Variational inference for Monte Carlo objectives
Supplementary Material

Andriy Mnih AMNIH@GOOGLE.COM
Danilo J. Rezende DANILOR@GOOGLE.COM
Google DeepMind

Algorithm for computing VIMCO gradients

Algorithm 1 provides an outline of our implementation of VIMCO gradient computation for a single training case. This version uses the geometric mean to estimate $f(x, h^j)$ from the other $K - 1$ terms. The computations are performed in the log domain for better numerical stability.

Algorithm 1 Compute gradient estimates for the model and proposal distribution parameters for a single observation

Require: x , $K \geq 2$
  for $i = 1$ to K **do**
    $h^i \sim Q(h|x)$
    $l[i] = \log f(x, h^i)$
  end for
  {Compute the multi-sample stochastic bound}
  $\hat{L} = \text{LogSumExp}(l) - \log K$
  {Precompute the sum of $\log f$}
  $s = \text{Sum}(l)$
  {Compute the baseline for each sample}
  for $i = 1$ to K **do**
    {Save the current $\log f$ for future use and replace it}
    {with the average of the other K-1 $\log f$ terms}
    $temp = l[i]$
    $l[i] = (s - l[i])/(K - 1)$
    $\hat{L}^{-i} = \text{LogSumExp}(l) - \log K$
    $l[i] = temp$ {Restore the saved value}
  end for
  $w = \text{SoftMax}(l)$ {Compute the importance weights}
  $\nabla\theta = 0, \nabla\psi = 0$
  {Sum the gradient contributions from the K samples}
  for $i = 1$ to K **do**
    {Proposal distribution gradient contributions}
    $\nabla\theta = \nabla\theta + (\hat{L} - \hat{L}^{-i})\nabla_\theta \log Q(h^i|x)$
    $\nabla\theta = \nabla\theta + w[i]\nabla_\theta \log f(x, h^i)$
    {Model gradient contribution}
    $\nabla\psi = \nabla\psi + w[i]\nabla_\psi \log f(x, h^i)$
  end for

Details of the experimental protocol

All models were trained using the Adam optimizer (Kingma & Ba, 2015) with minibatches of size 24. The input to the proposal distribution/inference network was centered by subtracting the mean. For each training method/number of samples combination we trained the model several times using different learning rates, saving the model with the best validation score achieved during each training run. The plots and the scores shown in the paper were obtained from the saved model with the highest validation score. For generative training, we considered the learning rates of $\{3 \times 10^{-4}, 1 \times 10^{-3}, 3 \times 10^{-3}\}$. For the structured output prediction experiments, the learning rates were $\{3 \times 10^{-4}, 1 \times 10^{-3}, 3 \times 10^{-3}\}$ for VIMCO and RWS and $\{1 \times 10^{-4}, 3 \times 10^{-4}, 1 \times 10^{-3}\}$ for NVIL.

Our NVIL implementation used both constant and input-dependent baselines as well as variance normalization. The input-dependent baseline for NVIL was a neural network with one hidden layer of 100 tanh units. VIMCO used the geometric mean for computing the per-sample learning signals.

Effect of variance reduction on the learning signal

As explained in Sections 2.4 and 2.5, the magnitude of the learning signal used for learning the proposal distribution parameters is closely related to the variance of the resulting gradient estimator. Both VIMCO and NVIL aim to reduce the estimator variance by subtracting a baseline from the original learning signal $\hat{L}(h^{1:K})$ in order to reduce its magnitude, while keeping the estimator unbiased. We examined the effectiveness of these two approaches by plotting a smoothed estimate of the magnitude of the resulting learning signal ($\hat{L}(h^j|h^{-j})$ for VIMCO and $\hat{L}(h^{1:K}) - b(x) - b$ for NVIL) as a function of the number of parameter updates when training the SBN on MNIST in Section 5.1. The magnitude of the learning signal was estimated by taking the square root of the mean of the squared signal values

Figure 1. Structured output prediction: Conditional completions generated by sampling from a three-layer SBN trained using VIMCO with the 20-sample objective. The top row shows the original full digit images. The remaining rows combine the top half from the original image with the bottom half generated from the model.

for each minibatch. The results for different numbers of samples shown in Figure 2 suggest that while VIMCO and NVIL are equally effective at reducing variance when using a 2-sample objective, VIMCO becomes much more effective than NVIL when using more than 2 samples. For 10 samples, the average magnitude of the learning signal for VIMCO is about 3 times lower than for NVIL, which suggests almost an order of magnitude lower variance of the gradient estimates.

Structured output prediction: digit completions

Figure 1 shows multiple completions for the same set of top digit image halves generated using a three-layer (200-200-200) SBN trained using VIMCO with the 20-sample objective. The completions were obtained by computing observation probabilities based on a single sample from the prior. The variability of the completions shows how the model captured the multimodality of the data distribution.

