Learning Deep Latent Gaussian Models with Markov Chain Monte Carlo

Matthew D. Hoffman [1]

Abstract

Deep latent Gaussian models are powerful and popular probabilistic models of high-dimensional data. These models are almost always fit using variational expectation-maximization, an approximation to true maximum-marginal-likelihood estimation. In this paper, we propose a different approach: rather than use a variational approximation (which produces biased gradient signals), we use Markov chain Monte Carlo (MCMC, which allows us to trade bias for computation). We find that our MCMC-based approach has several advantages: it yields higher held-out likelihoods, produces sharper images, and does not suffer from the variational overpruning effect. MCMC's additional computational overhead proves to be significant, but not prohibitive.

1. Introduction

Deep latent Gaussian models (DLGMs; Kingma & Welling, 2014; Rezende et al., 2014) are powerful generative models that can be efficiently fit to very large datasets of complicated, high-dimensional data. These models assume that the observed data were generated by sampling some latent variables, feeding them into a deep neural network, and adding some noise to that network's output.

The central computational challenge in fitting these models is approximate posterior inference; to make the model better fit an observation x, we need to estimate the posterior distribution $p(z \mid x)$ over the latent variables z that generated x. Exact posterior inference is intractable, so these models are almost always fit with variational inference (Jordan et al., 1999), which approximates the intractable posterior with a distribution q from some tractable family.

The choice of family for q matters—more expressive families will better approximate the posterior, leading to better learning signals. There has been a flurry of papers recently proposing more flexible variational families, all of which show that the quality of the variational approximation matters (Salimans et al., 2015; Rezende & Mohamed, 2015; Burda et al., 2016; Tran et al., 2016; Sønderby et al., 2016; Kingma & Salimans, 2016).

The other workhorse of approximate posterior inference is Markov chain Monte Carlo (MCMC; Neal, 1993), which sets up and simulates a random process whose stationary distribution is the posterior of interest. Compared to variational methods, MCMC methods have a reputation for being slow, but accurate. Perhaps surprisingly, there is little work exploring the use of MCMC to fit DLGMs (Salimans et al., 2015; Wolf et al., 2016; Han et al., 2017).

In this paper, we explore the advantages of the MCMC approach, and find that, as promised, it allows us to meaningfully trade off bias and computation. By initializing our chains with a sample from a variational approximation, we are able to both speed up convergence and guarantee that our posterior approximation is strictly better than that of the underlying variational approximation. Some minor additional refinements dramatically speed up the algorithm.

In section 5, we show that the MCMC approach eliminates overpruning and blurriness (two known issues with variational autoencoders), and achieves good held-out log-likelihood on the dynamically binarized permutation-invariant MNIST dataset.

2. Background

2.1. Variational Autoencoders and DLGMs

A deep latent Gaussian model (Kingma & Welling, 2014; Rezende et al., 2014, DLGM;) assumes the following generative process for each observation x:

1. Sample a vector $z \in \mathbb{R}^K$ variables independently from the standard normal distribution.
2. Compute a vector-valued nonlinear function $g_\theta(z)$. g is typically a deep neural network with some parameters θ.
3. Sample x from some distribution $f(g(z))$ that takes the output of g as a parameter.

[1]Google, San Francisco, California, USA. Correspondence to: Matthew D. Hoffman <mhoffman@google.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

For example, if x is a vector of real-valued numbers, a relatively simple DLGM would assume $g(z) = W \max\{0, Vz + c\} + b$ and $x_d \sim \mathcal{N}(g_d(z), \sigma)$; $g(z)$ is the output of a one-hidden-layer ReLU network, and the model assumes that g is further corrupted by Gaussian noise with known variance σ^2. In general, g could be more complex (e.g., it might be a deep convolutional network), and different elements of z could be injected at different layers of computation.

This procedure defines a joint probability distribution

$$p(z, x) = \mathcal{N}(z; 0, I) f(x; g_\theta(z)). \tag{1}$$

We can easily sample from and compute this joint distribution for any given parameters θ, latent vector z, and observation x.

We want to fit the model to data by maximizing the *marginal* likelihood of a dataset $x_{1,\ldots,N}$ under the model. This is an optimization problem; the goal is to choose

$$\begin{aligned} \theta^\star &= \arg\max_\theta \frac{1}{N} \sum_n \log p_\theta(x_n) \\ &= \arg\max_\theta \frac{1}{N} \sum_n \log \int_z p_\theta(z, x_n) dz. \end{aligned} \tag{2}$$

Unfortunately, the integral over z is usually intractable.

The typical solution is to apply variational expectation-maximization (VEM; Bishop, 2006), which approximates the intractable marginal $\log p(x)$ with the evidence lower bound (ELBO) $\mathcal{L}(\phi, \theta)$:

$$\begin{aligned} \mathcal{L}(\phi, \theta, x) &\triangleq \mathbb{E}_q[\log p_\theta(x \mid z)] - \mathrm{KL}(q_{z|x} \mid\mid p_z) \\ &= \log p_\theta(x) - \mathrm{KL}(q_{z|x} \mid\mid p_{z|x}) \leq \log p_\theta(x). \end{aligned} \tag{3}$$

$q_\phi(z \mid x)$ is any family of distributions that we choose; it is indexed by some parameters ϕ and an observation x. $\mathbb{E}_q$ denotes expectations with respect to q. $\mathrm{KL}(q \mid\mid p)$ denotes the Kullback-Leibler divergence (KLD) between two distributions q and p. The inequality follows from the non-negativity of the KLD, and since $\mathrm{KL}(p_{z|x} \mid\mid p_{z|x}) = 0$, the bound is tight when $q_\phi(z \mid x) = p_\theta(z \mid x)$.

