Reinforced Variational Inference

Theophane Weber∗1 Nicolas Heess∗1 S. M. Ali Eslami1 John Schulman2 David Wingate3 David Silver1
1 Google DeepMind 2 University of California, Berkeley 3 Brigham Young University

1 Introduction

Recent years have seen an increase in the complexity and scale of probabilistic models used to un-
derstand and analyze data, with a corresponding increase in the difficulty of performing inference. An important enabling factor in this context has been the development of stochastic gradient algo-
rithms for learning variational approximations to posterior distributions. In a separate line of work researchers have been investigating how to use probabilistic inference for the problem of optimal control [5, 6, 12, 17, 25]. By viewing control as an inference problem, they showed that they could ‘borrow’ algorithms from the inference literature (e.g. belief propagation) and turn them into control algorithms. In this work, we do just the opposite: we formally map the problem of learning approx-
imate posterior distributions in variational inference (VI) onto the policy optimization problem in reinforcement learning (RL), explaining this connection at two levels.

We first provide a high level connection, where draws from the approximate posterior (VI) corre-
spond to trajectory samples (RL), free energies (VI) to expected returns (RL), and where the core com-
putation involves computing gradients of expectations. We follow by a more detailed, sequen-
tial mapping where Markov Decision Processes concepts (state, action, rewards and transitions) are
clearly defined in the inference context. We then illustrate how this allows us to leverage ideas from RL for inference network learning, for instance by introducing the concept of value functions in sequential variational inference. For concreteness and simplicity, in the main text we focus on inference for a particular model class and derive the general case in the appendix.

2 Variational Inference

The inference problem: Given a model \( p(z)p(x|z) \) with latent variables \( z \) and observables \( x \) we are interested in the posterior \( p(z|x) \). The exact posterior is intractable for many models of interest so it is common to try and compute an approximate distribution \( q(z|x) \) that is close to the true posterior in the sense detailed below.

Model and approximate posterior: We are interested in the setup where model \( p \) and approx-
imate posterior \( q \) decompose into a product of local conditional distributions. For simplicity and clarity, we here consider the special case of a model in which the dependencies are Markovian (e.g. in an HMM or a multi-layer stochastic neural network); a fully general exposition is found in appendix [B]

\[ p(x, z) = p(z_1)p(z_2|z_1) \ldots p(z_K|z_{K-1})p(x|z_K). \]  

(1)

We choose an approximate posterior that can be factored in a similar way:

\[ q(x|z) = q(z_1|x)q(z_2|z_1, x) \ldots q(z_K|z_{K-1}, x). \]  

(2)

∗equal contribution
Objective function: In variational inference, we aim to maximize the following function:

\[ \mathcal{L}(q) = \int q(z|x) \log \frac{p(x|z)p(z)}{q(z|x)} dz. \] (3)

This objective function, which is known as the negative free energy, can be motivated in two ways: (a) maximizing \( \mathcal{L}(q) \) is equivalent to minimizing the KL divergence between the approximate posterior \( q(z|x) \) and the true posterior \( p(z|x) \). (b) \( \mathcal{L}(q) \) is a lower bound to the data log-likelihood \( \log p(x) \) and therefore maximizing \( \mathcal{L}(q) \) leads to algorithms for optimizing the data log-likelihood.

Stochastic optimization of the objective function: The approximate posterior \( q \) is frequently chosen from some parametric family with parameters \( \theta \) (\( \mathcal{L} \) is now a function of \( \theta \) rather than \( q \)). A Monte Carlo estimate of the gradient of \( \mathcal{L} \) with respect to \( \theta \) can be obtained by using the score function method (see appendix A for details) as follows. For \( z^{(i)} \sim q_{\theta}(\cdot|x) \), we have:

\[
\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E}_{z \sim q_{\theta}} \left[ \frac{\partial}{\partial \theta} \log q_{\theta}(z|x) \right] \left( \log \frac{p(x|z)p(z)}{q_{\theta}(z|x)} \right),
\]

or

\[
\approx \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \theta} \log q_{\theta}(z^{(i)}|x) \left( \log \frac{p(x|z^{(i)})p(z^{(i)})}{q_{\theta}(z^{(i)}|x)} \right). \quad (5)
\]

3 Reinforcement Learning

In RL an agent interacts with an environment in a sequential manner. In each step it observes the state of the environment, executes an action, and receives an instantaneous reward. The agent’s goal is to maximize the expected sum of these rewards.

