Learning Fast-Mixing Models for Structured Prediction

Jacob Steinhardt JSTEINHARDT@CS.STANFORD.EDU
Percy Liang PLIANG@CS.STANFORD.EDU
Stanford University, 353 Serra Street, Stanford, CA 94305 USA

Abstract

Markov Chain Monte Carlo (MCMC) algorithms are often used for approximate inference inside learning, but their slow mixing can be difficult to diagnose and the resulting approximate gradients can seriously degrade learning. To alleviate these issues, we define a new model family using strong Doeblin Markov chains, whose mixing times can be precisely controlled by a parameter. We also develop an algorithm to learn such models, which involves maximizing the data likelihood under the induced stationary distribution of these chains. We show empirical improvements on two inference tasks.

1. Introduction

Conventional wisdom suggests that rich features and highly-dependent variables necessitate intractable inference. Indeed, the dominant paradigm is to first define a joint model, and then use approximate inference (e.g., MCMC) to learn that model. While this recipe can generate good results in practice, it has two notable drawbacks: (i) diagnosing convergence of Markov chains is extremely difficult (Gelman & Rubin, 1992; Cowles & Carlin, 1996); and (ii) approximate inference can be highly suboptimal in the context of learning (Wainwright, 2006; Kulesza & Pereira, 2007).

In this paper, we instead use MCMC to *define* the model family itself: for a given T, we construct a family of Markov chains using arbitrary rich features, but whose mixing time is *guaranteed* to be at most $O(T)$. The corresponding stationary distributions determine the model family, with larger T leading to more expressive model families. We can think of our Markov chains as parameterizing a family of "computationally accessible" distributions, where the amount of computation is controlled by T.

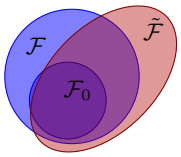
Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

Concretely, suppose we are performing a structured prediction task from an input x to an output y. We construct Markov chains of the following form, called *strong Doeblin chains* (Doeblin, 1940):

$$\tilde{A}_\theta(y_t \mid y_{t-1}, x) = (1-\epsilon)A_\theta(y_t \mid y_{t-1}, x) + \epsilon\, u_\theta(y_t \mid x), \tag{1}$$

where ϵ is a mixture coefficient and θ parameterizes A_θ and u_θ. For intuition, think of u_θ as a simple tractable model and A_θ as Gibbs sampling in a complex intractable model. With probability $1-\epsilon$, we progress according to A_θ, and with probability ϵ, we draw a fresh sample from u_θ, which corresponds to an informed random restart. Importantly, u_θ does not depend on the previous state y_{t-1}. When $\epsilon = 1$, we are drawing i.i.d. samples from u_θ; we therefore mix in a single step, but our stationary distribution must necessarily be very simple. When $\epsilon = 0$, the stationary distribution can be much richer, but we have no guarantees on the mixing time. For intermediate values of ϵ, we trade off between representational power and mixing time.

A classic result is that a given strong Doeblin chain mixes in time at most $\frac{1}{\epsilon}$ (Doeblin, 1940), and that we can draw an exact sample from the stationary distribution in expected time $O(\frac{1}{\epsilon})$ (Corcoran & Tweedie, 1998). In this work, we prove new results that help us understand the strong Doeblin model families. Let $\mathcal{F}$ and $\tilde{\mathcal{F}}$ be the family of stationary distributions corresponding to A_θ and $\tilde{A}_\theta$ as defined in (1), respectively. Our first result is that as ϵ decreases, the stationary distribution of any $\tilde{A}_\theta$ monotonically approaches the stationary distribution of A_θ (as measured by either direction of the KL divergence). Our second result is that if $\frac{1}{\epsilon}$ is much larger than the mixing time of A_θ, then the stationary distributions of A_θ and $\tilde{A}_\theta$ are close under a certain Mahalanobis distance. This shows that any member of $\mathcal{F}$ that is computationally accessible via the Markov chain is well-approximated by its counterpart in $\tilde{\mathcal{F}}$.



The figure above shows $\mathcal{F}$ and $\tilde{\mathcal{F}}$, together with the subset $\mathcal{F}_0$ of $\mathcal{F}$ whose Markov chains mix quickly. $\tilde{\mathcal{F}}$ (approximately) covers $\mathcal{F}_0$, and also contains some distributions outside of $\mathcal{F}$.

In order to learn over $\tilde{\mathcal{F}}$, we show how to maximize the likelihood of the data under the stationary distribution of $\tilde{A}_\theta$. Specifically, we show that we can compute a stochastic gradient of the log-likelihood in expected time $O(\frac{1}{\epsilon})$. Thus, in a strong sense, our objective function explicitly accounts for computational constraints.

We also generalize strong Doeblin chains, which are a mixture of two base chains, u_θ and A_θ, to *staged* strong Doeblin chains, which allow us to combine more than two base chains. We introduce an auxiliary variable z representing the "stage" that the chain is in. We then transition between stages, using the base chain corresponding to the current stage z to advance the concrete state y. A common application of this generalization is defining a sequence of increasingly more complex chains, similar in spirit to annealing. This allows sampling to become gradually more sensitive to the structure of the problem.

We evaluated our methods on two tasks: (i) inferring words from finger gestures on a touch screen and (ii) inferring DNF formulas for program verification. Unlike many structured prediction problems where local potentials provide a large fraction of the signal, in the two tasks above, local potentials offer a very weak signal; reasoning carefully about the higher-order potentials is necessary to perform well. On word inference, we show that learning strong Doeblin chains obtains a 3.6% absolute improvement in character accuracy over Gibbs sampling while requiring 5x fewer samples. On DNF formula inference, our staged strong Doeblin chain obtains an order of magnitude speed improvement over plain Metropolis-Hastings.

To summarize, the contributions of this paper are: We formally define a family of MCMC algorithms based on strong Doeblin chains with guaranteed fast mixing times (Section 2). We provide an extensive analysis of the theoretical properties of this family (Section 3), together with a generalization to a staged version (Section 3.1). We provide an algorithm for learning the parameters of a strong Doeblin chain (Section 4). We demonstrate superior experimental results relative to baseline MCMC samplers on two tasks, word inference and DNF formula synthesis (Section 5).

