Credit Assignment Techniques in Stochastic Computation Graphs

Théophane Weber[*,†] **Nicolas Heess**[*] **Lars Buesing** **David Silver**
DeepMind

Abstract

Stochastic computation graphs (SCGs) provide a formalism to represent structured optimization problems arising in artificial intelligence, including supervised, unsupervised, and reinforcement learning. Previous work has shown that an unbiased estimator of the gradient of the expected loss of SCGs can be derived from a single principle. However, this estimator often has high variance and requires a full model evaluation per data point, making this algorithm costly in large graphs. In this work, we address these problems by generalizing concepts from the reinforcement learning literature. We introduce the concepts of value functions, baselines and critics for arbitrary SCGs, and show how to use them to derive lower-variance gradient estimates from partial model evaluations, paving the way towards general and efficient credit assignment for gradient-based optimization. In doing so, we demonstrate how our results unify recent advances in the probabilistic inference and reinforcement learning literature.

1 Introduction

The machine learning community has recently seen breakthroughs in challenging problems in classification, density modeling, and reinforcement learning (RL). To a large extent, successful methods have relied on gradient-based optimization (in particular on the backpropagation algorithm (Rumelhart et al., 1985)) for credit assignment, i.e. for answering the question how individual parameters (or units) affect the value of the objective. Recently, Schulman et al. (2015a) have shown that such problems can be formalized as optimization in stochastic computation graphs (SCGs). Furthermore, they derive a general gradient estimator that remains valid even in the presence of stochastic or non-differentiable computational nodes. This unified view reveals that numerous previously proposed, domain-specific gradient estimators, such as the likelihood ratio estimator (Glasserman, 1992), also known as 'REINFORCE' (Williams, 1992), as well as the pathwise derivative estimator, also known as the "reparameterization trick" (Glasserman, 1991; Kingma & Welling, 2014; Rezende et al., 2014), can be regarded as instantiations of the general SCG estimator. While theoretically correct and conceptually satisfying, the resulting estimator often exhibits high variance in practice, and significant effort has gone into developing techniques to mitigate this problem (Ng et al., 1999; Sutton et al., 2000; Schulman et al., 2015b; Arjona-Medina et al., 2018). Moreover, like backpropagation, the general SCG estimator requires a full forward and backward pass through the entire graph for each gradient evaluation, making the learning dynamics global instead of local. This can become prohibitive for models consisting of hundreds of layers, or recurrent model trained over long temporal horizons.

In this paper, starting from the SCG framework, we target those limitations, by introducing a collection of results which unify and generalize a growing body of results dealing with credit assignment. In combination, they lead to a spectrum of approaches that provide estimates of model parameter gradients for very general deterministic or stochastic models. Taking advantage of the model structure, they allow to trade off bias and variance of these estimates in a flexible manner. Furthermore, they provide mechanisms to derive local and asynchronous gradient estimates that relieve the need for a full model evaluation. Our results are borrowing from and generalizing a class of methods popular primarily in the reinforcement learning literature, namely that of learned approximations to the surrogate loss or its gradients, also known as *value functions*, *baselines* and *critics*. As new models with increasing structure and complexity are actively being developed by the machine learning community, we expect these methods to contribute powerful new training algorithms in a variety of fields such as hierarchical RL, multi-agent RL, and probabilistic programming.

This paper is structured as follows. We review the stochastic computation graph framework and recall the core result

[*]: Equal contribution; [†]: theophane@google.com
Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

of Schulman et al. (2015a) in section 2. In section 3 we discuss the notions of value functions, baselines, and critics in arbitrary computation graphs, and discuss when and how they can be used to obtain both lower variance and local estimates of the model gradients, as well as local learning rules. In section 4 we provide similar results for gradient critics, i.e. estimates or approximations of the downstream loss gradient. In section 5, we discuss how the techniques and concepts introduced in the previous sections can be combined in different ways to obtain a wide spectrum of different gradient estimators with different strengths and weaknesses. Many examples, as well as a discussion of how our results can be special-cased to recent results from the literature, are discussed in the appendix.

Notation for derivatives

We use a 'physics style' notation by representing a function and its output by the same letter. For any two variables x and y in a computation graph, we use the partial derivative $\frac{\partial y}{\partial x}$ to denote the direct derivative of y with respect to x, and the $\frac{dy}{dx}$ to denote the total derivative of y with respect to x, taking into account all paths (or effects) from x on y; we use this notation even if y is still effectively a function of multiple variables. For any variable x, we let $\bar{x}$ denote the value of x which we treat as a constant in the gradient formula; i.e. $\bar{x}$ can be thought of a the output of a 'function' which behaves as the identity, but has gradient zero everywhere[1]. Finally, the gradient of any sampling operation is assumed to be 0.

All proofs are omitted from the main text and can be found in the appendix.

2 Preliminaries

An important class of problems in machine learning can be formalized as optimizing an expected loss $\mathbb{E}_{x_1,\ldots,x_n \sim p\ (x_1,\ldots,x_n)}[\ell(x_1; x_2; \ldots;\ )]$ over parameters , where *both* the sampling distribution p_θ as well as the loss function ℓ can depend on . As we explain in greater detail in the appendix, concrete examples of this setup are reinforcement learning (where p is a composition of known policy and potentially unknown system dynamics) and variational autoencoders (where p is a composition of data distribution and inference network); cf. Fig. 1. Because of the dependency of the distribution on , backpropagation does not directly apply to this problem.

Two well-known estimators, the score function estimator and the pathwise derivative, have been proposed in the literature (for a primer on both, see appendix B). Both turn the gradient of an expectation into an expectation of gradients and hence allow for unbiased Monte Carlo estimates by simulation from known distributions, thus opening the door to stochastic approximations of gradient-based algorithms such as gradient descent (Robbins & Monro, 1985). They can also be unified when seen from the lens of the stochastic computation graph framework, which we detail next.

[1] such an operation is often called 'stop gradient' in the deep learning community; (Schulman et al., 2015a) denote it $\hat{x}$.

2.1 Stochastic computation graphs

SCG framework. We quickly recall the main results from Schulman et al. (2015a).

Definition 1 (Stochastic Computation Graph)**.** *A stochastic computation graph G is a directed, acyclic graph , with two classes of nodes (also called variables): deterministic nodes, and stochastic nodes.*

1. ***Stochastic nodes**, (represented with circles, denoted S), which are distributed conditionally given their parents.*
2. ***Deterministic nodes** (represented with squares, denoted D), which are deterministic functions of their parents.*

We further specialize deterministic nodes as follows:

*Certain deterministic nodes with no parents in the graphs are referred to as **input nodes** 2 , which are set externally, including the parameters we differentiate with respect to.*

*Certain deterministic nodes are designated as **losses** or **costs** (represented with diamonds) and denoted $\ell \in L$. We aim to compute the gradient of the expected sum of costs with respect to some input node .*

A parent v of a node w is connected to it by a directed edge $(v; w)$. Let $L = \sum_{\ell \in \mathcal{L}} \ell$ be the total cost.

For a node v, we let h_v denote the set of parents of v in the graph. A path between v and w is a sequence of nodes $a_0 = v; a_1; \ldots; a_{K-1}; a_K = w$ such that all $(a_{k-1}; a_k)$ are directed edges in the graph; if any such path exists, we say that w descends from v and denote it $v \quad w$. We say the path is blocked by a set of variables V if any of the $a_i \in V$. By convention, v descends from itself. For a variable x and set V, we say x can be deterministically computed from V if there is no path from a stochastic variable to x which is not blocked by V (cf. Fig. 2). Algorithmically, this means that knowing the values of all variables in V allows to compute x without sampling any other variables; mathematically, it means that conditional on V, x is a constant. Finally, whenever we use the notion of conditional independence, we refer to the notion of conditional independence informed by the graph (i.e. d-separation) (Geiger et al., 1990; Koller & Friedman, 2009).

Gradient estimator for SCG. Consider the expected loss $J(\ ) = \mathrm{E}_{s\in\mathcal{S}}[L]$. We present the general gradient estimator for the gradient of the expected loss $\frac{d}{d\theta}J(\ )$ derived in (Schulman et al., 2015a).

For any stochastic variable v, we let $\log p(v)$ denote the conditional log-probability of v given its parents, i. e. the value $\log p(v|h_v)$, and let $s(v;\ )$ denote the score function $\frac{d\log p(v)}{d\theta}$.

Theorem 1. *[Theorem 1 from (Schulman et al., 2015a)] Under simple regularity conditions,*

$$\frac{d}{d}J(\ ) = \mathrm{E}\left[\sum_{\substack{v\in\mathcal{S}\\ \theta\prec v}} s(v;\ )L + \sum_{\substack{\ell\in\mathcal{L}\\ \theta\prec\ell}} \frac{d`}{d}\right]$$

Here, the first term corresponds to the influence has on the loss through the non-differentiable path mediated by stochastic nodes. Intuitively, when using this estimator for gradient descent, is changed so as to increase or 'reinforce' the probability of samples of v that empirically led to lower total cost L. The second term corresponds to the direct influence has on the total cost through differentiable paths. Note that differentiable paths include paths going through reparameterized random variables.

3 Value based methods

The gradient estimator from theorem 1 is very general and conceptually simple but it tends to have high variance (see for instance the analysis found in Mnih & Rezende, 2016), which affects convergence speed (see e.g. Schmidt et al., 2011). Furthermore, it requires a full evaluation of the graph. To address these issues, we first discuss several variations of the basic estimator in which the total cost L is replaced by its *deviation* from the expected total cost, or conditional expectations thereof, with the aim of reducing the variance of the estimator. We then discuss how approximations of these conditional expectations can be learned locally, leading to a scheme in which gradient computations from partial model evaluations become possible.

3.1 Values

In this section, we use the simple concept of conditional expectations to introduce a general definition of value function in a stochastic computation graph.

Definition 2 (Value function)**.** *Let X be an arbitrary subset of G, x an assignment of possible values to variables in X and S an arbitrary scalar value in the graph. The value function for set X is the expectation of the quantity S conditioned on X:*

$$V : \mathrm{x} \mapsto V(\mathrm{x};S) = \mathrm{E}_{\mathcal{G}\setminus\mathcal{X}|\mathcal{X}=\mathbf{x}}[S]:$$

Intuitively, a value function is an estimate of the cost which averages out the effect of stochastic variables not in X, therefore the larger the set, the fewer variables are averaged out.

The definition of the value function as conditional expectation results in the following characterization:

Lemma 1. *For a given assignment x of X, $V(\mathrm{x};S)$ is the optimal mean-squared error estimator of S given input X:*

$$V(\mathrm{x};S) = \mathrm{argmin}_{v_{\mathbf{x}}} \mathrm{E}_{\mathcal{G}\setminus\mathcal{X}|\mathcal{X}=\mathbf{x}}\ (S\quad v_{\mathbf{x}})^2\ :$$

Consider an arbitrary node $v \in G$, and let $L(v) = \sum_{\substack{\ell\in\mathcal{L}\\ v\prec\ell}} `$ denote the v-rooted cost-to-go, i.e. the sum of costs 'downstreams' from v (similar notation is used for $L(V)$ if V is a set). The scalar S will often be the cost-to-go $L(v)$ for some fixed node v; furthermore, when clear from context, we use X to both refer to the variables and the values they take. For notational simplicity, we will denote the corresponding value function $V(X)$.

Fig. 3 shows multiple examples of value functions for different graphs. The above definition is broader than the typical one used in reinforcement learning. There, due to the simple chain structure of the Markov Decision Processes, the resulting Markov properties of the graph, and the particular choice of X, the expectation is only with respect to downstream nodes. Importantly, according to Def. 2 the value can depend on X via ancestors of X (e.g. example in Fig. 3c). Lemma 1 remains valid nevertheless.

3.2 Baselines and critics

In this section, we will define the notions of baselines and critics and use them to introduce a generalization of theorem 1 which can be used to compute lower variance estimator of the gradient of the expected cost. We will then show how to use value functions to design baselines and critics.

Consider an arbitrary node v and input .

Definition 3 (Baseline)**.** *A* baseline B *for v is any function of the graph such that* $\mathrm{E}[s(v;\ )B] = 0$. *A* baseline set B *is an arbitrary subset of the non-descendants of v.*

Baseline sets are of interest because of the following property:

Property 1. *Let B be an arbitrary scalar function of B. Then B is a baseline for v.*

Common choices are constant baselines, i.e. $B = \ ;$, or baselines $B(h_v)$ only depending on the parents $B = h_v$ of v.

Definition 4 (Critic)**.** *A* critic Q *of cost $L(v)$ for v is any function of the graph such that* $\mathrm{E}[s(v;\ )(L(v)\quad Q)] = 0$.

By linearity of expectations, linear combinations of baselines are baselines, and convex combinations of critics are critics.

The use of the terms *critic* and *baseline* is motivated by their respective roles in the following theorem, which generalizes the policy gradient theorem (Sutton et al., 2000):

Theorem 2. *Consider an arbitrary baseline B_v and critic Q_v for each stochastic node v. Then,*

$$\frac{d}{d}J(\ ) = \mathsf{E}\left[\sum_{\substack{v\in\mathcal{S}\\ \theta\prec v}} G_v + \sum_{\substack{\ell\in\mathcal{L}\\ \theta\prec\ell}} \frac{d}{d}\log p(\ell)\,(Q_\ell - B_\ell)\right],$$

where $G_v = s(v;\ )\ Q_v \quad B_v$.

The difference $Q_v \quad B_v$ between a critic and a baseline is called an *advantage* function.

Theorem 2 enables the derivation of a surrogate loss. Let L^s be defined as $L^s = L + \sum_{\substack{\ell\in\mathcal{L}\\ \theta\prec\ell}} \log p(v)\ (\tilde{Q}\quad\tilde{B})$, where we recall that the tilde notation indicates a constant from the point of view of computing gradients. Then, the gradient of the expected cost $J(\ )$ equals the gradient of L^s in expectation: $\frac{d}{d\theta}\mathsf{E}[L] = \mathsf{E}\left[\frac{d}{d\theta}L^s\right]$.

Before providing intuition on this theorem, we see how value functions can be used to design baselines and critics:

Definition 5 (Baseline and critic value functions)**.**
For any node v and baseline set B, a special case of a baseline is to choose the value function with set B. Such a baseline is called a ***baseline value function****.*
Let a critic set C be a set such that $v \in C$, and $\log p(v)$ and $L(v)$ are conditionally independent given C; a special case is when C is such that $\log p(v)$ is deterministically computable given C. Then the value function for set C is a critic for v which we call a ***critic value function*** *for v.*

Figure 3 contains several examples of value functions which take the role of baselines and critics for different nodes. In the standard MDP setup of the RL literature, C consists of the state s and the action a which is taken by a stochastic policy in state s with probability $\log\ (a|s)$, which is a deterministic function of $(s; a)$. Definition 5 is more general than this conventional usage of critics since it does not require C to contain all stochastic ancestor nodes that are required to evaluate $\log p(v)$. For instance, assume that the action is conditionally sampled from the state s and some source of noise , for instance due to dropout, with distribution $(a|\ ; s)$[2]. The critic set may but does not need to include ; if it does not, $\log\ (a|\ ; s)$ is not a deterministic function of a and s. The corresponding critic remains useful and valid.

Three related ideas guide the derivation of theorem 2. To give intuition, let us analyze the term G_v, which replaces the score function weighted by the total cost $s(v;\ )L$. First, the conditional distribution of v only influences the costs downstream from v, hence we only have to reinforce the probability of v with the cost-to-go $L(v)$ instead the total cost L. Second, the extent to which a particular sample v contributed to cost-to-go $L(v)$ should be compared to the cost-to-go the graph typically produces in the first place. This is the intuition behind subtracting the baseline B, also known as a *control variate*. Third, we ideally would like to understand the precise contribution of v to the cost-to-go, not for a particular value of downstream random variables, but on average. This is the idea behind the critic Q. The advantage (difference between critic and baseline) therefore provides an estimate of 'how much better than anticipated' the cost was, as a function of the random choice v.

Baseline value functions are often used as baselines as they approximate the optimal baseline (see Appendix G.1). Critic value functions are often used as they provide an expected downstream cost given the conditioning set. Furthermore, as we will see in the next section, value functions can be estimated in a recursive fashion, enabling local learning of the values, and sharing of value functions between baselines and critics. For these reasons, in the rest of this paper, we will only consider baseline value functions and critic value functions.

In the remainder of this section, we consider an arbitrary value function with conditioning set X.

3.3 Recursive estimation and Markov properties

A fundamental principle in RL is given by the Bellman equation – which details how a value function can be defined recursively in terms of the value function at the next time step. In this section, we generalize the notion of recursive computation to arbitrary graphs.

The main result, which follows immediately from the law of iterated expectations, characterizes the value function for one set, as an expectation of a value function (or critic / baseline value function) of a larger set:

Lemma 2. *Consider two sets $X^1 \quad X^2$, and an arbitrary quantity S. Then we have:* $\mathsf{E}[V(X^2; S)|X^1] = V(X^1; S)$.

This lemma is powerful, as it allows to relate value functions as average of over value function. A simple example in RL is the relation (here, in the infinite discounted case) between the Q function $Q^\pi(s; a) = \mathsf{E}[R|s; a]$ of a policy and the corresponding value function $V^\pi(s) = \mathsf{E}[R|s]$, which is given by $V^\pi(s) = \sum_a (a|s)Q^\pi(s; a)$. Note this equation relates a critic value function to a value function typically used as baseline.

To fully leverage the lemma above, we proceed with a Markov property for graphs[3], which captures the following situation: given two conditioning sets $X^1 \quad X^2$, it may be the case that the additional information contained

[2] in this example, it is important is used only once; it cannot be used to compute other actions.

[3] borrowed from well known conditional independence conditions in graphical models, and adapted to our purposes.

in X^2 does not improve the accuracy of the cost prediction compared to the information contained in the smaller set X^1.

Definition 6. *For conditioning set X, we say that X is **Markov** (for $L(v)$) if for any W such that there exists a directed path from W to $L(v)$ not blocked by X, none of the descendants of W are in X.*

Let $X^\uparrow$ be the set of all ancestors of nodes X[4].

Property 2. *Let X be Markov, consider any X' such that $X \quad X' \quad X^\uparrow$. For any x' assignment of values to the variables in X', let $x'_{|\mathcal{X}}$ be the restriction of x' to the variables in X. Then, for all x', $V(X = x'_{|\mathcal{X}}) = V(X' = x')$;, which we will simply denote, with a slight abuse of notation, $V(X') = V(X)$:*

In other words, the information contained in $X^\uparrow \cap X$ is irrelevant in terms of cost prediction, given access to the information in X. Several examples are shown in Fig. 4. It is worth noting that Def. 6 does not rule out changes in the expected value of $L(v)$ after adding additional nodes to X (cf. Fig. 4(d,e)). Instead it rules out correlations between X and $L(v)$ that are mediated via *ancestors* of nodes in X as in the example in Fig. 4(a,b,c)).

The notion of Markov set can be used to refine Lemma 2:

Lemma 3 (Generalized Bellman equation)**.** *Consider two sets $X^1 \quad X^{2\uparrow}$, and suppose X^2 is Markov. Then we have: $\mathbb{E}[V(X^2)|X^1] = V(X^1)$.*

The Markov assumption is critical in allowing to 'push' the boundary at which the expectation is defined; without it, lemma 2 only allows to relate value functions of sets which are subset of one another. But notice here that no such inclusion is required between X^1 and X^2 themselves. In the context of RL, this corresponds to equations of the type $V(s) = \sum_a \pi(s,a)\,(r(s,a) + \sum_{s'} P(s'|s,a)V(s'))$ (see Fig. 5), though to get the separation between the reward and the value at the next time step, we will need a slight refinement, which we detail in the next section.

