace with nuisance
    y = sample(Normal (0 ,10))
    observe(obs**2, Likelihood=Normal(x**2 + y**2, 0.1)0
    return x, y
```

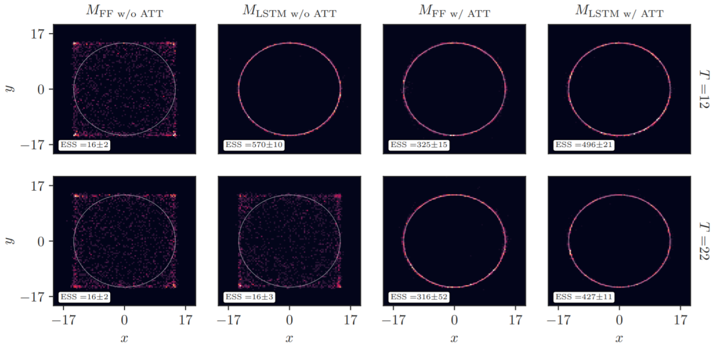


Figure 5: [Harvey et al., 2019]

Imperative vs Declarative PPLs

Declarative: Graph-based, samples instantiated graphical models (i.e. worlds) (BUGS, BLOG, Stan, beanmachine)

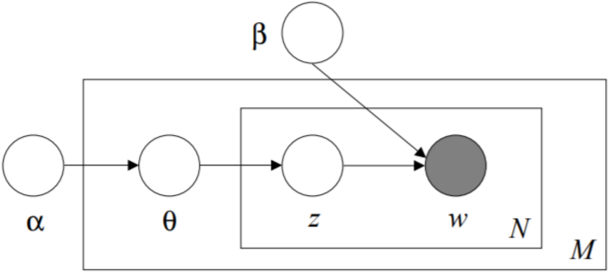


Figure 6: [Blei et al., 2003]

Key Idea: Markov blanket MB(X_i) available in declarative PPL

MCMC sampling of graphical models

Metropolis-within-Gibbs / Lightweight MH ([Wingate et al., 2011]):

- Fix observed nodes Y to their values.
- Repeat:
  - Pick single random unobserved node X_i
  - Sample proposal $q(X_i)$ to propose new value
  - Accept with probability α and revert otherwise

MCMC sampling of graphical models

Metropolis-within-Gibbs / Lightweight MH ([Wingate et al., 2011]):

- Fix observed nodes Y to their values.
- Repeat:
    - Pick single random unobserved node X_i
    - Sample proposal $q(X_i)$ to propose new value
    - Accept with probability α and revert otherwise

Theorem ([Hastings, 1970])
With appropriately chosen α, the above algorithm yields a Markov Chain with the posterior as the invariant distribution.

MH proposal distributions

Different $q(\cdot) \implies$ different MCMC algorithms

- Random walk MH: $q(\cdot)$ isotropic Gaussian

MH proposal distributions

Different $q(\cdot) \implies$ different MCMC algorithms

- Random walk MH: $q(\cdot)$ isotropic Gaussian
- Newtonian Monte Carlo [Arora et al., 2020]: $q(\cdot) =$ Gaussian with empirical Fisher information

MH proposal distributions

Different $q(\cdot) \implies$ different MCMC algorithms

- Random walk MH: $q(\cdot)$ isotropic Gaussian
- Newtonian Monte Carlo [Arora et al., 2020]: $q(\cdot) =$ Gaussian with empirical Fisher information
- Hamiltonian Monte Carlo: $q(\cdot)$ integrates iso-Hamiltonian system

MH proposal distributions

Different $q(\cdot) \implies$ different MCMC algorithms

