Probabilistic Programs with Stochastic Conditioning

David Tolpin [1] **Yuan Zhou** [2] **Tom Rainforth** [3] **Hongseok Yang** [4]

Abstract

We tackle the problem of conditioning probabilistic programs on distributions of observable variables. Probabilistic programs are usually conditioned on samples from the joint data distribution, which we refer to as deterministic conditioning. However, in many real-life scenarios, the observations are given as marginal distributions, summary statistics, or samplers. Conventional probabilistic programming systems lack adequate means for modeling and inference in such scenarios. We propose a generalization of deterministic conditioning to *stochastic conditioning*, that is, conditioning on the marginal distribution of a variable taking a particular form. To this end, we first define the formal notion of stochastic conditioning and discuss its key properties. We then show how to perform inference in the presence of stochastic conditioning. We demonstrate potential usage of stochastic conditioning on several case studies which involve various kinds of stochastic conditioning and are difficult to solve otherwise. Although we present stochastic conditioning in the context of probabilistic programming, our formalization is general and applicable to other settings.

1. Introduction

Probabilistic programs implement statistical models. Mostly, probabilistic programming closely follows the Bayesian approach (Kim & Pearl, 1983; Gelman et al., 2013): a prior distribution is imposed on latent random variables, and the posterior distribution is conditioned on observations (data). The conditioning may take the form of a hard constraint (a random variate must have a particular value, e.g. as in Church (Goodman et al., 2008)), but a more common practice is that conditioning is soft — the observation is assumed to come from a distribution, and the probability (mass or density) of the observation given the distribution is used in inference (Carpenter et al., 2017; Goodman & Stuhlmüller, 2014; Tolpin et al., 2016; Ge et al., 2018). In the standard setting, the conditioning is *deterministic* — observations are fixed samples from the joint data distribution. For example, in the model of an intensive care unit patient, an observation may be a vector of vital sign readings at a given time. This setting is well-researched, widely applicable, and robust inference is possible (Hoffman & Gelman, 2011; Wood et al., 2014; TensorFlow, 2018; Bingham et al., 2019).

However, instead of samples from the joint data distribution, observations may be independent samples from marginal data distributions of observable variables, summary statistics, or even data distributions themselves, provided in closed form or as samplers. These cases naturally appear in real life scenarios: samples from marginal distributions arise when different observations are collected by different parties, summary statistics are often used to represent data about a large population, and data distributions may express uncertainty during inference about future states of the world, e.g. in planning. Consider the following situations:

- A study is performed in a hospital on a group of patients carrying a certain disease. To preserve the patients' privacy, the information is collected and presented as summary statistics of each of the monitored symptoms, such that only marginal distributions of the symptoms are approximately characterized. It would be natural to condition the model on a combination of symptoms, but such combinations are not observable.
- A traveller regularly drives between two cities and wants to minimize the time this takes. However, some road sections may be closed due to bad weather, which can only be discovered at a crossing adjacent to the road section. A policy that minimizes average travel time, given the probabilities of each road closure, is required, but finding this policy requires us to condition on the *distribution* of states (Papadimitriou & Yannakakis, 1989).

Most existing probabilistic programming systems, which condition on samples from the joint data distribution, cannot be directly applied to such scenarios for either model specification or performing inference. In principle, such models

[1]Ben-Gurion University of the Negev [2]Artificial Intelligence Research Center, DII [3]University of Oxford [4]School of Computing, KAIST. Correspondence to: David Tolpin <david.tolpin@gmail.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

can be expressed as nested probabilistic programs (Rainforth, 2018). However, inference in such programs has limited choice of algorithms, is computationally expensive, and is difficult to implement correctly (Rainforth et al., 2018).

In some specific settings, models can be augmented with additional auxiliary information, and custom inference techniques can be used. For example, van de Meent et al. (2016) employs black-box variational inference on augmented probabilistic programs for policy search. But problem-specific program augmentation and custom inference compromise the core promise of probabilistic programming: programs and algorithms should be separated, and off-the-shelf inference methods should be applicable to a wide range of programs in an automated, black-box manner (Wood et al., 2014; Tolpin, 2019; Winn et al., 2019).

To address these issues and provide a general solution to defining models and running inference for such problems, we propose a way to extend *deterministic conditioning* $p(x|y = y_0)$, i.e. conditioning on some random variable in our program y taking on a particular value y_0, to *stochastic conditioning* $p(x|y{\sim}D_0)$, i.e. conditioning on y having the marginal distribution D_0. In the context of a higher-order sampling process in which we first sample a random probability measure $\mathbf{D} \sim p(\mathbf{D})$ and then sample a random variable $y \sim \mathbf{D}$, stochastic conditioning $p(x|y{\sim}D_0)$ amounts to conditioning on the event $\mathbf{D} = D_0$, that is on the random measure $\mathbf{D}$ itself taking the particular form D_0. Equivalently, we can think on conditioning on the event $y \sim D_0$ (based on the first step of our sampling process), which says that the marginal distribution of y is given by the distribution D_0. We can develop intuition for this by considering the special case of a discrete y, where $y \sim D_0$ means that the proportion of each possible instance of y that occurs will be D_0 if we conduct an infinite number of rollouts and sample a value of y for each.

To realize this intuition, we formalize stochastic conditioning and analyze its properties and usage in the context of probabilistic programming, further showing how effective automated inference engines can be set up for the resulting models. We note that our results also address a basic conceptual problem in Bayesian modeling, and are thus applicable to non-probabilistic programming settings as well.

We start with an informal introduction providing intuition about stochastic conditioning (Section 2). Then, we define the notion of stochastic conditioning formally and discuss its key properties (Section 3), comparing our definition with possible alternatives and related concepts. Following that, we discuss efficient inference for programs with stochastic conditioning (Section 4). In case studies (Section 5), we provide probabilistic programs for several problems of statistical inference which are difficult to approach otherwise, perform inference on the programs, and analyze the results.

