ARM: Augment-REINFORCE-merge gradient for discrete latent variable models

Mingzhang Yin
Department of Statistics and Data Sciences
The University of Texas at Austin
Austin, TX 78712
mzyin@utexas.edu

Mingyuan Zhou
McCombs School of Business
The University of Texas at Austin
Austin, TX 78712
mingyuan.zhou@mccombs.utexas.edu

Abstract

To propagate the gradients through discrete stochastic layers, we encode the true gradients into zeros combined with spikes, which are distributed over a random subset of iterations and amenable to backpropagation. To modulate the frequencies, amplitudes, and signs of the spikes to capture the temporal evolution of the true gradients, we propose the augment-REINFORCE-merge (ARM) estimator, which combines data augmentation, the score-function estimator, and variance reduction for Monte Carlo integration using common random numbers. The ARM estimator provides low-variance and unbiased gradient estimates for the parameters of discrete distributions, leading to state-of-the-art performance in both auto-encoding variational Bayes and maximum likelihood inference, for discrete latent variable models with one or multiple discrete stochastic layers.

1 Introduction

Given a function $f(\boldsymbol{z})$ of a random variable $\boldsymbol{z}$, which follows a distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ parameterized by $\boldsymbol{\phi}$, there has been significant recent interest in estimating $\boldsymbol{\phi}$ to maximize (or minimize) the expectation of $f(\boldsymbol{z})$ with respect to $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})$, expressed as

$$\mathcal{E}(\boldsymbol{\phi}) = \int f(\boldsymbol{z}) q_{\boldsymbol{\phi}}(\boldsymbol{z}) d\boldsymbol{z} = \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[f(\boldsymbol{z})]. \tag{1}$$

In particular, maximizing the marginal likelihood of a hierarchal Bayesian model [1] and maximizing the evidence lower bound (ELBO) for variational inference [2, 3], two fundamental problems in statistical inference, both boil down to maximizing an expectation as in (1). To maximize (1), if $\nabla_{\boldsymbol{z}} f(\boldsymbol{z})$ is tractable to compute and $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})$ can be generated via reparameterization as $\boldsymbol{z} = \mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}),\ \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ are random noises and $\mathcal{T}_{\boldsymbol{\phi}}(\cdot)$ denotes a deterministic transform parameterized by $\boldsymbol{\phi}$, then one may apply the reparameterization trick [4, 5] to compute the gradient as

$$\nabla_{\boldsymbol{\phi}} \mathcal{E}(\boldsymbol{\phi}) = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})}[f(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] = \mathbb{E}_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} f(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))]. \tag{2}$$

Unfortunately, the reparameterization trick is not directly applicable to discrete random variables, which are widely used to construct discrete latent variable models such as the sigmoid belief net [6, 7].

To maximize (1) for discrete $\boldsymbol{z}$, if $\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\nabla_{\boldsymbol{\phi}} f(\boldsymbol{z})] = 0$ almost surely (a.s.), using the score function $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}) = \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{z}) / q_{\boldsymbol{\phi}}(\boldsymbol{z})$, one may compute $\nabla_{\boldsymbol{\phi}} \mathcal{E}(\boldsymbol{\phi})$ via REINFORCE [8] as

$$\nabla_{\boldsymbol{\phi}} \mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[f(\boldsymbol{z}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z})] \approx \frac{1}{K} \sum_{k=1}^{K} f(\boldsymbol{z}^{(k)}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(k)}), \tag{3}$$

where $\boldsymbol{z}^{(k)} \overset{iid}{\sim} q_{\boldsymbol{\phi}}(\boldsymbol{z})$ are independent, and identically distributed (iid). This estimator is also known as (a.k.a.) the score-function [9] or likelihood-ratio estimator [10]. While it is very general in that

Preprint. Work in progress.

it only requires drawing iid random samples from $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ and computing $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(k)})$, the high variance of Monte Carlo integration often limits its use in practice.

To address the high-variance issue, one may introduce appropriate control variates (a.k.a. baselines) to reduce the variance of REINFORCE [11–18]. Alternatively, one may first relax the discrete random variables with continuous ones and then apply the reparameterization trick to estimate the gradients, which reduces the variance of Monte Carlo integration at the expense of introducing bias [19, 20]. Combining both REINFORCE and the continuous relaxation of discrete random variables, several recently proposed algorithms are able to produce low-variance and unbiased gradient estimates, but need to introduce control variates that are updated for each mini-batch, increasing both the computational complexity and risk of overfitting the training data [21, 22].

Distinct from all previously proposed stochastic gradient estimators for discrete latent variables, which in general try to estimate the true gradients as accurate as possible at each iteration, our strategy of variance control, which allows using a single random sample for Monte Carlo integration, is to encourage the estimated gradient of a discrete distribution parameter to be exactly zero with a high probability at each iteration, while allowing it to be a random spike from time to time. The spike frequency, amplitude, and sign are modulated by the estimator to track the true gradient that often smoothly evolves over iterations. To this end, we propose the augment-REINFORCE-merge (ARM) estimator, a novel low-variance and unbiased stochastic gradient estimator that manipulates the gradients of discrete distribution parameters in an augmented space, updating a model parameter with either zero or a spike at each gradient ascent/descent step. More specifically, we rewrite the expectation with respect to a discrete latent variable to that with respect to augmented exponential random variables, express the gradient as an expectation via REINFORCE, re-express that expectation as a one with respect to standard exponential distributions, and then let different expectations to share common random numbers to effectively control the variance of Monte Carlo integration.

Our experimental results on both auto-encoding variational Bayes and maximum likelihood inference for discrete latent variable models, with one or multiple discrete stochastic layers, show that the ARM estimator has low computational complexity and provides state-of-the-art out-of-sample prediction performance, suggesting the effectiveness of using ARM spikes for gradient backpropagation through discrete stochastic layers.

2 ARM: Augment-REINFORCE-merge estimator

Let us denote $z \sim \text{Discrete}(\sigma(\boldsymbol{\phi}))$ as a categorical random variable such that $P(z = i \,|\, \boldsymbol{\phi}) = \sigma(\boldsymbol{\phi})_i = e^{\phi_i} / \sum_{i=1}^M e^{\phi_i}$, where $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_M)'$ and $\sigma(\boldsymbol{\phi}) = (e^{\phi_1}, \ldots, e^{\phi_M})' / \sum_{i=1}^M e^{\phi_i}$ is the softmax function. For an expectation of the function $f(\boldsymbol{z})$ with respect to $z \sim \text{Discrete}(\sigma(\boldsymbol{\phi}))$, expressed as $\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{z\sim\text{Discrete}(\sigma(\boldsymbol{\phi}))}[f(z)]$, we need to evaluate its gradient with respect to $\boldsymbol{\phi}$. Without loss of generality, we assume $\mathbb{E}_{z\sim\text{Discrete}(\sigma(\boldsymbol{\phi}))}[\nabla_{\boldsymbol{\phi}} f(z)] = 0$ a.s. We first present the augment-REINFORCE-merge (ARM) estimator in the following Theorem. We denote $\mathbf{1}_M$ as a vector of M ones, and $(m \leftrightharpoons M)$ as a vector of indices constructed by swapping the m-th and M-th elements of vector $(1, \ldots, M)$, which means $(m \leftrightharpoons M)_M = m$, $(m \leftrightharpoons M)_m = M$, and $(m \leftrightharpoons M)_i = i$ for $i \notin \{m, M\}$. The ARM estimator can be further simplified for binary random variables, as summarized in Corollary 2. The detailed derivations are provided in Sections 2.1–2.4.

Theorem 1 (ARM for categorical)**.** *For a categorical random variable $\boldsymbol{z}$, the gradient of $\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{z\sim\text{Discrete}(\sigma(\boldsymbol{\phi}))}[f(z)] = \mathbb{E}_{z\sim\text{Discrete}(\sigma([\tilde{\boldsymbol{\phi}},0]'))}[f(z)]$ with respect to $\tilde{\boldsymbol{\phi}} = (\tilde{\phi}_1, \ldots, \tilde{\phi}_{M-1})'$, where $\tilde{\phi}_m = \phi_m - \phi_M$, can be expressed as*

$$\begin{aligned} &\nabla_{\tilde{\phi}_m} \mathcal{E}(\boldsymbol{\phi}) = \nabla_{\tilde{\phi}_m} \mathcal{E}([\tilde{\boldsymbol{\phi}}, 0]') = \mathbb{E}_{\boldsymbol{\pi}\sim\text{Dirichlet}(\mathbf{1}_M)}[f_\Delta(\boldsymbol{\pi}, \boldsymbol{\phi}, m)(1 - M\pi_M)], \quad \textit{where} \\ &f_\Delta(\boldsymbol{\pi}, \boldsymbol{\phi}, m) = f\Big(\underset{i\in\{1,\ldots,M\}}{\arg\min}\, \pi_{(m\leftrightharpoons M)_i} e^{-\phi_i}\Big) - \frac{1}{M}\sum\nolimits_{j=1}^M f\Big(\underset{i\in\{1,\ldots,M\}}{\arg\min}\, \pi_{(j\leftrightharpoons M)_i} e^{-\phi_i}\Big). \end{aligned} \tag{4}$$

