Sublinear Approximate Inference for Probabilistic Programs

Yutian Chen[1], Vikash Mansinghka[2] and Zoubin Ghahramani[1]

[1]Machine Learning Group, Department of Engineering, University of Cambridge
[2]Computer Science and Artificial Intelligence Laboratory, MIT

Abstract

Probabilistic programming languages can simplify the development of machine learning techniques, but only if inference is sufficiently scalable. Unfortunately, Bayesian parameter estimation for highly coupled models such as regressions and state-space models still scales badly. This paper describes a sublinear-time algorithm for making Metropolis-Hastings updates to latent variables in probabilistic programs. This approach generalizes recently introduced approximate MH techniques: instead of subsampling data items assumed to be independent, it subsamples edges in a dynamically constructed graphical model. It thus applies to a broader class of problems and interoperates with general-purpose inference techniques. Empirical results are presented for Bayesian logistic regression, nonlinear classification via joint Dirichlet process mixtures, and parameter estimation for stochastic volatility models (with state estimation via particle MCMC). All three applications use the same implementation, and each requires under 20 lines of probabilistic code.

1 Introduction

Machine learning methods can be difficult and time-consuming to design and implement. Probabilistic programming has the potential to mitigate these challenges: short probabilistic programs can represent models from multiple fields (Goodman et al., 2008), thus inference methods developed for arbitrary probabilistic programs immediately apply to a broad class of problems (Mansinghka et al., 2014).

Unfortunately, because of this generality, it has proved difficult to develop scalable inference algorithms for general probabilistic programs. Many architectures exhibit quadratic scaling on problems such as topic modeling Blei et al. (2003) and nonparametric mixture modeling (Neal, 2000); for example, the Bher transformational compiler (Goodman et al., 2008) requires the entire model to be resimulated for each single-site MH transition. The Venture and BLOG languages overcome this by tracking dependencies between random choices, using the resulting factorization of the joint probability density over all variables to recover linear scaling in many common cases. The probabilistic execution trace (PET) graph that tracks dependencies in Venture also handles exchangeable coupling and thus supports algorithms based on $O(1)$ updates to sufficient statistics. However, Bayesian parameter estimation for highly coupled models such as regressions and state-space models still scales badly, as each Metropolis-Hastings transition requires linear time.

This paper describes a sublinear-time algorithm for performing approximate Metropolis-Hastings transitions to latent variables in probabilistic programs with $O(N)$ outgoing dependencies. This algorithm generalizes ideas from recent work on approximate transition operators (Singh et al., 2012; Korattikara et al., 2014; Bardenet et al., 2014): instead of subsampling data items, it subsamples edges in a dynamically constructed graphical model, stochastically ignoring dependencies. The proposed algorithm can be interleaved with state-of-the-art general-purpose inference algorithms for probabilistic programs and thus applies to problems with widely varying structures.

This paper contains three contributions. The first is the algorithm, including its integration into an inference programming language. The second is a new proof of ergodicity for the approximate Markov chain under milder conditions than Korattikara et al. (2014), showing that the bias vanishes as the controlling parameter approaches 0. The third is an empirical demonstration of efficacy and broad applicability, via applications to parametric regression, nonparametric Bayesian mixtures of experts, and state-space models.

2 Background on Inference in Probabilistic Programs

Here we briefly define the key terms needed to describe the subsampled inference technique proposed in this paper. Due to space constraints, we cannot give a detailed or formal description of general inference procedures for probabilistic programs. Readers are referred to Mansinghka et al. (2014) for more details.

Definition 1. A *probabilistic execution trace* (PET) is a directed graph, $\rho = (V, E_\mathrm{d} \cup E_\mathrm{e})$, representing a single realization of a probabilistic model produced by a probabilistic program. V is the set of nodes corresponding to computations that must be performed to generate such a realization. E_d is the set of edges representing *probabilistic* dependencies. E_e is the set of edges representing *existential* dependencies, where the existence of the child node depends on the value of parent node, such as the branches of `if` statements.

Fig. 1 depicts an example of the PET for a Bayesian logistic regression model generated from the program in Fig. 3. Nodes in V may represent the application of a random (e. g. $\mathbf{w}$ and y) or deterministic (e. g. p) procedure; others represent constant variables, symbol look-ups (*Norm*, $\boldsymbol{\mu}$, y_obs, etc), and so on. All the edges in the figure belong to E_d.

Definition 2. The induced graphical model for a PET is a directed graph (V, E_d) representing a single execution of the program, and induces an identical factorization of the joint probability density.

$$p(\rho) = \prod_{n \in V} p(x_n | \mathrm{Par}_{E_\mathrm{d}}(n, \rho)) \tag{1}$$

where x_n is the value of node n and $\mathrm{Par}_{E_\mathrm{d}}(n, \rho) = \{x_{n'} : n' \in V, (n', n) \in E_\mathrm{d}\}$ is the parent set of node n in trace ρ.

2.1 Metropolis-Hastings Sampling Algorithm on Scaffolds

The inference of a probabilistic program can be performed by running a Metropolis-Hastings (MH) sampler sequentially on subsets of V. We define the *principal* set, denoted as $V_p \subseteq V$, as a set of random variables to be sampled at an MH iteration.

Definition 3. Given a PET ρ and a principal set V_p, a *scaffold*, $s(\rho, V_p)$, is a subgraph of ρ whose nodes consist of three mutually exclusive parts:

1. A *target* set, denoted by *target(s)*, containing random choices to re-propose, i.e. V_p, and all nodes whose values deterministically depend on the values of V_p.
2. A *transient* set, *trans(s)*, corresponding to nodes whose existence depends on the values of *target(s)* through edges in E_e.
3. An *absorbing* set, *absorb(s)*, containing all the children of *target(s)* and *trans(s)* in ρ.

