Reinforced Variational Inference

Theophane Weber[*1] **Nicolas Heess**[*1] **S. M. Ali Eslami**[1] **John Schulman**[2] **David Wingate**[3] **David Silver**[1]
[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 understand 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 algorithms 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 approximate 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) correspond to trajectory samples (RL), free energies (VI) to expected returns (RL), and where the core computation involves computing gradients of expectations. We follow by a more detailed, sequential 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.

We provide background on variational inference and reinforcement learning in Secs. 2 and 3. We then focus on the connection between the two frameworks in Sec. 4 and conclude with what entails from this connection in Sec. 5.

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 approximate 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.

$$p(x, z) = p(z_1)p(z_2|z_1)\dots p(z_K|z_{K-1})p(x|z_K). \tag{1}$$

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

$$q(x|z) = q(z_1|x)q(z_2|z_1, x)\dots q(z_K|z_{K-1}, x). \tag{2}$$

[*] equal contribution

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

$$\mathcal{L}(q) = \int q(z|x) \log \frac{p(x|z)p(z)}{q(z|x)} \mathrm{d}z. \tag{3}$$

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 θ ($\mathcal{L}$ is now a function of θ rather than q). A Monte Carlo estimate of the gradient of $\mathcal{L}$ with respect to θ 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:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E}_{z\sim q_\theta}\left[\frac{\partial}{\partial \theta} \log q_\theta(z|x) \left(\log \frac{p(x|z)p(z)}{q_\theta(z|x)}\right)\right] \tag{4}$$

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

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:

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[\sum_{t=1}^{T} r(s_t, a_t)\right] = \int p_\theta(\tau) R(\tau) \mathrm{d}\tau, \tag{6}$$

where s, a, r denote states, actions, and rewards; $\tau = (s_1, a_1, s_2, a_2, \ldots, s_T, a_T)$ 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:

$$p_\theta(\tau) = P(s_1)\pi_\theta(a_1|s_1) \prod_{t=2}^{T} P(s_t|s_{t-1}, a_{t-1})\pi_\theta(a_t|s_t). \tag{7}$$

This distribution is a composition of the policy π_θ which is the state-conditional action distribution (with parameters θ) 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):

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\tau\sim p_\theta}\left[\frac{\partial}{\partial \theta} \log p_\theta(\tau) R(\tau)\right] \tag{8}$$

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

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 probabilites 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 θ 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 $\log \frac{p_\theta(x,z_i)}{q_\theta(z_i|x)}$ takes the role of the return. Equations (3) and (6) then take the same form, as do their gradients in equations (5) and (9): in both cases we are maximizing an expectation $\int p_\theta(y) f(y) \mathrm{d}y$, with respect to the parameters θ of the distribution $p_\theta(y)$. This connection has been pointed out e.g. in [14].

Generic expectation		**RL**		**VI**	
Optimization var.	θ	Policy param.	θ	Variational param.	θ
Integration var.	y	Trajectory	τ	Latent trace	z
Distribution	$p_\theta(y)$	Trajectory dist.	$p_\theta(\tau)$	Posterior dist.	$q_\theta(z\|x)$
Integrand	$f(y)$	Total return	$R(\tau)$	Free energy	$\log\left(\frac{p(x,z)}{q_\theta(z\|x)}\right)$

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

$$\mathcal{L}(q) = \mathbb{E}_z \left[\log \frac{p(z_1)}{q(z_1|x)} + \log \frac{p(z_2|z_1)}{q(z_2|x,z_1)} + \cdots + \log \frac{p(z_K|z_{K-1})}{q(z_K|x,z_{K-1})} + \log p(x|z_K) \right] \tag{10}$$

$$= \mathbb{E}_z \left[\sum_{k=1}^{K} r(z_k, z_{k-1}, x) + r_f(z_K, x) \right], \tag{11}$$

where $r_k(z_k, z_{k-1}, x) = \log(p(z_k|z_{k-1})/q(z_k|z_{k-1}, x))$ 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.

