GENERATIVE ADVERSARIAL LEARNING OF MARKOV CHAINS

Jiaming Song, Shengjia Zhao & Stefano Ermon
Computer Science Department
Stanford University
{tsong,zhaosj12,ermon}@cs.stanford.edu

ABSTRACT

We investigate generative adversarial training methods to learn a transition operator for a Markov chain, where the goal is to match its stationary distribution to a target data distribution. We propose a novel training procedure that avoids sampling directly from the stationary distribution, while still capable of reaching the target distribution asymptotically. The model can start from random noise, is likelihood free, and is able to generate multiple distinct samples during a single run. Preliminary experiment results show the chain can generate high quality samples when it approaches its stationary, even with smaller architectures traditionally considered for Generative Adversarial Nets.

1 INTRODUCTION

A large number of deep generative models are *implicit* models, where a stochastic procedure is used to directly generate data without having to define a tractable likelihood function (Mohamed & Lakshminarayanan, 2016). There are two popular ways of sampling from implicit models: ancestral and iterative sampling. *Ancestral sampling* involves a single pass over all the variables in the model: each variable is sampled conditionally on its predecessors, based on an ordering specified by a directed model. Popular frameworks include the Variational Autoencoder (VAE, Kingma & Welling (2013); Rezende et al. (2014)) and Generative Adversarial Network (GAN, Goodfellow et al. (2014)). Alternatively, *iterative sampling* involves multiple passes over all the variables, iteratively improving the quality of the sample. Typically, the process involves simulating a Markov chain over the entire state space, and is often the method of choice for undirected models (Hastings, 1970). Several recent works (Bengio et al., 2013; 2014; Sohl-Dickstein et al., 2015; Bordes et al., 2017) have discussed procedures for learning iterative models, where samples are obtained by iterating over a neural network; iterative sampling, however, has generally received less attention compared to ancestral sampling.

In this work, we consider the general case of iterative sampling in which we train a Markov chain to mix quickly towards a given stationary distribution, starting from random noise. We utilize generative adversarial training (Goodfellow et al., 2014), which only requires samples from the chain, allowing for a likelihood-free approach. Empirical results show that we are able to train fast mixing Markov Chains with a stationary distribution close to the desired one.

2 PROBLEM SETUP

Let $\mathcal{S}$ be the state space for the sequence of random variables $\mathcal{X} = \{\mathbf{X}_t\}_{t=0}^{t=\infty}$, $\mathbf{X}_t \in \mathcal{S}$. Let π^0 be an initial probability distribution for $\mathbf{X}_0$, and $T_\theta(\cdot|\mathbf{x})$ be a transition kernel parameterized by θ, e.g., using a neural network. We assume T_θ is easy to sample from, and is a valid transition kernel for any choice of θ, i.e., it satisfies $\int_{\mathcal{S}} T_\theta(\mathbf{x}'|\mathbf{x})\mathrm{d}\mathbf{x}' = 1$ for all $\mathbf{x} \in \mathcal{S}$. Therefore, every T_θ defines a time-homogeneous Markov chain over $\mathcal{X}$. We denote $\pi_\theta^t(\mathbf{x})$ the resulting probability distribution at time step t. If we assume that $T_\theta(\mathbf{x}_t|\mathbf{x}_{t-1}) > 0$ for all $\mathbf{x}_t, \mathbf{x}_{t-1} \in \mathcal{S}$, then the Markov chain defined by T_θ is both irreducible and positive recurrent, and hence has a unique stationary

distribution $\pi_\theta = \lim_{t\to\infty} \pi_\theta^t$. For all $\mathbf{x} \in \mathcal{S}$, π_θ satisfies

$$\pi_\theta(\mathbf{x}) = \int_{\mathcal{S}} T_\theta(\mathbf{x}|\mathbf{x}')\pi_\theta(\mathbf{x}')\mathrm{d}\mathbf{x}' \tag{1}$$

Suppose there is an unknown distribution $p_d(\mathbf{x})$ from which we can obtain samples from, e.g., a data distribution. Our goal here is twofold: we want to find a θ such that 1) π_θ is close to $p_d(\mathbf{x})$, and 2) the corresponding Markov Chain mixes quickly.