The scaffold is the minimum subgraph of ρ that defines a coherent inference problem for V_p. Fig. 1 shows an example of a scaffold when sampling $\mathbf{w}$. If s is a valid scaffold for

[1]Some lookup nodes are not drawn for clarity.

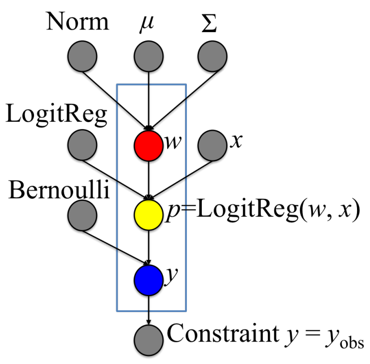


Figure 1: A simplified[1] example of PET representing the Bayesian logistic regression model: $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), y \sim \text{Logit}(y|\mathbf{x}, \mathbf{w})$ with one observation $(\mathbf{x}, \mathbf{y}_{\text{obs}})$. When sampling $\mathbf{w}$, the scaffold s consists of all the colored nodes and the edges between them. Principal nodes, V_p are denoted in red; *target(s)* is the union of red and yellow nodes; *absorb(s)* is denoted in blue. There is no *trans(s)* in this example. Gray nodes are only read from and do not belong to s.

a PET ρ, then *absorb(s)* screens off the *target(s)* and *trans(s)* from the rest of the trace. That is, there is no path to start from any node in *target(s)* or *trans(s)* and reach nodes in $\rho \backslash s$ by following directed edges, except by first intersecting a node in *absorb(s)*. The inference sub-problem may depend on the value of nodes outside s, such as the parents of V_p. They are, nevertheless, not considered as part of s, and the reason will become clear as we explain Eq. 3. The construction of a scaffold is implemented in the probabilistic program as a procedure *constructScaffold*$(\rho, V_p) \to s$ via a simple graph walk.

Given a set of variables to sample, V_p, the MH algorithm is performed by first removing all the nodes in $target(s) \cup trans(s)$ from ρ in some order π, and then propose new values for the target set, $\{x'_n, n \in target(s)\}$ in the reverse order π' and create a new set of transient nodes when necessary, $trans'(s)$. The values of the absorbing nodes and other nodes outside s remain the same, and we denote the new trace as ξ. The acceptance probability of this proposal is computed as

$$P_a = \min\left\{1, \frac{p(\xi)\prod_{n\in target(s)\cup trans(s)} q(x_n|\rho)}{p(\rho)\prod_{n\in target(s)\cup trans'(s)} q(x'_n|\xi)}\right\} \tag{2}$$

$$= \min\Bigg\{1, \prod_{n\in target(s)} \frac{p(x'_n|\text{Par}(n,\xi))}{q(x'_n|\xi)} \frac{q(x_n|\rho)}{p(x_n|\text{Par}(n,\rho))} \prod_{n\in absorb(s)} \frac{p(x_n|\text{Par}(n,\xi))}{p(x_n|\text{Par}(n,\rho))} \prod_{n\in trans(s)} \frac{p(x'_n|\text{Par}(n,\xi))}{q(x'_n|\xi)} \prod_{n\in trans'(s)} \frac{q(x_n|\rho)}{p(x_n|\text{Par}(n,\rho))}\Bigg\} \tag{3}$$

where we plugged in Eq. 1 in the second line. $q(x'_n|\xi)$ denotes the proposal distribution of node n given all the nodes in the new trace ξ that have been generated before n. Any terms for nodes outside s are canceled in the above equation, and therefore a scaffold s describes the set of all nodes that are involved in the computation of the acceptance probability.

Notice that for every node n in the scaffold, the product in Eq. 3 indexed by n can be written as a product between two terms, one depending on ρ and the other depending on ξ. Denote the product for n as weight $w_n = w_n^{detach} w_n^{regen}$ where w_n^{detach} refers to the term depending on ρ, and w_n^{regen} depending on ξ (either term is defined as 1 at absence). The

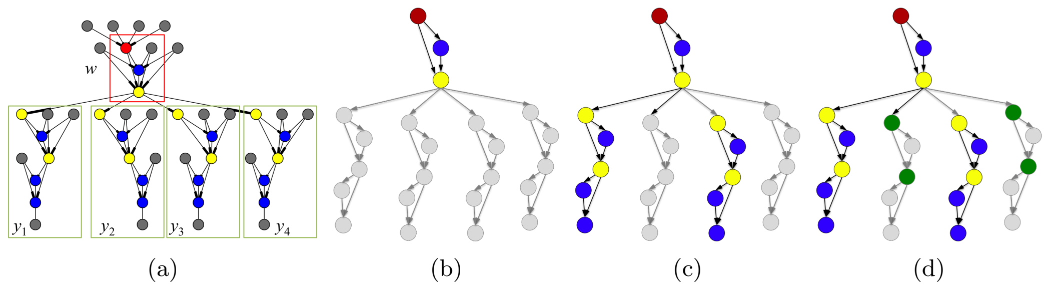


Figure 2: (a) The PET of a Bayesian logistic regression model with 4 observations, colored to encode the scaffold for a proposal on the weight variable $\mathbf{w}$; the corresponding probabilistic program is shown in Fig. 3. Sections corresponding to the weight and to individual observations are labeled. (b) The global section of the scaffold partition, with the border node in yellow. See Sec. 3.1 for details. (c) A subsampled scaffold with two local sections; compare to (a). Inference on this scaffold only considers two observations, not all four. (d) Nodes whose values are stale after a subsampled transition. See Sec. 3.4 for details.

acceptance probability P_a can then be written as

$$P_a = \min\left\{1, \prod_{n \in s} w_n\right\} = \min\left\{1, \prod_{n \in s} w_n^{detach} \prod_{n \in s} w_n^{regen}\right\} \tag{4}$$

In a probabilistic program it is computed with two operations described as follows:

$detach(\rho, s) \rightarrow (\pi, w^{detach}, \{x_n\})$ is a procedure that traverses s in an order π, removes *target(s)* and *trans(s)* from ρ, and calculates $w^{detach} = \prod_{n \in s} w_n^{detach}$. It also returns the original values of the detached nodes. In the example of Fig. 1, *detach* would remove $\mathbf{w}$ and p, and return the *detach* weight multiplied over the nodes $\mathbf{w}$, p, and y.