2. Intuition

To get an intuition behind stochastic conditioning, we take a fresh look at the Beta-Bernoulli generative model:

$$x \sim \mathrm{Beta}(\alpha, \beta), \quad y \sim \mathrm{Bernoulli}(x). \tag{1}$$

The Beta prior on x has α and β parameters, which are interpreted as the belief about the number of times $y{=}1$ and $y{=}0$ seen before. Since Beta is the conjugate prior for $\mathrm{Bernoulli}$, belief updating in (1) can be performed analytically:

$$x|y \sim \mathrm{Beta}(\alpha + y, \beta + 1 - y)$$

We can compose Bayesian belief updating. If after observing y we observed y', then[1]

$$x|y' \circ y \sim \mathrm{Beta}(\alpha + y + y', \beta + 2 - y - y').$$

In general, if we observe $y_{1:n} = y_n \circ ... \circ y_2 \circ y_1$, then

$$x|y_{1:n} \sim \mathrm{Beta}\Big(\alpha + \sum\nolimits_{i=1}^{n} y_i, \beta + n - \sum\nolimits_{i=1}^{n} y_i\Big).$$

In (1) (also in many more general exchangeable settings) belief updating is commutative — the posterior distribution does not depend on the order of observations. One may view $y_{1:n}$ as a multiset, rather than a sequence, of observations.

Let us now modify the procedure of presenting the evidence. Instead of observing the value of each of $y_n \circ .... \circ y_2 \circ y_1$ in order, we just observe n variates, of which $k = \sum_{i=1}^{n} y_i$ variates have value 1 (but we are not told which ones). It does not matter which of the observations are 1 and which are 0. We can even stretch the notion of a single observation and say that it is, informally, a 'combination' of 1 with probability $\theta = \frac{k}{n}$ and 0 with probability $1 - \theta$. In other words, we can view each observation y_i as an observation of distribution $\mathrm{Bernoulli}(\theta)$ itself; the posterior distribution of x given n observations of $\mathrm{Bernoulli}(\theta)$ should be the same as the posterior distribution of x given $y_{1:n}$. This extended interpretation of belief updating based on observing distributions lets us answer questions about the posterior of x given that we observe *the distribution* $\mathrm{Bernoulli}(\theta)$ of y:

$$x|\big(y{\sim}\mathrm{Bernoulli}(\theta)\big) \sim \mathrm{Beta}(\alpha + \theta, \beta + 1 - \theta).$$

Note that observing a distribution does not imply observing its parametric representation. One may also observe a distribution through a random source of samples, a black-box unnormalized density function, or summary statistics.

Commonly, probabilistic programming involves weighing different assignments to x by the conditional probability of y given x. For model (1),

$$p(y|x) = x^y(1 - x)^{1-y}.$$

[1] By $y' \circ y$ we denote that y' was observed after observing y and updating the belief about the distribution of x.

The conditional probability of observing a fixed value extends naturally to observing a distribution:

$$\begin{aligned} p(y{\sim}\text{Bernoulli}(\theta)|x) &= x^{\theta}(1-x)^{1-\theta} \\ &= \exp\left(\theta \log x + (1-\theta)\log(1-x)\right) \\ &= \exp\Big(\sum\nolimits_{y\in\{0,1\}} p_{\text{Bern}(\theta)}(y)\cdot \log p_{\text{Bern}(x)}(y)\Big) \end{aligned} \tag{2}$$

where $p_{\text{Bern}(r)}(y)$ is the probability mass function of the distribution $\text{Bernoulli}(r)$ evaluated at y. Note that $p_{\text{Bern}(x)}$ inside the log is precisely the conditional probability of y in model (1). Equation (2) lets us specify a probabilistic program for a version of model (1) with stochastic conditioning — on a distribution rather than on a value. In the next section, we introduce stochastic conditioning formally, using a general form of (2).

3. Stochastic Conditioning

Let us define stochastic conditioning formally. In what follows, we mostly discuss the continuous case where the observed distribution D has a density q. For the case that D does not have a density, the notation $q(y)dy$ in our discussion should be replaced with $D(dy)$, which means the Lebesgue integral with respect to the distribution (or probability measure) D. For the discrete case, probability densities should be replaced with probability masses, and integrals with sums. Modulo these changes, all the theorem and propositions in this section carry over to the discrete case. A general measure-theoretic formalization of stochastic conditioning, which covers all of these cases uniformly, is described in Appendix A.

Definition 1. *A probabilistic model with stochastic conditioning is a tuple $(p(x,y), D)$ where (i) $p(x,y)$ is the joint probability density of random variable x and observation y, and it is factored into the product of the prior $p(x)$ and the likelihood $p(y|x)$ (i.e., $p(x,y) = p(x)p(y|x)$); (ii) D is the distribution from which observation y is marginally sampled, and it has a density $q(y)$.*

Unlike in the usual setting, our objective is to infer $p(x|y{\sim}D)$, the distribution of x given *distribution* D, rather than an individual observation y. To accomplish this objective, we need to be able to compute $p(x, y{\sim}D)$, a possibly unnormalized density on x and distribution D. We define $p(x, y{\sim}D) = p(x)p(y{\sim}D|x)$ where $p(y{\sim}D|x)$ is the following unnormalized conditional density:

Definition 2. *The (unnormalized) conditional density $p(y{\sim}D|x)$ of D given x is*

$$p(y{\sim}D|x) = \exp\left(\int_Y (\log p(y|x))\, q(y)dy\right) \tag{3}$$

where q is the density of D.

An intuition behind the definition can be seen by rewriting (3) as a type II geometric integral:

$$p(y{\sim}D|x) = \prod\nolimits_Y p(y|x)^{q(y)dy}.$$

Definition 2 hence can be interpreted as the probability of observing *all* possible draws of y from D, each occurring according to its probability $q(y)dy$.

At this point, the reader may wonder why we do not take the following alternative, frequently coming up in discussions:

$$p_1(y{\sim}D|x) = \int_Y p(y|x)\, q(y)dy. \tag{4}$$

One may even see a connection between (4) and Jeffrey's soft evidence (Jeffrey, 1990)

$$p(x|y \overset{\text{soft}}{\sim} D) = \int_Y q(y)p(x|y)\, dy, \tag{5}$$

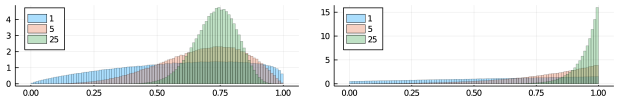
although the latter addresses a different setting. In soft evidence, an observation is a single value y, but the observer does not know with certainty which of the values was observed. Any value y from Y, the domain of D, can be observed with probability $q(y)$, but $p(y \overset{\text{soft}}{\sim} D|x)$ cannot be generally defined (Chan & Darwiche, 2003; Ben Mrad et al., 2013). In our setting, distribution D is observed, and the observation is certain.