Objective function: Formally, the goal is to maximize the following function:

\[
J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}} \left[ \sum_{t=1}^{T} r(s_t, a_t) \right] = \int p_{\theta}(\tau) R(\tau) d\tau,
\]

where \( s, a, r \) denote states, actions, and rewards; \( \tau = (s_1, a_1, s_2, \ldots, s_T, a_T) \) denotes trajectories; and \( R(\tau) = \sum_{t=1}^{T} r_t \) is the return over the course of an entire trajectory.

Policy and trajectory distribution: \( p_{\theta}(\tau) \) specifies the distribution over trajectories:

\[
p_{\theta}(\tau) = P(s_1)\pi_{\theta}(a_1|s_1) \prod_{t=2}^{T} P(s_t|s_{t-1}, a_{t-1})\pi_{\theta}(a_t|s_t).
\]

This distribution is a composition of the policy \( \pi_{\theta} \) which is the state-conditional action distribution (with parameters \( \theta \)) that characterizes the agent’s behavior, and of \( P(s_t|s_{t-1}, a_{t-1}) \) which is the transition probability of the Markov decision process (MDP) that models the environment.

Below we will use \( R_t(\tau) = \sum_{t'=t}^{T} r(s_t, a_t) \) for the sum of rewards that follows (and hence is a consequence of) action \( a_t \).

Stochastic optimization of the objective function: A basic Monte Carlo estimate of the gradient of \( J \) in eq. (6) can be obtained as follows (see appendix A for details):

\[
\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\tau \sim p_{\theta}} \left[ \frac{\partial}{\partial \theta} \log p_{\theta}(\tau) R(\tau) \right],
\]

or

\[
\approx \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \theta} \log p_{\theta}(\tau^{(i)}) R(\tau^{(i)}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \frac{\partial}{\partial \theta} \log \pi_{\theta}(a_t^{(i)}|s_t^{(i)}) R(\tau^{(i)}), \quad (9)
\]

where \( \tau \sim p_{\theta} \). This gradient estimate is also known as the REINFORCE algorithm [26]. This estimator has an intuitive interpretation: actions are ‘tried’ and their probabilities are then adjusted to make high-return trajectories more likely. An important aspect of basic REINFORCE is that it does not take advantage of the sequential aspect of the RL problem: it only considers total returns for entire trajectories and does not perform a more fine grained credit assignment to individual actions.
4 Variational Inference as Reinforcement Learning

4.1 The high level perspective: The monolithic inference problem

Maximizing the lower bound $\mathcal{L}$ with respect to the parameters of $q$ can be seen as an instance of REINFORCE where $q$ takes the role of the policy; the latent variables $z$ are actions; and $\log \frac{p(y,x,z)}{q(z|x)}$ takes the role of the return. Equations (3) and (9) then take the same form, as do their gradients in equations (5) and (10), in both cases we are maximizing an expectation $\int p_{\theta}(y)f(y)dy$, with respect to the parameters $\theta$ of the distribution $p_{\theta}(y)$. This connection has been pointed out e.g. in [14].

<table>
<thead>
<tr>
<th>Generic expectation</th>
<th>Optimization var.</th>
<th>Integration var.</th>
<th>Distribution</th>
<th>Integrand</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI</td>
<td>$\theta$</td>
<td>$y$</td>
<td>$p_{\theta}(y)$</td>
<td>$f(y)$</td>
</tr>
<tr>
<td>RL</td>
<td>$\theta$</td>
<td>Trajectory</td>
<td>$p_{\theta}(\tau)$</td>
<td>Total return</td>
</tr>
<tr>
<td></td>
<td>Variational param.</td>
<td>Latent trace</td>
<td>Posterior dist.</td>
<td>$R(\tau)$</td>
</tr>
<tr>
<td></td>
<td>$\tau$</td>
<td>$z$</td>
<td>$q_{\theta}(z</td>
<td>x)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\log \left( \frac{p(x,z)}{q_{\theta}(z</td>
</tr>
</tbody>
</table>

Table 1: High-level connections between VI and RL. Both settings are special cases of the generic problem of optimizing an expectation $\int p_{\theta}(y)f(y)dy$, with respect to the parameters $\theta$ of the distribution $p_{\theta}(y)$.