3.4 Decomposed costs and bootstrap

In the previous sections we have considered a value function with respect to a node v which predicts an estimate of the cost-to-go $L(v)$ from node v (note $L(v)$ was implicit in most of our notation). In this section, we write the cost-to-go at a node as a funtion of cost-to-go from other nodes or collection of nodes, and leverage the linearity of expectation to turn these relations between costs into relation between value functions.

Definition 7 (Decomposed costs)**.** *For a node v and a collection $\mathbb{V} = \{V_0, V_1, \ldots, V_D\}$ in the graph, we say that the cost $L(v)$ can be decomposed with set $\mathbb{V}$ if $L(v) = \sum_i L(V_i)$.*

This implies that cost nodes can be grouped in disjoint sets corresponding to the descendents of different sets V_i, without double-counting. A common special case is a tree, where each V_i is a singleton containing a single child $\{v_i\}$ of v.

Theorem 3 (Bootstrap principle for SCGs)**.** *Suppose the cost-to-go $L(v)$ from node v can be decomposed with sets $\mathbb{V} = \{V_0, \ldots, V_D\}$, and consider an arbitrary set X_v with associated value function $V(X_v; L(v))$. Furthermore, for each set V_i, consider a set X_{V_i} and associated value function: $V(X_{V_i}; L(V_i))$. If for each i, $X_v \quad X_{V_i}$, or if for each i, X_{V_i} is Markov and $X_v \quad X_{V_i}^\uparrow$, then:*

$$V(X_v) = \sum_i \mathbb{E}_{\mathcal{G}\setminus x_v|x_v}[V(X_{V_i})]:$$

Fig. 6 highlights potential difficulties of defining correct bootstrap equations for various graphs.

From the bootstrap equation follows a special case, which we call *partial averaging*, often used for critics:

Corollary 1 (Partial averages)**.** *Suppose that for each i, X_{V_i} is Markov and $X_{V_i} \quad X_v \quad X_{V_i}^\uparrow$. Without loss of generality, define V_0 as the collection of all cost nodes which can be deterministically computed from X_v. Then,*

$$V(X_v) = \sum_i V(X_{V_i}) = \sum_{\ell \in V_0} \ell + \sum_{i \geq 1} V(X_{V_i}):$$

The term 'partial average' indicates that the value function is a conditional expectation (i.e. 'averaging' variables) but that it combines averaged cost estimates (the value terms $V(X_{V_i})$) and empirical costs ($\sum_{\ell \in V_0} \ell_v$). Fig. 7 shows some examples for generic graphs.

In the case of RL for instance, a K-step return is a form of partial average, since the return R_t – sum of all rewards downstream from state s_t – can be written as the sum of all rewards in $V_0 = \{r_t, \ldots, r_{t+k-1}\}$ and downstream from $V_1 = \{s_{t+k}\}$; the critic value function $V(s_t, \ldots, s_{t+k})$ is therefore equal[5] to $\sum_{t'=t}^{t+k-1} r_{t'} + V(s_{t+k})$. This implies in turn that $V(s_t) = \mathbb{E}[V(s_t, \ldots, s_{t+k})]$ is also equal to $\mathbb{E}[\sum_{t'=t}^{t+k} r_{t'} + V(s_{t+k})]$.

3.5 Approximate Value functions

In practice, value functions often cannot be computed exactly. In such cases, one can resort to learning parametric approximations. For node v, conditioning set X, we will consider an approximate value function $\hat{V}^\phi(X)$ as an approximation (with parameters) to the value function $V(X) = \mathbb{E}_{\mathcal{G}\setminus\mathcal{X}_v|\mathcal{X}_v}[L(v)]$.

Following corollary 1, we know that for a possible assignment $\mathbf{x}$ of variables X, $V(\mathbf{x})$ minimizes $\mathbb{E}_{\mathcal{G}\setminus\mathcal{X}|\mathcal{X}} \quad (L(v) \quad v_{\mathbf{x}})^2$ over $v_{\mathbf{x}}$. We therefore elect to optimize by considering the following weighted average,

[4] Recall that by convention nodes are descendants of themselves, so $X \quad X^\uparrow$

[5] We assume for simplicity that the rewards are deterministic functions of the state; the result can be trivially generalized.

called a *regression on return* in reinforcement learning:

$$L^{\phi} = \mathbb{E}_{\mathcal{X}}\left[\mathbb{E}_{\mathcal{G}\setminus\mathcal{X}|\mathcal{X}}\left[(L(v) - \hat{V}^{\phi}(X))^2\right]\right] = \mathbb{E}_{\mathcal{G}}\left[(L(v) - \hat{V}^{\phi}(X))^2\right];$$

from which we obtain:

$$\frac{dL^{\phi}}{d} = \mathbb{E}_{\mathcal{G}}\left[\frac{d}{d}\hat{V}^{\phi}(X) \; \hat{V}^{\phi}(X) - L(v)\right] \quad (1)$$

which can easily be computed by forward sampling from $\mathcal{G}$, even if conditional sampling given X is difficult.

We now leverage the recursion methods from the previous sections in two different ways. The first is to use the combination of approximate value functions and partial averages to define other value functions. For a partial average as defined in theorem 1 and family of approximate value functions $\hat{V}^{\phi}_{v_i}(X_{v_i})$, we can define an approximate value function through the bootstrap equation: $\sum_{\ell\in V_0} \ell_v + \sum_i \hat{V}^{\phi}(X_{v_i})$. In other words, using the bootstrap equations, approximating value functions for certain sets automatically defines other approximate value functions for other sets.

In general, we can trade bias and variance by making V_0 larger (which will typically result in lower bias, higher variance) or not, i.e. by shifting the boundary at which variables are integrated out, for instance by using K-step returns or -weighted returns. An extreme case of a partial average is not an average at all, where $X = \mathcal{G}$, in which case the value function is the empirical return $L(v)$.

The second way to use the bootstrap equation is to provide a different, lower variance target to the value function regression. By combining theorem 3 and equation 1, we obtain:

$$\frac{dL_{\phi}}{d} = \mathbb{E}_{\mathcal{G}}\left[\frac{d}{d}\hat{V}^{\phi}(X) \; \hat{V}^{\phi}(X) - \sum_i \hat{V}^{\phi}(X_{v_i})\right]$$

By following this gradient, the value function $\hat{V}^{\phi}(X_v)$ will tend towards the bootstrap value $\sum_{\ell\in V_0} \ell_v + \sum_{i\geq 1} \hat{V}^{\phi}(X_{v_i})$ instead of the return $L(v)$. Because the former has averaged out stochastic nodes, it is a lower variance target, and should in practice provide a stronger learning signal. Furthermore, as it can be evaluated as soon as X_{v_i} is evaluated, it provides a local or online learning rule for the value at v; by this we mean the corresponding gradient update can be computed as soon as all sets X_{v_i} are evaluated. In RL, this local learning property can be found in actor-critic schemes: when taking action a_t in state s_t, as soon as the immediate reward r_t is computed and next state s_{t+1} is evaluated, the value function $V(s_t)$ (which is a baseline for a_t) can be regressed against low-variance target $r_t + V(s_{t+1})$, which can also be used as critic for a_t and therefore used to update the corresponding policy.

4 Gradient-based methods

In the previous section, we developed techniques to lower the variance of the score-function terms $\mathbb{E}\left[\frac{d}{d\theta}\log p(v) \; Q(C) - B(B_v)\right]$ in the gradient estimate. This led to the construction of a surrogate loss L^s which satisfies $\frac{d}{d\theta}\mathbb{E}[J(\ )] = \mathbb{E}\left[\frac{dL^s}{d\theta}\right]$.

In this section, we develop corresponding techniques to lower the variance estimates of the gradients of surrogate cost $\frac{d}{d\theta}L^s$. To this end, we will again make use of conditional expectations to partially average out variability from stochastic nodes. This leads to the idea of a *gradient-critic*, the equivalent of the value critic for gradient-based approaches.

4.1 Gradient-Critic

Definition 8 (Value-gradient)**.** *The* ***value-gradient*** *for v with set C is the following function of C:*

$$g(C) = \mathbb{E}_{\mathcal{G}\setminus C|C}\left[\frac{dL^s}{dv}\right];$$

Value-gradients are not directly useful in our derivations but we will see later that certain value-gradients can reduce the variance of our estimators. We call these value-gradient *gradient-critics*.

Definition 9 (Gradient-critic)**.** *Consider two nodes u and v, and a value-gradient g_v for node v with set C. If $\frac{dv}{du}$ and $\frac{dL^s}{dv}$ are conditionally independent given C[6], then we say the value-gradient is a* ***gradient-critic*** *for v with respect to u.*

Corollary 2. *If $\frac{dv}{du}$ is deterministically computable from C, then $g_v(C)$ is a gradient-critic for v with respect to u.*

We can use gradient-critics in the backpropagation equation. First, we recall the equation for backpropagation and stochastic backpropagation. Let u be an arbitrary node of H, and $\{v_1, \ldots, v_d\}$ be the children of u in $\mathcal{G}$. The backpropagation equations state that: $\frac{dL^s}{du} = \sum_i \frac{dL^s}{dv_i}\frac{\partial v_i}{\partial u}$. From this we obtain the stochastic backpropagation equations:

$$\mathbb{E}_{\mathcal{G}}\left[\frac{dL^s}{du}\right] = \mathbb{E}_{\mathcal{G}}\left[\sum_i \frac{dL^s}{dv_i}\frac{\partial v_i}{\partial u}\right]$$

Gradient-critics allow for replacing these stochastic estimates by conditional expectations, potentially achieving lower variance:

Theorem 4. *For each child v_i of v, let g_{v_i} be a gradient-critic for v_i with respect to u. We then have:*

$$\mathbb{E}_{\mathcal{G}}\left[\frac{dL^s}{du}\right] = \mathbb{E}_{\mathcal{G}}\left[\sum_i g_{v_i}\frac{\partial v_i}{\partial u}\right];$$

[6] See lemma 7 in Appendix for a characterization of conditional independence between total derivatives.

Note a similar intuition as the idea of critic defined in the previous section. In both cases, we want to evaluate the expectation of a product of two correlated random variables, and replace one by its expectation given a set which makes the variables conditionally independent.

4.2 Horizon gradient-critic and gradient-critic bootstrap

More generally, we do not have to limit ourselves to $fv_1; v_2; :::; v_dg$ being children of u. We define a *separator set* for u in H to be a set $fv_1; v_2; :::; v_dg$ such that every deterministic path from u to the loss L^s is blocked by a v_i 2 H. For simplicity, we further require the separator set to be *unordered*, which means that for any i 6= j, v_j cannot be an ancestor to v_i; we drop this assumption for a generalized result in the appendix C. Under these assumptions, the backpropagation rule can be rewritten (see (Naumann, 2008; Parmas, 2018)):

$$\mathrm{E}_{\mathcal{G}}\left[\frac{dL^s}{du}\right] = \mathrm{E}_{\mathcal{G}}\left[\sum_i \frac{dL^s}{dv_i}\frac{dv_i}{du}\right] : \tag{2}$$

Theorem 5. *Assume that for every i, g_{v_i} is a gradient critic for v_i with respect to u. We then have:*

$$\mathrm{E}_{\mathcal{G}}\left[\frac{dL^s}{du}\right] = \mathrm{E}_{\mathcal{G}}\left[\sum_i g_{v_i}\frac{dv_i}{du}\right] :$$

This theorem allows us to 'push' the horizon after which we start using gradient-critics. It constitutes the gradient equivalent of partial averaging, since it combines stochastic backpropagation (the terms $\frac{dv_i}{du}$) and gradient critics g_{v_i}.

We now show how this theorem to derive a generic notion of bootstrapping for gradient-critics:

Theorem 6 (Gradient-critic bootstrap)**.** *Consider a node u, unordered separator set $fv_1; :::; v_dg$. Consider value-gradient g_u with set C_u for node u, and $(g_{v_1}; g_{v_2}; :::; g_{v_d})$ with Markov sets $(C_{v_1}; :::; C_{v_d})$ critics for v_i with respect to u. Suppose that for all i, C_u C_{v_i}. Then,*

$$g_u = \sum_i \mathrm{E}_{C_{v_i}|C_u}\left[g_{v_i}\frac{dv_i}{du}\right] : \tag{3}$$

4.3 Gradient-critic and gradient of critic

The section above proposes an operational definition of a gradient critic, in that one can replace the sampled gradient $\frac{dL^s}{du}$ by the expectation of the gradient g_u. A natural question follows – is a value-gradient the gradient of a value function? Similarly, is a gradient-critic the gradient of a critic function?

It is in general not true that the value-gradient must be the gradient of a value function. However, if the critic set is Markov, the gradient-critic is the gradient of the critic.

Theorem 7. *Consider a node v and critic set C, and corresponding critic value function $Q(C)$ and gradient-critic $g_v(C)$. If C is Markov for v, then we have: $\frac{dQ(C)}{dv} = g_v(C)$.*

This characterization of the gradient-critic as gradient of a critic plays a key role in using reparametrization techniques when gradients are not computable. For instance, in a continuous control application of reinforcement learning, the state of the environment can be assumed to be an unknown but differentiable function of the previous state and of the action. In this context, a critic can readily be learned by predicting total costs. By the argument above, the gradient of this critic actually corresponds to the gradient-critic of the unknown environment dynamics. This technique is at the heart of differentiable policy gradients (Lillicrap et al., 2015) and stochastic value gradients (Heess et al., 2015).

4.4 Gradient-critic approximation and computation

Following the arguments regarding conditional expectation and square minimization from section 3.1, we know that g_v satisfies the following minimization problem:

$$g_v(C) = \mathrm{argmin}_{g_{c_v}} \mathrm{E}_{\mathcal{G}\setminus C|C}\left[\left(g_{c_v} - \frac{dL^s}{dv}\right)^2\right]$$

For a parametric approximation g_v^ϕ, and using the same weighting scheme as section 3.5, it follows that:

$$L^\phi = \mathrm{E}_{\mathcal{G}}\left[\left(g_v^\phi(C) - \frac{dL^s}{dv}\right)^2\right]$$

$$\frac{dL^\phi}{d} = \mathrm{E}_{\mathcal{G}}\left[\frac{d}{d}g_v^\phi(C)\left(g_v^\phi(C) - \frac{dL^s}{dv}\right)\right] \tag{4}$$

Finally, if C is Markovian for v, from Theorem 7, the gradient-critic g_v^ϕ can be defined in two ways: first, as the critic of a gradient ($\mathrm{E}[\frac{dL^s}{dv}|C]$), and second, as the gradient of a critic ($\frac{d}{dv}\mathrm{E}[L|C] = \frac{d}{dv}Q(C)$).

In this case, it makes sense to parameterize g_v^ϕ as the derivative of a function $Q^\phi(C)$, where v 2 C, i.e. define $g_v^\phi(C) = \frac{dQ(C)}{dv}$. The gradient-critic can therefore defined directly by the gradient-critic loss, and indirectly by the critic loss. It therefore makes sense to combine them:

$$L^\phi = \mathrm{E}_{\mathcal{G}}\left[\left(Q^\phi(C) - L\right)^2 + \left(\frac{dQ^\phi(C)}{dv} - \frac{dL^s}{dv}\right)^2\right] \tag{5}$$

where ; are relative weights for each norm. This is called a *Sobolev* norm, see also (Czarnecki et al., 2017).

5 Combination of estimators and critics

The techniques outlined above suggest a "menu" of choices for constructing gradient estimators for stochastic computation graphs. We lay out a few of these choices, highlighting how are results strictly generalize known methods from the literature.

Reparameterization; use of score function and pathwise derivative estimators. Many distributions can be reparameterized, including discrete random variables (Maddison et al., 2016; Jang et al., 2016). This opens a choice between SF and PD estimator. The latter allows gradients to flow through the graph. Where exact gradients are not available (e.g. in MDPs or in probabilistic programs and approximate Bayes computation (Meeds & Welling, 2014; Ong et al., 2018)) gradients of critics can under certain conditions be used in combination with reparameterized distributions (see e.g. (Heess et al., 2015).

Grouping of random variables. For many graphs there is a natural grouping of random variables that suggests obvious baseline and critic choices. Taking into account the detailed Markov structure of the computation graph may however reveal interesting alternatives. For instance, independent action dimensions allow updates in which baselines are conditioned on the values chosen for other action dimensions. Such ideas have been exploited e.g. in work on action-dependent baselines (Wu et al., 2018), and in multi-agent domains (Foerster et al., 2017). Other applications have been found in hierarchical RL, for instance in (Bacon et al., 2017), where the relation between options and actions directly informs the computation graph structure, in turn defining the correct value bootstrap equations and corresponding policy gradient theorem.

Use of value critics and baselines. The discussion in 3.2 highlights there are typically many different choices for constructing baselines and critics even beyond the choice of a particular variable grouping (e.g. the use of K-step returns or generalized advantage estimates in RL, Schulman et al. (2015a)) even for simple graphs (such as chains).

Use of gradient-critics. For reparameterized variables or general deterministic pathways through a computation graph gradient critics can be used. Gradient critics for a given node in the graph can be obtained by either directly approximating the (expected) gradient of the downstream loss, or by approximating the value of the future loss terms (as for value critics) and then using the gradient of this approximation. Gradient-critics allow to conceptualize the links between related notions of value-gradient found in (Fairbank & Alonso, 2012; Fairbank, 2014), stochastic value-gradients (Heess et al., 2015), and synthetic gradients (Jaderberg et al., 2016; Czarnecki et al., 2017).

Debiasing of estimators through policy-gradient correction. The use of critics or gradient-critics results in biased estimators. When using gradient-critic, it is often possible to debias the use of the gradient-critic by adding a correction term corresponding to the critic error. The resulting scheme is unbiased, but may or may not have lower variance than 'naive' estimators which do not use critics at all. This is sometimes known as 'action-conditional' baselines in the literature (Tucker et al., 2018), and is also strongly related to Stein variational gradient (Liu & Wang, 2016). See App. G.3 for more details.

Bootstrapping. Targets for baselines and critics can be obtained in a variety of ways: For instance, they can be regressed directly onto empirical sums of downstream losses ("Monte Carlo" returns in reinforcement learning). But targets can also constructed from other, downstream value or gradient approximations (e.g. "K-step returns" or " -weighted returns" in reinforcement learning), an idea discussed above under the name *bootstrapping* (sections 3.3 and 4.2). The appropriate choice here will again be highly application specific.

Decoupled updates. In its original form Theorem 1 requires a full and backward pass through the entire computation graph to compute a single sample approximation to its gradient. Through appropriate combination of surrogate signals and bootstrapping, however, updates for different parts of the graph can be decoupled to different extents. For instance, in reinforcement learning actor-critic algorithms compute updates to the policy parameters from single transitions $s_t; a_t; r_t; s_{t+1}$. The same ideas can be applied to general computation graphs where additional freedom can allow even more flexible schemes (e.g. individual parts of the graph can be updated more frequently than others).

6 Conclusion

In this paper, we have provided a detailed discussion and mathematical analysis of credit assignment techniques for stochastic computation graphs. Our discussion explains and unifies existing algorithms, practices, and results obtained in a number of particular models and different fields of the ML literature. They also provide insights about the particular form of algorithms, highlighting how they naturally result from the constraints imposed by the computation graph structure, instead of ad-hoc solutions to particular problems.

