ARSM: Augment-REINFORCE-Swap-Merge Estimator for Gradient Backpropagation Through Categorical Variables

Mingzhang Yin [*1] **Yuguang Yue** [*1] **Mingyuan Zhou** [2]

Abstract

To address the challenge of backpropagating the gradient through categorical variables, we propose the augment-REINFORCE-swap-merge (ARSM) gradient estimator that is unbiased and has low variance. ARSM first uses variable augmentation, REINFORCE, and Rao-Blackwellization to re-express the gradient as an expectation under the Dirichlet distribution, then uses variable swapping to construct differently expressed but equivalent expectations, and finally shares common random numbers between these expectations to achieve significant variance reduction. Experimental results show ARSM closely resembles the performance of the true gradient for optimization in univariate settings; outperforms existing estimators by a large margin when applied to categorical variational auto-encoders; and provides a "try-and-see self-critic" variance reduction method for discrete-action policy gradient, which removes the need of estimating baselines by generating a random number of pseudo actions and estimating their action-value functions.

1. Introduction

The need to maximize an objective function, expressed as the expectation over categorical variables, arises in a wide variety of settings, such as discrete latent variable models (Zhou, 2014; Jang et al., 2017; Maddison et al., 2017) and policy optimization for reinforcement learning (RL) with discrete actions (Sutton & Barto, 1998; Weaver & Tao, 2001; Schulman et al., 2015; Mnih et al., 2016; Grathwohl et al., 2018). More specifically, let us denote $z_k \in \{1, 2, \ldots, C\}$ as a univariate C-way categorical variable, and $\boldsymbol{z} = (z_1, \ldots, z_K) \in \{1, 2, \ldots, C\}^K$ as a K-dimensional C-way multivariate categorical vector. In discrete latent variable models, K will be the dimension of the discrete latent space, each dimension of which can be further represented as a C-dimensional one-hot vector. In RL, C represents the size of the discrete action space and $\boldsymbol{z}$ is a sequence of discrete actions from that space. In even more challenging settings, one may have a sequence of K-dimensional C-way multivariate categorical vectors, which appear both in categorical latent variable models with multiple stochastic layers, and in RL with a high dimensional discrete action space or multiple agents, which may consist of as many as C^K unique combinations at each time step.

With $f(\boldsymbol{z})$ and $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ denoted as the reward function and distribution for categorical $\boldsymbol{z}$, respectively, we need to optimize parameter $\boldsymbol{\phi}$ to maximize the expected reward as

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

Here we consider both categorical latent variable models and policy optimization for discrete actions, which arise in a wide array of real-world applications. A number of unbiased estimators for backpropagating the gradient through discrete latent variables have been recently proposed (Tucker et al., 2017; Grathwohl et al., 2018; Yin & Zhou, 2019; Andriyash et al., 2018). However, they all mainly, if not exclusively, focus on the binary case ($i.e.$, $C = 2$). The categorical case ($i.e.$, $C \geq 2$) is more widely applicable but generally much more challenging. In this paper, to optimize the objective in (1), inspired by the augment-REINFORCE-merge (ARM) gradient estimator restricted for binary variables (Yin & Zhou, 2019), we introduce the augment-REINFORCE-swap-merge (ARSM) estimator that is unbiased and well controls its variance for categorical variables.

The proposed ARSM estimator combines variable augmentation (Tanner & Wong, 1987; Van Dyk & Meng, 2001), REINFORCE (Williams, 1992) in an augmented space, Rao-Blackwellization (Casella & Robert, 1996), and a merge step that shares common random numbers between different but equivalent gradient expectations to achieve significant variance reduction. While ARSM with $C = 2$ reduces to the ARM estimator (Yin & Zhou, 2019), whose merge step can

[*]Equal contribution [1]Department of Statistics and Data Sciences, [2]Department of IROM, McCombs School of Business, The University of Texas at Austin, Austin, TX 78712, USA. Correspondence to: Mingyuan Zhou <mingyuan.zhou@mccombs.utexas.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

be realized by applying antithetic sampling (Owen, 2013) in the augmented space, the merge step of ARSM with $C > 2$ cannot be realized in this manner. Instead, ARSM requires distinct variable-swapping operations to construct differently expressed but equivalent expectations under the Dirichlet distribution before performing its merge step.

Experimental results on both synthetic data and several representative tasks involving categorical variables are used to illustrate the distinct working mechanism of ARSM. In particular, our experimental results on latent variable models with one or multiple categorical stochastic hidden layers show that ARSM provides state-of-the-art training and out-of-sample prediction performance. Our experiments on RL with discrete action spaces show that ARSM provides a "try-and-see self-critic" method to produce unbiased and low-variance policy gradient estimates, removing the need of constructing baselines by generating a random number of pseudo actions at a given state and estimating their action-value functions. These results demonstrate the effectiveness and versatility of the ARSM estimator for gradient backpropagation through categorical stochastic layers. Python code for reproducible research is available at https://github.com/ARM-gradient/ARSM.

1.1. Related Work

For optimizing (1) for categorical $\boldsymbol{z}$, the difficulty lies in developing a low-variance and preferably unbiased estimator for its gradient with respect to $\boldsymbol{\phi}$, expressed as $\nabla_{\boldsymbol{\phi}}\mathcal{E}(\boldsymbol{\phi})$. An unbiased but high-variance gradient estimator that is universally applicable to (1) is REINFORCE (Williams, 1992). Using the score function $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}) = \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{z})/q_{\boldsymbol{\phi}}(\boldsymbol{z})$, REINFORCE expresses the gradient as an expectation as

$$\nabla_{\boldsymbol{\phi}}\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[f(\boldsymbol{z})\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z})], \qquad (2)$$

and approximates it with Monte Carlo integration (Owen, 2013). However, the estimation variance with a limited number of Monte Carlo samples is often too high to make vanilla REINFORCE a sound choice for categorical $\boldsymbol{z}$.

To address the high-estimation-variance issue for categorical $\boldsymbol{z}$, one often resorts to a biased gradient estimator. For example, Maddison et al. (2017) and Jang et al. (2017) relax the categorical variables with continuous ones and then apply the reparameterization trick to estimate the gradients, reducing variance but introducing bias. Other biased estimators for backpropagating through binary variables include the straight-through estimator (Hinton, 2012; Bengio et al., 2013) and the ones of Gregor et al. (2014); Raiko et al. (2014); Cheng et al. (2018). With biased gradient estimates, however, a gradient ascent algorithm may not be guaranteed to work, or may converge to unintended solutions.

To keep REINFORCE unbiased while sufficiently reducing its variance, a usual strategy is to introduce appropriate control variates, also known as baselines (Williams, 1992), into the expectation in (2) before performing Monte Carlo integration (Paisley et al., 2012; Ranganath et al., 2014; Mnih & Gregor, 2014; Gu et al., 2016; Mnih & Rezende, 2016; Ruiz et al., 2016; Kucukelbir et al., 2017; Naesseth et al., 2017). For discrete $\boldsymbol{z}$, Tucker et al. (2017) and Grathwohl et al. (2018) improve REINFORCE by introducing continuous relaxation based baselines, whose parameters are optimized by minimizing the sample variance of gradient estimates.