We have two reasons to prefer (3) to (4). First, as Proposition 1 will explain, our $p(y{\sim}D|x)$ is closely related to the KL divergence between $q(y)$ and $p(y|x)$, while the alternative $p_1(y{\sim}D|x)$ in (4) lacks such connection. The connection helps understand how $p(y{\sim}D|x)$ alters the prior of x. Second, $p(y{\sim}D|x)$ treats all possible draws of y more equally than $p_1(y{\sim}D|x)$ in the following sense. Both $p(y{\sim}D|x)$ and $p_1(y{\sim}D|x)$ are instances of so called power mean (Bullen, 2003) defined by

$$p_\alpha(y{\sim}D|x) = \left(\int_Y p(y|x)^\alpha\, q(y)dy\right)^{\frac{1}{\alpha}}. \tag{6}$$

Setting α to 0 and 1 gives our $p(y{\sim}D|x)$ and the alternative $p_1(y{\sim}D|x)$, respectively. A general property of this power mean is that as α tends to ∞, draws of y with large $p(y|x)$ contribute more to $p_\alpha(y{\sim}D|x)$, and the opposite situation happens as α tends to $-\infty$. Thus, the $\alpha = 0$ case, which gives our $p(y{\sim}D|x)$, can be regarded as the option that is the least sensitive to $p(y|x)$.

As an illustration of difference between the two definitions, consider Figure 1 showing posteriors of x for the Beta-Bernoulli model (Section 2) after 1, 5, and 25 observations of $\text{Bernoulli}(\theta = 0.75)$, according to either (3) or (4). According to (3) the mode is between 0.5 and 0.75, approaching 0.75 as the number of observations grows; according to the mode is always 1.



(a) Definition (3) (b) Definition (4)

Figure 1: Comparison of definitions (3) and (4) on the discrete case of Beta-Bernoulli model (Section 2).

Proposition 1. *For a given D with density q,*

$$\arg\max_x p(y{\sim}D|x) = \arg\min_x \mathrm{KL}(q(y)||p(y|x)). \quad (7)$$

In particular, if there exists x^* such that $p(y|x^*) \equiv q(y)$, then $x^* = \arg\max_x p(y{\sim}D|x)$.

Proof. We prove the proposition by re-expressing $\log p(y{\sim}D|x)$ in terms of the negative KL divergence:

$$\begin{aligned}\log p(y{\sim}D|x) &= \int_Y (\log p(y|x))\, q(y)dy \\ &= \left(\int_Y (\log q(y))\, q(y)dy\right) - \int_Y \left(\log \frac{q(y)}{p(y|x)}\right) q(y)dy \\ &= \left(\int_Y (\log q(y))\, q(y)dy\right) - \mathrm{KL}(q(y)||p(y|x)).\end{aligned}$$

Since the first term in the last line does not depend on x, we have the equation (7). □

The proposition does not hold for the alternative definition $p_1(y{\sim}D|x)$ in (4). Even if there exists x^* such that $p(y|x^*) \equiv q(y)$, all we can say about $p(y|\arg\max_x p_1(y{\sim}D|x))$ is just that it maximizes $\int_Y p(y|x)\, q(y)dy$, not that it is $q(y)$. For example, consider the finite discrete case and assume that there also exists $x^\dagger$ with $p(y|x^\dagger) = \mathrm{Dirac}(\arg\max_y q(y))$. Then $x^\dagger$, not x^*, maximizes $p_1(y{\sim}D|x)$.

The next theorem explains our setting formally and shows that $p(y{\sim}D|x)$ has a finite normalization constant.

Theorem 1. *Assume that the distribution D_θ is parameterized by $\theta \in \Theta \subseteq \mathbb{R}^p$. Let q_θ be its density. Then, $p(y{\sim}D_\theta|x)$ has a finite normalization constant C over $\mathcal{D} = \{D_\theta \mid \theta \in \Theta\}$ if the following uniform-bound condition holds: $\sup_{y\in Y} \int_\Theta q_\theta(y)\, d\theta < \infty$. Thus, in this case, $p(y{\sim}D_\theta|x)/C$ is a conditional probability density on $\mathcal{D}$.*

Proof. Let $C = \int_\Theta p(y{\sim}D_\theta|x)\, d\theta$. Then,

$$C = \int_\Theta \exp\left(\int_Y (\log p(y|x))\, q_\theta(y)dy\right) d\theta. \quad (8)$$

We compute a finite upper bound of C as follows:

$$\begin{aligned}C &\leq^1 \int_\Theta \int_Y \exp(\log p(y|x))\, q_\theta(y)dyd\theta \\ &= \int_\Theta \int_Y p(y|x)\, q_\theta(y)dyd\theta \\ &=^2 \int_Y \left(\int_\Theta q_\theta(y)\, d\theta\right) p(y|x)dy \\ &\leq \left(\sup_{y'\in Y} \int_\Theta q_\theta(y')\, d\theta\right) \cdot \left(\int_Y p(y|x)dy\right) \\ &= \sup_{y'\in Y} \int_\Theta q_\theta(y')\, d\theta <^3 \infty.\end{aligned}$$

Here $\leq^1$ is by Jensen's inequality, $=^2$ follows from Fubini's theorem, and $<^3$ uses the uniform-bound condition. □

Let us illustrate the restriction on the set of distributions $\mathcal{D}$ imposed by the uniform-bound condition in Theorem 1. The set $\mathcal{D} = \{\mathrm{Normal}(\theta, 1) \mid \theta \in \mathbb{R}\}$ of normal distributions with the fixed variance 1 meets the condition, so that when normalized, $p(y{\sim}D|x)$ becomes a probability density over $\mathcal{D}$. However, if we permit the variance to vary, the resulting set $\mathcal{D}' = \{\mathrm{Normal}(\theta, \sigma^2) \mid \theta \in \mathbb{R}, \sigma^2 \in (0, \infty)\}$ does not satisfy the condition. Appendix A contains another set that violates a measure-theoretic generalization of our condition (described in Appendix A as well) and, furthermore, does not have a normalization constant for some choice of p.

The next proposition shows that stochastic conditioning generalizes conventional conditioning on a value. It uses a non-density version of Definition 1 where the integral over $q(y)dy$ is understood as the integral over the distribution (i.e., probability measure) D.

