Methods for Latent Variable Models

Tianyu Xie
2022/7/1

1.Variational Inference for Monte Carlo Objectives (VIMCO)

V.I. for Monte Carlo Objectives

Problem Setting

- Suppose we would like to fit an intractable latent variable model $P(x, h)$ to data. Here x is observed variable and h is hidden variable.

- For example: Gaussian Mixture, state-space Markov model.

- In a MLE perspective, we use consider estimators of the form $\hat{I}(h^{1:K}) = \frac{1}{K}\sum_{i=1}^{K} f(x, h^i)$ as a surrogate for the marginal likelihood $P(x)$, where $h^{1:K}$ can be sampled from some distribution $Q(h \mid x)$.

- How to determine f ?

  - One choice: sample h^i from the prior $P(h)$.

    $$\hat{I}(h^{1:K}) = \frac{1}{K}\sum_{i=1}^{K} P(x|h^i) \text{ with } h^i \sim P(h).$$

    unbiased but may suffer from large variance.

V.I. for Monte Carlo Objectives

Problem Setting

- Another choice: consider a poposal distribution $Q(h \mid x)$ and use importance sampling,

$$\hat{I}(h^{1:K}) = \frac{1}{K} \sum_{i=1}^{K} \frac{P(x, h^i)}{Q(h^i|x)}$$

where $h^{1:K} \sim Q(h^{1:K} \mid x) \equiv \prod_{i=1} Q(h^i \mid x)$. The variance of this estimator is much lower because $Q(h \mid x)$ can be close to the $P(h \mid x)$.

- We can construct $\hat{L}(h^{1:K}) = \log \hat{I}(h^{1:K})$ to estimator the lower bound on the loglikelihood $P(x)$, i.e.

$$\begin{aligned} E_{Q(h^{1:K}|x)} \left[\log \hat{I}(h^{1:K}) \right] &\leq \log E_{Q(h^{1:K}|x)} \left[\hat{I}(h^{1:K}) \right] \\ &= \log P(x), \end{aligned}$$

- The equality holds iff $Q(h \mid x) = P(h \mid x)$. We can consider $\hat{L}(h^{1:K}) = \log \hat{I}(h^{1:K})$ as a stochastic lower bound.

V.I. for Monte Carlo Objectives

Problem Setting

- Note that the estimator $\hat{L}$ admits multi-sample version (K>1).

- Instead of optimizing the classical variational lower bound (K=1), we can optimize the multi-sample lower bound (K>1), that is

$$\mathcal{L}^K(x) = E_{Q(h^{1:K}|x)}\left[\log \frac{1}{K}\sum_{i=1}^{K} f(x, h^i)\right],$$

- Since $E_{Q(h^{1:K}|x)}\hat{L}(h^{1:K}) = L^K(x)$, in each iteration we can:
  - Sample $h^{1:K}$ from $Q(h^{1:K}|x)$.
  - Compute and take gradient of $\hat{L}(h^{1:K})$.
  - Update the parameters.

V.I. for Monte Carlo Objectives
Gradient Analysis

- Let $Q_\theta(h \mid x)$ be a model parametrized by θ, then the gradient of

$$\mathcal{L}^K(x) = E_{Q(h^{1:K}|x)}\left[\log \frac{1}{K}\sum_{i=1}^{K} f(x, h^i)\right]$$

w.r.t. θ is

$$\nabla_\theta \mathcal{L}^K(x) = E_{Q(h^{1:K}|x)}\left[\sum_j \hat{L}(h^{1:K})\nabla_\theta \log Q(h^j|x)\right] +$$
$$E_{Q(h^{1:K}|x)}\left[\sum_j \tilde{w}^j \nabla_\theta \log f(x, h^j)\right], \qquad (5)$$

where $\tilde{w}^j = f(x, h^j) / \sum_{i=1}^{K} f(x, h^i)$ can be considered as multi-sample weight.

- The second term is well-behaved, since $\tilde{w}^j$ are non-negative and sum to 1. Therefore, the second term can be considered as the convex combination of the per-sample gradient $\nabla_\theta \log f(x, h^j)$.

- Moreover, $\tilde{w}^j$ can be considered as the resposibility of sample j on x.

V.I. for Monte Carlo Objectives
Gradient Analysis

- The first term is

$$E_{Q(h^{1:K}|x)}\left[\sum_j \hat{L}(h^{1:K})\nabla_\theta \log Q(h^j|x)\right]$$

- The per-sample gradient $\nabla_\theta Q_\theta(h^j \mid x)$ is multiplied by the same scalar $\hat{L}(h^{1:K})$. Therefore, unlike the second term, the first term does not assign more weight to the 'good sample'. ($\hat{L}(h^{1:K})$ can be considered as *learning signal*).

- Another important problem is, the learning signal is $\hat{L}(h^{1:K})$ indeed unbounded, which means the magnitude of the first term can be much larger than the second term.

- In the earlier training, the samples from Q explains the data poorly, resulting in a very small $\hat{I}(h^{1:K})$ and thus a very negative learning signal.

V.I. for Monte Carlo Objectives
Gradient Estimation

- The naive but problematic estimator is

$$\nabla_\theta \mathcal{L}^K(x) \simeq \sum_j \hat{L}(h^{1:K}) \nabla_\theta \log Q(h^j|x) + \sum_j \tilde{w}^j \nabla_\theta \log f(x, h^j),$$

  where h^i is sampled from $Q(h \mid x)$.

- (Neural Inference and Learning, NVIL). To reduce the magnitude of learning signal by substracting a baseline from it, that is

$$\nabla_\theta \mathcal{L}^K(x) \simeq \sum_j (\hat{L}(h^{1:K}) - b(x) - b) \nabla_\theta \log Q(h^j|x) + \sum_j \tilde{w}^j \nabla_\theta \log f(x, h^j), \quad (8)$$

  where h^i is sampled from $Q(h \mid x)$.

- The constant b is to track the mean of the learning signal. The $b(x)$ is fit to minimize the learning signal of $\hat{L}(h^{1:K}) - b(x) - b$.

V.I. for Monte Carlo Objectives
Gradient Estimation

- Although NVIL reduce the variance of the gradient estimator, it has the same learning signal for all samples.

- We can alleviate this problem by defining a *local* learning signal for each sample. This can be accomplished by using a separate baseline for each sample that depends on the value of all other samples and eliminates much of the variance due to them.

- Let h^{-j} denote the set of $K-1$ samples obtained by leaving out sample j from the original set.

- The contribution of sample j to the first term can be expressed as

$$E_{Q(h^{1:K}|x)}\left[\hat{L}(h^{1:K})\nabla_\theta \log Q(h^j|x)\right] =$$
$$E_{Q(h^{-j}|x)}\left[E_{Q(h^j|x)}\left[\hat{L}(h^{1:K})\nabla_\theta \log Q(h^j|x)\middle| h^{-j}\right]\right].$$

V.I. for Monte Carlo Objectives
Gradient Estimation

$$E_{Q(h^{1:K}|x)}\left[\hat{L}(h^{1:K})\nabla_\theta \log Q(h^j|x)\right] =$$
$$E_{Q(h^{-j}|x)}\left[E_{Q(h^j|x)}\left[\hat{L}(h^{1:K})\nabla_\theta \log Q(h^j|x)\middle| h^{-j}\right]\right].$$

- The inner-most expectation is conditioned on all sampled except for h^j. Thus adding any function of h^{-j} to the learning signal does not effect the gradient.

- To reduce the maginitude of learning signal, we want to substract $\hat{L}(h^{1:K})$ by a estimator of it by h^{-j}. That is

$$\hat{L}(h^j|h^{-j}) = \hat{L}(h^{1:K}) - \log\frac{1}{K}\left(\sum\nolimits_{i\neq j} f(x,h^i) + f(x)\right)$$

- Here $f(x)$ can be some estimation of $f(x, h^j)$, but do not include h^j.

- $f(x)$ can be a parametrized model to be trained.

V.I. for Monte Carlo Objectives
Gradient Estimation

- We can avoid to learning an additional model by defining the arithmetic mean,

$$\hat{f}(x, h^{-j}) \doteq \frac{1}{K-1} \sum_{i \neq j} f(x, h^i))$$

or the geometric mean,

$$\hat{f}(x, h^{-j}) = \exp\left(\frac{1}{K-1} \sum_{i \neq j} \log f(x, h^i)\right)$$

- The resulting local learning signals can be written as

$$\hat{L}(h^j | h^{-j}) = \hat{L}(h^{1:K}) - \log \frac{1}{K}\left(\sum_{i \neq j} f(x, h^i) + \hat{f}(x, h^{-j})\right). \quad (9)$$

- The resulting VIMCO estimator has the form

$$\nabla_\theta \mathcal{L}^K(x) \simeq \sum_j \hat{L}(h^j | h^{-j}) \nabla_\theta \log Q(h^j | x) + \sum_j \tilde{w}^j \nabla_\theta \log f(x, h^j).$$

V.I. for Monte Carlo Objectives
Reweighted Wake Sleep

- Instead of optimizing the multi-sample lower bound, we could also directly maximize the loglikelihood. i.e. $E_{P(h|x)} \log Q_\theta(h \mid x)$.

- The gradient is

$$\nabla_\theta E_{P(h|x)} \log Q_\theta(h \mid x) = E_{P(h|x)} \nabla_\theta \log Q_\theta(h \mid x) = E_{Q_\theta(h|x)} \frac{P(h,x)/P(x)}{Q_\theta(h \mid x)} \nabla_\theta \log Q_\theta(h \mid x)$$

- Denote $w^j = \frac{P(h^j, x)}{Q_\theta(h^j \mid x)}$, where $h^j \sim Q(h \mid x)$. Then $\sum_j w_j$ is an estimator of $P(x)$. Let $\tilde{w}^j = w^j / \sum_j w^j$ be the self normalized weight.

- An estimator of the gradient is $\sum_j \tilde{w}^j \nabla_\theta \log Q_\theta(h^j \mid x)$. (Biased).