The conceptual understanding and tools developed in this work do not just allow the derivation of existing solutions as special cases. Instead, they also highlight the fact that for any given model there typically is a menu of choices, each of which gives rise to a different gradient estimator with different advantages and disadvantages. For new models, these tools provide methodological guidance for the development of appropriate algorithms. In that sense our work emphasizes a similar separation of model and algorithm that has been proven fruitful in other domains, for instance in the probabilistic modeling and inference literature.

We believe that this separation as well as a good understanding of the underlying principles will become increasingly important as both models and training schemes become more complex and the distinction between different model classes blurs.

Acknowledgments

Wed like to thank the many people for useful discussions and feedback on the research and the manuscript, including Ziyu Wang, Sébastien Racanière, Yori Zwols, Chris Maddison, Arthur Guez and Andriy Mnih.

References

Archer, Evan, Park, Il Memming, Buesing, Lars, Cunningham, John, and Paninski, Liam. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Arjona-Medina, Jose A, Gillhofer, Michael, Widrich, Michael, Unterthiner, Thomas, and Hochreiter, Sepp. Rudder: Return decomposition for delayed rewards. *arXiv preprint arXiv:1806.07857*, 2018.

Bacon, Pierre-Luc, Harb, Jean, and Precup, Doina. The option-critic architecture. In *AAAI*, pp. 1726–1734, 2017.

Bayer, Justin and Osendorfer, Christian. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Bengio, Yoshua, Léonard, Nicholas, and Courville, Aaron. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Buesing, Lars, Weber, Theophane, Racaniere, Sebastien, Eslami, SM, Rezende, Danilo, Reichert, David P, Viola, Fabio, Besse, Frederic, Gregor, Karol, Hassabis, Demis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

Chung, Junyoung, Kastner, Kyle, Dinh, Laurent, Goel, Kratarth, Courville, Aaron C, and Bengio, Yoshua. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Czarnecki, Wojciech M, Osindero, Simon, Jaderberg, Max, Swirszcz, Grzegorz, and Pascanu, Razvan. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, pp. 4278–4287, 2017.

Eslami, SM Ali, Heess, Nicolas, Weber, Theophane, Tassa, Yuval, Szepesvari, David, Hinton, Geoffrey E, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Fairbank, Michael. *Value-gradient learning*. PhD thesis, City University London, 2014.

Fairbank, Michael and Alonso, Eduardo. Value-gradient learning. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–8. IEEE, 2012.

Figurnov, Michael, Mohamed, Shakir, and Mnih, Andriy. Implicit reparameterization gradients. *arXiv preprint arXiv:1805.08498*, 2018.

Foerster, Jakob N., Farquhar, Gregory, Afouras, Triantafyllos, Nardelli, Nantas, and Whiteson, Shimon. Counterfactual multi-agent policy gradients. *CoRR*, abs/1705.08926, 2017.

Fortunato, Meire, Azar, Mohammad Gheshlaghi, Piot, Bilal, Menick, Jacob, Osband, Ian, Graves, Alex, Mnih, Vlad, Munos, Remi, Hassabis, Demis, Pietquin, Olivier, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Fraccaro, Marco, Sønderby, Søren Kaae, Paquet, Ulrich, and Winther, Ole. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

Gan, Zhe, Li, Chunyuan, Henao, Ricardo, Carlson, David E, and Carin, Lawrence. Deep temporal sigmoid belief networks for sequence modeling. In *Advances in Neural Information Processing Systems*, pp. 2467–2475, 2015.

Geiger, Dan, Verma, Thomas, and Pearl, Judea. Identifying independence in bayesian networks. *Networks*, 20(5): 507–534, 1990.

Gershman, Samuel and Goodman, Noah. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 36, 2014.

Glasserman, Paul. *Gradient estimation via perturbation analysis*. Springer Science & Business Media, 1991.

Glasserman, Paul. Smoothing complements and randomized score functions. *Annals of Operations Research*, 39 (1):41–67, 1992.

Grathwohl, Will, Choi, Dami, Wu, Yuhuai, Roeder, Geoff, and Duvenaud, David. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Greensmith, Evan, Bartlett, Peter L, and Baxter, Jonathan. Variance reduction techniques for gradient estimates in reinforcement learning. *The Journal of Machine Learning Research*, 5:1471–1530, 2004.

Gregor, Karol, Papamakarios, George, Besse, Frederic, Buesing, Lars, and Weber, Theophane. Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*, 2018.

Gu, Shixiang, Levine, Sergey, Sutskever, Ilya, and Mnih, Andriy. Muprop: Unbiased backpropagation for stochastic neural networks. *arXiv preprint arXiv:1511.05176*, 2015.

Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, and Levine, Sergey. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Heess, Nicolas, Wayne, Gregory, Silver, David, Lillicrap, Tim, Erez, Tom, and Tassa, Yuval. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Heess, Nicolas, Wayne, Greg, Tassa, Yuval, Lillicrap, Timothy, Riedmiller, Martin, and Silver, David. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Heng, Jeremy, Bishop, Adrian N, Deligiannidis, George, and Doucet, Arnaud. Controlled sequential monte carlo. *arXiv preprint arXiv:1708.08396*, 2017.

Igl, Maximilian, Zintgraf, Luisa, Le, Tuan Anh, Wood, Frank, and Whiteson, Shimon. Deep variational reinforcement learning for POMDPs. *arXiv preprint arXiv:1806.02426*, 2018.

Jaderberg, Max, Czarnecki, Wojciech Marian, Osindero, Simon, Vinyals, Oriol, Graves, Alex, Silver, David, and Kavukcuoglu, Koray. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kingma, Diederik P and Welling, Max. Auto-encoding variational Bayes. *arXiv:1312.6114*, 2013.

Kingma, Diederik P and Welling, Max. Efficient gradient-based inference through transformations between bayes nets and neural nets. *arXiv preprint arXiv:1402.0480*, 2014.

Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Kosiorek, Adam R, Kim, Hyunjik, Posner, Ingmar, and Teh, Yee Whye. Sequential attend, infer, repeat: Generative modelling of moving objects. *arXiv preprint arXiv:1806.01794*, 2018.

Krishnan, Rahul G, Shalit, Uri, and Sontag, David. Deep Kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Levine, Sergey. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, Hao, He, Lirong, Bai, Haoli, and Xu, Zenglin. Efficient structured inference for stochastic recurrent neural networks. 2017.

Liu, Qiang and Wang, Dilin. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pp. 2378–2386, 2016.

Lowe, Ryan, Wu, Yi, Tamar, Aviv, Harb, Jean, Abbeel, Pieter, and Mordatch, Igor. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6382–6393, 2017.

Maddison, Chris J, Mnih, Andriy, and Teh, Yee Whye. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Meeds, Edward and Welling, Max. Gps-abc: Gaussian process surrogate approximate bayesian computation. *arXiv preprint arXiv:1401.2838*, 2014.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. *arXiv:1402.0030*, 2014.

Mnih, Andriy and Rezende, Danilo J. Variational inference for monte carlo objectives. *arXiv preprint arXiv:1602.06725*, 2016.

Moreno, Pol, Humplik, Jan, Papamakarios, George, Avila Pires, Bernardo, Buesing, Lars, Heess, Nicolas, and Weber, Theophane. Neural belief states for partially observed domains. *NeurIPS 2018 workshop on Reinforcement Learning under Partial Observability*, 2018.

Naumann, Uwe. Optimal jacobian accumulation is np-complete. *Mathematical Programming*, 112(2):427–441, 2008.

Ng, Andrew Y, Harada, Daishi, and Russell, Stuart. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Ong, Victor MH, Nott, David J, Tran, Minh-Ngoc, Sisson, Scott A, and Drovandi, Christopher C. Variational bayes with synthetic likelihood. *Statistics and Computing*, 28 (4):971–988, 2018.

Paisley, John, Blei, David, and Jordan, Michael. Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.

Parmas, Paavo. Total stochastic gradient algorithms and applications in reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10224–10234. 2018.

Peng, Xue Bin, Andrychowicz, Marcin, Zaremba, Wojciech, and Abbeel, Pieter. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*, 2017.

Piché, Alexandre, Thomas, Valentin, Ibrahim, Cyril, Bengio, Yoshua, and Pal, Chris. Probabilistic planning with sequential monte carlo methods. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByetGn0cYX.

Plappert, Matthias, Houthooft, Rein, Dhariwal, Prafulla, Sidor, Szymon, Chen, Richard Y, Chen, Xi, Asfour, Tamim, Abbeel, Pieter, and Andrychowicz, Marcin. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Ranganath, Rajesh, Gerrish, Sean, and Blei, David M. Black box variational inference. *arXiv preprint arXiv:1401.0118*, 2013.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. *arXiv:1401.4082*, 2014.

Robbins, Herbert and Monro, Sutton. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pp. 102–109. Springer, 1985.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Schmidt, Mark, Roux, Nicolas L, and Bach, Francis R. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems*, pp. 1458–1466, 2011.

Schulman, John, Heess, Nicolas, Weber, Theophane, and Abbeel, Pieter. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pp. 3528–3536, 2015a.

Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael, and Abbeel, Pieter. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. Deterministic policy gradient algorithms. In *ICML*, 2014.

Sutton, Richard S, McAllester, David A, Singh, Satinder P, and Mansour, Yishay. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Tucker, George, Mnih, Andriy, Maddison, Chris J, Lawson, John, and Sohl-Dickstein, Jascha. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pp. 2627–2636, 2017.

Tucker, George, Bhupatiraju, Surya, Gu, Shixiang, Turner, Richard E, Ghahramani, Zoubin, and Levine, Sergey. The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031*, 2018.

Weber, Theophane, Heess, Nicolas, Eslami, Ali, Schulman, John, Wingate, David, and Silver, David. Reinforced variational inference. In *Advances in Neural Information Processing Systems (NIPS) Workshops*, 2015.

Werbos, Paul J. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pp. 762–770. Springer, 1982.

Wierstra, Daan and Schmidhuber, Jürgen. Policy gradient critics. In *European Conference on Machine Learning*, pp. 466–477. Springer, 2007.

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

Wingate, David and Weber, Theophane. Automated variational inference in probabilistic programming. *arXiv preprint arXiv:1301.1299*, 2013.

Wu, Cathy, Rajeswaran, Aravind, Duan, Yan, Kumar, Vikash, Bayen, Alexandre M., Kakade, Sham, Mordatch, Igor, and Abbeel, Pieter. Variance reduction for policy gradient with action-dependent factorized baselines. *CoRR*, abs/1803.07246, 2018. URL http://arxiv.org/abs/1803.07246.

A figures

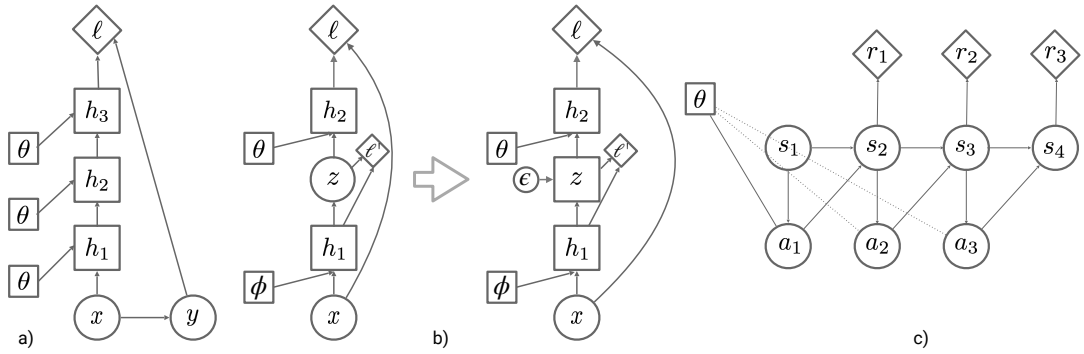

Figure 1: *Typical models as SCGs.* (a) **Supervised learning.** The loss is $\mathsf{E}_{x;y}\left[\log p(y|x;\ )\right]$ where $x;y\quad p_D$ are drawn from the data distribution. $p(y|x;\ )$ is realized as a multi-layer neural network with hidden layers $h_1;h_2;h_3$ and parameters . (b) **Variational autoencoder.** The observed data $x\quad p_D$ is fed into an inference network $q(z|x;\ )$ (with hidden units h_1) to infer the latent variables z. The loss ` is given by the Evidence lower bound (ELBO) computed from the likelihood $p(x|z;\ )$ (with hidden units h_2). We show two variants: the not-reparameterized variant (left), where the ELBO is given by $\mathsf{E}_x\left[\mathsf{E}_z\left[\log p(x|z;\ )+\log p(z;\ )\right]\quad H\left[q(\ |x;\ )\right]$; and the reparameterized variant (right), where the ELBO is expressed as $\mathsf{E}_x\left[\mathsf{E}\ \left[\log p(x|h(x;\ ;\ );\ )+\log p(h(x;\ ;\ );\ )\right]\quad H\left[q(\ |x;\ )\right]\right]$. Note that z is a deterministic node in the reparametrized graph. (c) **Markov decision process** . The objective here is the (undiscounted) return $\mathsf{E}\left[\ \ {}_t\, r(s_t)\right]$ and trajectories $=(:::;s_t;a_t;:::)$ are drawn from the composition of a policy and the system dynamics: $p(\ ;\ )=p(s_0)\ \ {}_t\ \ (a_t|s_t;\ )p(s_{t+1}|s_t;a_t)$.

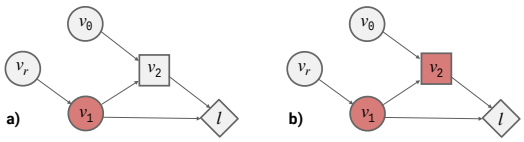

Figure 2: *Blocked paths and deterministic computability.* a) $X=\{v_1\}$, shown in red, blocks the path from v_r to `, but not $v_0$ to `. b) ` can be deterministically computed from $X=\{v_1;v_2\}$ (red) because all stochastic variables have their path to L blocked by X (v_0, blocked by v_2; v_r, blocked by v_1; and v_1, blocked by itself).

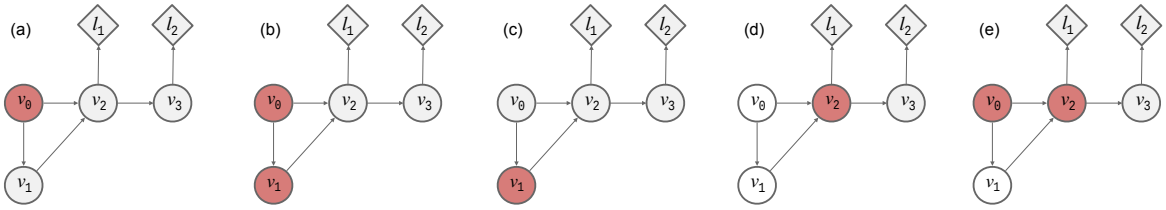

Figure 3: *Examples of baseline value functions and critic value functions.* (a-e) Different value functions for the same SCG with $X^{(a)}=\{v_0\}$, $X^{(b)}=\{v_0;v_1\}$, $X^{(c)}=\{v_1\}$, $X^{(d)}=\{v_2\}$, $X^{(e)}=\{v_0;v_2\}$. The variables averaged over are shaded in light gray. The value function in (c) involves a marginalization over 'upstream' variables: $V(v_1)=\mathsf{E}_{v_0;v_2;v_3|v_1}[`_1+`_3]$.
(a,b,c) are valid baselines for v_2; (d,e) cannot act as baselines for v_2 since v_2 belongs to those sets, but can act as baselines for v_3.
(a,b,e) are critics for v_0, but (c,d) are not. (b) is a critic for v_1. (c) is not a critic for v_1 since $L(v_1)$ and $\log\ \ {}_{v_1}$ are correlated conditionally on $X^{(c)}$ (through v_0). Finally (d,e) are critics for v_2; note however $\log p(v_2)$ is not a deterministic function of either $X^{(d)}$ or $X^{(e)}$.

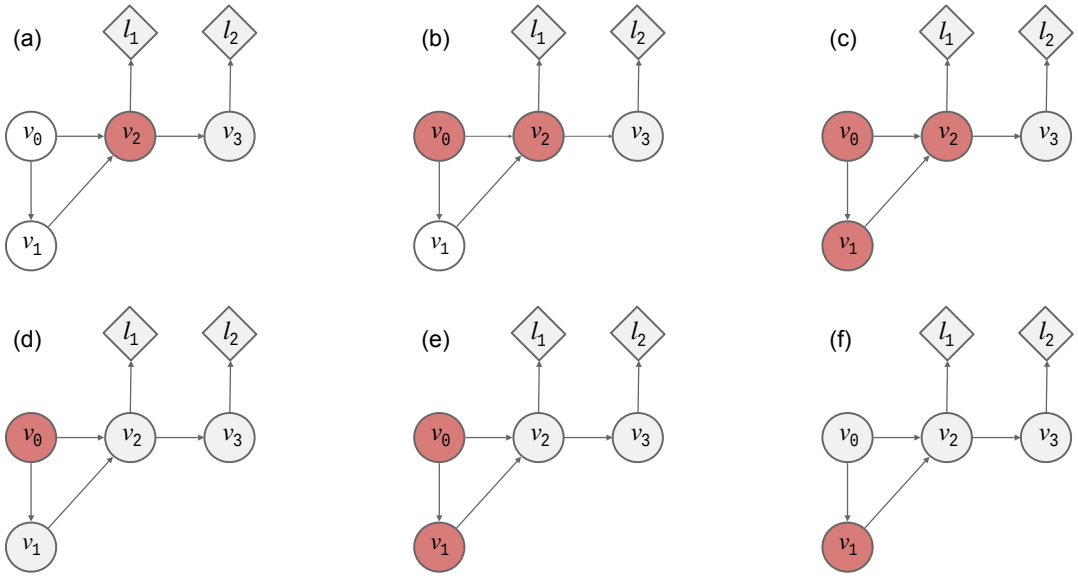

Figure 4: *Markov property for value functions.* Consider node v_2 and corresponding cost $L(v_2) = \ell_1 + \ell_2$; we consider the Markov property for $L(v_2)$. For (a,b,c), $X^{(a)} \quad X^{(b)} \quad X^{(c)} = X^{(a)\uparrow}$ and the Markov property holds since no ancestor of X has an unblocked path to $L(v_2)$. This implies in particular it is sufficient to condition on $X^{(a)}$; all associated value functions compute the same value, i.e. $V(X^{(a)}) = V(X^{(b)}) = V(X^{(c)})$. The same applies to (d,e); note however than $V(X^{(d)}) \neq V(X^{(e)})$, though both sets are Markov as well. This is because neither $X^{(d)} \quad X^{(e)} \quad X^{(d)\uparrow}$ nor $X^{(e)} \quad X^{(d)} \quad X^{(e)\uparrow}$ hold. While the set inclusion property holds for (e,f), i. e. $X^{(f)} \quad X^{(e)} \quad X^{(f)\uparrow}$, we actually have $V(X^{(e)}) \neq V(X^{(f)})$. This is because $X^{(f)}$ is not Markov, which leads to an implicit marginalization of v_0 ($p(v_2|v_0, v_1) \neq \sum_{v_0} p(v_0|v_1)p(v_2|v_1, v_0)$).