The standard choice in DLGMs is to choose $q_\phi(z \mid x) \triangleq h(z; r_\phi(x))$, where $r_\phi(x)$ is the vector-valued output of an additional neural network and $h(z; r)$ is a tractable distribution with parameters r. A common form for q is a multivariate Gaussian; when $q(z \mid x) = \prod_k q(z_k \mid x)$, q is called a "mean-field" approximation (Jordan et al., 1999). In this framework, sometimes r is called an "encoder" network and g is called a "decoder" network by analogy to classical neural autoencoders. This analogy suggested the name "variational autoencoder" (VAE; Kingma & Welling, 2014); in this paper we will use "DLGM" to refer to the generative model and "VAE" to refer to the paired encoder/decoder architecture for inference/generation.

In VAEs the first expectation $\mathbb{E}_q[\log p_\theta(x \mid z)]$ is usually intractable, since it is the integral of a complicated nonlinear function. But we can estimate it by Monte Carlo sampling from q. Usually a single sample suffices. Depending on h, the KLD term may or may not be tractable to compute exactly, but again it can be approximated by Monte Carlo as long as h can be computed.

This architecture has a lot to offer: if the variational distribution q is inexpensive to compute, then we can quickly and easily estimate and optimize the ELBO using stochastic optimization.

But this speed and ease comes at the price of accuracy. To the extent that $q(z \mid x)$ diverges from $p(z \mid x)$, the ELBO may be a poor approximation of the true marginal likelihood, and the VEM estimate of θ may be systematically different than the true maximum-likelihood estimate (MLE). In particular, there are two ways to adjust p_θ to increase the ELBO: increase the true marginal likelihood $\log p_\theta(x)$, or make the typical posterior $p_\theta(z \mid x)$ close in KLD to $q_\phi(z \mid x)$.

2.2. Variational pruning

In VAEs this issue tends to manifest itself most dramatically as "pruning"—a phenomenon where the optimizer severs the connections between most of the latent variables z and the data (MacKay, 2001). Pruning occurs because of the $\mathrm{KL}(q_{z|x} \mid\mid p_{z|x})$ term in the ELBO. Imagine that we sever the connections between z_1 and x, so that z_1 and x are independent. (This can be easily done by zeroing out some weights in the neural network $g_\theta(z)$.) Then

$$\mathrm{KL}(q_{z|x} \mid\mid p_{z|x}) = \mathrm{KL}(q_{z_1|x} \mid\mid p_{z_1}) + \mathrm{KL}(q_{z_{2:K}|x} \mid\mid p_{z_{2:K}|x}). \tag{4}$$

The first KLD term can be set to zero by setting $q(z_1 \mid x) = \mathcal{N}(z_1; 0, 1)$. To the extent that it is easier for the optimizer to improve the ELBO by such a strategy, it will find a (possibly bad) local optimum that uses only a few latent dimensions. Hoffman & Blei (2015); Theis & Hoffman (2015) found that for classic mixture models and latent factor models such as latent Dirichlet allocation (Blei et al., 2003) this sort of pruning was mostly a local optimum problem that could be remedied by improved variational approximations or optimization schemes. Bowman et al. (2016) and Sønderby et al. (2016) found improved local optima in VAEs by initially ignoring and then gradually introducing the $\mathrm{KL}(q_{z|x} \mid\mid p_z)$ term, and Burda et al. (2016) found that initializing a mean-field VAE with a model that had been trained with a more flexible q distribution yielded less pruning and better local optima. But unlike the results of Hoffman & Blei (2015) for LDA, Burda et al. (2016)

still found significant pruning even with the more flexible posterior approximation, and that pruning increased when moving from the flexible approximation to the mean-field approximation.

The results of Burda et al. (2016) suggest that this pruning phenomenon is not solely due to local optima. Adding another latent variable to the model should make it possible to improve the average true marginal log-likelihood $\log p_\theta(x)$, but it will also make the typical posterior $p(z \mid x)$ more difficult to approximate, increasing the average $\mathrm{KL}(q_{z|x} \mid\mid p_{z|x})$ term. At some point the marginal cost in KLD will be greater than the marginal benefit in $\log p_\theta(x)$, even at the global optimum.

What makes the VAE ELBO so susceptible to pruning compared to shallow mixture and factor models? One explanation is that the expressive power of shallow models is tightly bound to latent dimensionality—e.g., fewer mixture components makes for a less-powerful mixture model. So if a shallow model is fit to a large, high-dimensional dataset drawn from a difficult-to-model population, it may be possible to justify using many latent dimensions, since there is no other way to explain the observed variation in the data.

By contrast, the expressive power of DLGMs is a function of both the latent dimensionality *and* the complexity of the nonlinear function $g_\theta(z)$. Even if z is one-dimensional, a DLGM can in principle approximate arbitrary smooth densities in $\mathbb{R}^D$ if g approximates a space-filling curve in $\mathbb{R}^D$ and f adds some low-variance noise to g. An example of a one-latent-dimension DLGM approximating a two-dimensional uniform distribution is given in figure 1. This example is contrived, but it could be easily implemented by a neural network (though it might be hard to fit), and it demonstrates the decoupling of latent dimensionality from expressive power.

2.3. Markov chain Monte Carlo and Hamiltonian Monte Carlo

Markov chain Monte Carlo (MCMC; Neal, 1993) is the classic alternative to variational methods for approximate posterior inference. MCMC methods proceed by designing a Markov chain whose stationary distribution is the distribution of interest, then simulating that chain to draw samples from that distribution. For our purposes here, MCMC methods have many weaknesses compared to variational methods: they may require many "burn in" iterations to forget their initial state, successive samples from the chain may be highly correlated, it is sometimes challenging to diagnose when a chain has burned in (Gelman & Shirley, 2011), and getting an estimate of marginal probabilities using MCMC is relatively expensive and complicated (Neal, 2001).