$regen(\rho, s, \pi') \rightarrow (\xi, w^{regen})$ takes ρ, s and reverse order π', proposes new values for *target(s)*, deals with possible creation of transient nodes, and calculates $w^{regen} = \prod_{s \in n} w_n^{regen}$. A variant can be used to restore the nodes to their original values returned from *detach* when a proposal is rejected. In Fig. 1, *regen* would propose new values for $\mathbf{w}$ and p, and return the *regen* weight multiplied over nodes $\mathbf{w}$, p, and y.

After P_a is obtained, a uniform random variable u is drawn from the interval $[0, 1]$ and the new trace ξ is accepted if $u < P_a$, or rejected otherwise. In the latter case, one has to call *detach* and *regen* again to detach proposed values and restore the old trace ρ.

3 Sublinear Time Inference via Subsampling Scaffolds

The accept/reject step guarantees that the desired distribution over PETs is left invariant, but can be unacceptably slow when any variable in V_p has too many dependencies. For example, consider the PET of a Bayesian logistic regression model with $N = 4$ observations, shown in Fig. 2a, and the scaffold for sampling the weights, i.e. $V_p = \{\mathbf{w}\}$. When a random variable in V_p has N dependencies outside of V_p — be their observations or other latent variables — then the scaffold s will be of size $O(N)$, e.g. $|absorbing(s)| \in O(N)$. It will take $O(N)$ time to construct, $O(N)$ time to detach and regenerate all the nodes, and in the case of a rejection, $O(N)$ time to revert. This is often impractical.

This paper presents an approximate MH transition whose runtime scales sublinearly. The approximation introduces some bias into the stationary distribution, but the experiments show this can still improve accuracy by allowing for more transitions.

3.1 A Global/Local Partition

The scaffold from Fig. 2a can be partitioned into two parts:

1. A global section, denoted as *global*, that contains the principal set V_p.

2. A set of N local sections, $\{local_n\}_{n=1}^{N}$, which share a similar structure and represent the N dependencies.

The size of the scaffold grows with the number of local sections while the size of each section remains constant.

Our algorithm applies to scaffolds of this form, where in addition (1) V_p contains a single node, (2) all the local dependencies are connected to V_p via a common node with a single link, which is V_p or deterministically depends on it. Under these assumptions, the single node in (2), e.g. the top yellow node in Fig. 2b, separates *global* from *local* sections and has N children. We call it the border node, denoted by b. We also assume that $trans(s) = \emptyset$, i.e. making proposals within the scaffold does not change the structure of the PET contained within it. Note that this does *not* exclude models with dynamically evolving structure; only approximate transitions are prohibited from introducing structural changes. These are mild restrictions, still admitting a broad class of probabilistic programs.

We define a procedure $findBorder(V_p, \rho) \to b$ that traverses the trace ρ from V_p and returns the first node with multiple branches. The global section of scaffold s is then constructed by a procedure $constructBoundedScaffold(\rho, V_p, b) \to global$ which is similar to *constructScaffold* except that it stops when encountering b. A local section is constructed with the regular $constructScaffold(\rho, n)$ procedure for any $n \in children(b)$. These procedures partition the scaffold into a *global* and N *local* sections.

3.2 Austerity MCMC for Probabilistic Programs with Partitioned Scaffolds

Korattikara et al. (2014) proposed the austerity idea, a sublinear-time algorithm to incrementally approximate the acceptance probability until the approximation error is under a tolerance level. They applied the algorithm to the posterior inference problem for Bayesian models with N *iid* observations. We adopt this idea for the inference problem on PETs by proposing a sublinear-time algorithm for approximate MH transitions on partitioned scaffolds.

Given a partition of the scaffold, we can factorize the acceptance probability in Eq. 4 as

$$P_a = \min\left[1, \left(\prod_{n\in global} w_n\right)\prod_{i=1}^{N}\left(\prod_{n\in local_i} w_n\right)\right] \tag{5}$$

After drawing a uniform random variable u, the inequality $u < P_a$ is equivalent to $\mu_0 < \mu$ with μ_0 and μ defined as follows:

$$\mu_0 = \frac{1}{N}(\log u - \sum_{n\in global} \log w_n), \mu = \frac{1}{N}\sum_{i=1}^{N} l_i, \tag{6}$$

where $l_i = \sum_{n\in local_i} \log w_n$. The term μ_0 is usually easy to compute, while μ is an average over N terms. We follow Korattikara et al. (2014) and reformulate the decision problem as a statistical test with the hypothesis $H_1 : \mu > \mu_0$, and $H_2 : \mu < \mu_0$. Given a tolerance level ε, we conduct the hypothesis testing by iteratively sampling mini-batches of m local sections, and updating the estimate of μ until a high confident result is reached. The algorithm is given in Alg. 1. The last term in $\sqrt{\cdot}$ in step 7 is the finite population correction factor introduced by sampling without replacement. For problems with a large data set, the approximate MH method requires only a small fraction of the data to make a high quality decision, and therefore can significantly speed up the MH algorithm.

By comparing the definition of μ_0 and μ with those in Korattikara et al. (2014), one can see that μ_0 in both methods contains the same prior probability of the target variables and the proposal distribution. However, the quantities μ in our method has a more general meaning. In Korattikara et al. (2014), each l_i refers to the log-ratio of the likelihood of one *iid* data point, while in our PET setting each l_i is the product of weights associated with a local partition of absorbing nodes. They do not have to be observed data, and can even have strong dependence between each other; our third experiment illustrates this case.

