
Hindsight Credit Assignment

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Nicolas Heess, Mohammad G. Azar,
Bilal Piot, Hado van Hasselt, Satinder Singh, Greg Wayne, Doina Precup, Rémi Munos
DeepMind
{harutyunyan, wdabney, munos}@google.com

Abstract

We consider the problem of efficient credit assignment in reinforcement learning. In order to efficiently and meaningfully utilize new data, we propose to explicitly assign credit to past decisions based on the likelihood of them having led to the observed outcome. This approach uses new information in hindsight, rather than employing foresight. Somewhat surprisingly, we show that value functions can be rewritten through this lens, yielding a new family of algorithms. We study the properties of these algorithms, and empirically show that they successfully address important credit assignment challenges, through a set of illustrative tasks.

1 Introduction

A reinforcement learning (RL) agent is tasked with two fundamental, interdependent problems: exploration (how to discover useful data), and credit assignment (how to incorporate it). In this work, we take a careful look at the problem of credit assignment. The instrumental learning object in RL – the value function – quantifies the following question: “*how does choosing an action a in a state x affect future return?*”. This is a challenging question for several reasons.

Issue 1: Variance. The simplest way of estimating the value function is by averaging returns (future discounted sums of rewards) starting from taking a in x . This Monte Carlo style of estimation is inefficient, since there can be a lot of randomness in trajectories.

Issue 2: Partial observability. To amortize the search and reduce variance, temporal difference (TD) methods, like Sarsa and Q-learning, use a learned approximation of the value function and bootstrap. This introduces bias due to the approximation, as well as a reliance on the Markov assumption, which is especially problematic when the agent operates outside of a Markov Decision Process (MDP), for example if the state is partially observed, or if there is function approximation. Bootstrapping may then cause the value function to not converge at all, or to remain permanently biased [18].

Issue 3: Time as a proxy. $TD(\lambda)$ methods control this bias-variance trade-off, but they rely on *time* as the sole metric for relevance: the more recent the action, the more credit or blame it receives from a future reward [19, 20]. Although time is a reasonable proxy for cause-and-effect (especially in MDPs), in general it is a heuristic, and can hence be improved by learning.

Issue 4: No counterfactuals. The only data used for estimating an action’s value are trajectories that contain that action, while ideally we would like to be able to use the same trajectory to update *all* relevant actions, not just the ones that happened to (serendipitously) occur.

Figure 1 illustrates these issues concretely. At the high-level, we wish to achieve credit assignment mechanisms that are both sample-efficient (issues 1 and 4), and expressive (issues 2 and 3). To this end, we propose to reverse the key learning question, and learn estimators that measure: “*given the future outcome (reward or state), how relevant was the choice of a in x to achieve it?*”, which is essentially the credit assignment question itself. Although eligibility traces consider the same

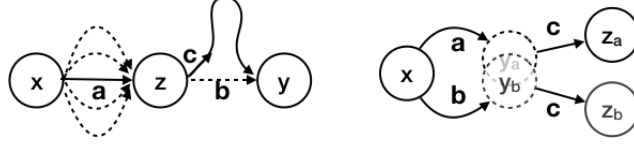


Figure 1: **Left.** Consider the trajectory shown by solid arrows to be the sampled trajectory, τ . An RL algorithm will typically assign credit for the reward obtained in state y to the actions along τ . This is unsatisfying for two reasons: (1) action a was not essential in reaching state z , any other a' would have been just as effective; hence, overemphasizing a is a source of variance; (2) from z , action c was sampled, leading to a multi-step trajectory into y , but action b transitions to y from z directly; so, it should get more of the credit for y . Note that c could have been an exploratory action, but also could have been more likely according to the policy in z , but *given that y was reached*, b was more likely **Right.** The choice between actions a or b at state x causes a transition to either y_a or y_b , but they are perceptually aliased. On the next decision, the same action c transitions the agent to different states, depending on the true underlying y . The state y can be a single state, or could itself be a trajectory. This scenario can happen e.g. when the features are being learned. A TD algorithm that bootstraps in y will not be able to learn the correct values of a and b , since it will average over the rewards of z_a and z_b . When y is a potentially long trajectory with a noisy reward, a Monte Carlo algorithm will incorporate the noise along y into the values of both a and b , despite it being irrelevant to the choice between them. We would like to be able to directly determine the relevance of a to being in z_a .

