Adaptive Strategy for Resetting a Non-stationary Markov Chain during Learning via Joint Stochastic Approximation

Hyunsu Kim KIM.HYUNSU@KAIST.AC.KR
School of Computing, KAIST, Daejeon, South Korea

Juho Lee JUHOLEE@KAIST.AC.KR
Graduate School of AI, KAIST, Daejeon, South Korea,
AITRICS, Seoul, South Korea

Hongseok Yang HONGSEOK.YANG@KAIST.AC.KR
School of Computing, KAIST, Daejeon, South Korea

1. Introduction

A standard approach for learning a deep generative model is to adopt the principle of maximum likelihood, where we attempt to find the model parameters that maximize the probability of the observed data, also called model evidence. Unfortunately, model evidence and its gradient are difficult to compute precisely for most deep generative models. Overcoming this difficulty has been the topic of active research in the ML and statistics communities, which resulted in techniques for optimizing computationally-tractable lower or upper bounds of model evidence (Ranganath et al., 2014; Kingma and Welling, 2014; Burda et al., 2016), as well as techniques for optimizing model evidence directly using sample-based estimates of its gradient (Bornschein and Bengio, 2015; Ou and Song, 2020).

Our goal is to make further progress in the second group of techniques, which are less explored than the ones in the first but are known to work well with discrete latent variables. In this paper, we report our preliminary results based on the recent work by Ou and Song (2020). Instead of following the traditional approach of approximating the gradient of model evidence using importance sampling (IS), Ou and Song (2020) proposed to use Markov Chain Monte Carlo (MCMC) for the gradient estimation. Their algorithm employs a non-stationary Markov chain that approximately generates samples for a moving target distribution, namely, the posterior of the model that changes throughout the gradient-based optimization. Since MCMC is known to perform better than IS for high-dimensional models, the algorithm is expected to work better for high-dimensional deep generative model than IS-based alternatives. In fact, the experiments in Ou and Song (2020) show improvement over the Reweighted-Wake-Sleep algorithm (RWS) (Bornschein and Bengio, 2015) (the best known among the IS-alternatives) for training deep generative models with discrete latent variables, such as Helmholtz Machine (Dayan et al., 1995).

We focus on one particular yet important problem of Ou and Song (2020)'s algorithm, which we call non-stationary kernel problem. Intuitively, the problem asks for an effective strategy to decide when we should reset a Markov chain used in the algorithm. The transition kernel of this chain is not fixed, because its target distribution keeps getting updated via the gradient ascent. The algorithm thus has to decide if the sample of the chain for the model before the gradient-ascent step should be used to generate a sample after the step, or if it should be discarded and the next sample should be generated from scratch. Although getting this decision right is crucial to get good results, Ou and Song (2020) do not provide an algorithmic solution for it; in their experiments, they used an ad-hoc strategy of always resetting the chain (i.e., generating the next sample from scratch) up until a fixed number of gradient-update steps, after which running the chain without interruption.

We present an automatic strategy for addressing this non-stationary kernel problem. Our strategy estimates the amount of change in a generative model and its accompanying approximate posterior and, in proportion to this estimate, decides whether or not to reset the chain and generate the next sample from scratch (using the approximate posterior). In doing so, it exploits the fact that if the model and the approximate posterior do not change much, the previous sample is likely to help produce a better sample. Our experimental analysis with Sigmoid Belief Networks (SBNs) (Saul et al., 1996) confirms this exploitation, and shows that our strategy gives results comparable to or slightly better than Ou and Song (2020), while avoiding their nontrivial requirement of manually choosing an appropriate threshold.

