Variational Inference for Monte Carlo Objectives

Andriy Mnih, Danilo Rezende

June 21, 2016
Variational training of directed generative models has been widely adopted.

- Results depend heavily on the choice of the variational posterior.
- A variational posterior that is too simple can prevent the model from using much of its capacity.
- Making it more expressive is one way to avoid this (see e.g. DRAW, normalizing flows).

- Simpler (orthogonal) alternative: optimize a tighter lower bound on the log-likelihood by throwing more computation at the problem.
  - Burda et al. (2016) used the reparameterization trick to implement this approach in variational autoencoders.

- We develop a more general version than can also handle the harder case of models with discrete latent variables.
  - We use the availability of multiple samples to implement highly effective variance reduction at virtually no additional cost.
Motivation

- Continuous latent variables are not always appropriate.
  - Some properties of the world, such as absence/presence, number of objects, are fundamentally discrete.
  - Dependencies between discrete latent variables can be easier to capture (e.g. DARN).
- Inference-based learning provides a principled way of training deep models without backpropagation.
  - Inferring the latent variable values effectively makes them observed, breaking the flow of gradients through them.
  - An inference network propagates the information contained in the target/output, which is captured by the backpropagated gradient in differentiable/reparameterized models.
Multi-sample objective for variational inference

- The standard variational lower bound on $\log P_\theta(x)$ with a variational posterior $Q(h|x)$:

$$
\mathcal{L}(x) = \mathbb{E}_{Q(h|x)} \left[ \log \frac{P(x, h)}{Q(h|x)} \right] = \log P(x) + \mathbb{E}_{Q(h|x)} \left[ \log \frac{P(h|x)}{Q(h|x)} \right].
$$

- There is a big penalty for having regions with $Q(h|x) \gg P(h|x)$.
- As a result, the learned $Q(h|x)$ provides very incomplete coverage of $P(h|x)$.

- A tighter lower bound on $\log P_\theta(x)$ (IWAE, Burda et al., 2016):

$$
\mathcal{L}^K(x) = \mathbb{E}_{Q(h^1:K|x)} \left[ \log \frac{1}{K} \sum_{k=1}^{K} \frac{P(x, h^k)}{Q(h^k|x)} \right]
$$

- Now we have $K$ shots at hitting a high-probability region of $P(h|x)$.
- The learned $Q(h|x)$ is now less conservative.
Monte-Carlo objectives

- Generalization: Objectives of the form
  \[ \mathcal{L}^K(x) = E_{Q(h^{1:K}|x)} \left[ \log \frac{1}{K} \sum_{k=1}^{K} f(x, h^k) \right], \]
  where \( h^1, \ldots, h^K \) are independent samples from some distribution \( Q(h|x) \).

- Special case: If \( f(x, h) \) is an unbiased Monte Carlo estimator of \( P(x) \), then so is \( \hat{I}(h^{1:K}) = \frac{1}{K} \sum_{k=1}^{K} f(x, h^k) \).
  \[ E_{Q(h^{1:K}|x)} \left[ \log \hat{I}(h^{1:K}) \right] \text{ is a lower bound on } \log P(x). \]
  Can think of \( \log \hat{I}(h^{1:K}) \) as a \textit{stochastic} lower bound on \( \log P(x) \).
  The bound becomes tighter as \( K \) increases, converging to \( \log P(x) \) in the limit.

- \( Q(h|x) \) can be thought of as a proposal distribution as opposed to a variational posterior.
Examples of Monte Carlo objectives

- The simplest case involves Monte Carlo sampling from the prior:

\[
\mathcal{L}_K^r(x) = E_P \left[ \log \frac{1}{K} \sum_{k=1}^{K} P(x | h^k) \right] \text{ with } h^k \sim P(h).
\]

- Importance sampling with a learned proposal is usually much more efficient:

\[
\mathcal{L}_K^r(x) = E_{Q(h_{1:k}|x)} \left[ \log \frac{1}{K} \sum_{k=1}^{K} \frac{P(x, h^k)}{Q(h^k|x)} \right].
\]

- Many other possibilities:
  - Can incorporate variance reduction techniques from IS such as control variates.
  - Can use \(\alpha\)-divergence based objectives.
Gradients of the lower bound