- Random walk MH: $q(\cdot)$ isotropic Gaussian
- Newtonian Monte Carlo [Arora et al., 2020]: $q(\cdot) =$ Gaussian with empirical Fisher information
- Hamiltonian Monte Carlo: $q(\cdot)$ integrates iso-Hamiltonian system
- Lightweight Inference Compilation: $q(\cdot \mid \mathrm{MB}(X_i))$ a neural network function of the current Markov Blanket

MH proposal distributions

Different $q(\cdot) \implies$ different MCMC algorithms

- Random walk MH: $q(\cdot)$ isotropic Gaussian
- Newtonian Monte Carlo [Arora et al., 2020]: $q(\cdot) =$ Gaussian with empirical Fisher information
- Hamiltonian Monte Carlo: $q(\cdot)$ integrates iso-Hamiltonian system
- Lightweight Inference Compilation: $q(\cdot \mid \mathrm{MB}(X_i))$ a neural network function of the current Markov Blanket

Theorem ([Pearl, 1987])
Gibbs distributions $P(X_i \mid X_i^c) = P(X_i \mid MB(X_i))$ are optimal proposers $q(\cdot)$

MH proposal distributions

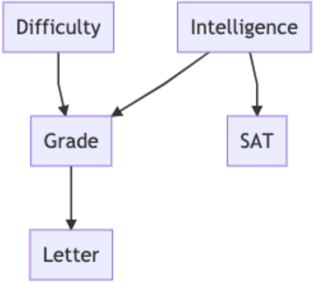
Different $q(\cdot) \implies$ different MCMC algorithms

- Random walk MH: $q(\cdot)$ isotropic Gaussian
- Newtonian Monte Carlo [Arora et al., 2020]: $q(\cdot) =$ Gaussian with empirical Fisher information
- Hamiltonian Monte Carlo: $q(\cdot)$ integrates iso-Hamiltonian system
- Lightweight Inference Compilation: $q(\cdot \mid \mathrm{MB}(X_i))$ a neural network function of the current Markov Blanket

Theorem ([Pearl, 1987])
Gibbs distributions $P(X_i \mid X_i^c) = P(X_i \mid MB(X_i))$ are optimal proposers $q(\cdot)$

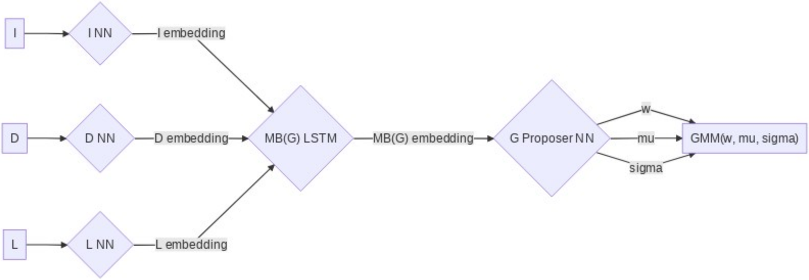
$\therefore$ $\mathrm{MB}(X_i)$ is the minimal sufficient inputs for generating proposal distribution

LIC artifacts for example student network

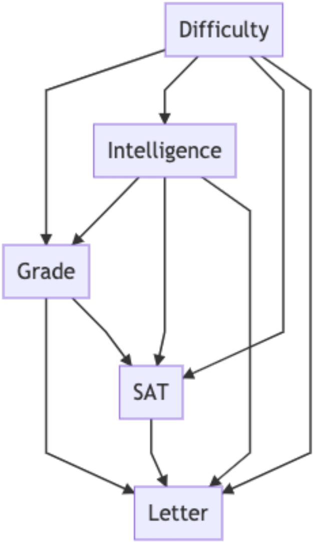


$$q(d|g,i)$$
$$q(i|g,d,s)$$
$$q(g|i,d,l)$$
$$q(s|i)$$
$$q(l|g)$$

Example LIC proposer for grade



Compare against SIS IC's (non-minimal) proposer



$q(d|\text{observations})$

$q(i|d, \text{observations})$

$q(g|i, d, \text{observations})$

$q(s|g, i, d, \text{observations})$

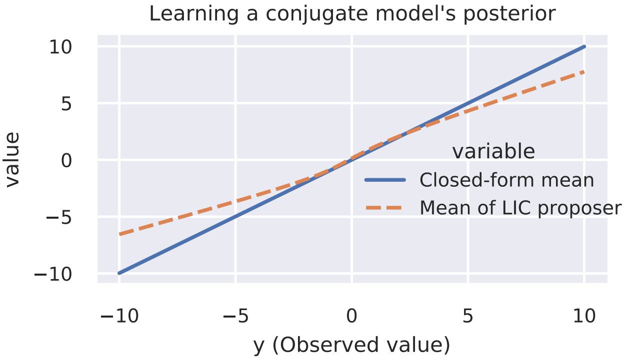