2. ARSM Gradient For Categorical Variables

Let us denote $z \sim \text{Cat}(\sigma(\boldsymbol{\phi}))$ as a categorical variable such that $P(z = c \,|\, \boldsymbol{\phi}) = \sigma(\boldsymbol{\phi})_c = e^{\phi_c}/\sum_{i=1}^{C} e^{\phi_i}$, where $\boldsymbol{\phi} := (\phi_1, \ldots, \phi_C)$ and $\sigma(\boldsymbol{\phi}) := (e^{\phi_1}, \ldots, e^{\phi_C})/\sum_{i=1}^{C} e^{\phi_i}$ is the softmax function. For the expectated reward defined as

$$\mathcal{E}(\boldsymbol{\phi}) := \mathbb{E}_{z\sim\text{Cat}(\sigma(\boldsymbol{\phi}))}[f(z)] = \textstyle\sum_{i=1}^{C} f(i)\sigma(\boldsymbol{\phi})_i,$$

the gradient can be expressed analytically as

$$\nabla_{\phi_c}\mathcal{E}(\boldsymbol{\phi}) = \sigma(\boldsymbol{\phi})_c f(c) - \sigma(\boldsymbol{\phi})_c\mathcal{E}(\boldsymbol{\phi}) \qquad (3)$$

or expressed with REINFORCE as

$$\nabla_{\phi_c}\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{z\sim\text{Cat}(\sigma(\boldsymbol{\phi}))}\left[f(z)(\mathbf{1}_{[z=c]} - \sigma(\boldsymbol{\phi})_c)\right], \qquad (4)$$

where $\mathbf{1}_{[\cdot]}$ is an indicator function that is equal to one if the argument is true and zero otherwise. However, the analytic expression quickly becomes intractable for a multivariate setting, and the REINFORCE estimator often comes with significant estimation variance. While the ARM estimator of Yin & Zhou (2019) is unbiased and provides significant variance reduction for binary variables, it is restricted to $C = 2$ and hence has limited applicability.

Below we introduce the augment-REINFORCE (AR), AR-swap (ARS), and ARS-merge (ARSM) estimators for a univariate C-way categorical variable, and later generalize them to multivariate, hierarchical, and sequential settings.

2.1. AR: Augment-REINFORCE

Let us denote $\boldsymbol{\pi} := (\pi_1, \ldots, \pi_C) \sim \text{Dir}(\mathbf{1}_C)$ as a Dirichlet distribution whose C parameters are all ones. We first state three statistical properties that can directly lead to the proposed AR estimator. We describe in detail in Appendix A how we actually arrive at the AR estimator, with these properties obtained as by-products, by performing variable augmentation, REINFORCE, and Rao-Blackwellization. Thus we are in fact reverse-engineering our original derivation of the AR estimator to help concisely present our findings.

Property I. *The categorical variable $z \sim \text{Cat}(\sigma(\boldsymbol{\phi}))$ can be equivalently generated as*

$$z := \arg\min_{i\in\{1,\ldots,C\}} \pi_i e^{-\phi_i},\ \boldsymbol{\pi} \sim \text{Dir}(\mathbf{1}_C).$$

Property II. $\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{\pi}\sim\text{Dir}(\mathbf{1}_C)}[f(\arg\min_i \pi_i e^{-\phi_i})]$.
Property III. $\mathbb{E}_{\boldsymbol{\pi}\sim\text{Dir}(\mathbf{1}_C)}[f(\arg\min_i \pi_i e^{-\phi_i})C\pi_c] = \mathcal{E}(\boldsymbol{\phi}) + \sigma(\boldsymbol{\phi})_c\mathcal{E}(\boldsymbol{\phi}) - \sigma(\boldsymbol{\phi})_c f(c)$.

These three properties, Property III in particular, are previously unknown to the best of our knowledge. They are directly linked to the AR estimator shown below.

Theorem 1 (AR estimator)**.** *The gradient of* $\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{z\sim\text{Cat}(\sigma(\boldsymbol{\phi}))}[f(z)]$*, as shown in* (3)*, can be re-expressed as an expectation under a Dirichlet distribution as*

$$\begin{aligned}\nabla_{\phi_c}\mathcal{E}(\boldsymbol{\phi}) &= \mathbb{E}_{\boldsymbol{\pi}\sim\text{Dir}(\mathbf{1}_C)}[g_{\text{AR}}(\boldsymbol{\pi})_c],\\ g_{\text{AR}}(\boldsymbol{\pi})_c &:= f(z)(1 - C\pi_c),\\ z &:= \arg\min_{i\in\{1,\dots,C\}} \pi_i e^{-\phi_i}.\end{aligned} \tag{5}$$

Distinct from REINFORCE in (4), the AR estimator in (5) now expresses the gradient as an expectation under a Dirichlet distributed random noise. From this point of view, it is somewhat related to the reparameterization trick (Kingma & Welling, 2013; Rezende et al., 2014), which is widely used to express the gradient of an expectation under reparameterizable random variables as an expectation under random noises. Thus one may consider AR as a special type of reparameterization gradient, which, however, requires neither z to be reparameterizable nor $f(\cdot)$ to be differentiable.

2.2. ARS: Augment-REINFORCE-Swap

Let us swap the mth and jth elements of $\boldsymbol{\pi}$ to define vector

$$\boldsymbol{\pi}^{m\leftrightharpoons j} := (\pi_1^{m\leftrightharpoons j}, \dots, \pi_C^{m\leftrightharpoons j}),$$

where $\pi_m^{m\leftrightharpoons j} = \pi_j$, $\pi_j^{m\leftrightharpoons j} = \pi_m$, and $\forall\ c \notin \{m, j\}$, $\pi_c^{m\leftrightharpoons j} = \pi_c$. Another property to be repeatedly used is:

Property IV. *If* $\boldsymbol{\pi} \sim \text{Dir}(\mathbf{1}_C)$*, then* $\boldsymbol{\pi}^{m\leftrightharpoons j} \sim \text{Dir}(\mathbf{1}_C)$.

This leads to a key observation for the AR estimator in (5): swapping any two variables of the probability vector $\boldsymbol{\pi}$ inside the expectation does not change the expected value. Using the idea of sharing common random numbers between different expectations to potentially significantly reduce Monte Carlo integration variance (Owen, 2013), we propose to swap π_c and π_j in (5), where $j \in \{1, \dots, C\}$ is a reference category chosen independently of $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$. This variable-swapping operation changes the AR estimator to

$$\begin{aligned}\nabla_{\phi_c}\mathcal{E}(\boldsymbol{\phi}) &= \mathbb{E}_{\boldsymbol{\pi}\sim\text{Dir}(\mathbf{1}_C)}[g_{\text{AR}}(\boldsymbol{\pi}^{c\leftrightharpoons j})_c]\\ g_{\text{AR}}(\boldsymbol{\pi}^{c\leftrightharpoons j})_c &:= f(z^{c\leftrightharpoons j})(1 - C\pi_j),\\ z^{c\leftrightharpoons j} &:= \arg\min_{i\in\{1,\dots,C\}} \pi_i^{c\leftrightharpoons j} e^{-\phi_i},\end{aligned} \tag{6}$$

where we have applied identity $\pi_c^{c\leftrightharpoons j} = \pi_j$ and Property IV. We refer to z defined in (5) as the "true action," and $z^{c\leftrightharpoons j}$ defined in (6) as the cth "pseudo action" given j as the reference category. Note the pseudo actions satisfy the following properties: $z^{c\leftrightharpoons j} = z^{j\leftrightharpoons c}$ and $z^{c\leftrightharpoons j} = z$ if $c = j$, and the number of unique values in $\{z^{c\leftrightharpoons j}\}_{c,j}$ that are different from the true action z is between 0 and $C - 1$.