Proposition 2. *When observing a distribution reduces to observing a single value, $D = \mathrm{Dirac}(y)$, conditioning on a distribution reduces to conventional conditioning on a value: $p(\mathrm{Dirac}(y)|x) = p(y|x)$.*

4. Inference

Algorithms for deterministic conditioning cannot be applied without modification to probabilistic programs with stochastic conditioning. One approach is to rely on nested Monte Carlo estimation (Rainforth et al., 2018). However, probabilistic programs with stochastic conditioning constitute an important special case of nested models, and it is possible to leverage properties of such programs to apply a wider class of inference algorithms. In our setting, $\log p(y{\sim}D|x)$ is easy to estimate, and $p(y{\sim}D|x)$ can be estimated in a bias-adjusted manner, hence inference algorithms using these estimates can be applied effectively.

In particular, Definition 2 implies that an unbiased Monte Carlo estimate of log likelihood $\log p(y{\sim}D|x)$ is available based on samples $y_i \sim D$ through which D is observed:

$$\log p(y{\sim}D|x) \approx \frac{1}{N} \sum_{i=1}^N \log p(y_i|x) \quad (9)$$

Another setting in which an unbiased Monte Carlo estimate of log likelihood is available is subsampling for inference in models with tall data (Korattikara et al., 2014; Bardenet et al., 2014; 2017; Maclaurin & Adams, 2014; Quiroz et al., 2018; 2019; Dang et al., 2019). In models considered for subsampling, K observations $y_1, y_2, ..., y_K$ are conditionally independent given x:

$$p(y_1, y_2, ..., y_K|x) = \prod_{i=1}^{K} p(y_i|x) \tag{10}$$

Most inference algorithms require evaluation of likelihood $p(y_1, y_2, ..., y_K|x)$, which is expensive if K is large. For example, in importance sampling, the likelihood is involved in the computation of importance weights. In many Markov chain Monte Carlo methods, the ratio of likelihoods of the proposed and the current state is a factor in the Metropolis-Hastings acceptance rate. Subsampling replaces $p(y_1, y_2, ..., y_K|x)$ by an estimate based on N samples $y_{i_1}, y_{i_2}, ..., y_{i_N}$, $N < K$, which results in an unbiased Monte Carlo estimate of log likelihood:

$$\log p(y_1, y_2, ..., y_K|x) \approx \frac{K}{N} \sum_{j=1}^{N} \log p(y_{i_j}|x) \tag{11}$$

The only difference between (9) and (11) is in factor K, and inference algorithms for subsampling can be applied to stochastic conditioning with minor modifications.

A simple bias-adjusted likelihood estimate $\hat{p}(x, y{\sim}D)$, required for the computation of the weights in importance sampling as well as of the acceptance ratio in pseudo-marginal Markov chain Monte Carlo (Andrieu & Roberts, 2009), can be computed based on (9) (Ceperley & Dewing, 1999; Nicholls et al., 2012; Quiroz et al., 2018). Stochastic gradient-based inference algorithms (Chen et al., 2014; Ma et al., 2015; Hoffman et al., 2013; Ranganath et al., 2014; Kucukelbir et al., 2017) rely on an unbiased estimate of the gradient of log likelihood, which is trivially obtained by differentiating both sides of (9).

We implemented inference in probabilistic programs with stochastic conditioning for Infergo (Tolpin, 2019). To facilitate support for stochastic conditioning in other probabilistic programming systems, we provide details on likelihood estimation and some possible adaptations of inference algorithms to stochastic conditioning, as well as pointers to alternative adaptations in the context of subsampling, in Appendix B.

5. Case Studies

In the case studies, we explore several problems cast as probabilistic programs with stochastic conditioning. We place $y \sim D$ above a rule to denote that distribution D is observed through y and is otherwise unknown to the model, as in (12). Some models are more natural to express in terms of the joint probability that they compute than in terms of distributions from which x is drawn and y is observed. In that case, we put the expression for the joint probability $p(x, y)$ under the rule, as in (13).

$$\frac{\begin{array}{c} y \sim D \\ \hline x \sim \textit{Prior} \\ y|x \sim \textit{Conditional}(x) \end{array}}{} \tag{12} \qquad \frac{y \sim D}{p(x, y) = ...} \tag{13}$$

The code and data for the case studies are provided in repository `https://bitbucket.org/dtolpin/stochastic-conditioning`.

5.1. Inferring the Accuracy of Weather Forecast

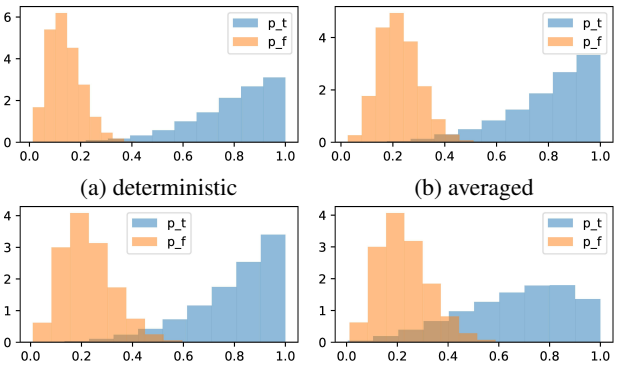
A person commutes to work either by motorcycle or, on rainy days, by taxi. When the weather is good, the motorcycle ride takes 15 ± 2 minutes via a highway. If rain is expected, the commuter takes a taxi, and the trip takes 30 ± 4 minutes, because of crowded roads which slow down a four-wheeled vehicle. Sometimes, however, the rain catches the commuter in the saddle, and the commuter rides slowly and carefully through rural roads, arriving at 60 ± 8 minutes. Given weather observations and trip durations, we want to estimate the accuracy of rain forecasts, that is, the probability of the positive forecast on rainy days p_t (true positive) and on dry days p_f (false positive).

The problem is represented by the following model:

$$\begin{aligned} p_r, p_t, p_f &\sim \text{Beta}(1, 1) \\ \textit{rain}|p_r &\sim \text{Bernoulli}(p_r) \\ \textit{willRain}|p_t, p_f, \textit{rain} &\sim \begin{cases} \text{Bernoulli}(p_t) \text{ if } \textit{rain} \\ \text{Bernoulli}(p_f) \text{ otherwise} \end{cases} \\ \textit{duration}|\textit{rain}, \textit{willRain} &\sim \begin{cases} \text{Normal(30, 4) if } \textit{willRain} \\ \text{Normal(15, 2) if } \neg\textit{rain} \\ \text{Normal(60, 8) otherwise} \end{cases} \end{aligned} \tag{14}$$

Model (14) can be interpreted as either a simulator that draws samples of (rain, duration) given p_r, p_t, and p_f, or as a procedure that computes the conditional probability of (rain, duration) given p_r, p_t, and p_f. We use the simulator interpretation to generate synthetic observations for 30 days and $p_r = 0.2, p_t = 0.8, p_f = 0.1$. The conditional probability interpretation lets us write down a probabilistic program for posterior inference of p_t and p_f given observations.