question, they do so in a way that is (purposefully) equivalent to the forward view [19], and so they have to rely mainly on “vanilla” features, like time, to decide credit assignment. Reasoning in the backward view explicitly opens up a new family of algorithms. Specifically, we propose to use a form of *hindsight conditioning* to determine the relevance of a past action to a particular outcome. We show that the usual value functions can be rewritten in hindsight, yielding a new family of estimators, and derive policy gradient algorithms that use these estimators. We demonstrate empirically the ability of these algorithms to address the highlighted issues through a set of diagnostic tasks, which are not handled well by other means.

2 Background and Notation

A Markov decision process (MDP) [13] is a tuple $(\mathcal{X}, \mathcal{A}, p, r, \gamma)$, with $\mathcal{X}$ being the state space, $\mathcal{A}$ – the action space, $p : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \rightarrow [0, 1]$ – the state-transition distribution (with $p(y|x, a)$ denoting the probability of transitioning to state y from x by choosing action a), $r : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ – the reward function, and $\gamma \in [0, 1]$ – the scalar discount factor. A stochastic policy π maps each state to a distribution over actions: $\pi(a|x)$ denotes the probability of choosing action a in state x . Let $\mathcal{T}(x, \pi)$ and $\mathcal{T}(x, a, \pi)$ be the distributions over trajectories $\tau = (X_k, A_k, R_k)_{k \in \mathbb{N}^+}$ generated by a policy π , given $X_0 = x$ and $(X_0, A_0) = (x, a)$, respectively. Let $Z(\tau) \stackrel{\text{def}}{=} \sum_{k \geq 0} \gamma^k R_k$ be the return obtained along the trajectory τ . The value (or V-) function V^π and the action-value (or Q-) function Q^π denote the expected return under the policy π given $X_0 = x$ and $(X_0, A_0) = (x, a)$, respectively:

$$V^\pi(x) \stackrel{\text{def}}{=} \mathbb{E}_{\tau \sim \mathcal{T}(x, \pi)} [Z(\tau)], \quad Q^\pi(x, a) \stackrel{\text{def}}{=} \mathbb{E}_{\tau \sim \mathcal{T}(x, a, \pi)} [Z(\tau)]. \quad (1)$$

The benefit of choosing a given action a over the usual policy π is measured by the advantage function $A^\pi(x, a) \stackrel{\text{def}}{=} Q^\pi(x, a) - V^\pi(x)$. Policy gradient algorithms improve the policy by changing π in the direction of the gradient of the value function [21]. This gradient at some initial state x_0 is

$$\nabla V^\pi(x_0) = \sum_{x, a} d^\pi(x|x_0) Q^\pi(x, a) \nabla \pi(a|x) = \mathbb{E}_{\tau \sim \mathcal{T}(x_0, \pi)} \left[\sum_a \sum_{k \geq 0} \gamma^k A^\pi(X_k, a) \nabla \pi(a|X_k) \right],$$

where $d^\pi(x|x_0) \stackrel{\text{def}}{=} \sum_k \gamma^k \mathbb{P}_{\tau \sim \mathcal{T}(x_0, \pi)}(X_k = x)$ is the (unnormalized) discounted state-visitation distribution. Practical algorithms such as REINFORCE [24] approximate Q^π or A^π with an n -step truncated return, possibly combined with a bootstrapped approximate value function V , which is also often used as baseline (see [21, 11]) along a trajectory $\tau = (X_k, A_k, R_k)_k \sim \mathcal{T}(x, a, \pi)$:

$$A^\pi(x, a) \approx \sum_{k=0}^{n-1} \gamma^k R_k + \gamma^n V(X_n) - V(x).$$

3 Conditioning on the Future

The classical value function attempts to answer the question: "how does the current action affect future outcomes?" By relying on predictions about these future outcomes, existing approaches often exacerbate problems around variance (issue 1) and partial observability (issue 2). Furthermore, these methods tend to use temporal distance as a proxy for relevance (issue 3) and are unable to assign credit counter-factually (issue 4). We propose to learn estimators that explicitly consider the credit assignment question: "*given an outcome, how relevant were past decisions?*", and try to answer it explicitly.