2. A Fast-Mixing Family of Markov Chains

Given a Markov chain with transition matrix $A(y_t \mid y_{t-1})$ and a distribution $u(y_t)$, define a new Markov chain with transitions given by $\tilde{A}(y_t \mid y_{t-1}) \stackrel{\text{def}}{=} (1-\epsilon)A(y_t \mid y_{t-1}) + \epsilon u(y_t)$. (We suppress the dependence on θ and x for now.)

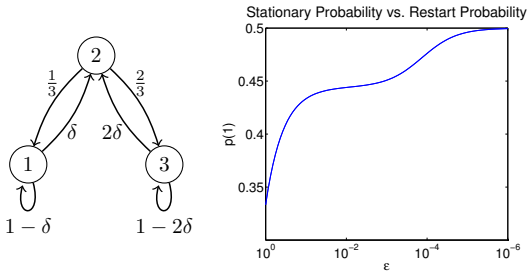


Figure 1. Left: A simple 3-state Markov chain. Arcs denote transition probabilities. Right: plot of the stationary probability of state 1 as a function of restart probability ϵ, for $\delta = 10^{-4}$; closer to 0.499 (the true probability) is better. Note the two regimes for $\epsilon \ll \delta$ and $\epsilon \gg \delta$.

In matrix notation, we can write $\tilde{A}$ as

$$\tilde{A} \stackrel{\text{def}}{=} (1-\epsilon)A + \epsilon u \mathbb{1}^\top. \tag{2}$$

In other words, with probability ϵ we restart from u; otherwise, we transition according to A. Intuitively, $\tilde{A}$ should mix quickly because a restart from u renders the past independent of the future (we formalize this in Section 3). We think of u as a simple tractable model that provides *coverage*, and A as a complex model that provides *precision*.

Simple example. To gain some intuition, we work through a simple example with the Markov chain A depicted in Figure 1. The stationary distribution of this chain is $\left[\begin{array}{ccc} \frac{1}{2+3\delta} & \frac{3\delta}{2+3\delta} & \frac{1}{2+3\delta} \end{array}\right]$, splitting most of the probability mass evenly between states 1 and 3. The mixing time of this chain is approximately $\frac{1}{\delta}$, since once the chain falls into either state 1 or state 3, it will take about $\frac{1}{\delta}$ steps for it to escape back out. If we run this Markov chain for T steps with $T \ll \frac{1}{\delta}$, then our samples will be either almost all in state 1 or almost all in state 3, and thus will provide a poor summary of the distribution. If instead we perform random restarts with probability ϵ from a uniform distribution u over $\{1, 2, 3\}$, then the restarts give us the opportunity to explore both modes of the distribution. After a restart, however, the chain will more likely fall into state 3 than state 1 ($\frac{5}{9}$ probability vs. $\frac{4}{9}$), so for $\epsilon \gg \delta$ the stationary distribution will be noticeably perturbed by the restarts. If $\epsilon \ll \delta$, then there will be enough time for the chain to mix between restarts, so this bias will vanish. See Figure 1 for an illustration of this phenomenon.

3. Theoretical Properties

Markov chains that can be expressed according to (2) are said to have a *strong Doeblin parameter* ϵ (Doeblin, 1940). In this section, we characterize the stationary distribution and mixing time of $\tilde{A}$, and also relate the stationary distribution of $\tilde{A}$ to that of A as a function of ϵ. Often the easiest

way to study the mixing time of $\tilde{A}$ is via its spectral gap, which is defined as $1 - \lambda_2(\tilde{A})$, where $\lambda_2(\tilde{A})$ is the second-largest eigenvalue (in complex norm). A standard result for Markov chains is that, under mild assumptions, the mixing time of $\tilde{A}$ is $O\left(\frac{1}{1-\lambda_2(\tilde{A})}\right)$. We assume throughout this section that A is ergodic but not necessarily reversible. See Section 12.4 of Levin et al. (2008) for more details.

Our first result relates the spectral gap (and hence the mixing time) to ϵ. This result (as well as the next) are well-known but we include them for completeness. For most results in this section, we sketch the proof here and provide the full details in the supplement.

Proposition 3.1. *The spectral gap of $\tilde{A}$ is at least ϵ; that is, $1 - \lambda_2(\tilde{A}) \geq \epsilon$. In particular, $\tilde{A}$ mixes in time $O(\frac{1}{\epsilon})$.*

The key idea is that all eigenvectors of $\tilde{A}$ and A are equal except for the stationary distribution, and that $\lambda_k(\tilde{A}) = (1-\epsilon)\lambda_k(A)$ for $k > 1$.

Having established that $\tilde{A}$ mixes quickly, the next step is to characterize its stationary distribution:

Proposition 3.2. *Let $\tilde{\pi}$ be the stationary distribution of $\tilde{A}$. Then*

$$\tilde{\pi} = \epsilon \sum_{j=0}^{\infty} (1-\epsilon)^j A^j u = \epsilon (I - (1-\epsilon)A)^{-1} u. \qquad (3)$$

This can be directly verified algebraically. The summation over j shows that we can in fact draw an exact sample from $\tilde{\pi}$ by drawing $T \sim \text{Geometric}(\epsilon)$,[1] initializing from u, and transitioning T times according to A. This is intuitive, since at a generic point in time we expect the most recent sample from u to have occurred $\text{Geometric}(\epsilon)$ steps ago. Note that $\mathbb{E}[T+1] = \frac{1}{\epsilon}$, which is consistent with the fact that the mixing time is $O(\frac{1}{\epsilon})$ (Proposition 3.1).

We would like to relate the stationary distributions $\tilde{\pi}$ and π of $\tilde{A}$ and A, respectively. The next two results (which are new) do so.

Let $\tilde{\pi}_\epsilon$ denote the stationary distribution of $\tilde{A}$ at a particular value of ϵ; note that $\tilde{\pi}_1 = u$ and $\tilde{\pi}_0 = \pi$. We will show that $\tilde{\pi}_\epsilon$ approaches π *monotonically*, for both directions of the KL divergence. In particular, for any $\epsilon < 1$, $\tilde{\pi}_\epsilon$ is at least as good as u at approximating π.