2. Setup and Background

We consider the problem of learning a model $p_\theta(\mathbf{x}, \mathbf{h})$ and an approximate posterior $q_\phi(\mathbf{h}|\mathbf{x})$ from observed data $\mathcal{D}$. Here $\mathbf{x}$ and $\mathbf{h}$ represent observed and hidden variables, respectively, and $\mathcal{D}$ is the multiset $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of the observed values of $\mathbf{x}$. Let p_d be the uniform distribution over $\mathcal{D}$. A popular approach for learning $p_\theta(\mathbf{x}, \mathbf{h})$ and $q_\phi(\mathbf{h}|\mathbf{x})$, which we focus on in the paper, is to optimize the following objectives: the maximization of $\mathbb{E}_{p_d(\mathbf{x})}[\log p_\theta(\mathbf{x})]$ over θ, and the minimization of the average Kullback-Leibler (KL) divergence $\mathbb{E}_{p_d(\mathbf{x})}[\mathrm{KL}(p_\theta(\mathbf{h}|\mathbf{x})||q_\phi(\mathbf{h}|\mathbf{x}))]$ over ϕ. The approach optimizes both objectives by gradient ascent or decent, based on the following characterizations of the gradients:

$$\begin{aligned}\nabla_\theta \mathbb{E}_{p_d(\mathbf{x})}[\log p_\theta(\mathbf{x})] &= \mathbb{E}_{p_d(\mathbf{x})}[\mathbb{E}_{p_\theta(\mathbf{h}|\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{h})]],\\ \nabla_\phi \mathbb{E}_{p_d(\mathbf{x})}[\mathrm{KL}(p_\theta(\mathbf{h}|\mathbf{x})||q_\phi(\mathbf{h}|\mathbf{x}))] &= -\mathbb{E}_{p_d(\mathbf{x})}[\mathbb{E}_{p_\theta(\mathbf{h}|\mathbf{x})}[\nabla_\phi \log q_\phi(\mathbf{h}|\mathbf{x})]].\end{aligned} \tag{1}$$

Note that the inner expectations are taken with respect to the true posterior $p_\theta(\mathbf{h}|\mathbf{x})$ from which it is difficult to generate samples, particularly because we have a limited computation budget for a single gradient update step. MCMC, for instance, is not a viable option here, because it takes a while to mix. Different strategies for overcoming this challenge have been developed and materialized into different learning algorithms.

The standard strategy is to estimate the expectations in (1) using the self-normalizing importance sampler. The representative example is RWS (Bornschein and Bengio, 2015), which uses, as the proposal of the sampler, the approximate posterior $q_\phi(\mathbf{h}|\mathbf{x})$ that is being updated throughout the run of the algorithm. The next formulas for each $\mathbf{x} \in \mathcal{D}$ describe the core of the algorithm:[1]

$$\begin{aligned}\mathbb{E}_{p_\theta(\mathbf{h}|\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{h})] &\simeq \sum_{k=1}^{K} \tilde{\omega}_k \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{h}^k),\\ \mathbb{E}_{p_\theta(\mathbf{h}|\mathbf{x})}[\nabla_\phi \log q_\phi(\mathbf{h}|\mathbf{x})] &\simeq \sum_{k=1}^{K} \tilde{\omega}_k \nabla_\phi \log q_\phi(\mathbf{h}^k|\mathbf{x}),\end{aligned} \tag{2}$$

1. Strictly speaking, we are describing the wake-wake version of the algorithm.

where $\mathbf{h}^1, \dots, \mathbf{h}^K \overset{\text{i.i.d.}}{\sim} q_\phi(\,\cdot\,|\mathbf{x})$ and $\tilde{\omega}_k = \frac{\omega_k}{\sum_{k'=1}^K \omega_{k'}}$ for $\omega_k = \frac{p_\theta(\mathbf{x},\mathbf{h}^k)}{q_\phi(\mathbf{h}^k|\mathbf{x})}$. Note that the gradient estimates in (2) are biased. For the instances where $p_\theta(\mathbf{h}|\mathbf{x})$ and $q_\phi(\mathbf{h}|\mathbf{x})$ tend not to match well, as in the case of high-dimensional models and datasets, the qualities of these estimates deteriorate due to their biases, the variances of weights ω_k, and also the algorithm's failure to learn reasonable model and approximate posterior.

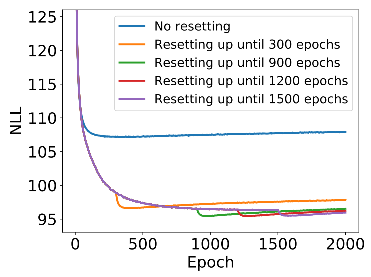
Algorithm 1: Joint Stochastic Approximation (JSA).