With (3), we have another useful property as

Property V. $\sum_{c=1}^C \nabla_{\phi_c}\mathcal{E}(\boldsymbol{\phi}) = 0$.

Combining it with the estimator in (6) leads to

$$\mathbb{E}_{\boldsymbol{\pi}\sim\text{Dir}(\mathbf{1}_C)}\big[\tfrac{1}{C}\textstyle\sum_{c=1}^C g_{\text{AR}}(\boldsymbol{\pi}^{c\leftrightharpoons j})_c\big] = 0. \tag{7}$$

Thus we can utilize $\frac{1}{C}\sum_{c=1}^C g_{\text{AR}}(\boldsymbol{\pi}^{c\leftrightharpoons j})_c$ as a baseline function that is nonzero in general but has zero expectation under $\boldsymbol{\pi} \sim \text{Dir}(\mathbf{1}_C)$. Subtracting (7) from (6) leads to another unbiased estimator, with category j as the reference, as

$$\begin{aligned}&\nabla_{\phi_c}\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{\pi}\sim\text{Dir}(\mathbf{1}_C)}[g_{\text{ARS}}(\boldsymbol{\pi}, j)_c],\\ &g_{\text{ARS}}(\boldsymbol{\pi}, j)_c := g_{\text{AR}}(\boldsymbol{\pi}^{c\leftrightharpoons j})_c - \tfrac{1}{C}\textstyle\sum_{m=1}^C g_{\text{AR}}(\boldsymbol{\pi}^{m\leftrightharpoons j})_m,\\ &= \big[f(z^{c\leftrightharpoons j}) - \tfrac{1}{C}\textstyle\sum_{m=1}^C f(z^{m\leftrightharpoons j})\big](1 - C\pi_j),\end{aligned} \tag{8}$$

which is referred to as the AR-swap (ARS) estimator, due to the use of variable-swapping in its derivation from AR.

2.3. ARSM: Augment-REINFORCE-Swap-Merge

For ARS in (8), when the reference category j is randomly chosen from $\{1, \dots, C\}$ and hence is independent of $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$, it is unbiased. Furthermore, we find that it can be further improved, especially when C is large, by adding a merge step to construct the ARS-merge (ARSM) estimator:

Theorem 2 (ARSM estimator)**.** *The gradient of* $\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{z\sim\text{Cat}(\sigma(\boldsymbol{\phi}))}[f(z)]$ *with respect to* ϕ_c*, can be expressed as*

$$\begin{aligned}&\nabla_{\phi_c}\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{\pi}\sim\text{Dir}(\mathbf{1}_C)}\big[g_{\text{ARSM}}(\boldsymbol{\pi})_c\big],\\ &g_{\text{ARSM}}(\boldsymbol{\pi})_c := \tfrac{1}{C}\textstyle\sum_{j=1}^C g_{\text{ARS}}(\boldsymbol{\pi}, j)_c\\ &= \textstyle\sum_{j=1}^C \big[f(z^{c\leftrightharpoons j}) - \tfrac{1}{C}\sum_{m=1}^C f(z^{m\leftrightharpoons j})\big](\tfrac{1}{C} - \pi_j).\end{aligned} \tag{9}$$

Note ARSM requires $C(C-1)/2$ swaps to generate pseudo actions, the unique number of which that differ from z is between 0 and $C - 1$; a naive implementation requires $O(C^2)$ $\arg\min$ operations, which, however, is totally unnecessary, as in general it can at least be made below $O(2C)$ and hence is scalable even C is very large (*e.g.*, $C = 10,000$); please see Appendix B and the provided code for more details. Note if all pseudo actions $z^{c\leftrightharpoons j}$ are the same as the true action z, then the gradient estimates will be zeros for all ϕ_c.

Corollary 3. *When* $C = 2$*, both the ARS estimator in* (8) *and ARSM estimator in* (9) *reduce to the unbiased binary ARM estimator introduced in Yin & Zhou (2019).*

Detailed derivations and proofs are provided in Appendix A. Note for $C = 2$, Proposition 4 of Yin & Zhou (2019) shows that the ARM estimator is the AR estimator combined with

an optimal baseline that is subject to an anti-symmetric constraint. When $C > 2$, however, such type of theoretical analysis becomes very challenging for both the ARS and ARSM estimators. For example, it is even unclear how to define anti-symmetry for categorical variables. Thus in what follows we will focus on empirically evaluating the effectiveness of both ARS and ARSM for variance reduction.

3. ARSM Estimator for Multivariate, Hierarchical, and Sequential Settings

This section shows how the proposed univariate ARS and ARSM estimators can be generalized into multivariate, hierarchical, and sequential settings. We summarize ARS and ARSM (stochastic) gradient ascent for various types of categorical latent variables in Algorithms 1-3 of the Appendix.

3.1. ARSM for Multivariate Categorical Variables and Stochastic Categorical Network

We generalize the univariate AR/ARS/ARSM estimators to multivariate ones, which can backpropagate the gradient through a K dimensional vector of C-way categorical variables as $\boldsymbol{z} = (z_1, \ldots, z_K)$, where $z_k \in \{1, \ldots, C\}$. We further generalize them to backpropagate the gradient through multiple stochastic categorical layers, the tth layer of which consists of a K_t-dimensional C-way categorical vector as $\boldsymbol{z}_t = (z_{t1}, \ldots, z_{tK_t})' \in \{1, \ldots, C\}^{K_t}$. We defer all the details to Appendix C due to space constraint.

Note for categorical variables, especially in multivariate and/or hierarchical settings, the ARS/ARSM estimators may appear fairly complicated due to their variable-swapping operations. Their implementations, however, are actually relatively straightforward, as shown in Algorithms 1 and 2 of the Appendix, and the provided Python code.