	RL	**VIMDP**
Context	—	x
Dynamic state	s_t	z_{k-1}
State	s_t	(z_{k-1}, x)
Action	a_t	$z_k \sim q_\theta(z_k \vert z_{k-1}, x)$
Transition	$(s_t, a_t) \to s_{t+1} \sim P(s \vert s_t, a_t)$	$((z_{k-1}, x), z_k) \to (z_k, x)$
Instant reward	r_t	$\log\left(\frac{p(z_k \vert z_{k-1}, x)}{q_\theta(z_k \vert z_{k-1}, x)}\right)$
Final reward	0	$\log p(x \vert z_K)$

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:

$$\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} = \mathbb{E}_z \left[\sum_{k=1}^{K} \frac{\partial}{\partial \theta} \log q_\theta(z_k \vert z_{k-1}, x) \left(R_k - b_k(z_{k-1})\right) \right], \tag{12}$$

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 $V_k(x, z) = \mathbb{E}_{q_\theta(z_{k:K} \vert z_k = z, x)} \left[R_{k+1}(z_k, \ldots, z_K, x)\right]$.

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_θ. For eq. (11) we can recursively express the value function as follows:

$$V_k^\theta(x, z_k) = \mathbb{E}_{z_{k+1} \sim q(z_{k+1} \vert x, z_k)} \left[r(z_{k+1}, z_k, x) + V_{k+1}^\theta(z_{k+1}) \right], \tag{13}$$

$$V_K^\theta(x, z_K) = \log p(x \vert z_K). \tag{14}$$

The gradient in eq. (12) can now be rewritten as:

$$\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} = \sum_{k=1}^{K} \frac{\partial}{\partial \theta} \log q(z_k \vert x, z_{k-1}) \mathbb{E} \left[[\underbrace{r_k(z_k, z_{k-1}, x) + V_k^\theta(x, z_k)}_{\text{“action value”}} - \underbrace{V_{k-1}^\theta(x, z_{k-1})}_{\text{“baseline”}} \right]. \tag{15}$$

In practice we do not know V^θ but we can learn an approximation $\hat{V}^\phi$ with parameters ϕ. In the simplest case this is achieved by “regression on returns”, i.e. we minimize $\mathbb{E}_{q(z \vert x)} \left[(R_{k+1} - V^\phi(x, z_k))^2 \right]$, 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].

References

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple Object Recognition with Visual Attention. In *ICLR'15*. 2015.

[2] Leemon Baird and Andrew W Moore. Gradient descent for general reinforcement learning. *Advances in Neural Information Processing Systems*, pages 968–974, 1999.

[3] Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res. (JAIR)*, 15:319–350, 2001.

[4] David M Blei, Michael I Jordan, and John W Paisley. Variational bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1367–1374, 2012.

[5] Peter Dayan and Geoffrey E Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

[6] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[7] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[8] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1242–1250, 2014.

[9] Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Gradient Learning Continuous Control Policies by Stochastic Value Gradients. *Proceedings of the 2015 conference on Neural Information Processing Systems*, 2015.

[10] Geoffrey E. Hinton, Brian Sallans, and Zoubin Ghahramani. A Hierarchical Community of Experts. In Michael I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic, 1997.

[11] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[12] Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

[13] Peter Marbach and John N Tsitsiklis. Simulation-based optimization of Markov reward processes. *Automatic Control, IEEE Transactions on*, 46(2):191–209, 2001.

[14] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1791–1799, 2014.

[15] Gerhard Neumann. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 817–824, 2011.

[16] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 814–822, 2014.

[17] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 3052–3056. AAAI Press, 2013.

[18] Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1278–1286, 2014.

[19] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.

[20] Tim Salimans. Markov chain Monte Carlo and variational inference: Bridging the gap. *NIPS 2014 Workshop on Advances in Variational Inference*, 2014.

[21] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient Estimation Using Stochastic Computation Graphs. *Proceedings of the 2015 conference on Neural Information Processing Systems*, 2015.

[22] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, 1998.

[23] Emanuel Todorov. General duality between optimal control and estimation. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4286–4292. IEEE, 2008.

[24] Emanuel Todorov and Weiwei Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, 2005. Proceedings of the 2005*, pages 300–306. IEEE, 2005.