Inputs: Training set $\mathcal{D}$ of size N, mini-batch size B, total number of epochs T with T_0 to stop resetting, number of MCMC steps M, learning rates $\{\alpha_t\}_{t\geq 1}$ and $\{\beta_t\}_{t\geq 1}$.
Outputs: Parameters (θ, ϕ) for $p_\theta(\mathbf{x}, \mathbf{h})$ and $q_\phi(\mathbf{h}|\mathbf{x})$.

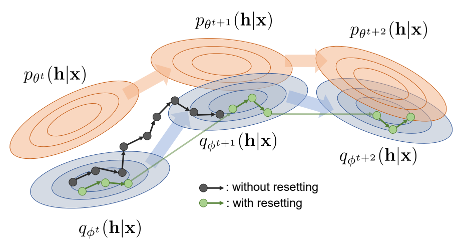
```
Initialize $\theta$, $\phi$, and $\{\mathbf{h}^0_\mathbf{x}\}_{\mathbf{x}\in\mathcal{D}}$;
for $t = 1$ to $T$ do
    for $s = 1$ to $N/B$ do
        Sample a mini-batch $\mathcal{B} = \{\mathbf{x}_i\}_{i=1}^B$ of size $B$ from the dataset $\mathcal{D}$;
        $\{\mathbf{h}^0_\mathbf{x}\}_{\mathbf{x}\in\mathcal{B}} \leftarrow$ Initialize($\mathcal{B}$, $\{\mathbf{h}^0_\mathbf{x}\}_{\mathbf{x}\in\mathcal{B}}$, $t$, $T_0$);
        for $\mathbf{x} \in \mathcal{B}$ do
            for $m = 1$ to $M$ do
                Sample $\tilde{\mathbf{h}} \sim q_\phi(\,\cdot\,|\mathbf{x})$ and $u \sim \text{Uniform}([0,1])$;
                if $\left(u < \frac{p_\theta(\mathbf{x},\tilde{\mathbf{h}})q_\phi(\mathbf{h}^{m-1}_\mathbf{x}|\mathbf{x})}{p_\theta(\mathbf{x},\mathbf{h}^{m-1}_\mathbf{x})q_\phi(\tilde{\mathbf{h}}|\mathbf{x})}\right)$ then $\left\{ \mathbf{h}^m_\mathbf{x} \leftarrow \tilde{\mathbf{h}}; \right\}$ else $\left\{ \mathbf{h}^m_\mathbf{x} \leftarrow \mathbf{h}^{m-1}_\mathbf{x}; \right\}$ end
            end
            $\mathbf{h}^0_\mathbf{x} \leftarrow \mathbf{h}^M_\mathbf{x}$;
        end
        $\theta \leftarrow \theta + \frac{\alpha_t}{MB} \sum_{\mathbf{x}\in\mathcal{B}} \sum_{m=1}^M \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{h}^m_\mathbf{x})$;
        $\phi \leftarrow \phi + \frac{\beta_t}{MB} \sum_{\mathbf{x}\in\mathcal{B}} \sum_{m=1}^M \nabla_\phi \log q_\phi(\mathbf{h}^m_\mathbf{x}|\mathbf{x})$;
    end
end
PROCEDURE Initialize($\mathcal{B}$, $\{\mathbf{h}^0_\mathbf{x}\}_{\mathbf{x}\in\mathcal{B}}$, $t$, $T_0$)
   if $(t \geq T_0)$ then { return $\{\mathbf{h}^0_\mathbf{x}\}_{\mathbf{x}\in\mathcal{B}}$; } end
   for $(\mathbf{x} \in \mathcal{B})$ do { Sample $\tilde{\mathbf{h}}_\mathbf{x} \sim q_\phi(\,\cdot\,|\mathbf{x})$; } end
   return $\{\tilde{\mathbf{h}}_\mathbf{x}\}_{\mathbf{x}\in\mathcal{B}}$;
```

Joint Stochastic Approximation (JSA) (Ou and Song, 2020) is a recent alternative to the importance-sampling-based approach, which uses MCMC to generate approximate posterior samples. It is an instance of the Stochastic Approximation Procedure (Robbins and Monro, 1951), which has been shown to work well for training large-scale Markov Random Fields (Salakhutdinov, 2008, 2009; Salakhutdinov and Hinton, 2009; Tieleman, 2008) and deep energy-based models (Xie et al., 2016; Nijkamp et al., 2019).