Unlike in most RL settings, the rewards here depend directly on the (parameters of) the policy rather than just through the state action distribution (but see e.g. the discussion in [2,3,7,13]); in practice, this has no impact on the applicability of RL algorithms to inference in most situations, particularly when using policy networks.

4.2 Breaking it down: Bringing structure of the inference problem

A key idea of reinforcement learning is to exploit the structure of the optimization (often through Markovian properties induced by the sequentiality of the problem) to create more refined versions of naïve algorithms. Similarly, we can exploit structure in the prior $p$ and the posterior $q$ to create structured, sequential variational inference algorithms. Concepts and ideas in reinforcement learning may then translate to new ideas in inference. Consider again the Markovian model from equations (1) and (2). The variational lower bound can be decomposed as follows:

$$\mathcal{L}(q) = \mathbb{E}_z \left[ \log \frac{p(z_1)}{q(z_1|x)} + \log \frac{p(z_2|z_1)}{q(z_2|x,z_1)} + \cdots + \log \frac{p(z_K|z_{K-1})}{q(z_K|x,z_{K-1})} + \log p(x|z_K) \right]$$

$$= \mathbb{E}_z \left[ \sum_{k=1}^{K} r_k(z_k, z_{k-1}, x) + r_f(z_K, x) \right],$$

where $r_k(z_k, z_{k-1}, x) = \log \left( \frac{p(z_k|z_{k-1})}{q(z_k|z_{k-1}, x)} \right)$ can be regarded as instantaneous state-dependent reward from step $k$, and $r_f(z_K, x) = \log p(x|z_K)$ is the final reward. Recall that $R_k = \sum_{k'=k}^{K} r_k + r_f$ is the return from step $k$. See table 2 for a precise mapping between variational inference and RL for this problem. Note a few particularities of the MDP derived from our variational problem:

- The variational inference MDP (VIMDP) structure depends on the structure of the posterior distribution (e.g. the order in which variables are sampled in $q(z|x)$).
- When performing armortized inference (computing a parametric mapping from datapoint $x$ to posterior $q(z|x)$), the datapoint shows up in the VIMDP as a context $x$. Having a context (a part of the state which is random for each episode but stays constant throughout the episode) is less commonly done in RL (but has no impact on applicability of algorithms).
- The state is composed of the constant context $x$ and the dynamic part of the state $z_k$. For a more complex model and posterior, the state would be composed of the context, and some function of the history of actions (i.e. latent variables) taken so far (see appendix B).
- The state transition for VI is a deterministic function of the current state and action; the stochasticity comes from the choice of the action. In RL, the environments are often stochastic themselves. Again, this has no practical impact.
Table 2: Fine-grained connections between variational inference and reinforcement learning.

<table>
<thead>
<tr>
<th>Context</th>
<th>RL</th>
<th>VIMDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic state</td>
<td>$s_t$</td>
<td>$z_{k-1}$</td>
</tr>
<tr>
<td>State</td>
<td>$s_t$</td>
<td>$(z_{k-1}, x)$</td>
</tr>
<tr>
<td>Action</td>
<td>$a_t$</td>
<td>$z_k \sim q_\theta(z_k</td>
</tr>
<tr>
<td>Transition</td>
<td>$(s_t, a_t) \rightarrow s_{t+1} \sim P(s</td>
<td>s_t, a_t)$</td>
</tr>
<tr>
<td>Instant reward</td>
<td>$r_t$</td>
<td>$\log p(z_k</td>
</tr>
<tr>
<td>Final reward</td>
<td>0</td>
<td>$\log p(x</td>
</tr>
</tbody>
</table>

4.3 Exploiting structure in the inference problem using techniques from RL

This notation makes the connection to reinforcement learning more stringent and allows us to leverage approaches developed in RL to mitigate the high variance of the REINFORCE estimator:

**Variance reduction with baselines:** Two simple insights can reduce the variance of $[9]$: (a) Only rewards that succeed an action (and thus have been caused by it) are informative wrt. that action. (b) We can compare the sampled returns to a reference (or baseline). The result is the following estimate for the gradient:

$$
\frac{\partial L(q_\theta)}{\partial \theta} = E_z \left[ \sum_{k=1}^{K} \frac{\partial}{\partial \theta} \log q_\theta(z_k|z_{k-1}, x) \left( R_k - b_k(z_{k-1}) \right) \right],
$$

(12)

where $b_k$ is an arbitrary function of the latent variable $z_{k-1}$ (note that it must not depend on any $z_{\geq k}$). It is often a learned function. Both modifications leave the gradient estimate unchanged in expectation (this is due to the fact that the integral of a grad log-probability density function is always 0, see appendix A) but affect its variance. One intuitive and convenient choice for $b_k$ is an approximation to the value function (see $[22]$) which is defined as the future expected return from time $k$ in state $z$ as $V_k(x, z) = E_{q_\theta(z_{k,K}|z_k=x,x)} [R_{k+1}(z_k, \ldots, z_K, x)]$.

**Variance reduction with value functions:** State-value functions aim to summarize the average future return incurred at step $k$ in a state $z_k$ given that the choices for future $z_{k'>k}$ follow the policy $q_\theta$. For eq. $[11]$ we can recursively express the value function as follows:

$$
V_\theta^\rho(x, z_k) = E_{z_{k+1} \sim q(z_{k+1}|z_k, x)} \left[ r_k(z_{k+1}, z_k, x) + V_\theta^\rho(z_{k+1}) \right],
$$

(13)

$$
V_\rho^\beta(x, z_K) = \log p(x|z_K).
$$

(14)

The gradient in eq. $[12]$ can now be rewritten as:

$$
\frac{\partial L(q_\theta)}{\partial \theta} = \sum_{k=1}^{K} \frac{\partial}{\partial \theta} \log q(z_k|x, z_{k-1}) E \left[ \underbrace{r_k(z_k, z_{k-1}, x)}_{\text{"action value"}} + V_\theta^\rho(x, z_k) - V_\theta^{\beta}(x, z_{k-1}) \right].
$$

(15)

In practice we do not know $V^\rho$ but we can learn an approximation $\hat{V}^\phi$ with parameters $\phi$. In the simplest case this is achieved by "regression on returns", i.e. we minimize $E_{q(z|x)} [(R_{k+1} - \hat{V}^\phi(x, z_k))^2]$, but it can also be achieved by a bootstrapped regression, akin to temporal difference learning in RL (e.g. $[22]$).

5 Conclusion

In this abstract we have provided a new view of inference as reinforcement learning. We hope this will provide inspiration to VI practitioners to create new inference techniques inspired by reinforcement learning. We exemplify this with two specific strategies (baselines and value functions). Many other concepts in RL can in principle be used in variational inference, such as temporal difference methods or exploration, and we hope to show relevance of those ideas in future work. In this abstract we have focused on the score function estimator which makes no assumption about differentiability, but a similar mapping applicable to and can be combined with differentiable models and associated techniques $[9, 21, 24]$. 

4
References


A Score Function Estimator

The score function trick simply uses the equality $\frac{d \log p_\theta(y)}{d \theta} = \frac{dp_\theta(y)}{p_\theta(y)}$ to rewrite an integral as an expectation. Assuming that $\frac{dp_\theta(y)}{d \theta}$ exists and is continuous, we have

$$
\frac{d}{d \theta} \int p_\theta(y)f(y)dy = \int \frac{dp_\theta(y)}{d \theta} f(y)dy
$$

$$= \int \frac{d \log p_\theta(y)}{d \theta} p_\theta(y)f(y) = E_{p_\theta(y)} \left[ \frac{d \log p_\theta(y)}{d \theta} f(y) \right].
$$

(16)

By using the appropriate $y$, $p_\theta(y)$ and $f(y)$ we recover the REINFORCE estimator for RL and score function for variational inference. Also note that if $f(y)$ is a constant $f$ with respect to $y$, then the expression is clearly 0, since the integral evaluates to the constant $f$.