Algorithm 1 Sequential Test for MH Decison

```
1: procedure SequentialTest($\mu_0$, set $\mathcal{X} = \{l_i\}$ of a size $N$, mini-batch size $m$, error tolerance $\varepsilon$)
2:     Set of observed data $\mathcal{Y} \leftarrow \emptyset$, $n \leftarrow 0$.
3:     repeat
4:         Draw a subset of $m$ $l_i$'s, $X_m \subseteq \mathcal{X}$
5:         $\mathcal{X} \leftarrow \mathcal{X} \backslash X_m$, $\mathcal{Y} \leftarrow \mathcal{Y} \cup X_m$, $n \leftarrow n + m$
6:         Estimate mean, $\hat{\mu}$, and std, $s_l$, of $\mathcal{Y}$.
7:         std of $\hat{\mu}$ is $s \leftarrow \frac{s_l}{\sqrt{n}} \sqrt{1 - (n-1)/(N-1)}$
8:     until p-value of $t_{n-1}(|\frac{\hat{\mu} - \mu_0}{s}|)$ is lower than $\varepsilon$
9:     Accept $H_1$ if $\hat{\mu} > \mu_0$ else $H_2$
10: end procedure
```

The same error analysis in Korattikara et al. (2014) applies to our algorithm. Additionally, we prove that even when the condition for the central limit theorem does not approximately hold for Austerity MCMC, the bias of the approximate Markov chain will still diminish as the tolerance parameter ε defined in Alg. 1 approaches 0 under a set of milder conditions.

Let θ and θ^* be the current and proposed value of the target variable, m be the size of a mini-batch, and $P_{a,\varepsilon}(\theta, \theta^*)$ be the probability of accepting using the sequential test procedure with ε. We have the following theorem with proof in the appendix.

Theorem 1. *If the number of identical values of l_i's is less than m for any pair (θ, θ^*), the likelihood function is continuous w.r.t. θ, and the domain of θ, Θ, is a compact set, then there exists a function $\delta(\varepsilon)$ such that $|P_{a,\varepsilon}(\theta, \theta^*) - P_a(\theta, \theta^*)| \leq \delta(\varepsilon), \forall \theta, \theta^* \in \Theta$, and $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$.*

This theorem states that the difference between the approximate and exact acceptance probability goes to zero uniformly for θ, θ^*. The first two conditions are usually trivial to satisfy when there are not many duplicate local sections. The compactness condition might be violated if θ has an unbounded domain. Nevertheless, its impact to the bias of the stationary distribution is negligible when the actual samples of θ reside in a restricted region. Alternatively, one can impose a truncated prior distribution for θ. As long as the truncation is sufficiently large, the impact to the inference procedure can be ignored. It is worth mentioning that if θ is a discrete variable with a finite domain, Theorem 1 also holds with the first condition.

When the exact MH kernel satisfies regularity conditions in Pillai and Smith (2014), we can show the following corollary:

Corollary 2. *The Markov chain of the approximate MH algorithm is uniformly ergodic for any sufficiently small ε. Its stationary distribution approaches the target distribution as $\varepsilon \to 0$.*

3.3 Subsampling the Scaffold

In order to obtain sublinear running time, we should avoid any operations with a $O(N)$ complexity, including the construction of the entire scaffold. Our proposed algorithm interleaves the scaffold construction with the *detach* & *regen* operations. This way the local sections of the scaffold will not be created until the sequential test requires more data to improve its estimate. Our subsampled MH algorithm is given in Alg. 2. Fig. 2b and 2c illustrate the state after step 4 and after two local partitions are constructed respectively.

3.4 Update stale nodes on demand

When a proposed move is accepted, all the deterministic nodes of a scaffold should be updated accordingly in the standard MH algorithm. However, with the subsampling approach, it is not guaranteed to visit every local node after one MH step. This may leave some nodes with old values and break the deterministic dependencies. The green nodes in Fig. 2d are

Algorithm 2 Sublinear-Time Metropolis-Hastings Algorithm with Scaffolds

```
1: procedure SubsampledMH(Principal nodes $V_p$, Trace $\rho$, mini-batch size $m$, error tolerance $\varepsilon$)
2:     Sample $u \sim \text{Uniform}[0, 1]$.
3:     $b \leftarrow \text{findBorder}(V_p, \rho)$, $N \leftarrow \#(b\text{'s children})$
4:     $global \leftarrow \text{constructBoundedScaffold}(\rho, V_p, b)$
5:     $\sum_{n \in global} \log w_n \leftarrow \text{Detach\&Regen}(\rho, global)$ and compute $\mu_0$ with Eq. 6.
6:     $\mathcal{Y} \leftarrow \emptyset$, $n \leftarrow 0$
7:     repeat
8:         Sample $m$ children of *border*'s w/o replacement as local principal nodes $\{p_i\}$.
9:         for all $p_i$ do
10:            $local_i \leftarrow \text{constructScaffold}(\rho, p_i)$
11:            $l_i \leftarrow \text{Detach\&Regen}(\rho, local_i)$
12:        end for
13:        $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{l_i\}$, update $n$, $\hat{\mu}$, and $s$ as in Alg. 1
14:    until The p-value falls below $\varepsilon$ as in Alg. 1
15:    Accept if $\hat{\mu} > \mu_0$ else Reject and Restore *global*
16: end procedure
```

examples of the stale nodes caused by the subsampled MH method. In order to solve the broken consistency, one can revisit every local partition and update their value after every call to the sampler. However, that takes $O(N)$ time and will forgo any benefit earned by the subsampling method. We proposed a lazy updating approach, that is, whenever the subsampled MH algorithm is involved in the inference program, after a scaffold (global or local section) is constructed, the program first recomputes the values of the deterministic nodes and then call the usual detach&regen procedure. That retains the sublinear running time, and the overhead is negligible compared to the speed up of subsampled MH in the experiments.