To show this, we make use of the following lemma from Murray & Salakhutdinov (2008):

Lemma 3.3. *If B is a transition matrix with stationary distribution π, then $\text{KL}(\pi \parallel B\pi') \leq \text{KL}(\pi \parallel \pi')$ and $\text{KL}(B\pi' \parallel \pi) \leq \text{KL}(\pi' \parallel \pi)$.*

[1] If $T \sim \text{Geometric}(\epsilon)$, we have $\mathbb{P}[T = j] = \epsilon(1-\epsilon)^j$ for $j \geq 0$.

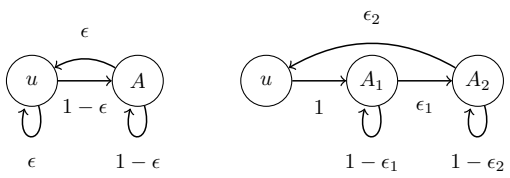


Figure 2. Markov chains over (a) two stages (strong Doeblin chains); and (b) three stages (restart from u followed by transitions from A_1 and then from A_2).

Using this lemma, we can prove the following monotonicity result:

Proposition 3.4. *Both $\text{KL}(\tilde{\pi}_\epsilon \parallel \pi)$ and $\text{KL}(\pi \parallel \tilde{\pi}_\epsilon)$ are monotonic functions of ϵ.*

The idea is to construct a transition matrix B that maps $\tilde{\pi}_{\epsilon_1}$ to $\tilde{\pi}_{\epsilon_2}$ for given $\epsilon_2 < \epsilon_1$, then show that its stationary distribution is π and apply Lemma 3.3.

With Proposition 3.4 in hand, a natural next question is how small ϵ must be before $\tilde{\pi}$ is reasonably close to π. Proposition 3.5 provides one such bound: $\tilde{\pi}$ is close to π if ϵ is small compared to the spectral gap $1 - \lambda_2(A)$.

Proposition 3.5. *Suppose that A satisfies detailed balance with respect to π. Let $\tilde{\pi}$ be the stationary distribution of $\tilde{A}$. Define $d_\pi(\pi') \stackrel{\text{def}}{=} \|\pi - \pi'\|_{\text{diag}(\pi)^{-1}} = \sqrt{-1 + \sum_y \frac{\pi'(y)^2}{\pi(y)}}$, where $\|v\|_M$ is the Mahalonobis distance $\sqrt{v^\top M v}$. Then $d_\pi(\tilde{\pi}) \leq \frac{\epsilon}{1-\lambda_2(A)} \cdot d_\pi(u)$. (In particular, $d_\pi(\tilde{\pi}) \ll 1$ if $\epsilon \ll (1 - \lambda_2(A))/d_\pi(u)$.)*

The proof is somewhat involved though based on classical arguments (for instance, Chapter 12 of Levin et al. (2008)). The key step is to establish that $d_\pi(\pi')$ is convex in π' and contractive with respect to A (more precisely, that $d_\pi(A\pi') \leq \lambda_2(A) d_\pi(\pi')$).

Proposition 3.5 says that if A mixes quickly (in particular, in time much smaller than $\frac{1}{\epsilon}$), then $\tilde{A}$ and A will have similar stationary distributions. This serves as a sanity check: if A already mixes quickly, then $\tilde{\pi}$ is a good approximation to π; otherwise, the Doeblin construction ensures that we are at least converging to *some* distribution, which by Proposition 3.4 approximates π at least as well as u does.

3.1. Staged strong Doeblin chains

Recall that to run a strong Doeblin chain $\tilde{A}$, we first sample from u, and then transition according to A for approximately $\frac{1}{\epsilon}$ steps. The intuition is that sampling from the crude distribution u faciliates global exploration of the state space, while the refined transition A hones in on a mode. However, for complex problems, there might be a considerable gap between what is possible with exact inference (u)

and what is needed for accurate modeling (A). This motivates using multiple stages of MCMC to bridge the gap.

To do this, we introduce an auxiliary variable $z \in \mathcal{Z}$ denoting which stage of MCMC we are currently in. For each stage z, we have a Markov chain $A_z(y_t \mid y_{t-1})$ over the original state space. We also define a Markov chain $C(z_t \mid z_{t-1})$ over the stages. To transition from (y_{t-1}, z_{t-1}) to (y_t, z_t), we first sample z_t from $C(z_t \mid z_{t-1})$ and then y_t from $A_{z_t}(y_t \mid y_{t-1})$. If there is a special state z^* for which $A_{z^*}(y_t \mid y_{t-1}) = u(y_t)$ (i.e., A_{z^*} does not depend on y_{t-1}), then we call the resulting chain a *staged strong Doeblin chain*.

For example, if $z \in \{0, 1\}$ and we transition from 0 to 1 with probability $1 - \epsilon$ and from 1 to 0 with probability ϵ, then we recover strong Doeblin chains assuming $z^* = 0$ (Figure 2(a)). As another example (Figure 2(b)), let $z \in \{0, 1, 2\}$. When $z = z^* = 0$, we transition according to a restart distribution $u\mathbb{1}^\top$; when $z = 1$, we transition according to a simple chain A_1; and when $z = 2$, we transition according to a more complex chain A_2. If we transition from $z = 0$ to $z = 1$ with probability 1, from $z = 1$ to $z = 2$ with probability ϵ_1, and from $z = 2$ to $z = 0$ with probability ϵ_2, then we will on average draw 1 sample from u, $\frac{1}{\epsilon_1}$ samples from A_1, and $\frac{1}{\epsilon_2}$ samples from A_2.

We now show that staged strong Doeblin chains mix quickly as long as we visit z^* reasonably often. In particular, the following theorem provides guarantees on the mixing time that depend only on z^* and on the structure of $C(z_t \mid z_{t-1})$, analogous to the dependence only on ϵ for non-staged chains. The condition of the theorem asks for times a and b such that the first time after a that we hit z^* is almost independent of the starting state z_0, and is less than b with high probability.