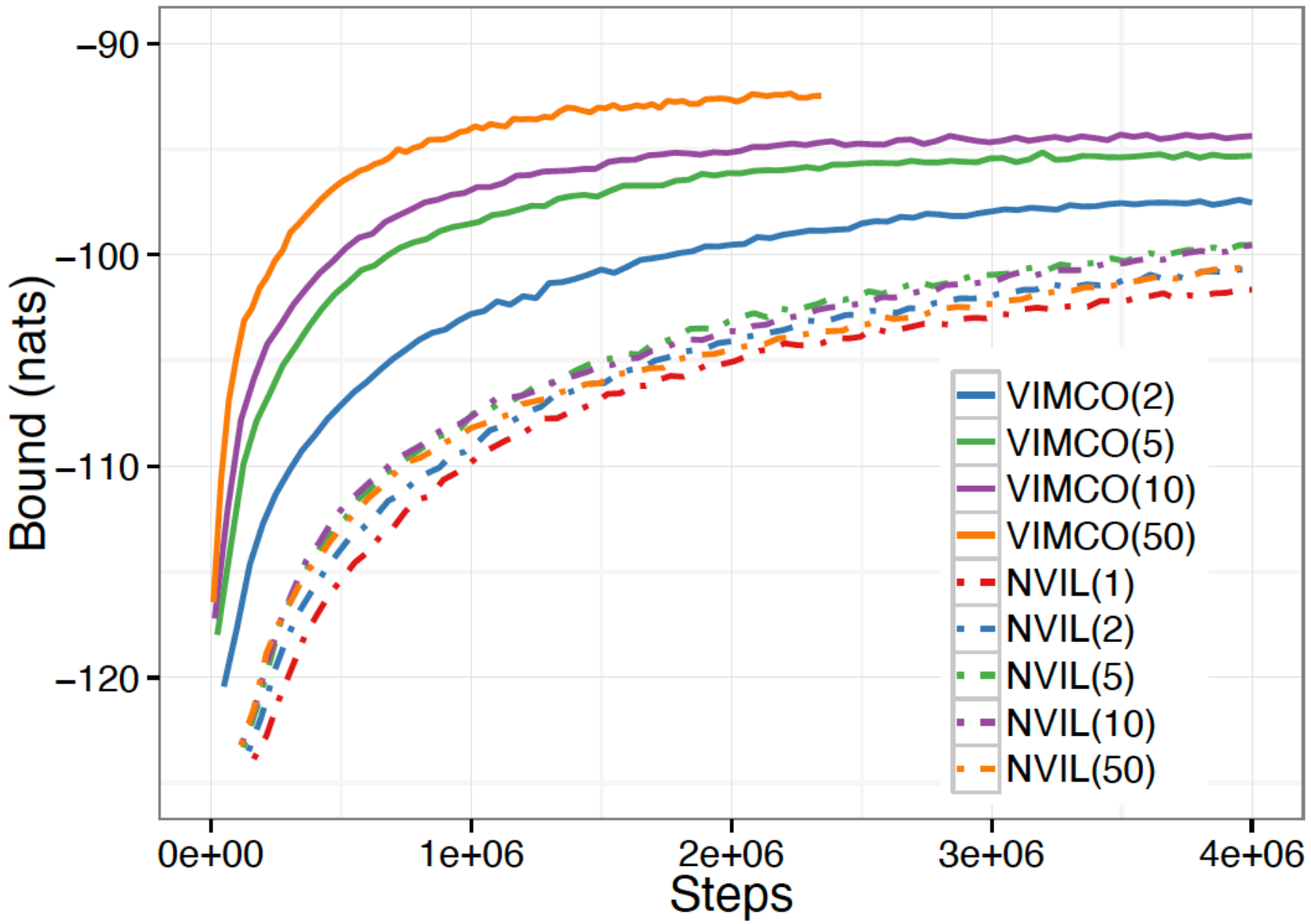
V.I. for Monte Carlo Objectives

Experiments

- 28*28 MNIST (binarization), 50000/10000/10000 samples for training, validation and test.
- We use sigmoid belief network to model $P(x, h)$ and $Q_\theta(h \mid x)$.
    - $P(x, h)$: 200-200-200-768, (three hidden layers of binary latent variables.)
    - $Q(h \mid x)$: 768-200-200-200.
- K = 2, 5, 10, 50 for VIMCO, NVIL, RWS to explore the multi-sample estimator.
- K = 1 using NVIL to serve as a single-sample baseline.

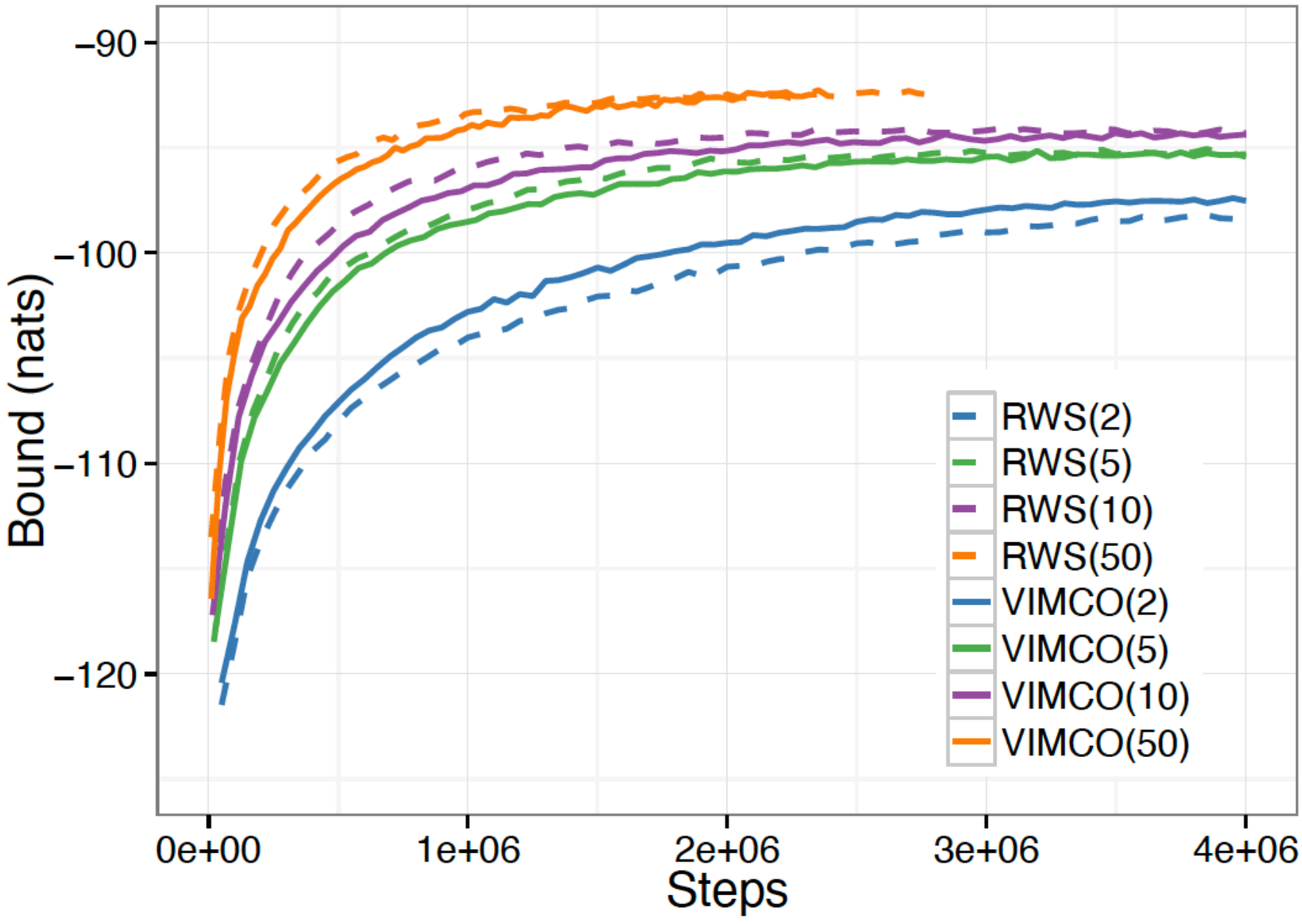
V.I. for Monte Carlo Objectives

Experiments



V.I. for Monte Carlo Objectives

Experiments



V.I. for Monte Carlo Objectives

Experiments

Table 1. Estimates of the negative log-likelihood (in nats) for generative modelling on MNIST. The model is an SBN with three latent layers of 200 binary units.

NUMBER OF SAMPLES	TRAINING ALG.		
	VIMCO	NVIL	RWS
1	—	95.2	—
2	93.5	93.6	94.6
5	92.8	93.7	93.4
10	92.6	93.4	93.0
50	91.9	96.2	92.5

- The NLL is calculated using 1000 proposal samples for each data point.
- The authors did not make comparisons about number of samples used.

V.I. for Monte Carlo Objectives

Experiments

- Effect of variance reduction

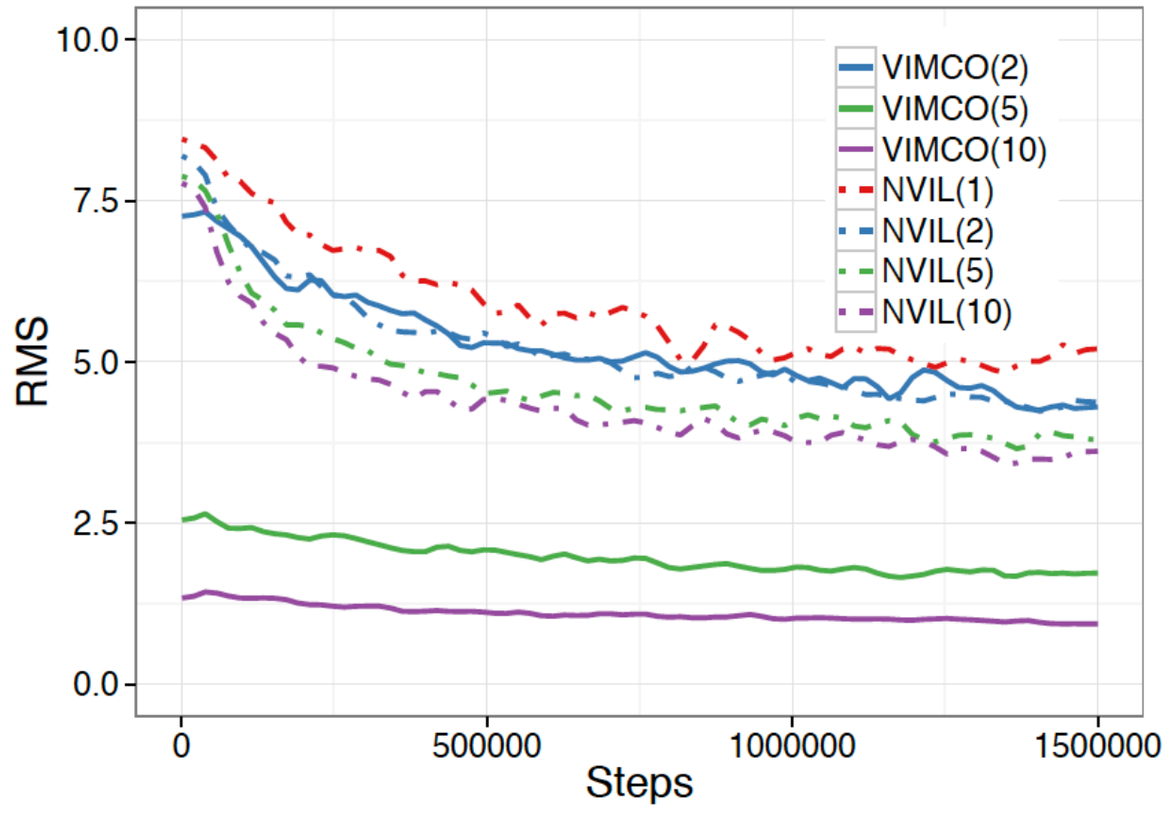


Figure 3. The magnitude (root mean square) of the learning signal for VIMCO and NVIL as a function of the number of samples used in the objective and the number of parameter updates.

2. Markovian Score Climbing (MSC)

Markovian Score Climbing
Problem Setting

- Assume the untractable distribution is $p(z, x)$, where z is the latent variable and x is the observed variable. Also, we assume a parametrized variational distribution $q(z; \lambda)$

- We want to optimize the inclusive KL divergence,

$$\mathrm{KL}\left(p(\mathbf{z}|\mathbf{x}) \,\|\, q(\mathbf{z};\lambda)\right) := \mathbb{E}_{p(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{z}|\mathbf{x}) - \log q(\mathbf{z};\lambda)\right].$$

- Minimizing this objective is equivalent to optmize the log likelihood:

$$\min_{\lambda} L_{\mathrm{KL}}(\lambda) := \min_{\lambda} \mathbb{E}_{p(\mathbf{z}|\mathbf{x})}\left[-\log q(\mathbf{z};\lambda)\right].$$

- We define the score function by $s(\mathbf{z};\lambda) := \nabla_\lambda \log q(\mathbf{z};\lambda)$. Then the RWS propose to estimate the gradient by

$$\nabla_\lambda L_{\mathrm{KL}}(\lambda) \approx -\sum_{s=1}^{S} \frac{w_s}{\sum_{r=1}^{S} w_r} s(\mathbf{z}^s;\lambda),$$

where $w_s = {}^{p(\mathbf{z}^s,\mathbf{x})}/_{q(\mathbf{z}^s;\lambda)}$, $\mathbf{z}^s \sim q(\mathbf{z}^s;\lambda)$, and $s(\mathbf{z};\lambda) = \nabla_\lambda \log q(\mathbf{z};\lambda)$.