B General Case

In this appendix, we will detail a general graphical model extension of the Markov chain framework considered in section 4.2. We start by a sequential point of view, leveraging some but not all the structure found in $p$ and $q$, and follow by the more complex graph point of view, using the conceptual framework of stochastic computation graphs (SCG) [21].

B.1 Sequential view

Let us assume that both $x$ and $z$ are multi-dimensional (with respective dimensions $M$ and $K$) and that the $x_j$ and $z_i$ can be arranged into a directed acyclic graph $G^p$. Because $x$ and $z$ play very similar roles in what follows, for simplicity of notation, let $y_j$ represent a variable that could be a single latent $z_j$ or $x_j$ (for instance, we could have $y_j = z_j$ for $j = 1 \ldots K$ and $y_{K+j} = x_j$ for $j = 1 \ldots M$). We can write:

$$p(x, z) = \prod_k p(z_k | h(z_k)) \prod_j p(x_j | h(x_j)),$$

(17)

where $h(y_j)$ represents the subset of $(z, x)$ that form the immediate parents of the variable $y_j$ in the graph $G^p$. It follows that $\log p(x, z)$ can be decomposed as a sum of functions $r_j(s_j) = \log p(y_j | h(y_j))$, with $s_j = (y_j, h(y_j))$.

It is natural to decompose $q$ into a product of conditional distributions, analogously to $p$. Arranging the $z_i$ into a directed acyclic graph $G^q$ we can write

$$q(z_1, \ldots, z_K | x) = q(z_1 | \tilde{h}_1)q(z_2 | \tilde{h}_1)q(z_3 | \tilde{h}_2) \ldots q(z_K | \tilde{h}_K) = \prod_{k=1}^K q(z_k | \tilde{h}_k),
$$

(18)

where $\tilde{h}_i$ denote the parents of $z_i$ according to $G^q$ (note that in order for inference to take the data $x$ into account, the posterior histories $\tilde{h}$ will typically include $x$ or a subset of it). The log posterior $\log q$ defines a second set of reward functions $r_k(\tilde{s}_k) = - \log q_\theta(z_k | \tilde{h}_k, x)$.

Note that the structure of dependencies between variables according to $G^p$ and $G^q$ need not to be identical. Furthermore, the sampling order for $G^q$ is arbitrary as long as no variable is sampled before its parents. We choose an ordering of variables in $G^q$ consistent with the partial ordering imposed by the graph, and suppose latent variables will be sampled sequentially in that order.

Using the notation defined above, the variational objective function is

$$\mathcal{L}(\theta) = E_{z \sim q_\theta} \left[ \sum_j r_j(s_j) + \sum_k \tilde{r}_k(\tilde{s}_k) \right].
$$

(19)

The equation above does not make clear the sequential nature of reward accumulation, but it can be made apparent with a simple rule: if, after sampling the first $k$ variables $z_1, \ldots, z_k$, a particular
reward function can be computed, that corresponding reward is added to the instantaneous reward at time \( k \). Formally, for each \( j \), define \( t(j) = \min\{ k : s_j \subset \{ z_1, z_2, \ldots, z_k \} \} \). Then, letting

\[
r_k(z_1, \ldots, z_k) = \sum_{j:t(j)=k} r_j(s_j) + \tilde{r}_k(\tilde{s}_k),
\]

we obtain:

\[
\mathcal{L}(\theta) = \mathbb{E}_{z_1, z_2, z_3, \ldots, z_K \sim q_\theta(z)} \left[ \sum_k r_k \right].
\]

The formal MDP state at step \( k \) is the latent history \( H_k = (z_1, \ldots, z_k) \); it’s a growing set, but the policy does not have to depend on the entire policy directly (it can for instance use a compressed history encoded with a recurrent neural network); in other words, we can make the MDP a POMDP by making the observation be a subset or function of the actual MDP state. Furthermore, at time \( k \), any \( z_t \) which does not participate either in future reward \( r_{>k} \) or conditional probability \( q_\theta(z_{>k}\mid h(z_{>k})) \) can be removed from the state. For finite order Markov models, this implies the MDP state is always finite dimensional (as in the main body of this abstract).