3.2. ARSM for Discrete-Action Policy Optimization

In RL with a discrete action space with C possible actions, at time t, the agent with state $\boldsymbol{s}_t$ chooses action $a_t \in \{1, \ldots, C\}$ according to policy

$$\pi_{\boldsymbol{\theta}}(a_t \,|\, \boldsymbol{s}_t) := \mathrm{Cat}(a_t; \sigma(\boldsymbol{\phi}_t)), \;\; \boldsymbol{\phi}_t := \mathcal{T}_{\boldsymbol{\theta}}(\boldsymbol{s}_t),$$

where $\mathcal{T}_{\boldsymbol{\theta}}(\cdot)$ denotes a neural network parameterized by $\boldsymbol{\theta}$; the agent receives award $r(\boldsymbol{s}_t, a_t)$ at time t, and state $\boldsymbol{s}_t$ transits to state $\boldsymbol{s}_{t+1}$ according to $\mathcal{P}(\boldsymbol{s}_{t+1} \,|\, \boldsymbol{s}_t, a_t)$. With discount parameter $\gamma \in (0, 1]$, policy gradient methods optimize $\boldsymbol{\theta}$ to maximize the expected reward $J(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{P}, \pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, a_t)]$ (Sutton & Barto, 1998; Sutton et al., 2000; Peters & Schaal, 2008; Schulman et al., 2015). With $Q(\boldsymbol{s}_t, a_t) := \mathbb{E}_{\mathcal{P}, \pi_{\boldsymbol{\theta}}}[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(\boldsymbol{s}_{t'}, a_{t'})]$ denoted as the action-value functions, $\hat{Q}(\boldsymbol{s}_t, a_t) := \sum_{t'=t}^{\infty} \gamma^{t'-t} r(\boldsymbol{s}_{t'}, a_{t'})$ as their sample estimates, and $\rho_\pi(\boldsymbol{s}) := \sum_{t=0}^{\infty} \gamma^t \mathcal{P}(\boldsymbol{s}_t = \boldsymbol{s} \,|\, \boldsymbol{s}_0, \pi_{\boldsymbol{\theta}})$ as the unnormalized discounted state visitation frequency, the policy gradient via REINFORCE (Williams, 1992) can be expressed as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{a_t \sim \pi_{\boldsymbol{\theta}}(a_t | \boldsymbol{s}_t), \, \boldsymbol{s}_t \sim \rho_\pi(\boldsymbol{s})}[\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(a_t | \boldsymbol{s}_t) Q(\boldsymbol{s}_t, a_t)].$$

For variance reduction, one often subtracts state-dependent baselines $b(\boldsymbol{s}_t)$ from $\hat{Q}(\boldsymbol{s}_t, a_t)$ (Williams, 1992; Greensmith et al., 2004). In addition, several different action-dependent baselines $b(\boldsymbol{s}_t, a_t)$ have been recently proposed (Gu et al., 2017; Grathwohl et al., 2018; Wu et al., 2018; Liu et al., 2018), though their promise in appreciable variance reduction without introducing bias for policy gradient has been questioned by Tucker et al. (2018).

Distinct from all previous baseline-based variance reduction methods, in this paper, we develop both the ARS and ARSM policy gradient estimators, which use the action-value functions $Q(\boldsymbol{s}_t, a_t)$ themselves combined with pseudo actions to achieve variance reduction:

Proposition 4 (ARS/ARSM policy gradient)**.** *The policy gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ can be expressed as*

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\varpi}_t \sim \mathrm{Dir}(\mathbf{1}_C), \, \boldsymbol{s}_t \sim \rho_\pi(\boldsymbol{s})}\big[\nabla_{\boldsymbol{\theta}} \textstyle\sum_{c=1}^{C} g_{tc} \phi_{tc}\big], \quad (10)$$

where $\boldsymbol{\varpi}_t = (\varpi_{t1}, \ldots, \varpi_{tC})'$ and ϕ_{tc} is the cth element of $\boldsymbol{\phi}_t = \mathcal{T}_{\boldsymbol{\theta}}(\boldsymbol{s}_t) \in \mathbb{R}^C$; under the ARS estimator, we have

$$\begin{aligned} g_{tc} &:= f_{t\Delta}^{c \leftrightharpoons j_t}(\boldsymbol{\varpi}_t)(1 - C\varpi_{tj_t}), \\ f_{t\Delta}^{c \leftrightharpoons j_t}(\boldsymbol{\varpi}_t) &:= Q(\boldsymbol{s}_t, a_t^{c \leftrightharpoons j_t}) - \tfrac{1}{C} \textstyle\sum_{m=1}^{C} Q(\boldsymbol{s}_t, a_t^{m \leftrightharpoons j_t}), \\ a_t^{c \leftrightharpoons j_t} &:= \arg\min_{i \in \{1, \ldots, C\}} \varpi_{ti}^{c \leftrightharpoons j_t} e^{-\phi_{ti}}, \end{aligned} \quad (11)$$

where $j_t \in \{1, \ldots, C\}$ is a randomly selected reference category for time step t; under the ARSM estimator, we have

$$g_{tc} := \textstyle\sum_{j=1}^{C} f_{t\Delta}^{c \leftrightharpoons j}(\boldsymbol{\varpi}_t)(\frac{1}{C} - \varpi_{tj}). \quad (12)$$

Note as the number of unique actions among $a_t^{m \leftrightharpoons j}$ is as few as one, in which case the ARS/ARSM gradient is zero and there is no need at all to estimate the Q function, and as many as C, in which case one needs to estimate the Q function C times. Thus if the computation of estimating Q once is $O(1)$, then the worst computation for an episode that lasts T time steps before termination is $O(TC)$. Usually the number of distinct pseudo actions will decrease dramatically as the training progresses. We illustrate this in Figure 7, where we show the trace of categorical variable's entropy and number of distinct pseudo actions that differ from the true action. Examining (11) and (12) shows that the ARS/ARSM policy gradient estimator can be intuitively understood as a "try-and-see self-critic" method, which eliminates the need of constructing baselines and estimating their parameters for variance reduction. To decide the gradient direction of whether increasing the probability of action c at a given state, it compares the pseudo-action

reward $Q(\boldsymbol{s}_t, a_t^{c \leftrightharpoons j})$ with the average of all pseudo-action rewards $\{Q(\boldsymbol{s}_t, a_t^{m \leftrightharpoons j})\}_{m=1,C}$. If the current policy is very confident on taking action a_t at state $\boldsymbol{s}_t$, which means ϕ_{ta_t} dominates the other $C-1$ elements of $\boldsymbol{\phi}_t = \mathcal{T}_{\boldsymbol{\theta}}(\boldsymbol{s}_t)$, then it is very likely that $a_t^{m \leftrightharpoons j_t} = a_t$ for all m, which will lead to zero gradient at time t. On the contrary, if the current policy is uncertain about which action to choose, then more pseudo actions that are different from the true action are likely to be generated. This mechanism encourages exploration when the policy is uncertain, and balance the tradeoff of exploration and exploitation intrinsically. It also explains our empirical observations that ARS/ARSM tends to generate a large number of unique pseudo actions in the early stages of training, leading to fast convergence, and significantly reduced number once the policy becomes sufficiently certain, leading to stable performance after convergence.

4. Experimental Results

In this section, we use a toy example for illustration, demonstrate both multivariate and hierarchical settings with categorical latent variable models, and demonstrate the sequential setting with discrete-action policy optimization. Comparison of gradient variance between various algorithms can be found in Figures 1 and 3-6.