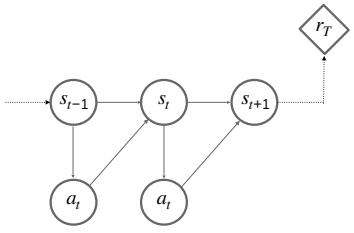

Figure 5: *Bellman equation in RL.*
The Bellman equation for Markov Decision Processes with single terminal reward r_T. The Q-function is $Q(s_t, a_t) = \mathbb{E}[r_T|s_t, a_t]$, the value function $V(s_t) = \mathbb{E}[r_T|s_t]$. From Lemma 2, $V(s_t) = \mathbb{E}[Q(s_t, a_t)|s_t]$. Since s_{t+1} is Markov and $s_t \quad s_{t+1}^{\uparrow} = \{s_j, a_{j-1}; j \quad t+1\}$, from Lemma 3, we have $V(s_t) = \mathbb{E}[V(s_{t+1})|s_t]$.

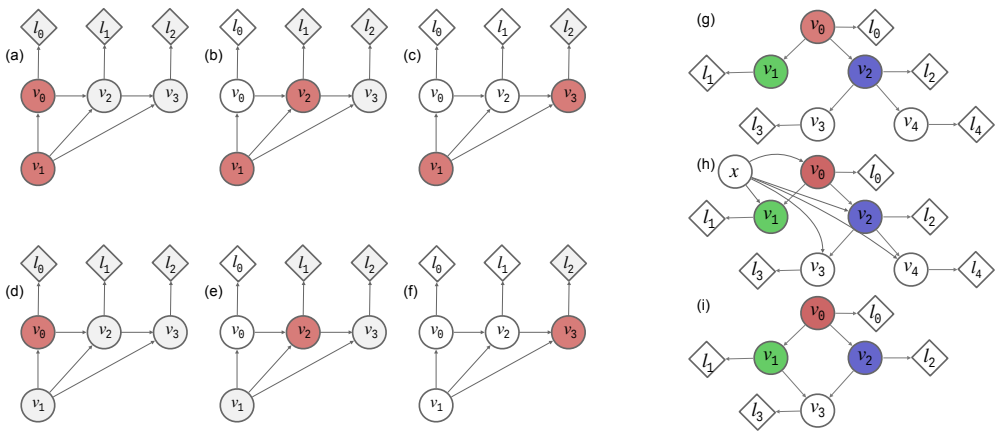

Figure 6: *Bootstrapping.* The value functions in (a,b,c) and (d,e,f) illustrate, for the same graph, ensembles of value functions that do and do not allow bootstrapping respectively. For (a,b,c) we have $V(v_0; v_1) = \mathsf{E}_{v_2;v_3|v_0;v_1}[\grave{}_1 + \grave{}_2] + \grave{}_0 = \mathsf{E}_{v_2|v_0;v_1}[\mathsf{E}_{v_3|v_2;v_1}[\grave{}_2] + \grave{}_1] + \grave{}_0 = \mathsf{E}_{v_2|v_0;v_1}[\mathsf{E}_{v_3|v_2;v_1}[V(v_3)] + \grave{}_1] + \grave{}_0 = \mathsf{E}_{v_2|v_0;v_1}[V(v_1; v_2)] + \grave{}_0$. In (d,e,f) a Markov chain is conditioned on an additional shared parent v_1. Here, $V(v_0) = \mathsf{E}_{v_1;v_2;v_3|v_0}[\grave{}_1 + \grave{}_2] + \grave{}_0$ cannot be expressed in terms of $V(v_2) = \mathsf{E}_{v_1;v_3|v_2}[\grave{}_2] + \grave{}_1 = \mathsf{E}_{v_1;v_3|v_2}[V(3)] + \grave{}_1$ due to the implicit marginalization over the shared parent in each value function. Note, however, that $V(v_0)$ can be expressed in terms of $V(v_0; v_2)$. This is akin to a POMDP that can be translated into a MDP by treating the entire history as the state (cf. Section D.1). (g,h,i) provide additional examples for three related graphs: In (g), thanks to Markov properties, value functions at parent nodes can be expressed in terms of the value functions of the children, e.g. $V(v_0) = \grave{}_0 + \mathsf{E}_{v_1|v_0}[V(v_1)] + \mathsf{E}_{v_2|v_0}[V(v_2)]$. In (h), simple Markov properties are missing, so bootstrap is possible only when value functions are additionally conditioned on x, i.e. $V(v_0; x) = \grave{}_0 + \mathsf{E}_{v_1|v_0;x}[V(v_1)] + \mathsf{E}_{v_2|v_0;x}[V(v_2; x)]$, or when the X_v form supersets of each other $V(v_0) = \grave{}_0 + \mathsf{E}_{v_1|v_0}[V(v_1; v_0)] + \mathsf{E}_{v_2|v_0}[V(v_2; v_0)]$. But it is not possible to express e.g. $V(v_0)$ in terms of $V(v_2)$. In (i) the cost does not decompose if value functions $V(v_0); V(v_1); V(v_2)$ are naively defined in terms of all downstream costs.

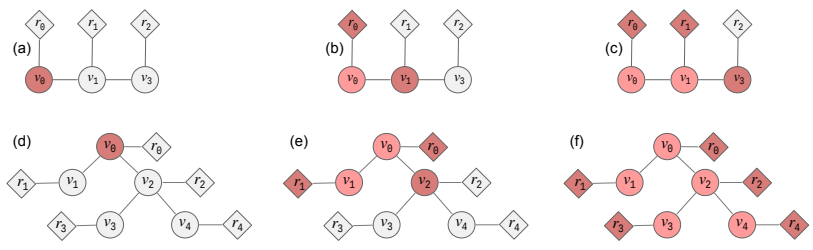

Figure 7: *Partial averages.* Examples of partial averages for a chain-structured and tree-structured graph respectively. In both cases the picture shows critics for node, computing the conditional expectations of $R = r_0 + r_1 + r_2$. Conditioning nodes are in red; dark red nodes contain information, but light nodes do not contain information not already contained in a dark node. The corresponding value functions are: a) $V(v_0)$, b) $r_0 + V(v_1)$, c) $r_0 + r_1 + V(v_3)$, d) $V(v_0)$, e) $r_0 + r_1 + V(v_2)$, f) $r_0 + r_1 + r_2 + r_3 + r_4$.

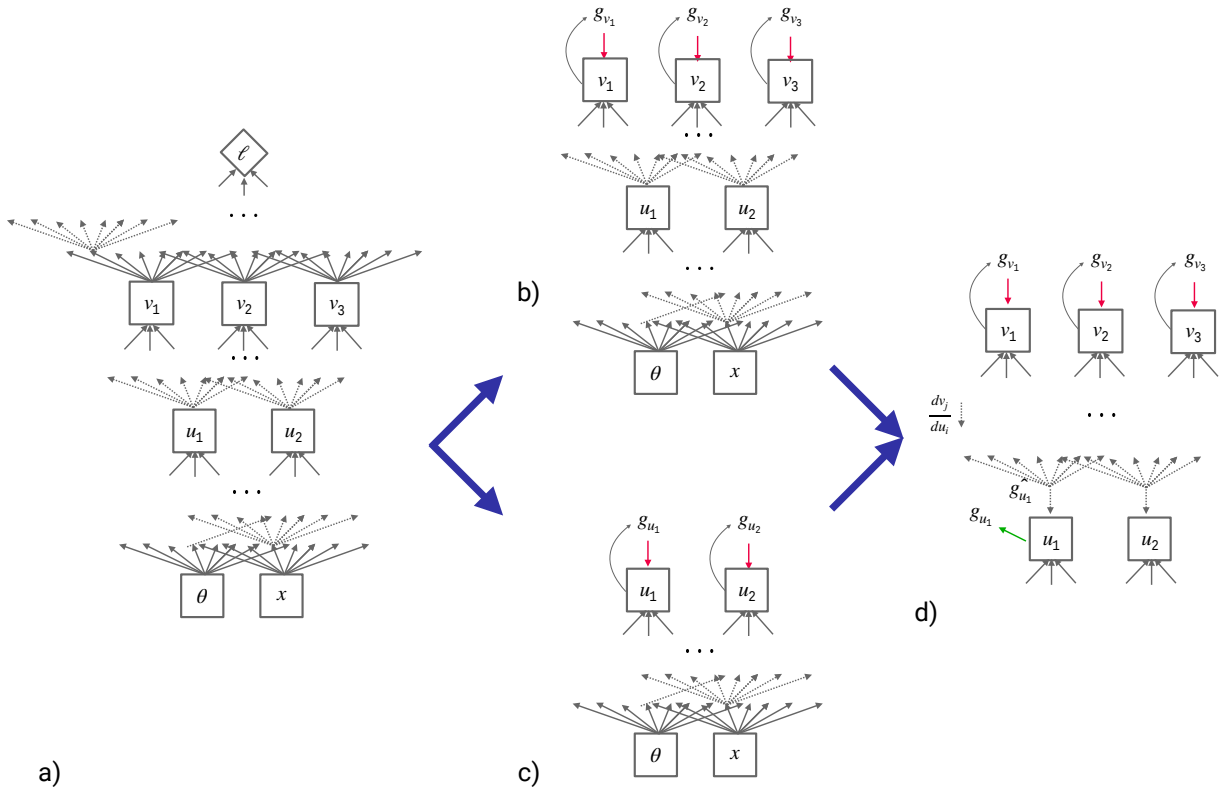

Figure 8: *Gradient-critics.*
a) A computation graph where $fu_1; u_2g$ and $fv_1; v_2; v_3g$ are both unordered separator sets. For the latter for instance, every path from to ` goes through $fv_1; v_2; v_3g$, and given the set of all parents of $fv_1; v_2; v_3g$, they can be computed in any order.
b) The (horizon) gradient-critic for set $fv_1; v_2; v_3g$: the forward and backward computations occurring after the separator set can be replaced by a gradient-critic: $\frac{d`}{d} = \sum_i g_{v_i} \frac{dv_i}{d}$.
c) The gradient-critic for set $fu_1; u_2g$.
d) The gradient-critic bootstrap: for node u_1 one can use the separator set $fv_1; v_2; v_3g$ to estimate the gradient of the loss with respect to u_1 as the 'partial gradient-critic' $\hat{g}_{u_1} = \sum_i g_{v_i} \frac{dv_i}{du_1}$. The gradient critic g_{u_1} can be regressed against either the empirical gradient $\frac{dL}{du_1}$ or the partially averaged gradient $\hat{g}_{u_1}$.

B Gradient estimator for expectation of a single function

Likelihood ratio estimator For a random variable x parameterized by , i.e. $x \sim p_\theta(\,)$, the gradient of an expectation can be obtained using the following estimator: $\frac{d}{d\theta}\mathbb{E}_{p\ (x)}[`(x)] = \mathbb{E}_{p\ (x)}\ \frac{d}{d\theta} \log p_\theta(x)`(x)$. This classical result from the literature is known under a variety of names, including likelihood ratio estimator, score function estimator, or REINFORCE, and can readily be derived by noting that $\frac{d}{dz} \log g(z) = \frac{1}{g(z)} \frac{d}{dz} g(z)$.

Pathwise derivative estimator In many cases, a random variable $x \sim p_\theta$ can be rewritten as a differentiable, deterministic function $x(\ ;\ )$ of a fixed random variable with parameterless distribution p_ϵ[7]. A change of variables from x to (reparameterization) removes the dependency of the expectation on , and now only appears inside the expectation and the gradient can be straightforwardly estimated: $\frac{d}{d\theta}\mathbb{E}_\epsilon[`(x(\ ;\ ))] = \mathbb{E}_\epsilon\ \frac{d}{d\theta}`(x(\ ;\ ))\ = \mathbb{E}_\epsilon\ \frac{\partial \ell}{\partial x} \frac{\partial x(\epsilon,\theta)}{\partial \theta}$ (Kingma & Welling, 2013; Rezende et al., 2014).

Other estimators have been introduced recently, relying on the implicit function theorem (Figurnov et al., 2018), continuous approximations to discrete sampling using Gumbel random variables (Maddison et al., 2016; Jang et al., 2016), law of large numbers applied to sums of discrete samples (Bengio et al., 2013), and others.

[7]Note that reparameterization is always possible, but differentiable reparameterization is not.

C Gradient-critic extensions

For variables x, y, and a set of variables $v_1, \dots, v_d$, $\frac{dy}{dx}|_{(v_1,\dots,v_d)}$ is the total gradient of y with respect to x, but keeping the values of v_i constant (i. e. the nodes v_i are not back-propagated through when computing $\frac{dy}{dx}$).

The backpropagation equations state that:

$$\frac{dL^s}{du} = \sum_i \frac{dL^s}{dv_i}\frac{\partial v_i}{\partial u} \tag{6}$$

and we recall the more general form given in 7:

$$\mathbb{E}_{\mathcal{G}}\left[\frac{dL^s}{du}\right] = \mathbb{E}_{\mathcal{G}}\left[\sum_i \frac{dL^s}{dv_i}\frac{dv_i}{du}\right] : \tag{7}$$

C.1 General form of the 'horizon' backpropagation

We first extend equation (7) to the more general case where the separator set $(v_1, v_2, \dots, v_d)$ is not necessarily unordered. We only require the nodes to be topologically ordered (i.e. if v_i is an ancestor of v_j, $i < j$). The more general form of 'horizon' backpropagation is given by the following:

$$\frac{dL^s}{dv} = \sum_i \frac{dL^s}{dv_i}\frac{dv_i}{dv}\Big|_{v_1,\dots,v_{i-1}} \tag{8}$$

From which the corresponding horizon stochastic backpropagation immediately follows:

$$\mathbb{E}_{\mathcal{G}}\left[\frac{dL^s}{dv}\right] = \mathbb{E}_{\mathcal{G}}\left[\sum_i \frac{dL^s}{dv_i}\frac{dv_i}{dv}\Big|_{v_1,\dots,v_{i-1}}\right] \tag{9}$$

C.2 Notational hazards for gradients

Before going further, we work out through a few examples to highlight subtleties regarding the notation used for gradients, as well as the backpropagation equations given in equation (6) and in particular the version found in (9). Note the latter is equivalent to backpropagation, but written in a way which tries to decouple the backpropagation updates which happen 'upstream' and 'downstream' of a group of nodes which are not necessarily the children of v.

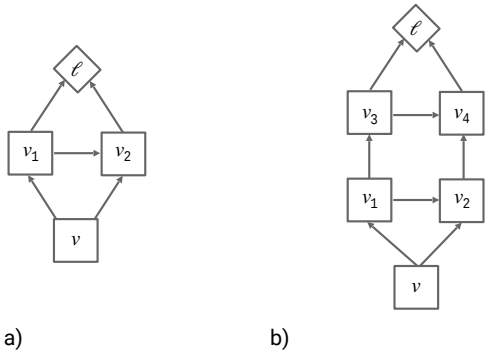

Figure 9: *Examples for backpropagation equations.* a) In this example, we have $v_1 = f_1(v)$, $v_2 = f_2(v, v_1)$ and $\ell = f_\ell(v_1, v_2)$.
b) An example for the total gradient version of backpropagation. In this example, we have the following equations:
At the first layer, $v_1 = f_1(x)$ and $v_2 = f_2(v_1, x)$
At the second layer, $v_3 = f_3(v_1)$ and $v_4 = f_4(v_2, v_3)$
Finally, $\ell = f_\ell(v_3, v_4)$.

We wish to compute $\frac{d\ell}{dv}$. We first work through the first example. Following the chain rule, we have

$$\frac{d\ell}{dv} = \frac{\partial \ell}{\partial v_1}\frac{\partial v_1}{\partial v} + \frac{\partial \ell}{\partial v_2}\frac{\partial v_2}{\partial v_1}\frac{\partial v_1}{\partial v} + \frac{\partial v_2}{\partial v} \tag{10}$$

We also have the following equations:

$$\frac{d\grave{}}{dv_1} = \frac{@\grave{}}{@v_1} + \frac{@\grave{}}{@v_2}\frac{@v_2}{@v_1} \quad ; \quad \frac{d\grave{}}{dv_2} = \frac{@\grave{}}{@v_2} \quad ; \quad \frac{dv_1}{dv} = \frac{@v_1}{@v} \quad ; \quad \frac{dv_2}{dv} = \frac{@v_2}{@v} + \frac{@v_2}{@v_1}\frac{@v_1}{@v} \tag{11}$$

From which we have the two equivalent forms of the chain rule. The first is equation (6), and is the last computation performed by backpropagation:

$$\frac{d\grave{}}{dv} = \frac{d\grave{}}{dv_1}\frac{@v_1}{@v} + \frac{d\grave{}}{dv_2}\frac{@v_2}{@v} \tag{12}$$

The second is related to forward differentiation:

$$\frac{d\grave{}}{dv} = \frac{@\grave{}}{@v_1}\frac{dv_1}{dv} + \frac{@\grave{}}{@v_2}\frac{dv_2}{dv} \tag{13}$$

We now move on to the second example. Again, fully expanding the chain rule, and summing over all paths, we have:

$$\frac{d\grave{}}{dv} = \frac{@\grave{}}{@v_3}\frac{@v_3}{@v_1}\frac{@v_1}{@v} + \frac{@\grave{}}{@v_4}\frac{@v_4}{@v_3}\frac{@v_3}{@v_1}\frac{@v_1}{@v} + \frac{@\grave{}}{@v_4}\frac{@v_4}{@v_2}\frac{@v_2}{@v_1}\frac{@v_1}{@v} + \frac{@\grave{}}{@v_4}\frac{@v_4}{@v_2}\frac{@v_2}{@v} \tag{14}$$

Let us first consider the horizon set $\{v_2, v_3\}$. Note this set is unordered, we could compute v_2 first, then v_3, or vice-versa. Equation (7) applies, which takes the form: $\frac{d\ell}{dv} = \frac{d\ell}{dv_3}\frac{dv_3}{dv} + \frac{d\ell}{dv_2}\frac{dv_2}{dv}$ follows, which can be verified from the following equations:

$$\frac{d\grave{}}{dv_3} = \frac{@\grave{}}{@v_3} + \frac{@\grave{}}{@v_4}\frac{@v_4}{@v_3} \quad ; \quad \frac{d\grave{}}{dv_2} = \frac{@\grave{}}{@v_4}\frac{@v_4}{@v_2} \quad ; \quad \frac{dv_3}{dv} = \frac{@v_3}{@v_1}\frac{@v_1}{@v} \quad ; \quad \frac{dv_2}{dv} = \frac{@v_2}{@v_1}\frac{@v_1}{@v} + \frac{@v_2}{@v} \tag{15}$$

Next, consider the horizon set $\{v_3, v_4\}$. Note that v_3 has to be computed before v_4. First note we have $\frac{d\ell}{dv_3} = \frac{\partial\ell}{\partial v_3} + \frac{\partial\ell}{\partial v_4}\frac{\partial v_4}{\partial v_3}$ and $\frac{d\ell}{dv_4} = \frac{\partial\ell}{\partial v_4}$. We can compute the total gradients of v_3 and v_4 with respect to v:

$$\frac{dv_3}{dv} = \frac{@v_3}{@v_1}\frac{@v_1}{@v} \tag{16}$$

$$\frac{dv_4}{dv} = \frac{@v_4}{@v_3}\frac{@v_3}{@v_1}\frac{@v_1}{@v} + \frac{@v_4}{@v_2}\frac{@v_2}{@v_1}\frac{@v_1}{@v} + \frac{@v_4}{@v_2}\frac{@v_2}{@v} \tag{17}$$

A naive application of horizon backprop (equation (7)), i.e. $\frac{d\ell}{dv} = \frac{d\ell}{dv_3}\frac{dv_3}{dv} + \frac{d\ell}{dv_4}\frac{dv_4}{dv}$ is **incorrect**, because the term $\frac{\partial\ell}{\partial v_4}\frac{\partial v_4}{\partial v_3}\frac{\partial v_3}{\partial v_1}\frac{\partial v_1}{\partial v}$ is double-counted. Following equation (9) instead, we keep v_3 constant when computing the gradient of v_4 with respect to v, and obtain:

$$\left.\frac{dv_4}{dv}\right|_{v_3} = \frac{@v_4}{@v_2}\frac{@v_2}{@v_1}\frac{@v_1}{@v} + \frac{@v_4}{@v_2}\frac{@v_2}{@v} \tag{18}$$

and we correctly have $\frac{d\ell}{dv} = \frac{d\ell}{dv_3}\frac{dv_3}{dv} + \frac{d\ell}{dv_4}\frac{dv_4}{dv}j_{v_3}$, as per equation (7). Note we also have $\frac{d\ell}{dv} = \frac{d\ell}{dv_3}j_{v_4}\frac{dv_3}{dv} + \frac{d\ell}{dv_4}\frac{dv_4}{dv}$, which is closer to equation (13).