Theorem 3.6. *Let M be a staged strong Doeblin chain on $\mathcal{Z} \times \mathcal{Y}$. Let τ_a be the earliest time $s \geq a$ for which $z_s = z^*$. Let $\beta_{a,s} = \min_{z \in \mathcal{Z}} \mathbb{P}[\tau_a = s \mid z_0 = z]$ and $\gamma_{a,b} \stackrel{\text{def}}{=} \sum_{s=a}^{b} \beta_{a,s}$. Then M^b has strong Doeblin parameter $\gamma_{a,b}$, and the spectral gap of M is at least $\frac{\gamma_{a,b}}{b}$. (Setting $a = b = 1$ recovers Proposition 3.1.)*

The key idea is that, conditioned on τ_a, (y_b, z_b) is independent of (y_0, z_0) for all $b \geq \tau_a$. For the special case that the stages form a cycle as in Figure 2, we have the following corollary:

Corollary 3.7. *Let C be a transition matrix on $\{0, \ldots, k-1\}$ such that $C(z_t = i \mid z_{t-1} = i) = 1 - \delta_i$ and $C(z_t = (i+1) \bmod k \mid z_{t-1} = i) = \delta_i$. Suppose that $\delta_{k-1} \leq \frac{1}{\max(2,k-1)} \min\{\delta_0, \ldots, \delta_{k-2}\}$. Then the spectral gap of the joint Markov chain is at least $\frac{\delta_{k-1}}{78}$.*

The key idea is that, when restricting to the time interval $[2/\delta_{k-1}, 3/\delta_{k-1}]$, the time of first transition from $k-1$ to 0 is approximately $\text{Geometric}(\delta_{k-1})$-distributed (independent of the initial state), which allows us to invoke Theorem 3.6. We expect the optimal constant to be much smaller than 78.

4. Learning strong Doeblin chains

Section 3 covered properties of strong Doeblin chains $(1-\epsilon)A_\theta + \epsilon u_\theta \mathbb{1}^\top$ for a fixed parameter vector θ. Now we turn to the problem of learning θ from data. We will focus on the discriminative learning setting where we are given a dataset $\{(x^{(i)}, y^{(i)})\}_{i=1}^n$ and want to maximize the conditional log-likelihood:

$$\mathcal{O}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(y^{(i)} \mid x^{(i)}), \tag{4}$$

where now p_θ is the stationary distribution of $\tilde{A}_\theta = (1-\epsilon)A_\theta + \epsilon u_\theta \mathbb{1}^\top$. We assume for simplicity that the chains A_θ and u_θ are given by conditional exponential families:

$$\begin{aligned} A_\theta(y \mid y', x) &\stackrel{\text{def}}{=} \exp\left(\theta^\top \phi(x, y', y) - \log Z(\theta; x, y)\right), \\ u_\theta(y \mid x) &\stackrel{\text{def}}{=} \exp\left(\theta^\top \phi(x, y) - \log Z(\theta; x)\right), \end{aligned} \tag{5}$$

where each ϕ outputs a feature vector and the Z are partition functions. By Proposition 3.1, $\tilde{A}_\theta$ mixes quickly for all θ. On the other hand, the parameterization of A_θ captures a rich family of transition kernels, including Gibbs sampling.

At a high level, our learning algorithm performs stochastic gradient descent on the negative log-likelihood. However, the negative log-likelihood is only defined implicitly in terms of the stationary distribution of a Markov chain, so the main challenge is to show that it can be computed efficiently. To start, we assume that we can operate on the base chains u_θ and A_θ for one step efficiently:

Assumption 4.1. *We can efficiently sample y from $u_\theta(\cdot \mid x)$ and $A_\theta(\cdot \mid y', x)$, as well as compute $\frac{\partial \log u_\theta(y|x)}{\partial \theta}$ and $\frac{\partial \log A_\theta(y|y',x)}{\partial \theta}$.*

Under Asssumption 4.1, we will show how to efficiently compute the gradient of $\log p_\theta(y^{(i)} \mid x^{(i)})$ with respect to θ. The impatient reader may skip ahead to the final pseudocode, which is given in Algorithm 1.

For convenience, we will suppress the dependence on x and i and just refer to $p_\theta(y)$ instead of $p_\theta(y^{(i)} \mid x^{(i)})$. Computing the gradient of $\log p_\theta(y)$ is non-trivial, since the formula for p_θ is somewhat involved (see Proposition 3.2):

$$p_\theta(y) = \epsilon \sum_{j=0}^{\infty} (1-\epsilon)^j [A_\theta^j u_\theta](y). \tag{6}$$

It is useful to invoke the following generic identity on gradients of conditional log-probabilities, proved in the supplement.

Lemma 4.2. *Let z have distribution $p_\theta(z)$ parameterized by a vector θ. Let S be any measurable set. Then*

$$\frac{\partial \log p_\theta(z \in S)}{\partial \theta} = \mathbb{E}_{z \sim p_\theta}\left[\frac{\partial \log p_\theta(z)}{\partial \theta}\middle| z \in S\right]. \quad (7)$$

We can utilize Lemma 4.2 by interpreting $y \mid \theta$ as the output of the following generative process, which by Proposition 3.2 yields the stationary distribution of $\tilde{A}_\theta$:

- Sample y_0 from u_θ and $y_{t+1} \mid y_t$ from A_θ for $t = 0, 1, \ldots$.
- Sample $T \sim \text{Geometric}(\epsilon)$ and let $y = y_T$.

We then invoke Lemma 4.2 with $z = (T, y_{0:T})$ and S encoding the event that $y_T = y$. As long as we can sample from the posterior distribution of $(T, y_{0:T})$ conditioned on $y_T = y$, we can compute an estimate of $\frac{\partial}{\partial \theta} \log p_\theta(y)$ as follows:

- Sample $(T, y_{0:T}) \mid y_T = y$.
- Return $\frac{\partial \log p_\theta(T, y_{0:T})}{\partial \theta} = \frac{\partial \log u_\theta(y_0)}{\partial \theta} + \sum_{t=1}^{T} \frac{\partial \log A_\theta(y_t \mid y_{t-1})}{\partial \theta}$.