Corollary 2 (ARM for binary)**.** *For a binary random variable, the gradient of $\mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[f(z)]$ with respect to ϕ, the logit of the Bernoulli probability parameter, can be expressed as*

$$\begin{aligned} &\nabla_\phi \mathcal{E}(\phi) = \mathbb{E}_{u\sim\text{Uniform}(0,1)}[f_\Delta(u, \phi)(u - 1/2)], \;\; \textit{where} \\ &f_\Delta(u, \phi) = f\big(\mathbf{1}[u > \sigma(-\phi)]\big) - f\big(\mathbf{1}[u < \sigma(\phi)]\big). \end{aligned} \tag{5}$$

We refer to both $f_\Delta(\boldsymbol{\pi}, \boldsymbol{\phi}, m)(1 - M\pi_M)$ in (4) and $f_\Delta(u, \phi)$ in (5) as spike gradients, which are the estimates of the corresponding true gradients using a single Monte Carlo sample. For a categorical random variable, when there is a ϕ_k whose value is dominant, which means $\phi_k \gg \phi_i$ for $i \neq k$, it is highly likely that $\arg\min_i \pi_{(j=M)_i} e^{-\phi_i} = k$ for all $j \in \{1, \ldots, M\}$, and consequently, $f_\Delta(\boldsymbol{\pi}, \boldsymbol{\phi}, m) = 0$ and the spike gradient is zero. For a binary random variable, when $\sigma(\phi)$ is close to either one or zero, it is likely that $f_\Delta(u, \phi) = 0$ and hence the spike gradient is zero.

2.1 Augmentation of a categorical random variable

Let us denote $z \sim \text{Exp}(\lambda)$ as the exponential distribution, with probability density function $f_Z(z) = \lambda e^{-\lambda z}$, where $\lambda > 0$ and $z > 0$. Its mean and variance are $\mathbb{E}[z] = \lambda^{-1}$ and $\text{var}[z] = \lambda^{-2}$, respectively. It is well known that, e.g. in [23], if $z_i \sim \text{Exp}(\lambda_i)$ are independent exponential random variables for $i = 1, \ldots, M$, then the probability that z_i is the smallest can be expressed as

$$P\big(i = \arg\min_j z_j\big) = P\left(z_i < z_j,\ \forall\, j \neq i\right) = \lambda_i \Big/ \sum\nolimits_{i=1}^{M} \lambda_i \ . \tag{6}$$

Note this property is closely related to the Gumbel distribution (a.k.a. Type-I extreme-value distribution) based latent-utility-maximization representation of multinomial logistic regression [24, 25], as well as the Gumbel-softmax trick [19, 20]. This is because the exponential random variable $z \sim \text{Exp}(\lambda)$ can be reparameterized as $z = \epsilon/\lambda,\ \epsilon \sim \text{Exp}(1)$, where $\epsilon \sim \text{Exp}(1)$ can be equivalently generated as $\epsilon = -\log u,\ u \sim \text{Uniform}(0, 1)$, and hence we have

$$\arg\min_i z_i \overset{d}{=} \arg\min_i\{-\log u_i/\lambda_i\} = \arg\max_i\{\log\lambda_i - \log(-\log u_i)\},$$

where $z_i \sim \text{Exp}(\lambda_i)$, the symbol "$\overset{d}{=}$" denotes "equal in distribution," and $u_i \overset{iid}{\sim} \text{Uniform}(0, 1)$; note that if $u \sim \text{Uniform}(0, 1)$, then $-\log(-\log u)$ follows the Gumbel distribution [25].

From (6), it becomes clear that $z \sim \text{Discrete}(\sigma(\boldsymbol{\phi}))$ can be augmented as

$$z = \arg\min_{i\in\{1,\ldots,M\}} e_i \text{ , where } e_i \sim \text{Exp}(e^{\phi_i}). \tag{7}$$

Consequently, the expectation with respect to the categorical random variable of M categories can be rewritten as one with respect to M augmented exponential random variables as

$$\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{z\sim\text{Discrete}(\sigma(\boldsymbol{\phi}))}[f(z)] = \mathbb{E}_{e_1\sim\text{Exp}(e^{\phi_1}),\ldots,e_M\sim\text{Exp}(e^{\phi_M})}[f(\arg\min_i e_i)]. \tag{8}$$

Since the exponential random variable $x \sim \text{Exp}(e^{\phi})$ can be reparameterized as $x = \epsilon e^{-\phi},\ \epsilon \sim \text{Exp}(1)$, by the law of the unconscious statistician (LOTUS) [26], we also have

$$\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\epsilon_1,\ldots,\epsilon_M \overset{iid}{\sim} \text{Exp}(1)}[f(\arg\min_i \epsilon_i e^{-\phi_i})]. \tag{9}$$

Note as the function $\arg\min_i \epsilon_i e^{-\phi_i}$ is not differentiable, the reparameterization trick shown in (2) is not applicable to computing the gradient of $\mathcal{E}(\boldsymbol{\phi})$ via the reparameterized representation in (9).

2.2 REINFORCE estimator in the augmented space

Using REINFORCE on (8), we have $\nabla_{\boldsymbol{\phi}}\mathcal{E}(\boldsymbol{\phi}) = [\nabla_{\phi_1}\mathcal{E}(\boldsymbol{\phi}), \ldots, \nabla_{\phi_M}\mathcal{E}(\boldsymbol{\phi})]'$, where

$$\begin{aligned}\nabla_{\phi_m}\mathcal{E}(\boldsymbol{\phi}) &= \mathbb{E}_{e_1\sim\text{Exp}(e^{\phi_1}),\ldots,e_M\sim\text{Exp}(e^{\phi_M})}[f(\arg\min_i e_i)\nabla_{\phi_m}\log\text{Exp}(e_m; e^{\phi_m})] \\ &= \mathbb{E}_{e_1\sim\text{Exp}(e^{\phi_1}),\ldots,e_M\sim\text{Exp}(e^{\phi_M})}[f(\arg\min_i e_i)(1 - e_m e^{\phi_m})].\end{aligned} \tag{10}$$

Using LOTUS again, we can re-express the gradient in (10) as

$$\nabla_{\phi_m}\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\epsilon_1,\ldots,\epsilon_M \overset{iid}{\sim} \text{Exp}(1)}[f(\arg\min_i \epsilon_i e^{-\phi_i})(1 - \epsilon_m)]. \tag{11}$$

As $\sigma(\boldsymbol{\phi} - \phi_M\mathbf{1}_M) = \sigma(\boldsymbol{\phi})$ almost surely (a.s.), one may choose category M for reference and only update $\tilde{\phi}_m = \phi_m - \phi_M$ for $m \leq M - 1$. Denoting $\tilde{\boldsymbol{\phi}} = (\tilde{\phi}_1, \ldots, \tilde{\phi}_{M-1})' = \mathbf{A}\boldsymbol{\phi}$, where $\mathbf{A} = [\text{diag}(\mathbf{1}_{M-1}), -\mathbf{1}_{M-1}]$, we have $\mathcal{E}([\tilde{\boldsymbol{\phi}}, 0]') = \mathcal{E}(\boldsymbol{\phi})$ and hence

$$\nabla_{\boldsymbol{\phi}}\mathcal{E}(\boldsymbol{\phi})' = \nabla_{\tilde{\boldsymbol{\phi}}}\mathcal{E}([\tilde{\boldsymbol{\phi}}, 0]')'\frac{\partial\tilde{\boldsymbol{\phi}}}{\partial\boldsymbol{\phi}} = \nabla_{\tilde{\boldsymbol{\phi}}}\mathcal{E}([\tilde{\boldsymbol{\phi}}, 0]')'\mathbf{A}. \tag{12}$$

Therefore, we have a solution as $\nabla_{\tilde{\boldsymbol{\phi}}}\mathcal{E}([\tilde{\boldsymbol{\phi}},0]')' = \nabla_{\boldsymbol{\phi}}\mathcal{E}(\boldsymbol{\phi})'\mathbf{A}'(\mathbf{A}\mathbf{A}')^{-1}$, where $\mathbf{A}'(\mathbf{A}\mathbf{A}')^{-1} = [\text{diag}(\mathbf{1}_{M-1}), \mathbf{0}_{M-1}]' - \frac{1}{M}\mathbf{1}_{M\times(M-1)}$, and hence

$$\nabla_{\tilde{\phi}_m}\mathcal{E}([\tilde{\boldsymbol{\phi}},0]') = \frac{1}{M}\sum\nolimits_{j=1}^{M}(\nabla_{\phi_m}\mathcal{E}(\boldsymbol{\phi}) - \nabla_{\phi_j}\mathcal{E}(\boldsymbol{\phi})) = \nabla_{\phi_m}\mathcal{E}(\boldsymbol{\phi}) - \frac{1}{M}\sum\nolimits_{j=1}^{M}\nabla_{\phi_j}\mathcal{E}(\boldsymbol{\phi}). \quad (13)$$

Below we show how to merge $\nabla_{\phi_m}\mathcal{E}(\boldsymbol{\phi})$ and $-\nabla_{\phi_j}\mathcal{E}(\boldsymbol{\phi})$, by sharing common random numbers [27] to well control the variance of Monte-Carlo integration.