JSA with a particular Metropolis-Hastings kernel is shown in Algorithm 1. To estimate the inner expectations in (1) for $\mathbf{x} \in \mathcal{D}$, JSA uses a Markov chain with a non-stationary transition kernel $K_{\theta,\phi,\mathbf{x}}(\mathbf{h}'|\mathbf{h})$, which depends on the parameters θ and ϕ that change throughout the run of the algorithm. More precisely, JSA stores a sample $\mathbf{h}^0_\mathbf{x}$ for every $\mathbf{x} \in \mathcal{D}$ and, at each epoch, it generates new samples $\mathbf{h}^1_\mathbf{x}, \dots, \mathbf{h}^M_\mathbf{x}$ for each $\mathbf{x}$ in a mini-batch of $\mathcal{D}$ (where M is typically ≤ 4) by running the Markov chain of the kernel $K_{\theta,\phi,\mathbf{x}}$ from the stored $\mathbf{h}^0_\mathbf{x}$. The generated samples are then used to estimate gradients in (1), and the last sample $\mathbf{h}^M_\mathbf{x}$ serves as the initial state of the chain at the next epoch, despite the fact that the kernel $K_{\theta,\phi,\mathbf{x}}$ at the next epoch will be different from the one used to



(a) Test NLL trace plots


$p_{\theta^t}(\mathbf{h}|\mathbf{x})$
$p_{\theta^{t+1}}(\mathbf{h}|\mathbf{x})$
$p_{\theta^{t+2}}(\mathbf{h}|\mathbf{x})$
$q_{\phi^t}(\mathbf{h}|\mathbf{x})$
$q_{\phi^{t+1}}(\mathbf{h}|\mathbf{x})$
$q_{\phi^{t+2}}(\mathbf{h}|\mathbf{x})$
: without resetting
: with resetting

(b) Non-stationary kernel problem

Figure 1: (a) Test NLLs for different epochs for stopping reset. For instance, "*resetting up until 300 epochs*" means that if the current epoch t is less than 300, the algorithm resets the chain at this epoch and draws its initial state $\mathbf{h}_{\mathbf{x}}^0$ from $q_\phi(\mathbf{h}|\mathbf{x})$, and if not, it uses the sample $\mathbf{h}_{\mathbf{x}}^M$ generated at some previous epoch $t' < t$. In both cases, the chain is run for M steps at the epoch t. (b) A diagram that shows how resetting works in the non-stationary kernel problem.

generate $\mathbf{h}_{\mathbf{x}}^M$. As the algorithm optimizes θ and ϕ, the kernel $K_{\theta,\phi,\mathbf{x}}$ changes in a way to ensure that the posterior $p_\theta(\mathbf{h}|\mathbf{x})$ at the current θ is an invariant distribution of the kernel $K_{\theta,\phi,\mathbf{x}}$. The maintenance of this relationship between the kernel and the posterior guarantees that θ and ϕ converge to local optimums of the marginal-log-likelihood and KL objectives.[2]

3. Non-stationary Kernel Problem and Our Solution

Problem. Although the constant change of the kernel $K_{\theta,\phi,\mathbf{x}}(\mathbf{h}'|\mathbf{h})$ and the posterior $p_\theta(\mathbf{h}|\mathbf{x})$ throughout the run of the JSA algorithm does not break theoretical convergence guarantee, in practice, it significantly affects the performance of the algorithm. Without a proper counter-measure for this change, the algorithm does not perform well. Figure 1(a) shows the test negative log likelihood (NLL) of running JSA for the binarized MNIST dataset with or without such counter-measures. The blue line labeled "*no resetting*" is the result without any counter-measure, and shows the poor performance in this setup.