4.1. Sampling schemes for $(T, y_{0:T})$

By the preceding discussion, it suffices to construct a sampler for $(T, y_{0:T}) \mid y_T = y$. A natural approach is to use importance sampling: sample $(T, y_{0:T-1})$, then weight by $p(y_T = y \mid y_{T-1})$. However, this throws away a lot of work — we make T MCMC transitions but obtain only one sample $(T, y_{0:T})$ with which to estimate the gradient.

We would like to ideally make use of all the MCMC transitions when constructing our estimate of $(T, y_{0:T}) \mid y_T = y$. For any $t \leq T$, the pair $(t, y_{0:t})$ would itself have been a valid sample under different randomness, and we would like to exploit this. Suppose that we sample T from some distribution F and let $q(t)$ be the probability that $T \geq t$ under F. Then we can use the following scheme:

- Sample T from F, then sample $y_{0:T-1}$.
- For $t = 0, \ldots, T$, weight $(t, y_{0:t-1})$ by $\frac{\epsilon(1-\epsilon)^t}{q(t)} \times p(y_t = y \mid y_{t-1})$.

For any q, this yields unbiased (although unnormalized) weights (see Section B in the supplement). Typically we will choose $q(t) = (1-\epsilon)^t$, e.g. F is a geometric distribution. If the y_t are perfectly correlated, this will not be any more effective than vanilla importance sampling, but in practice this method should perform substantially better. Even though we obtain weights on all of $y_{0:T}$, these weights will typically be highly correlated, so we should still repeat the sampling procedure multiple times to minimize the bias from estimating the normalization constant. The full procedure is given as pseudocode in Algorithm 1.

Algorithm 1 Algorithm for computing an estimate of $\frac{\partial}{\partial \theta} \log p_\theta(y \mid x)$. This estimate is asymptotically unbiased as the number of samples $k \to \infty$, but will be biased for a finite number of samples due to variance in the estimate of the normalization constant.

SampleGradient$(x, y, \theta, \epsilon, k)$
$\triangleright$ k is the number of samples to take
$Z \leftarrow 0; g \leftarrow 0$ $\triangleright$ Z is the total importance mass of all samples, $\frac{g}{Z}$ is the gradient
for $i = 1$ **to** k **do**
  Sample $T \sim \text{Geometric}(\epsilon)$
  Sample y_0 from $u_\theta(\cdot \mid x)$
  For $1 \leq t \leq T-1$: sample y_t from $A_\theta(\cdot \mid y_{t-1}, x)$
  $w_0 \leftarrow \epsilon \cdot u_\theta(y)$
  For $1 \leq t \leq T$: $w_t \leftarrow \epsilon \cdot A_\theta(y \mid y_{t-1}, x)$
  $Z \leftarrow Z + \sum_{t=0}^{T} w_t$
  $g \leftarrow g + w_0 \frac{\partial \log u_\theta(y|x)}{\partial \theta}$
    $+ \sum_{t=1}^{T} w_t \left(\frac{\partial \log u_\theta(y_0|x)}{\partial \theta} \right.$
      $\left. + \sum_{s=1}^{t-1} \frac{\partial \log A_\theta(y_s|y_{s-1},x)}{\partial \theta} + \frac{\partial \log A_\theta(y|y_{t-1},x)}{\partial \theta} \right)$.
end for
Output $\frac{g}{Z}$

4.2. Implementation

With the theory above in place, we now describe some important implementation details of our learning algorithm. At a high level, we can just use Algorithm 1 to compute estimates of the gradient and then apply an online learning algorithm such as AdaGrad (Duchi et al., 2010) to identify a good choice of θ. Since the log-likelihood is a non-convex function of θ, the initialization is important. We make the following (weak) assumption:

Assumption 4.3. *The chains u_θ and A_θ are controlled by disjoint coordinates of θ, and for any setting of u_θ there is a corresponding choice of A_θ that leaves u_θ invariant (i.e., $A_\theta u_\theta = u_\theta$).*

In practice, Assumption 4.3 is easy to satisfy. For instance, suppose that $\phi : \mathcal{Y} \to \mathbb{R}^d$ is a feature function, $\theta = [\theta_0 \ \theta_1] \in \mathbb{R}^{d_0+d}$ are the features controlling u and A, and u_{θ_0} is made tractable by zeroing some features out: $u_{\theta_0}(y) \propto \exp([\theta_0 \ \vec{0}_{d-d_0}]^\top \phi(y))$. Also suppose that A_{θ_1} is a Gibbs sampler that uses all the features: $A_{\theta_1}(y \mid y') \propto \exp(\theta_1^\top \phi(y_i, y'_{\neg i}))$, where i is a randomly chosen coordinate of y. Then, we can satisfy Assumption 4.3 by setting $\theta_1 = [\theta_0 \ \vec{0}_{d-d_0}]$.

Under Assumption 4.3, we can initialize θ by first training u in isolation (which is a convex problem since u_θ parameterizes an exponential family), then initializing A to leave u invariant; this guarantees that the initial log-likelihood is what we would have obtained by just using u by itself. We

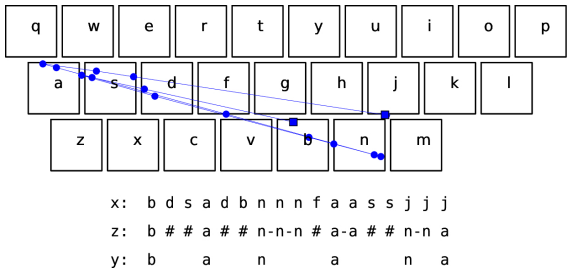


Figure 3. Generated sequence of keyboard gestures for the word *banana*. The input x is a sequence of characters (the recorded key presses), and the output y is the intended word. Most characters in x are incidental and do not correspond to any character in y; this is reflected by the (unobserved) alignment z.

found this to work well empirically.

As another note, Algorithm 1 naïvely looks like it takes $O(T^2)$ time to compute the gradient for each sample, due to the nested sum. However, most terms are of the form $w_t \frac{\partial \log A_\theta(y_s|y_{s-1},x)}{\partial \theta}$; by grouping them for a fixed s we can compute the sum in $O(T)$ time, leading to expected runtime $O\left(\frac{k}{\epsilon}\right)$ for Algorithm 1 (since $\mathbb{E}[T+1] = \frac{1}{\epsilon}$).