2.3 Merge of REINFORCE gradients

A key observation of the paper is that the expectation in (11) can be equivalently expressed as

$$\nabla_{\phi_m}\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\epsilon_1,\ldots,\epsilon_M \overset{iid}{\sim}\text{Exp}(1)}[f(\arg\min_i \epsilon_{(m\leftrightharpoons M)_i}e^{-\phi_i})(1-\epsilon_M)]. \quad (14)$$

Using (14) on (13), we introduce the augment-REINFORCE-merge (ARM) estimator as

$$\nabla_{\tilde{\phi}_m}\mathcal{E}([\tilde{\boldsymbol{\phi}},0]') = \mathbb{E}_{\epsilon_1,\ldots,\epsilon_M \overset{iid}{\sim}\text{Exp}(1)}[f_\Delta(\boldsymbol{\epsilon},\boldsymbol{\phi},m)(1-\epsilon_M)], \quad (15)$$

in which we share common random numbers to compute different expectations, with

$$f_\Delta(\boldsymbol{\epsilon},\boldsymbol{\phi},m) = f(\arg\min_i \epsilon_{(m\leftrightharpoons M)_i}e^{-\phi_i}) - \frac{1}{M}\sum\nolimits_{j=1}^{M} f(\arg\min_i \epsilon_{(j\leftrightharpoons M)_i}e^{-\phi_i}). \quad (16)$$

Distinct from a usual REINFORCE estimator, an attractive property of the ARM estimator in (15) is that inside the expectation, $1-\epsilon_M$, which is equal in distribution to the score function $\nabla_{\phi_M}\log\prod_{m=1}^{M}\text{Exp}(e_m;e^{\phi_m})$, is multiplied by a function with zero expectation as

$$\mathbb{E}_{\epsilon_1,\ldots,\epsilon_M \overset{iid}{\sim}\text{Exp}(1)}[f_\Delta(\boldsymbol{\epsilon},\boldsymbol{\phi},m)] = 0.$$

Thus there is no more need for the ARM estimator to find a control variate for variance reduction. Moreover, if $f_\Delta(\boldsymbol{\epsilon},\boldsymbol{\phi},m) = 0$ happens with a high probability, then it is likely that the gradients estimated by a single Monte Carlo sample will be zeros during most iterations, while becoming spikes from time to time. In other words, rather than making an estimated gradient be as close as possible to the true gradient at each iteration, the ARM estimator makes it only moves as frequently as necessary, using its time-averaged values to approximate the true time-evolving gradients.

Furthermore, noting that $\text{Exp}(1) \overset{d}{=} \text{Gamma}(1,1)$, letting $\epsilon_1,\ldots,\epsilon_M \overset{iid}{\sim}\text{Exp}(1)$ is the same (e.g., as proved in Lemma IV.3 of [28]) in distribution as letting

$$\epsilon_i = \pi_i\epsilon, \quad \text{for } i = 1,\ldots,M, \text{ where } \boldsymbol{\pi}\sim\text{Dirichlet}\,(\mathbf{1}_M),\ \epsilon\sim\text{Gamma}(M,1),$$

and $\arg\min_i \pi_{(m\leftrightharpoons M)_i}e^{-\phi_i} = \arg\min_i \epsilon\pi_{(m\leftrightharpoons M)_i}e^{-\phi_i}$, we can re-express the gradient in (14) as

$$\begin{aligned}\nabla_{\phi_m}\mathcal{E}(\boldsymbol{\phi}) &= \mathbb{E}_{\epsilon\sim\text{Gamma}(M,1),\boldsymbol{\pi}\sim\text{Dirichlet}(\mathbf{1}_M)}[f(\arg\min_i \pi_{(m\leftrightharpoons M)_i}e^{-\phi_i})(1-\epsilon\pi_M)]\\ &= \mathbb{E}_{\boldsymbol{\pi}\sim\text{Dirichlet}(\mathbf{1}_M)}[f(\arg\min_i \pi_{(m\leftrightharpoons M)_i}e^{-\phi_i})(1-M\pi_M)]. \quad (17)\end{aligned}$$

Thus, plugging (17) into (13), we have now concluded the proof for Theorem 1. In Appendix, we summarize ARM stochastic gradient ascent in Algorithm 1.

2.4 ARM for binary random variables

Note in distribution, drawing $(u_1,u_2)\sim\text{Dirichlet}(1,1)$ is the same as drawing $u\sim\text{Uniform}[0,1]$ and letting $u_1 = u$ and $u_2 = 1-u$. Denoting $\phi = \phi_1 - \phi_2$, for the expectation as

$$\mathcal{E}([\phi_1,\phi_2]') = \mathbb{E}_{[z,1-z]\sim\text{Discrete}(\sigma([\phi_1,\phi_2]))}[f(z)] = \mathbb{E}_{z\sim\text{Bernoulli}(e^{\phi_1}/(e^{\phi_1}+e^{\phi_2}))}[f(z)],$$

we can deduce Corollary 2 from Theorem 1. Furthermore, the following Theorem shows that the ARM spike gradient for a binary random variable, expressed as $f_\Delta(u,\phi)(u-1/2)$, concentrates around 0, is unbiased, and achieves low variance.

Theorem 3. *Assume $f\geq 0$ (or $f\leq 0$) is a function of Bernoulli random variable z. Using a single Monte Carlo sample, the ARM spike gradient is expressed as $g_A(u,\phi) = f_\Delta(u,\phi)(u-1/2)$, where f_Δ is defined as in* (5) *and $u\sim\text{Uniform}(0,1)$, and the REINFORCE gradient is expressed as $g_R(z,\phi) = f(z)\nabla_\phi\log\text{Bernoulli}(z;\sigma(\phi)) = f(z)(z-\sigma(\phi))$, where $z\sim\text{Bernoulli}(\sigma(\phi))$. We have the following properties:*

(i) $f_\Delta(u,\phi)=0$ *with probability* $\sigma(|\phi|)-\sigma(-|\phi|)$, $f_\Delta(u,\phi)=f(1)-f(0)$ *with probability* $1-\sigma(|\phi|)$, *and* $f_\Delta(u,\phi)=f(0)-f(1)$ *with probability* $\sigma(-|\phi|)$.

(ii) $g_A(u,\boldsymbol{\phi})$ *is unbiased with* $\mathbb{E}_{u\sim\text{Uniform}(0,1)}[g_A(u,\boldsymbol{\phi})]=\mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[g_R(z,\boldsymbol{\phi})]$.

(iii) $g_A(u,\boldsymbol{\phi})$ *reaches its largest variance at* $0.039788[f(1)-f(0)]^2$ *when* $\frac{P(f_\Delta=0)}{P(f_\Delta\neq 0)}$ *is equal to the golden ratio* $\frac{\sqrt{5}+1}{2}$.

(iv) $\frac{\sup_\phi \text{Var}[g_A]}{\sup_\phi \text{Var}[g_R]}\leq\frac{16}{25}(1-2\frac{f(0)}{f(0)+f(1)})^2$ *and* $\sup_\phi \text{Var}[g_A]\leq\frac{1}{25}[f(1)-f(0)]^2$.

3 Backpropagation through multiple discrete stochastic layers

A latent variable model with multiple stochastic hidden layers can be constructed as

$$\boldsymbol{x}\sim p_{\boldsymbol{\theta}_0}(\boldsymbol{x}\,|\,\boldsymbol{b}_1),\ \boldsymbol{b}_1\sim p_{\boldsymbol{\theta}_1}(\boldsymbol{b}_1\,|\,\boldsymbol{b}_2),\ldots,\boldsymbol{b}_t\sim p_{\boldsymbol{\theta}_t}(\boldsymbol{b}_t\,|\,\boldsymbol{b}_{t+1}),\ldots,\boldsymbol{b}_T\sim p_{\boldsymbol{\theta}_T}(\boldsymbol{b}_T),\tag{18}$$

whose joint likelihood given the distribution parameters $\boldsymbol{\theta}_{0:T}=\{\boldsymbol{\theta}_0,\ldots,\boldsymbol{\theta}_T\}$ can be expressed as

$$p(\boldsymbol{x},\boldsymbol{b}_{1:T}\,|\,\boldsymbol{\theta}_{0:T})=p_{\boldsymbol{\theta}_0}(\boldsymbol{x}\,|\,\boldsymbol{b}_1)\Big[\prod\nolimits_{t=1}^{T-1}p_{\boldsymbol{\theta}_t}(\boldsymbol{b}_t\,|\,\boldsymbol{b}_{t+1})\Big]p_{\boldsymbol{\theta}_T}(\boldsymbol{b}_T).\tag{19}$$

In comparison to deterministic feedforward neural networks, stochastic ones can represent complex distributions and show natural resistance to overfitting [6, 7, 14, 29, 30]. However, training such kind of networks, especially if different stochastic layers are linked through discrete latent variables, is often considerably more difficult [29]. Below we show for both auto-encoding variational Bayes and maximum likelihood inference, how to apply the ARM estimator for gradient backpropagation in stochastic binary networks.