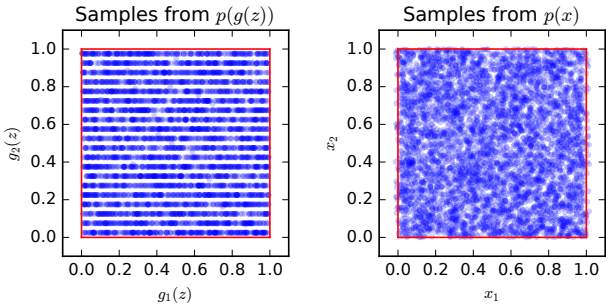

Figure 1. Approximating a 2-d distribution with a 1-d DLGM. Here $z \sim \mathcal{N}(0,1)$, $g_1(z) = (\Phi(z)-1)n + \sum_{i=0}^{n-1} \sigma(\tau(\Phi(z)n - i - 1))$, $g_2(z) = \frac{1}{n}(\frac{1}{2} + \sum_{i=0}^{n-1} \sigma(\tau(\Phi(z)n - i))$, and $x_d \sim \mathrm{Uniform}(g_d(z) - \frac{1}{2n}, g_d(z) + \frac{1}{2n})$. $\Phi(\cdot)$ denotes the standard normal CDF, and $\sigma(\cdot)$ denotes the sigmoid function. Here $n = 20$ and $\tau = 1000$. As n and τ get large, the model converges to a 2-d uniform distribution.

Algorithm 1 Hamiltonian Monte Carlo

Input: previous latent variable z, data x,
step size vector ϵ, number of steps L.
Sample $r \sim \mathcal{N}(0, I)$.
Initialize $z' \leftarrow z$, $r' \leftarrow r$.
for $l = 1$ **to** L **do**
Update $r' \leftarrow r' + \frac{\epsilon}{2} \circ \nabla_z \log p(z', x)$.
Update $z' \leftarrow z' + \epsilon \circ r'$
Update $r' \leftarrow r' + \frac{\epsilon}{2} \circ \nabla_z \log p(z', x)$.
end for
Sample $u \sim \mathrm{Uniform}(0, 1)$.
if $u < \exp\{-\frac{1}{2}||r'||^2 + \frac{1}{2}||r||^2\} \frac{p(z',x)}{p(z,x)}$ **then**
Return z'
else
Return z
end if

MCMC's great advantage is that it allows us to trade computation for accuracy without limit. We can estimate any expectation with respect to the posterior arbitrarily precisely if we run the chain long enough. In contrast, even very powerful variational methods may be limited by the form of the variational approximation.

A particularly powerful MCMC algorithm is Hamiltonian Monte Carlo (HMC, sometimes called hybrid Monte Carlo; Duane et al., 1987; Neal, 1993; 2011). Suppose the goal is to sample from $p(z \mid x) \propto p(z, x)$; HMC augments this model to $p(r, z \mid x) \triangleq \mathcal{N}(r; 0, I)p(z \mid x)$, where the "momentum" variable r has the same dimensionality as z.

HMC transforms the problem of posterior sampling into the problem of simulating the time-evolution of a fictitious physical system. The algorithm interprets the latent vari-

ables z as a position vector, $\log p(z, x)$ as a negative potential energy function, and the Gaussian log-likelihood $-\frac{1}{2}||r||^2 - \frac{D}{2}\log 2\pi$ as a negative kinetic energy function. The sum of the potential and kinetic energy functions defines a *Hamiltonian*, which in turn defines a set of dynamics that are time-reversible, conserve volume, and conserve energy. Those dynamics can be simulated with the "leapfrog" integrator, which is also time-reversible and volume-preserving, and for small enough integration time steps is approximately energy-conserving over long trajectories (Leimkuhler & Reich, 2004). Since the leapfrog integrator is reversible and volume-preserving, we can use it to define a *deterministic* Metropolis proposal. The acceptance probability of this proposal is $\min\{1, \frac{p(r',z',x)}{p(r,z,x)}\}$, which depends only on the difference in energy between the time-evolved state r', z' and the original state r, z; if the simulation is accurate, that difference will be small and the probability of acceptance will be high.

Algorithm 1 outlines the procedure for a single iteration of HMC. Each iteration, we resample the momentum r, apply L iterations of the symplectic "leapfrog" integrator, and then apply a Metropolis correction to accept or reject the proposal z'. Each call requires L calls to the gradient function $\nabla_z \log p(z', x)$—each leapfrog update begins with the gradient from the previous update, and if the algorithm is applied repeatedly then we can cache the initial gradient $\nabla_z \log p(z, x)$.

The algorithm can be used with separate step sizes for each dimension of z and r (Neal, 2011), which is analogous to using a diagonal preconditioner in gradient descent. The optimal step size vector will depend on the problem, but we can get intuitions from the idealized case where $p(z \mid x) = \prod_k \mathcal{N}(z_k; \mu_k, \sigma_k)$. In this case, the optimal setting is $\epsilon_k \propto \sigma_k$; this allows each dimension to be explored equally quickly. We will use this intuition below.