C.3 Extension to the gradient-critic theorem in the general case

In this section, we provide an extension to the gradient-critic theorem 5 to the more general case.

Theorem 8. *If for each i, $\left.\frac{dv_i}{dv}\right|_{v_1,\dots,v_{i-1}}$ and $\frac{dL^s}{dv_i}$ are conditionally independent given C_{v_i},*

$$\mathbb{E}_{\mathcal{G}}\left[\frac{dL^s}{dv}\right] = \mathbb{E}_{\mathcal{G}}\left[\sum_i g_{v_i}\left.\frac{dv_i}{dv}\right|_{v_1,\dots,v_{i-1}}\right]$$

Similarly:

Theorem 9. *Consider a node v, separator set $(v_1; \dots; v_d)$. Consider critics g_v with set C_v and $(g_{v_1}; g_{v_2}; \dots; g_{v_d})$ with sets $(C_{v_1}; \dots; C_{v_d})$. Suppose that for all i, $C_{v_i} \quad C_v$ and let $\quad_i = C_{v_i} \cap C_v$. Then,*

$$g_v = \sum_i \mathbb{E}_{\quad_i | C_v} \left[g_{v_i} \frac{dv_i}{dv}_{(v_j)_j \quad_i} \right] \tag{19}$$

Note that while the quantities $\frac{dv_i}{dv} j_{v_1,\dots,v_i \quad 1}$ may appear intimidating, they are implicitly the coefficients applied by back-propagation to the loss sensitivity $\frac{dL^s}{dv_i}$ at a node v_i when computing $\frac{dL^s}{dv}$. In other words, if one replaces or initializes $\frac{dL^s}{dv_i}$ by g_{v_i} in the backward graph and proceeds with backpropagation as usual, the resulting quantity compute at the root node will be effectively $\sum g_{v_i} \frac{dv_i}{dv} j_{(v_j)_j \quad_i}$; this implies that the bias in the gradient estimator will only come from the bias in critics themselves (not how they are used), and that for asymptotically perfect critics, the gradient estimator will be unbiased.

D Applications in the RL literature

In this section and next we discuss multiple examples of models from the RL and probabilistic modeling literature. We present the corresponding SCGs and associate surrogate losses.

D.1 Markov decision processes and partially observed Markov decision processes

Figure 10 shows several examples of decision processes from the reinforcement learning literature. For simplicity we focus on the undiscounted, finite horizon case but note that generalizations to the discounted, infinite horizon case are straightforward.[8]

Markov decision processes (MDPs) As explained in in Fig. 1 the MDP objective is the (in our case undiscounted) return $\mathbb{E}_\tau[\sum_{t=1}^T r(s_t; a_t)]$ where trajectories $\quad = (\dots; s_t; a_t; \dots)$ are drawn from the distribution obtained from the composition of policy $\quad_\theta$ and system dynamics: $p(\ ; \ ) = p(s_0) \prod_t \quad (a_t | s_t; \ ) p(s_{t+1} | s_t; a_t)$.

Fig. 10(a) shows a vanilla, undiscounted MDP with a policy $\quad_\theta(a|s)$ parameterized by $\ $. A large number of different estimators have been proposed for this model using a variety of different critics including Monte-Carlo returns, k-step returns, $\ $-weighted returns etc.:

$$Q^{MC}_{a_t} = \sum_{t' \geq t} r(s_{t'}; a_{t'}) \tag{20}$$

$$Q^0_{a_t} = Q_{a_t}(s_t; a_t) \tag{21}$$

$$Q^k_{a_t} = \sum_{t'=t}^{t+k} r(s_{t'}; a_{t'}) + V(s_{t+k+1}); \quad \forall k > 0 \tag{22}$$

$$Q^\lambda_{a_t} = (1 \quad) \sum_{k=0}^{\infty} \quad^k Q^k_{a_t}. \tag{23}$$

The $\ $-weighted returns are an instance of a convex combination of a different set of critics, in this case of k-step returns critics. K-step returns are examples of partial averages. These critics can be used both to estimate the advantage for a policy update using the policy gradient theorem. Furthermore, they can be valuable as bootstrap targets for learning value functions. In general, the critic used for estimating the advantage can be different from the one used to construct a target for the value function update.

Fig. 10(c) shows an example of an MDP where independence between the components of a multi-dimensional action is assumed, corresponding to a factorized policies with $\quad_\theta(a_t | s_t) = \prod_i \quad_\theta(a^i_t | s_t)$.[9] This motivates the use of action-conditional baselines (e.g. Wu et al., 2018) or marginalized critics. For instance, the action-conditional baseline for action

[8] In fact, the finite horizon requires particular care in that policies and value functions become time-indexed but we ignore this here in favor of a simple notation.

[9] This is the predominant case in practice, especially in continuous action spaces where policies are frequently chosen to be factorized Gaussian distributions.

dimension i, in state s_t is given by $V_{a_t^i}(s_t; a_t^{-i}) = \mathsf{E}_{a_t^i|s_t,a_t{}^i}[Q(s_t; a_t)]$, where $a_t^{-i} = (a_t^1; :::; a_t^{i-1}; a_t^{i_1}; :::a_t^N)$. This is a valid baseline value function according to our Def. 3 as the remaining a_t^{-i} are non-descendants of the action a_t^i.

Partially Observable Markov Decision Processes (POMDPs) Fig. 10(c,d,e) show two examples of POMDPs. (c) shows the standard setup where the state s_t is unobserved. Information about the state of the environment are available to the agent only via observations $o_t \quad p(\,js_t)$. Observations typically provide only partial information about the state. To act optimally (or to predict the value in some state) the agent therefore needs to infer (the distribution over) the state at timestep t given the interaction history $h_t = (o_0; a_0; :::; o_t)$: $p(s_t jh_t)$. The aggregated interaction history is often referred to the internal agent state or belief state b_t. Internal state and action choice are usually trained from return (prediction - when training value functions - and maximization - when optimizing the policy). Note that when training the internal state from returns only, there is no guarantee that b_t will correspond to a true 'belief state', e.g. the sufficient statistics of the filtering distribution $p(s_t jo_0; a_0; ::: o_t)$; for a discussion of differences between internal and belief states, see for instance (Igl et al., 2018; Gregor et al., 2018; Moreno et al., 2018).

The Markov structure of the model naturally suggests that value functions for time step t should be conditioned on the entire observation history up to time-step t. Since the policy shown in the figure is dependent on the entire observation history, a critic has to be conditioned on the entire observation history (through b_t) too in order to satisfy Definition 4. Furthermore, the Markov property of the model also requires conditioning of the value function on the entire observation history for bootstrapping to be valid, independent of the dependency structure chosen for the policy.

But the theorems presented in this paper suggest interesting, less explored alternatives, in particular when the state s_t is available at training time (but not at testing time, so that the agent policy cannot depend on s_t). For instance, since s_t is a non-descendendant of the action a_t, the baseline for action a_t may be trained to depend on the full state, for instance by using a value function $V(s_t; b_t)$. This baseline is likely to be significantly more accurate since it has access to information which may be very predictive of the return. It can also be used to help train the internal state b_t of the agent better, since $V(s_t; b_t)$ is a valid, lower variance bootstrap target for training the value function $V(b_t)$, which in turn will affect the representation b_t learned by the agent. s_t may also be used for critics, for instance by using Q functions which depend on both the environment and agent state: $Q(s_t; b_t; a_t)$.

The example in Fig. 10(d) is a special case of the general POMDP. Shown is a multi-task MDP with shared transition dynamics but with reward function that depends on a goal g (which varies across tasks). The Markov structure suggests conditioning both policy and value functions on the goal variable g if observed (in which case the model is a MDP with g being part of the state), or the entire interaction history when g is unobserved. As for a general POMDP setting, conditioning on g or on the state-history is optional for the policy but required for bootstrapping of the value function (of course performance will suffer when the policy does not have access to sufficient information).

Figure 10(e) shows a similar setup but with the transition dynamics dependent on an unobserved variable d affecting the dynamics. The same arguments as for (d) and (b) apply. The option of conditioning the value function but not the policy on the system dynamics d has been exploited e.g. in the sim-to-real work in (Peng et al., 2017). The setup gives better baselines and allows bootstrapping of the value function, while the policy learns to act robustly without knowledge of the true dynamics d.

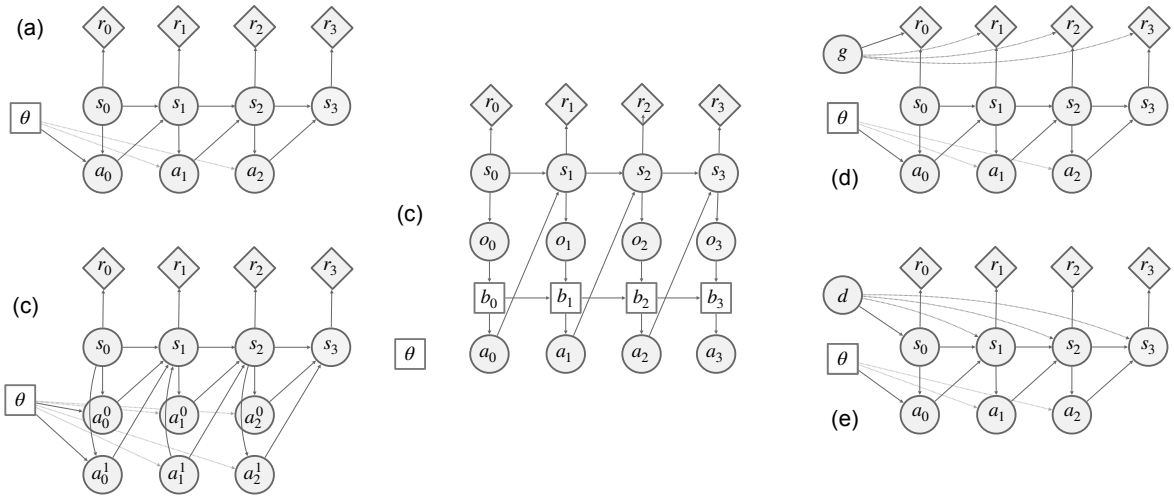

Figure 10: Example for SCG representations of decision processes. (a) Vanilla MDP. (b) MDP with two-dimensioal, independent (factored) actions a_t^0 and a_t^1. (c) General POMDP. (d) Multi-task MDP with goal variable g. (e) MDP with unobserved system dynamics d.

D.2 Reparameterized MDPs, value-gradients, and black-box policy search

Reparameterization The method of reparameterization is heavily exploited in the probabilistic modeling literature, but it can also be useful in RL by applying it to MDPs. Fig. 11 (a) shows again the regular MDP from Fig. 10a. Fig. 11 (b) shows the fully reparameterized version of Heess et al. (2015) where $(a|s)$ and $p(s_t|s_{t-1}; a_{t-1})$ are replaced by deterministic functions of independent noise: $a_t = (s_t; \ _t)$ and $s_t = f(s_{t-1}; a_{t-1}; \ _{t-1})$ respectively. (Deterministic policies and deterministic system dynamics can be treated as simple special cases of this general setup.) If the (differentiable) system dynamics and distributions of the noise sources $_t$ are know, we can use the standard backpropagation algorithms to compute policy gradients, as all stochastic nodes are root nodes in the graph (cf. e.g. Eqs. 6-8 in Heess et al. 2015). When the system dynamics are unknown or non-differentiable, gradients of learned critics $Q(s; a)$ w.r.t. the actions can be used to obtain gradients both for deterministic and stochastic policies, e.g.

$$\frac{@J}{@} = \sum_t \mathsf{E} \ \frac{@Q_{a_t}}{@a_t} \frac{@ \ _\theta}{@} \ ; \tag{24}$$

as discussed in Section 4.3. For a deterministic policy this corresponds to the *Deterministic Policy Gradients* (DPG) algorithm (Silver et al., 2014; Lillicrap et al., 2015); for stochastic policies it is a special case of the stochastic value gradients (SVG; Heess et al. 2015) family, SVG(0). The SVG(K) family also contains the analogue of partial averages. For instance, the policy gradient computed from 1-step MC returns (SVG(1)) is given by

$$\frac{@J}{@} = \sum_t \mathsf{E} \ \frac{@V_{s_{t+1}}}{@s_{t+1}} \frac{@s_{t+1}}{@a_t} + \frac{@r_t}{@a_t} \ \frac{@ \ _\theta}{@} \ : \tag{25}$$

We can further construct convex combinations (e.g. similar to -weighted returns) of such partial averages. These ideas have also been studied in the literature e.g. by Werbos (1982); Fairbank & Alonso (2012).

Black-box policy search These methods ignore the temporal structure of the MDP and instead perform search directly at the level of the parameter . A variety of different algorithms exist, with a particular simple form arising from representing an MDP in the equivalent from shown in Fig. 11(c). The standard score-function estimator is used to learn a distribution over policy parameters , which is parameterized by $_0$, such as mean and standard deviation of :

$$J(\ _0) = \mathsf{E}_{\theta|\theta_0} \left[\mathsf{E}_{\mathcal{G}|\theta} \left[\sum_t r(s_t; a_t) \right] \right] \tag{26}$$

The reparameterized version of this model is shown Fig. 11(d); it is closely related to a variety of recent proposals in the literature Fortunato et al. (2017); Plappert et al. (2017).

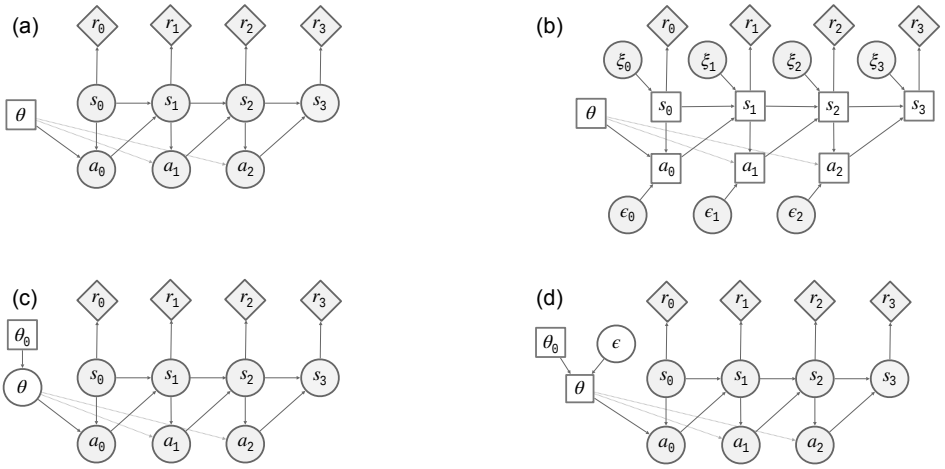

Figure 11: (a) MDP from Fig. 10 (for reference). (b) Fully reparameterized MDP as discussed in Heess et al. (2015). The stochastic action and state nodes have been replaced by deterministic nodes and independent noise variables. (c) Evolution strategies learn the parameters $_0$ of a distribution over parameters . (d) Same as (c) but with the distribution over reparameterized, similar to *noisy networks*.

D.3 Hierarchical RL and Hierarchical policies

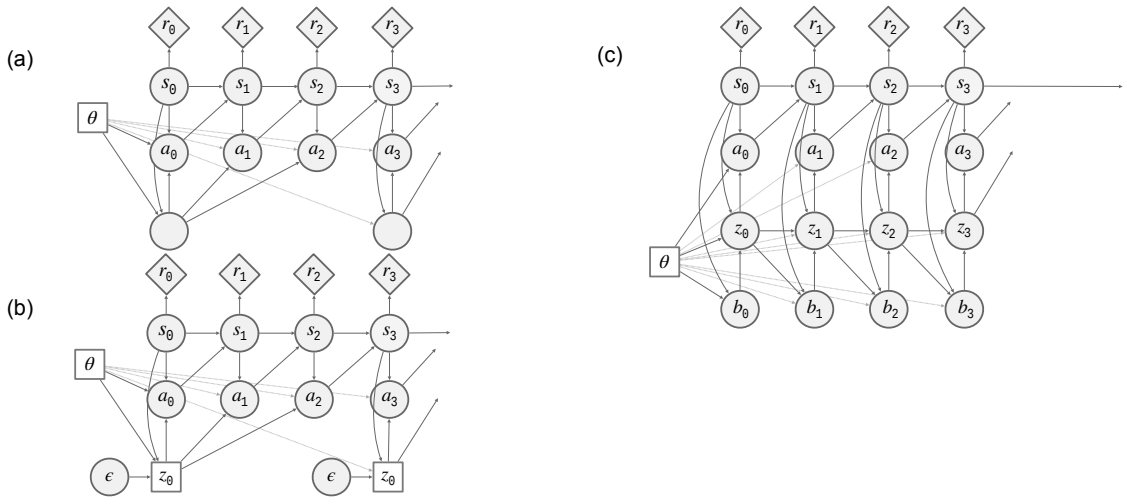

Figure 12: *Examples of "hierarchical" policies: a) Policy with latent variable ("option") that is fixed for* $K = 3$ *steps. b) similar to (a) but the option is reparameterized (as e.g. in Heess et al. (2016)). c) Options with variable duration as in the* Option Critic *(Bacon et al., 2017).*

Figure 12 shows several simple examples of MDPs in which the policies have been augmented with latent variables. Such latent variables can, for instance, be seen as implementing the notion of options. For the discussion we assume that the objective remains unchanged (i.e. that no additional loss terms are introduced and we aim to optimize the full architecture to maximize expected reward).

Fig. 12(a) shows a simple example of a policy with "options" that have a fixed duration of three steps. For options of duration M ($M = 3$ in the example) the trajectory distribution is drawn from $p(\ ;\ ) = p(s_0) \prod_{t=0,M,2M,\ldots} (z_t|s_t;\ ) \prod_{t'=0}^{M-1} (a_{t+t'}|s_{t+t'}; z_t;\ ) p(s_{t+t'+1}|s_{t+t'}; a_{t+t'})$. Below we also use $[\ ]_M$ to denote the "current" option, i.e. $[t]_M = bt/Mc M$.