5. Experiments

We validated our method on two non-trivial inference tasks. These tasks are difficult due to the importance of high-arity factors; local information is insufficient to even identify high-probability regions of the space.

Inferring Words from Keyboard Gestures. We first considered the task of inferring words from keyboard gestures. We generated the data by sampling words from the New York Times corpus (Sandhaus, 2008). For each word, we used a time series model to synthetically generate finger gestures for the word. A typical instantiation of this process is given in Figure 3. The learning task is to discriminatively infer the intended word y given the sequence of keys x that the finger was over (for instance, predicting `banana` from `bdsadbnnnfaassjjj`). In our model, we posit a latent alignment z between key presses and intended letter. Given an input x of length l, the alignment z also has length l; each z_i is either 'c' (x_i starts an output letter c), '$-c$' (x_i continues an output letter c), or '#' (x_i is unaligned); see Figure 3 for an example. Note that y is a deterministic function of z.

The base model u_θ consists of indicator features on (x_i, z_i), (x_i, z_{i-1}, z_i), and (x_i, x_{i-1}, z_i). The full A_θ is a Gibbs sampler in a model where we include the following features in addition to those above:

- Indicator features on (x_i, y_i, y_{i-1})
- Indicator of y being in the dictionary, as well as log of word frequency (conditioned on being in the dictionary)
- For each i, indicator of $y_{1:i}$ matching a prefix of a word in the dictionary

We compared three approaches:

- Our approach (*Doeblin sampling*)
- Regular Gibbs sampling, initialized by setting $z_i = x_i$ for all i (*basic-Gibbs*)
- Gibbs sampling initialized from u_θ (u_θ-*Gibbs*)

At test time, all three of these methods are computationally almost identical: they all initialize from some distribution, then make a certain number of Gibbs samples. For basic-Gibbs and u_θ-Gibbs, this is always a fixed number of steps T, while for Doeblin sampling, the number of steps is a geometric distribution with mean T.

The main difference is in how the methods are trained. Our method is trained using the ideas in Section 4; for the other two methods, we train by approximating the gradient:

$$\nabla \log p_\theta(y \mid x) = \mathbb{E}_{\hat{z} \sim p_\theta(z|x,y)}[\phi(y, \hat{z}, x)] - \mathbb{E}_{\hat{y},\hat{z} \sim p_\theta(y,z|x)}[\phi(\hat{y}, \hat{z}, x)],$$

where $\phi(y, z, x)$ is the feature function and p_θ is the stationary distribution of A_θ. For the second term, we use MCMC samples from A_θ to approximate $p_\theta(y, z \mid x)$. For the first term, we could take the subset of samples where $\hat{y} = y$, but this is problematic if no such samples exist. Instead, we reweight all samples with $\hat{y} \neq y$ by $\exp(-(D+1))$, where D is the edit distance between y and $\hat{y}$. We use the same reweighting approach for the Doeblin sampler, using this as the importance weight rather than using $A_\theta(y \mid y_{t-1})$ as in Algorithm 1.

To provide a fair comparison of the methods, we set ϵ in the Doeblin sampler to the inverse of the number of transitions T, so that the expected number of transitions of all algorithms is the same. We also devoted the first half of each chain to burn-in.

All algorithms are trained with AdaGrad (Duchi et al., 2010) with 16 independent chains for each example. We measure word-level accuracy by computing the fraction of (non-burn-in) samples whose output y is correct.

The results are reported in Figure 4. Overall, our Doeblin sampler outperforms u_θ-Gibbs by a significant margin, which in turn outperforms basic-Gibbs. Interestingly, while the accuracy of our method continues to improve with more training time, u_θ-Gibbs quickly asymptotes and then slightly decreases, even for training accuracy.

What is happening to u_θ-Gibbs? Since the inference problem in this task is hard, the samples provide a poor gradient approximation. As a result, optimization methods that take the approximation at face value may not converge to even a

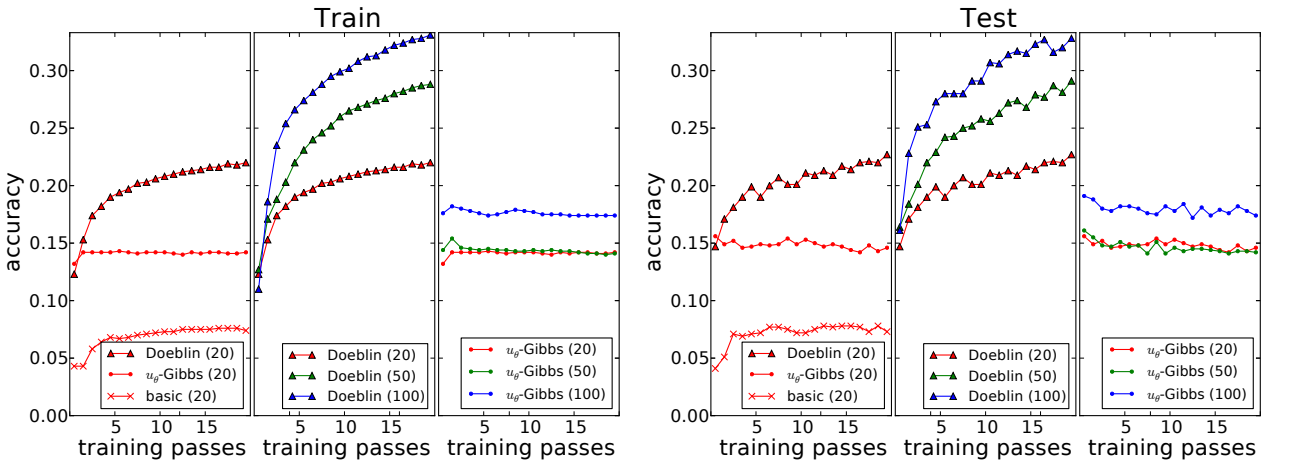


Figure 4. Plots of word-level (left) and character-level (right) accuracy. The first panel gives the performance of all 3 methods (Doeblin sampling, u_θ-Gibbs, and basic-Gibbs) for a computational budget of 20 transitions per example. The second and third panels show the accuracy of Doeblin sampling and u_θ-Gibbs, respectively, for increasing computational budgets (20, 50, and 100 transitions).

local optimum. This phenomenon has already been studied in other contexts, for instance by Kulesza & Pereira (2007) and Huang et al. (2012).