3.1 ARM variational auto-encoder

For auto-encoding variational Bayes inference [4, 5], we construct a variational distribution as

$$q_{\boldsymbol{w}_{1:T}}(\boldsymbol{b}_{1:T}\,|\,\boldsymbol{x})=q_{\boldsymbol{w}_1}(\boldsymbol{b}_1\,|\,\boldsymbol{x})\Big[\prod\nolimits_{t=1}^{T-1}q_{\boldsymbol{w}_{t+1}}(\boldsymbol{b}_{t+1}\,|\,\boldsymbol{b}_t)\Big],\tag{20}$$

with which the ELBO can be expressed as

$$\begin{aligned}&\mathcal{E}(\boldsymbol{w}_{1:T})=\mathbb{E}_{\boldsymbol{b}_{1:T}\sim q_{\boldsymbol{w}_{1:T}}(\boldsymbol{b}_{1:T}\,|\,\boldsymbol{x})}\left[f(\boldsymbol{b}_{1:T})\right],\ \text{where}\\&f(\boldsymbol{b}_{1:T})=\log p_{\boldsymbol{\theta}_0}(\boldsymbol{x}\,|\,\boldsymbol{b}_1)+\log p_{\boldsymbol{\theta}_{1:T}}(\boldsymbol{b}_{1:T})-\log q_{\boldsymbol{w}_{1:T}}(\boldsymbol{b}_{1:T}\,|\,\boldsymbol{x}).\end{aligned}\tag{21}$$

Theorem 4 (ARM backpropagation)**.** *For a stochastic binary network with* T *binary stochastic hidden layers, constructing a variational auto-encoder (VAE) defined wth* $\boldsymbol{b}_0=\boldsymbol{x}$ *and*

$$q_{\boldsymbol{w}_t}(\boldsymbol{b}_t\,|\,\boldsymbol{b}_{t-1})=\text{Bernoulli}(\boldsymbol{b}_t;\sigma(\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1})))\tag{22}$$

for $t=1,\ldots,T$, *the gradient of the ELBO with respect to* $\boldsymbol{w}_t$ *can be expressed as*

$$\begin{aligned}&\nabla_{\boldsymbol{w}_t}\mathcal{E}(\boldsymbol{w}_{1:T})=\mathbb{E}_{q(\boldsymbol{b}_{1:t-1})}\left[\mathbb{E}_{\boldsymbol{u}_t\sim\text{Uniform}(0,1)}[f_\Delta(\boldsymbol{u}_t,\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}),\boldsymbol{b}_{1:t-1})(\boldsymbol{u}_t-1/2)]\nabla_{\boldsymbol{w}_t}\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1})\right],\\&\text{where } f_\Delta(\boldsymbol{u}_t,\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}),\boldsymbol{b}_{1:t-1})=\mathbb{E}_{\boldsymbol{b}_{t+1:T}\sim q(\boldsymbol{b}_{t+1:T}\,|\,\boldsymbol{b}_t),\ \boldsymbol{b}_t=\mathbf{1}[\boldsymbol{u}_t>\sigma(-\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]}[f(\boldsymbol{b}_{1:T})]\\&\qquad\qquad\qquad\qquad-\mathbb{E}_{\boldsymbol{b}_{t+1:T}\sim q(\boldsymbol{b}_{t+1:T}\,|\,\boldsymbol{b}_t),\ \boldsymbol{b}_t=\mathbf{1}[\boldsymbol{u}_t<\sigma(\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]}[f(\boldsymbol{b}_{1:T})],\end{aligned}$$

which can be estimated with a single Monte Carlo sample as

$$\hat{f}_\Delta(\boldsymbol{u}_t,\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}),\boldsymbol{b}_{1:t-1})=\begin{cases}0, & \text{if } \boldsymbol{b}_t^{(1)}=\boldsymbol{b}_t^{(2)}\\ f(\boldsymbol{b}_{1:t-1},\boldsymbol{b}_{t:T}^{(1)})-f(\boldsymbol{b}_{1:t-1},\boldsymbol{b}_{t:T}^{(2)}), & \text{otherwise}\end{cases},\tag{23}$$

where $\boldsymbol{b}_t^{(1)}=\mathbf{1}[\boldsymbol{u}_t>\sigma(-\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]$, $\boldsymbol{b}_{t+1:T}^{(1)}\sim q(\boldsymbol{b}_{t+1:T}\,|\,\boldsymbol{b}_t^{(1)})$, $\boldsymbol{b}_t^{(2)}=\mathbf{1}[\boldsymbol{u}_t<\sigma(\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]$, *and* $\boldsymbol{b}_{t+1:T}^{(2)}\sim q(\boldsymbol{b}_{t+1:T}\,|\,\boldsymbol{b}_t^{(2)})$.

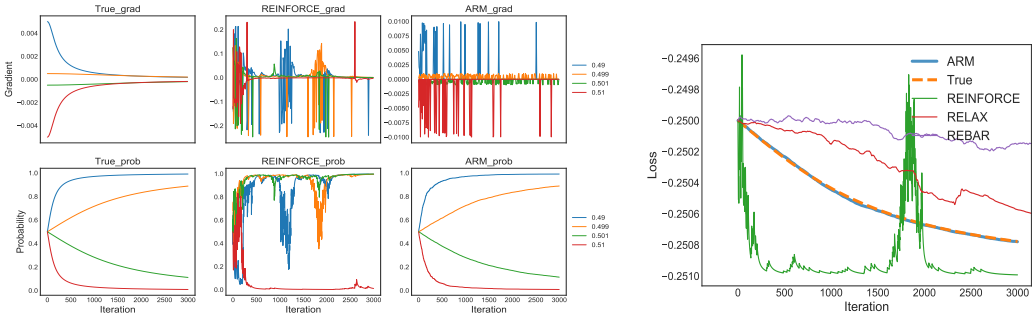


Figure 1: Left: Trace plots of the true/estimated gradients and estimated Bernoulli probability parameters for $p_0 \in \{0.49, 0.499, 0.501, 0.51\}$; Right: Trace plots of the loss functions for $p_0 = 0.499$.

3.2 ARM maximum likelihood inference

For maximum likelihood inference, the log marginal likelihood can be expressed as

$$\begin{aligned}\log p_{\boldsymbol{\theta}_{0:T}}(\boldsymbol{x}) &= \log \mathbb{E}_{\boldsymbol{b}_{1:T}\sim p_{\boldsymbol{\theta}_{1:T}}(\boldsymbol{b}_{1:T})}[p_{\boldsymbol{\theta}_0}(\boldsymbol{x}\,|\,\boldsymbol{b}_1)] \\ &\geq \mathcal{E}(\boldsymbol{\theta}_{1:T}) = \mathbb{E}_{\boldsymbol{b}_{1:T}\sim p_{\boldsymbol{\theta}_{1:T}}(\boldsymbol{b}_{1:T})}[\log p_{\boldsymbol{\theta}_0}(\boldsymbol{x}\,|\,\boldsymbol{b}_1)]. \end{aligned} \tag{24}$$

Generalizing Thoerem (4) leads to the following Corollary.