This approach can in fact be linked to some classical methods in statistical estimation. In particular, Monte Carlo simulation is known to be inaccurate when there are rare events that are of interest: the averaging requires an infeasible number of samples to obtain an accurate estimate [15]. One solution is to *change measures*, that is, to use another distribution for which the events are less rare, and correct with importance sampling. The Girsanov theorem is a well-known example of this in processes with Brownian dynamics [3], known to produce lower variance estimates.

This scenario of rare random events is particularly relevant to efficient credit assignment in RL. When a new significant outcome is experienced, the agent ought to quickly update its estimates and policy accordingly. Let $\tau \sim \mathcal{T}(x, \pi)$ be a sampled trajectory, and F some function of it. By changing measures from the policy π with which it was sampled to a future-conditional, or *hindsight* distribution $h(\cdot|x, \pi, F(\tau))$, we hope to improve the efficiency of credit assignment. The importance sampling ratio $\frac{h(a|x, \pi, F(\tau))}{\pi(a|x)}$ then precisely denotes the relevance of an action a to the specific future $F(\tau)$. If the distribution $h(a|x, \pi, F(\tau))$ is accurate, this allows us to quickly assign credit to all actions relevant to achieving $F(\tau)$. In this work, we consider F to be a future *state*, or a future *return*. To highlight the use of the future-conditional distribution, we refer to the resulting family of methods as Hindsight Credit Assignment (HCA).

The remainder of this section formalizes the insight outlined above, and derives the usual value functions in terms of the hindsight distributions, while the subsequent section presents novel policy gradient algorithms based on these estimators.

3.1 Conditioning on Future States

The agent composes its estimates of the return from an action a by summing over the rewards obtained from future states X_k . One option of hindsight conditioning is to consider, at each step, the likelihood of an action a *given that the future state X_k was reached*.

Definition 1 (*State-conditional hindsight distributions*). *For any action a and any state y , define $h_k(a|x, \pi, y)$ to be the conditional probability over trajectories $\tau \sim \mathcal{T}(x, \pi)$ of the first action A_0 of trajectory τ being equal to a , given that the state y has occurred at step k along trajectory τ :*

$$h_k(a|x, \pi, y) \stackrel{\text{def}}{=} \mathbb{P}_{\tau \sim \mathcal{T}(x, \pi)}(A_0 = a | X_k = y). \quad (2)$$

Intuitively, $h_k(a|x, \pi, y)$ quantifies the relevance of action a to the future state X_k . If a is not relevant to reaching X_k , this probability is simply the policy $\pi(a|x)$ (there is no relevant information in X_k). If a is instrumental to reaching X_k , $h_k(a|x, \pi, y) > \pi(a|x)$, and vice versa, if a detracts from reaching X_k , $h_k(a|x, \pi, y) < \pi(a|x)$. In general, h_k is a lower-entropy distribution than π . The relationship of h_k to more familiar quantities can be understood through the following identity obtained by an application of Bayes' rule:

$$\frac{h_k(a|x, \pi, y)}{\pi(a|x)} = \frac{\mathbb{P}(X_k = y | X_0 = x, A_0 = a, \pi)}{\mathbb{P}(X_k = y | X_0 = x, \pi)} = \frac{\mathbb{P}_{\tau \sim \mathcal{T}(x, a, \pi)}(X_k = y)}{\mathbb{P}_{\tau \sim \mathcal{T}(x, \pi)}(X_k = y)}.$$

Using this identity and importance sampling, we can rewrite the usual Q-function in terms of h_k . Since throughout there is only one policy π involved, we will drop the explicit conditioning, but it is implied.