If, instead of observing (rain, duration) simultaneously, we observe weather conditions and trip durations separately and do not know correspondence between them (a common situation when measurements are collected by different parties), we can still write a conventional probabilistic program conditioned on the Cartesian product of weather conditions and trip durations, but the number of observations and, thus, time complexity of inference becomes quadratic in the num-



(a) deterministic (b) averaged (c) stochastic (d) intensity

Figure 2: Commute to work: posteriors of p_t and p_f for each of the four models. Despite being exposed to partial information only, models with stochastic conditioning let us infer informative posteriors.

ber of days. In general, when a model is conditioned on the Cartesian product of separately obtained observation sets, inference complexity grows exponentially with the dimensionality of observations, and inference is infeasible in problems with more than a couple of observed features.

Alternatively, we can draw rain and duration from the observation sets randomly and independently, and stochastically condition on $D = \mathit{Rains} \times \mathit{Durations}$:

$$\frac{\mathit{rain}, \mathit{duration} \sim \mathit{Rains} \times \mathit{Durations}}{\ldots} \tag{15}$$

One can argue that the probabilistic program for the case of independent sets of observations of rain and duration can still be implemented with linear complexity by noting that the domain of rain contains only two values, true and false, and analytically averaging the evidence over rain. However, such averaging is often impossible. Consider a variant of the problem in which the duration of a motorcycle trip in rain depends on rain intensity. Stochastic conditioning, along with inference algorithms that use a small number of samples to estimate the log likelihood (magnitude or gradient), lets us preserve linear complexity in the number of observations (Doucet et al., 2015; Bardenet et al., 2017).

We fit the model using stochastic gradient Hamiltonian Monte Carlo and used 10 000 samples to approximate the posterior. Figure 2 shows marginal posteriors of p_t and p_f for each of the four models, on the same simulated data set. Posterior distributions should be the same for the analytically averaged and stochastic models. The deterministic model is exposed to more information (correspondence between rain occurrence and trip duration). Hence, the posterior distributions are more sharply peaked. The stochastic model with observation of intensity should be less confident about p_t, since now the observation of a motorcycle trip duration slowed down by rain is supposed to come from a distribution conditioned on rain intensity.

Table 1: Summary statistics for populations of municipalities in New York State in 1960; all 804 municipalities and two random samples of 100. From Rubin (1983).

	Population	Sample 1	Sample 2
total	13,776,663	1,966,745	3,850,502
mean	17,135	19,667	38,505
sd	139,147	142,218	228,625
lowest	19	164	162
5%	336	308	315
25%	800	891	863
median	1,668	2,081	1,740
75%	5,050	6,049	5,239
95%	30,295	25,130	41,718
highest	2,627,319	1,424,815	1809578

5.2. Estimating the Population of New York State

This case study is inspired by Rubin (1983), also appearing as Section 7.6 in Gelman et al. (2013). The original case study evaluated Bayesian inference on the problem of estimating the total population of 804 municipalities of New York state based on a sample of 100 municipalities. Two samples were given, with different summary statistics, and power-transformed normal model was fit to the data to make predictions consistent among the samples. The authors of the original study apparently had access to the full data set (population of each of 804 municipalities). However, only summary description of the samples appears in the publication: mean, standard deviation, and quantiles (Table 1). We show how such summary description can be used to perform Bayesian inference, with the help of stochastic conditioning.

The original case study in Rubin (1983) started with comparing normal and log-normal models, and finally fit a truncated three-parameter power-transformed normal distribution to the data, which helped reconcile conclusions based on each of the samples while producing results consistent with the total population. Here, we use a model with log-normal sampling distribution, the normal prior on the mean, based on the summary statistics, and the improper uniform prior on the log of the variance. To complete the model, we stochastically condition on the piecewise-uniform distribution D of municipality populations according to the quantiles:

$$\begin{gathered} y_{1\ldots n} \sim \mathit{Quantiles} \\ \hline m \sim \mathrm{Normal}\Big(\mathit{mean}, \mathit{sd}/\sqrt{n}\Big),\ \log s^2 \sim \mathrm{Uniform}(-\infty, \infty) \\ \sigma = \sqrt{\log\left(s^2/m^2 + 1\right)}, \quad \mu = \log m - \sigma^2/2 \\ y_{1\ldots n} | m, s^2 \sim \mathrm{LogNormal}(\mu, \sigma) \end{gathered} \tag{16}$$

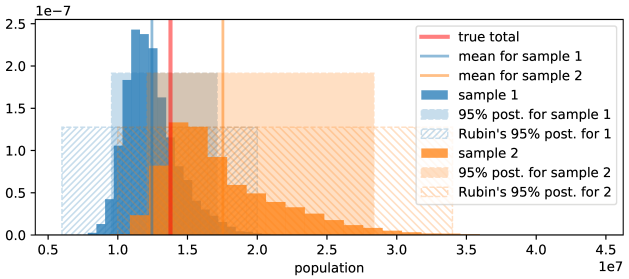
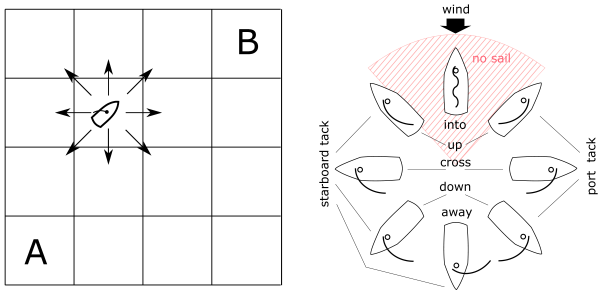


Figure 3: Estimating the population of NY state. 95% intervals inferred from the summary statistics include the true total, and are tighter than Rubin's results.



(a) lake (b) points of sail

Figure 4: The sailing problem

As in Section 5.1, we fit the model using stochastic gradient HMC and used 10 000 samples to approximate the posterior. We then used 10 000 draws with replacement of 804-element sample sets from the predictive posterior to estimate the total population. The posterior predictive distributions of the total population from both samples are shown in Figure 3. The 95% intervals inferred from the summary statistics, $[9.6\times10^6, 17.2\times10^6]$ for sample 1, $[12.1\times10^6, 28.1\times10^6]$ for sample 2, cover the true total 13.8×10^6, and are tighter than the best intervals based on the full samples reported by Rubin (1983), $[6\times10^6, 20\times10^6]$ for sample 1 and $[10\times10^6, 34\times10^6]$ for sample 2.