Corollary 5. *For a stochastic binary network defined as*

$$p_{\boldsymbol{\theta}_t}(\boldsymbol{b}_t\,|\,\boldsymbol{b}_{t+1}) = \text{Bernoulli}(\boldsymbol{b}_t; \sigma(\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}))), \tag{25}$$

the gradient of the lower bound in (24) *with respect to* $\boldsymbol{\theta}_t$ *can be expressed as*

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}_t}\mathcal{E}(\boldsymbol{\theta}_{1:T}) = \mathbb{E}_{p(\boldsymbol{b}_{t+1:T})}\left[\mathbb{E}_{\boldsymbol{u}_t\sim\text{Uniform}(0,1)}[f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}), \boldsymbol{b}_{t+1:T})(\boldsymbol{u}_t - 1/2)]\nabla_{\boldsymbol{\theta}_t}\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1})\right], \\ &\textit{where } f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}), \boldsymbol{b}_{t+1:T}) = \mathbb{E}_{\boldsymbol{b}_{1:t-1}\sim p(\boldsymbol{b}_{1:t-1}\,|\,\boldsymbol{b}_t),\ \boldsymbol{b}_t=\mathbf{1}[\boldsymbol{u}_t>\sigma(-\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}))]}[\log p_{\boldsymbol{\theta}_0}(\boldsymbol{x}\,|\,\boldsymbol{b}_1)] \\ &\qquad\qquad - \mathbb{E}_{\boldsymbol{b}_{1:t-1}\sim p(\boldsymbol{b}_{1:t-1}\,|\,\boldsymbol{b}_t),\ \boldsymbol{b}_t=\mathbf{1}[\boldsymbol{u}_t<\sigma(\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}))]}[\log p_{\boldsymbol{\theta}_0}(\boldsymbol{x}\,|\,\boldsymbol{b}_1)].\end{aligned}$$

4 Experimental results

To illustrate the working mechanism of the ARM estimator, related to [21, 22], we consider learning ϕ to maximize $\mathcal{E}(\phi) = \mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[(z - p_0)^2]$, where $p_0 \in \{0.49, 0.499, 0.501, 0.51\}$, or equivalently, minimize the loss as $-\mathcal{E}(\phi)$. The optimal solution is $\sigma(\phi) = \mathbf{1}(p_0 < 0.5)$. The closer p_0 is to 0.5, the more challenging the optimization becomes. We compare the ARM estimator to the true gradient as $g_\phi = (1 - 2p_0)\sigma(\phi)(1 - \sigma(\phi))$ and three previously proposed unbiased estimators, including REINFORCE, REBAR [21], and RELAX [22]. Following Theorem 3, with a single random sample $u \sim \text{Uniform}(0, 1)$ for Monte Carlo integration, the ARM spike gradient can be expressed as

$$g_{\phi,\text{ARM}} = \{[\mathbf{1}(u > \sigma(-\phi)) - p_0]^2 - [\mathbf{1}(u < \sigma(\phi)) - p_0]^2\}(u - 1/2),$$

while the REINFORCE gradient can be expressed as

$$g_{\phi,\text{REINFORCE}} = [\mathbf{1}(u < \sigma(\phi)) - p_0]^2[\mathbf{1}(u < \sigma(\phi)) - \sigma(\phi)].$$

See [21, 22] for the details about REBAR and RELAX.

As shown in Figure 1 (a), the REINFORCE gradients have large variances. Consequently, a REINFORCE based gradient ascent algorithm may diverge. For example, when $p_0 = 0.501$, the optimal value for the Bernoulli probability $\sigma(\phi)$ is 0, but the algorithm infers it to be close to 1 at the end of 3000 iterations of a random trial. By contrast, the ARM estimator well approximates the time-evolving true gradients by adjusting the frequencies, amplitudes, and signs of its gradient estimates, with larger and more frequent spikes for larger true gradients. Even though the trace plots of the ARM gradients, characterized by random spikes, look very different from these of the true gradients, which are slowly evolving over iterations, using them in gradient ascent leads to almost indistinguishable trace plots for the Bernoulli probability parameter $\sigma(\phi)$. As shown in Figure 1 (b), using the ARM estimator is indistinguishable from using the true gradient for updating ϕ to minimize the loss

Table 1: The constructions of three differently structured discrete variational auto-encoders. The following symbols "$\rightarrow$", "]",)", and "$\rightsquigarrow$" represent deterministic linear transform, leaky rectified linear units (LeakyReLU) [31] nonlinear activation, sigmoid nonlinear activation, and discrete stochastic activation, respectively, in the encoder (a.k.a., recognition network); their reversed versions are used in the decoder (a.k.a. generator).

	Nonlinear	Linear	Linear two layers
Encoder	784$\rightarrow$200]$\rightarrow$200]$\rightarrow$200)$\rightsquigarrow$200	784$\rightarrow$200)$\rightsquigarrow$200	784$\rightarrow$200)$\rightsquigarrow$200$\rightarrow$200)$\rightsquigarrow$200
Decoder	784$\leftsquigarrow$(784$\leftarrow$[200$\leftarrow$[200$\leftarrow$200	784$\leftsquigarrow$(784$\leftarrow$200	784$\leftsquigarrow$(784$\leftarrow$200$\leftsquigarrow$(200$\leftarrow$200

Table 2: Test negative ELBOs of discrete VAEs trained with four different stochastic gradient estimators.

			ARM	RELAX	REBAR	ST Gumbel-Softmax
Bernoulli	Nonlinear	MNIST	**101.3**	110.9	111.6	112.5
		OMNIGLOT	129.5	**128.2**	128.3	140.7
	Linear	MNIST	**110.3**	122.1	123.2	129.2
		OMNIGLOT	**124.2**	124.4	124.9	129.8
	Two layers	MNIST	**98.2**	114.0	113.7	NA
		OMNIGLOT	**118.3**	119.1	118.8	NA
Categorical	Nonlinear	MNIST	**105.8**	NA	NA	107.9
		OMNIGLOT	**121.9**	NA	NA	127.6

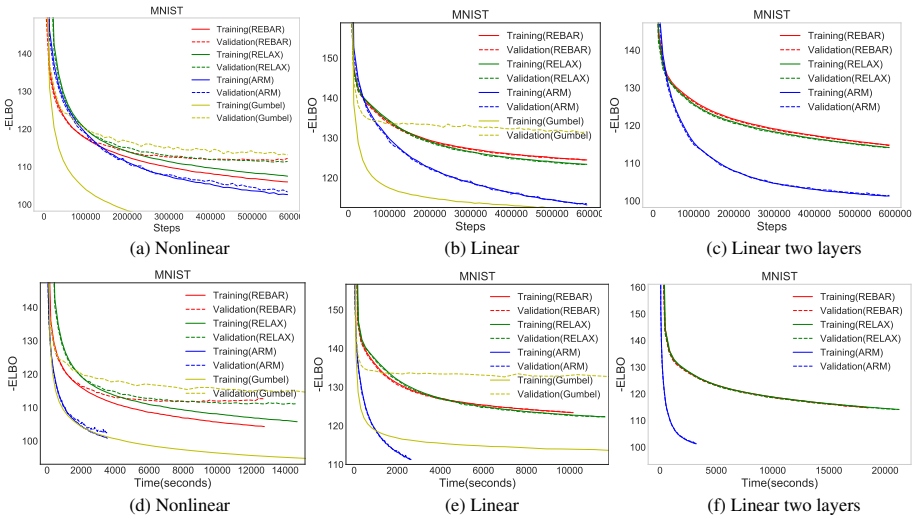
$-\mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[(z-0.499)^2]$, significantly outperforming not only REINFORCE, which has a large variance, but also both REBAR and RELAX, which improve on REINFORCE by introducing carefully constructed control variates for variance reduction. Updating ϕ using the true gradient, we further plot in Figure 5 of the Appendix the gradient estimated with multiple Monte Carlo samples against the true gradient at each iteration, showing that the ARM estimator has significantly lower variance than REINFORCE does given the same number of Monte Carlo samples.

4.1 Discrete variational auto-encoders

To optimize a variational auto-encoder (VAE) for a discrete latent variable model, existing solutions often rely on biased but low-variance stochastic gradient estimators [20, 32], unbiased but high-variance ones [13], or both unbiased and low-variance but computationally expensive ones [21, 22]. Comparing to previously proposed ones for discrete latent variables, the ARM estimator exhibits low variance and is unbiased, computationally efficient, and simple to implement.

For discrete VAEs, we compare ARM with Gumbel-Softmax [19,20], REBAR [22], and RELAX [21], three representative stochastic gradient estimators for discrete latent variables. Following the settings in [21, 22], for the encoder defined in (19) and decoder defined in (20), we consider three different network structures, including "Nonlinear" that has one stochastic but two LeaklyReLU deterministic hidden layers, "Linear" that has one stochastic hidden layer, and "Linear two layers" that has two stochastic hidden layers; we summarize the network structure in Table 1. We apply all four methods to both the MNIST and OMNIGLOT datasets.