- We would like to maximize the objective

\[ L^K(x) = E_{Q(h^{1:K}|x)} \left[ \log \frac{1}{K} \sum_{k=1}^{K} f(x, h^k) \right] = E_{Q(h^{1:K}|x)} \left[ \log \hat{l}(h^{1:K}) \right]. \]

- Its gradient can be expressed as

\[ \frac{\partial}{\partial \theta} L^K(x) = E_{Q(h^{1:K}|x)} \left[ \sum_j \log \hat{l}(h^{1:K}) \frac{\partial}{\partial \theta} \log Q(h^j|x) \right] + E_{Q(h^{1:K}|x)} \left[ \sum_j \tilde{w}^j \frac{\partial}{\partial \theta} \log f(x, h^j) \right] \]

where \( \tilde{w}^j \equiv \frac{f(x,h^j)}{\sum_{k=1}^{K} f(x,h^k)} \).

- The second term is easy to estimate.
- The first term is much harder.
Estimating the gradients (NVIL-style)

- Can use the Neural Variational Inference and Learning (NVIL) estimator developed for the single-sample variational objective.

- Applying it to the multi-sample objective gives:

\[
\frac{\partial}{\partial \theta} \mathcal{L}^K(x) \simeq \sum_j (\log \hat{I}(h^{1:K}) - b(x)) \frac{\partial}{\partial \theta} \log Q(h^i|x) + \sum_j \tilde{w}^j \frac{\partial}{\partial \theta} \log f(x, h^j),
\]

with \( h^k \sim Q(h|x) \)

- \( b(x) \) is a predictor/baseline trained to predict \( \log \hat{I}(h^{1:K}) \).

- Drawback: uses the same learning signal for all \( h^k \), even though some samples will be much better than others.

- There is no credit assignment within a set of \( K \) samples.
Disentangling the learning signals

- Would like to have a different learning signal for each sample.
- Key observation: since the $K$ samples are independent, when considering the learning signal for one of the samples, can treat all other samples as constant.
  - Can “subtract-out” the effect of the other samples to isolate the effect of the sample of interest.
- What to subtract from $\log \hat{I}(h^{1:K})$?
  - Something very close to it that does not depend on the sample of interest.
- One idea: train a baseline-like predictor for $f(x, h^i)$. 
  - This introduces additional complexity.
  - Can we avoid learning an extra mapping?
Better idea: estimate $f(x, h^i)$ from the other $K - 1$ $f(x, h^k)$.

Two natural choices for estimating $f(x, h^i)$ are:

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

This gives the following local learning signals:

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

The second term can be seen as a hand-crafted sample-dependent baseline with no free parameters.

VIMCO local learning signals work well without any additional variance reduction.
Variance reduction: VIMCO vs. NVIL

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.
Generative modelling: 200-200-200 SBN on MNIST

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.

<table>
<thead>
<tr>
<th>Number of Samples</th>
<th>Training Alg.</th>
<th>VIMCO</th>
<th>NVIL</th>
<th>RWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>—</td>
<td>95.2</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>93.5</td>
<td>93.6</td>
<td>94.6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>92.8</td>
<td>93.7</td>
<td>93.4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>92.6</td>
<td>93.4</td>
<td>93.0</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>91.9</td>
<td>96.2</td>
<td>92.5</td>
<td></td>
</tr>
</tbody>
</table>
Structured prediction with inference (MNIST)
Structured prediction without inference (MNIST)
Conclusions

- A principled, unbiased approach to optimizing multi-sample variational objectives that just works.
  - Implements effective variance reduction essentially for free.
  - Does not need a learned baseline to work well.
- Performs better than NVIL and Reweighted Wake Sleep.
- Can handle both discrete and continuous latent variables and can be combined with the reparameterization trick.
Thank you!
Generative modelling: 200-200-200 SBN on MNIST

![Graph showing the performance of different algorithms over steps]

- **VIMCO(2)**
- **VIMCO(5)**
- **VIMCO(10)**
- **VIMCO(50)**
- **NVIL(1)**
- **NVIL(2)**
- **NVIL(5)**
- **NVIL(10)**
- **NVIL(50)**
Structured prediction: completion examples