$q(l|s, g, i, d, \text{observations})$

Results

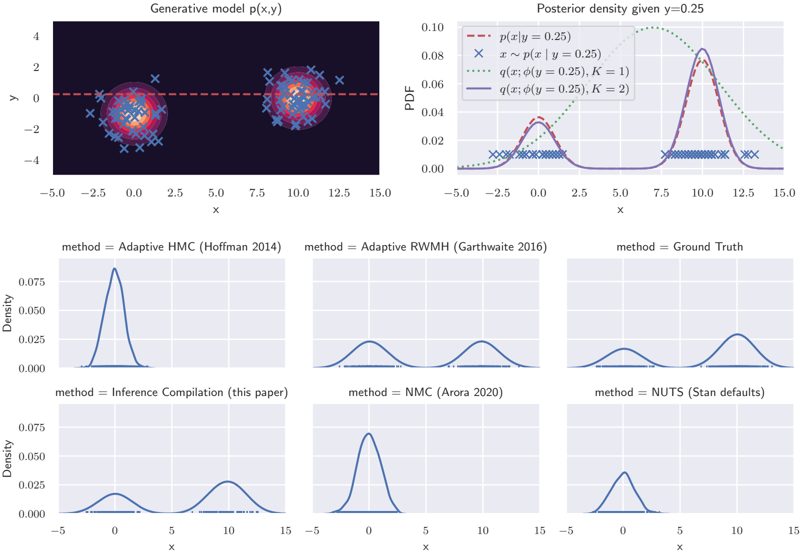
Recovering conjugate expressions in normal-normal

$$x \sim N(0, 2),\ y \mid x \sim N(x, 0.1)$$

Know: $y \mid x \sim N(0.999x, 0.0001)$



GMM Mode Escape



Robustness to nuisance random variables

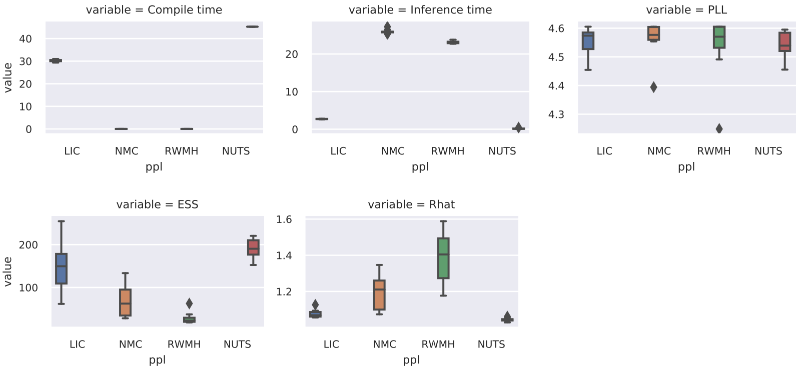
```
def magnitude(obs):
  x = sample(Normal(0, 10))
  for _ in range(100):
    nuisance = sample(Normal(0, 10))
  y = sample(Normal(0, 10))
  observe(
    obs**2,
    likelihood=Normal(x**2 + y**2, 0.1))
  return x
```

```
class NuisanceModel:
  @random_variable
  def x(self):
      return dist.Normal(0, 10)
  @random_variable
  def nuisance(self, i):
      return dist.Normal(0, 10)
  @random_variable
  def y(self):
      return dist.Normal(0, 10)
  @random_variable
  def noisy_sq_length(self):
      return dist.Normal(
          self.x()**2 + self.y()**2,
          0.1)
```

	# params	compile time	ESS
LIC (this paper)	3,358	44 sec.	49.75
[Le et al., 2017]	21,952	472 sec.	10.99

Bayesian Logistic Regression

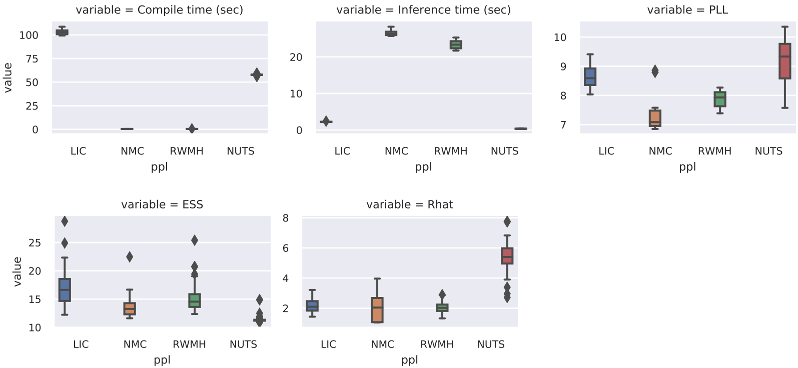
$$\beta \sim \mathcal{N}_{d+1}(0_{d+1}, \mathrm{diag}(10, 2.51_d)),$$
$$y_i \mid \boldsymbol{x}_i \overset{\text{iid}}{\sim} \mathrm{Bernoulli}(\sigma(\beta^\top \boldsymbol{x}_i)) \text{ where } \sigma(t) = (1 + e^{-t})^{-1}$$



n-Schools

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

n-Schools


variable = Compile time (sec)
variable = Inference time (sec)
variable = PLL
variable = ESS
variable = Rhat
value
ppl
LIC
NMC
RWMH
NUTS
100
75
50
25
0
20
10
9
8
7
15
6
4
2

Future directions

Adaptive LIC

Problem: forward samples during compilation may not sufficiently represent observations (obs) encountered at inference

RWMH adapts step size [Garthwaite et al., 2016], HMC adapts mass matrix [Hoffman and Gelman, 2014], can LIC adapt NN weights?

Solution: perform MCMC with IC artifact to draw posterior samples $(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)} = \text{obs}) \sim P(\boldsymbol{x} \mid \boldsymbol{y} = \text{obs})$, hill-climb inclusive KL between *conditional* (rather than joint) posterior

