ARM: Augment-REINFORCE-merge gradient for stochastic binary networks

Mingzhang Yin, Mingyuan Zhou

Department of Statistics and Data Sciences
The University of Texas at Austin

October 12, 2018

Discrete latent variables

- Discrete latent variables are widely used in mixture models, mixed membership model, variable selection, and sigmoid belief net etc.
- A common task is to optimize

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

- This objective includes
  - Maximizing the marginal likelihood of a hierarchical Bayesian model
  - Maximizing the evidence lower bound (ELBO) in variational inference
  - Maximizing the expected reward in reinforcement learning
  - ...

Unbiased gradient estimators

Reparameterization
If $\nabla_{\boldsymbol{z}} f(\boldsymbol{z})$ is tractable to compute and $\boldsymbol{z} \sim q_\phi(\boldsymbol{z})$ can be generated via reparameterization as $\boldsymbol{z} = \mathcal{T}_\phi(\boldsymbol{\epsilon}),\ \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, then one may apply the reparameterization trick

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

REINFORCE (score-function estimator)
If $\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z})}[\nabla_\phi f(\boldsymbol{z})] = 0$, using the score function $\nabla_\phi \log q_\phi(\boldsymbol{z}) = \nabla_\phi q_\phi(\boldsymbol{z})/q_\phi(\boldsymbol{z})$, one may use REINFORCE as

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

Problems

However, neither estimator is problem free:

- The reparameterization trick requires $f(\boldsymbol{z})$ to be differentiable and cannot be applied to discrete $\boldsymbol{z}$
- REINFORCE suffers from high Monte Carlo estimation variance

Gradient estimation for discrete latent variables

For discrete latent variable $\boldsymbol{z}$, to compute the gradient of $\mathcal{E}(\phi) = \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z})}[f(\boldsymbol{z})]$, existing solutions include

- Biased but low-variance gradient estimator via a continuous relaxation of discrete random variables
    - Gumbel-softmax trick, $\operatorname{argmax}(g_i + \log \pi_i)$
- Variance reduction by adding control variates (a.k.a. baselines)

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

where $\mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z})}[b(\boldsymbol{z})] = 0$, for example

- $b(\boldsymbol{z}) = m\nabla_\phi \log q_\phi(\boldsymbol{z})$
- Learnable control variate function parameterized with neural network (REBAR/RELAX)

ARM gradient

Algorithm 1: ARM gradient for V-dimensional binary latent vector

```
input  : Bernoulli distribution $\{q_{\phi_v}(z_v)\}_{v=1:V}$ with probability $\{\sigma(\phi_v)\}_{v=1:V}$, target $f(\boldsymbol{z})$;
         $\boldsymbol{z} = (z_1, \cdots, z_V)$, $\boldsymbol{\phi} = (\phi_1, \cdots, \phi_V)$
output : $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ that maximize $\mathcal{E}(\boldsymbol{\phi}, \boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{z} \sim \prod_{v=1}^{V} q_{\phi_v}(z_v)}[f(\boldsymbol{z}; \boldsymbol{\psi})]$

Initialize $\boldsymbol{\phi}$, $\boldsymbol{\psi}$ randomly;
while not converged do
    Sample $z_v \sim \text{Bernoulli}(\sigma(\phi_v))$ for $v = 1, \cdots, V$ ;
    sample $u_v \sim \text{Uniform}(0,1)$ for $v = 1, \cdots, V$, $\boldsymbol{u} = (u_1, \cdots, u_V)$ ;
    $g_{\boldsymbol{\psi}} = \nabla_{\boldsymbol{\psi}} f(\boldsymbol{z}; \boldsymbol{\psi})$ ;
    $f_{\Delta}(\boldsymbol{u}, \boldsymbol{\phi}) = f\big(\mathbf{1}_{[\boldsymbol{u} > \sigma(-\boldsymbol{\phi})]}\big) - f\big(\mathbf{1}_{[\boldsymbol{u} < \sigma(\boldsymbol{\phi})]}\big)$ ;
    $g_{\boldsymbol{\phi}} = f_{\Delta}(\boldsymbol{u}, \boldsymbol{\phi})(\boldsymbol{u} - \frac{1}{2})$
    $\boldsymbol{\phi} = \boldsymbol{\phi} + \rho_t g_{\boldsymbol{\phi}}, \quad \boldsymbol{\psi} = \boldsymbol{\psi} + \eta_t g_{\boldsymbol{\psi}}$    with step-size $\rho_t$, $\eta_t$
end
```