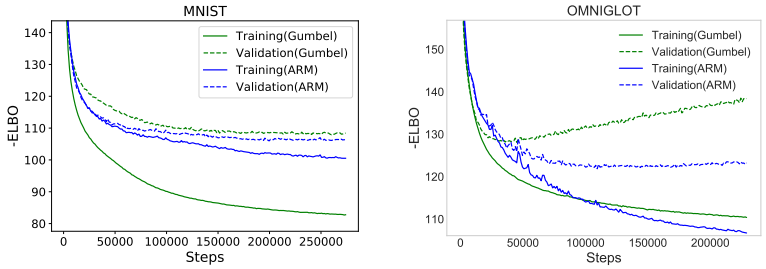
We train a discrete VAE with either Bernoulli or Categorical latent variables. More specifically, for each stochastic hidden layer, we use either 200 conditionally iid latent Bernoulli random variables or the concatenation of 20 conditionally iid categorical latent variables, each of which is represented as a 10 dimension one-hot vector. We maximize a single-Monte-Carlo-sample ELBO using Adam [33], with the learning rate set as 10^{-4} and batch size as 25 for Bernoulli VAEs and 100 for Categorical VAEs. Using the standard training-validation-testing splitting, we train all methods on the training set, calculate ELBO on the validation set for every epoch, and report the negative ELBO on the test set when the validation negative ELBO reaches its minimum. Training and validation curves are shown in Figure 2 and all numerical results are summarized in Table 2. Note to make a fair comparison between different methods, we report the results produced by our own experiments with public available REBAR and RELAX code provided for [22] and ST Gumbel-Softmax code provided for [20], sharing the same hyper-parameters and optimization procedure used in ARM. We also provide in Table 4 of the Appendix the comparison with the reported results of some additional algorithms, using the "Nonlinear" network structure on the MNIST dataset. The MNIST results show ARM outperforms the other competing methods in all tested network structures no matter at given steps or given times. On OMNIGLOT data, for nonlinear network, RELAX/REBAR achieve slightly higher ELBO but may be due to its severe overfitting caused by the auxiliary network used



(a) Nonlinear (b) Linear (c) Linear two layers

(d) Nonlinear (e) Linear (f) Linear two layers

Figure 2: Test negative ELBOs on MNIST with respect to training iterations, shown in the top row, and wall clock times on Tesla-K40 GPU, shown in the bottom row, for three differently structured Bernoulli VAEs.



(a) (b)

Figure 3: Comparison of the negative ELBOs for categorical variational auto-encoders trained by ARM and ST Gumbel-softmax on MNIST and OMNIGLOT, using the "Nonlinear" network.

Table 3: For the MNIST conditional distribution estimation benchmark task, comparison of the test negative log-likelihood between various gradient estimators, with the best results in [14, 20] reported here.

Gradient estimator	ARM	ST	$1/2$	Annealed ST	ST Gumbel-S.	SF	MuProp
$-\log p(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$	**54.8**	56.1	57.2	58.7	59.3	72.0	56.7

for variance reduction. For less overfitting linear and two-stochastic-layer networks, ARM performs on par with or better than RELAX/REBAR and converges significantly faster (about 6-8 times faster).

4.2 Maximum likelihood inference for a stochastic binary network

Denoting $\boldsymbol{x}_l, \boldsymbol{x}_u \in \mathbb{R}^{394}$ as the lower and upper halves of an MNIST digit, respectively, we consider a standard benchmark task of estimating the conditional distribution $p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$ [14, 20, 21, 30, 32], using a stochastic binary network with two stochastic binary hidden layers, expressed as

$$\boldsymbol{x}_l \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_0}(\boldsymbol{b}_1))),\ \boldsymbol{b}_1 \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_1}(\boldsymbol{b}_2))),\ \boldsymbol{b}_2 \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u))). \quad (26)$$

We set the network structure as $392 \leftrightsquigarrow (392 \leftarrow [300 \leftarrow 200 \leftrightsquigarrow (200 \leftarrow [300 \leftarrow 200 \leftrightsquigarrow (200 \leftarrow [300 \leftarrow 392$, which means both $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are 200 dimensional binary vectors, and $\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1})$ can be represented as $\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}) = \boldsymbol{\theta}_{t,1}\text{LeakyReLU}(\boldsymbol{\theta}_{t,2}\boldsymbol{b}_{t+1} + \boldsymbol{\theta}_{t,3}) + \boldsymbol{\theta}_{t,4}$.

We approximate $\log p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$ with $\log \frac{1}{K}\sum_{k=1}^{K} \text{Bernoulli}(\boldsymbol{x}_l; \sigma(\mathcal{T}_{\boldsymbol{\theta}_0}(\boldsymbol{b}_1^{(k)})))$, where $\boldsymbol{b}_1^{(k)} \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_1}(\boldsymbol{b}_2^{(k)})))$, $\boldsymbol{b}_2^{(k)} \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u)))$. We perform training with $K = 1$, which can also be considered as optimizing on a single-Monte-Carlo-sample estimate of the lower bound of the log likelihood shown in (24). We use Adam [33], with the learning rate set as 10^{-4}, mini-batch size as 100, and number of epochs for training as 2000. Given the inferred point estimate of $\boldsymbol{\theta}_{0:2}$ after training, we evaluate the accuracy of conditional density estimation by estimating the negative

log-likelihood as $-\log p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$, averaging over the test set using $K = 1000$. We show example results of predicting $\boldsymbol{x}_l$ given $\boldsymbol{x}_u$ in Figure 3 of the Appendix.

As shown in Table 3, optimizing a stochastic binary network with the ARM estimator, which is unbiased and computationally efficient, achieves the lowest test negative log-likelihood, outperforming all previously proposed biased stochastic gradient estimators on similarly structured stochastic networks, including $1/2$ [14, 34], straight through (ST) [32], slope-annealed ST [35], and ST Gumbel-softmax [20], and unbiased ones, including score-function (SF) and MuProp [14].

5 Conclusions

To train a discrete latent variable model with one or multiple discrete stochastic layers, we propose the augment-REINFORCE-merge (ARM) estimator to provide unbiased and low-variance gradient estimates of the parameters of discrete distributions. The ARM estimator modulates the frequencies, amplitudes, and signs of random spikes to facilitate gradient backpropagation and to track the temporal evolutions of the true gradients. Without relying on learning control variates for variance reduction, it maintains efficient computation and avoids increasing the risk of overfitting. Applying the ARM gradient leads to state-of-the-art out-of-sample prediction performance on both auto-encoding variational and maximum likelihood inference for discrete stochastic feedforward neural networks.

References

[1] Christopher M Bishop. *Neural Networks for Pattern Recognition*. Oxford university press, 1995.

[2] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[3] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[5] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pages 1278–1286, 2014.

[6] R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, pages 71–113, 1992.

[7] 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.

[8] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

[9] Michael C Fu. Gradient estimation. *Handbooks in operations research and management science*, 13:575–616, 2006.

[10] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[11] John Paisley, David M Blei, and Michael I Jordan. Variational Bayesian inference with stochastic search. In *ICML*, pages 1363–1370, 2012.

[12] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *AISTATS*, pages 814–822, 2014.

[13] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, pages 1791–1799, 2014.

[14] Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. MuProp: Unbiased backpropagation for stochastic neural networks. In *ICLR*, 2016.

[15] Andriy Mnih and Danilo Rezende. Variational inference for Monte Carlo objectives. In *ICML*, pages 2188–2196, 2016.

[16] Francisco J. R. Ruiz, Michalis K. Titsias, and David M. Blei. The generalized reparameterization gradient. In *NIPS*, pages 460–468, 2016.

[17] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017.

[18] Christian Naesseth, Francisco Ruiz, Scott Linderman, and David Blei. Reparameterization gradients through acceptance-rejection sampling algorithms. In *AISTATS*, pages 489–498, 2017.

[19] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.

[21] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *NIPS*, pages 2624–2633, 2017.

[22] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. In *ICLR*, 2018.

[23] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, 10th edition, 2006.

[24] Daniel McFadden. Conditional Logit Analysis of Qualitative Choice Behavior. In P. Zarembka, editor, *Frontiers in Econometrics*, pages 105–142. Academic Press, New York, 1974.

[25] Kenneth E. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2nd edition, 2009.

[26] Sheldon Ross. *A First Course in Probability*. Macmillan Publishing Co., New York, 1976.

[27] Art B. Owen. *Monte Carlo Theory, Methods and Examples*, chapter 8 Variance Reduction. 2013.

[28] Mingyuan Zhou and Lawrence Carin. Negative binomial process count and mixture modeling. *arXiv preprint arXiv:1209.3442v1*, 2012.

[29] Yichuan Tang and Ruslan R Salakhutdinov. Learning stochastic feedforward neural networks. In *NIPS*, pages 530–538, 2013.

[30] Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. Techniques for learning binary stochastic feedforward neural networks. *arXiv preprint arXiv:1406.2989*, 2014.

[31] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

[32] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[34] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. *arXiv preprint arXiv:1310.8499*, 2013.

[35] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Supplementary material for ARM: Augment-REINFORCE-merge gradient for discrete latent variable models

Algorithm 1: ARM stochastic gradient ascent

input : Categorical distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ with probability $\sigma(\boldsymbol{\phi})$, number of category M, target $f(\boldsymbol{z})$
output : $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ that maximize $\mathcal{E}(\boldsymbol{\phi}, \boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{z}\sim\text{Discrete}(\sigma(\boldsymbol{\phi}))}[f(\boldsymbol{z} \mid \boldsymbol{\psi})]$

while *not converged* **do**
  Let $\boldsymbol{\phi} = [\tilde{\boldsymbol{\phi}}, 0]$;
  Sample $\boldsymbol{z} \sim \text{Discrete}(\sigma(\boldsymbol{\phi}))$;
  sample $\boldsymbol{\pi} \sim \text{Dirichlet}(\mathbf{1}_M)$;
  $g_{\boldsymbol{\psi}} = \nabla_{\boldsymbol{\psi}} f(\boldsymbol{z}; \boldsymbol{\psi})$;
  $\tilde{f} = \frac{1}{M}\sum_{j=1}^{M} f(\arg\min_i \boldsymbol{\pi}_{(j=M)_i} e^{-\phi_i})$;
  for $m = 1$ **to** $M-1$ **do**
    $f_\Delta(\boldsymbol{\pi}, \boldsymbol{\phi}, m) = f(\arg\min_i \boldsymbol{\pi}_{(m=M)_i} e^{-\phi_i}) - \tilde{f}$;
    $(g_{\tilde{\boldsymbol{\phi}}})_m = f_\Delta(\boldsymbol{\pi}, \boldsymbol{\phi}, m)(1 - M\pi_M)$
  end
  $\tilde{\boldsymbol{\phi}} = \tilde{\boldsymbol{\phi}} + \rho_t g_{\tilde{\boldsymbol{\phi}}}, \quad \boldsymbol{\psi} = \boldsymbol{\psi} + \eta_t g_{\boldsymbol{\psi}}$ with step-size ρ_t, η_t
end

Proof of Theorem 3.