$$\arg\min_{\phi} D_{KL}(p(\boldsymbol{x}|\boldsymbol{y} = \text{obs}) || q(\boldsymbol{x}|\boldsymbol{y} = \text{obs}; \phi))$$

$$\approx \arg\min_{\phi} \sum_{m=1}^{N} \log Q(\boldsymbol{x}^{(m)} \mid \boldsymbol{y} = \text{obs}, \phi)$$

IAF density estimators

Problem: GMM in LIC may not be as expressive as more recently developed density estimators [Kingma et al., 2016]

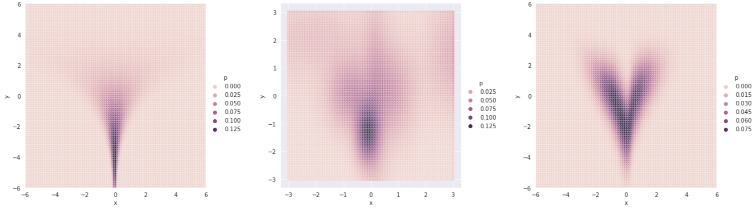
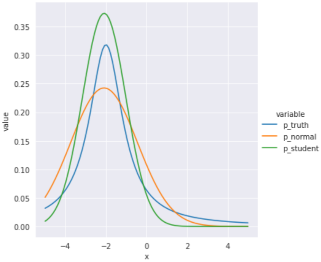


Figure 7: Neal's funnel (left) and a 7-component isotropic GMM (middle) and 7-layer IAF (right) density approximation

Heavy-tailed density estimators

Problem: GMMs and standard IAFs (Lipschitz functions of Gaussians) remain sub-Gaussian, *n*-schools is heavy tailed

Idea: IAFs with heavy-tailed base distribution



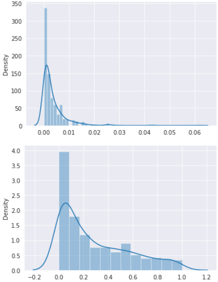


Figure 8: IAF density estimation of a Cauchy$(-2, 1)$ (left) and their K-S statistics when using a Normal (right top) and StudentT (right bottom) base distribution

References i

Arora, N. S., Tehrani, N. K., Shah, K. D., Tingley, M., Li, Y. L., Torabi, N., Noursi, D., Masouleh, S. A., Lippert, E., and Meijer, E. (2020).
Newtonian monte carlo: single-site mcmc meets second-order gradient methods.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003).
Latent dirichlet allocation.
Journal of machine Learning research, 3(Jan):993–1022.

References ii

Garthwaite, P. H., Fan, Y., and Sisson, S. A. (2016).
Adaptive optimal scaling of metropolis–hastings algorithms using the robbins–monro process.
Communications in Statistics-Theory and Methods, 45(17):5098–5111.

Ghahramani, Z. (2015).
Probabilistic machine learning and artificial intelligence.
Nature, 521(7553):452–459.

References iii

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016).
***Deep learning*.**
MIT press Cambridge.

Goodman, N. D. (2013).
The principles and practice of probabilistic programming.