Augmentation of binary random variable

Bernoulli random variable $z \sim \text{Bernoulli}(\sigma(\phi))$ can be reparameterized by racing two augmented exponential random variables as

$$z = \mathbf{1}_{[\epsilon_1 < \epsilon_2 e^{\phi}]},\ \epsilon_1 \sim \text{Exp}(1),\ \epsilon_2 \sim \text{Exp}(1).$$

The objective can be rewritten with respect to exponential random variables as

$$\mathcal{E}(\phi) = \mathbb{E}_{z \sim \text{Bernoulli}(\sigma(\phi))}[f(z)] = \mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)}[f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]})].$$

REINFORCE in augmented space

Use REINFORCE with $t_1 = \epsilon_1 e^{-\phi} \sim \text{Exp}(e^{\phi})$ we have

$$\begin{aligned}
\nabla_\phi \mathcal{E}(\phi) &= \mathbb{E}_{t_1 \sim \text{Exp}(e^{\phi}),\ \epsilon_2 \sim \text{Exp}(1)} [f(\mathbf{1}_{[t_1 < \epsilon_2]}) \nabla_\phi \log \text{Exp}(t_1; e^{\phi})] \\
&= \mathbb{E}_{t_1 \sim \text{Exp}(e^{\phi}),\ \epsilon_2 \sim \text{Exp}(1)} [f(\mathbf{1}_{[t_1 < \epsilon_2]})(1 - t_1 e^{\phi})] \\
&= \mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)} [f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]})(1 - \epsilon_1)]
\end{aligned}$$

Similarly with $t_2 = \epsilon_2 e^{\phi} \sim \text{Exp}(e^{-\phi})$

$$\begin{aligned}
\nabla_\phi \mathcal{E}(\phi) &= \mathbb{E}_{\epsilon_1 \sim \text{Exp}(1),\ t_2 \sim \text{Exp}(e^{-\phi})} [f(\mathbf{1}_{[\epsilon_1 < t_2]}) \nabla_\phi \log \text{Exp}(t_2; e^{-\phi})] \\
&= -\mathbb{E}_{\epsilon_1 \sim \text{Exp}(1),\ t_2 \sim \text{Exp}(e^{-\phi})} [f(\mathbf{1}_{[\epsilon_1 < t_2]})(1 - t_2 e^{-\phi})] \\
&= -\mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)} [f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]})(1 - \epsilon_2)]
\end{aligned}$$

Merge the gradients

Swapping the indices of the two *iid* standard exponential random variables , the gradient $\nabla_\phi \mathcal{E}(\phi)$ can be equivalently expressed as

$$\nabla_\phi \mathcal{E}(\phi) = -\mathbb{E}_{\epsilon_1,\epsilon_2 \overset{iid}{\sim} \text{Exp}(1)} [f(\mathbf{1}_{[\epsilon_2 e^{-\phi} < \epsilon_1]})(1 - \epsilon_1)].$$

Taking the average leads to

$$\nabla_\phi \mathcal{E}(\phi) = \mathbb{E}_{\epsilon_1,\epsilon_2 \overset{iid}{\sim} \text{Exp}(1)} \left[\left(f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]}) - f(\mathbf{1}_{[\epsilon_2 e^{-\phi} < \epsilon_1]}) \right) (1/2 - \epsilon_1/2) \right]$$

Merge the gradients

Letting $\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)$ is the same in distribution as letting

$$\epsilon_1 = \epsilon u,\ \epsilon_2 = \epsilon(1-u), \quad \text{where } u \sim \text{Uniform}(0,1),\ \epsilon \sim \text{Gamma}(2,1)$$