[25] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In *Proceedings of the 23rd international conference on Machine learning*, pages 945–952. ACM, 2006.

[26] Ronald J . Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[27] David Wingate and Theophane Weber. Automated variational inference in probabilistic programming. *NIPS 2012 Workshop on Probabilistic Programming*, 2013.

[28] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

A Score Function Estimator

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

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\theta}\int p_\theta(y)f(y)\mathrm{d}y &= \int_y \frac{\mathrm{d}p_\theta(y)}{\mathrm{d}\theta}f(y)dy \\ &= \int_y \frac{\mathrm{d}\log p_\theta(y)}{\mathrm{d}\theta}p_\theta(y)f(y) = \mathbb{E}_{p_\theta(y)}\left[\frac{\mathrm{d}\log p_\theta(y)}{\mathrm{d}\theta}f(y)\right]. \end{aligned} \tag{16}$$

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 $\mathcal{G}^p$. 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 $j = 1 \dots K$ and $y_{K+j} = x_j$ for $j = 1 \dots M$). We can write:

$$p(\mathbf{x}, \mathbf{z}) = \prod_k p(z_k|h(z_k)) \prod_j p(x_j|h(x_j)), \tag{17}$$

where $h(y_j)$ represents the subset of (z, x) that form the immediate parents of the variable y_j in the graph $\mathcal{G}^p$. 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 $\mathcal{G}^q$ we can write

$$q(z_1, \dots, z_K|x) = q(z_1|\tilde{h}_1)q(z_2|\tilde{h}_1)q(z_3|\tilde{h}_2)\dots q(z_K|\tilde{h}_K) = \prod_{k=1}^{K} q(z_k|\tilde{h}_k), \tag{18}$$

where $\tilde{h}_i$ denote the parents of z_i according to $\mathcal{G}^q$ (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 $\mathcal{G}^p$ and $\mathcal{G}^q$ need not to be identical. Furthermore, the sampling order for $\mathcal{G}^q$ is arbitrary as long as no variable is sampled before its parents. We choose an ordering of variables in $\mathcal{G}^q$ 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

$$\mathcal{L}(\theta) = \mathbb{E}_{z\sim q_\theta}\left[\sum_j r_j(s_j) + \sum_k \tilde{r}_k(\tilde{s}_k)\right]. \tag{19}$$

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 $z_1, \dots, z_k$, a particular

reward function can be computed, that corresponding reward is added to the instantaneous reward at time k. Formally, for each j, define $t(j) = \min\{k : s_j \subset \{z_1, z_2, \dots, z_k\}\}$. Then, letting

$$r_k(z_1, \dots, z_k) = \sum_{j:t(j)=k} r_j(s_j) + \tilde{r}_k(\tilde{s}_k), \tag{20}$$

we obtain:

$$\mathcal{L}(\theta) = \mathbb{E}_{z_1,z_2,z_3,\dots,z_K \sim q_\theta(z)} \left[\sum_k r_k \right]. \tag{21}$$

The formal MDP state at step k is the latent history $H_k = (z_1, \dots, z_k)$; 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_i which does not participate either a in future reward $r_{>k}$ or conditional probability $q_\theta(z_{>k}|h(z_{>k})$ 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 $\mathcal{S}^q$. 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 r_j and $\tilde{r}_k$, and consider each reward as its own variable r, deterministic function $r(s)$ of its parents $s = h(r)$. We let $\mathcal{R}$ be the set (of cardinality $2K + M$) of reward functions. Finally, we abuse notations and also let $\mathcal{G}^q$ be the set of all latent variable indices (note that we are not using any particular ordering anymore).