4.1. Example Results on Toy Data

To illustrate the working mechanism of the ARSM estimator, we consider learning $\boldsymbol{\phi} \in \mathbb{R}^C$ to maximize

$$\mathbb{E}_{z \sim \text{Cat}(\sigma(\boldsymbol{\phi}))}[f(z)], \quad f(z) := 0.5 + z/(CR), \tag{13}$$

where $z \in \{1, \ldots, C\}$. The optimal solution is $\sigma(\boldsymbol{\phi}) = (0, \ldots, 0, 1)$, which leads to the maximum expected reward of $0.5 + 1/R$. The larger the C and/or R are, the more challenging the optimization becomes. We first set $C = R = 30$ that are small enough to allow existing algorithms to perform reasonably well. Further increasing C or R will often fail existing algorithms and ARS, while ARSM always performs almost as good as the true gradient when used in optimization via gradient ascent. We include the results for $C = 1,000$ and $10,000$ in Figures 4 and 5 of the Appendix.

We perform an ablation study of the proposed AR, ARS, and ARSM estimators. We also make comparison to two representative low-variance estimators, including the biased Gumbel-Softmax estimator (Jang et al., 2017; Maddison et al., 2017) that applies the reparameterization trick after continuous relaxation of categorical variables, and the unbiased RELAX estimator of Grathwohl et al. (2018) that combines reparameterization and REINFORCE with an adaptively estimated baseline. We compare them in terms of the expected reward as $\sum_{c=1}^{C} \sigma(\boldsymbol{\phi})_c f(c)$, gradients for ϕ_c, probabilities $\sigma(\boldsymbol{\phi})_c$, and gradient variance. Note when $C = 2$, both ARS and ARSM reduce to the ARM estimator, which has been shown in Yin & Zhou (2019) to outperform a wide variety of estimators for binary variables, including the REBAR estimator of Tucker et al. (2017). The true gradient in this example can be computed analytically as in (3). All estimators in comparison use a single Monte Carlo sample for gradient estimation. We initialize $\phi_c = 0$ for all c and fix the gradient-ascent stepsize as one.

As shown in Figure 1, without appropriate variance reduction, both AR and REINFORCE either fail to converge or converge to a low-reward solution. We notice RELAX for $C = R = 30$ is not that stable across different runs; in this particular run, it manages to obtain a relatively high reward, but its probabilities converge towards a solution that is different from the optimum $\sigma(\boldsymbol{\phi}) = (0, \ldots, 0, 1)$. By contrast, Gumbel-Softmax, ARS, and ARSM all robustly reach probabilities close to the optimum $\sigma(\boldsymbol{\phi}) = (0, \ldots, 0, 1)$ after 5000 iterations across all random trials. The gradient variance of ARSM is about one to four magnitudes less than these of the other estimators, which helps explain why ARSM is almost identical to the true gradient in moving $\sigma(\boldsymbol{\phi})$ towards the optimum that maximizes the expected reward. The advantages of ARSM become even clearer in more complex settings where analytic gradients become intractable to compute, as shown below.

4.2. Categorical Variational Auto-Encoders

For optimization involving expectations with respect to multivariate categorical variables, we consider a variational auto-encoder (VAE) with a single categorical stochastic hidden layer. We further consider a categorical VAE with two categorical stochastic hidden layers to illustrate optimization involving expectations with respect to hierarchical multivariate categorical variables.

Following Jang et al. (2017), we consider a VAE with a categorical hidden layer to model D-dimensional binary observations. The decoder parameterized by $\boldsymbol{\theta}$ is expressed as $p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z}) = \prod_{i=1}^{D} p_{\boldsymbol{\theta}}(x_i \,|\, \boldsymbol{z})$, where $\boldsymbol{z} \in \{1, \ldots, C\}^K$ is a K-dimensional C-way categorical vector and $p_{\boldsymbol{\theta}}(x_i \,|\, \boldsymbol{z})$ is Bernoulli distributed. The encoder parameterized by $\boldsymbol{\phi}$ is expressed as $q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x}) = \prod_{k=1}^{K} q_{\boldsymbol{\phi}}(z_k \,|\, \boldsymbol{x})$. We set the prior as $p(z_k = c) = 1/C$ for all c and k. For optimization, we maximize the evidence lower bound (ELBO) as

$$\mathcal{L}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x})}\big[\ln \tfrac{p_{\boldsymbol{\theta}}(\boldsymbol{x} \,|\, \boldsymbol{z}) p(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z} \,|\, \boldsymbol{x})}\big]. \tag{14}$$

We also consider a two-categorical-hidden-layer VAE, whose encoder and decoder are constructed as

$$q_{\boldsymbol{\phi}_{1:2}}(\boldsymbol{z}_1, \boldsymbol{z}_2 \,|\, \boldsymbol{x}) = q_{\boldsymbol{\phi}_1}(\boldsymbol{z}_1 \,|\, \boldsymbol{x}) q_{\boldsymbol{\phi}_2}(\boldsymbol{z}_2 \,|\, \boldsymbol{z}_1),$$
$$p_{\boldsymbol{\theta}_{1:2}}(\boldsymbol{x} \,|\, \boldsymbol{z}_1, \boldsymbol{z}_2) = p_{\boldsymbol{\theta}_1}(\boldsymbol{x} \,|\, \boldsymbol{z}_1) p_{\boldsymbol{\theta}_2}(\boldsymbol{z}_1 \,|\, \boldsymbol{z}_2),$$

where $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \{1, \ldots, C\}^K$. The ELBO is expressed as

$$\mathcal{L}(\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}_{1:2}}(\boldsymbol{z}_1, \boldsymbol{z}_2 \,|\, \boldsymbol{x})}\Big[\ln \tfrac{p_{\boldsymbol{\theta}_1}(\boldsymbol{x} \,|\, \boldsymbol{z}_1) p_{\boldsymbol{\theta}_2}(\boldsymbol{z}_1 \,|\, \boldsymbol{z}_2) p(\boldsymbol{z}_2)}{q_{\boldsymbol{\phi}_1}(\boldsymbol{z}_1 \,|\, \boldsymbol{x}) q_{\boldsymbol{\phi}_2}(\boldsymbol{z}_2 \,|\, \boldsymbol{z}_1)}\Big]. \tag{15}$$