3 Adversarial Training of Markov Chains

Although π_θ exists for any θ due to the uniqueness of the stationary distribution, calculating the actual likelihood of $\mathbf{x}$ under that distribution is intractable in most cases. However, it is straightforward to obtain samples from π_θ^t, which will be close to π_θ if t is large enough. This aligns well with the framework of GANs, which only requires the ability to sample from the model.

Generative Adversarial Network (GAN) (Goodfellow et al., 2014) is a framework for training deep generative models using a two player minimax game. GANs train a generator network G to generate samples by transforming a noise variable $\mathbf{z} \sim p(\mathbf{z})$ into $G(\mathbf{z})$. A discriminator network $D(\mathbf{x})$ is trained to distinguish between samples from the generator and true samples from a given data distribution p_d. Formally, this defines the following objective

$$\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_d}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2}$$

In our setting, we could choose $\mathbf{z} \sim \pi^0$ and let $G_\theta(\mathbf{z})$ be the state of the Markov Chain after t steps, which is a good approximation of π_θ if t is large enough. However, we would run into optimization problems, because the gradient is required to back propagate through the entire chain, resulting in an expensive gradient step update, while having slow convergence due to high variance in the estimated gradients. Therefore, we propose a more efficient approximation, with the following objective:

$$\min_\theta \max_D \mathbb{E}_{\mathbf{x}\sim p_d}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim \pi_\theta^{\bar{t}}}[\log(1 - D(\mathbf{x}))] + \mathbb{E}_{\mathbf{x}_d\sim p_d, \mathbf{x}\sim T_\theta^{\hat{t}}(\mathbf{x}|\mathbf{x}_d)}[\log(1 - D(\mathbf{x}))] \tag{3}$$

where $T_\theta^{\hat{t}}(\mathbf{x}|\mathbf{x}_d)$ denotes the distribution of $\mathbf{x}$ when the transition kernel is applied $\hat{t}$ times, starting from $\mathbf{x}_d$. We use two types of samples from the generator for training, optimizing θ such that the samples will fool the discriminator:

1. Sample in $\bar{t}$ steps, given an initial sample $\mathbf{x}_0 \sim \pi^0$.
2. Sample in $\hat{t}$ steps, given a data sample $\mathbf{x} \sim p_d$ with some small random perturbation.

Intuitively, the first condition encourages the Markov Chain to converge towards p_d over relatively short runs (of length t). If we only consider this requirement, the approach would correspond to ancestral sampling in a latent variable model, as in the cases of Sohl-Dickstein et al. (2015), Salimans et al. (2015) and Bordes et al. (2017). However, in contrast with these models, our goal is train an iterative procedure, where the quality of the samples can be improved by increasing the number of simulation steps, and multiple samples can be cheaply generated after the burn-in period of the chain. This is accomplished by the second condition, which enforces convergence to the stationary, where each point from p_d has to transition to another point on the data manifold. [1]

The objective in Equation 3 is much easier to optimize than Equation 2 for the stationary distribution. Instead of sampling the chain until convergence, which will be especially time-consuming if the initial Markov chain takes many steps to mix, the generator would run only $(\bar{t} + \hat{t})/2$ steps on average, with the advantage of estimating gradients with lower variance.

4 Experiments

We train our model on the MNIST dataset, where the goal is to match the data generating distribution with π_θ, and we prefer fitting complex distributions with simple transition operators. We consider

[1] We provide a more rigorous justification in Appendix A.