Markovian Score Climbing
Problem Setting

- One issue with the RWS estimator is it is biased.

- In fact, to calculate the gradient, we need samples z^s from the true posterior distribution $P(z \mid x)$. To avoid this untractable procedure,

    - RWS proposes to sample z^s from the variational distribution and then assign them with self-normlized weights $\tilde{w}^s$.

- We can also sample z from a Markov kernel $M(\cdot \mid z_{t-1})$, here z_{t-1} is sample in the last step. $M(\cdot \mid z_{t-1})$ is constructed such that $P(z \mid x)$ is its invariant distribution.

Markovian Score Climbing
Method

- To obtain high-quality samples from $p(z \mid x)$, we have to run sufficiently many updates in the inner loop MCMC.

- However, we do not have to initialized the MCMC in each step. Instead, the sample $z[k-1]$ in the step $k-1$ can be pass to the Markov kernel $z[k] \sim M(\cdot \mid z[k-1])$. This kernel could also be updated in each step.

- Therefore, in a convergence state, we obtain both a Markov kernel and a approximate distribution.

- We could use the variational distribution $q(z : \lambda_{k-1})$ to construct the Markov kernel $z[k] \sim M(\cdot \mid z[k-1])$ in the $k-1$ step.

- We are moving in an ascent direction of the score function at each iteration and using MCMC, we refer to the method developed in this paper as **Markovian score climbing**.

Markovian Score Climbing
Conditional Importance Sampling

- To use the variational distribution $q(z : \lambda_{k-1})$ to construct the Markov kernel $z[k] \sim M(\cdot \,|\, z[k-1])$, we repeat the following procedure.

    - Set the first proposed sample to be equal to the conditional sample from the previous iteration, i.e. $z^1 = z[k-1]$.

    - Propose the remaining $S-1$ samples from a proposal distribution, $z^i \sim q(z; \lambda_{k-1}), i = 2, \ldots, S$.

    - Compute the importance weights for all S samples, denoted by $w^i = p(z^i, x)/q(z^i, \lambda_{k-1}), \tilde{w}^i = w^i / \sum_{j=1}^{S} w^j$.

    - Generate an updated conditional sample by pick one of the proposed values with probability proportional to its (normalized) weight.

Markovian Score Climbing
Conditional Importance Sampling

- To estimate the gradient (score function), one option is

$$s(\mathbf{z}[k]\,;\,\lambda) = \nabla_\lambda \log q(\mathbf{z}[k]\,;\,\lambda),$$

another option is to make use of all samples in each iteration, i.e. the Rao-Blackwellized estimate

$$\widehat{g}_{\mathrm{KL}}(\lambda) = \sum_{i=1}^{S} \bar{w}^i s(\mathbf{z}^i\,;\,\lambda).$$

Algorithm 2: Conditional Importance Sampling

Input : Model $p(\mathbf{z},\mathbf{x})$, proposal $q(\mathbf{z}\,;\,\lambda)$, conditional sample $\mathbf{z}[k-1]$, and total number of internal samples S.

Output: $\mathbf{z}[k] \sim M(\cdot\,|\,\mathbf{z}[k-1]\,;\,\lambda)$, updated conditional sample.

1 Set $\mathbf{z}^1 = \mathbf{z}[k-1]$

2 Sample $\mathbf{z}^i \sim q(\mathbf{z}\,;\,\lambda)$ for $i = 2,\ldots,S$

3 Compute $w^i = {}^{p(\mathbf{z}^i,\mathbf{x})}/_{q(\mathbf{z}^i\,;\,\lambda)}$, $\bar{w}^i = {}^{w^i}/_{\sum_{j=1}^{S} w^j}$ for $i = 1,\ldots,S$

4 Sample J with probability $\mathbb{P}(J = j) \propto \bar{w}^j$

5 Set $\mathbf{z}[k] = \mathbf{z}^J$

Markovian Score Climbing
Algorithm

- Having estimate for the gradient, we obtain the MSC algorithm for minimization of the inclusive KL divergence.

Algorithm 3: Markovian Score Climbing with ML

Input : Markov kernel $M(\mathbf{z}' \mid \mathbf{z}; \theta, \lambda)$ with stationary distribution $p(\mathbf{z} \mid \mathbf{x}; \theta)$, variational family $q(\mathbf{z}; \lambda)$, initial $\lambda_0, \mathbf{z}[0], \theta_0$, step size sequences $\varepsilon_k, \epsilon_k$, and iterations K.

Output: $\lambda_K \approx \lambda^\star$, $\theta_K \approx \theta^\star$.

for $k = 1, \ldots, K$ **do**
  Sample $\mathbf{z}[k] \sim M(\cdot \mid \mathbf{z}[k-1]; \theta_{k-1}, \lambda_{k-1})$
  Compute $s(\mathbf{z}[k]; \lambda_{k-1}) = \nabla_\lambda \log q(\mathbf{z}[k]; \lambda_{k-1})$
  Compute $\widehat{g}_{\mathrm{ML}}(\theta_{k-1}) = \nabla_\theta \log p(\mathbf{z}[k], \mathbf{x}; \theta_{k-1})$
  Set $\lambda_k = \lambda_{k-1} + \varepsilon_k s(\mathbf{z}[k]; \lambda_{k-1})$
  Set $\theta_k = \theta_{k-1} + \epsilon_k \widehat{g}_{\mathrm{ML}}(\theta_{k-1})$
end

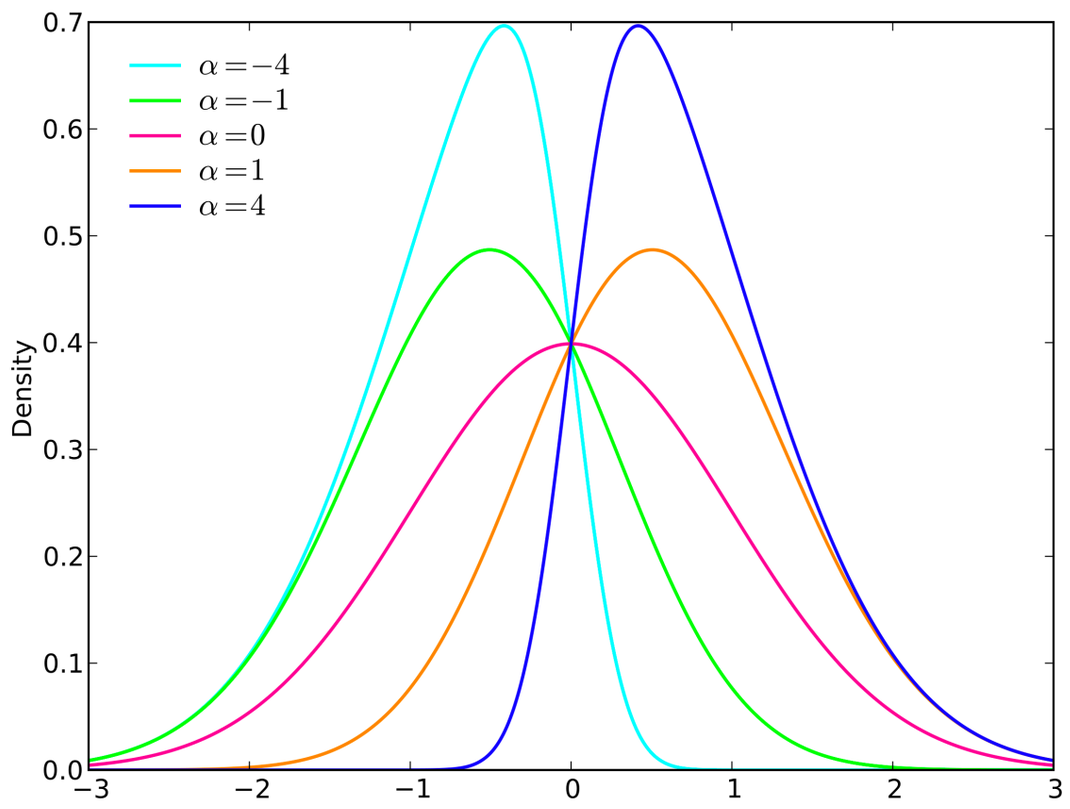
Markovian Score Climbing
Expriments: Skewed Normal Distribution

- The p.d.f. of skewed normal distribution is

Parameters	ξ location (real) ω scale (positive, real) α shape (real)
Support	$x \in (-\infty; +\infty)$
PDF	$\frac{2}{\omega\sqrt{2\pi}} e^{-\frac{(x-\xi)^2}{2\omega^2}} \int_{-\infty}^{\alpha\left(\frac{x-\xi}{\omega}\right)} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt$

- Let $p(z \mid x)$ be a skewed normal distribution with parameters $(\xi, \omega, \alpha) = (0.5,2,5)$

- Let variational approximation be a normal distribution $N(z; \mu, \sigma^2)$.



Markovian Score Climbing
Expriments: Skewed Normal Distribution

- We set the number of samples to $S = 2$. The biased gradient estimator self-normalized IS leads to systematic errors when estimating the variational parameters.

- RWS tends to underestimate the uncertainty.

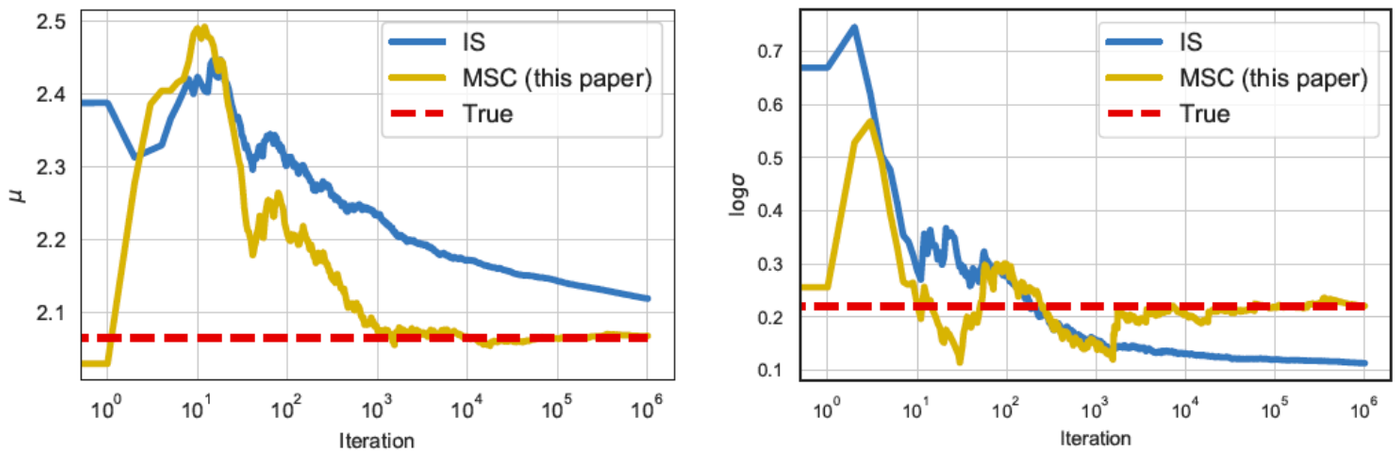


Figure 1: MSC converges to the true solution, while the biased IS approach does not. Example of learnt variational parameters for IS- and MSC-based gradients of the inclusive KL, as well as true parameters. Gaussian approximation to a skew normal distribution. Iterations in log-scale.