In contrast, our method directly optimizes the log-likelihood of the data under the distribution $\tilde{\pi}_\theta$, so that accuracy continues to increase with more passes through the training data. This demonstrates that the MCMC samples do provide enough signal to train from, but that naïvely plugging them into a method designed for exact inference will fail to exploit that signal.

Inferring DNF Formulas. Next, we study the use of our staged Doeblin chain construction as a tool for hierarchical initialization. We ignore learning for now, instead treating MCMC as a stochastic search algorithm. Our task of interest is to infer a DNF formula f from its input-output behavior. This is an important subroutine in loop invariant synthesis, where MCMC methods have recently shown great promise (Gulwani & Jojic, 2007; Sharma & Aiken, 2014).

Concretely, we are given the output of an unknown DNF formula f for various inputs $x = (x_1, x_2, x_3)$; for instance:

$$f(1,2,3) = \texttt{True} \qquad f(1,4,4) = \texttt{True}$$
$$f(0,1,0) = \texttt{False} \qquad f(0,2,2) = \texttt{True}$$
$$\text{et cetera.}$$

Our task is to induce f; in this case, $f(x_1, x_2, x_3) = [x_1 \neq 0] \vee [x_2 = x_3]$. In general, we avoid trivial solutions that overfit the data by imposing limits on the size of f.

More formally, we consider DNF formulae with linear inequalities: $f(x) = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} [a_{ij}^\top x \leq b_{ij}]$, where $a_{ij}, x \in \mathbb{Z}^d$ and $b_{ij} \in \mathbb{Z}$. The formula f maps input vectors x to $\{\texttt{True}, \texttt{False}\}$. Given a collection of example inputs and outputs, our goal is to find an f consistent with all examples. Our evaluation metric is the time to find such a formula.

The search space for this problem is extremely large. Even if we set $n = m = 3$ and restrict our search to $a_{ij} \in \{-1, 0, 1\}^5$, $b \in \{-1, 0, 1\}$, the total number of candidate formulae is still $\left(3^6\right)^{3\times 3} \approx 5.8 \times 10^{25}$.

We consider three MCMC methods: no restarts (0-stage), uniformly random restarts (1-stage), and a staged method (2-stage) as in Section 3.1. All base chains perform Metropolis-Hastings using proposals that edit individual atoms (e.g., $[a_{ij}^\top x \leq b_{ij}]$), either by changing a single entry of $[a_{ij}\ b_{ij}]$ or by changing all entries of $[a_{ij}\ b_{ij}]$ at once. For the staged method, we initialize f uniformly at random, take $\text{Geometric}(0.04)$ transitions based on a simplified cost function, then take $\text{Geometric}(0.0002)$ steps with the full cost (this is the staged Doeblin chain in Figure 2).

The full cost function is $I(f)$, the number of examples f errs on. We stop the Markov chain when it finds a formula with $I(f) = 0$. The simplified cost function decomposes over the disjuncts: for each disjunct $d(x)$, if $f(x) = \texttt{False}$ while $d(x) = \texttt{True}$, we incur a large cost (since in order for $f(x)$ to be false, all disjuncts comprising $f(x)$ must also be false). If $f(x) = \texttt{True}$ while $d(x) = \texttt{False}$, then we incur a smaller cost. If $f(x) = d(x)$ then we incur no cost.

We used all three methods as a subroutine in verifying properties of C programs; each such verification requires solving many instances of DNF formula inference. Using the staged method we are able to obtain a 30% speedup over uniformly random restarts and a 50x improvement over no restarts, as shown in Table 1.

Table 1. Comparison of 3 different MCMC algorithms. 0-stage uses no restarts, 1-stage uses random restarts, and 2-stage uses random restarts followed by a short period of MH with a simplified cost function. The table gives mean time and standard error (in seconds) taken to verify 5 different C programs, averaged over 1000 trials. Each verification requires inferring many DNF formulae as a sub-routine.

Task	fig1	cegar2	nested	tacas06	hard
0-stage	2.6 ± 1.0	320 ± 9.3	120 ± 7.0	≥ 600	≥ 600
1-stage	0.074 ± 0.001	0.41 ± 0.01	2.4 ± 0.10	6.8 ± 0.15	52 ± 1.5
2-stage	0.055 ± 0.005	0.33 ± 0.007	2.3 ± 0.12	4.6 ± 0.12	31 ± 0.90

6. Discussion

We have proposed a model family based on strong Doeblin Markov chains, which guarantee fast mixing. Our construction allows us to simultaneously leverage a simple, tractable model (u_θ) that provides coverage together with a complex, accurate model (A_θ) that provides precision. As such, we sidestep a typical dilemma—whether to use a simple model with exact inference, or to deal with the consequences of approximate inference in a more complex model.

While our approach works well in practice, there are still some outstanding issues. One is the non-convexity of the learning objective, which makes the procedure dependent on initialization. Another issue is that the gradients returned by Algorithm 1 can be large, heterogeneous, and high-variance. The adaptive nature of AdaGrad alleviates this somewhat, but ideally we would like a sampling procedure that has lower variance than Algorithm 1.

Though Gibbs sampling is the de facto method for many practitioners, there are also many more sophisticated approaches to MCMC (Green, 1995; Earl & Deem, 2005). Since our framework is orthogonal to the particular choice of transition kernel, it would be interesting to apply our method in these contexts.

Finally, we would like to further explore the staged construction from Section 3.1. As the initial results on DNF formula synthesis are encouraging, it would be interesting to apply the construction to high-dimensional feature spaces as well as rich, multi-level hierarchies. We believe this might be a promising approach for extremely rich models in which a single level of re-initialization is insufficient to capture the complexity of the cost landscape.