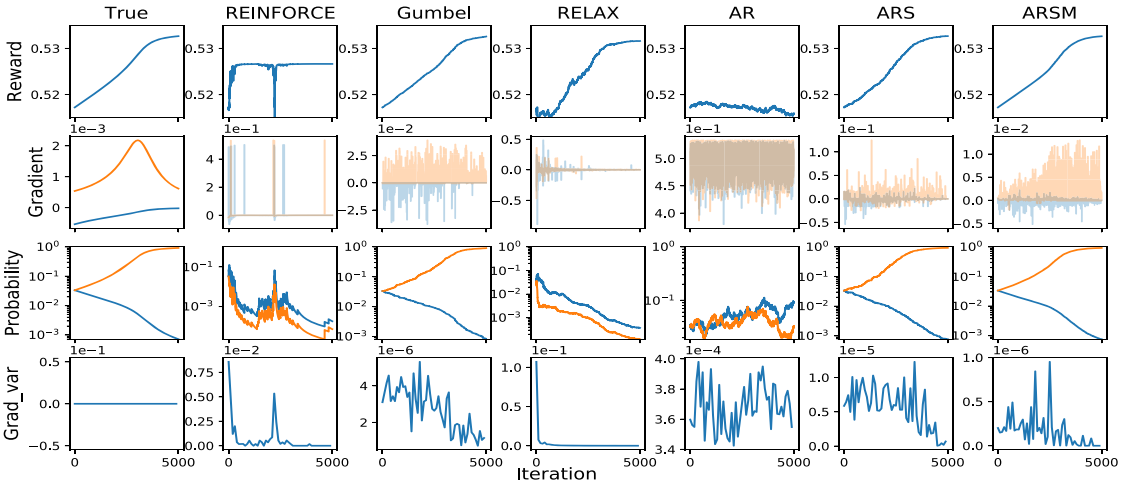

Figure 1. Comparison of a variety of gradient estimators in maximizing (13). The optimal solution is $\sigma(\phi) = (0, \dots, 1)$, which means $z = C$ with probability one. The reward is computed analytically by $\mathbb{E}_{z\sim\text{Cat}(\sigma(\phi))}[f(z)]$ with maximum as 0.533. Rows 1, 2, and 3 show the trace plots of reward $\mathbb{E}[f(z)]$, the gradients with respect to ϕ_1 and ϕ_C, and the probabilities $\sigma(\phi)_1$ and $\sigma(\phi)_C$, respectively. Row 4 shows the gradient variance estimation with 100 Monte Carlo samples at each iteration, averaged over categories 1 to C.

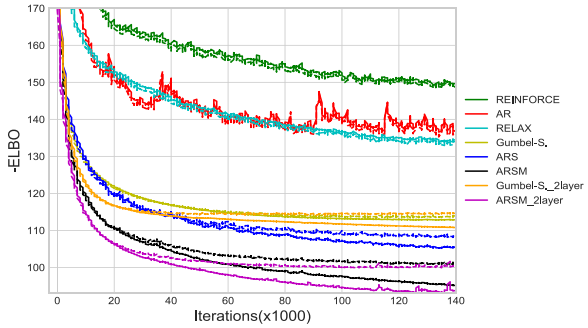

Figure 2. Plots of negative ELBOs (nats) on binarized MNIST against training iterations (analogous ones against times are shown in Figure 8). The solid and dash lines correspond to the training and testing respectively (best viewed in color).

For both categorical VAEs, we set $K = 20$ and $C = 10$. We train them on a binarized MNIST dataset as in van den Oord et al. (2017) by thresholding each pixel value at 0.5. Implementations of the VAEs with one and two categorical hidden layers are summarized in Algorithms 1 and 2, respectively; see the provided code for more details.

We consider the AR, ARS, and ARSM estimators, and include the REINFORCE (Williams, 1992), Gumbel-Softmax (Jang et al., 2017), and RELAX (Grathwohl et al., 2018) estimators for comparison. We note that Jang et al. (2017) has already shown Gumbel-Softmax outperforms a wide variety of previously proposed estimators; see Jang et al. (2017) and the references therein for more details.

We present the trace plots of the training and validation negative ELBOs in Figure 2 and gradient variance in Figure 6. The numerical values are summarized in Table 1. We use the Gumbel-Softmax code [1] to obtain the results of the VAE with a single categorical hidden layer, and modify it with our best effort for the VAE with two categorical hidden layers; we modify the RELAX code [2] with our best effort to allow it to optimize VAE with a single categorical hidden layer. For the single-hidden-layer VAE, we connect its latent categorical layer $\boldsymbol{z}$ and observation layer $\boldsymbol{x}$ with two nonlinear deterministic layers; for the two-hidden-layer VAE, we add an additional categorical hidden layer $\boldsymbol{z}_2$ that is linearly connected to the first one. See Table 3 of the Appendix for detailed network architectures. In our experiments, all methods use exactly the same network architectures and data, set the mini-batch size as 200, and are trained by the Adam optimizer (Kingma & Ba, 2014), whose learning rate is selected from $\{5, 1, 0.5\} \times 10^{-4}$ using the validation set.

The results in Table 1 and Figure 2 clearly show that for optimizing the single-categorical-hidden-layer VAE, both ARS and ARSM estimators outperform all the other ones in both training and testing ELBOs. In particular, ARSM outperforms all the other estimators by a large margin. We also consider Gumbel-Softmax by computing its gradient with 25 Monte Carlo samples, making it run as fast as the provided ARSM code does per iteration. In this case, both algorithms take similar time but ARSM achieves $-$ELBOs for the training and testing sets as 94.6 and 100.6, respectively, while those of Gumbel-Softmax are 102.5 and 103.6, respectively. The performance gain of ARSM can be explained by both its unbiasedness and a clearly lower variance exhibited by its gradient estimates in comparison to all the other estimators, as shown in Figure 6 of the Appendix. The results on the two-categorical-hidden-layer VAE, which adds a linear categorical layer on top of the single-categorical-hidden-layer VAE, also suggest that ARSM can further improve its

[1] https://github.com/ericjang/gumbel-softmax

[2] https://github.com/duvenaud/relax

Table 1. Comparison of training and testing negative ELBOs (nats) on binarized MNIST between ARSM and various gradient estimators.

Gradient estimator	REINFORCE	RELAX	ST Gumbel-S.	AR	ARS	ARSM	Gumbel-S.-2layer	ARSM-2layer
−ELBO (Training)	142.4	129.1	112.7	135.8	101.5	**94.6**	111.4	**92.5**
−ELBO (Testing)	141.3	130.3	113.4	136.6	106.7	**100.6**	113.8	**99.9**

Table 2. Comparison of the test negative log-likelihoods between ARSM and various gradient estimators in Jang et al. (2017), for the MNIST conditional distribution estimation benchmark task.

Gradient estimator	ARSM	ST	Gumbel-S.	MuProp
$-\log p(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$	**58.3** $\pm$ **0.2**	61.8	59.7	63.0

performance by adding more stochastic hidden layers and clearly outperforms the biased Gumbel-Softmax estimator.

4.3. Maximum Likelihood Estimation for a Stochastic Categorical Network

Denoting $\boldsymbol{x}_l, \boldsymbol{x}_u \in \mathbb{R}^{392}$ as the lower and upper halves of an MNIST digit, respectively, we consider a standard benchmark task of estimating the conditional distribution $p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$ (Raiko et al., 2014; Bengio et al., 2013; Gu et al., 2016; Jang et al., 2017; Tucker et al., 2017). We consider a stochastic categorical network with two stochastic categorical hidden layers, expressed as

$$\begin{aligned}
&\boldsymbol{x}_l \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_0}(\boldsymbol{b}_1))),\\
&\boldsymbol{b}_1 \sim \textstyle\prod_{c=1}^{20} \text{Cat}(b_{1c}; \sigma(\mathcal{T}_{\boldsymbol{\theta}_1}(\boldsymbol{b}_2)_{[10(c-1)+(1:10)]})),\\
&\boldsymbol{b}_2 \sim \textstyle\prod_{c=1}^{20} \text{Cat}(b_{2c}; \sigma(\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u)_{[10(c-1)+(1:10)]})),
\end{aligned}$$