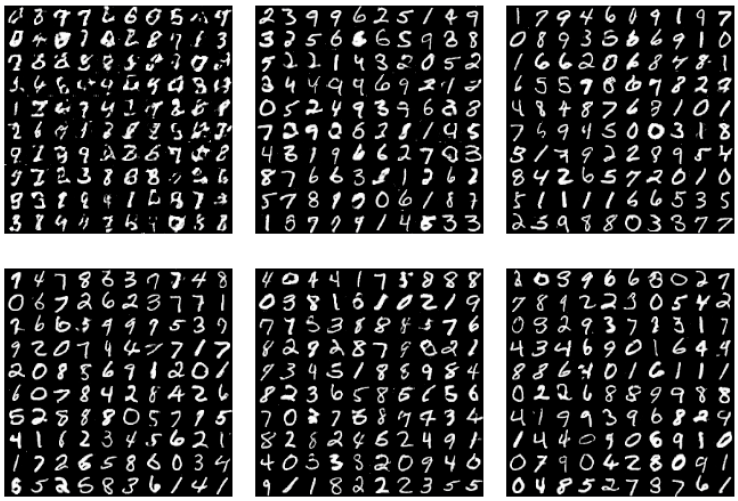


Figure 1: Samples from a chain with the **mlp** architecture. From top left to bottom right, each subplot are samples from π_θ^1, π_θ^2, π_θ^5, π_θ^{10}, π_θ^{20}, π_θ^{50}, respectively. The figure is generated by starting with a batch of 100 initial samples from $\mathbf{x}_0$, and repeatedly applying the transition operator to it.

three types of architectures for our transition operator $T_\theta(\cdot|\mathbf{x})$. Each has a symmetric encoder-decoder architecture where we inject factored Gaussian noise into the latent code. The decoder architectures are respectively:

1. The generative network architecture for DCGAN (Radford et al., 2015), which has two fully connected layers followed by two transposed convolutions. This model is powerful enough to generate sharp images in one step. (**dcgan**)
2. A weaker DCGAN, with a fully connected layer and a transposed convolution. (**conv**)
3. A MLP composed of two fully connected layers, which is the weakest model. (**mlp**)

To see whether the stationary distribution closely matches the data distribution, we visualize samples of π^t at different time steps t for the models in Figure 1 for **mlp**, Figure 2 for **conv** and Figure 3 for **dcgan**[2]. π^0 is a factored Gaussian distribution with mean and standard deviation being the mean and standard deviation of the training set.

In the case of **conv** and **mlp**, where it is difficult to generate clear images in one step, the model is able to generate sharp images by running the chain. Remarkably, the model is able to generalize to longer runs (such as 10, 20 and 50), even if the operator was trained for shorter simulations. In addition, running the chain from a single sample in π^0 will not result in convergence to a particular sample. At each step the class distribution is relatively balanced, without indication of missing particular modes.

5 Conclusion and Future Work

We presented an efficient generative adversarial training method for learning the transition operator in a Markov chain, with few conditions enforced on the model. In extension, we are interested in applying this method to larger datasets, as well as performing detailed analysis over the effect of hyperparameters. It is also interesting to consider chains with certain desirable properties, such as detailed balance.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2]Figures 2 and 3 are in Appendix C.

Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907, 2013.

Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. 2014.

Floria Bordes, Sina Honari, and Pascal Vincent. Learning to generate samples from noise through infusion training. *ICLR*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Tim Salimans, Diederik P Kingma, Max Welling, et al. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, volume 37, pp. 1218–1226, 2015.

Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

A JUSTIFICATION FOR OBJECTIVE IN EQUATION 3

We consider two necessary conditions for p_d to be the stationary distribution of the Markov chain, which can be translated into a new algorithm with better optimization properties.

Proposition 1. *Consider a sequence of ergodic Markov chains over state space $\mathcal{S}$. Define π_n as the stationary distribution for the n-th Markov chain, and π_n^t as the probability distribution at time step t for the n-th chain. If the following two conditions hold:*

1. $\exists \bar{t} > 0$ *such that the sequence* $\{\pi_n^{\bar{t}}\}_{n=1}^{\infty}$ *converges to* p_d *in total variation;*
2. $\exists \epsilon > 0$*, such that if* $\|\pi^t - p_d\|_{TV} < \epsilon$*, then* $\|\pi^{t+1} - p_d\|_{TV} < \rho\|\pi^t - p_d\|_{TV}$ *for some* $\rho < 1$*;*

then the sequence of stationary distributions $\{\pi_n\}_{n=1}^{\infty}$ *converges to* p_d *in total variation.*

Proof. The goal is to prove that $\forall \delta > 0, \exists N > 0, T > 0$, such that $\forall n > N, t > T$, $\|\pi_n^t - p_d\|_{\text{TV}} < \epsilon$.