3. Practical MCMC for DLGMs

We are interested in optimizing the average marginal log-likelihood

$$\frac{1}{N}\sum_n \log p_\theta(x_n) = \frac{1}{N}\sum_n \log \int_z p_\theta(z, x_n) dz. \quad (5)$$

Its gradient with respect to θ is

$$\frac{1}{N}\sum_n \nabla_\theta \log \int_z p_\theta(z, x_n) dz = \frac{1}{N}\sum_n \int_z p_\theta(z \mid x_n) \nabla_\theta \log p_\theta(z, x_n) dz, \quad (6)$$

since this is the gradient of the ELBO when $q(z \mid x) = p(z \mid x)$ and the ELBO is tight (Dempster et al., 1977).

The standard VAE training recipe typically approximates this expectation by replacing $p(z \mid x)$ with a more tractable distribution $q(z \mid x)$ and computing a Monte Carlo estimate of $\frac{1}{N}\sum_n \mathbb{E}_q[\nabla_\theta \log p(z, x_n)]$:

$$\gamma^{\text{VAE}} = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \nabla_\theta \log p_\theta(z_s, x_s), \quad (7)$$

where $\mathcal{S}$ is a randomly (or cyclically) chosen set of integers and z_s is drawn from $q(z \mid x_s)$. That is, the expectation with respect to q is approximated with a single sample, and the expectation with respect to the training set is approximated with a minibatch. These stochastic approximations introduce noise, but no bias.

However, replacing the true posterior $p(z \mid x)$ with $q(z \mid x)$ does introduce bias that cannot be averaged away. We would like to reduce or eliminate this bias.

A naive approach would be to simply run HMC from some generic initialization to estimate $\nabla_\theta \log p(x)$. However, if we do not run the chain for long enough, this approach may actually produce a *more* biased estimate, since it may not have had time to forget its initialization. Running it for "long enough" to compete with a variational approximation may be quite expensive. Instead, we propose using HMC to *improve* on an initial variational approximation.

The core idea is this: initialize an HMC sampler with a sample from a variational approximation, run it for a small number of iterations, and use the last sample to get a (hopefully nearly unbiased) estimate of $\nabla_\theta \log p(x)$. More formally, we define the refined distribution q' as

$$q'_{\epsilon,L,M}(z \mid x) \triangleq \int_{\tilde{z}} q(\tilde{z} \mid x) \text{HMC}_{\epsilon,L,M}(z \mid \tilde{z}, x) d\tilde{z}, \quad (8)$$

where $\text{HMC}_{\epsilon,L,M}(z \mid \tilde{z}, x)$ is the distribution of the last sample from an M-step HMC chain with step-size vector ϵ and L leapfrog steps per iteration. In this approach, we substitute q' for q when estimating the gradient of $\log p_\theta(x)$.

Regardless of how many HMC refinement steps we run, q' is guaranteed to have lower KL divergence to the posterior $p(z \mid x)$ than q does (Cover & Thomas, 1991, section 2.9). In fact, in some cases the KLD from $q'(z \mid x)$ to $p(z \mid x)$ may go down *exponentially* fast in M (Ottobre & Pavliotis, 2011). This is an appealing property. The bias in the gradient of the ELBO comes precisely from the $\text{KL}(q_{z|x} \,||\, p_{z|x})$ term, and as that term approaches 0 so too must its gradient (since 0 is the minimum possible KLD).

This observation suggests that we should still care about making the initial variational distribution $q(z \mid x)$ as close as possible to the posterior $p(z \mid x)$, since that should reduce the number of HMC steps needed to bring q' tolerably close to the posterior. To that end, we use a standard mean-field inference network approach as proposed by Kingma & Welling (2014) and Rezende et al. (2014). This inference network's parameters ϕ are trained as usual to maximize

$\mathbb{E}_q[\log \frac{p_\theta(z,x)}{q_\phi(z|x)}]$. But, unlike in VAEs, we will not optimize θ using this objective.

3.1. Refinements

In practice, two refinements to the approach outlined above make it possible to achieve good results with far less computation.

Learning a shared shearing matrix to rotate $q(z \mid x)$. DLGMs are unidentifiable up to a rotation. For example, suppose we have two simple DLGMs defined by

$$\begin{gathered} p(z) = \mathcal{N}(z; 0, I);\ g(z) = V \max\{0, Wz + b\} + c; \\ p(x \mid z) = f(x; g(z)); \\ p'(z') = \mathcal{N}(z'; 0, I);\ g'(z') = V \max\{0, WUz' + b\} + c; \\ p'(x \mid z') = f(x; g(z')), \end{gathered} \tag{9}$$

where U is an arbitrary rotation matrix. These models define the same marginal distribution, i.e., $p(x) = p'(x)$. We can see this by substituting $z \triangleq Uz'$; the rotation does not change the marginal distribution of z, and it makes $g(z) = g'(z')$, and therefore $p(x \mid z) = p'(x \mid z')$.

Such a rotation does, however, rotate the posteriors; that is, $p(z \mid x) = p'(U^\top z \mid x)$. This is bad, because mean-field variational inference is not rotationally invariant—it works best when the posterior exhibits no correlations. In normal VAE training, the generative network is optimized to be as friendly as possible to mean-field variational inference, so it learns to rotate its weight matrices to break this rotational symmetry. In our setup, this does not naturally happen; our whole goal is in a sense to avoid letting information about the variational approximation leak through to the generative network.

To address this issue, we introduce an extra lower-triangular matrix A in the generative network that can partially correct for such rotations, so that our model is

$$z \sim \mathcal{N}(0, I); \quad z' \triangleq Az; \quad p(x \mid z) \triangleq f(g(z')). \tag{10}$$

A is optimized with ϕ to maximize the variational ELBO $\mathbb{E}_q[\log \frac{p_\theta(z,x)}{q_\phi(z|x)}]$. A is constrained to have all of its diagonal elements equal to 1 so that it cannot prune out any latent dimensions. If $q(z \mid x) = \mathcal{N}(z; \mu(x), \mathrm{diag}(\sigma(x)^2))$, then $q(z' \mid x) = \mathcal{N}(z'; A\mu(x), A\mathrm{diag}(\sigma(x)^2)A^\top)$, so adding A allows $q(z' \mid x)$ to have correlation structure. We found that adding this correlation structure significantly improves the quality of the initial mean-field approximation. In conjunction with the per-variable step size technique outlined below, we found that using this shearing matrix speeds up the algorithm by 2–3x.