A variety of gradient estimators can be constructed. The main change compared to the choices discussed in D.1 is that critic and baselines have to reflect the dependence of $\geq t$ on $z_{[t]_M}$. Valid baselines for z_t and a_t would, for instance, be $V_{z_t}(s_t) = \mathrm{E}_{\tau\ t|s_t}[\sum_{t' \geq t} r(s_{t'}; a_{t'})]$ for $t = 0; M; \ldots$ and $V_{a_t}(s_t; z_{[t]_M}) = \mathrm{E}_{\tau\ t|s_t, z_{[t]_M}}[\sum_{t' \geq t} r(s_{t'}; a_{t'})]$, and similarly for critics:

$$Q^0_{a_t} = Q_{a_t}(s_t; a_t; z_{[t]_M}) = \mathrm{E}_{\tau\ t|s_t, a_t, z_{[t]_M}}[\sum_{t' \geq t} r(s_{t'}; a_{t'})] \tag{27}$$

$$Q^1_{a_t} = \begin{cases} r(s_t; a_t) + V_{t+1}(s_{t+1}; z_{[t]_M}) & \text{if } [t+1]_M = [t]_M \\ r(s_t; a_t) + V_{t+1}(s_{t+1}) & \text{otherwise} \end{cases} \tag{28}$$

$$Q^0_{z_t} = Q_{z_t}(s_t; z_t) = \mathrm{E}_{\tau\ t|s_t, z_t}[\sum_{t' \geq t} r(s_{t'}; a_{t'})] \quad \text{for } t = 0; M; 2M; \ldots; \tag{29}$$

where we have made explicit that value functions may depend directly on the time step (due to the fixed duration of the option).

Fig. 12(b) shows the same model but with the random variables z_t being replaced by a deterministic function of independent noise $\ _t$. This allows the gradient with respect to $\ $ at z_t to be computed by backpropagation, e.g.

$$\frac{\partial}{\partial z_t} = \sum_{t'=t}^{t+M-1} \mathrm{E}_{\tau\ t|s_t, z_t} \frac{\partial Q_{a_{t'}}}{\partial a_{t'}} \frac{\partial a_{t'}}{z_t} \frac{\partial z_t}{\partial} \ ; \tag{30}$$

where $Q_{a_{t'}}$ may be a function of z_t as discussed in the previous paragraph.

Fig. 12(c) shows a more complex which captures the essential features e.g. of the *Option Critic* architecture of Bacon et al. (2017). Unlike in (a,b) the option duration is variable and option termination depends on the state. This can be modeled with a binary random variable b_t which controls whether z remains unchanged compared to the previous timestep. The full trajectory distribution is given by $p(\ ;\ ) = p(s_0) p(z_0|s_0;\ ) p(a_0|s_0; z_0;\ ) \prod_{t>0} p(s_{t+1}|s_t; a_t)\ (b_t|s_t)\ (z_t|s_t; b_t; z_{t-1};\ )\ (a_t|s_t; z_t;\ )$ where $p(z_t|s_t; b_t; z_{t-1};\ ) = (z_t \quad z_{t-1})^{b_t}\ (z_t|s_t;\ )^{b_t - 1}$.

Value functions of interest are, for instance, $Q_{a_t}(s_t; a_t; z_t) = \mathrm{E}_{\tau\ t|s_t, z_t}[\sum_t r(s_t; a_t)]$ (note the dependence on z_t due to the dependence of future time steps on that value); $Q_{z_t}(s_t; z_t) = V_{s_t, z_t}(s_t; z_t) = \mathrm{E}_{a_t, s_{t+1}|s_t, z_t}[r(s_t; a_t) + V_{s_{t+1}, z_t}(s_{t+1}; z_t)] = \mathrm{E}_{\tau\ t|z_t, s_t}[\sum_{t' \geq t} r(s_t; a_t)]] = \mathrm{E}_{a_t|s_t, z_t}[Q_{a_t}(s_t; a_t; z_t)]$, as well as $V_{s_t}(s_t; z_{t-1}; b_t = 1) = V_{s_t}(s_t) = \mathrm{E}_{z_t|s_t}[\mathrm{E}_{\tau\ t|z_t, s_t}[\sum_{t' \geq t} r(s_t; a_t)]]$, $V_{s_t, t\ \ 1}(s_t; z_{t-1}) = (B_t = 1|s_t) V(s_t) + (1 \quad (B_t = 1)) V_{s_t, z_t}(s_t; z_{t-1})$. Whereas the former two value functions are of interest as critics, the latter two are primarily interesting for bootstrapping purposes.

D.4 Multi-agent MDPs

Multi-agent MDPs can be seen as special cases of MDPs with a particular factorization. Two examples are shown in Fig. 13 for the fully and partially observed case respectively. The factored structure of the MDP suggests particular choices for value function and critic, including conditioning the baseline on other agents' actions in the same vein as for action conditional baselines (Foerster et al., 2017). A valid bootstrap target requires access to the full state. For a collection of agents $a\ 2\ A = (A; B; \ldots)$, denote u^a_t the action of agent a at time t. The tuple of all actions is $u = (a^A; a^B; \ldots)$; the tuple of all actions except that of agent a is u^{-a}. The shared state is s_t. The value function $Q(s_t; u_t)$ is a valid critic for action u^a_t (Def. 5). Since all actions other than u^a_t are non-descendents of u^a_t, they can be used to form an improved, valid baseline $V(s_t; u^{-a}_t)$. From the Bellman principle (Lemma. 3), we also have $V(s_t; u^{-a}_t) = \mathrm{E}[Q(s_t; u_t)] = \sum_{u^a_t} (u^a_t|s_t) Q(s_t; u_t)$

Note that in the fully observed case (all agents observe their own as well as the other agents' states), the above essentially takes the same form as the factored action model from Fig. 10, due to identical structure in the graphical model.

An actor-critic formulation that employs state-action critic to derive action-value gradients based has been considered by Lowe et al. (2017). Note that the use of action-value gradients means that no baseline is needed.

Fig. 10b shows the POMDP formulation of the same problem in which each agent has access to only a partial observation of the full system. While centralized baselines and critics may still be desirable, "factored" baselines and critics conditioned on the full history of observations accessible to each agents policy are also valid according to the definition of a critic (Def. 4) and admit bootstrapping according to Theorem 3.

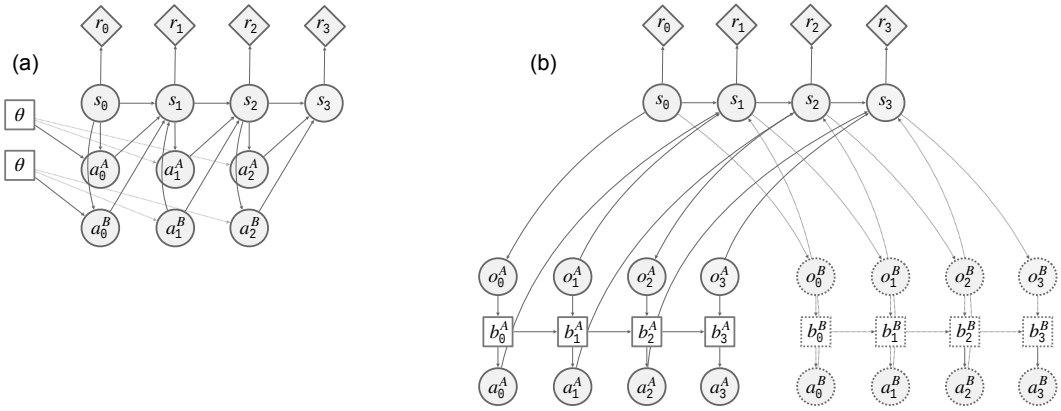

Figure 13: *Examples: (a) Multiagent MDP. (b) Multiagent POMDP*

E Applications in Probabilistic Modeling

E.1 Mapping inference in probabilistic models to stochastic computation graphs

In this section, we briefly explicit a general technique to turn probabilistic inference problems into stochastic computation graphs, in particular in the framework of variational inference. This section closely follows (Weber et al., 2015).

Let us consider an arbitrary latent variable model $p(\mathbf{z}, \mathbf{x}; \quad)$, where $\mathbf{x} = (x_1, \ldots, x_M)$ represents the observations, $\mathbf{z} = (z_1, \ldots, z_K)$ the latent variables, and $\quad = (\ _1, \ldots, \ _D)$ the parameters of p. Because $\mathbf{x}$, $\mathbf{z}$ and $\quad$ play very similar roles in what follows, for simplicity of notation, let y_j be a variable that represents a single latent $z_{j'}$ or observed variable $x_{j'}$ for some corresponding j' (for instance we could have $y_j = z_j$ for $j = 1, \ldots K$ and $y_{K+j} = x_j$ for $j = 1, \ldots, M$) Assume that the joint distribution of $(\mathbf{z}, \mathbf{x})$ is that of a directed graphical model with directed acyclic graph G^p; for any variable $v \in G^p$, let h^p_v be the parents of v in G^p (this includes parameters in $\quad$). We then have:

$$p(\mathbf{z}, \mathbf{x}) = \prod_k p(z_k | h^p_{z_k}) \prod_m p(x_m | h^p_{x_m}) = \prod_j p(y_j | h^p_{y_j}).$$

Let us consider a posterior distribution $q(\mathbf{z}|\mathbf{x}; \quad)$ which is also a directed graphical model G^q (G^p and G^q may have different topology). For a node v in G^q, let h_v be the parents of v in G^q (again, this includes parameters in $\quad$ or $\quad$). Then,

$$q(\mathbf{z}|\mathbf{x}) = \prod_k q(z_k | h_{z_k}).$$

The variational objective is $\mathrm{E}_q[\log p(x, z) \quad \log q(z|x)]$, which can be decomposed into a sum $\sum_j r_j(s_j)$, where for each j, we either have $r_j(s_j) = \log p(y_j | h^p_{y_j})$ with $s_j = (y_j, h_{y_j})$ or $r_j(s_j) = \quad \log q(z_j | h_{z_j})$ with $s_j = (z_j, h_{z_j})$.

The stochastic computation graph composed of G^q and costs r_j implements variational inference for p using variational distribution q.

E.2 Variational auto-encoders and Neural variational inference

Simple stochastic computation graphs can be obtained from the combination of a latent variable model $p(z;x)$ and an amortized posterior network $q(z|x)$ (Gershman & Goodman, 2014). In the most common implementations, the latent variable model maps a latent variable z with known prior (typically a normally distributed random vector, or a vector of Bernoulli random variables) to the distribution parameters (typically a Gaussian or Bernoulli) of the observation x. The model is in effect a mixture model (with an infinite number of components in the continuous case) where mixture components parameters are functions of the mixture 'index'.

When the network is not reparametrized, the score function estimator is commonly known as NVIL (neural variational inference and learning) or automated/black-box VI and has been developed in various works (Paisley et al., 2012; Wingate & Weber, 2013; Ranganath et al., 2013; Mnih & Gregor, 2014). Baseline functions are commonly used. When the distribution are reparametrizable, the model is often known as a VAE (variational autoencoder) (Kingma & Welling, 2013; Rezende et al., 2014); many variants of the VAE were since developed, often differing on the form of the posterior network, decoder, or variational bound.

Because of the simplicity and lack of structure of the model, it is not trivial to leverage critic or gradient-critics in these models. One could in principle learn a gradient-critic for NVIL and avoid the high-variance issue due to the score function estimator (see action parameter critics in section F); to our knowledge this has not been explicitly attempted. Multiple sample techniques (Mnih & Rezende, 2016) can be used to lower variance of the estimators by estimating value function as the empirical average of independent samples (the validity of using other samples as baseline for one sample automatically follows from value functions from non-descendent sets of variables).

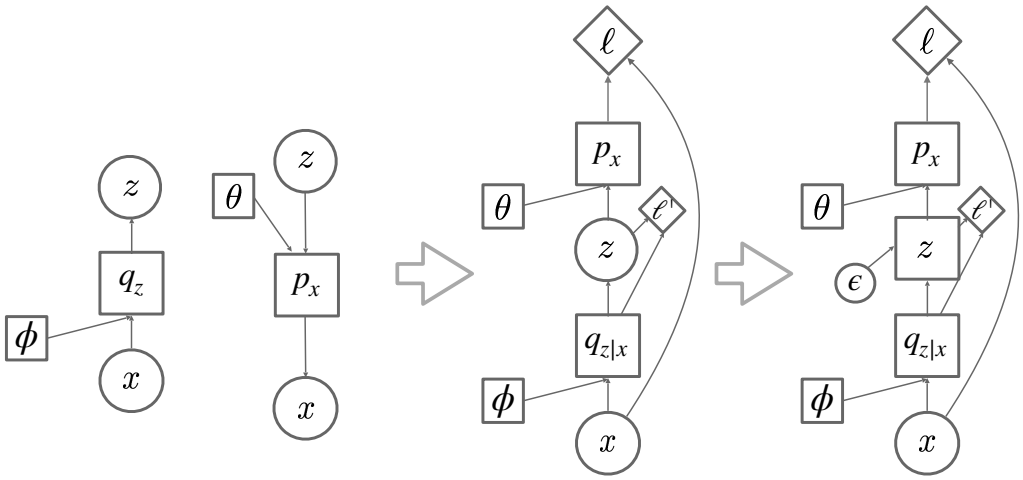

Figure 14: *Variational Autoencoder and Neural Variational Inference.*
a) Graphical model of neural latent variable model $p(x;z)$ and amortized posterior $q(z|x)$
b) Computation graph for Neural Variational Inference
c) Reparametrized computation graph for Variational Autoencoder.

E.3 State-space models

State-space models are powerful models of sequential data $\mathbf{x} = (x_1;\dots;x_T)$, which capture dependencies between variables by way of a sequence of *states* z_t,

There is a large amount of recent work on these type of models, which differ in the precise details of model components (Bayer & Osendorfer, 2014; Chung et al., 2015; Krishnan et al., 2015; Archer et al., 2015; Fraccaro et al., 2016; Liu et al., 2017; Buesing et al., 2018).

They generally consist of decoder or prior networks, which detail the generative process of states and observations, and encoder or posterior networks, which estimate the distribution of latents given the observed data.

Let $\mathbf{z} = (z_1, \dots, z_T)$ be a state sequence and $\mathbf{x} = (x_1, \dots, x_T)$ an observation sequence. We assume a general form of state-space model, where the joint state and observation likelihood can be written as $p(\mathbf{x}, \mathbf{z}) = \prod_t p(z_t \mid z_{t-1}) p(x_t \mid z_t)$.[10] These models are commonly trained with a VAE-inspired bound, by computing a posterior $q(\mathbf{z} \mid \mathbf{x})$ over the states given the observations. Often, the posterior is decomposed autoregressively: $q(\mathbf{z} \mid \mathbf{x}) = \prod_t q(z_t \mid z_{t-1}, \psi_t(\mathbf{x}))$, where ψ_t is a function of $(x_1, \dots, x_t)$ for filtering posteriors or the entire sequence $\mathbf{x}$ for smoothing posteriors. This leads to the following lower bound:

$$\log p(\mathbf{x}) \quad \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} \mid \mathbf{x})} \left[\sum_t \log p(x_t \mid z_t) + \log p(z_t \mid z_{t-1}) \quad \log q(z_t \mid z_{t-1}, \psi_t(\mathbf{x})) \right] : \tag{31}$$

Continuous state-space models are often reparametrized (see (Krishnan et al., 2015; Buesing et al., 2018)). Comparatively less work investigates state-space model with discrete variables (see (Gan et al., 2015) for a recurrent sigmoid network, and (Eslami et al., 2016; Kosiorek et al., 2018) for hybrid models with both discrete and continuous variables). As with NVIL, discrete models suffer from higher variance estimates and could be combined with critics and gradient-critics, though it has not been extensively investigated.

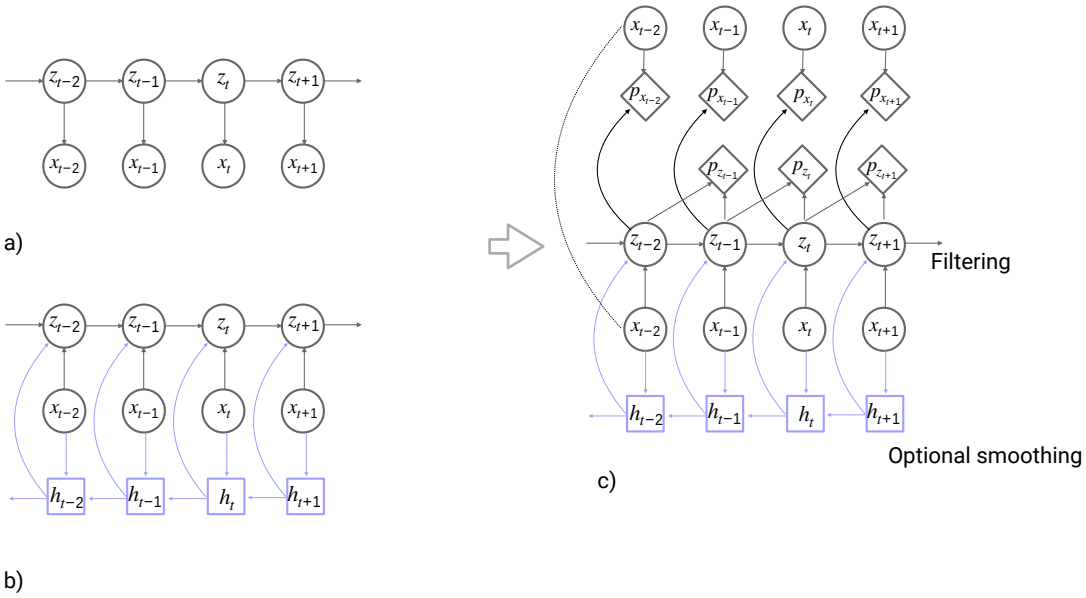

Figure 15: *State-space model.*
a) Forward model or decoder $p(z; zx)$
b) Inverse model or encoder $q(zjx)$ based on filtering or smoothing (with purple arrows) posterior
c) Stochastic computation graph

E.3.1 More general inference schemes; RL as Inference

The previous section addresses inference from a variational point of view, which maps closely to the RL problem[11]. More general inference schemes can be used, from message-passing to sequential Monte-Carlo. While we do not extensively cover the connections, many of the notions developed in the previous sections can be extended to the general case. The main difference is that value-function, which conserve the intuition of 'summarizing the future', are now defined in a 'soft' fashion, by way of log-sum-exp operators (so do the corresponding bootstrap equations). For instance, for a state-space model, we can define a value function $V(z_t)$ as $V(z_t) = \log p(x_{t+1}, \dots, x_T \mid z_t)$ and $Q(z_t, z_{t+1}) = \log p(x_{t+1}, \dots, x_T \mid z_t, z_{t+1})$;

[10] For notational simplicity, $p(z_1 \mid z_0) = p(z_1)$.

[11] Note that an important distinction between RL and inference is that in RL, costs typically depend only on states and actions, while in inference, the cost depends on the parametrization of the random variables as well as on their samples; this is the reason why optimal posterior distributions are still entropic, as opposed to deterministic optimal policies in MDPs when not using entropy bonus.

they are connected through a soft-Q update:

$$V(z_t) = \log \mathsf{E}[\exp(Q(z_t; z_{t+1}))|z_t]:$$

The connection between inference and reinforcement learning is explored in details in (Levine, 2018), see also soft-actor critic methods (Haarnoja et al., 2018). Just as value functions can be used to improve the quality of variational inference scheme (an idea proposed in (Weber et al., 2015)), soft-value functions can be used to improve the quality of other inference schemes, for instance sequential Monte-Carlo (Heng et al., 2017; Piché et al., 2019).