where both $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are 20-dimensional 10-way categorical variables, $\mathcal{T}_{\boldsymbol{\theta}}(\cdot)$ denotes linear transform, $\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u)_{[10(c-1)+(1:10)]}$ is a 10-dimensional vector consisting of elements $10(c-1)+1$ to $10c$ of $\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u) \in \mathbb{R}^{200}$, $\mathcal{T}_{\boldsymbol{\theta}_1}(\boldsymbol{b}_2) \in \mathbb{R}^{200}$, and $\mathcal{T}_{\boldsymbol{\theta}_0} \in \mathbb{R}^{392}$. Thus we can consider the network structure as 392-200-200-392, making the results directly comparable with these in Jang et al. (2017) for stochastic categorical network. We approximate $\log p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$ with K Monte Carlo samples as

$$\log \tfrac{1}{K} \textstyle\sum_{k=1}^{K} \text{Bernoulli}(\boldsymbol{x}_l; \sigma(\mathcal{T}_{\boldsymbol{\theta}_0}(\boldsymbol{b}_1^{(k)}))), \qquad (16)$$

where $\boldsymbol{b}_1^{(k)} \sim \prod_{c=1}^{20} \text{Cat}(b_{1c}^{(k)}; \sigma(\mathcal{T}_{\boldsymbol{\theta}_1}(\boldsymbol{b}_2^{(k)})_{[10(c-1)+(1:10)]}))$, $\boldsymbol{b}_2^{(k)} \sim \prod_{c=1}^{20} \text{Cat}(b_{2c}^{(k)}; \sigma(\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u)_{[10(c-1)+(1:10)]}))$. 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 marginal likelihood. We use Adam (Kingma & Ba, 2014), with the learning rate set as 10^{-4}, mini-batch size as 100, and number of training epochs as 2000. Given the inferred point estimate of $\boldsymbol{\theta}_{0:2}$, we evaluate the accuracy of conditional density estimation by estimating the negative log-likelihood $-\log p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$ using (16), averaging over the test set with $K = 1000$.

As shown in Table 2, optimizing a stochastic categorical network with the ARSM estimator achieves the lowest test negative log-likelihood, outperforming all previously proposed gradient estimators on the same structured stochastic networks, including straight through (ST) (Bengio et al., 2013) and ST Gumbel-softmax (Jang et al., 2017) that are biased, and MuProp (Gu et al., 2016) that is unbiased.

4.4. Discrete-Action Policy Optimization

The key of applying the ARSM policy gradient shown in (12) is to provide, under the current policy $\pi_{\boldsymbol{\theta}}$, the action-value functions' sample estimates $\hat{Q}(\boldsymbol{s}_t, a_t) := \sum_{t'=t}^{\infty} \gamma^{t'-t} r(\boldsymbol{s}_{t'}, a_{t'})$ for all unique values in $\{a_t^{c \rightleftharpoons j}\}_{c,j}$. Thus ARSM is somewhat related to the *vine* method proposed in Schulman et al. (2015), which defines a heuristic rollout policy that chooses a subset of the states along the true trajectory as the "rollout set," samples K pseudo actions uniformly at random from the discrete-action set at each state of the rollout set, and performs a single rollout for each state-pseudo-action-pair to estimate its action-value function Q. ARSM chooses its rollout set in the same manner, but is distinct from the *vine* method in having a rigorously derived rollout policy: it swaps the elements of $\boldsymbol{\varpi}_t \sim \text{Dir}(\mathbf{1}_C)$ to generate pseudo actions if state $\boldsymbol{s}_t$ belongs to the rollout set; the number of unique pseudo actions that are different from the true action a_t is a random number, which is positively related to the uncertainty of the policy and hence often negatively related to its convergence; and a single rollout is then performed for each of these unique pseudo actions to estimate its Q.

As ARSM requires the estimation of Q function for each unique state-pseudo-action pair using Monte Carlo rollout, it could have high computational complexity if (1) the number of unique pseudo actions is large, and (2) each rollout takes many expensive steps (interactions with the environments) before termination. However, there exist ready solutions and many potential ones. As given a true trajectory, all the state-pseudo-action rollouts of ARSM can be independently simulated and hence all pseudo-action related Q's can be estimated in an embarrassingly parallel manner. Furthermore, in addition to Monte Carlo estimation, we can potentially adapt for ARSM a wide variety of off-the-shelf action-value function estimation methods (Sutton & Barto, 1998), to either accelerate the estimation of Q or further reduce the variance (though possibly at the expense of introducing bias). In our experiment, for simplicity and clarity, we choose to use Monte Carlo estimation to obtain $\hat{Q}$ for

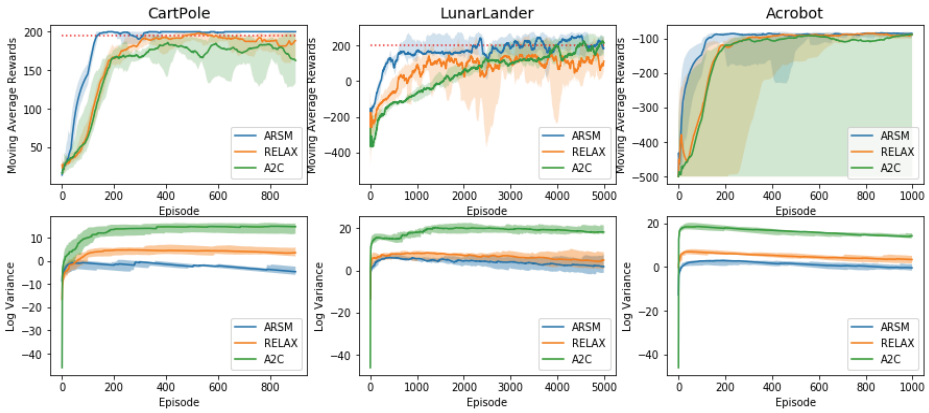

Figure 3. Top row: Moving average reward curves. *Bottom row:* Log-variance of gradient estimator. In each plot, the solid lines are the median value of ten independent runs (ten different random seeds for random initializations). The opaque bars are 10th and 90th percentiles. Dashed straight lines in Cart Pole and Lunar Lander represent task-completion criteria.

both the true trajectory and all state-pseudo-action rollouts. The results for RELAX and A2C are obtained by running the code provided by Grathwohl et al. (2018)[3].

We apply the ARSM policy gradient to three representative RL tasks with discrete actions, including the Cart Pole, Acrobot, and Lunar Lander environments provided by OpenAI Gym (Brockman et al., 2016), and compare it with advantage actor-critic algorithm (A2C) (Sutton et al., 2000) and RELAX (Grathwohl et al., 2018). We report the moving-average rewards and the estimated log-variance of the gradient estimator at every episode; for each episode, the reward score is obtained by running the updated policy on a new random environment; and the variance is obtained by first applying exponential moving averages to the first and second moments of each neural network parameter with decay 0.99, and then taking the average of the estimated variances of all neural network parameters.