(i): Using the definition of $f_\Delta(u, \phi)$ in (5), when $\phi > 0$, we have

$$f_\Delta(u, \phi) = \begin{cases} 0 & \text{if } \sigma(-\phi) < u < \sigma(\phi) \\ f(1) - f(0) & \text{if } u > \sigma(\phi) \\ f(0) - f(1) & \text{if } u < \sigma(-\phi) \end{cases}$$

and when $\phi < 0$, we have

$$f_\Delta(u, \phi) = \begin{cases} 0 & \text{if } \sigma(\phi) < u < \sigma(-\phi) \\ f(1) - f(0) & \text{if } u > \sigma(-\phi) \\ f(0) - f(1) & \text{if } u < \sigma(\phi) \end{cases}$$

These two cases can be summarized as

$$f_\Delta(u, \phi) = \begin{cases} 0 & \text{if } \sigma(-|\phi|) < u < \sigma(|\phi|) \\ f(1) - f(0) & \text{if } u > \sigma(|\phi|) \\ f(0) - f(1) & \text{if } u < \sigma(-|\phi|) \end{cases} \tag{27}$$

(ii): With Eq.(27), we have

$$\begin{aligned} &\mathbb{E}_{u\sim\text{Uniform}(0,1)}[g_A(u, \phi)] = \mathbb{E}_{u\sim\text{Uniform}(0,1)}[f_\Delta(u, \phi)(u - 1/2)] \\ =&\int_{\sigma(|\phi|)}^{1} (f(1) - f(0))(u - 1/2)du + \int_{0}^{\sigma(-|\phi|)} (f(0) - f(1))(u - 1/2)du \\ =&\left[\sigma(|\phi|)(1 - \sigma(|\phi|))/2 + \sigma(-|\phi|)(1 - \sigma(-|\phi|))/2\right][f(1) - f(0)] \\ =&\sigma(|\phi|)(1 - \sigma(|\phi|))[f(1) - f(0)] \\ =&\sigma(\phi)(1 - \sigma(\phi))[f(1) - f(0)]. \end{aligned}$$

Since $\frac{d\sigma(\phi)}{d\phi} = \sigma(\phi)(1 - \sigma(\phi))$, the expectation for the REINFORCE gradient is

$$\begin{aligned} &\mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[g_R(z, \phi)] = \mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[f(z)\nabla_\phi \log(\sigma(\phi)^z(1 - \sigma(\phi))^{1-z})] \\ =&\mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[f(z)(z(1 - \sigma(\phi)) - \sigma(\phi)(1 - z))] \\ =&\sigma(\phi)(1 - \sigma(\phi))[f(1) - f(0)]. \end{aligned}$$

Thus $\mathbb{E}[g_A] = \mathbb{E}[g_R]$ and $g_A(u, \phi)$ is an unbiased estimator for the true gradient.

(iii): The second moment of $g_A(u,\phi)$ can be expressed as

$$\begin{aligned}
&\mathbb{E}_{u\sim\text{Uniform}(0,1)}[g_A^2(u,\phi)] = \mathbb{E}_{u\sim\text{Uniform}(0,1)}[f_\Delta^2(u,\phi)(u-1/2)^2] \\
=&\int_{\sigma(|\phi|)}^{1}[f(1)-f(0)]^2(u-1/2)^2du + \int_0^{\sigma(-|\phi|)}[f(0)-f(1)]^2(u-1/2)^2du \\
=&\frac{1}{12}[1-(\sigma(|\phi|)-\sigma(-|\phi|))^3][f(1)-f(0)]^2
\end{aligned}$$

Denoting $t = \sigma(|\phi|) - \sigma(-|\phi|) = P(f_\Delta = 0)$, we can re-express (28) as $\mathbb{E}_{u\sim\text{Uniform}(0,1)}[g_A(u,\phi)] = \frac{1}{4}(1-t^2)[f(1)-f(0)]$. Thus, the variance of $g_A(u,\phi)$ can be expressed as

$$\begin{aligned}
\text{Var}[g_A(u,\phi)] =&\frac{1}{4}\left[\frac{1}{3}(1-t^3)-\frac{1}{4}(1-t^2)^2\right][f(1)-f(0)]^2 \\
=&\frac{1}{16}(1-t)(t^3+\frac{7}{3}t^2+\frac{1}{3}t+\frac{1}{3})[f(1)-f(0)]^2 \\
\leq&0.039788[f(1)-f(0)]^2 \\
\leq&\frac{1}{25}[f(1)-f(0)]^2,
\end{aligned}$$

where $\text{Var}(g_A)$ reaches its maximum at $0.039788[f(1)-f(0)]^2$ when $t=\frac{\sqrt{5}-1}{2}$, which means $\frac{P(f_\Delta=0)}{P(f_\Delta\neq 0)} = \frac{\sqrt{5}+1}{2}$.

(iv): For the REINFORCE gradient, we have

$$\begin{aligned}
\mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[g_R^2(z,\phi)] =&\mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}\left[f^2(z)(z(1-\sigma(\phi))-\sigma(\phi)(1-z))^2\right] \\
=&f^2(1)(1-\sigma(\phi))^2\sigma(\phi)+f^2(0)(-\sigma(\phi))^2(1-\sigma(\phi)) \\
=&\sigma(\phi)(1-\sigma(\phi))[(1-\sigma(\phi))f^2(1)+\sigma(\phi)f^2(0)].
\end{aligned}$$

Therefore the variance can be expressed as

$$\begin{aligned}
&\text{Var}[g_R(u,\phi)] \\
=&\sigma(\phi)(1-\sigma(\phi))\left[(1-\sigma(\phi))f^2(1)+\sigma(\phi)f^2(0)-\sigma(\phi)(1-\sigma(\phi))[f(1)-f(0)]^2\right] \\
=&\sigma(\phi)(1-\sigma(\phi))[(1-\sigma(\phi))f(1)+\sigma(\phi)f(0)]^2
\end{aligned}$$

The largest variance satisfies

$$\sup_\phi \text{Var}[g_R(z,\phi)] \geq \text{Var}[g_R(z,0)] = \frac{1}{16}(f(1)+f(0))^2,$$

and hence when $f \geq 0$ a.s. or $f \leq 0$ a.s., we have

$$\sup_\phi \text{Var}[g_R(z,\phi)] \geq \text{Var}[g_R(z,0)] > \frac{1}{25}(f(1)-f(0))^2 \geq \sup_\phi \text{Var}[g_A(u,\phi)],$$

which means the ARM spike gradient has a variance that is bounded by $\frac{1}{25}(f(1)-f(0))^2$, and its worst-case variance is smaller than that of the REINFORCE gradient.

□

Proof of Theorem 4. First, to compute the gradient with respect to $\boldsymbol{w}_1$, since

$$\mathcal{E}(\boldsymbol{\phi}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_1)}\mathbb{E}_{q(\boldsymbol{b}_{2:T}\,|\,\boldsymbol{b}_1)}[f(\boldsymbol{b}_{1:T})] \tag{28}$$

we have

$$\nabla_{\boldsymbol{\phi}_1}\mathcal{E}(\boldsymbol{\phi}_{1:T}) = \mathbb{E}_{\boldsymbol{u}_1\sim\text{Uniform}(0,1)}[f_\Delta(\boldsymbol{u}_1,\mathcal{T}_{\boldsymbol{\phi}_1}(\boldsymbol{x}))(\boldsymbol{u}_1-1/2)]\nabla_{\boldsymbol{\phi}_1}\mathcal{T}_{\boldsymbol{\phi}_1}(\boldsymbol{x}), \tag{29}$$

where

$$\begin{aligned}
f_\Delta(\boldsymbol{u}_1,\mathcal{T}_{\boldsymbol{\phi}_1}(\boldsymbol{x})) = &\mathbb{E}_{\boldsymbol{b}_{2:T}\sim q(\boldsymbol{b}_{2:T}\,|\,\boldsymbol{b}_1),\ \boldsymbol{b}_1=\mathbf{1}[\boldsymbol{u}_1>\sigma(-\mathcal{T}_{\boldsymbol{\phi}_1}(\boldsymbol{x}))]}[f(\boldsymbol{b}_{1:T})] \\
&-\mathbb{E}_{\boldsymbol{b}_{2:T}\sim q(\boldsymbol{b}_{2:T}\,|\,\boldsymbol{b}_1),\ \boldsymbol{b}_1=\mathbf{1}[\boldsymbol{u}_1<\sigma(\mathcal{T}_{\boldsymbol{\phi}_1}(\boldsymbol{x}))]}[f(\boldsymbol{b}_{1:T})]
\end{aligned} \tag{30}$$

Figure 4: Randomly selected example results of predicting the lower half of a MNIST digit given its upper half, using a binary stochastic network, which has two binary stochastic hidden layers and is trained by ARM maximum likelihood inference.