F A worked example

In this section, we go in more details through a simple chain graph example. This will allow us to present the menu of estimators discussed in the previous section in a concrete situation, without having to deal with the complexities arising from more structured graphs.

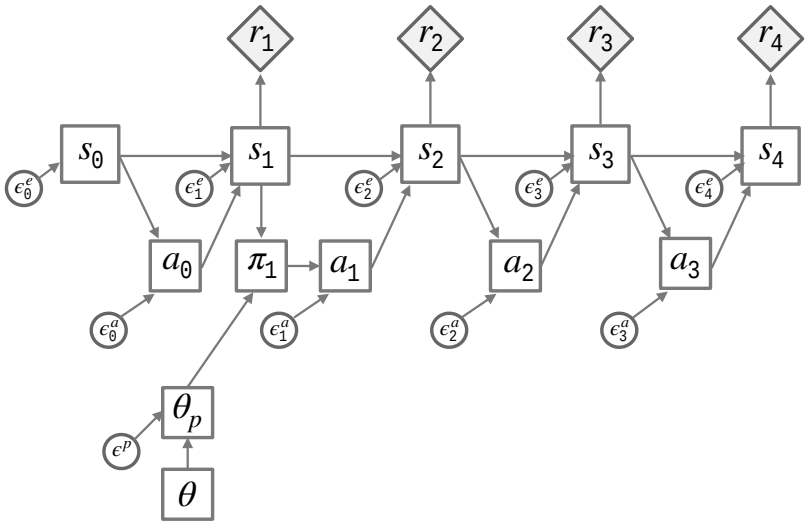

Figure 16: A simple MDP example, here shown fully reparametrized. Only parameters of the policy at time-step $t = 1$ are shown.

We will use reinforcement learning terminology in order to connect to the wide literature on policy gradient and value-based methods, however the example is more general, and can be mapped to inference and learning in state-space models, time-series analysis, etc.

In this example, the state is fully observed by the agent, and we assume every function in the graph is known and differentiable. We assume the graph can be arbitrarily reparametrized (with environment noise variables ^{e}, action noise ^{a}, parameter noise ^{p}). We assume a distribution over the parameters $_p$ used to compute the policy; this distribution has hyperparameter . That distribution may be a Dirac at (i.e. $= {}_p$).

The aim is to compute the parameters of the policy for the second action a_1 (this is done for simplicity but can easily be extended to the case where policy parameters are shared across time-steps); for reasons which will be apparent later, we make the parameters $_1$ of the distribution of action a_1 a node of the graph (for other actions, those parameters are implicit in the state to action mappings); the mapping from $_1$ and $^a_1$ to a_1 is parameterless. We detail different estimators of the gradient of the expected loss $\mathsf{E}[R] = \mathsf{E}[\sum_i r_i]$ with respect to .

Black-box estimators: no reparametrization.

The first and simplest estimator is the black-box or 'evolution strategies' estimator, obtained by not reparametrizing $_p$;

it is given by $\frac{d \log p(\theta_p|\theta)}{d\theta}(R \quad V(\ ))$. It is hard to compute a value baseline or value critic for R here. A critic that could be used in principle is an estimator $V(\ _p)$ which averages over the entire environment and policy, and produces the expected return as a function of the sampled parameter $\ ^p$. The corresponding estimator would be $\frac{d \log p(\theta_p|\theta)}{d\theta}(V(\ _p) \quad V(\ ))$. It is of course highly unlikely to be accurate in practice.

Reparametrizing policy parameters: policy gradient algorithms.

The next family of estimators is obtained by reparametrizing $\ _p$, but not the action or the environment. If the conditional distribution of is a Dirac (i.e. noiseless), we have classical version of the estimators; if not, we have 'noisy' or Bayesian estimators(Blundell et al., 2015; Fortunato et al., 2017; Plappert et al., 2017).

We obtain 'model-free' estimators, which differ by the choice of the critic. One choice averages the entire future with a critic $Q(s_1; a_1)$, which leads to the update

$$\frac{@}{@\ _p}\frac{d \log\ _1(a_1)}{d}(Q(s_1; a_1) \quad V(s_1)):$$

Other choices involve the use of k-step returns, e.g.

$$\frac{@}{@\ _p}\frac{d \log\ _1(a_1)}{d}(r_1 + V(s_2) \quad V(s_1));$$

$$\frac{@}{@\ _p}\frac{d \log\ _1(a_1)}{d}(r_1 + r_2 + V(s_3) \quad V(s_1));$$
$$\vdots;$$

the use of TD( ) estimators, of empirical returns

$$\frac{@}{@\ _p}\frac{d \log\ _1(a_1)}{d}(R \quad V(s_1)):$$

These methods are all viable and used in practice.

Another more unusual option is to use a parameter gradient-critic $g(\ )$ which will 'average out' the entire environment and policy. Using a parameter gradient-critic leads to the update $\frac{\partial\theta}{\partial\theta_p}g(\ )$. The gradient critic $g(\ )$ can be learned in two ways. First, by learning a critic $Q(\ )$ which approximates the return R, and take the derivative with respect to , i. e. $g(\ ) = \frac{dQ(\theta)}{d\theta}$. Second, by directly learning a gradient critic by regressing against a valid gradient update, for instance

$$g(\ ) = \mathsf{E}\left[\frac{d \log\ _1(a_1)}{d}(R \quad V(s_1))\right] \quad ;$$

$$\text{or} \quad g(\ ) = \mathsf{E}\left[\frac{d \log\ _1(a_1)}{d}(r_1 + V(s_2) \quad V(s_1))\right] \quad ;$$

the latter of which learns a gradient-critic from a value critic. But no method which tries to approximates the entire actor-environment loop by a gradient $g(\ )$ estimated solely from the parameter is likely to work in practice.

Reparametrizing actions.

The next estimator involves reparametrizing the parameters, actions, but not the environment. In this situation, only one value critic is possible, the critic $Q(s_1; a_1)$ which averages the entire future. The corresponding estimator is the DDPG/SVG(0) estimator, which is

$$\frac{@}{@\ _p}\frac{@\ _1(a_1)}{@}\frac{@a_1}{@\ _1}\frac{dQ(s_1; a_1)}{da_1}.$$

The term $\frac{dQ(s_1,a_1)}{da_1}$ can be replaced by a gradient critic as we see next.

Full reparametrization: differential dynamic programming and trajectory optimization.

Finally, when reparametrizing the environment as well, we open the door to trajectory optimization type estimators. The general form is given by

$$\frac{@}{@\ _p}\frac{@\ _1(a_1)}{@}\frac{@a_1}{@\ _1}g(s_1; a_1; \ );$$

where $g(s_1; a_1; \ )$ is some form of a gradient-critic for the future given $(s_1; a_1)$. The most classical involves using sampled environment gradients, and corresponds to the following expression, also known as SVG(1):

$$\frac{@s_2}{@a_1}(\frac{@r_2}{@s_2} + \frac{@s_3}{@s_2}(\frac{@r_3}{@s_3} + \frac{@s_4}{@s_3}\frac{@r_4}{@s_4})):$$

But by using partially averaged gradient-critic, for instance a one-step gradient critic:

$$\frac{@s_2}{@a_1}(\frac{@r_2}{@s_2} + \frac{@s_3}{@s_2}g(s_3; a_3));$$

where $g(s_3; a_3)$ either deterministically approximates the stochastic gradient $(\frac{\partial r_3}{\partial s_3} + \frac{\partial s_4}{\partial s_3}\frac{\partial r_4}{\partial s_4}))$, or is the gradient of $Q(s_3; a_3)$. The most aggressive averaging would involve a gradient-critic $g(s_1; a_1)$ which averages the entire future, and which can be computed either by differentiating $Q(s_1; a_1)$, or by regression directly against a valid gradient target.

Action parameter critics.

Finally, let us consider a final, slightly more unusual gradient-critic estimator, explored in (Wierstra & Schmidhuber, 2007), related to the DDP/SVG(0) operators, but where the critic is not conditioned on $(s_1; a_1)$ but on $(s_1; \ _1)$ instead, resulting in the following:

$$\frac{@}{@\ _p}\frac{@\ _1(a_1)}{@}g(s_1; \ _1):$$

The gradient-critic $g(s_1; \ _1)$ can be estimated in many ways. First, by regressing against a correct gradient estimate, for instance a quantity like $\frac{\partial a_1}{\partial \pi_1}\frac{dQ(s_1,a_1)}{da_1}$; this is only possible if a_1 can be differentiable reparametrized as a function of $\ _1$.

Second, as a gradient $\frac{dV}{d\pi_1}$ of the value $V(s_1; \ _1)$ (note again this depends on the parameters $\ _1$ and not the sample a_1); when using function approximators, this requires (during training at least) $\ _1$ to be a stochastic function of s_1 (for the same reasons as highlighted in section 4.3).

Third, it can regressed against an estimate of the gradient where a_1 is not reparametrized, for instance $\frac{d \log \pi_1(a_1)}{d\pi_1}(R \quad V(s_1))$ or $\frac{d \log \pi_1(a_1)}{d\pi_1}(Q(s_1; a_1) \quad V(s_1))$.

What is peculiar and particularly interesting about the second and third option is that they apply to non-differentiable actions, and that the gradient critic implicitly sums over all actions. This results in a potentially significantly lower variance gradient estimator for non-differentiable actions; note this does not require any relaxation to the discrete sampling. This could for instance be applied in inference and learning in discrete generative models, for instance as an alternative to the high variance estimators score function estimators.
In this entire section, any bias introduced by the use of a critic or gradient-critic can be removed by using combined operators, as detailed in Appendix G.3.

Here, we don't discuss in depth the different options for bootstrap targets or linear combinations of critic, gradient critics and value functions, and yet, by making different choices on what quantities to condition on and which to average over, investigated a rich number of options available to us even in this very simple graph.

G Discussions

G.1 Optimal baseline and value function baseline

The estimator of theorem 2 is unbiased regardless of the choice of the baseline $B(B)$, which only affects the variance. A well-chosen baseline will reduce the variance of the gradient estimator. It is in general difficult to estimate or minimize the variance of the full estimator directly, however, we can derive a baseline which minimizes the variance of each q_v.

Definition 10. *We say that a baseline set B is* congruent *with a critic set C if it subset of C.*

Theorem 10. *Let $s(v; \ )$ be the score function $\frac{d}{d\theta}\log p_v$, and consider a fixed critic set C and congruent baseline set B. The following $B^*(B)$ is the baseline for B which minimizes the variance of G_v:*

$$B^*(B) = \frac{E_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}} \ s(v; \ )^2 Q(C)}{E_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}}[s(v; \ )^2]} = \frac{E_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}} \ s(v; \ )^2 L(v)}{E_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}}[s(v; \ )^2]}:$$

Furthermore, for two congruent sets B_1 B_2*, the variance of* G_v *using* $B^*(B_1)$ *is larger or equal than when using* $B^*(B_2)$.

This form of the optimal baseline is folklore, and can be found (in the context of RL) in (Greensmith et al., 2004) and (Ranganath et al., 2013) (in the context of VI).

G.2 Choosing conditioning sets of value functions and critics

From our main theorem 2, we know that valid critic and baseline sets together define a new estimate for the gradient of a stochastic computation graph. But how should one choose the critic and baseline sets? We discuss here the tradeoffs made when choosing different sets.

Given a critic with conditioning set C, how to choose an appropriate set B for the value?

For congruent optimal baselines, theorem 10 states that variance is always reduced by increasing the conditioning set B. For the optimal baseline, this implies it is optimal to choose B to be the intersection of C and the set of non-descendants of v. A related statement happens to be true for value baselines:

Lemma 4. *Consider two congruent baseline sets* B_1 B_2 C*. Then the variance of the advantage using* B_1 *is higher than that using* B_2*:*

$$\mathbb{E}\left[\left(Q(C) \quad V(B_2)\right)^2\right] \quad \mathbb{E}\left[\left(Q(C) \quad V(B_1)\right)^2\right]$$

Is there any point in using non-congruent value functions (i.e. using a conditioning set B which is not a subset of the critic set C)?

The previous lemma suggests, but does not prove, that value functions should also be conditioned on the maximal congruent set B^*, intersection of C with the non-descendants of v. One may wonder the usefulness of non-congruent value functions. Intuitively, variables in $B \cap C$ have already been averaged in the critic, so it would be tempting to disregard non-congruent baselines. The answer is here more complicated; non-congruent baselines may increasing or decrease variance, depending on correlations between value and critics, as we see in the following two examples.

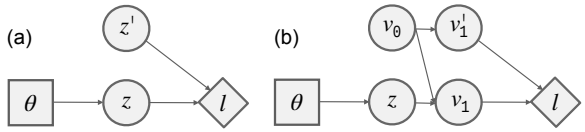

Figure 17: *Constructing value sets.* In (a), non-congruent baselines increase variance, in (b), they decrease it.

Example 1. *Consider the example of figure 17(a), where we consider two variables* z $p_\theta(z); z'$ $N(0; \ )$*, a loss function* $\ell(z; z') = \ell(z) + z'$*. Assume that the variance of* z' *is large, that all variables are observed. Because* z' *has large variance, it tends to dominate the loss, even if it is not controllable by changing . The gradient estimate for* $\frac{d}{d\theta}\mathbb{E}_{z,z^0}[\ell(z; z')]$ *is given, per theorem 2, by* $\frac{d}{d\theta}\log p_\theta(z)(Q(C) \quad V_z(B))$.

The 'naive' estimate consists in using the empirical loss $(\ell(z) + z')$ *as critic (which is equivalent to choosing* $C = \{z; z'\}$*) and a baseline equal to* $\mathbb{E}[\ell(z)]$ *(obtained by choosing the value set* $B = ?$*); the resulting advantage* $\ell(z) \quad \mathbb{E}[\ell(z)] + z'$ *has high variance due to* z'*. A lower variance estimate can be obtained by choosing either* $C = \{z\}$ *and maximally congruent* $B = ?$*, or* $C = \{z; z'\}$ *and maximally congruent* $B = \{z'\}$*; both lead to the same low-variance estimate* $\ell(z) \quad \mathbb{E}[\ell(z)]$ *of the advantage.*

However, if we use $C = \{z\}$ *and non-congruent* $B = \{z'\}$*, we obtain the high variance estimate again.*

Example 2. *Consider next the example in figure 17(b), with* z $p_\theta(z)$*,* v_0 $N(0; \ )$*;* v_1 *and* v_1' *are conditionally independent with respective distributions* $N(\ell(z) + v_0; \ ')$ *and* $N(v_0; \ ')$*, with* $' << .$ *Suppose that* $\ell(v_1; v_1') = v_1 + v_1'$ *and that all variables but* v_0 *are observed. Taking for instance critic set* $C = \{z; v_1\}$ *and congruent set* $B = ?$*, we have* $Q(C) \quad V(B) = (\ell(z) \quad \mathbb{E}[\ell(z)]) + 2(v_1 \quad \ell(z))$*, which is high variance because* $(v_1 \quad \ell(z))$ *has variance at least as high as* v_0.

Note however that if we choose the non-congruent set $B = \{v_1'\}$*, then* $V(B) = \mathbb{E}[\ell(z)] + 2v_1'$*, so that* $Q \quad V = (\ell(z) \quad \mathbb{E}[\ell(z)]) + 2(v_1 \quad \ell(z) \quad v_1')$*, which is much lower variance since* $(v_1 \quad \ell(z) \quad v_1')$ *only has variance* $'$.

Having looking at the conditioning set of the value function, we look at the following question: how should we construct a critic conditioning set C?

A Rao-Blackwellization argument suggests that marginalizing out as many variables as possible leads to the lowest variance. However, this is assuming exact conditional expectations. However, in practice those are not easily computed and we will resort to function approximation (see section 3.5). When using function approximations to our estimates of value and functions, there will be a bias-variance trade-off between using approximate critic which marginalizes out random variable (which leads to bias) or using model samples of the losses (which leads to variance). Furthermore, the re-usability of a particular value function for multiple purposes (e.g. to serve as part of critics for multiple nodes, and / or as bootstrap target(s), see below) can motivate more limited marginalization).

G.3 Critic correction when using gradient-critics

A common pattern is to use critic to approximate a model that has no gradient, or whose gradients are unknown. In the case we use a Markovian set for the critic, we can learn the critic from predicted losses only (i.e. set to 0 in equation 5), but then use the corresponding gradient as a gradient-critic in the reparameterized graph. The use of the critic or gradient critic incurs a bias in the estimator. Here we see a technique which combines gradient-critics and critics in order to build an unbiased estimator which still leverages the gradient-critic (this technique can in fact be used anytime a gradient-critic is used) Consider a stochastic node v which can be reparametrized, let $L(v)$ be the sum of all downstream losses from v. Consider a Markovian critic set C_v for v and let $Q(C_v)$ be the corresponding value critic, and $\hat{Q}(C_v)$ an approximation of it. Let $_v$ be an arbitrary parent of v. From theorem 7, $\frac{dQ(C_v)}{dv}$ is a valid gradient-critic for v, and we have

$$\frac{d\mathrm{E}[L(v)]}{d\ _v} = \mathrm{E}\left[\frac{d\log p(v)}{d\ _v}L(v)\right] = \mathrm{E}\left[\frac{d\log p(v)}{d\ _v}Q(C_v)\right] = \mathrm{E}\left[\frac{dQ(C_v)}{dv}\frac{dv}{d\ _v}\right] \quad \mathrm{E}\left[\frac{d\log p(v)}{d\ _v}\hat{Q}(C_v)\right]$$

The second equality is the regular score function estimator; the second, the score function estimator using critics; the third, a reparametrized estimator using gradient-critics. The last equality is the same as the second, but taking into account the bias induced by the use of an approximate critic.

By writing the loss $L(v)$ as the sum of two terms $(L(v) \quad Q(C_v)) + Q(C_v)$, we can use the score function estimator on the first term, reparameterize the second term, and therefore potentially leverage the lower variance of the reparameterized gradient while keeping unbiasedness thanks to the score function term. The resulting estimator will take the following form:

$$\frac{d\mathrm{E}[L(v)]}{d\ _v} = \mathrm{E}\left[\frac{d\log p(v)}{d\ _v}\quad L(v)\quad \hat{Q}(C_v)\quad + \frac{d\hat{Q}(C_v)}{dv}\frac{dv}{d\ _v}\right] :$$

This estimator is strongly related to Stein variational estimator, see e.g. (Liu & Wang, 2016). Note if the critic is exact, i.e. $Q = \hat{Q}$, the first term has zero expectation and may be excluded. A common use pattern is when v is an action a (in RL) or sample z (in generative models), the critic set is $(s; a)$, where s is the state (in RL) or context/previous state (in generative models). This technique sometimes called 'action-conditional baselines' in the literature, see for instance (Tucker et al., 2017; Gu et al., 2015; Grathwohl et al., 2017). Is it however not clear that the bias correction does not increase the variance in such a way that the gradients from reparameterization become dominated by the noise of the score function (Tucker et al., 2018). Interestingly, the action-conditional baseline literature takes an almost opposed interpretation to ours: while we see the $(L(v) \quad \hat{Q}(C))$ as a correction to the biased estimator resulting using an approximate critic $\hat{Q}(C)$, (Tucker et al., 2017; Grathwohl et al., 2017) view instead the critic as an 'invalid' baseline (as in, it biases the gradient of the score function estimator), which needs to be corrected with the reparametrized term $\frac{d\hat{Q}(C_v)}{dv}\frac{dv}{d\theta_v}$. In our view, this interpretation suffers from not having a natural way for trading off bias and variance, for instance weighting the correction term $L(v) \quad \hat{Q}(C_v)\ )$ with a coefficient less than 1 or leaving it out entirely.