B.2 Structured view

We can further refine our model by not seeing the latents as sequential, but instead by only considering the graph structure induced by \( \mathcal{G}^q \). This is made easier by modifying the posterior graph \( \mathcal{G}^q \) into a stochastic computation graph \( S^q \). In order to do so, we simply add reward nodes to the graph: each reward \( r_j \) (resp. \( \tilde{r}_k \)) becomes a node, and is a deterministic function of its variable set \( s_j \) (resp. \( \tilde{s}_k \)). To further simplify notation, we no longer make a distinction between each reward function can be computed, that corresponding reward is added to the instantaneous reward at time \( k \). Formally, for each \( j \), define \( t(j) = \min\{ k : s_j \subset \{ z_1, z_2, \ldots, z_k \} \} \). Then, letting

\[
r_k(z_1, \ldots, z_k) = \sum_{j:t(j)=k} r_j(s_j) + \tilde{r}_k(\tilde{s}_k),
\]

we obtain:

\[
\mathcal{L}(\theta) = \mathbb{E}_{z_1, z_2, z_3, \ldots, z_K \sim q_\theta(z)} \left[ \sum_k r_k \right].
\]

The stochastic gradient for the variational cost is given by:

\[
\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} = \sum_{v \in \mathcal{G}^q} \mathbb{E}_{z \sim q_\theta} \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v \mid h(z_v)) \left( \sum_{r \in \mathcal{R}} r \right) \right].
\]

We now explore how to exploit the graph structure. For each latent variable \( z_v \) in our stochastic computation graph \( S^q \), another node \( n \) (either a latent variable or a reward node) is called a non-descendant of \( z_v \) if it can be computed without sampling \( z_v \) first (in other words, \( h(n) \) does not include any variable which depends, directly or indirectly, stochastically or deterministically, on \( z_v \)). Otherwise, it is called a descendant node. Intuitively, descendant nodes of \( z_v \) are ‘downstream’ from \( z_v \) in the computation graph. Also, if \( n \) is a descendant of \( m \), we say that \( m \) is an ancestor of \( n \) (ancestors always include parents). We let \( C(z_v) \) be the set of all nondescendants of \( z_v \), \( D(z_v) \) the set of descendants of \( z_v \), and \( D^+(z_v) = (z_v, D(z_v)) \). Note that \( C(z_v) \cup D^+(z_v) \) is the set of all latent variables. For any \( r \) an element of \( C(z_v) \)

\[
\mathbb{E}_{z \sim q_\theta} \left( \frac{\partial}{\partial \theta} \log q_\theta(z_v \mid h(z_v)) r(s) \right) = \mathbb{E}_{C(z_v)} \left( \frac{\partial}{\partial \theta} \log q_\theta(z_v \mid h(z_v)) r(s) \right)
\]

\[
= \mathbb{E}_{C(z_v)} \left( r(s) \mathbb{E}_{D^+(z_v) \mid C(z_v)} \left( \frac{\partial}{\partial \theta} \log q_\theta(z_v \mid h(z_v)) \right) \right)
\]

\[
= 0,
\]

where the second equality comes from the fact that \( r(s) \) is a deterministic function of \( C(z_v) \), and the third equality comes from the fact that \( z_v \) is independent from its nondescendants given its parents and from the following:

\[
\mathbb{E}_{D^+(z_v) \mid C(z_v)} \left( \frac{\partial}{\partial \theta} \log q_\theta(z_v \mid h(z_v)) \right) = \mathbb{E}_{z_v \mid h(z_v)} \left( \frac{\partial}{\partial \theta} \log q_\theta(z_v \mid h(z_v)) \right)
\]

\[
= \frac{\partial}{\partial \theta} \mathbb{E}_{z_v \mid h(z_v)} q_\theta(z_v \mid h(z_v)) = \frac{\partial}{\partial \theta} 1 = 0.
\]
We conclude that any \( r \) which is not a descendant of \( z \) can be excluded from the gradient term corresponding to \( z \). Letting \( R(z_v) = \sum_{r \in \mathcal{R} \cap \mathcal{D}(z_v)} r \), we obtain the stronger form of the stochastic gradient:

\[
\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} = \sum_{v \in q^*} \mathbb{E}_{z \sim q_\theta} \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) R(z_v) \right].
\]  

This formula is the generalization of (12), without baseline or value function. In the next section, we see how to introduce those notions for the stochastic computation graph framework.