Markovian Score Climbing
Expriments: Skewed Normal Distribution

- We set the number of samples to $S = 2$. The biased gradient estimator self-normalized IS leads to systematic errors when estimating the variational parameters.

- RWS tends to underestimate the uncertainty.

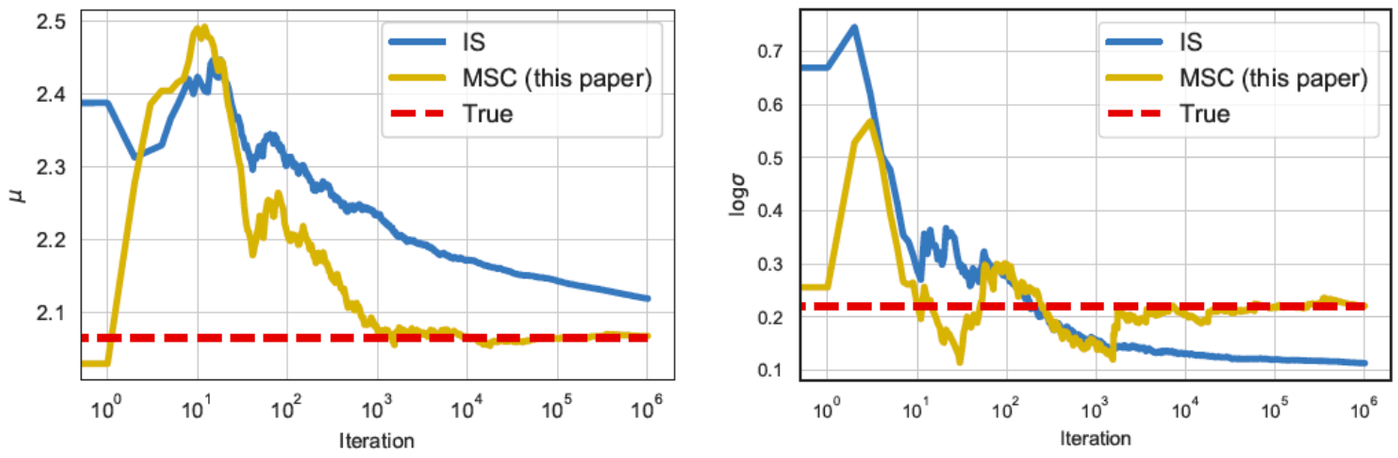


Figure 1: MSC converges to the true solution, while the biased IS approach does not. Example of learnt variational parameters for IS- and MSC-based gradients of the inclusive KL, as well as true parameters. Gaussian approximation to a skew normal distribution. Iterations in log-scale.

Markovian Score Climbing
Expriments: Bayesian Probit Regression

- The Bayesian probit regression model assigns a Gaussian prior to the parameters. Here, we consider the parameters as the hidden variable z.

- The prior and likelihood are

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I),\ \mathbb{P}(\mathbf{y}_t = y \mid \mathbf{z}, \mathbf{x}_t) = \Phi(\mathbf{x}_t^\top \mathbf{z})^y \left(1 - \Phi(\mathbf{x}_t^\top \mathbf{z})\right)^{1-y}$$

- The variational approximation is a Gaussian family: $q(z; \lambda) = N(z; \mu, \Sigma)$. Here, Σ is a diagonal covariance matrix.

- We apply this model for prediction in several UCI datasets.

Markovian Score Climbing
Expriments: Bayesian Probit Regression

- The test error reported by the authors is not satisfactory.

Dataset	EP [43]	IS [7]	MSC (adaptive)	MSC (prior)
Pima	0.227 ± 0.048	0.229 ± 0.047	0.227 ± 0.046	0.456 ± 0.093
Ionos	0.115 ± 0.053	0.115 ± 0.054	0.117 ± 0.053	0.182 ± 0.070
Heart	0.161 ± 0.066	0.163 ± 0.066	0.160 ± 0.063	0.342 ± 0.11

Table 1: Test error for Bayesian probit regression; lower is better. Estimated using EP [43], IS (cf. [7]), and MSC (this paper) with proposal $q(\mathbf{z}; \lambda)$ (adaptive) or $p(\mathbf{z})$ (prior) for 3 UCI datasets. Predictive performance is comparable, but MSC is more robust and generically applicable.

Markovian Score Climbing
Expriments: Bayesian Probit Regression

- The authors argue that MSC is more robust to the sample size S.

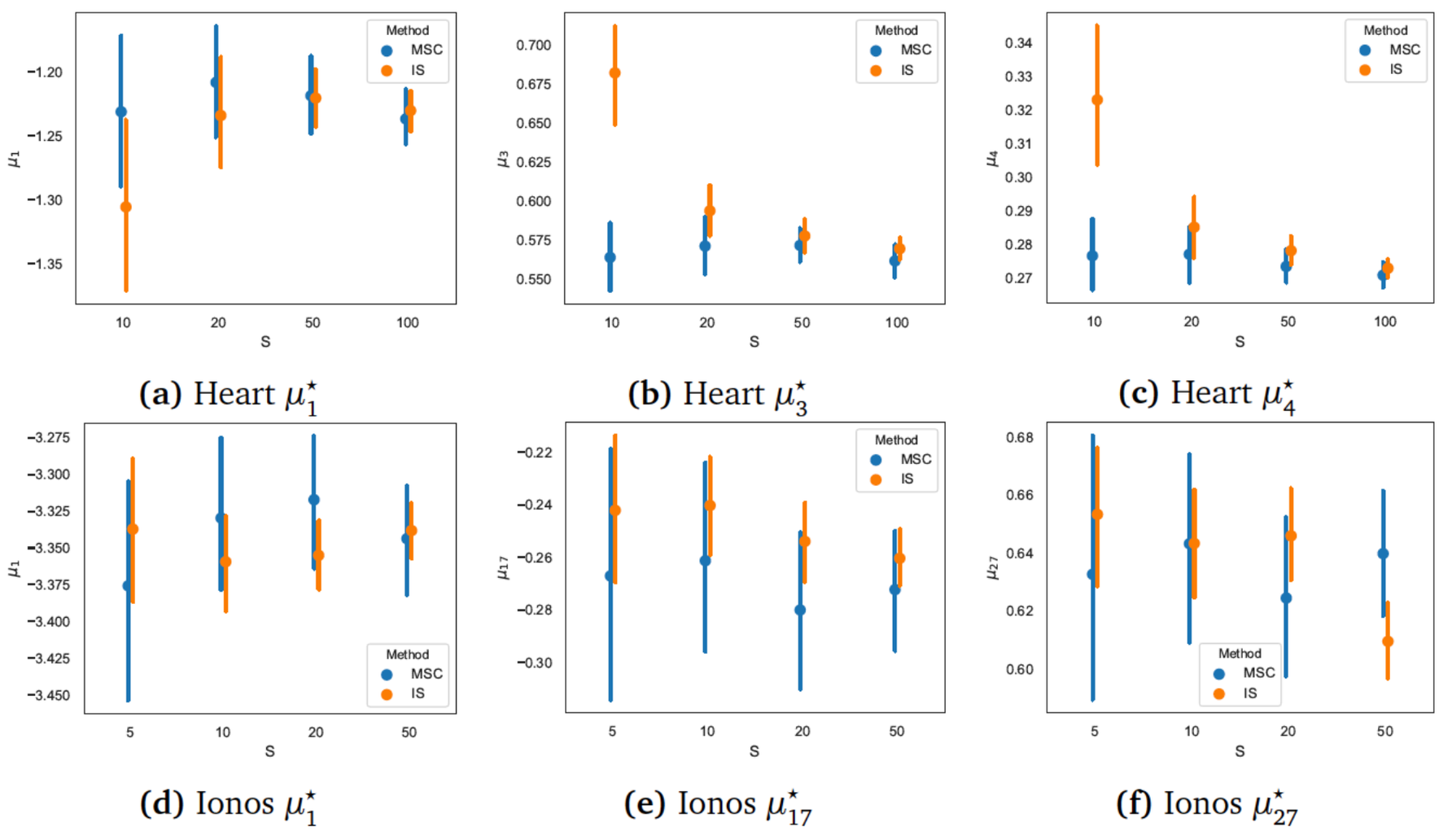


Figure 2: MSC is more robust to the number of samples S. The fitted mean parameter $\mu^\star$, for three representative dimensions of $\mathbf{z}$, of MSC (this paper) and IS (cf. [7]) on the Ionos and Heart datasets. The error bars corresponds to 100 random initializations.

3. Joint Stochastic Approximation and Its Application to Learning Discrete Latent Variable Models (JSA)

Joint Stochastic Approximation
Problem Setting

- In some problems, the 'true density' $p_\theta(h, x)$ may also contains unknown parameter θ.

- Similar to MSC, we want minimize w.r.t. ϕ the inclusive divergence

$$KL[p_\theta(h|x)||q_\phi(h|x)]$$

  and maximize the the marginal log-likelihood $\log p_\theta(x)$.

- The lower bound is

$$\begin{aligned} ELBO(\theta, \phi; x) &\triangleq E_{q_\phi(h|x)} \log \frac{p_\theta(x, h)}{q_\phi(h|x)} \\ &= \log p_\theta(x) - KL\left[q_\phi(h|x)||p_\theta(h|x)\right] \end{aligned}$$

Joint Stochastic Approximation
Problem Setting

- Formally, our goal is to

$$\begin{cases} \min_{\theta} KL\left[\tilde{p}(x)||p_\theta(x)\right] \\ \min_{\phi} E_{\tilde{p}(x)} KL\left[p_\theta(h|x)||q_\phi(h|x)\right] \end{cases} \tag{2}$$

  where $\tilde{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \delta(x - x_i)$ denote the empirical distribution for a training dataset consisting of n independent and i.i.d. data points $\{x_1, x_2, \ldots, x_n\}$.

- In such a way, we pursue direct maximum likelihood estimation of θ and at the same time avoid the high-variance gradient estimate w.r.t. ϕ.

- We can use the stochastic framework to solve the joint optimization problem (2).

Joint Stochastic Approximation

Stochastic Approximation

- Stochastic approximation (SA) is introduced bby Robbins and Monro (1951).
- SA is to find the solution λ^* of $f(\lambda) = 0$ with

$$f(\lambda) = E_{z \sim p_\lambda(\cdot)}[F_\lambda(z)],$$

  This system consists of d constraints for determining d-dimensional λ.

- SA propose to:
  - sample from $z^{(t)}$ with a Markov kernel $K_{\lambda^{(t-1)}}(z^{(t-1)}, \cdot)$, which admits $p_{\lambda^{t-1}}$ as the invariant distribution.
  - Update the parameter using $F_{\lambda^{t-1}}(z^{(t)})$.

Joint Stochastic Approximation

Stochastic Approximation

Algorithm 1 The general stochastic approximation (SA) algorithm

for $t = 1, 2, \cdots$ **do**

Monte Carlo sampling: Draw a sample $z^{(t)}$ with a Markov transition kernel $K_{\lambda^{(t-1)}}(z^{(t-1)}, \cdot)$, which starts with $z^{(t-1)}$ and admits $p_{\lambda^{(t-1)}}(\cdot)$ as the invariant distribution.