Shown in Figure 3 are the mean rewards over the last 100 steps; the opaque bar indicates 10th and 90th percentiles obtained by ten independent runs for each method (using 10 different random seeds for random initializations); the solid line is the median value of these ten independent runs. ARSM outperform both baselines in all three tasks in terms of stability, moving average rewards, and log-variance of gradient estimator. All methods are cross validated by optimizers $\{$Adam Optimizer, RMSProp Optimizer$\}$ and learning rates $\{1, 3, 10, 30\} \times 10^{-3}$. Both the policy and critic networks for A2C and RELAX have two 10-unit hidden layers with ReLU activation functions (Nair & Hinton, 2010). The discount factor γ is 0.99 and entropy term is 0.01. The policy network of ARSM is the same as that of A2C and RELAX, and the maximum number of allowed state-pseudo-action rollouts of ARSM is set as 16, 64, and 1024 for Cart Pole, Acrobot, and Lunar Lander, respectively; see Algorithm 3 and the provided code for more details. Using our current implementation that has not been optimized to fully take the advantage of parallel computing, to finish the number of episodes as in Figure 3, ARSM on average takes 677, 425, and 19050 seconds for CartPole, Acrobot, and LunarLander, respectively. For comparison, for these three tasks, RELAX on average takes 139, 172, and 3493 seconds and A2C on average takes 92, 120, and 2708 seconds.

[3] https://github.com/wgrathwohl/BackpropThroughTheVoidRL

5. Conclusion

To backpropagate the gradients through categorical stochastic layers, we propose the augment-REINFORCE-swap-merge (ARSM) estimator that is unbiased and exhibits low variance. The performance of ARSM is almost identical to that of the true gradient when used for optimization involving a C-way categorical variable, even when C is very large (such as $C = 10,000$). For multiple C-way categorical variables organized into a single stochastic layer, multiple stochastic layers, or a sequential setting, the ARSM estimator clearly outperforms state-of-the-art methods, as shown in our experimental results for both categorical latent variable models and discrete-action policy optimization. We attribute the outstanding performance of ARSM to both its unbiasedness and its ability to control variance by simply combing its reward function with randomly generated pseudo actions, where the number of unique pseudo actions is positively related to the uncertainties of categorical distributions and hence negatively correlated to how well the optimization algorithm has converged; there is no more need to construct separate baselines and estimate their parameters, which also help make the optimization more robust. Some natural extensions of the proposed ARSM estimator include applying it to reinforcement learning with high-dimensional discrete-action spaces or multiple discrete-action agents, and various tasks in natural language processing such as sentence generation and machine translation.

Acknowledgements

This research was supported in part by Award IIS-1812699 from the U.S. National Science Foundation and the McCombs Research Excellence Grant. The authors acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research, and the computational support of Texas Advanced Computing Center.

References

Andriyash, E., Vahdat, A., and Macready, B. Improved gradient-based optimization over discrete distributions. *arXiv preprint arXiv:1810.00116*, 2018.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Casella, G. and Robert, C. P. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

Cheng, P., Liu, C., Li, C., Shen, D., Henao, R., and Carin, L. Straight-through estimator as projected Wasserstein gradient flow. In *NeurIPS 2018 Bayesian Deep Learning Workshop*, 2018.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. In *ICLR*, 2018.

Greensmith, E., Bartlett, P. L., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. *J. Mach. Learn. Res.*, 5(Nov):1471–1530, 2004.

Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. Deep autoregressive networks. In *ICML*, pp. 1242–1250, 2014.

Gu, S., Levine, S., Sutskever, I., and Mnih, A. MuProp: Unbiased backpropagation for stochastic neural networks. In *ICLR*, 2016.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-Prop: Sample-efficient policy gradient with an off-policy critic. In *ICLR*, 2017.

Hinton, G. Neural networks for machine learning coursera video lectures - Geoffrey Hinton. 2012.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14): 1–45, 2017.

Liu, H., Feng, Y., Mao, Y., Zhou, D., Peng, J., and Liu, Q. Action-dependent control variates for policy optimization via Stein identity. In *ICLR*, 2018.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

Maddison, C. J., Mnih, A., and Teh, Y. W. The Concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

McFadden, D. Conditional Logit Analysis of Qualitative Choice Behavior. In Zarembka, P. (ed.), *Frontiers in Econometrics*, pp. 105–142. Academic Press, New York, 1974.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. In *ICML*, pp. 1791–1799, 2014.

Mnih, A. and Rezende, D. J. Variational inference for Monte Carlo objectives. *arXiv preprint arXiv:1602.06725*, 2016.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *ICML*, pp. 1928–1937, 2016.

Naesseth, C., Ruiz, F., Linderman, S., and Blei, D. Reparameterization gradients through acceptance-rejection sampling algorithms. In *AISTATS*, pp. 489–498, 2017.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In *ICML*, pp. 807–814, 2010.

Owen, A. B. *Monte Carlo Theory, Methods and Examples*, chapter 8 Variance Reduction. 2013.

Paisley, J., Blei, D. M., and Jordan, M. I. Variational Bayesian inference with stochastic search. In *ICML*, pp. 1363–1370, 2012.

Peters, J. and Schaal, S. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

Raiko, T., Berglund, M., Alain, G., and Dinh, L. Techniques for learning binary stochastic feedforward neural networks. *arXiv preprint arXiv:1406.2989*, 2014.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *AISTATS*, pp. 814–822, 2014.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pp. 1278–1286, 2014.

Ross, S. M. *Introduction to Probability Models*. Academic Press, 10th edition, 2006.

Ruiz, F. J. R., Titsias, M. K., and Blei, D. M. The generalized reparameterization gradient. In *NIPS*, pp. 460–468, 2016.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *ICML*, pp. 1889–1897, 2015.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. 1998.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pp. 1057–1063, 2000.

Tanner, M. A. and Wong, W. H. The calculation of posterior distributions by data augmentation. *J. Amer. Statist. Assoc.*, 82(398):528–540, 1987.

Titsias, M. K. and Lázaro-Gredilla, M. Local expectation gradients for black box variational inference. In *NIPS*, pp. 2638–2646, 2015.

Train, K. E. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2nd edition, 2009.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In *NIPS*, pp. 2624–2633, 2017.

Tucker, G., Bhupatiraju, S., Gu, S., Turner, R., Ghahramani, Z., and Levine, S. The mirage of action-dependent baselines in reinforcement learning. In *ICML*, pp. 5015–5024, 2018.

van den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *NIPS*, pp. 6306–6315, 2017.

Van Dyk, D. A. and Meng, X.-L. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

Weaver, L. and Tao, N. The optimal reward baseline for gradient-based reinforcement learning. In *UAI*, pp. 538–545, 2001.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pp. 5–32. Springer, 1992.

Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A. M., Kakade, S., Mordatch, I., and Abbeel, P. Variance reduction for policy gradient with action-dependent factorized baselines. In *ICLR*, 2018.

Yin, M. and Zhou, M. ARM: Augment-REINFORCE-merge gradient for stochastic binary networks. In *ICLR*, 2019.

Zhang, Q. and Zhou, M. Nonparametric Bayesian Lomax delegate racing for survival analysis with competing risks. In *NeurIPS*, pp. 5002–5013, 2018.

Zhou, M. Beta-negative binomial process and exchangeable random partitions for mixed-membership modeling. In *NIPS*, pp. 3455–3463, 2014.

Zhou, M. and Carin, L. Negative binomial process count and mixture modeling. *arXiv preprint arXiv:1209.3442v1*, 2012.