Marginalizing Gamma R.V. out reduces variance by the law of total variance

$$\begin{aligned}\nabla_\phi \mathcal{E}(\phi) &= \mathbb{E}_{u\sim \text{U}(0,1),\ \epsilon\sim\text{Ga}(2,1)}\left[\left(f(\mathbf{1}_{[u>\sigma(-\phi)]}) - f(\mathbf{1}_{[u<\sigma(\phi)]})\right)(\epsilon u/2 - 1/2)\right], \\ &= \mathbb{E}_{u\sim\text{Uniform}(0,1)}\left[\left(f(\mathbf{1}_{[u>\sigma(-\phi)]}) - f(\mathbf{1}_{[u<\sigma(\phi)]})\right)(u-1/2)\right]\end{aligned}$$

Multivariate generalization

Apply the univariate result to the inner expectation

$$\begin{aligned}\nabla_{\phi_v}\mathcal{E}(\phi) &= \mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\left\{\nabla_{\phi_v}\mathbb{E}_{z_v\sim\text{Bernoulli}(\sigma(\phi_v))}[f(\boldsymbol{z})]\right\}\\ &= \mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\Big\{\mathbb{E}_{u_v\sim\text{Uniform}(0,1)}\Big[(u_v-1/2)\\ &\qquad\times\left(f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v>\sigma(-\phi_v)]})-f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v<\sigma(\phi_v)]})\right)\Big]\Big\}.\end{aligned}$$

then sample $\boldsymbol{z}_{\setminus v}$ as $\mathbf{1}_{[\boldsymbol{u}_{\setminus v}<\sigma(\phi_{\setminus v})]}$ or as $\boldsymbol{z}_{\setminus v}=\mathbf{1}_{[\boldsymbol{u}_{\setminus v}>\sigma(-\phi_{\setminus v})]}$,

$$\begin{aligned}\nabla_{\phi_v}\mathcal{E}(\phi) &= \mathbb{E}_{u_v\sim\text{Uniform}(0,1)}\Big\{(u_v-1/2)\ \mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\Big[\\ &\qquad f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v>\sigma(-\phi_v)]})-f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v<\sigma(\phi_v)]})\Big]\Big\}\\ &= \mathbb{E}_{\boldsymbol{u}\sim\prod_{v=1}^{V}\text{Uniform}(u_v;0,1)}\left[(u_v-1/2)\left(f(\mathbf{1}_{[\boldsymbol{u}>\sigma(-\phi)]})-f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\phi)]})\right)\right]\end{aligned}$$

Antithetic Sampling

The merging step is equivalent to the antithetic sampling

$$\begin{aligned}\nabla_{\phi_v}\mathcal{E}(\phi) &= \mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\{\nabla_{\phi_v}\mathbb{E}_{z_v\sim\text{Bernoulli}(\sigma(\phi_v))}[f(\boldsymbol{z})]\} \\ &= \mathbb{E}_{\boldsymbol{u}\sim\prod_{v=1}^{V}\text{Uniform}(u_v;0,1)}\left[(1-2u_v)f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\phi)]})\right],\end{aligned}$$

With the antithetic sample $\tilde{\boldsymbol{u}} = 1 - \boldsymbol{u}$

$$\begin{aligned}\nabla_{\phi}\mathcal{E}(\phi) =& \mathbb{E}_{\boldsymbol{u}\sim\prod_v\text{Uniform}(0,1)}[f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\phi)]})(1/2-\boldsymbol{u})] \\ &+ \mathbb{E}_{\tilde{\boldsymbol{u}}\sim\prod_v\text{Uniform}(0,1)}[f(\mathbf{1}_{[\tilde{\boldsymbol{u}}<\sigma(\phi)]})(1/2-\tilde{\boldsymbol{u}})] \\ =& \mathbb{E}_{\boldsymbol{u}\sim\prod_v\text{Uniform}(0,1)}[f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\phi)]})(1/2-\boldsymbol{u}) + f(\mathbf{1}_{[\tilde{\boldsymbol{u}}<\sigma(\phi)]})(1/2-\tilde{\boldsymbol{u}})]\end{aligned}$$

Relation to control variates

Proposition

Let $\mathcal{B} = \{b(\boldsymbol{u}) : b(\boldsymbol{u}) = -b(1-\boldsymbol{u}), \}$*, and* $g_{AR}(\boldsymbol{u}) = f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\phi)]})(1-2\boldsymbol{u})$*, then with*