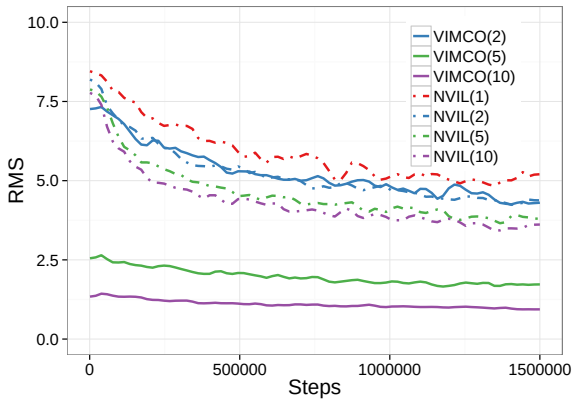

Figure 2. The magnitude (root mean square) of the learning signal for VIMCO and NVIL as a function of the number of samples used in the objective and the number of parameter updates.

Gradient derivation for the multi-sample objective

In this section we will derive the gradient for the multi-sample objective

$$\begin{aligned}\mathcal{L}^K(x) =& E_{Q(h^{1:K}|x)}\left[\hat{L}(h^{1:K})\right] \\ =& E_{Q(h^{1:K}|x)}\left[\log \hat{I}(h^{1:K})\right] \\ =& E_{Q(h^{1:K}|x)}\left[\log \frac{1}{K}\sum_{i=1}^{K} f(x,h^i)\right]. \end{aligned} \tag{1}$$

We start by using the product rule:

$$\begin{aligned}\nabla_\theta \mathcal{L}^K(x) =& \nabla_\theta E_{Q(h^{1:K}|x)}\left[\hat{L}(h^{1:K})\right] \\ =& \nabla_\theta \sum_{h^{1:K}} Q(h^{1:K}|x)\hat{L}(h^{1:K}) \\ =& \sum_{h^{1:K}}\Big[\hat{L}(h^{1:K})\nabla_\theta Q(h^{1:K}|x) + \\ & Q(h^{1:K}|x)\nabla_\theta \hat{L}(h^{1:K})\Big]. \end{aligned} \tag{2}$$

Using the identity $\nabla_\theta g(x) = g(x)\nabla_\theta \log g(x)$, we can express the gradient of $Q(h^{1:K}|x)$ as

$$\begin{aligned}\nabla_\theta Q(h^{1:K}|x) =& Q(h^{1:K}|x)\nabla_\theta \log Q(h^{1:K}|x) \\ =& Q(h^{1:K}|x)\nabla_\theta \log \prod_{j=1}^{K} Q(h^j|x) \\ =& Q(h^{1:K}|x)\sum_{j=1}^{K}\nabla_\theta \log Q(h^j|x). \end{aligned} \tag{3}$$

We use the chain rule along with the same identity to compute the gradient of $\hat{L}(h^{1:K})$:

$$\begin{aligned}\nabla_\theta \hat{L}(h^{1:K}) =& \nabla_\theta \log \frac{1}{K}\sum_{j=1}^{K} f(x,h^j) \\ =& \frac{1}{\sum_{i=1}^{K} f(x,h^i)}\sum_{j=1}^{K}\nabla_\theta f(x,h^j) \\ =& \frac{1}{\sum_{i=1}^{K} f(x,h^i)}\sum_{j=1}^{K} f(x,h^j)\nabla_\theta \log f(x,h^j) \\ =& \sum_{j=1}^{K}\tilde{w}^j \nabla_\theta \log f(x,h^j) \end{aligned} \tag{4}$$

where $\tilde{w}^j \equiv \frac{f(x,h^j)}{\sum_{i=1}^{K} f(x,h^i)}$. Substituting Eq. 3 and Eq. 4 into Eq. 2 we obtain

$$\begin{aligned}\nabla_\theta \mathcal{L}^K(x) =& \sum_{h^{1:K}}\Big(\hat{L}(h^{1:K})Q(h^{1:K}|x)\sum_{j=1}^{K}\nabla_\theta \log Q(h^j|x) + \\ & Q(h^{1:K}|x)\sum_{j=1}^{K}\tilde{w}^j\nabla_\theta \log f(x,h^j)\Big), \\ =& \sum_{h^{1:K}} Q(h^{1:K}|x)\hat{L}(h^{1:K})\sum_{j=1}^{K}\nabla_\theta \log Q(h^j|x) + \\ & \sum_{h^{1:K}} Q(h^{1:K}|x)\sum_{j=1}^{K}\tilde{w}^j\nabla_\theta \log f(x,h^j), \\ =& E_{Q(h^{1:K}|x)}\left[\sum_j \hat{L}(h^{1:K})\nabla_\theta \log Q(h^j|x)\right] + \\ & E_{Q(h^{1:K}|x)}\left[\sum_j \tilde{w}^j\nabla_\theta \log f(x,h^j)\right]. \end{aligned} \tag{5}$$

References

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *ICLR*, 2015.