5.3. The Sailing Problem

The sailing problem (Figure 4) is a popular benchmark problem for search and planning (Péret & Garcia, 2004; Kocsis & Szepesvári, 2006; Tolpin & Shimony, 2012). A sailing boat must travel between the opposite corners A and B of a square lake of a given size. At each step, the boat can *head* in 8 directions (*legs*) to adjacent squares (Figure 4a). The unit distance cost of movement depends on the wind (Figure 4b), which can also blow in 8 directions. There are five relative boat and wind directions and associated costs: *into*, *up*, *cross*, *down*, and *away*. The cost of sailing into the wind is prohibitively high, upwind is the highest feasible, and away from the wind is the lowest. The side of the boat off which the sail is hanging is called the *tack*. When the angle between the boat and the wind changes sign, the sail must be *tacked* to the opposite tack, which incurs an additional *tacking delay* cost. The objective is to find a policy that minimizes the expected travel cost. The wind is assumed to follow a random walk, either staying the same or switching to an adjacent direction, with a known probability.

Table 2: Sailing problem parameters

cost						wind probability		
into	up	cross	down	away	delay	same	left	right
∞	4	3	2	1	4	0.4	0.3	0.3

For any given lake size, the optimal policy can be found using value iteration (Bellman, 1957). The optimal policy is non-parametric: it tabulates the leg for each combination of location, tack, and wind. In this case study, we learn a simple parametric policy, which chooses a leg that maximizes the sum of the leg cost and the remaining travel cost after the leg, estimated as the Euclidean distance to the goal multiplied by the average unit distance cost:

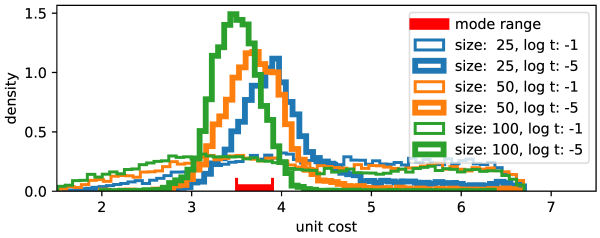
$$\begin{aligned} leg = \arg\min_{leg} \big[&cost(tack, leg, wind) + \\ &unit\text{-}cost \cdot distance(next\text{-}location, goal)\big] \end{aligned} \tag{17}$$

The average unit distance cost is the policy variable which we infer. Model (18) formalizes our setting. Stochastic conditioning on $D = RandomWalk$ models non-determinism in wind directions.

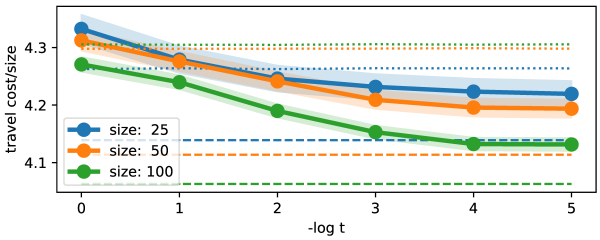
$$\begin{aligned} &wind\text{-}history \sim RandomWalk \\ &p(wind\text{-}history, unit\text{-}cost) = \\ &\frac{1}{Z}\exp\left(\frac{-travel\text{-}cost(wind\text{-}history, unit\text{-}cost)}{lake\text{-}size \cdot temperature}\right) \end{aligned} \tag{18}$$

Under policy (17), the boat trajectory and the travel cost are determined by the wind history and the unit cost. The joint probability of the wind history and the unit cost is given by the Boltzmann distribution of trajectories with the travel cost as the energy, a common physics-inspired choice in stochastic control and policy search (Kappen, 2007; Wingate et al., 2011a; van de Meent et al., 2016). The temperature is a model parameter: the lower the temperature is, the tighter is the concentration of policies around the optimal policy. A uniform prior on the unit cost, within a feasible range, is implicitly assumed. If desirable, an informative prior can be added as a factor depending on the unit cost.

The model parameters (cost and wind change probabilities), same as in Kocsis & Szepesvári (2006); Tolpin & Shimony (2012), are shown in Table 2. We fit the model using pseudo-marginal Metropolis-Hastings (Andrieu & Roberts, 2009) and used 10 000 samples to approximate the posterior. The



(a) unit cost



(b) expected travel cost

Figure 5: The sailing problem. The optimum unit cost is $\approx$ 3.5–3.9. The dashed lines are the expected travel costs of the optimal policies, dotted — of the greedy policy.

inferred unit and expected travel costs are shown in Figure 5. Figure 5a shows the posterior distribution of the unit cost, for two temperatures. For all lake sizes in the experiment (25, 50, 100), the optimal unit cost, corresponding to the mode of the posterior, is $\approx$ 3.5–3.9. Distributions for lower temperatures are tighter around the mode. Figure 5b shows the expected travel costs, with the expectations estimated both over the unit cost and the wind. The 95% posterior intervals are shaded. The inferred travel costs are compared to the travel costs of the optimal policy (the dashed line of the same color) and of the greedy policy (the dotted line of the same color), according to which the boat always heads in the direction of the steepest decrease of the distance to the goal. One can see that the inferred policies attain a lower expected travel cost than the greedy policy and become closer to the optimal policy as the temperature decreases.

5.4. Comparison to nested conditioning

We implemented and evaluated nested conditioning models for each of the case studies. Table 3 compares the running times and the number of evaluations of the conditional probability to reach the effective sample size of 1000 samples. Nested conditioning requires explicit manual coding in the current probabilistic programming frameworks. Inference in the presence of nested conditioning is inferior for two reasons:

1. Nested MC estimation requires $\approx \sqrt{N}$ inner samples for N outer samples, for **each** stochastically conditioned variable.

Table 3: Stochastic vs. nested conditioning

model	*stochastic*		*nested*	
	time	*evaluations*	*time*	*evaluations*
commute	4.1s	$2 \cdot 10^5$	520s	$4 \cdot 10^7$
nypopu	34s	$1 \cdot 10^7$	280s	$3 \cdot 10^8$
sailing	5.5s	$4 \cdot 10^5$	45s	$4 \cdot 10^6$

2. A model with nested conditioning is not differentiable even if latent variables are continuous. Gradient-based inference algorithms cannot be used, which leaves us with slow mixing.

As one can see from the comparison in Table 3, implementing the models as probabilistic programs with nested, rather than stochastic, conditioning results in much slower inference.