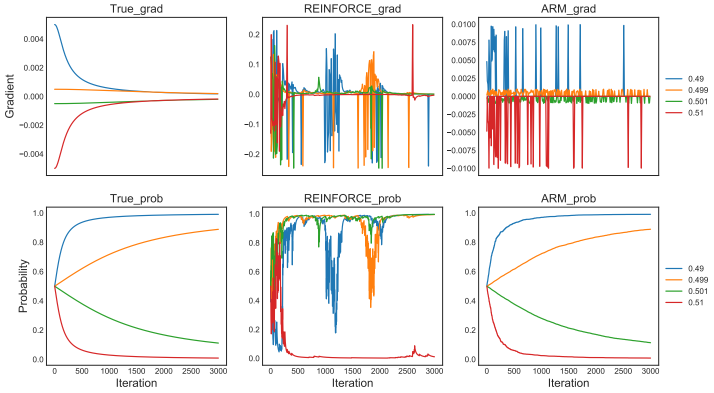
$$b^*(\boldsymbol{u}) = \underset{b(\boldsymbol{u})\in\mathcal{B}}{\arg\min}\, Var[g_{AR,v}(\boldsymbol{u}) - b_v(\boldsymbol{u})]$$

we have $g_{ARM}(\boldsymbol{u}) = g_{AR}(\boldsymbol{u}) - b^*(\boldsymbol{u})$ *with guaranteed variance reduction.*

The common score baseline $b(\boldsymbol{u}) = m(1-2\boldsymbol{u}) \in \mathcal{B}$, which means the variance reduction of ARM gradient is better than score baseline even with optimal m.

A simple example

Learning ϕ to maximize $\mathcal{E}(\phi) = \mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[(z-p_0)^2]$, where $p_0 \in \{0.49, 0.499, 0.501, 0.51\}$



A simple example

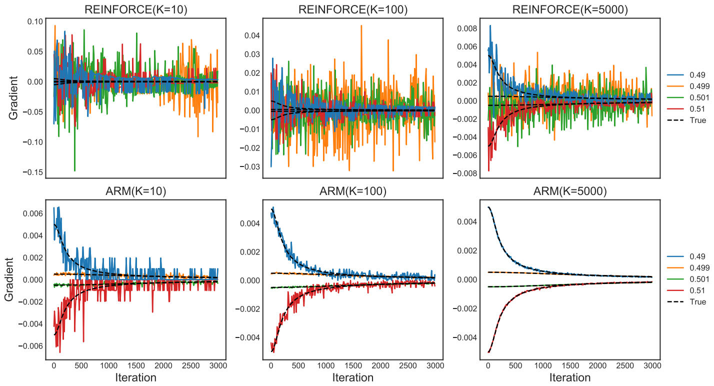


Figure: Estimation of the true gradient at each iteration using $K > 1$ Monte Carlo samples, using REINFORCE, shown in the top row, or ARM, shown in the bottom row. The ARM estimator exhibits significant lower variance given the same number of Monte Carlo samples.

Multiple discrete stochastic layers

- A latent variable model with multiple stochastic hidden layers is

$$\boldsymbol{x} \sim p_{\boldsymbol{\theta}_0}(\boldsymbol{x} \mid \boldsymbol{b}_1),\ \ \boldsymbol{b}_1 \sim p_{\boldsymbol{\theta}_1}(\boldsymbol{b}_1 \mid \boldsymbol{b}_2), \dots, \boldsymbol{b}_T \sim p_{\boldsymbol{\theta}_T}(\boldsymbol{b}_T),$$

- The joint likelihood is

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

VAE with multiple discrete stochastic layers

- VAE optimizes ELBO as $\mathbb{E}_{q_{\boldsymbol{w}}(\boldsymbol{b}|\boldsymbol{x})} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{b})p(\boldsymbol{b})}{q_{\boldsymbol{w}}(\boldsymbol{b}|\boldsymbol{x})}$
- The encoder is designed as

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

- $q_{\boldsymbol{w}_t}(\boldsymbol{b}_t \mid \boldsymbol{b}_{t-1}) = \text{Bernoulli}(\boldsymbol{b}_t; \sigma(\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1})))$
- The ELBO can be expressed as