Setting per-variable step sizes and the number of leapfrog steps. HMC has two critical tuning parameters: the number of leapfrog steps L, and the step size vector ϵ.

As in gradient descent, large step sizes allow for fast progress, but step sizes that are too large lead to unstable results (and therefore Metropolis rejections). It makes sense to use smaller step sizes in the dimensions with larger gradients and more oscillatory dynamics.

If L is too small, then the algorithm will make small updates each iteration, leading to random-walk behavior and slow mixing (Neal, 2011). If L is too large, eventually the trajectory will turn back and retrace its steps, wasting computation. Hoffman & Gelman (2014) proposed the no-U-turn sampler (NUTS), a variant that automatically determines an appropriate number of steps, but NUTS is a complicated recursive algorithm that is quite difficult to implement in computational graph languages such as TensorFlow (Abadi et al., 2016).

We adopt a simple heuristic for setting the step sizes and leapfrog steps based on intuitions from the case where the posterior is Gaussian with diagonal covariance. In this case, the continuous-time Hamiltonian dynamics in each dimension k are sinusoidal with period $2\pi\sigma_k$, where σ_k is the scale of dimension k. If we use a step size vector ϵ such that $\epsilon_k = \epsilon_0\sigma_k$, then after L steps of leapfrog integration the dynamics in each dimension will have advanced by $\frac{\epsilon_0 L}{2\pi}$ periods. We therefore set $L = \lceil \frac{1}{\epsilon_0} \rceil$, so that each HMC iteration simulates approximately $(2\pi)^{-1} \approx 0.16$ periods. This number is somewhat arbitrary; it is chosen to be slightly less than 0.25 periods. We found that using larger integration times sometimes caused undesirable resonances and aliasing (Neal, 2011).

We tune ϵ_0 throughout training to give a reasonable worst-case acceptance rate. In each minibatch, we compute the average acceptance rate over the M HMC iterations for each example in the minibatch. If the smallest average acceptance rate is less than 0.25, we decrease ϵ_0 by 0.5%; otherwise we increase ϵ_0 by 0.5%. This worst-case procedure ensures that ϵ_0 is small enough to allow the chains for *all* training examples to mix; simply tuning the average acceptance rate would not guarantee this. Although this resembles an adaptive MCMC algorithm (Andrieu & Thoms, 2008), note that adaptation is done between MCMC runs, so the usual caveats and requirements of vanishing adaptation do not apply.

We summarize our HMC-based approach to DLGM training in algorithm 2. Figure 2 sketches a specific training architecture.

3.2. Evaluating held-out likelihood

A benefit of variational methods over MCMC is that they immediately provide easy-to-estimate lower bounds on the marginal likelihood. A relatively reliable (but expensive)

Algorithm 2 Hamiltonian Monte Carlo for DLGMs

Input: dataset $\mathcal{D}$, SGD step size schedule $\gamma(t)$, minibatch size S, initial HMC step size ϵ_0, number of HMC steps M, number of iterations T.
Randomly initialize the inference network parameters ϕ, the generative network parameters θ, and the shearing matrix A.
for $t = 1$ **to** T **do**
  Compute $L = \lceil \epsilon_0(t)^{-1} \rceil$.
  Select a minibatch $\mathcal{S}$ from the dataset $\mathcal{D}$.
  for $x_s \in \mathcal{S}$ **do**
    Sample $z_{s,0} \sim \mathcal{N}(\mu_\phi(x_s), \sigma_\phi(x_s))$.
    Compute $\gamma^s_\phi = \nabla_\phi \log \frac{p_\theta(z_{s,0}, x_s)}{\mathcal{N}(z_{s,0}; \mu_\phi(x_s), \sigma_\phi(x_s))}$.
    Compute $\gamma^s_A = \nabla_A \log \frac{p_\theta(z_{s,0}, x_s)}{\mathcal{N}(z_{s,0}; \mu_\phi(x_s), \sigma_\phi(x_s))}$.
    Compute $\epsilon_s = \epsilon_0(t)\sigma_\phi(x_s)$.
    for $m = 1$ **to** M **do**
      sample $z_{s,m}$ from $\mathrm{HMC}(z_{s,m-1}, x_s, \epsilon_s, L)$.
    end for
    Compute $\gamma^s_\theta = \nabla_\theta \log p_\theta(z_{s,M}, x_s)$.
  end for
  Apply gradient update to ϕ using $\gamma_\phi = \frac{1}{S}\sum_s \gamma^s_\phi$.
  Apply gradient update to θ using $\gamma_\theta = \frac{1}{S}\sum_s \gamma^s_\theta$.
  Apply gradient update to A using $\gamma_A = \frac{1}{S}\sum_s \gamma^s_A$, set $A_{k,k} = 1$, $A_{k,l} = 0$ for all $k \in \{1, \ldots, K\}$ and $l > k$.
end for

way to use MCMC to estimate marginal likelihoods is annealed importance sampling (AIS; Neal, 2001), and we follow Wu et al. (2016) in using AIS to evaluate our trained models on held-out data. We use 20 samples per test example, 1000 annealing steps, and the same HMC step size and number of leapfrog steps as during training.