Table 4: Comparison of the test negative ELBOs of various algorithms, using the same "Nonlinear" network structure.

	ARM	ST	$1/2$	Annealed ST	ST Gumbel-S.	SF	MuProp
VAE (Bernoulli)	**101.3**	116.0	110.9	111.5	111.5	112.2	109.7
VAE (Categorical)	**107.1**	110.9	128.8	107.8	107.8	110.6	107.8

Second, to compute the gradient with respect to $\boldsymbol{\phi}_t$, where $2 \leq t \leq T-1$, since

$$\mathcal{E}(\boldsymbol{\phi}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_{1:t-1})}\mathbb{E}_{q(\boldsymbol{b}_t \mid \boldsymbol{b}_{t-1})}\mathbb{E}_{q(\boldsymbol{b}_{t+1:T} \mid \boldsymbol{b}_t)}[f(\boldsymbol{b}_{1:T})] \tag{31}$$

we have

$$\nabla_{\boldsymbol{\phi}_t}\mathcal{E}(\boldsymbol{\phi}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_{1:t-1})}\left[\mathbb{E}_{\boldsymbol{u}_t \sim \text{Uniform}(0,1)}[f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{\phi}_t}(\boldsymbol{b}_{t-1}), \boldsymbol{b}_{1:t-1})(\boldsymbol{u}_t - 1/2)]\nabla_{\boldsymbol{\phi}_t}\mathcal{T}_{\boldsymbol{\phi}_t}(\boldsymbol{b}_{t-1})\right], \tag{32}$$

where

$$\begin{aligned} f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{\phi}_t}(\boldsymbol{b}_{t-1}), \boldsymbol{b}_{1:t-1}) = {} & \mathbb{E}_{\boldsymbol{b}_{t+1:T} \sim q(\boldsymbol{b}_{t+1:T} \mid \boldsymbol{b}_t),\ \boldsymbol{b}_t = \mathbf{1}[\boldsymbol{u}_t > \sigma(-\mathcal{T}_{\boldsymbol{\phi}_t}(\boldsymbol{b}_{t-1}))]}[f(\boldsymbol{b}_{1:T})] \\ & - \mathbb{E}_{\boldsymbol{b}_{t+1:T} \sim q(\boldsymbol{b}_{t+1:T} \mid \boldsymbol{b}_t),\ \boldsymbol{b}_t = \mathbf{1}[\boldsymbol{u}_t < \sigma(\mathcal{T}_{\boldsymbol{\phi}_t}(\boldsymbol{b}_{t-1}))]}[f(\boldsymbol{b}_{1:T})] \end{aligned} \tag{33}$$

Finally, to compute the gradient with respect to $\boldsymbol{\phi}_T$, we have

$$\nabla_{\boldsymbol{\phi}_T}\mathcal{E}(\boldsymbol{\phi}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_{1:T-1})}\left[\mathbb{E}_{\boldsymbol{u}_T \sim \text{Uniform}(0,1)}[f_\Delta(\boldsymbol{u}_T, \mathcal{T}_{\boldsymbol{\phi}_T}(\boldsymbol{b}_{T-1}), \boldsymbol{b}_{1:T-1})(\boldsymbol{u}_T - 1/2)]\nabla_{\boldsymbol{\phi}_T}\mathcal{T}_{\boldsymbol{\phi}_T}(\boldsymbol{b}_{T-1})\right], \tag{34}$$

$$\begin{aligned} f_\Delta(\boldsymbol{u}_T, \mathcal{T}_{\boldsymbol{\phi}_T}(\boldsymbol{b}_{T-1}), \boldsymbol{b}_{1:T-1}) = {} & f(\boldsymbol{b}_{1:T-1}, \boldsymbol{b}_T = \mathbf{1}[\boldsymbol{u}_T > \sigma(-\mathcal{T}_{\boldsymbol{\phi}_T}(\boldsymbol{b}_{T-1}))])) \\ & - f(\boldsymbol{b}_{1:T-1}, \boldsymbol{b}_T = \mathbf{1}[\boldsymbol{u}_T < \sigma(\mathcal{T}_{\boldsymbol{\phi}_T}(\boldsymbol{b}_{T-1}))]) \end{aligned} \tag{35}$$

□

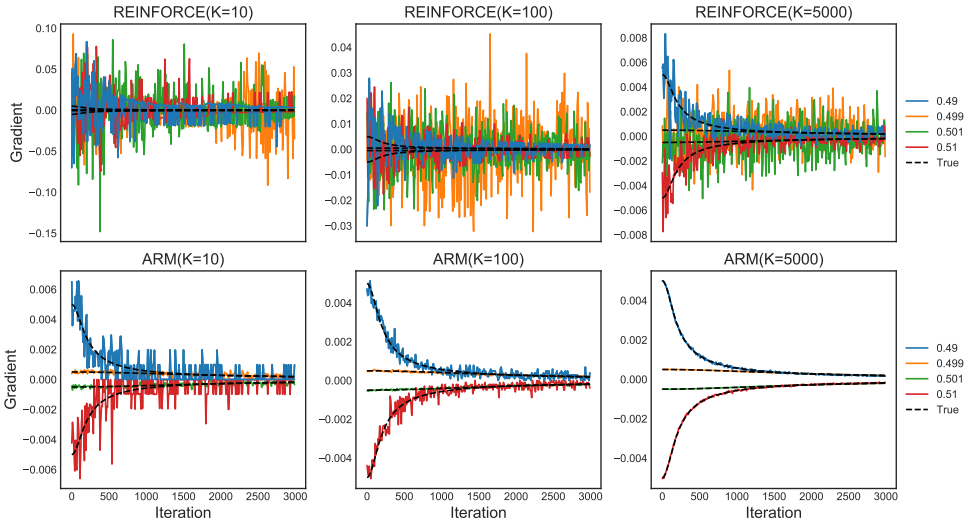
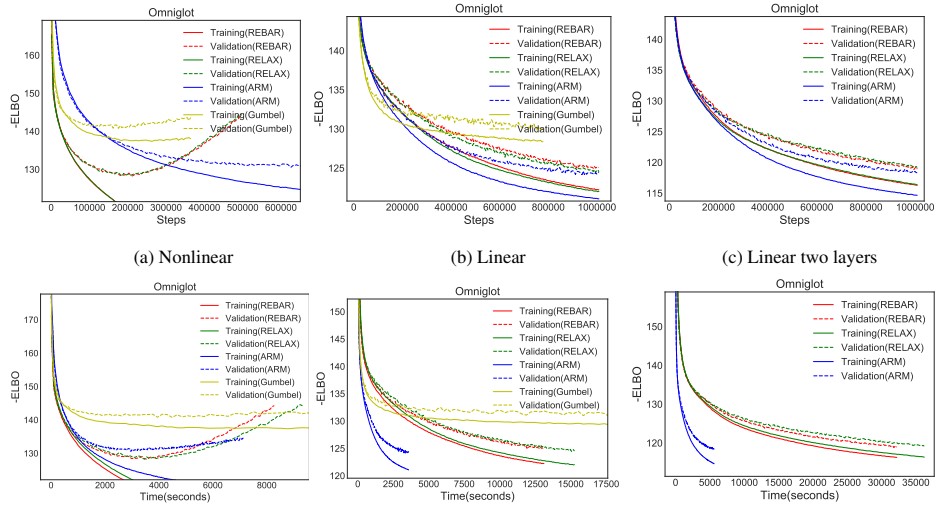


Figure 5: Estimation of the true gradient at each iteration using $K > 1$ Monte Carlo samples, using REINFORCE, shown in the top row, or ARM, shown in the bottom row. The ARM estimator exhibits significant lower variance given the same number of Monte Carlo samples.



(a) Nonlinear (b) Linear (c) Linear two layers

(d) Nonlinear (e) Linear (f) Linear two layers

Figure 6: Test negative ELBOs on OMNIGLOT with respect to training iterations, shown in the top row, and wall clock times on Tesla-K40 GPU, shown in the bottom row, for three differently structured Bernoulli VAEs.