According to the first assumption, $\exists N > 0$, such that $\forall n > N$, $\|\pi_n^{\bar{t}} - p_d\|_{\text{TV}} < \epsilon$.

Therefore, $\forall \delta > 0$, we are able to propose $\exists T = \bar{t} + \max(0, \lceil \log_\rho \delta - \log_\rho \epsilon \rceil)$, such that $\forall t > T$,

$$\begin{aligned} & \|\pi_n^t - p_d\|_{\text{TV}} \\ < & \|\pi_n^{\bar{t}} - p_d\|_{\text{TV}} \cdot \rho^{T-t} \\ < & \epsilon \cdot \frac{\delta}{\epsilon} = \delta \end{aligned} \tag{4}$$

Hence the sequence $\{\pi_n\}_{n=1}^{\infty}$ converges to p_d in total variation. □

Moreover, convergence in total variation distance is equivalent to convergence in Jensen-Shannon (JS) divergence(Arjovsky et al., 2017), which is what GANs try to minimize (Goodfellow et al., 2014), so we can use GANs to achieve the two conditions in Proposition 1. This suggests a new optimization criterion, where we look for a θ that satisfies both conditions in Proposition 1, which translates to Equation 3.

B RELATED WORK

Previous works have considered using Markov chains to represent implicit generative models. Intuitively, a Markov Chain can potentially transition between a large number of different modes even with a simple unimodal transition kernel. Even when the desired target distribution is complex and has a large number of modes, it might still be representable with a relatively simple Markov Chain, as mentioned in Bengio et al. (2014).

Generative Stochastic Networks (Bengio et al., 2013; 2014) use transition operators defined by auto-encoders with a stochastic component, and fit the stationary distribution to training data. However, it requires the chain to start close to training data points, due to the use of autoencoders. Diffusion training (Sohl-Dickstein et al., 2015) obtains a Markov chain by inverting a slow, fixed diffusion process from data to a simple distribution, which may take thousands of tiny steps to mix in the case of high dimensional images. Infusion training (Bordes et al., 2017) allows progressive stochastic denoising from random noise to samples, using an infusion process which biases samples to move towards a specific training data point during training. The goal is to converge to a particular sample in a finite number of denoising steps.

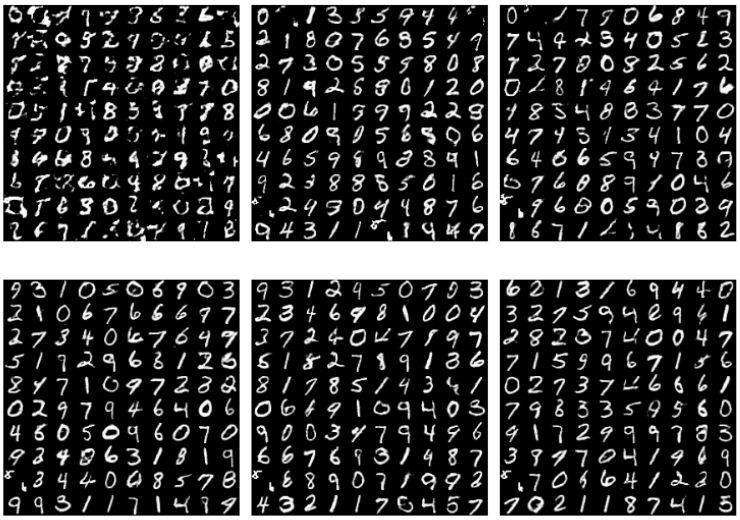


Figure 2: Samples from a chain with the **conv** architecture. Settings are the same as in Figure 1.