6. Related Work

Works related to this research belong to several interconnected areas: non-determinism in probabilistic programs, nesting of probabilistic programs, inference in nested statistical models, and conditioning on distributions.

Stochastic conditioning can be viewed as an expression of non-determinism with regard to the observed variate. The problem of representing and handling non-determinism in probabilistic programs was raised in Gordon et al. (2014), as an avenue for future work. Non-determinism arises, in particular, in application of probabilistic programming to policy search in stochastic domains. van de Meent et al. (2016) introduce a policy-search specific model specification and inference scheme based on black-box variational inference. We suggest, and show in a case study, that policy search in stochastic domains can be cast as inference in probabilistic programs with stochastic conditioning.

It was noted that probabilistic programs, or queries, can be nested, and that nested probabilistic programs are able to represent models beyond those representable by flat probabilistic programs. Stuhlmüller & Goodman (2014) describe how probabilistic programs can represent nested conditioning as a part of the model, with examples in diverse areas of game theory, artificial intelligence, and linguistics. Seaman et al. (2018) apply nested probabilistic programs to reasoning about autonomous agents. Some probabilistic programming languages such as Church (Goodman et al., 2008), WebPPL (Goodman & Stuhlmüller, 2014), Anglican (Tolpin et al., 2016), and Gen (Cusumano-Towner et al., 2019) support nesting of probabilistic programs. Stochastic conditioning can be, in principle, represented through nesting, however nesting in general incurs difficulties in inference (Rainforth et al., 2018; Rainforth, 2018). Stochastic

conditioning, introduced in this work, allows both simpler specification and more efficient inference, eliminating the need for nesting in many important cases.

Conditioning of statistical models on distributions or distributional properties is broadly used in machine learning (Chen & Gopinath, 2001; Kingma & Welling, 2019; Goodfellow et al., 2014; Makhzani et al., 2015; Bingham et al., 2019). Conditioning on distributions represented by samples is related to subsampling in deep probabilistic programming (Tran et al., 2017; TensorFlow, 2018; Bingham et al., 2019). Subsampling used with stochastic variational inference (Ranganath et al., 2014) can be interpreted as a special case of stochastic conditioning. Tavares et al. (2019) approach the problem of conditioning on distributions by extending probabilistic programming language OMEGA with constructs for conditioning on distributional properties such as expectation or variance. This work takes a different approach by generalizing deterministic conditioning on values to stochastic conditioning on distributions, without the need to explicitly compute or estimate particular distributional properties, and leverages inference algorithms developed in the context of subsampling (Korattikara et al., 2014; Bardenet et al., 2014; 2017; Maclaurin & Adams, 2014; Quiroz et al., 2018; 2019; Dang et al., 2019) for efficient inference in probabilistic programs with stochastic conditioning.

There is a connection between stochastic conditioning and Jeffrey's soft evidence (Jeffrey, 1990). In soft evidence, the observation is uncertain; any one out of a set of observations could have been observed with a certain known probability. A related concept in the context of belief networks is virtual evidence (Pearl, 1988). Chan & Darwiche (2003) demonstrate that Jeffrey's soft evidence and Pearl's virtual evidence are different formulations of the same concept. Ben Mrad et al. (2013); Dietrich et al. (2016); Jacobs (2018) elaborate on connection between soft and virtual evidence and their role in probabilistic inference. In probabilistic programming, some cases of soft conditioning (Wood et al., 2014; Goodman & Stuhlmüller, 2014) can be interpreted as soft evidence. In this work, the setting is different: a distribution is observed, and the observation is certain.

7. Discussion

In this work, we introduced the notion of stochastic conditioning. We described kinds of problems for which deterministic conditioning is insufficient, and showed on case studies how probabilistic programs with stochastic conditioning can be used to represent and efficiently analyze such problems. We believe that adoption of stochastic conditioning in probabilistic programming frameworks will facilitate convenient modeling of new classes of problems, while still supporting robust and efficient inference. The idea of stochastic conditioning is very general, and we believe our work opens up a wide array of new research directions because of this. Support for stochastic conditioning in other existing probabilistic programming languages and libraries is a direction for future work. While we provide a reference implementation, used in the case studies, we believe that stochastic conditioning should eventually become a part of most probabilistic programming frameworks, just like other common core concepts.

Acknowledgments

Yuan Zhou is partially supported by the National Natural Science Foundation of China (NSFC). Tom Rainforth's research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007- 2013)/ ERC grant agreement no. 617071. Tom Rainforth was also supported in part by Junior Research Fellowship from Christ Church, University of Oxford, and and in part by EPSRC funding under grant EP/P026753/1. Hongseok Yang was supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), and also by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2017M3C4A7068177). We thank PUB+ for supporting development of Infergo.

References

Andrieu, C. and Roberts, G. O. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

Bardenet, R., Doucet, A., and Holmes, C. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 405–413, 2014.

Bardenet, R., Doucet, A., and Holmes, C. On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research*, 18(47):1–43, 2017.

Bellman, R. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

Ben Mrad, A., Delcroix, V., Piechowiak, S., Maalej, M. A., and Abid, M. Understanding soft evidence as probabilistic evidence: Illustration with several use cases. In *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, pp. 1–6, 2013.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: deep universal prob-

abilistic programming. *Journal of Machine Learning Research*, 20(28):1–6, 2019.

Bullen, P. S. *Handbook of Means and Their Inequalities*. Springer Netherlands, 2003.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. Stan: a probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1): 1–32, 2017.

Ceperley, D. M. and Dewing, M. The penalty method for random walks with uncertain energies. *The Journal of Chemical Physics*, 110(20):9812–9820, May 1999.

Chan, H. and Darwiche, A. On the revision of probabilistic beliefs using uncertain evidence. *Artificial Intelligence*, 163:67–90, 2003.

Chen, S. S. and Gopinath, R. A. Gaussianization. In *Advances in Neural Information Processing Systems*, pp. 423–429. 2001.

Chen, T., Fox, E. B., and Guestrin, C. Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on on Machine Learning*, pp. 1683–1691, 2014.

Cusumano-Towner, M. F., Saad, F. A., Lew, A. K., and Mansinghka, V. K. Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 221–236, 2019.

Dang, K.-D., Quiroz, M., Kohn, R., Tran, M.-N., and Villani, M. Hamiltonian Monte Carlo with energy conserving subsampling. *Journal of Machine Learning Research*, 20 (100):1–31, 2019.