4 Experiments

We implemented our algorithm in a lightweight research variant of the Venture probabilistic programming system, written in unoptimized Python, and applied it to multiple problems. Asymptotic scaling and relative comparisons between standard and subsampled Metropolis-Hastings are meaningful, but absolute runtimes reflect large constant factor overheads that are straightforward to reduce.

Table 1: Overview of models used in experiments, including scaling parameters for exact MH. $N_k = \sum_{i=1}^{N} \mathbb{I}[z_i = k]$.

Model	Domain of Sublinear MH	Scaling
BayesLR	$\mathbf{w} \sim p(\mathbf{w}) \prod_{i=1}^{N} \text{Logit}(y\|\mathbf{x}_i, \mathbf{w})$	N
JointDPM	$\mathbf{w}_k \sim p(\mathbf{w}_k) \prod_{i:z_i=k} \text{Logit}(y\|\mathbf{x}_i, \mathbf{w})$	N_k
SV	$\phi/\sigma \sim p(\phi/\sigma) \prod_{t=1}^{T} \mathcal{N}(h_t\|\phi h_{t-1}, \sigma^2)$	T

4.1 Bayesian Logistic Regression

We first demonstrate that our general-purpose implementation recovers the time/accuracy advantages of the custom implementation of austery for Bayesian logistic regression from Korattikara et al. (2014). Our Bayesian logistic regression model uses a standard isotropic Gaussian prior on the weights:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_D), \quad y_i \overset{\text{iid}}{\sim} \text{Logit}(y|\mathbf{x}_i, \mathbf{w}) \tag{7}$$

We evaluate performance on a classification task on the MNIST digit image data. We train on 12214 images of '7' and '9', each transformed to a 50-dimensional feature vector via

```
[assume w   (scope_include 'w 0 (multivariate_normal {mu} {Sig}))]
[assume y_x (lambda (x) (bernoulli (linear_logistic w x)))]

for n in 1...N: #load data
  # features X[n] = [x0_n x1_n ...], class Y[n]
  [observe (y_x {X[n]}) {Y[n]}]

# do T iterations of subsampled MH with a Gaussian drift proposal
# of bandwidth sig
[infer (subsampled_mh w all {nbatch} {eps} 'drift {sig} {T}]
[infer (mh w all {T})] #standard MH provided for reference
```

Figure 3: Probabilistic program for the Bayesian logistic regression model, data, and subsampled inference scheme. {} denotes external parameters.

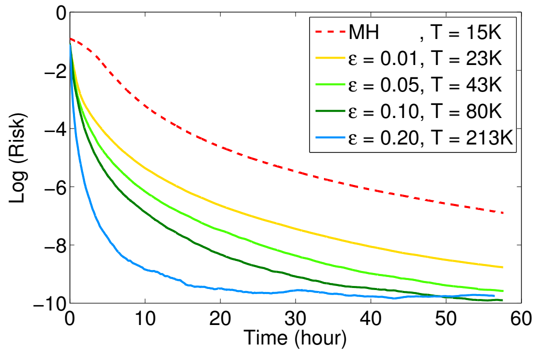


Figure 4: Risk of predictive mean vs. running time for Bayesian logistic regression on MNIST. T shows the number of samples drawn in about 60 hours.

normalization and principal component analysis. We evaluate predictive accuracy versus computation time on 2037 test images. The Venture programs for the model and inference are given in Fig. 3. We use the same parameter and data settings as in Korattikara et al. (2014), and run the inference algorithm with the same random walk proposal distribution. We use a smaller mini-batch size of 100. The risk of predictive mean over an extensive run over 50 hours are given in Fig. 4. See Korattikara et al. (2014) for the definition of risks. The predictive performance of subsampled MH increases significantly faster than the standard Metropolis-Hastings. It can make more than one order of magnitude more transitions, and takes 5 hours to reach the risk achieved by standard Metropolis-Hastings after 50 hours.

4.2 Joint Dirichlet Process Mixture Models

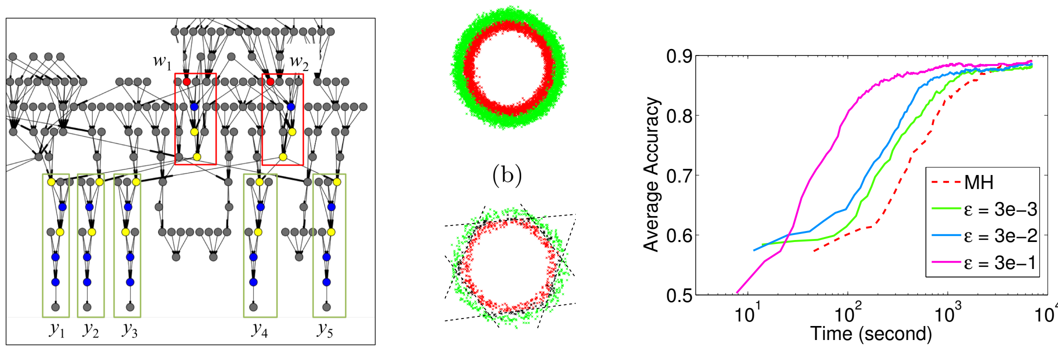
We also assess performance on a joint Dirichlet process mixture (JointDPM) model (Wade et al., 2014), a flexible nonlinear classifier that combines logistic regressions using nonparametric Bayes. This model uses a Dirichlet process mixture of Gaussians to model input features, where each component has a distinct set of logistic regression weights to classify the vectors it contains. More formally, we have

$$(\mathbf{x}_i, y_i)|P \overset{\text{iid}}{\sim} f(\mathbf{x}, y|P), \quad P \sim DP(\alpha P_0) \tag{8}$$

where P_0 is the base measure. Every sample of P is a countable mixture model

$$f(\mathbf{x}, y|P) = \sum_{k=1}^{\infty} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)\text{Logit}(y|\mathbf{x}, \mathbf{w}_k) \tag{9}$$

with each normal distribution parameter pair $(\boldsymbol{\mu}_k, \Sigma_k)$ sampled from a conjugate normal-inverse-Wishart prior. The regression parameter $\mathbf{w}_k$ is sampled from an isotropic Gaussian prior as in the Bayesian logistic regression model. The graphical model is shown in Fig. 7a.