### B.3 Value functions and critics for Stochastic Computation Graphs

Consider again equation (25), and let use total expectation again, but this time with the set of \( z_v \), and its parents \( h(z_v) \). Let \( \mathcal{O}(z_v) \) be the set of variables other than \( z_v \) and its parents. Also, let \( V(h(z_v)) \) be an arbitrary function of the parents of \( z_v \). We have:

\[
\mathbb{E}_z \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) R(z_v) \right] = \mathbb{E}_{h(z_v)} \left[ \mathbb{E}_{z_v|h(z_v)} \left[ \mathbb{E}_{\mathcal{O}(z_v)|z_v,h(z_v)} \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) R(z_v) \right] \right] \right] \\
= \mathbb{E}_{h(z_v)} \left[ \mathbb{E}_{z_v|h(z_v)} \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) \mathbb{E}_{\mathcal{O}(z_v)|z_v,h(z_v)} [R(z_v)] \right] \right] \\
= \mathbb{E}_{h(z_v)} \left[ \mathbb{E}_{z_v|h(z_v)} \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) Q(z_v, h(z_v)) \right] \right] \\
= \mathbb{E}_{h(z_v)} \left[ \mathbb{E}_{z_v|h(z_v)} \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) \left( Q(z_v, h(z_v)) - V(h(z_v)) \right) \right] \right] \\
= \mathbb{E}_z \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) \left( Q(z_v, h(z_v)) - V(h(z_v)) \right) \right],
\]

where we have defined the critic:

\[
Q(z_v, h(z_v)) = \mathbb{E}_{\mathcal{O}(z_v)|z_v,h(z_v)} [R(z_v)].
\]  

The first equality follows the law of total expectation, used twice. The second equality follows from the fact that conditional on \( z_v, h(z_v) \), \( \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) \) is a constant. The third equality defines from the definition of the critic. The fourth follows from the fact that

\[
\mathbb{E}_{z_v|h(z_v)} \left[ \frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) V(h(z_v)) \right] = 0,
\]  

using an identical argument to last section. We finish by suggesting how to evaluate the critic by regression on return: suppose we have a function approximator \( Q_\phi(z_v, h(z_v)) \) and we want to minimize the weighted squared error

\[
\mathbb{E}_{z_v,h(z_v) \sim q_\theta} \left[ (Q_\phi(z_v, h(z_v)) - \mathbb{E}_{\mathcal{O}(z_v)|z_v,h(z_v)} [R(z_v)])^2 \right].
\]  

The gradient with respect to \( \phi \) is

\[
\mathbb{E}_{z_v,h(z_v) \sim q_\theta} \left[ \frac{\partial}{\partial \phi} Q_\phi(z_v, h(z_v)) (Q_\phi(z_v, h(z_v)) - \mathbb{E}_{\mathcal{O}(z_v)|z_v,h(z_v)} [R(z_v)]) \right],
\]  

where \( (z_v, h(z_v)) \) is sampled from \( q_\theta(z_v, h(z_v)) \). Since \( Q_\phi(z_v, h(z_v)) \) is a constant of \( (z_v, h(z_v)) \), this is also equal to:

\[
\mathbb{E}_{z_v,h(z_v) \sim q_\theta} \left[ \mathbb{E}_{\mathcal{O}(z_v)|z_v,h(z_v)} \left[ \frac{\partial}{\partial \phi} Q_\phi(z_v, h(z_v)) (Q_\phi(z_v, h(z_v)) - R(z_v)) \right] \right],
\]  

which is finally equal to

\[
\mathbb{E}_{z \sim q_\theta} \left[ \frac{\partial}{\partial \phi} Q_\phi(z_v, h(z_v)) (Q_\phi(z_v, h(z_v)) - R(z_v)) \right],
\]

which can simply be computed by forward sampling from \( q_\theta(z) \).