SA updating: Set $\lambda^{(t)} = \lambda^{(t-1)} + \gamma_t F_{\lambda^{(t-1)}}(z^{(t)})$, where γ_t is the learning rate.

end for

- The gradient for optimizing the two objectives can be derived as follows:

$$\begin{cases} g_\theta \triangleq -\nabla_\theta KL[\tilde{p}(x)||p_\theta(x)] \\ \qquad\qquad = E_{\tilde{p}(x)p_\theta(h|x)}\left[\nabla_\theta log p_\theta(x,h)\right] \\ g_\phi \triangleq -\nabla_\phi E_{\tilde{p}(x)} KL\left[p_\theta(h|x)||q_\phi(h|x)\right] \\ \qquad\qquad = E_{\tilde{p}(x)p_\theta(h|x)}\left[\nabla_\phi log q_\phi(h|x)\right] \end{cases}$$

Joint Stochastic Approximation

Stochastic Approximation

Proposition 1. *The gradients w.r.t. θ and ϕ as in Eq.(3) can be recast in the expectation form of Eq.(1) (i.e. as expectation of stochastic gradients), by letting $\lambda \triangleq (\theta, \phi)^T$, $z \triangleq (\kappa, h_1, \cdots, h_n)^T$, $p_\lambda(z) \triangleq \frac{1}{n}\prod_{i=1}^{n} p_\theta(h_i|x_i)$, $f(\lambda) \triangleq (g_\theta, g_\phi)^T$, and*

$$F_\lambda(z) \triangleq \begin{pmatrix} \sum_{i=1}^{n} \delta(\kappa = i)\nabla_\theta log p_\theta(x_i, h_i) \\ \sum_{i=1}^{n} \delta(\kappa = i)\nabla_\phi log q_\phi(h_i|x_i) \end{pmatrix}.$$

In order to avoid visiting all $h_1, \cdots, h_n$ at every SA iteration (to be explained below), we introduce an index variable κ which is uniformly distributed over $1, \cdots, n$. $\delta(\kappa = i)$ denotes the indicator of κ taking i.

Joint Stochastic Approximation
Stochastic Approximation

- The JSA algorithm is

Algorithm 2 The JSA algorithm

repeat

Monte Carlo sampling:

Draw κ over $1, \cdots, n$, pick the data point x_κ along with the cached $h_\kappa^{(old)}$, and use MIS to draw h_κ;

SA updating:

Update θ by ascending: $\nabla_\theta log p_\theta(x_\kappa, h_\kappa)$;

Update ϕ by ascending: $\nabla_\phi log q_\phi(h_\kappa | x_\kappa)$;

until convergence

- To design the transition kernel, given current sample $h_i^{(t-1)}$, MIS works as follows,
  - Propose $h_i \sim q_\phi(h_i | x_i)$.
  - Accept $h_i^{(t)} = h_i$ with prob = $\min\left\{1, \frac{w(h_i)}{w(h_i^{(i-1)})}\right\}$, where $w(h_i) = \frac{p_\theta(h_i | x_i)}{q_\phi(h_i | x_i)}$.

Joint Stochastic Approximation
Connection to MCMC-SAEM

- At iteration t, the E-step calculate the Q-function,

$$Q(\theta|\theta^{(t-1)}) = E_{p_{\theta^{(t-1)}}(h|x)}\left[\hat{\nabla}_\theta log p_\theta(x,h)\right]$$

  and the M step updates θ by maximizing $Q(\theta\,|\,\theta^{(t-1)})$.

- If the M-step can't be solved analytically, we can solve this problem using stochastic approximation.

- When exact sampling from $q_{\theta^{(t-1)}}(h\,|\,x)$ is difficult, we can draw a sample h by applying a Markov transition kernel which admits $p_{\theta^{(t-1)}}(h\,|\,x)$ as the invariant distribution.

Joint Stochastic Approximation
Experiments

- We train generative models with Bernoulli (or binary) latent variable. For JSA , RWS and VIMCO, we use particle number = 2.

- Three different network are examined,

Table 4: The three different network architectures in generative modeling with Bernoulli variables on MNIST, which are the same as Table 1 in (Yin and Zhou, 2019). The following symbols "$\rightarrow$", "]", ")", and "$\rightsquigarrow$" represent deterministic linear transform, leaky rectified linear units (LeakyReLU) nonlinear activation, sigmoid nonlinear activation, and random sampling respectively.

	Nonlinear	Linear	Linear two layers
$q_\phi(h\|x)$	$784 \rightarrow 200] \rightarrow 200] \rightarrow 200) \rightsquigarrow 200$	$784 \rightarrow 200) \rightsquigarrow 200$	$784 \rightarrow 200) \rightsquigarrow 200 \rightarrow 200 \rightsquigarrow 200$
$p_\theta(x\|h)$	$784 \leftsquigarrow (784 \leftarrow [200 \leftarrow [200 \leftarrow 200$	$784 \leftsquigarrow (784 \leftarrow 200$	$784 \leftsquigarrow (784 \leftarrow 200 \leftsquigarrow (200 \leftarrow 200$

- We use the binarized MNIST with the standard training/validation/testing partition 50000/10000/10000.

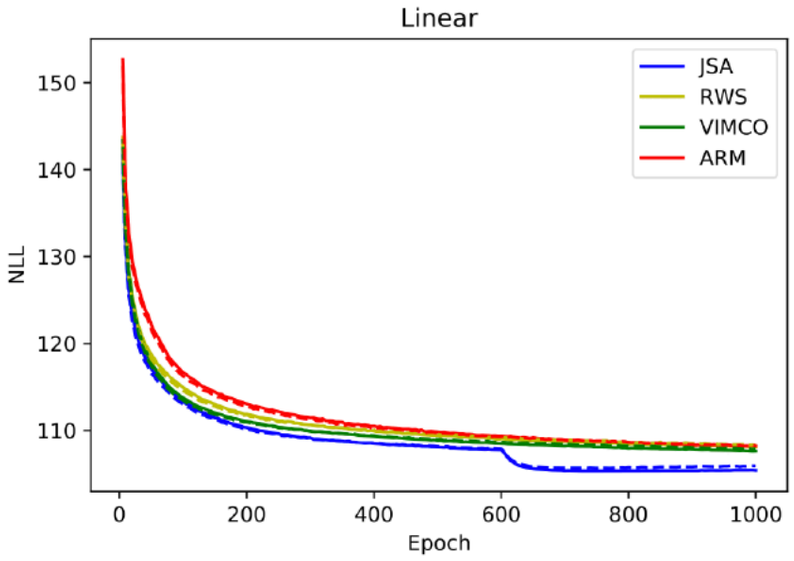
Joint Stochastic Approximation
Experiments

Table 1: Test NLL of different methods with three different network architectures on generative modeling with Bernoulli variables on MNIST, where $*$ denotes the results reported in (Yin and Zhou, 2019), and the others are obtained based on our implementation[12]. The mean and standard deviation results are computed over five independent trials with different random seeds.

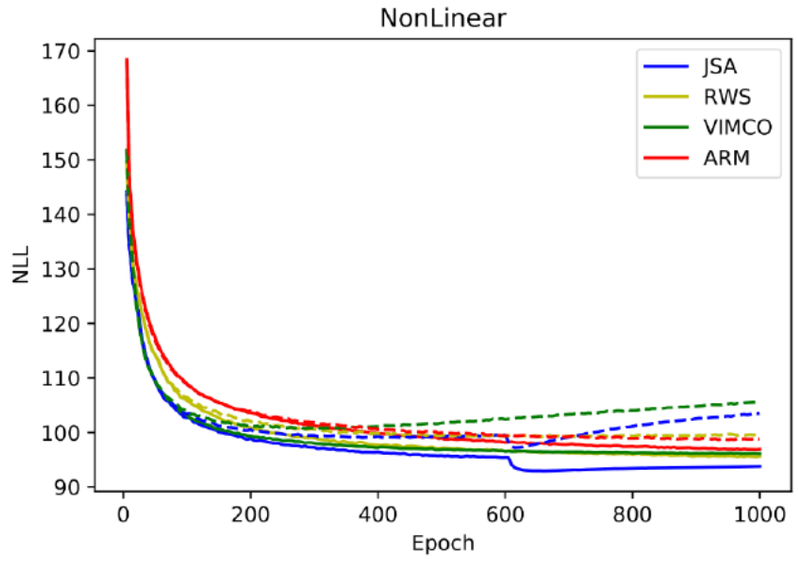
Method	Linear	Nonlinear	Two layers
REINFORCE*	170.1	114.1	159.2
RWS	108.0 ± 0.3	99.2 ± 0.2	96.5 ± 0.2
NVIL*	113.1	102.2	99.8
Concrete*	107.2	99.6	95.6
REBAR*	107.7	100.5	95.5
VIMCO	107.5 ± 0.3	100.6 ± 0.3	95.8 ± 0.1
ARM*	107.2 ± 0.1	98.4 ± 0.3	96.7 ± 0.3
JSA	105.5 ± 0.1	98.2 ± 0.4	95.3 ± 0.1

Joint Stochastic Approximation

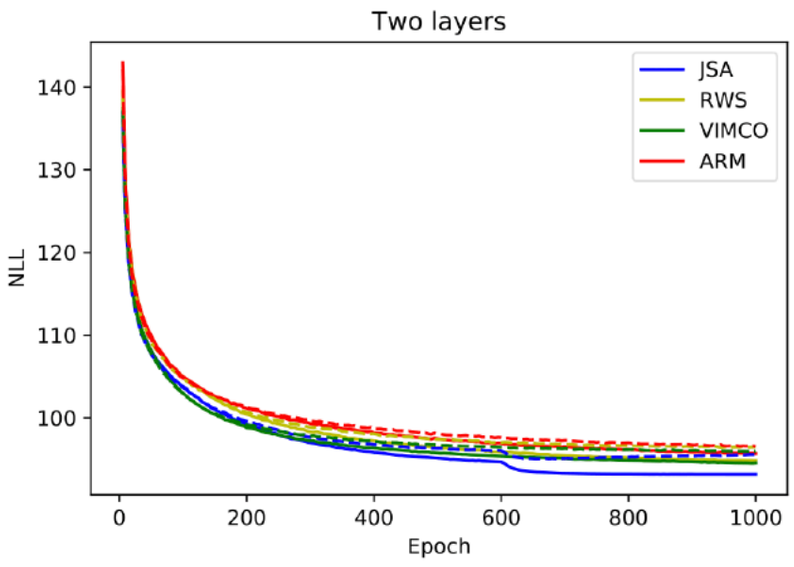
Experiments



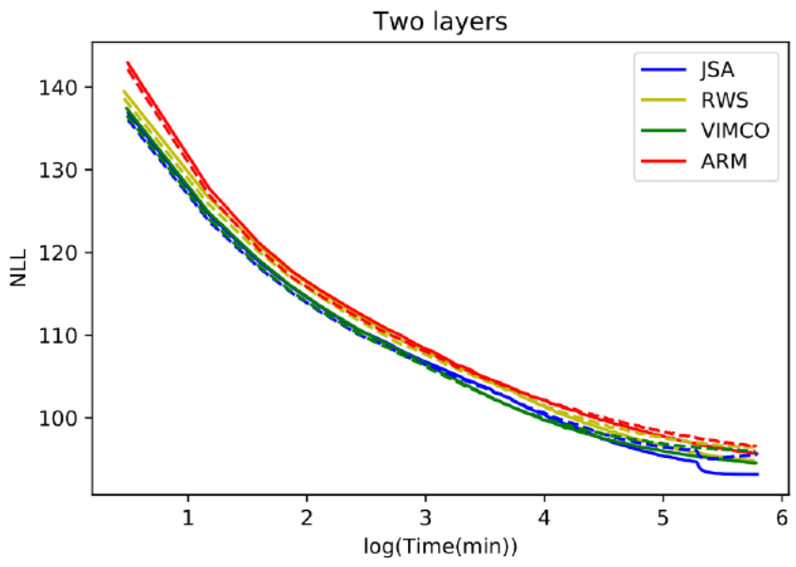
(a) NLL vs training epochs for "Linear"



(b) NLL vs training epochs for "Nonlinear"



(c) NLL vs training epochs for "Two layers"



(d) NLL vs training time for "Two layers"

Figure 1: Plots of training and testing NLL curves for training models with Bernoulli variables on MNIST. (a)(b)(c) are NLL against training epochs for three different network architectures, and (d) is NLL against training wall-clock times for the "Two layers" architecture (See Table 4 in Appendix for the other two architectures). The solid and dash lines correspond to the training and testing NLL respectively. For completeness, we include the testing NLLs against prolonged training epochs in Figure 1 and 2, which show overfitting (see Appendix C for comments).