(a) (b) (c) (d)

Figure 5: Joint Dirichlet Process mixture model of logistic regression experts, evaluated on synthetic data. (a): PET with scaffolds for two weight parameters. (b): training data. (c): prediction with $\varepsilon = 0.3$ after 2 hours. 6 clusters are found with decision boundaries shown with dashed lines. Black dots are incorrect predictions. (d): Accuracy of averaged predictions in time vs running time in log domain.

We can write the model in under 20 lines of probabilistic code (Fig. 6), collapsing the component models by marginalizing out (μ_k, Σ_k) and collapsing the DP into a CRP. A trace fragment containing two scaffolds, each containing the regression weights for one cluster, is shown in Fig. 5a. In this model, the number of simultaneous instantiations of the austerity scheme is an object of inference.

We use subsampled Metropolis-Hastings to accelerate inference over the parameters w_k for each cluster's logistic regression model. The CRP hyperparameter α and N component assignment variables z_i must also be inferred, but as inference for these variables already requires constant time due to properties of the PET, approximate transitions are only used for the w_ks. We set the parameters of our inference program to allocate roughly equivalent computation time to sampling w_k and to a series of transitions to randomly chosen z_i's; the balance is struck using the `step_z` parameter (see Fig. 6). Because subsampled MH runs faster than MH, the w's are updated more frequently than standard MH to maintain the balance.

We apply JointDPM to a synthetic data set with 10,000 data points as shown in Fig. 5b. A snapshot of the prediction results on another 1,000 test points and the accuracy of the prediction averaged over the Markov chain are shown in Fig. 5c and 5d. The sublinear algorithm with $\varepsilon = 0.3$ reaches the same accuracy as exact MH in 10x less time.

4.3 Joint Parameter and State Estimation in a State-Space Model

We also applied our implementation to a stochastic volatility model for a time series with two parameters:

$$x_t = \exp(h_t/2)\varepsilon_t,\ h_t \sim \mathcal{N}(\phi h_{t-1}, \sigma^2),\ \varepsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$$

where we set $h_0 = 0$ and assign a Beta(5,1) and an inverse Gamma(5, 0.05) distribution for parameter ϕ and σ^2 respectively. This model is a state-space model and has unknown hidden states as well as unknown parameters. Fig. 7b shows the graphical model. Fig. 8a illustrates the PET of the model with $t = 1, 2, 3$. Note that there are dependencies between the subsampled local partitions in this problem, transitions from h_{t-1} to h_t. The dynamics are sensitive to the precise values of ϕ and σ^2 and also the hidden states, yielding a challenging joint inference problem.

We generate a synthetic data set of 200 series of length 5 with correlation $\phi = 0.95$ and noise $\sigma = 0.1$. We apply particle Gibbs to sample latent states and (subsampled) MH to sample parameters ϕ and σ. Fig. 8b and c show the histogram of the samples after the burn-in period, and Fig. 8d shows the autocorrelation of the samples measured in running time.

```
# fixed: dimensionality D, hypers mu_w sig_w m0 k0 v0 S0
[assume alpha (scope_include 'hypers 0 (gamma 1 1))]
[assume crp   (make_crp alpha)]
[assume z     (mem (lambda (i) (scope_include 'z i (crp))))]
[assume w (mem (lambda (z) (scope_include 'w z
                            (multivariate_normal mu_w sig_w))))]
[assume c (mem (lambda (z)
                (make_collapsed_multivariate_normal m0 k0 v0 S0)))]
[assume x (lambda (i) ((c (z i))))]
[assume y (lambda (i x) (bernoulli (linear_logistic (w (z i)) x)))]
for i in 1...X: # load data
    [observe (x i) {X[i]}] #X[i] is ith feature vector
    [observe (y i) {Y[i]}] #Y[i] is ith class label

# T steps of MH for hyperparams, single-site gibbs for z, and
# subsampled MH over weights for a randomly chosen expert
[infer (cycle {T} ((mh alpha all 1)
                  (gibbs z one step_z)
                  (subsampled_mh w one {Nbatch} {eps}
                                 'drift {sigma} 1)) 1)]
```

```
[assume sig (scope_include 'sig 0 (sqrt (inv_gamma 5 0.05)))]
[assume phi (scope_include 'phi 10 1)]
[assume h   (mem (lambda (t) (scope_include 'h t
              (if (<= t 0) 0
                  (normal (* phi (h (- t 1))) sig)))))]
[assume x (lambda (t) (normal 0 (/ (h t) 2)))]
for t in 1...T:
    [observe (x t) {X[t]}] # X[t] is the observation at index t

# state estimation via particle gibbs over subsequences of length L
for n in 1...N-L:
    [infer (pgibbs h (ordered_range h h+L) P 1)]
# subsampled MH inference on the parameters
[infer (cycle ((subsampled_mh sig 0 {Nbatch} {eps} 1)
              (subsampled_mh phi 0 {Nbatch} {eps} 1)) 1)]
```

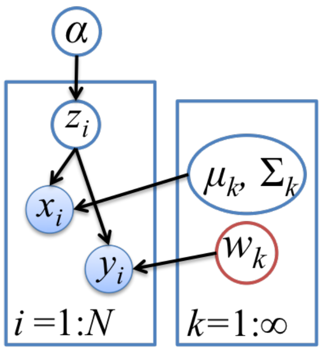
Figure 6: (top) Probabilistic program containing model and inference scheme for the joint DP mixture of logistic regression experts. (bottom) Probabilistic program and joint state and parameter estimation scheme for the stochastic volatility model, combining subsampled Metropolis-Hastings with particle Gibbs.