4. Comparison to Related Work

The idea of using MCMC to improve variational approximations has come up a number of times (e.g., De Freitas et al., 2001; Mimno et al., 2012).

Our approach is most closely related to the Hamiltonian variational inference (HVI) approach proposed by Salimans et al. (2015). HVI also initializes an HMC algorithm with a sample from an inference network. But HVI learns to explicitly lower-bound the entropy of the marginal distribution of the final sample from the MCMC chain, making it possible to directly optimize and estimate a bound on the ELBO obtained by using this marginal distribution as a variational distribution. By contrast, we make no attempt to estimate that ELBO, and instead optimize our variational parameters ϕ to maximize the much looser ELBO defined by the initial variational distribution. This is a relative strength of HVI.

Our approach also has several relative strengths over HVI. First, it is simpler—HVI requires training an auxiliary inverse inference network to reverse the HMC Markov chain that approximates the posterior. Second, to the extent that this inverse network cannot exactly reverse the Markov chain, the HVI lower bound on the ELBO will not be tight, reintroducing some bias in the gradient estimates. Finally, using HVI with Markov chains that involve multiple accept/reject steps is difficult and/or expensive, since the inverse inference network must then learn to infer the binary accept/reject random variables. This last issue is highlighted by our results in section 5, where (like Wolf et al., 2016) we find that using multiple HMC steps is important.

More recently, Han et al. (2017) proposed using Metropolis-adjusted Langevin (i.e., HMC with one leapfrog step) to fit DLGMs in a stochastic EM framework. Their approach is effective for small datasets, but it is inherently a batch method that involves maintaining a Markov chain for each example throughout training. This limits its ability to scale to very large datasets.

There has been a flurry of work in the last few years developing more and more accurate variational approximations for VAEs (Salimans et al., 2015; Rezende & Mohamed, 2015; Burda et al., 2016; Tran et al., 2016; Sønderby et al., 2016; Kingma & Salimans, 2016). This work has demonstrated the benefits of using accurate posterior approximations, which originally motivated us to explore using highly accurate MCMC approximations.

5. Experiments

Below, we compare the results of our proposed method to a baseline mean-field VAE and to other published results on the binarized MNIST dataset (LeCun et al., 1998). In all experiments, we used a simple fully connected architecture with one stochastic layer of 64 latent variables and two deterministic hidden layers with 1024 hidden units each and softplus nonlinearities. The architecture is sketched in figure 2. For MNIST, we used 60,000 images for training and held out 10,000 for testing. The training images were re-binarized each epoch to prevent overfitting (as done by, e.g., Burda et al., 2016). All models were optimized using Adam (Kingma & Ba, 2015) with default parameters and a minibatch size of 250. We trained all models for 500 epochs with a learning rate of 0.001, then for another 100 epochs with a learning rate of 0.0001.

5.1. Held-out likelihoods

We estimate held-out likelihood using annealed importance sampling (AIS; Neal, 2001). Performance as a function of number of steps of HMC is summarized in figure 3.

Using more HMC steps during training clearly leads to

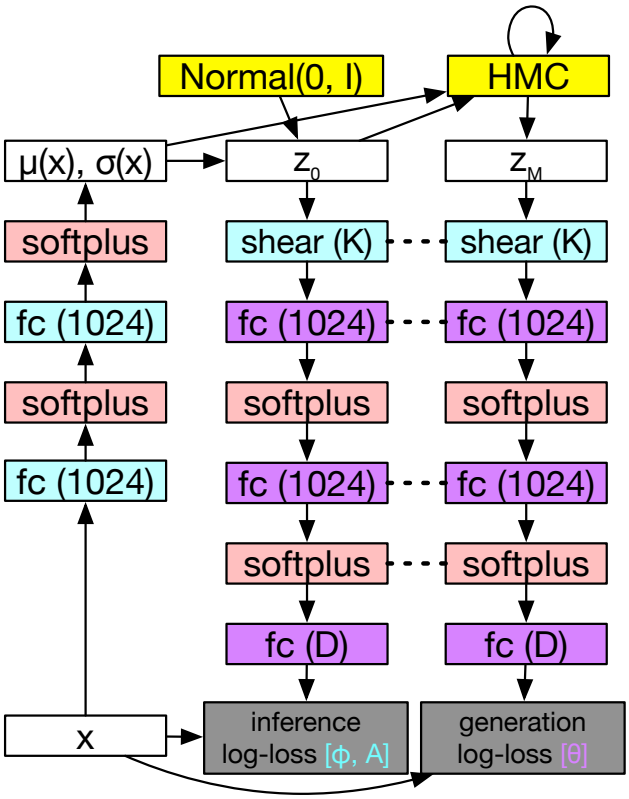

Figure 2. Architecture used in the experiments. Dotted lines denote weight sharing. "fc (N)" denotes a dense matrix multiply with output dimension N. "shear (N)" denotes a unit-diagonal lower-triangular matrix multiply with output dimension N. Parameters to cyan nodes are optimized with respect to the inference log-loss, parameters to purple nodes are optimized with respect to the generation log-loss.

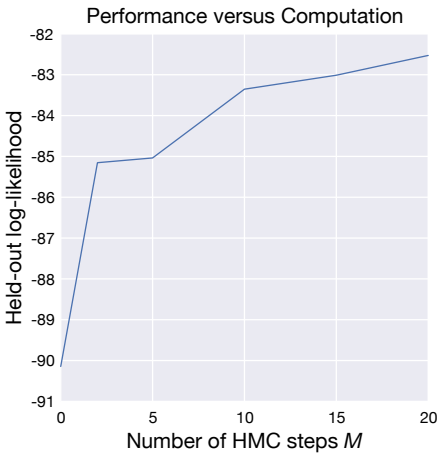

Figure 3. Held-out likelihood as a function of number of HMC steps M used during training. More computation leads to better results, but even a few HMC steps provide a major advantage over standard VAE training ($M = 0$).