Theorem 1. *Consider an action a and a state x for which $\pi(a|x) > 0$. Then the following holds*

$$Q^\pi(x, a) = r(x, a) + \mathbb{E}_{\tau \sim \mathcal{T}(x, \pi)} \left[\sum_{k \geq 1} \gamma^k \frac{h_k(a|x, X_k)}{\pi(a|x)} R_k \right].$$

So, each of the rewards R_k along the way is weighted by the ratio $\frac{h_k(a|x, X_k)}{\pi(a|x)}$, which exactly quantifies how relevant a was in achieving the corresponding state X_k . Following the discussion above, this ratio is 1 if a is irrelevant, and larger or smaller than 1 in the other cases. The expression for the Q-function is similar to that in Eq. (1), but the new expectation is no longer conditioned on the initial action a – the policy π is followed from the start ($A_0 \sim \pi(\cdot|x)$ instead of $A_0 = a$). This is an important point, as it will allow us to use returns generated by *any* action A_0 to update the values of all actions, to the extent that they are relevant according to $\frac{h_k(a|x, X_k)}{\pi(a|x)}$. Theorem 1 implies the following expression for the advantage:

$$A^\pi(x, a) = r(x, a) - r^\pi(x) + \mathbb{E}_{\tau \sim \mathcal{T}(x, \pi)} \left[\sum_{k \geq 1} \left(\frac{h_k(a|x, X_k)}{\pi(a|x)} - 1 \right) \gamma^k R_k \right], \quad (3)$$

where $r^\pi(x) = \sum_{a \in \mathcal{A}} \pi(a|x) r(x, a)$. This form of the advantage is particularly appealing, since it directly removes irrelevant rewards from consideration. Indeed, whenever $\frac{h_k(a|x, X_k)}{\pi(a|x)} = 1$, the reward R_k does not participate in the advantage for the value of action a . When there is inconsequential noise that is outside of the agent’s control, this may greatly reduce the variance of the estimates.

Removing time dependence. For clarity of exposition, here we have considered the hindsight distribution to be additionally conditioned on time. Indeed, h_k depends not only on reaching the state, but also on the number of timesteps k that it takes to do so. In general, this can be limiting, as it introduces a stronger dependence on the particular trajectory, and a harder estimation problem of the hindsight distribution. It turns out we can generalize all of the results presented here to a *time-independent* distribution $h_\beta(a|x, y)$, which gives the probability of a conditioned on reaching y at some point in the future. The scalar $\beta \in [0, 1)$ is the "probability of survival" at each step. This can either be the discount γ , or a termination probability if the problem is undiscounted. In the discounted reward case Eq. (3) can be re-expressed in terms of h_β as follows:

$$A^\pi(x, a) = r(x, a) - r^\pi(x) + \mathbb{E}_{\tau \sim \mathcal{T}(x, \pi)} \left[\sum_{k \geq 1} \left(\frac{h_\beta(a|x, X_k)}{\pi(a|x)} - 1 \right) \gamma^k R_k \right], \quad (4)$$

with the choice of $\beta = \gamma$. The interested reader may find the relevant proofs in the appendix.

Finally, it is possible to obtain a hindsight V-function, analogously to the Q-function from Theorem 1. The next section does this for *return*-conditional HCA. We include other variations in appendix.

3.2 Conditioning on Future Returns

The previous section derived Q-functions that explicitly reweigh the rewards at each step, based on the corresponding states’ connection to the action whose value we wish to estimate. Since ultimately we are interested in the return, we could alternatively use it for future conditioning itself.

Definition 2 (*Return-conditional hindsight distributions*). For any action a and any possible return z , define $h_z(a|x, \pi, z)$ to be the conditional probability over trajectories $\tau \sim \mathcal{T}(x, \pi)$ of the first action A_0 being a , given that z has been observed along τ :

$$h_z(a|x, \pi, z) \stackrel{\text{def}}{=} \mathbb{P}_{\tau \sim \mathcal{T}(x, \pi)} (A_0 = a | Z(\tau) = z).$$

The distribution $h_z(a|x, \pi, z)$ is intuitively similar to h_k , but instead of future states, it directly quantifies the relevance of a to obtaining the entire return z . This is appealing, since in the end we care about returns. Further, this could be simpler to learn, since instead of the possibly high-dimensional state, we now need to worry only about a scalar outcome. On the other hand, it is no longer "jumpy" in time, so may benefit less from structure in the dynamics. As with h_k , we will drop the explicit conditioning on π , but it is implied. We have the following result.