Dietrich, F., List, C., and Bradley, R. Belief revision generalized: A joint characterization of Bayes' and Jeffrey's rules. *Journal of Economic Theory*, 162:352–371, 2016.

Doucet, A., Pitt, M. K., Deligiannidis, G., and Kohn, R. Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2):295–313, 03 2015.

Ge, H., Xu, K., and Ghahramani, Z. Turing: composable inference for probabilistic programming. In *Proceedings of the 21st Conference on International Conference on Artificial Intelligence and Statistics*, pp. 1682–1690, 2018.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., and Rubin, D. *Bayesian Data Analysis*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2013.

Ghosal, S. and van der Vaart, A. *Fundamentals of Nonparametric Bayesian Inference*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2017. doi: 10.1017/9781139029834.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680. 2014.

Goodman, N. D. and Stuhlmüller, A. *The Design and Implementation of Probabilistic Programming Languages*. 2014. URL `http://dippl.org/`. electronic; retrieved 2019/3/29.

Goodman, N. D., Mansinghka, V. K., Roy, D. M., Bonawitz, K., and Tenenbaum, J. B. Church: a language for generative models. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pp. 220–229, 2008.

Gordon, A. D., Henzinger, T. A., Nori, A. V., and Rajamani, S. K. Probabilistic programming. In *Proceedings of the 36th International Conference on Software Engineering (ICSE, FOSE track)*, pp. 167–181, 2014.

Hoffman, M. D. and Gelman, A. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. arXiv:1111.4246, 2011.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

Jacobs, B. The mathematics of changing one's mind, via Jeffrey's or via Pearl's update rule. arXiv:1807.05609, 2018.

Jeffrey, R. *The Logic of Decision*. McGraw-Hill series in probability and statistics. University of Chicago Press, 1990.

Kappen, H. J. An introduction to stochastic control theory, path integrals and reinforcement learning. In *American Institute of Physics Conference Series*, volume 887, pp. 149–181, 2007.

Kim, J. and Pearl, J. A computational model for causal and diagnostic reasoning in inference systems. In *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 190–193, 1983.

Kingma, D. P. and Welling, M. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019.

Kocsis, L. and Szepesvári, C. Bandit based Monte-Carlo planning. In *Proceedings of the European Conference on Machine Learning*, pp. 282–293, 2006.

Korattikara, A., Chen, Y., and Welling, M. Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 181–189, 2014.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(1): 430–474, January 2017.

Ma, Y.-A., Chen, T., and Fox, E. A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, pp. 2917–2925. 2015.

Maclaurin, D. and Adams, R. Firefly Monte Carlo: Exact MCMC with subsets of data. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pp. 543–552, 2014.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. arXiv:1511.05644, 2015.

Nicholls, G. K., Fox, C., and Watt, A. M. Coupled MCMC with a randomized acceptance probability. arXiv:1205.6857, 2012.

Papadimitriou, C. H. and Yannakakis, M. Shortest paths without a map. In *Proceedings of 16th International Colloquium on Automata, Languages and Programming*, pp. 610–620, 1989.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

Péret, L. and Garcia, F. On-line search for solving Markov decision processes via heuristic sampling. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pp. 530–534, 2004.

Quiroz, M., Villani, M., Kohn, R., Tran, M.-N., and Dang, K.-D. Subsampling mcmc — an introduction for the survey statistician. *Sankhya A*, 80(1):33–69, Dec 2018.

Quiroz, M., Kohn, R., Villani, M., and Tran, M.-N. Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*, 114(526):831–843, 2019.

Rainforth, T. Nesting probabilistic programs. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, pp. 249–258, 2018.

Rainforth, T., Cornish, R., Yang, H., Warrington, A., and Wood, F. On nesting Monte Carlo estimators. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4267–4276, 2018.

Ranganath, R., Gerrish, S., and Blei, D. M. Black box variational inference. In *Proceedings of the 17th Conference on Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Rubin, D. B. A case study of the robustness of Bayesian methods of inference: Estimating the total in a finite population using transformations to normality. In Box, G., Leonard, T., and Wu, C.-F. (eds.), *Scientific Inference, Data Analysis, and Robustness*, pp. 213–244. Academic Press, 1983.

Seaman, I. R., van de Meent, J.-W., and Wingate, D. Nested reasoning about autonomous agents using probabilistic programs. arXiv:1812.01569, 2018.

Staton, S., Yang, H., Wood, F., Heunen, C., and Kammar, O. Semantics for probabilistic programming: Higher-order functions, continuous distributions, and soft constraints. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 525–534, 2016.

Stuhlmüller, A. and Goodman, N. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. *Cognitive Systems Research*, 28:80–99, 2014. Special Issue on Mindreading.

Tavares, Z., Zhang, X., Minasyan, E., Burroni, J., Ranganath, R., and Lezama, A. S. The random conditional distribution for higher-order probabilistic inference. arXiv:1903.10556, 2019.

TensorFlow. TensorFlow probability. `https://www.tensorflow.org/probability/`, 2018.

Tolpin, D. Deployable probabilistic programming. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, pp. 1–16, 2019.

Tolpin, D. and Shimony, S. E. MCTS based on simple regret. In *Proceedings of The 26th AAAI Conference on Artificial Intelligence*, pp. 570–576, 2012.

Tolpin, D., van de Meent, J.-W., Yang, H., and Wood, F. Design and implementation of probabilistic programming language Anglican. In *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages*, pp. 6:1–6:12, 2016.

Tran, D., Hoffman, M. D., Saurous, R. A., Brevdo, E., Murphy, K., and Blei, D. M. Deep probabilistic programming. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

van de Meent, J.-W., Paige, B., Tolpin, D., and Wood, F. Black-box policy search with probabilistic programs. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 1195–1204, 2016.

Wingate, D., Goodman, N. D., Roy, D. M., Kaelbling, L. P., and Tenenbaum, J. B. Bayesian policy search with policy priors. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 1565–1570, 2011a.

Wingate, D., Stuhlmüller, A., and Goodman, N. D. Lightweight implementations of probabilistic programming languages via transformational compilation. In *Proceedings of the 14th Conference on Artificial Intelligence and Statistics*, pp. 770–778, 2011b.

Winn, J., Bishop, C. M., Diethe, T., and Zaykov, Y. *Model Based Machine Learning*. 2019. URL `http://mbmlbook.com/`. electronic; retrieved 2019/4/21.

Wood, F., van de Meent, J.-W., and Mansinghka, V. A new approach to probabilistic programming inference. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, pp. 1024–1032, 2014.