Related Work. Our learning algorithm is reminiscent of policy gradient algorithms in reinforcement learning (Sutton et al., 1999), as well as Searn, which tries to learn an optimal search policy for structured prediction (Daume et al., 2009). Shi et al. (2015) applied reinforcement learning to choose which variable to update next in a Gibbs sampler. Our staged construction is also similar in spirit to path sampling (Gelman & Meng, 1998), which uses a multi-stage approach to smoothly transition from a very simple to a very complex distribution.

Our staged Doeblin construction belongs to the family of coarse-to-fine inference methods, which operate on progressively more complex models (Viola & Jones, 2004; Shen et al., 2004; Collins & Koo, 2005; Gu et al., 2009; Weiss et al., 2010; Sapp et al., 2010; Petrov & Klein, 2007; Yadollahpour et al., 2013).

On the theoretical front, we make use of the well-developed theory of *strong Doeblin chains*, often also referred to with the terms *minorization* or *regeneration time* (Doeblin, 1940; Roberts & Tweedie, 1999; Meyn & Tweedie, 1994; Athreya & Ney, 1978). The strong Doeblin property is typically used to study convergence of continuous-space Markov chains, but Rosenthal (1995) has used it to analyze Gibbs sampling, and several authors have provided algorithms for sampling exactly from arbitrary strong Doeblin chains (Propp & Wilson, 1996; Corcoran & Tweedie, 1998; Murdoch & Green, 1998). We are the first to use strong Doeblin properties to construct model families and learn them from data.

At a high level, our idea is to identify a family of models for which an approximate inference algorithm is known to work well, thereby constructing a computationally tractable model family that is nevertheless more expressive than typical tractable families such as low-tree-width graphical models. We believe this general research program is very interesting, and could be applied to other inference algorithms as well, thus solidfying the link between statistical theory and practical reality.

Acknowledgments. The first author was supported by a Fannie & John Hertz Fellowship as well as an NSF Graduate Fellowship. The second author was supported by a Microsoft Research Faculty Fellowship. We also thank the anonymous referees for their helpful comments.

Reproducibility. Code, data, and experiments for this paper are available on the CodaLab platform at https://www.codalab.org/worksheets/0xc6edf0c9bec643ac9e74418bd6ad4136/.

References

Athreya, K. B. and Ney, P. A new approach to the limit theory of recurrent Markov chains. *Transactions of the American Mathematical Society*, 245:493–501, 1978.

Collins, M. and Koo, T. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1): 25–70, 2005.

Corcoran, J. and Tweedie, R. Perfect sampling of Harris recurrent Markov chains. *preprint*, 1998.

Cowles, M. K. and Carlin, B. P. Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association (JASA)*, 91(434): 883–904, 1996.

Daume, H., Langford, J., and Marcu, D. Search-based structured prediction. *Machine Learning*, 75:297–325, 2009.

Doeblin, W. Elements d'une theorie generale des chaines simples constantes de markoff. In *Annales scientifiques de l'École Normale Supérieure*, volume 57, pp. 61–111, 1940.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*, 2010.

Earl, D. J. and Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

Gelman, A. and Meng, X. Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical science*, 13(2):163–185, 1998.

Gelman, A. and Rubin, D. B. A single series from the Gibbs sampler provides a false sense of security. *Bayesian statistics*, 4:625–631, 1992.

Green, P. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

Gu, C., Lim, J. J., Arbeláez, P., and Malik, J. Recognition using regions. In *Computer Vision and Pattern Recognition (CVPR)*, pp. 1030–1037, 2009.

Gulwani, S. and Jojic, N. Program verification as probabilistic inference. *ACM SIGPLAN Notices*, 42(1):277–289, 2007.

Huang, L., Fayong, S., and Guo, Y. Structured Perceptron with inexact search. In *Association for Computational Linguistics (ACL)*, pp. 142–151, 2012.

Kulesza, A. and Pereira, F. Structured learning with approximate inference. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 785–792, 2007.

Levin, D., Peres, Y., and Wilmer, E. *Markov Chains and Mixing Times*. American Mathematical Society, 2008.

Meyn, S. and Tweedie, R. Computable bounds for geometric convergence rates of Markov chains. *The Annals of Applied Probability*, 4(4):981–1011, 1994.

Murdoch, D. J. and Green, P. J. Exact sampling from a continuous state space. *Scandinavian Journal of Statistics*, 25(3):483–502, 1998.

Murray, I. and Salakhutdinov, R. Notes on the KL-divergence between a Markov chain and its equilibrium distribution. *preprint*, 2008.

Petrov, S. and Klein, D. Learning and inference for hierarchically split PCFGs. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pp. 404–411, 2007.

Propp, J. and Wilson, D. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random structures and Algorithms*, 9:223–252, 1996.

Roberts, G. and Tweedie, R. Bounds on regeneration times and convergence rates for Markov chains. *Stochastic Processes and their applications*, 80(2):211–229, 1999.

Rosenthal, J. S. Minorization conditions and convergence rates for Markov chain Monte Carlo. *Journal of the American Statistical Association (JASA)*, 90(430):558–566, 1995.

Sandhaus, E. *The New York Times annotated corpus*, 2008.

Sapp, B., Toshev, A., and Taskar, B. Cascaded models for articulated pose estimation. In *European Conference on Computer Vision (ECCV)*, pp. 406–420, 2010.

Sharma, R. and Aiken, A. From invariant checking to invariant inference using randomized search. In *Computer Aided Verification (CAV)*, pp. 88–105, 2014.

Shen, L., Sarkar, A., and Och, F. J. Discriminative reranking for machine translation. In *North American Association for Computational Linguistics (NAACL)*, pp. 177–184, 2004.

Shi, T., Steinhardt, J., and Liang, P. Learning where to sample in structured prediction. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 875–884, 2015.

Sutton, R., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

Viola, P. and Jones, M. J. Robust real-time face detection. *International Journal of Computer Vision*, 57(2): 137–154, 2004.

Wainwright, M. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research (JMLR)*, 7:1829–1859, 2006.

Weiss, D., Sapp, B., and Taskar, B. Sidestepping intractable inference with structured ensemble cascades. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2415–2423, 2010.

Yadollahpour, P., Batra, D., and Shakhnarovich, G. Discriminative re-ranking of diverse segmentations. In *Computer Vision and Pattern Recognition (CVPR)*, pp. 1923–1930, 2013.