The stochastic gradient for the variational cost is given by :

$$\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} = \sum_{v \in \mathcal{G}^q} \mathbb{E}_{z \sim q_\theta} \left[\frac{\partial}{\partial \theta} \log q_\theta(z_v|h(z_v)) \Big(\sum_{r \in \mathcal{R}} r \Big) \right]. \tag{22}$$

We now explore how to exploit the graph structure. For each latent variable z_v in our stochastic computation graph $\mathcal{S}^q$, another node n (either a latent variable or a reward node) is called a nondescendant 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_k 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 $\mathcal{C}(z_v)$ be the set of all nondescendant variables of z_v, $\mathcal{D}(z_v)$ the set of descendants of z, and $\mathcal{D}^+(z_v) = (z_v, \mathcal{D}(z_v))$. Note that $\mathcal{C}(z_v) \cup \mathcal{D}^+(z_v)$ is the set of all latent variables. For any r an element of $\mathcal{C}(z_v)$

$$\begin{aligned}
\mathbb{E}_{z \sim q_\theta} \left[\frac{\partial}{\partial \theta} \log q_\theta(z_v|h(z_v)) r(s) \right] =& \mathbb{E}_{\mathcal{C}(z_v)} \left[\mathbb{E}_{\mathcal{D}^+(z_v)|\mathcal{C}(z_v)} \left[\frac{\partial}{\partial \theta} \log q_\theta(z_v|h(z_v)) r(s) \right] \right] \\
=& \mathbb{E}_{\mathcal{C}(z_v)} \left[r(s)\, \mathbb{E}_{\mathcal{D}^+(z_v)|\mathcal{C}(z_v)} \left[\frac{\partial}{\partial \theta} \log q_\theta(z_v|h(z_v)) \right] \right] \\
=& 0,
\end{aligned}$$

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

$$\mathbb{E}_{\mathcal{D}^+(z_v)|\mathcal{C}(z_v)} \left[\frac{\partial}{\partial \theta} \log q_\theta(z_v|h(z_v)) \right] = \mathbb{E}_{z_v|h(z_v)} \left[\frac{\partial}{\partial \theta} \log q_\theta(z_v|h(z_v)) \right] \tag{23}$$

$$= \frac{\partial}{\partial \theta} \int_{z_v} q_\theta(z_v|h(z_v)) = \frac{\partial}{\partial \theta} 1 = 0. \tag{24}$$

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:

$$\frac{\partial \mathcal{L}(q_\theta)}{\partial \theta} = \sum_{v \in \mathcal{G}^q} \mathbb{E}_{z \sim q_\theta} \left[\frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) R(z_v) \right]. \tag{25}$$

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:

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

where we have defined the critic:

$$Q(z_v, h(z_v)) = \mathbb{E}_{\mathcal{O}(z_v) | z_v, h(z_v)} \left[R(z_v) \right]. \tag{26}$$

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

$$\mathbb{E}_{z_v | h(z_v)} \left[\frac{\partial}{\partial \theta} \log q_\theta(z_v | h(z_v)) V(h(z_v)) \right] = 0, \tag{27}$$

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

$$\mathbb{E}_{z_v, h(z_v) \sim q_\theta} \left[\Big(Q_\phi(z_v, h(z_v)) - \mathbb{E}_{\mathcal{O}(z_v) | z_v, h(z_v)} \left[R(z_v) \right] \Big)^2 \right]. \tag{28}$$

The gradient with respect to ϕ is

$$\mathbb{E}_{z_v, h(z_v) \sim q_\theta} \left[\frac{\partial}{\partial \phi} Q_\phi(z_v, h(z_v)) \left(Q_\phi(z_v, h(z_v)) - \mathbb{E}_{\mathcal{O}(z_v) | z_v, h(z_v)} \left[R(z_v) \right] \right) \right], \tag{29}$$

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:

$$\mathbb{E}_{z_v, h(z_v) \sim q_\theta} \left[\mathbb{E}_{\mathcal{O}(z_v) | z_v, h(z_v)} \left[\frac{\partial}{\partial \phi} Q_\phi(z_v, h(z_v)) \left(Q_\phi(z_v, h(z_v)) - R(z_v) \right) \right] \right], \tag{30}$$

which is finally equal to

$$\mathbb{E}_{z \sim q_\theta} \left[\frac{\partial}{\partial \phi} Q_\phi(z_v, h(z_v)) \left(Q_\phi(z_v, h(z_v)) - R(z_v) \right) \right], \tag{31}$$

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