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

ARM gradient of the ELBO

To compute the gradient with respect to $\boldsymbol{w}_t$,

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

we treat $\mathbb{E}_{q(\boldsymbol{b}_{t+1:T} \mid \boldsymbol{b}_t)}[f(\boldsymbol{b}_{1:T})]$ as a function

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

where

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

ARM for VAE

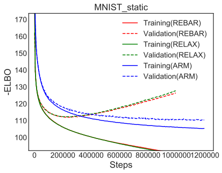
Linear		Nonlinear		Two layers	
Algrithm	$-\log p(x)$	Algrithm	$-\log p(x)$	Algrithm	$-\log p(x)$
NVIL *	$= 113.1$	NVIL *	$= 102.2$	NVIL*	$= 99.8$
WS*	$= 120.8$	WS*	-	WS*	$= 107.7$
REBAR*	$= 107.7$	REBAR*	$= 101.4$	REBAR*	$= \mathbf{95.4}$
MuProp†	≤ 113.0	MuProp*	$= 99.1$	MuProp†	≤ 100.4
Concrete*	$= 107.3$	Concrete*	$= 99.6$	Concrete*	$= 95.6$
RELAX‡	≤ 113.6	RELAX‡	≤ 119.2	RELAX‡	≤ 100.9
LeGrad	≤ 117.5	LeGrad	-	LeGrad	-
ARM	$= \mathbf{107.2 \pm 0.1}$	ARM	$= \mathbf{98.4 \pm 0.3}$	ARM	$= 96.7 \pm 0.3$

Table 2: Negative log-likelihood of discrete VAEs on test set trained with mutiple stochastic gradient estimators on MNIST where $*$, $\star$, $\dagger$, $\ddagger$ represent the reported results from Mnih & Gregor (2014), Tucker et al. (2017), Gu et al. (2016), Grathwohl et al. (2018) respectively. LeGrad result is obtained by running the author provided code. ARM rerult is over 5 random initializations with standard error of the mean.

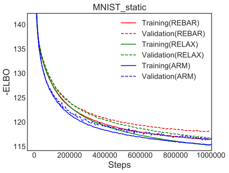
Linear		Nonlinear		Two layers	
Algrithm	$-\log p(x)$	Algrithm	$-\log p(x)$	Algrithm	$-\log p(x)$
NVIL*	$= 117.6$	NVIL*	$= \mathbf{116.6}$	NVIL*	$= 111.4$
REBAR*	$= 117.7$	REBAR*	$= 118.0$	REBAR*	$= 110.8$
MuProp*	$= 117.6$	MuProp*	$= 117.5$	MuProp*	$= 111.2$
Concrete*	$= 117.7$	Concrete*	$= 116.7$	Concrete*	$= 111.3$
RELAX‡	≤ 122.1	RELAX‡	≤ 128.2	RELAX‡	≤ 115.4
ARM	$= \mathbf{115.8 \pm 0.2}$	ARM	$= 117.6 \pm 0.4$	ARM	$= \mathbf{109.8 \pm 0.3}$

Table 3: Negative log-likelihood of discrete VAEs on test set trained with mutiple stochastic gradient estimators on OMNIGLOT where $*$, $\star$, $\ddagger$ represent the reported results from Mnih & Gregor (2014), Tucker et al. (2017),Grathwohl et al. (2018) respectively.

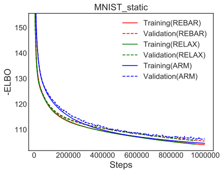
ARM for VAE



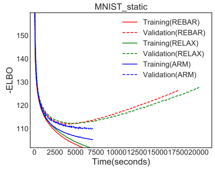
(a) Nonlinear



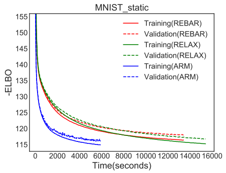
(b) Linear



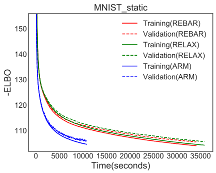
(c) Linear two layers



(d) Nonlinear



(e) Linear



(f) Linear two layers

Figure 5: Test negative ELBOs on MNIST(static) 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.