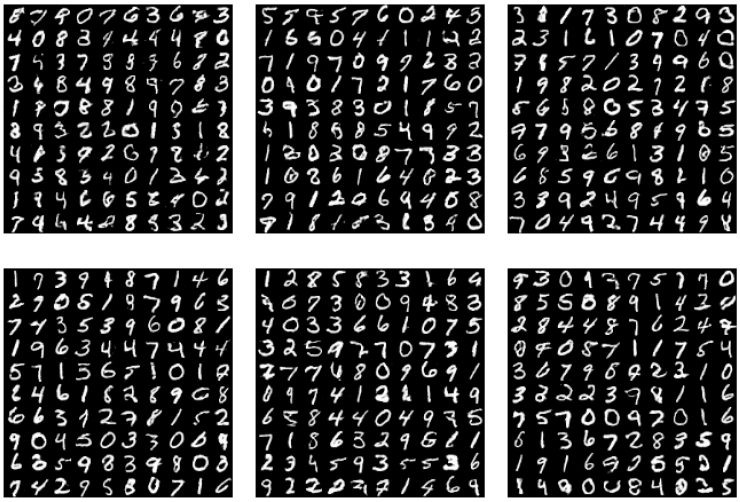


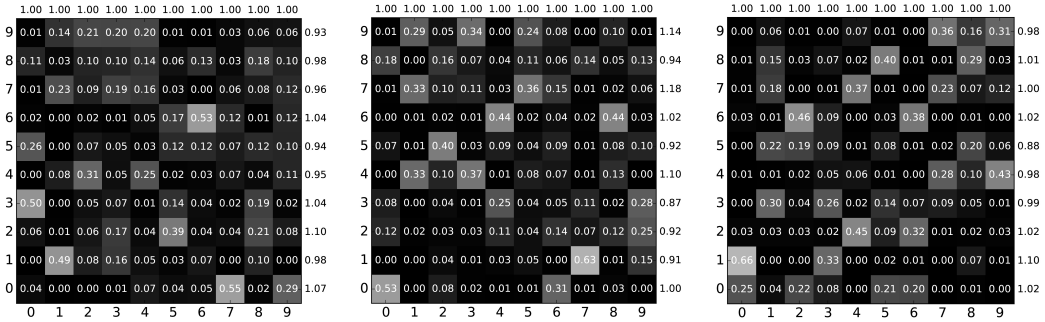
Figure 3: Samples from a chain with the **dcgan** architecture. Settings are the same as in Figure 1.

C Additional Experimental Details

All architectures use the DCGAN discriminator as the discriminator. To stabilize training and avoid missing modes, we consider the Wasserstein GAN (WGAN) framework (Arjovsky et al., 2017), where the objective approximates the Earth-Mover distance. We briefly tried WGAN over MNIST without using the chain, and were unable to generate realistic images with **conv** and **mlp** architectures.

At each iteration, we obtain $\bar{t}$ and $\hat{t}$ by uniformly sampling from $\{1, 2, \ldots, T\}$, where T is a hyper-parameter we set to 4 for $\bar{t}$ and 2 for $\hat{t}$. We empirically find applying this would improve convergence speed, whereas the conclusion in Proposition 1 still holds.

To further investigate the frequency of various modes in the stationary distribution, we consider the class-to-class transition probabilities, to see if Equation 1 holds for our model. We run one step of the transition operator starting from real data samples, where we have class labels y, and classify the generated samples using a convolutional network classifier that has 99% test accuracy on the MNIST test set. Given the labels y, we are thus able to quantify $T(y_t|y_{t-1})$, corresponding to a



(a) $T(y_t|y_{t-1})$ for **dcgan** (b) $T(y_t|y_{t-1})$ for **conv** (c) $T(y_t|y_{t-1})$ for **mlp**

Figure 4: Example transition matrices $T(y_t|y_{t-1})$, where the x-axis and y-axis denote y_{t-1} and y_t respectively. We normalize the sum of each column to 1. Each number within grid (a, b) is the probability of $T(b|a)$. The numbers on the top and right corresponds to the sum of values for each individual column and row. For visualization purposes, we truncate the values to have two floating points of precision. Multiple runs over the same architecture will result in different transitions.

Markov Chain with 10 states. Results show that although the stationary is not perfectly uniform among different classes, class probabilities are fairly uniform and range between 0.09 and 0.11 [3]. The transition probabilities indicate that detailed balance conditions are not satisfied, which is reasonable given that we do not pose any restrictions on T_θ to satisfy this condition.

[3]Given the transition matrix we are able to compute the stationary distribution for each class.