Due to the high correlation between latent state variables, the overall mixing rate highly depends on the mixing rate in h_t's. Therefore we assign 10 more times of computation time to sample h_t than other variables. We observe that the gain of subsampled MH is not as significant as previous experiments due to the slow mixing of latent states. Nevertheless, subsampled MH still obtains about twice the efficiency of exact MH without introducing significant bias.

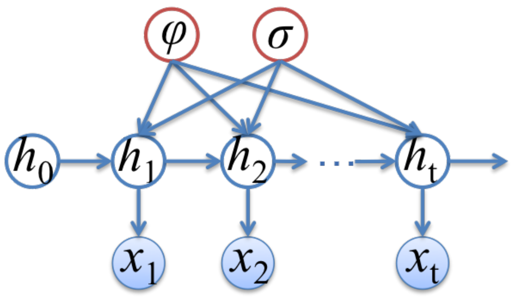
5 Discussion

This paper shows that it is feasible to define sublinear approximate MH transitions for highly-coupled variables in general probabilistic programs and integrate them into an inference programming language. The results also suggest that it is useful: a single unoptimized implementation applies to a broad class of problems, going beyond previous approximate MH schemes, and yields significant (2x-10x) improvements in runtime at no cost in accuracy.

The algorithm presented here uses frequentist statistical inference to accelerate Bayesian inference. It is interesting to consider alternatives to Algorithm 1, where the sequential test is replaced with a model-based inference — potentially itself written as a probabilistic program. It may also be interesting to consider probabilistic programming adaptations of

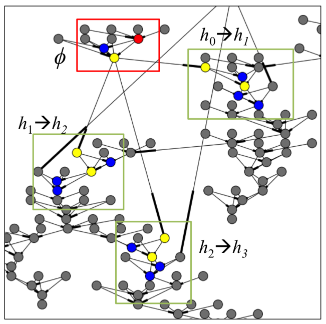


(a)

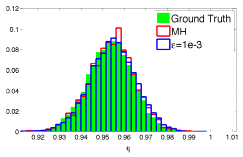


(b)

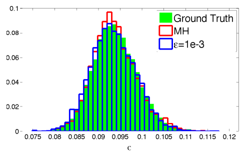
Figure 7: Graphical model of (a) Joint DPM (b) Stochastic volatility model. Subsampled MH applies to red nodes. The algorithm from this paper applies to problems with diverse dependency structures.



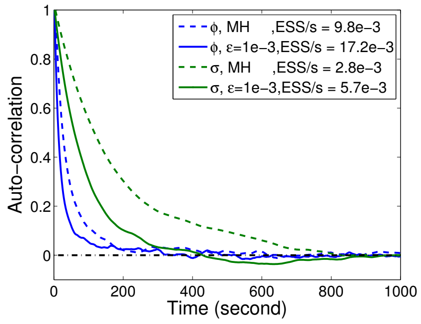
(a)



(b) Histogram of ϕ.



(c) Histogram of σ.



(d)

Figure 8: Stochastic Volatility Model. (a): PET and the scaffold for sampling ϕ. (b, c): Histogram of the samples of ϕ and σ from ground truth (green), exact MH (red) and subsampled MH with ε=1e-3 (blue). (d): Auto-correlation of samples for exact (dashed) and subsampled MH (solid). ESS per second shown in the legend.

other techniques for accelerating inference stochastically ignoring or suppressing dependencies, such as decayed MCMC filtering (Golightly et al., 2014).

Probabilistic programs can represent model classes, datasets, queries and custom inference strategies from many different application domains. Some domains will require faithful, fully Bayesian inference. In these cases, approximate MH may be useful for accelerating burn-in. For other problems, such as parameter estimation for motion models in robotics or machine learning from internet user behavior, speed may be preferred to accuracy, and results from sublinear approximate MH transitions may be adequate on their own. We hope that by integrating sublinear, approximate transitions into a higher-order probabilistic programming system, we have taken a step towards making it more practical to use probabilistic programming in both these kinds of applications.

References

Rémi Bardenet, Arnaud Doucet, and Chris Holmes. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *Proceedings of The 31st International Conference on Machine Learning*, pages 405–413, 2014.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

Andrew Golightly, DanielA. Henderson, and Chris Sherlock. Delayed acceptance particle MCMC for exact inference in stochastic kinetic models. *Statistics and Computing*, pages 1–17, 2014. ISSN 0960-3174.

Noah Goodman, Vikash Mansinghka, Daniel Roy, Keith Bonawitz, and Joshua Tenenbaum. Church: a language for generative models. In *Proceedings of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pages 220–229, Corvallis, Oregon, 2008. AUAI Press.

Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC land: Cutting the metropolis-hastings budget. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 181–189, 2014.

Vikash Mansinghka, Daniel Selsam, and Yura Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

R.M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:283–297, 2000.

Natesh S Pillai and Aaron Smith. Ergodicity of approximate MCMC chains with applications to large data sets. *arXiv preprint arXiv:1405.0182*, 2014.

Robert J Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, pages 39–48, 1974.

Sameer Singh, Michael Wick, and Andrew McCallum. Monte Carlo MCMC: efficient inference by approximate sampling. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1104–1113. Association for Computational Linguistics, 2012.

Sara Wade, David B. Dunson, Sonia Petrone, and Lorenzo Trippa. Improving prediction from Dirichlet process mixtures via enrichment. *Journal of Machine Learning Research*, 15:1041–1071, 2014.

A Proof of Theorems 1

Proof. Given any value of (θ, θ^*), since the size of the set $\{l_i\}$ is finite, it follows from the first premise of the theorem that the estimated standard deviation of l_i will be lower bounded by a positive number $s_l \geq s_l^* > 0$ for any subset of $\{l_i\}$.