ACM SIGPLAN Notices, 48(1):399–402.

References iv

- Harvey, W., Munk, A., Baydin, A. G., Bergholm, A., and Wood, F. (2019).
  Attention for inference compilation.
  arXiv preprint arXiv:1910.11961.

- Hastings, W. K. (1970).
  Monte carlo sampling methods using markov chains and their applications.

- Hoffman, M. D. and Gelman, A. (2014).
  The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo.
  J. Mach. Learn. Res., 15(1):1593–1623.

References v

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016).
Improved variational inference with inverse autoregressive flow.
In *Advances in neural information processing systems*, pages 4743–4751.

Koller, D. and Friedman, N. (2009).
Probabilistic graphical models: principles and techniques.
MIT press.

Le, T. A., Baydin, A. G., and Wood, F. (2017).
Inference compilation and universal probabilistic programming.
In *Artificial Intelligence and Statistics*, pages 1338–1348.

LeCun, Y., Bengio, Y., and Hinton, G. (2015).
Deep learning.
nature, 521(7553):436–444.

Norvig, P. R. and Intelligence, S. A. (2002).
***A modern approach*.**
Prentice Hall.

- Pearl, J. (1987).
  Evidential reasoning using stochastic simulation of causal models.
  Artificial Intelligence, 32(2):245–257.
- Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011).
  How to grow a mind: Statistics, structure, and abstraction.
  science, 331(6022):1279–1285.

References viii

van de Meent, J.-W., Paige, B., Yang, H., and Wood, F. (2018).
An introduction to probabilistic programming.
arXiv preprint arXiv:1809.10756.

Wingate, D., Stuhlmüller, A., and Goodman, N. (2011).
Lightweight implementations of probabilistic programming languages via transformational compilation.
In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 770–778.

References ix

Yuan, C. and Druzdzel, M. J. (2007).
Theoretical analysis and practical insights on importance sampling in bayesian networks.
International Journal of Approximate Reasoning, 46(2):320–333.