We refer to the problem of finding an effective counter-measure as non-stationary kernel problem. A solution for this problem should be able to interact well with other components of the JSA algorithm, in particular, the mini-batch method, which widens the gap between the target posteriors for the previous $\mathbf{h}_{\mathbf{x}}^0$ and the current $\mathbf{h}_{\mathbf{x}}^1$ for each observation $\mathbf{x}$ and effectively increases the amount of change of the kernel.

Ou and Song (2020) tackles this non-stationary kernel problem by resetting the Markov chain (determined by the kernel $K_{\theta,\phi,\mathbf{x}}$). For all epochs before a manually-chosen epoch T_0, they let JSA abandon the previous state $\mathbf{h}_{\mathbf{x}}^0$ of the Markov chain and reset the chain with a sample $\tilde{\mathbf{h}}_{\mathbf{x}}$ newly drawn from the current approximate posterior $q_\phi(\tilde{\mathbf{h}}_{\mathbf{x}}|\mathbf{x})$. The algorithm then executes the transition kernel $K_{\theta,\phi,\mathbf{x}}(\mathbf{h}_{\mathbf{x}}^1|\tilde{\mathbf{h}}_{\mathbf{x}})$ on $\tilde{\mathbf{h}}_{\mathbf{x}}$, and generates the next sample $\mathbf{h}^1$. From the epoch T_0 onward, they stop resetting altogether and generate samples by running the chain uninterrupted across epochs. See Algorithm 1 for the detail. Figure 1(b) shows a conceptual illustration highlighting the benefit of resetting.

2. For this convergence result to hold, the learning rates α_t and β_t for θ and ϕ have to satisfy the condition $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.

Algorithm 2: Adative resetting strategy for JSA.

Inputs: A mini-batch $\mathcal{B}$ of size B and samples $\{\mathbf{h}_{\mathbf{x}}^0\}_{\mathbf{x}\in\mathcal{B}}$ from the previous epoch.
Outputs: Initial set of samples for the current Markov chain.

```
PROCEDURE AdaptiveInitialize($\mathcal{B}$, $\{\mathbf{h}_{\mathbf{x}}^0\}_{\mathbf{x}\in\mathcal{B}}$)
    for ($\mathbf{x} \in \mathcal{B}$) do { Sample $\tilde{\mathbf{h}}_{\mathbf{x}} \sim q_\phi(\cdot\,|\mathbf{x})$; } end
    $\widehat{R} \leftarrow \frac{1}{B}\sum_{\mathbf{x}\in\mathcal{B}} \min\left\{0, \log\frac{p_\theta(\mathbf{x},\tilde{\mathbf{h}}_{\mathbf{x}})q_\phi(\mathbf{h}_{\mathbf{x}}^0|\mathbf{x})}{p_\theta(\mathbf{x},\mathbf{h}_{\mathbf{x}}^0)q_\phi(\tilde{\mathbf{h}}_{\mathbf{x}}|\mathbf{x})}\right\}$;    Sample $d \sim \text{Bernoulli}(\tanh(-\gamma\cdot\widehat{R}))$;
    if ($d = 1$) then { return $\{\tilde{\mathbf{h}}_{\mathbf{x}}\}_{\mathbf{x}\in\mathcal{B}}$; } else { return $\{\mathbf{h}_{\mathbf{x}}^0\}_{\mathbf{x}\in\mathcal{B}}$; } end
```

However, Ou and Song (2020)'s approach is incomplete. Its effectiveness crucially depends on the right choice of T_0, but the approach is silent about how to make such a choice. Also, the approach is inflexible in the use of resetting; it never attempts to reset the chain after the threshold T_0, while doing so may bring further improvement in accuracy.

Our goal is to overcome these shortcomings. Next we present our strategy that automatically decides when to reset the non-stationary Markov chain of the JSA algorithm.