ARM for VAE

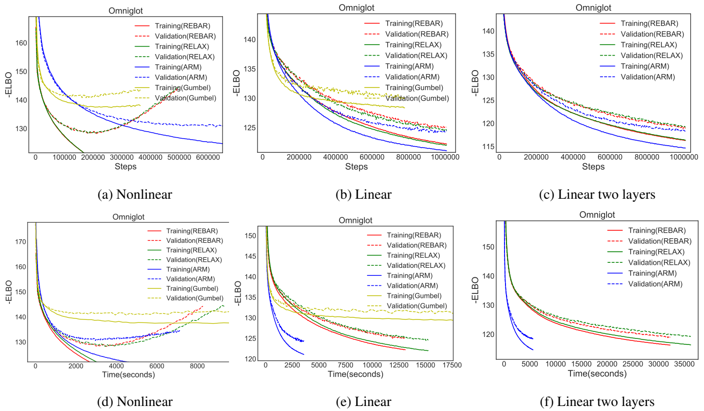


Figure 6: Test negative ELBOs on OMNIGLOT with respect to training iterations, shown in the top row, and wall clock times on Tesla-K40 GPU, shown in the bottom row, for three differently structured Bernoulli VAEs.

MLE with multiple discrete stochastic layers

- Predict lower half of MNIST digit $\boldsymbol{x}_l$ given the upper half $\boldsymbol{x}_u$;
- Maximizing the conditional likelihood $p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \,|\, \boldsymbol{x}_u)$
- Approximate $\log p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \,|\, \boldsymbol{x}_u)$ 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 $\boldsymbol{b}_1^{(k)} \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_1}(\boldsymbol{b}_2^{(k)}))),\ \boldsymbol{b}_2^{(k)} \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u)))$.
- Training with K=1 and approximate negative log-likelihood $-\log p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \,|\, \boldsymbol{x}_u)$ with $K = 1000$

MLE with multiple discrete stochastic layers

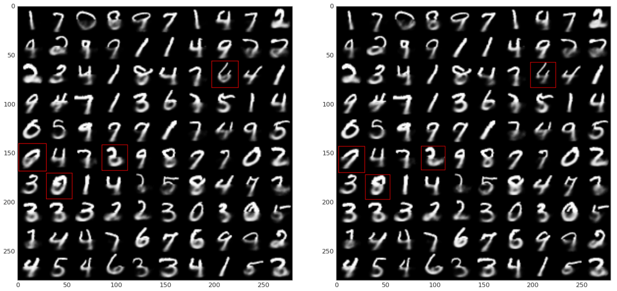


Figure 4: Randomly selected example results of predicting the lower half of a MNIST digit given its upper half, using a binary stochastic network, which has two binary linear stochastic hidden layers and is trained by ARM maximum likelihood inference.

Table 4: For the MNIST conditional distribution estimation benchmark task, comparison of the test negative log-likelihood between various gradient estimators in Jang et al. (2017) is reported here.

Gradient estimator	ARM	ST	DARN	Annealed ST	ST Gumbel-S.	SF	MuProp
$-\log p(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$	**57.9 ± 0.1**	58.9	59.7	58.7	59.3	72.0	58.9

Future work

- ARM gradient can be generalized to categorical variables by the the exponential racing property: $x_i \sim \text{Exp}(\lambda_i)$ are independent exponential random variables for $i = 1, \ldots, M$, then

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

- Combination with other variance reduction methods
- Applications in the reinforcement learning with discrete actions and natural language processing

Reference

- Connectionist learning of belief networks Neal (1992)
- Neural Variational Inference and Learning in Belief Networks Mnih and Gregor (2014)
- Local expectation gradients for black box variational inference Titsias and Lázaro-Gredilla (2015)
- MuProp: Unbiased backpropagation for stochastic neural networks Gu et al. (2016)
- Categorical reparameterization with Gumbel-softmaxMaddison et al. (2017); Jang et al. (2017)
- REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models Tucker et al. (2017)
- ARM: Augment-REINFORCE-merge gradient for discrete latent variable models Yin and Zhou (2018)

Thank you!