In the accept/reject phase of a Metropolis-Hastings iteration, given the uniform random variable u, the sequential test algorithm will make a wrong decision if at some step in the sequence of tests $\hat{\mu}-\mu_0 > st_{n-1}^{-1}(1-\varepsilon)$ while the true mean $\mu \leq \mu_0$ or $\hat{\mu}-\mu_0 < -st_{n-1}^{-1}(1-\varepsilon)$ while $\mu > \mu_0$, where t_n^{-1} is the inverse CDF of student t distribution with a degree of freedom n. Letting $C = \max_i\{|l_i|\}$, we can bound the probability of making an error at step t by applying the one side Hoeffding's inequality with adjustment for sampling without replacement (Serfling, 1974) as

$$\begin{aligned}
&P\left(\hat{\mu} - \mu_0 > st_{n-1}^{-1}(1-\varepsilon)\right) \\
&\leq P\left(\hat{\mu} - \mu > \frac{s_l^*}{\sqrt{n}}\sqrt{1 - \frac{n-1}{N-1}}t_{n-1}^{-1}(1-\varepsilon)\right) \\
&\leq \exp\left(-\frac{1}{2}\left(\frac{s^* t_{n-1}^{-1}(1-\varepsilon)}{C}\right)^2\right) \\
&\overset{\text{def}}{=} f_1(\varepsilon, n, \theta, \theta^*) \qquad (10)
\end{aligned}$$

when $\mu \leq \mu_0$. In the fist inequality we plugged in the assumption $\mu \leq \mu_0$, the definition of s in Step 8 of Alg. 2 and the lower bound of s_l. In the second inequality we applied the concentration bound. It is obvious to observe that the error is also upper bounded by f_1 when $\mu > \mu_0$. Notice that $f_1(\varepsilon, n, \theta, \theta^*) \to 0$ as $\varepsilon \to 0, \forall n, \theta, \theta^*$.

We can furthur bound the total probability of making a wrong decision in the sequential test given u, denoted as $\mathcal{E}(u, \varepsilon)$, using the union bound:

$$\begin{aligned}
\mathcal{E}(u, \varepsilon, \theta, \theta^*) &= P\left(\cup_t \{\text{wrong decision at step } t\}\right) \\
&\leq \sum_{t=1}^{\lceil N/n \rceil} f_1(\varepsilon, mt, \theta, \theta^*) \overset{\text{def}}{=} f_2(\varepsilon, \theta, \theta^*) \qquad (11)
\end{aligned}$$

We omit the dependency of f_2 on N and m. Again, we have $f_2(\varepsilon, \theta, \theta^*) \to 0$ as $\varepsilon \to 0, \forall \theta, \theta^*$.

Denote the acceptance probability of our approximate Markov chain at (θ, θ^*) as $P_{a,\varepsilon}$ and we can bound its error as

$$\begin{aligned}
&|P_{a,\varepsilon}(\theta, \theta^*) - P_a(\theta, \theta^*)| \\
&= \left|\int_0^1 P(A|u)\mathrm{d}u - \int_0^{P_a} \mathrm{d}u\right| \\
&= \left|\int_{P_a}^1 P(A|u)\mathrm{d}u - \int_0^{P_a} (1 - P(A|u))\mathrm{d}u\right| \\
&\leq \int_0^1 \mathcal{E}(u, \varepsilon, \theta, \theta^*)\mathrm{d}u \leq f_2(\varepsilon, \theta, \theta^*) \qquad (12)
\end{aligned}$$

where A denotes the event that the sequential test procedure accepts the proposal. So $P(A|u)$ with $u > P_a$ is the probability of accepting a proposal while we should reject it and $1 - P(A|u)$ with $u < P_a$ is the probability of rejecting a proposal when we should accept it.

Following the second premise of the theorem, the value of l_i as a function of (θ, θ^*) which is defined as $\log P(x_i|\theta^*) - \log P(x_i|\theta)$ is continuous. Therefore, there exist bounds s^* and C that are continuous w.r.t. (θ, θ^*). So is the upper bound $f_2(\theta, \theta^*)$.

Now combined with the last premise of the theorem, we can conclude the proof by claiming that function f_2 will achieve its maximum in the domain of $\Theta \times \Theta$, denoted as $\delta(\varepsilon)$, and because the function f_2 approaches 0 everywhere in the compact set $\Theta \times \Theta$ as $\varepsilon \to 0$, $\delta(\varepsilon)$ also approaches 0 with ε. $\square$

B Proof of Corollary 2

Proof. Denote the proposal distribution with $q(\theta, \theta')$. For a variable defined in a compact space Θ with measure Ω, the error of expected rejection probability of the approximate Markov chain is bounded by

$$|P_{r,\varepsilon} - P_r| = \left| \int_{\theta'} (-P_{a,\varepsilon} + P_a) q(\theta, \theta') \mathrm{d}\Omega(\theta') \right| \leq \delta(\varepsilon) \tag{13}$$

The transition kernel of M-H is $\mathcal{T}(\theta, \theta') = P_a q(\theta, \theta') + P_r \delta(\theta, \theta')$, where $\delta(\theta, \theta')$ denotes the Dirac delta function. We can bound the total variation distance between the approximate M-H kernel and the exact kernel as

$$\begin{aligned}
&\|\mathcal{T}_\varepsilon(\theta, \cdot) - \mathcal{T}(\theta, \cdot)\|_{\mathrm{TV}} \\
&= \frac{1}{2} \int_{\theta'} \Big| \left(P_{a,\varepsilon}(\theta, \theta^*) - P_a(\theta, \theta')\right) q(\theta, \theta') \\
&\qquad\qquad + (P_{r,\varepsilon} - P_r)\delta(\theta, \theta') \Big| \mathrm{d}\Omega(\theta') \\
&\leq \delta(\varepsilon), \forall \theta \in \Theta
\end{aligned} \tag{14}$$

Since $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$, for any sufficiently small ε we can apply the Lemma 3.6 of Pillai and Smith (2014) to prove the uniform ergodicity and obtain the convergence rate. □