Table 1. Reported held-out log-likelihoods on dynamically binarized permutation-invariant MNIST, with number of stochastic layers. HMC-DLGM-2 and HMC-DLGM-20 denote our approach with 2 and 20 HMC steps, respectively. Other approaches are due to [1] Salimans et al. (2015), [2] Burda et al. (2016), [3] Tran et al. (2016), [4] Sønderby et al. (2016).

MODEL	# LAYERS	$\leq \log p(x)$
VAE BASELINE	1	-90.14
HVI [1]	1	-85.51
HMC-DLGM-2	1	-85.15
IWAE [2]	1	-84.78
IWAE [2]	2	-82.90
HMC-DLGM-20	1	-82.53
VGP [3]	2	-81.90
LVAE [4]	5	-81.74

better models. Our best log-likelihood of -82.53 compares favorably with other reported values in the literature for permutation-invariant (i.e., non-convolutional) binarized MNIST models (see table 1). These number could doubtless be improved—we did very little experimentation with architectural choices. For example, all models in table 1 that outperform ours (and some that don't) use 2–5 stochastic layers, whereas we use only one.

The held-out likelihood with two HMC steps is very close to that obtained by Salimans et al. (2015), who used only one HMC update due to the computational difficulty of using many Metropolis accept/reject steps in their framework. This suggests that the improved performance of our approach is indeed due to the use of many HMC steps (rather than due to architecture, hyperparameters, etc.).

The per-minibatch cost of our approach goes up linearly with M, the number of HMC steps. Training an MNIST model with $M = 20$ took about 8 hours on an NVIDIA K20 GPU, 21 times longer than with $M = 1$[1]. It may be instructive to compare this number with the cost of training an IWAE (Burda et al., 2016), which scales linearly in the number of samples used during training—Burda et al. (2016) used 50 samples in their experiments.

5.2. Pruning and blurriness

In this section, we evaluate some qualitative features of the MNIST model trained with our HMC-based procedure.

Figure 4 shows some samples from the VAE and from the DLGM trained with $M = 20$. The VAE samples are blurry, while the HMC-trained model generates sharp samples. Some of the HMC-trained model's samples look strange,

[1]This is less than might be expected, given that each HMC iteration applies several leapfrog steps (typically 2–3). The cost of a leapfrog step is less than the cost of computing gradients in a VAE, since we need not backprop through the inference network.

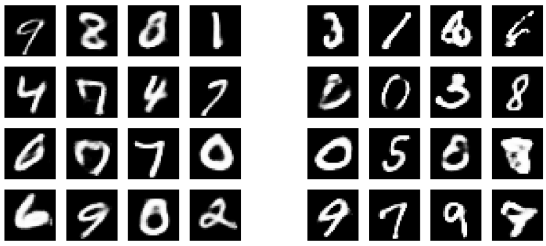

Figure 4. Left: Samples from a VAE. Right: Samples from a DLGM trained with HMC. More examples are in the supplement.

but many look good, which is consistent with the observation of Theis et al. (2016) that maximum-likelihood training places a relatively small penalty on generating many bad samples (e.g., generating 95% garbage costs only 3 nats of log-likelihood).

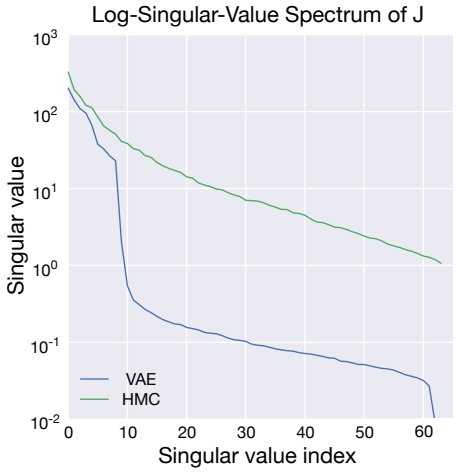

Figure 5. Influence of latent directions on output space.

Following Krishnan & Hoffman (2016), to determine which directions in z-space are most and least important, we sample 10,000 random vectors $z_{1:10000}$ from their $\mathcal{N}(0, I)$ marginal distribution and compute the singular value decomposition of the average Jacobian matrix $\mathcal{J}$ of the expected observation vector with respect to z:

$$\mathcal{J} \triangleq \frac{1}{10000} \sum_{i=1}^{10000} \left.\frac{\partial g_\theta(z)}{\partial z}\right|_{z_i} = USV^\top. \qquad (11)$$

Figure 5 shows the singular value spectra given by the diagonal of S for the VAE and the HMC-trained DLGM. The VAE's spectrum falls off sharply at about 10, showing that the remaining 118 dimensions are not used to explain the data. The spectrum for the DLGM trained with HMC falls off much more gradually.

To get a qualitative idea of what is encoded in which dimensions, we can marginalize out all but the first C latent dimensions to estimate

$$\mathbb{E}[x|z_{1:C}] = \int_{z'} \mathcal{N}(z'; 0, I)\mathbb{E}[x|z = (z_{1:C}, z')]dz', \quad (12)$$

where z' is a vector of dimension $K - C$, and $(\cdot, \cdot)$ denotes concatenation. If we first rotate z by V, then $z_{1:C}$ will in a sense denote the C most-important directions in z-space, and information that is encoded by the less-important directions $z_{C+1:K}$ will be blurred out in $\mathbb{E}[x \mid z_{1:C}]$.