H Proofs

For any node v, let $\overline{G}_v$ be the set of non-descendants of v.

H.1 Computation lemmas

We provide a couple of useful lemmas describing properties of the computation resulting from a stochastic computation graph.

Our first lemma describes a set which deterministically computes ` given information in a set $C$, where $C$ is 'as far away' from ` as possible.

Lemma 5. *Consider a cost ` and arbitrary set $C$ for which ` descends from C. Let a sequence of set V_i be described as follows:$V_0 = f`g$, and for any i 1, we let V_i be defined as follows:*

Any stochastic or input node in V_{i-1} is included in V_i.

Any node in $C \setminus V_{i-1}$ is included in V_i.

The parents of any deterministic node in $V_{i-1} \cap C$ are included in V_i.

Then, for any i, ` can be deterministically computed from V_i. Furthermore, the set converges to a set V which only contains stochastic nodes, input nodes, and nodes in C.

Proof. The first part of the proof is a simple recursion. ` is clearly a deterministic function of `. Assume now that ` is a deterministic function of $V_i$, and consider the set $V_{i+1}$. For any variable $v$ in $V_i$, it is either in $V_{i+1}$, or a deterministic function of its parents, which are included in $V_{i+1}$. It follows that ` is a deterministic function of V_{i+1}. For proving convergence, the set of nodes which are are in V_i and either in C, stochastic, or input nodes is non-decreasing (since nodes of that type stay in V_i if they are in V_{i-1}). As for deterministic nodes, they can only be in V_i if they are in C or if they are in a directed path of distance i to `. This means that for $j$ greater than the the longest directed path between a node and `, V_j can only contain deterministic nodes if they are in C (which proves the second part of the lemma). Together, these statements imply that for j' j, the set $V_{j'}$ is non-increasing, and it is bounded, therefore it converges. □

We provide a pictorial representation of the algorithm for a given graph as follows:

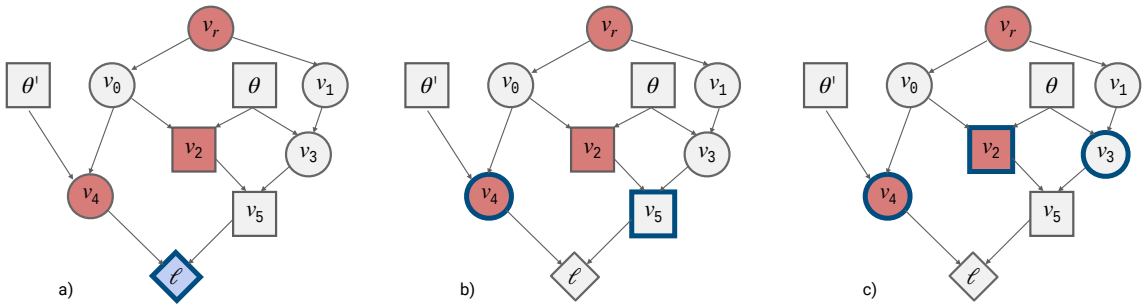

Figure 18: Illustration of the algorithm described in Lemma 5. The set C is indicated by nodes with dark red background, the sets V_i by nodes with blue frame. The algorithm finishes at the third step, since the resulting set is stable. ` can indeed be deterministically computed from $v_2; v_3$ and v_4.

When C is Markovian, the set V has an important property which we will later use.

Property 3. *Let $W(V)$ be the set of stochastic ancestors of nodes in V, unblocked by C. We have:*

$$P(W|C) = \prod_{w \in cW} p(w|h_w(C)) \tag{32}$$

where the notation $h_w(C)$ is simply to indicate that the simple forward computation of h_w depends on the values of variables in C which are ancestors of variables in W.

In other words, to sample all the variables in W, we can simply use the forward model; the conditioning set C only plays a role in setting the value of ancestors for some of the variables in that set.

In the example above, the set W is $fv_3; v_1g$; this is because v_3 is in V (but not in C) and v_1 is an ancestor of v_3, unblocked by C. The node v_r is *not* included in W because it is in C. Conditional on C, the distribution of W is given by $p(v_1|v_r)p(v_3|v_1)$.

Proof of Prop. 3. By the definition of Markovianity, nodes in W cannot have a descendent in C. The property follows from applying Bayes' rule. Using a graph-compatible ordering of variables in w_i, we have: $P(W|C) = \prod_i p(w_i|C; w_{<i}) = \prod_i \frac{p(w_i, C, w_{<i})}{p(C, w_{<i})}$. At this point we simply write numerator and denominators as product of probabilities of variables conditioned on parents; because none of the variables in C descend from a variable in W, all terms cancel out except for $p(w_i|h_{w_i})$, which gives us the desired result. □

Our second lemma relates conditional independence between variables to conditional independence between derivatives.

Lemma 6. *Assume that* $\log p_v$ *is conditionally independent of cost* ℓ *given* C. *Then* $\frac{d \log p_v}{d\theta}$ *is conditionally independent of* ℓ *given* C.

Proof. By forward mode differentiation, the stochastic computation which describes the computation of C, ℓ and $\log p_v$ can also describe the computation of $C; \ell$ and $(\log p_v; \frac{d \log p_v}{d\theta})$. If the graph properties imply conditional independence of $\log p_v$ and ℓ given C, they therefore also imply conditional independence of ℓ and $\frac{d \log p_v}{d\theta}$ given C. Note this is only true because we only consider graph-induced conditional independence properties. □

For similar reasons, we have:

Lemma 7. *Let* $u; v; w$ *be nodes in a graph, and let* C *be an arbitrary set.* $\frac{dv}{du}$ *and* $\frac{dw}{dv}$ *are conditionally independent given* C *if* $fu' : u'$ *is in a path between* u *and* vg *and* $fv' : v'$ *is in a path between* v *and* wg *are conditionally independent given* C.

H.2 Proofs of results for value-based methods

Proof of Theorem 2. We prove theorem 2 by proving that for any node v in S, $\mathbb{E}\left[\frac{d}{d\theta} \log p_v L\right] = \mathbb{E}[q_v]$. To do so, we follow three steps: 1) show that L can be replaced by $L(v)$, 2) show that any baseline B can be subtracted from $L(v)$ (and hence, a value baseline can be subtracted), and 3) show that $L(v)$ can be replaced by a critic $Q(C)$.

For any node v in S, consider any cost ℓ non-descendant from v, i $\ell \perp \overline{G}_v$. the contribution of v to the gradient estimate with respect to ℓ is $\mathbb{E}\left[\frac{d}{d\theta} \log p_v \ell\right]$. Using $G = \overline{G}_v \cup (G \cap \overline{G}_v)$ and the law of iterated expectations, we obtain:

$$\begin{aligned}\mathbb{E}\left[\frac{d}{d} \log p_v \ell\right] &= \mathbb{E}_{\overline{G}_v}\left[\mathbb{E}_{\mathcal{G}\setminus\overline{G}_v|\overline{G}_v}\left[\frac{d}{d} \log p_v \ell\right]\right] \\ &= \mathbb{E}_{\overline{G}_v}\left[\ell \, \mathbb{E}_{\mathcal{G}\setminus\overline{G}_v|\overline{G}_v}\left[\frac{d}{d} \log p_v\right]\right] = 0\end{aligned}$$

The second equality comes from the fact that this particular ℓ is a deterministic function of $\overline{G}_v$ (since it is a non-descendant of v). The third equality comes from the following:

$$\begin{aligned}E_{\mathcal{G}\setminus\overline{G}_v|\overline{G}_v}\left[\frac{d}{d} \log p_v\right] &= \int_v p(v|h_v) \frac{d}{d} \log p(v|h_v) \\ &= \int_v p(v|h_v) \frac{\frac{d}{d\theta} p(v|h_v)}{p(v|h_v)} \\ &= \int_v \frac{d}{d} p(v|h_v) = \frac{d}{d} \int_v p(v|h_v) = \frac{d}{d} 1 = 0\end{aligned}$$

It is important to note that conditional on the non-descendants $\overline{G}_v$ (which includes the parents h_v of v, but no descendants) the distribution of v is simply $p(v|h_v)$. Using the same idea, we show that subtracting a baseline does not bias the expectation:

$$\mathbb{E}\left[\frac{d}{d} \log p_v V(B)\right] = \mathbb{E}_{\overline{G}_v}\left[\mathbb{E}_{\mathcal{G}\setminus\overline{G}_v|\overline{G}_v}\left[\frac{d}{d} \log p_v B(B)\right]\right] = \mathbb{E}_{\overline{G}_v}\left[B(B) \mathbb{E}_{\mathcal{G}\setminus\overline{G}_v|\overline{G}_v}\left[\frac{d}{d} \log p_v\right]\right] = 0$$

Finally,

$$\begin{aligned}
\mathsf{E}\left[\frac{d}{d}\log p_v L(v)\right] &= \mathsf{E}_{\mathcal{C}_v}\left[\mathsf{E}_{\mathcal{G}\setminus\mathcal{C}_v|\mathcal{C}_v}\left[\frac{d}{d}\log p_v \quad L(v)\right]\right] \\
&= \mathsf{E}_{\mathcal{C}_v}\left[\mathsf{E}\left[\frac{d}{d}\log p_v | C\right] \mathsf{E}[L(v)|C]\right] \\
&= \mathsf{E}_{\mathcal{C}_v}\left[\mathsf{E}\left[\frac{d}{d}\log p_v | C\right] Q(C)\right] \\
&= \mathsf{E}_{\mathcal{C}_v}\left[\mathsf{E}\left[Q(C)\frac{d}{d}\log p_v | C\right]\right] \\
&= \mathsf{E}\left[Q(C)\frac{d}{d}\log p_v\right]
\end{aligned}$$

where the second line comes from the conditional independence of $L(v)$ and $\frac{d}{d\theta}\log p_v$ given C (following the conditional independence of $L(v)$ and $\log p_v$, and applying lemma 6), the third line follows from the definition of the critic, and the fourth from the fact that the critic is constant as a function of C.

□

Proof of Theorem 10. The variance of q_v is $\mathsf{Var}(q_v) = \mathsf{E}[q_v^2] - \mathsf{E}[q_v]^2$. Since the later term is unaffected by the choice of the baseline function, minimizing the variance is equivalent to minimizing $\mathsf{E}[q_v^2]$.

$$\begin{aligned}
\mathsf{E}[q_v^2] &= \mathsf{E}\left[s_\theta(v;h_v)^2 \left(Q(v;C) - V(B)\right)^2\right] \\
&= \sum_b p(B=b)\left[\mathsf{E}_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}=b}\left[s_\theta^2 \left(Q(v;C) - B(b)\right)^2\right]\right]
\end{aligned}$$

The expression above is a sum of non-negative expressions, each involving a distinct $B(b)$ to optimize. The sum is jointly minimized if each is minimized. For a given b, we take the gradient of $\left[\mathsf{E}_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}=b}\left[s_\theta^2 \left(Q(v;C) - B(b)\right)^2\right]\right]$ with respect to $B(b)$ and set it to zero; we find:

$$B^*(B) = \frac{\mathsf{E}_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}}\left[s_\theta^2 Q(v;C)\right]}{\mathsf{E}_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}}\left[s_\theta^2\right]}$$

To prove the second equality, simply note that $B \perp C$ (since the baseline is congruent); it follows that:

$$\begin{aligned}
\left[\mathsf{E}_{\mathcal{G}\setminus\mathcal{B}|\mathcal{B}}\left[s_\theta^2 L(v)\right]\right. &= \mathsf{E}_{\mathcal{C}\setminus\mathcal{B}|\mathcal{B}}\left[\mathsf{E}_{\mathcal{G}\setminus\mathcal{C}|\mathcal{C}} s_\theta^2 L(v)\right] \\
&= \mathsf{E}_{\mathcal{C}\setminus\mathcal{B}|\mathcal{B}}\left[s_\theta^2 Q(v;C)\right]
\end{aligned}$$

Next, we consider two conditioning sets $B_1 \subset B_2$, with respective optimal baselines B_1^* and B_2^* (as defined above). For an assignment b of variables in B_2, let $b_{|1}$ be the restriction of b to the variables found in B_1. We can construct a baseline B with set B_2 by choosing $V(b) = V_1^*(b_{|1})$; this is a valid baseline for set B_2, but is strictly equivalent to using the optimal baseline B_1^* for set B_1. By optimality of $B^*(B_2)$, the variance of q_v using $B^*(B_1)$ is higher than using $B^*(B_2)$. □

Proof of lemma 2. Consider $X_v^1 \subset X_v^2$. By definition of the conditional expectation, we have $V(X_v^1) = \mathsf{E}[L(v)|X_v^1]$. By the law of iterated expectation, $\mathsf{E}[L(v)|X_v^1] = \mathsf{E}[\mathsf{E}[L(v)|X_v^2]|X_v^1] = \mathsf{E}[V(X_v^2)|X_v^1]$. □

Proof of property 2. Applying property 3, we can see that the conditional distribution of W given X_v and $X_v^\uparrow$ is the same (direct stochastic ancestors to a variable in W cannot be in $X_v^\uparrow \cap X_v$), therefore the conditional expectations are the same. By law of iterated expectations, for $X_v \subset X_v' \subset X_v^\uparrow$, $V(X_v') = \mathsf{E}[\mathsf{E}[L(v)|X_v^\uparrow]|X_v'] = \mathsf{E}[V(X_v^\uparrow)|X_v'] = \mathsf{E}[V(X_v)|X_v'] = V(X_v)$. □

Proof of lemma 3. Since $X^1 \subset X^{2\uparrow}$, by Lemma 2, we have $V(X^1) = E[V(X^{2\uparrow})|X^1]$. But since X^2 is Markov, $V(X^{2\uparrow}) = V(X^2)$ and the lemma follows. □

Proof of theorem 3. By the decomposition assumption, we have:

$$L(v) = \sum_i L(V_i)$$

Taking expectations conditional on X, we obtain

$$V(X) = \mathrm{E}[L(v)|X] = \sum_i \mathrm{E}[L_{v_i}|X]$$

For each i, $V(X_{V_i}) = \mathrm{E}[L(V_i)|X_{V_i}]$; by lemma 3, $\mathrm{E}[L(V_i)|X] = \mathrm{E}[V(X_{V_i}|X]$. □

Proof of Lemma 4. This follows from classical manipulation regarding conditional expectation:

$$\begin{aligned}\mathrm{E}\left[(Q(C) - V(B_2))^2|B_2\right] &= \mathrm{E}\left[(Q(C) - V(B_1)) + (V(B_1) - V(B_2))\right]^2 B_2 \\ &= \mathrm{E}\Big[(Q(C) - V(B_1))^2 + (V(B_1) - V(B_2))^2 \\ &\quad + 2(Q(C) - V(B_1))(V(B_1) - V(B_2))\Big] B_2\end{aligned}$$

The last term can be simplified:

$$\begin{aligned}\mathrm{E}\left[(Q(C) - V(B_1))(V(B_1) - V(B_2))\right] B_2 &= (V(B_1) - V(B_2))\mathrm{E}\left[Q(C) - V(B_1)\right] B_2 \\ &= (V(B_1) - V(B_2))^2\end{aligned}$$

The first equality is due to the fact that since $B_1 \perp B_2$, conditioned on B_2, both values are constant; the second equality due to $\mathrm{E}[Q(C)|B_2] = V(B_2)$ due to $B_2 \perp C$ and the law of iterated expectations. We obtain $\mathrm{E}\left[(Q(C) - V(B_2))^2|B_2\right] = \mathrm{E}\left[(Q(C) - V(B_1))^2 - (V(B_1) - V(B_2))^2\right] B_2$, which from applying the law of iterated expectation again becomes

$$\mathrm{E}\left[(Q(C) - V(B_2))^2\right] = \mathrm{E}\left[(Q(C) - V(B_1))^2 - (V(B_1) - V(B_2))^2\right] - \mathrm{E}\left[(Q(C) - V(B_1))^2\right]$$

□

H.3 Proofs of results for gradient-based methods

Proof of theorem 4. Using the iterated law of expectations once again:

$$\begin{aligned}\mathrm{E}_{\mathcal{G}}\left[\sum_i \frac{dL^s}{dv_i}\frac{@v_i}{@v}\right] &= \sum_i \mathrm{E}_{\mathcal{C}_{v_i}}\left[\mathrm{E}_{\mathcal{G}|\mathcal{C}_{v_i}}\left[\frac{dL^s}{dv_i}\frac{@v_i}{@v}\right]\right] \\ &= \sum_i \mathrm{E}_{\mathcal{C}_{v_i}}\left[\mathrm{E}_{\mathcal{G}|\mathcal{C}_{v_i}}\left[\frac{dL^s}{dv_i}\right]\mathrm{E}_{\mathcal{G}|\mathcal{C}_{v_i}}\left[\frac{@v_i}{@v}\right]\right] = \sum_i \mathrm{E}_{\mathcal{C}_{v_i}}\left[g_{v_i}\,\mathrm{E}_{\mathcal{G}|\mathcal{C}_{v_i}}\left[\frac{@v_i}{@v}\right]\right] \\ &= \sum_i \mathrm{E}_{\mathcal{C}_{v_i}}\left[\mathrm{E}_{\mathcal{G}|\mathcal{C}_{v_i}}\left[g_{v_i}\frac{@v_i}{@v}\right]\right] = \sum_i \mathrm{E}\left[g_{v_i}\frac{@v_i}{@v}\right]\end{aligned}$$

The third equality comes the conditional independence assumption, the fourth from the definition of the gradient-critic, and the fourth from the fact that the gradient-critic is a deterministic function of $\mathcal{C}_{v_i}$. □

Proof of theorem 7. From the Markov assumption, following Lemma 5 and property 3, we can write $L(v)$ as a deterministic function of a set $V \ni v$ with stochastic ancestors W with $P(W|C) = \prod_{w\in\mathcal{W}} p(w|h_w(C))$.

We have $Q(C) = \int_{\mathbf{w}\in\mathcal{W}} \prod_{w\in\mathcal{W}} p(w|h_w) L(v)(V)$. Taking the gradient with respect to v, we obtain:

$$\begin{aligned}
\frac{dQ(C)}{dv} &= \int_{\mathbf{w}\in\mathcal{W}} \left(\prod_{w\in\mathcal{W}} p(w|h_w)\right) \left(\frac{dL(v)}{dv} + \sum_{w\in\mathcal{W}} \frac{d\log p(w|h_w)}{dv} L(v)\right) \\
&= \int_{\mathbf{w}\in\mathcal{W}} \left(\prod_{w\in\mathcal{W}} p(w|h_w)\right) \left(\frac{d}{dv} \quad L(v) + \sum_{w\in\mathcal{A}^S} \log p(w|h_w) L(v)\right) \\
&= \int_{\mathbf{w}\in\mathcal{W}} \left(\prod_{w\in\mathcal{W}} p(w|h_w)\right) \left(\frac{d}{dv} L_v^s = \mathbb{E}\left[\frac{d}{dv} L_v^s | C\right] = g_v(C)\right)
\end{aligned}$$

□