Adaptive Strategy for Resetting the Markov Chain of JSA. We replace the statement $\{\mathbf{h}_{\mathbf{x}}^0\}_{\mathbf{x}\in\mathcal{B}} \leftarrow$ Initialize(...) of Algorithm 1 by the following procedure call: $\{\mathbf{h}_{\mathbf{x}}^0\}_{\mathbf{x}\in\mathcal{B}} \leftarrow$ AdaptiveInitialize($\mathcal{B}$, $\{\mathbf{h}_{\mathbf{x}}^0\}_{\mathbf{x}\in\mathcal{B}}$). The procedure AdaptiveInitialize is defined in Algorithm 2, and it implements our adaptive resetting strategy.

Our strategy works as follows. At each epoch, we monitor the acceptance probability values and reset the chain when it exhibits low acceptance ratio. Recall that p_d is the uniform distribution over the training set $\mathcal{D}$. Let t be the current epoch number. For each $\mathbf{x} \in \mathcal{D}$, let $\mathbf{h}_{\mathbf{x}}^0$ be the value of the latent variable generated at some previous epoch but currently stored for $\mathbf{x}$. Then at the current epoch t, we compute the following quantity,

$$\mathbb{E}_{p_d(\mathbf{x})}\left[\mathbb{E}_{q_\phi(\tilde{\mathbf{h}}_{\mathbf{x}}|\mathbf{x})}\left[\log\min\left\{1, \frac{p_\theta(\mathbf{x},\tilde{\mathbf{h}}_{\mathbf{x}})q_\phi(\mathbf{h}_{\mathbf{x}}^0|\mathbf{x})}{p_\theta(\mathbf{x},\mathbf{h}_{\mathbf{x}}^0)q_\phi(\tilde{\mathbf{h}}_{\mathbf{x}}|\mathbf{x})}\right\}\right]\right] \approx \frac{1}{B}\sum_{\mathbf{x}\in\mathcal{B}}\min\left\{0, \log\frac{p_\theta(\mathbf{x},\tilde{\mathbf{h}}_{\mathbf{x}})q_\phi(\mathbf{h}_{\mathbf{x}}^0|\mathbf{x})}{p_\theta(\mathbf{x},\mathbf{h}_{\mathbf{x}}^0)q_\phi(\tilde{\mathbf{h}}_{\mathbf{x}}|\mathbf{x})}\right\} =: \widehat{R}, \quad (3)$$

where $\mathcal{B}$ is a mini-batch of size B uniformly sampled from $\mathcal{D}$, and θ and ϕ are the parameter values at the current epoch t. This ratio inside the log is the one used to compute the acceptance probability in the Metropolis-Hastings kernel of the JSA algorithm; see Algorithm 1. Based on the estimate $\widehat{R}$, we decide whether or not to reset the chain via the Bernoulli trial, $d \sim \text{Bernoulli}(\tanh(-\gamma\cdot\widehat{R}))$, where γ is a hyperparameter to be specified.

Note that d being close to one, meaning that the chain is more likely to reset, happens when the estimate $\widehat{R}$ is small. This itself usually happens when the acceptance probability of the Metropolis-Hastings kernel of JSA tends to be small, indicating that the Markov chain is not mixing well. This slow mixing is particularly problematic in JSA because JSA does not run a full Markov chain at each epoch but only draws a few number of samples $M \in [2, 4]$. As a result, for a Markov chain simulated by JSA, when showing low acceptance ratio, the effective sample size drawn at each epoch is close to one, and so the JSA update cannot effectively update the parameters. For such cases, our strategy encourages the resetting of the Markov chain by increasing the chance of d being 1. See Algorithm 2 for detail.

Method	NLL ($M = 2$)	NLL ($M = 4$)
RWS	96.5±0.2*	95.2±0.2
VIMCO	95.8±0.1*	95.2±0.1
JSA	95.3±0.1*	94.9±0.1
Ours	**95.2±0.1**	**94.6±0.2**

Table 1: Test NLLs of different methods for learning a two-layer SBN model and its approximate posterior for the binarized MNIST. Here * indicates the results taken from Ou and Song (2020). JSA resets the chain up until 600 epoch. The reported mean and standard deviation are computed over five independent runs.