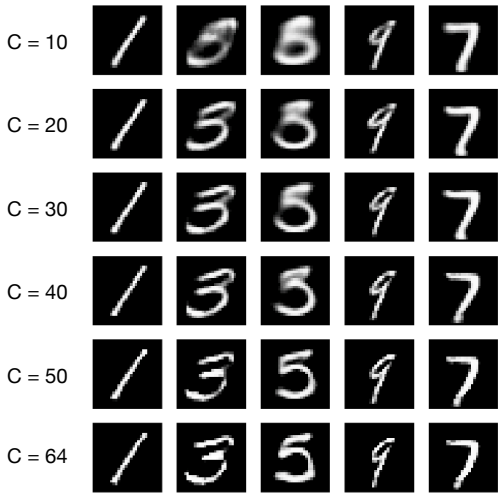

Figure 6. Effects of keeping only the first C most important dimensions of z, marginalizing out the remaining dimensions. More examples are in the supplement.

Figure 6 displays $\mathbb{E}[x \mid z_{1:C}]$ for various values of C for five z vectors[2] for the model trained with $M = 20$. Identity and broad structure is indeed encoded by the lower-order principal components, but the images remain somewhat blurry until many higher-order components are added. This suggests that variational pruning is at least partially responsible for some of the blurriness associated with VAEs.

6. Conclusion

We have proposed a practical approach to using HMC to train DLGMs. This approach yields less-biased gradients of the true marginal likelihood of the data than mean-field VAEs, and thereby learns better generative models. It is more expensive than standard VAE training, but not prohibitively so.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Gregory S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu,

[2]The z vectors are randomly sampled, but curated due to space constraints. Uncurated examples are in the supplement.

Ghemawat, Sanjay, Goodfellow, Ian J., Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Józefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek Gordon, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul A., Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda B., Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Andrieu, Christophe and Thoms, Johannes. A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373, 2008.

Bishop, C. *Pattern Recognition and Machine Learning*. Springer New York., 2006.

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

Bowman, Samuel R, Vilnis, Luke, Vinyals, Oriol, Dai, Andrew M, Jozefowicz, Rafal, and Bengio, Samy. Generating sentences from a continuous space. *CoNLL*, pp. 10, 2016.

Burda, Yuri, Grosse, Roger, and Salakhutdinov, Ruslan. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

Cover, Thomas M and Thomas, Joy A. *Elements of information theory*. John Wiley & Sons, 1991.

De Freitas, Nando, Højen-Sørensen, Pedro, Jordan, Michael I, and Russell, Stuart. Variational MCMC. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 120–127. Morgan Kaufmann Publishers Inc., 2001.

Dempster, A., Laird, N., and Rubin, D. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

Duane, Simon, Kennedy, Anthony D, Pendleton, Brian J, and Roweth, Duncan. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

Gelman, Andrew and Shirley, Kenneth. Inference from simulations and monitoring convergence. In *Handbook of Markov chain Monte Carlo*, pp. 163–174. Chapman and Hall/CRC, 2011.

Han, Tian, Lu, Yang, Zhu, Song-Chun, and Wu, Ying Nian. Alternating back-propagation for generator network. In *AAAI Conference on Artificial Intelligence*, 2017.

Hoffman, Matthew D. and Blei, David M. Structured stochastic variational inference. In *International Conference on Artificial Intelligence and Statistics*, pp. 361–369, 2015.

Hoffman, Matthew D. and Gelman, Andrew. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. Introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. In *International conference on learning representations*, 2015.

Kingma, Diederik P and Salimans, Tim. Improving variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.

Kingma, D.P. and Welling, Max. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Krishnan, Rahul G and Hoffman, Matthew D. Inference & introspection in deep generative models of sparse data. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2016.

LeCun, Yann, Cortes, Corinna, and Burges, Christopher JC. The mnist database of handwritten digits, 1998.

Leimkuhler, Benedict and Reich, Sebastian. *Simulating hamiltonian dynamics*, volume 14. Cambridge University Press, 2004.

MacKay, David J. C. Local minima, symmetry-breaking, and model pruning in variational free energy minimization. Technical report, 2001.

Mimno, D., Hoffman, M., and Blei, D. Sparse stochastic inference for latent Dirichlet allocation. In *International Conference on Machine Learning*, 2012.

Neal, R. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

Neal, Radford M. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Neal, Radford M. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, pp. 113–162. Chapman and Hall/CRC, 2011.

Ottobre, M and Pavliotis, GA. Asymptotic analysis for the generalized Langevin equation. *Nonlinearity*, 24(5): 1629, 2011.

Rezende, Danilo and Mohamed, Shakir. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Salimans, Tim, Kingma, Diederik P, Welling, Max, et al. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, volume 37, pp. 1218–1226, 2015.

Sønderby, Casper Kaae, Raiko, Tapani, Maaløe, Lars, Sønderby, Søren Kaae, and Winther, Ole. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 3738–3746, 2016.

Theis, L., van den Oord, A., and Bethge, M. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016. URL http://arxiv.org/abs/1511.01844.

Theis, Lucas and Hoffman, Matthew D. A trust-region method for stochastic variational inference with applications to streaming data. In *International Conference on Machine Learning*, pp. 2503–2511, 2015.

Tran, D., Ranganath, R., and Blei, D. The variational Gaussian process. In *International Conference on Learning Representations*, 2016.

Wolf, Christopher, Karl, Maximilian, and van der Smagt, Patrick. Variational inference with Hamiltonian Monte Carlo. *arXiv preprint arXiv:1609.08203*, 2016.

Wu, Yuhuai, Burda, Yuri, Salakhutdinov, Ruslan, and Grosse, Roger. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.