4. Tighter Variational Bounds are Not Necessarily Better

Tighter Variational Bounds are Not Necessarily Better

Problem Setting

- For training deep generative models, evidence lower bound provides surrogate targets that are more amenable to optimization.
    - In general, these methods simultaneously learn an inference network alongside the target generative network. (Such as VAE).
- In traditional Bayesian inference contexts, the generative model is fixed and the inference model is the target for training.
- The performance of variational approaches depends upon:
    - The choice of inference network.
    - The choice of the evidence lower bound.
- The authors claim that tighter lower bounds are better for training generative model, and looser bounds are preferable for training the inference network.

Tighter Variational Bounds are Not Necessarily Better

Problem Setting

- In VAE, the target is

$$\begin{aligned}\text{ELBO}_{\text{VAE}}(\theta, \phi, x) &:= \int q_\phi(z|x) \log \frac{p_\theta(x, z)}{q_\phi(z|x)} \, \mathrm{d}z \\ &= \log p_\theta(x) - \mathrm{KL}(q_\phi(z|x)||p_\theta(z|x)).\end{aligned}$$

  where sampling from $q_\phi(z \mid x)$ is using reparameterization.

- The IWAE (important-weighted auto-encoder) objective is defined by

$$\begin{aligned}Q_{\text{IS}}(z_{1:K}|x) &:= \prod_{k=1}^{K} q_\phi(z_k|x), \\ \hat{Z}_{\text{IS}}(z_{1:K}, x) &:= \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(x, z_k)}{q_\phi(z_k|x)}, \qquad (2) \\ \text{ELBO}_{\text{IS}}(\theta, \phi, x) &:= \int Q_{\text{IS}}(z_{1:K}|x) \log \hat{Z}_{\text{IS}}(z_{1:K}, x) \, \mathrm{d}z_{1:K}\end{aligned}$$

  This multi-sample bound seperate the tightness and the expressiveness of the class q_ϕ. Setting $K > 1$ leads to significant gains over the VAE in terms of learning the generative model.

Tighter Variational Bounds are Not Necessarily Better

SNR

- We sduty the IWAE gradient estimator constructed as the average of M estimates, each built from K independent particles. Here changing M changes the variance, and changing K changes the ELBO itself.

- Assume that reparametrization is possible, we can express our gradient estimate in the general form

$$\Delta_{M,K} := \frac{1}{M}\sum_{m=1}^{M} \nabla_{\theta,\phi} \log \frac{1}{K}\sum_{k=1}^{K} w_{m,k}, \qquad (3)$$

$$\text{where } w_{m,k} = \frac{p_\theta(z_{m,k}, x)}{q_\phi(z_{m,k}|x)} \text{ and } z_{m,k} \overset{i.i.d.}{\sim} q_\phi(z_{m,k}|x).$$

- We will use $\Delta_{M,K}(\theta)$ and $\Delta_{M,K}(\phi)$ refer to gradient estimates w.r.t. θ and ϕ.

- If $K = 1$, this corresponds to the VAE objectives.

- If $M = 1$, this corresponds to the IWAE objectives.

Tighter Variational Bounds are Not Necessarily Better

SNR

- When $K \to \infty$, it must be the case that $\Delta_{M,K}(\phi) \to 0$. Therefore, when K becomes large, the variance of gradient estimate must decrease along with its expectation.

- To formmaly investigate the relation of expectation and variance, we define the signal-to-noise-ratio (SNR).

$$\mathrm{SNR}_{M,K}(\theta) = \left|\mathbb{E}\left[\Delta_{M,K}(\theta)\right] / \sigma\left[\Delta_{M,K}(\theta)\right]\right|$$

  The SNR is defined separately on each dimension and similarly for $\mathrm{SNR}_{M,K}(\phi)$.

- SNR provides a measure of the relative accuracy of the gradient estimates.

- We claim that $\mathrm{SNR}_{M,K}(\theta) = O(\sqrt{MK})$ and $\mathrm{SNR}_{M,K}(\phi) = O(\sqrt{M/K})$.

Tighter Variational Bounds are Not Necessarily Better

SNR

- As $M \to \infty$, CLT tells us that SNR at a rate $O(M)$.
- As for the effect of K,

$$\nabla_{\theta,\phi} \log \hat{Z}_{m,K} = \sum_{k=1}^{K} \frac{w_{m,k}}{\sum_{\ell=1}^{K} w_{m,k}} \nabla_{\theta,\phi} \log \left(w_{m,k}\right)$$

This can be interpreted as a self-normalized importance sampling estimate. Hesterberg(1988) stated that the bias of SN importance sampler converges at a rate $O(1/K)$, and std converges at a rate $O(1/\sqrt{K})$.
We thus see that SNR converge at $O(1/\sqrt{K})$ if the asymptotic gradient is 0 and $O(\sqrt{K})$ otherwise.

Theorem 1. *Assume that when $M = K = 1$, the expected gradients; the variances of the gradients; and the first four moments of $w_{1,1}$, $\nabla_\theta w_{1,1}$, and $\nabla_\phi w_{1,1}$ are all finite and the variances are also non-zero. Then the signal-to-noise ratios of the gradient estimates converge at the following rates*

$$\text{SNR}_{M,K}(\theta) = \sqrt{M} \left| \frac{\sqrt{K}\, \nabla_\theta Z - \frac{1}{2Z\sqrt{K}} \nabla_\theta \left(\frac{\text{Var}[w_{1,1}]}{Z^2} \right) + O\left(\frac{1}{K^{3/2}}\right)}{\sqrt{\mathbb{E}\left[w_{1,1}^2 \left(\nabla_\theta \log w_{1,1} - \nabla_\theta \log Z \right)^2 \right]} + O\left(\frac{1}{K}\right)} \right| \tag{5}$$

$$\text{SNR}_{M,K}(\phi) = \sqrt{M} \left| \frac{\nabla_\phi \text{Var}\left[w_{1,1}\right] + O\left(\frac{1}{K}\right)}{2Z\sqrt{K}\, \sigma\left[\nabla_\phi w_{1,1}\right] + O\left(\frac{1}{\sqrt{K}}\right)} \right| \tag{6}$$

where $Z := p_\theta(x)$ is the true marginal likelihood.

Tighter Variational Bounds are Not Necessarily Better

SNR

- Therefore, SNR for IWAE inference network gets worse as we increase K.
- In training deep generative models, one does not optimize a single ELBO but instead its average over multiple data points, i.e.

$$\mathcal{J}(\theta, \phi) := \frac{1}{N} \sum_{n=1}^{N} \text{ELBO}_{\text{IS}}(\theta, \phi, x^{(n)})$$

- So

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N} \Delta_{M,K}^{(n)}\right] = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}\left[\Delta_{M,K}^{(n)}\right],$$

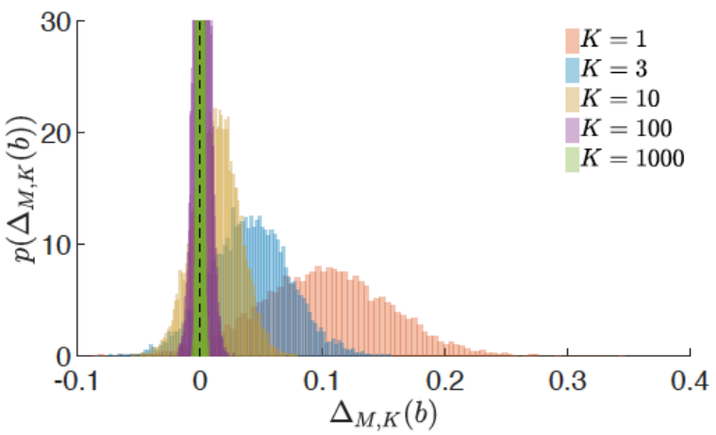
$$\text{Var}\left[\frac{1}{N}\sum_{n=1}^{N} \Delta_{M,K}^{(n)}\right] = \frac{1}{N^2}\sum_{n=1}^{N} \text{Var}\left[\Delta_{M,K}^{(n)}\right].$$

- Therefore, $\text{SNR}_{N,M,K}(\theta) = O(\sqrt{NMK})$ and $\text{SNR}_{N,M,K}(\phi) = O(\sqrt{NM/K})$

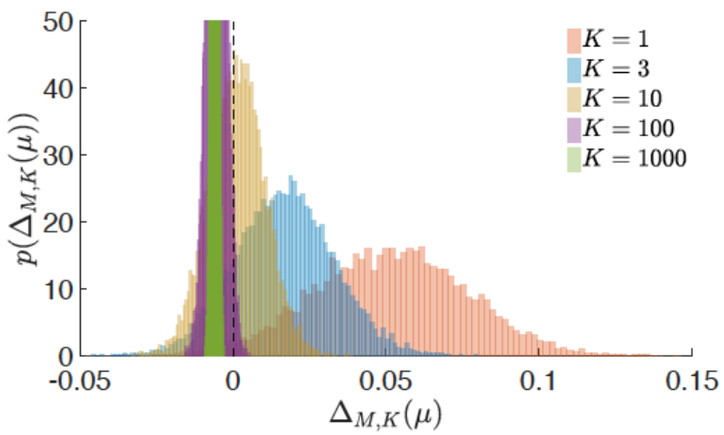
Tighter Variational Bounds are Not Necessarily Better

Empirical Comfirmation

- For the generative model, $z \sim N(z; \mu, I), x \mid z \sim N(x; z, I)$. For the inference model, $q_\phi(z \mid x) = N(z; Ax + b, \frac{2}{3} I)$.
- We generate a dataset with $D = 20$ dimensions, $N = 1024$ data points.$\mu_{\text{true}} \sim N(0,I)$.
- The analytic solution is $\mu^* = \frac{1}{N} \sum_{i=1}^{N} x^{(n)}, A^* = I/2, b^* = \mu^*/2$



(a) IWAE inference network gradient estimates

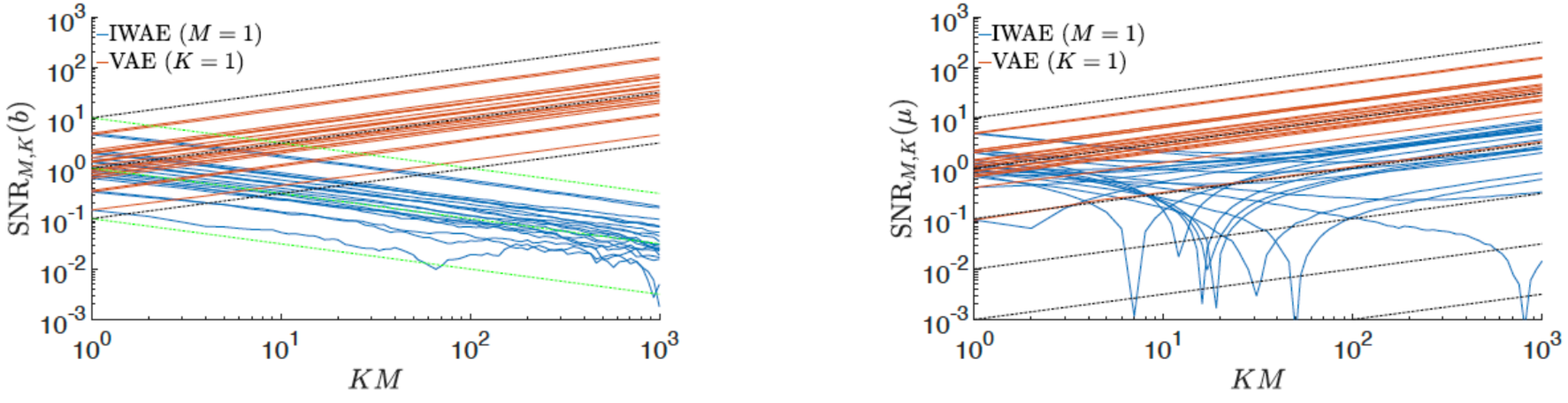


(b) IWAE generative network gradient estimates

Figure 1: Histograms of gradient estimates $\Delta_{M,K}$ for the generative network and the inference network using the IWAE $(M = 1)$ objective with different values of K.