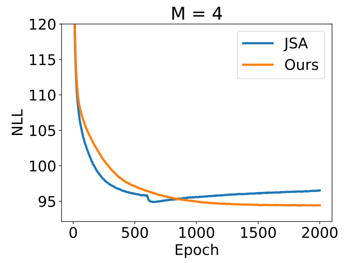


Figure 2: Test NLLs of JSA with the original resetting approach and our approach during the learning of two-layer SBN and approximate posterior over the binarized MNIST dataset. We set $M = 4$, i.e., the Markov chain is run for four steps for each $\mathbf{x}$ in the mini-batch at each epoch.

4. Experiments

We report the preliminary evaluation of the JSA with our resetting strategy against the baselines. We used the binarized MNIST dataset (Salakhutdinov and Murray, 2008), with 50K-10K-10K splits for training, validation, and test, respectively. Following the setting in Yin and Zhou (2019); Ou and Song (2020), we compared ours to RWS (Bornschein and Bengio, 2015), VIMCO (Mnih and Rezende, 2016), and JSA (Ou and Song, 2020) on learning a discrete generative model $p_\theta(\mathbf{x}, \mathbf{h})$ with an approximate posterior $q_\phi(\mathbf{h}|\mathbf{x})$ where both $p_\theta(\mathbf{x}, \mathbf{h})$ and $q_\phi(\mathbf{h}|\mathbf{x})$ are constructed as SBNs with two hidden layers. For training, we used the ADAM (Kingma and Ba, 2015) with a learning rate 0.0003 and a batch size 50. We tested two settings with different numbers of Monte-Carlo samples for the gradient approximation ($M = 2$ or 4). The hyperparameter γ in Algorithm 2 was fixed to 0.05.

We measured the test negative log-likelihood (NLL) of each model for every 5 epochs and report the value when the validation NLL reached a minimum within $2,000$ epochs. To estimate NLLs, we used importance sampling with $1,000$ samples drawn from $q_\phi(\mathbf{h}|\mathbf{x})$, as in Bornschein and Bengio (2015). Table 1 shows the test NLLs of our approach and the baselines. The learning curves of JSA with our resetting strategy and the original JSA are shown in Figure 2. These curves and test NLLs show that our strategy leads to the accuracy comparable with or slightly better than that of the original JSA algorithm, while, more importantly, letting us avoid the manual selection of an appropriate epoch threshold T_0 of the algorithm.

5. Conclusion

In this paper, we proposed a way to improve JSA, an MCMC-based learning algorithm for deep generative models. In particular, we highlighted the non-stationary kernel problem in JSA and proposed a solution to resolve it where Markov chains are adaptively reset according to the estimates of average acceptance probabilities. Our experimental results demonstrate the effectiveness of our approach, but we think that there is still plenty of room for improvement. One interesting future work is to design a better proposal distribution instead of the one based on the approximate posterior $q_\phi(\mathbf{h}|\mathbf{x})$.

Acknowledgments

JL acknowledges the support by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)). Special thanks to Gloria McGahan for her thoughtful efforts in detailed language editing and proofreading.

References

Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Andriy Mnih and Danilo J. Rezende. Variational inference for monte carlo objectives. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2188–2196, 2016.

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems*, pages 5232–5242, 2019.

Zhijian Ou and Yunfu Song. Joint stochastic approximation and its application to learning discrete latent variable models. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI 2020, virtual online, August 3-6, 2020*, page 377, 2020.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Ruslan Salakhutdinov. Learning and evaluating boltzmann machines. *Utml Tr*, 2:21, 2008.

Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455, 2009.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307, pages 872–879, 2008.

Russ R Salakhutdinov. Learning in markov random fields using tempered transitions. In *Advances in neural information processing systems*, pages 1598–1606, 2009.

Lawrence K Saul, Tommi Jaakkola, and Michael I Jordan. Mean field theory for sigmoid belief networks. *Journal of artificial intelligence research*, 4:61–76, 1996.

Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644, 2016.

Mingzhang Yin and Mingyuan Zhou. ARM: augment-reinforce-merge gradient for stochastic binary networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.