Tighter Variational Bounds are Not Necessarily Better

Empirical Comfirmation



(a) Convergence of SNR for inference network (b) Convergence of SNR for generative network

Figure 2: Convergence of signal-to-noise ratios of gradient estimates with increasing M and K. Different lines correspond to different dimensions of the parameter vectors. Shown in blue is the IWAE where we keep $M = 1$ fixed and increase K. Shown in red is the VAE where $K = 1$ is fixed and we increase M. The black and green dashed lines show the expected convergence rates from our theoretical results, representing gradients of $1/2$ and $-1/2$ respectively.

- The zero point in Figure 2(b) is because $E\Delta_{M,\infty}(\mu)$ may has a opposite sign to $E\Delta_{M,1}(\mu)$. (Corresponding to Figure 1).

Tighter Variational Bounds are Not Necessarily Better

DSNR

- Therefore, the definition of SNR is somewhat problematic for this problem.
- We split each gradient estimate into two component vectors, one parallel o the true gradient and one perpendicular.
- Let $u = E\Delta_{M,K} / ||E\Delta_{M,K}||_2$ be the true normalized gradient direction, then the DSNR is defined by

$$\text{DSNR}_{M,K} = \mathbb{E}\left[\frac{\|\Delta_{\|}\|_2}{\|\Delta_{\perp}\|_2}\right] \quad \text{where}$$

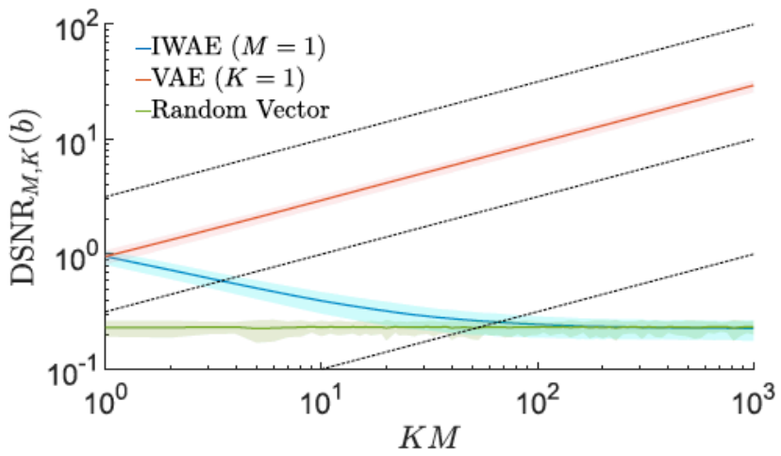
$$\Delta_{\|} = \left(\Delta_{M,K}^T u\right) u \quad \text{and} \quad \Delta_{\perp} = \Delta_{M,K} - \Delta_{\|}.$$

- The perfect estimate of gradients, DSNR $\to \infty$.
- We can find DSNR shows an expected behavior as $K \to \infty$ or $M \to \infty$.

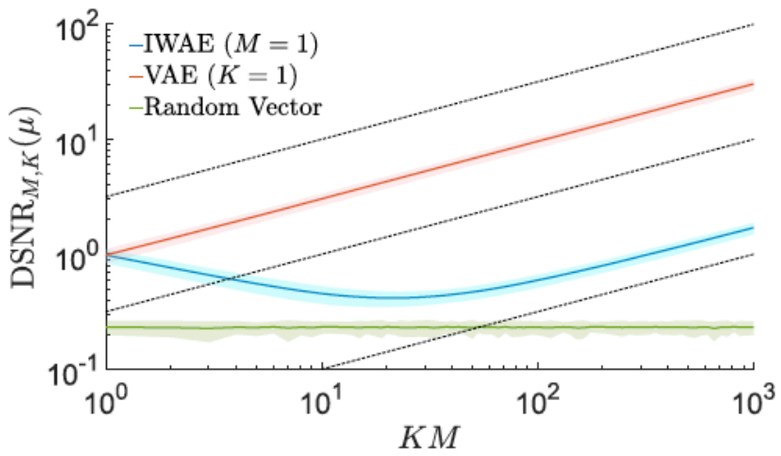
Tighter Variational Bounds are Not Necessarily Better

DSNR

- Increasing K is still detrimental for the inference network by the metric.

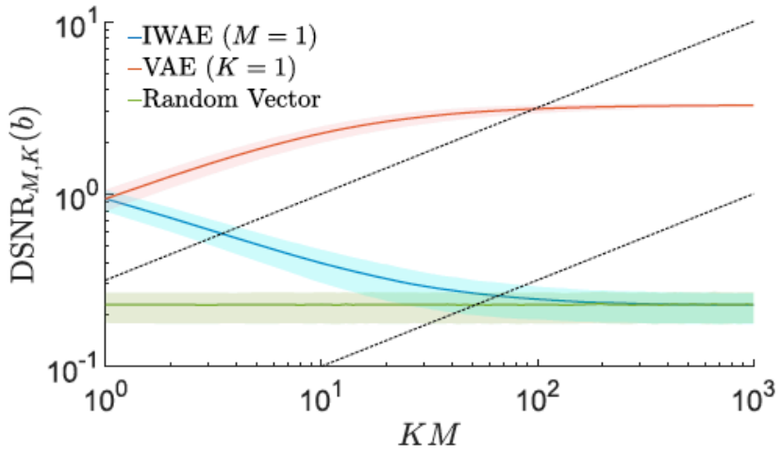


(a) Convergence of DSNR for inference network

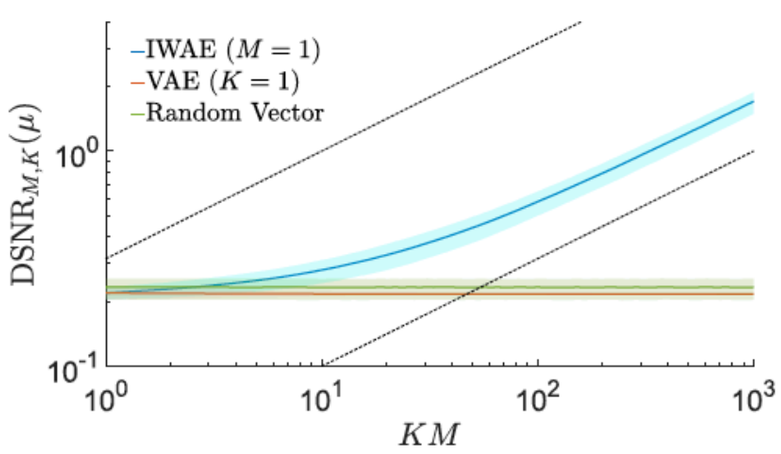


(b) Convergence of DSNR for generative network

Figure 3: Convergence of the directional SNR of gradients estimates with increasing M and K. The solid lines show the estimated DSNR and the shaded regions the interquartile range of the individual ratios. Also shown for reference is the DSNR for a randomly generated vector where each component is drawn from a unit Gaussian.



(a) Convergence of DSNR for inference network



(b) Convergence of DSNR for generative network

Figure 4: Convergence of the DSNR when the target gradient is taken as $u = \mathbb{E}\left[\Delta_{1,1000}\right]$. Conventions as per Figure 3.

Tighter Variational Bounds are Not Necessarily Better

Methods

- To address the issue of diminishing SNR for the inference network, the authors propse three new algorithms.
  - The first is MIWAE. The gradient estimator is ($M > 1, K > 1$)

$$\Delta_{M,K} := \frac{1}{M} \sum_{m=1}^{M} \nabla_{\theta,\phi} \log \frac{1}{K} \sum_{k=1}^{K} w_{m,k}, \qquad (3)$$

$$\text{where } w_{m,k} = \frac{p_\theta(z_{m,k}, x)}{q_\phi(z_{m,k}|x)} \text{ and } z_{m,k} \overset{i.i.d.}{\sim} q_\phi(z_{m,k}|x).$$

- The second, CIWAE, is

$$\text{ELBO}_{\text{CIWAE}} = \beta\, \text{ELBO}_{\text{VAE}} + (1-\beta)\, \text{ELBO}_{\text{IWAE}}$$

with gradient estimator

$$\nabla_{\theta,\phi} \left(\beta \frac{1}{K} \sum_{k=1}^{K} \log w_k + (1-\beta) \log \left(\frac{1}{K} \sum_{k=1}^{K} w_k \right) \right)$$

Tighter Variational Bounds are Not Necessarily Better

Methods

- The third method, PIWAE, uses the IWAE target when training the generative network and MIWAE target for training the inference network. The gradient estimator is

$$\Delta^{\mathrm{C}}_{K,\beta}(\theta) = \nabla_\theta \log \frac{1}{K} \sum\nolimits_{k=1}^{K} w_k$$

$$\Delta^{\mathrm{C}}_{M,K,\beta}(\phi) = \frac{1}{M} \sum\nolimits_{m=1}^{M} \nabla_\phi \log \frac{1}{L} \sum\nolimits_{\ell=1}^{L} w_{m,\ell}$$

Tighter Variational Bounds are Not Necessarily Better

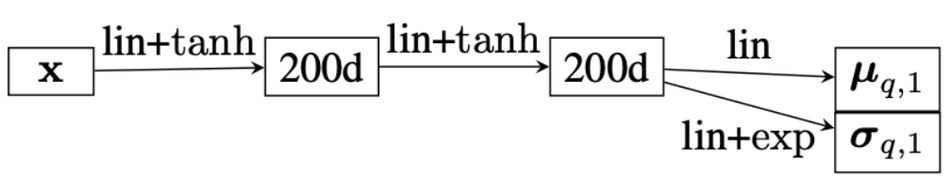
Experiments

- Comparision these estimators to train deep generative models for the MNIST dataset.

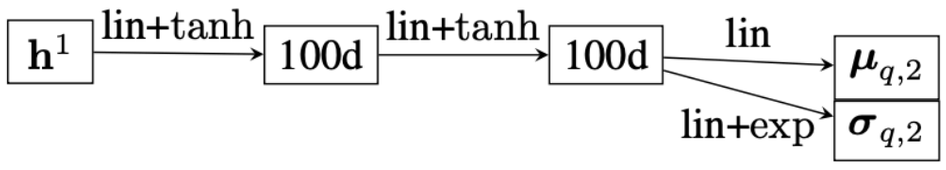
- The network architecture is

Here is a summary of the network architectures used in the experiments:

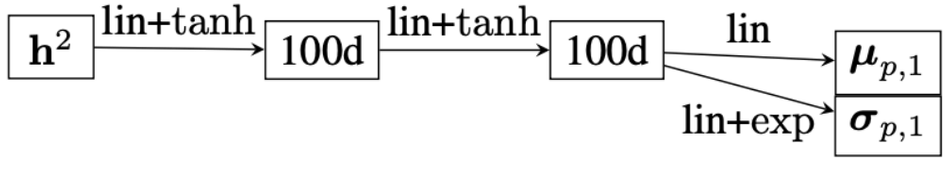
$q(\mathbf{h}^1|\mathbf{x}) = \mathcal{N}(\mathbf{h}^1|\boldsymbol{\mu}_{q,1}, \mathrm{diag}(\boldsymbol{\sigma}_{q,1}))$



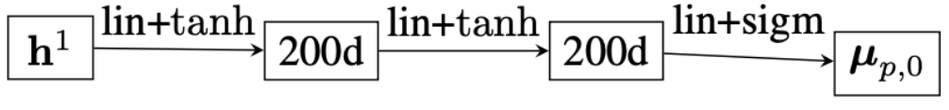
$q(\mathbf{h}^2|\mathbf{h}^1) = \mathcal{N}(\mathbf{h}^2|\boldsymbol{\mu}_{q,2}, \mathrm{diag}(\boldsymbol{\sigma}_{q,2}))$



$p(\mathbf{h}^1|\mathbf{h}^2) = \mathcal{N}(\mathbf{h}^1|\boldsymbol{\mu}_{p,1}, \mathrm{diag}(\boldsymbol{\sigma}_{p,1}))$



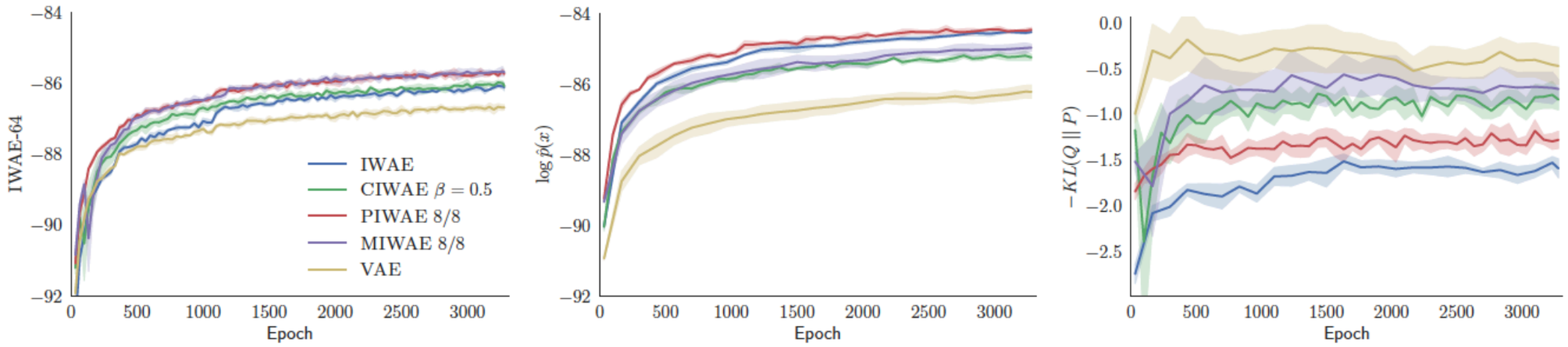
$p(\mathbf{x}|\mathbf{h}^1) = \mathrm{Bernoulli}(\mathbf{x}|\boldsymbol{\mu}_{p,0})$



- IWAE-64: multi-sample lower bound with K=64; log p: multi-sample lower bound with K=5000; KL: the latter minus the former.

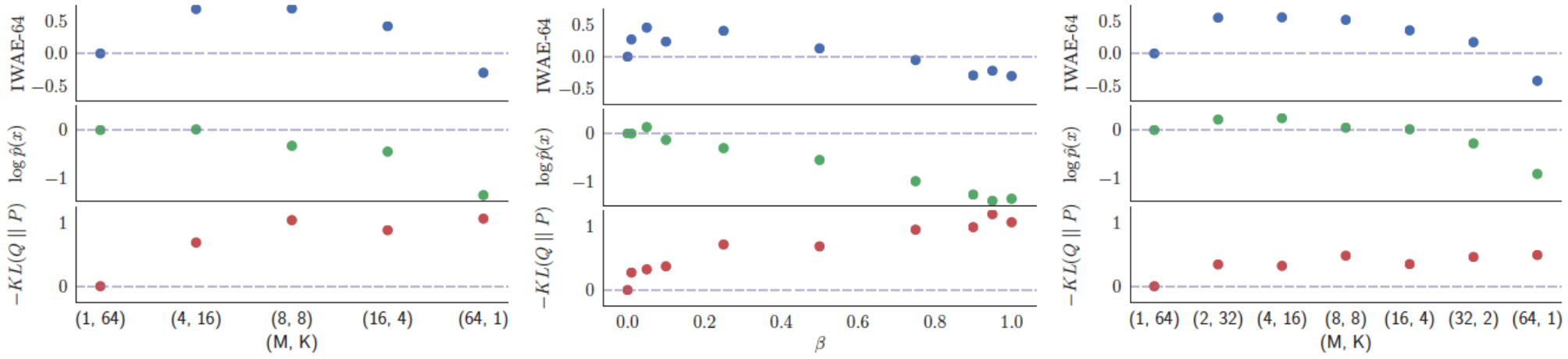
Tighter Variational Bounds are Not Necessarily Better

Experiments



(a) IWAE_{64} (b) $\log \hat{p}(x)$ (c) $-\text{KL}(Q_\phi(z|x)||P_\theta(z|x))$

Figure 5: Convergence of evaluation metrics on the test set with increased training time. All lines show mean ± standard deviation over 4 runs with different random initializations. Larger values are preferable for each plot.



(a) Comparing MIWAE and IWAE (b) Comparing CIWAE and IWAE (c) Comparing PIWAE and IWAE

Figure 6: Test set performance of MIWAE, CIWAE, and PIWAE relative to IWAE in terms of the IWAE-64 (top), $\log \hat{p}(x)$ (middle), and $-\text{KL}(Q_\phi(z|x)||P_\theta(z|x))$ (bottom) metrics. All dots are the difference in the metric to that of IWAE. Dotted line is the IWAE baseline. Note that in all cases, the far left of the plot correspond to settings equivalent to the IWAE.

Tighter Variational Bounds are Not Necessarily Better

Experiments

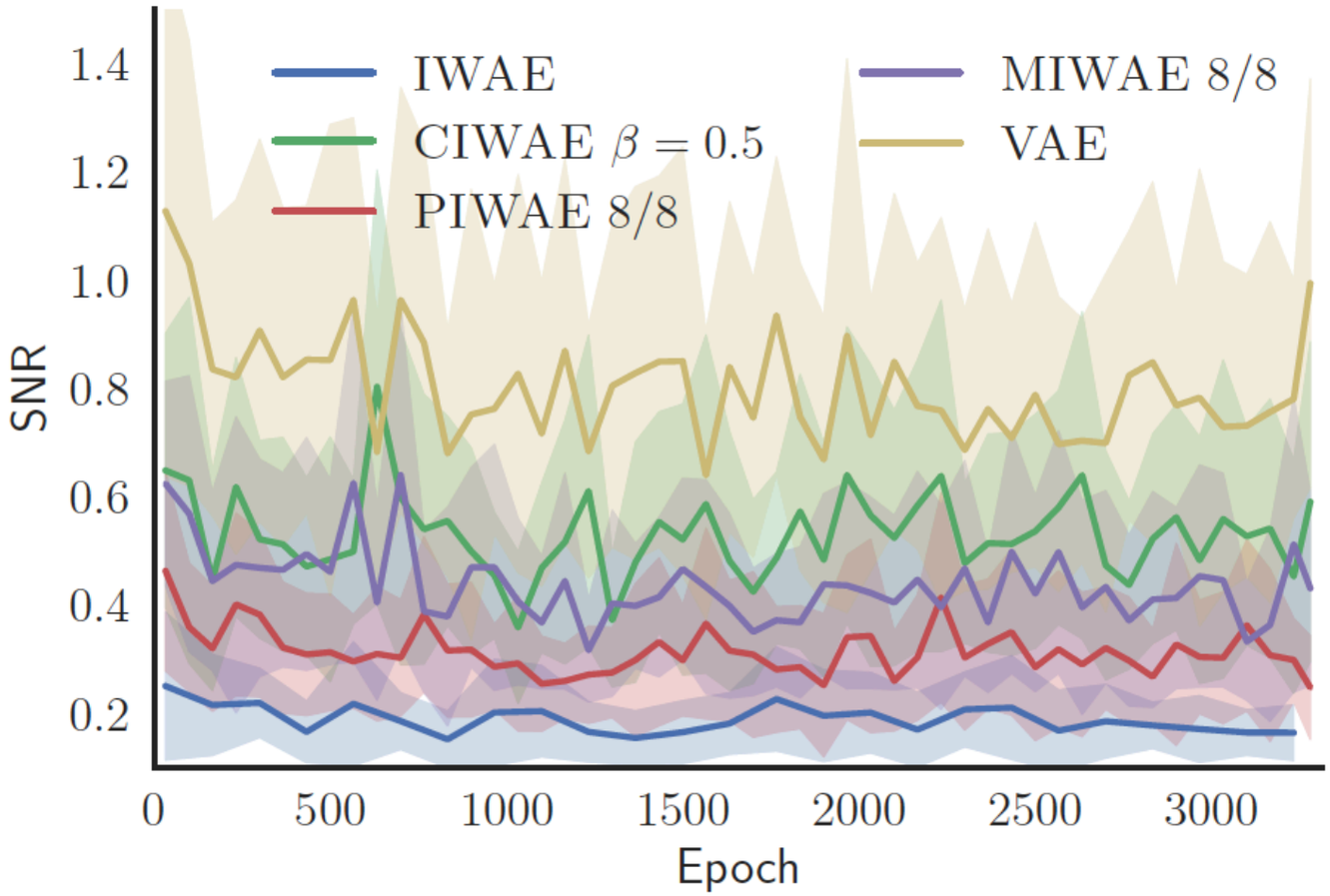


Figure 8: SNR of inference network weights during training. All lines are mean $\pm$ standard deviation over 20 randomly chosen weights per layer.

References

- Mnih, Andriy, and Danilo Rezende. "Variational inference for monte carlo objectives." *International Conference on Machine Learning*. PMLR, 2016.

- Naesseth, Christian, Fredrik Lindsten, and David Blei. "Markovian score climbing: Variational inference with KL (p|| q)." *Advances in Neural Information Processing Systems* 33 (2020): 15499-15510.

- Ou, Z., & Song, Y. (2020, August). Joint stochastic approximation and its application to learning discrete latent variable models. In *Conference on Uncertainty in Artificial Intelligence* (pp. 929-938). PMLR.

- Rainforth, Tom, et al. "Tighter variational bounds are not necessarily better." *International Conference on Machine Learning*. PMLR, 2018.