Theorem 2. Consider an action a , and assume that for any possible random return $z = Z(\tau)$ for some trajectory $\tau \sim \mathcal{T}(x, \pi)$ we have $h_z(a|x, z) > 0$. Then we have:

$$V^\pi(x) = \mathbb{E}_{\tau \sim \mathcal{T}(x, a, \pi)} \left[Z(\tau) \frac{\pi(a|x)}{h_z(a|x, Z(\tau))} \right]. \quad (5)$$

The V- (rather than Q-) function form here has interesting properties that we will discuss in the next section. Mathematically, the two forms are analogous to derive, but the ratio is now flipped. Equations (5) and (1) imply the following expression for the advantage:

$$A^\pi(x, a) = \mathbb{E}_{\tau \sim \mathcal{T}(x, a, \pi)} \left[\left(1 - \frac{\pi(a|x)}{h_z(a|x, Z(\tau))} \right) Z(\tau) \right]. \quad (6)$$

The factor $c(a|x, Z) = 1 - \frac{\pi(a|x)}{h_z(a|x, Z)}$ expresses how much a single action a contributed to obtaining a return Z . If other actions (drawn from $\pi(\cdot|x)$) would have yielded the same return, $c(a|x, Z) = 0$, and the advantage is 0. If an action a has made achieving Z more likely, then $c(a|x, Z) > 0$, and conversely, if other actions would have contributed to achieving Z more than a , then $c(a|x, Z) < 0$. Hence, $c(a|x, Z)$ expresses the impact an action has on the environment, in terms of the return, if everything else (future decisions as well as randomness of the environment) is unchanged.

Both h_β and h_z can be learned online from sampled trajectories (see Sec. 4 for algorithms, and a discussion in Sec. 4.1). Finally, while we chose to focus on state and return conditioning, one could consider other options. For example, conditioning on the reward (instead of the state) at a future time k , or an embedding of (or part of) the future trajectory, could have interesting properties.

3.3 Policy Gradients

We now give a policy gradient theorem based on the new expressions of the value function.

Theorem 3. *Let π_θ be the policy parameterized by θ , and $\beta = \gamma$. Then, the gradient of the value at some state x_0 is:*

$$\nabla_\theta V^{\pi_\theta}(x_0) = \mathbb{E}_{\tau \sim \mathcal{T}(x_0, \pi_\theta)} \left[\sum_{k \geq 0} \gamma^k \sum_a \nabla \pi_\theta(a|X_k) Q^x(X_k, a) \right] \quad (7)$$

$$= \mathbb{E}_{\tau \sim \mathcal{T}(x_0, \pi_\theta)} \left[\sum_{k \geq 0} \gamma^k \nabla \log \pi_\theta(A_k|X_k) A^z(X_k, A_k) \right], \quad (8)$$

$$Q^x(X_k, a) \stackrel{\text{def}}{=} r(X_k, a) + \sum_{t \geq k+1} \gamma^{t-k} \frac{h_\beta(a|X_k, X_t)}{\pi_\theta(a|X_k)} R_t,$$

$$A^z(x, a) \stackrel{\text{def}}{=} \left(1 - \frac{\pi_\theta(a|x)}{h_z(a|x, Z(\tau_{k:\infty}))} \right) Z(\tau_{k:\infty}).$$

Note that the expression for state HCA in Eq. (7) is written for all actions, rather than only the sampled one. Interestingly, this form does not require (or benefit from) a baseline. Contrary to the usual all-actions algorithm which uses the critic, the HCA reweighting allows us to use returns sampled from a particular starting action to obtain value estimates for all actions.

4 Algorithms

Using the new policy gradient theorem, we will now give novel algorithms based on sampling the expectations (7) and (8). Then, we will discuss the training of the relevant hindsight distributions.

State-Conditional HCA Consider a parametric representation of the policy $\pi(\cdot|x)$ and the future-state-conditional distribution $h_\beta(a|x, y)$, as well as the baseline V and an estimate of the immediate reward $\hat{r}$. Generate T -step trajectories $\tau^T = (X_s, A_s, R_s)_{0 \leq s \leq T}$. We can compose an estimate of the return for all actions a (see Theorem 7 in appendix):

$$Q^x(X_s, a) \approx \hat{r}(X_s, a) + \sum_{t=s+1}^{T-1} \gamma^{t-s} \frac{h_\beta(a|X_s, X_t)}{\pi(a|X_s)} R_t + \gamma^{T-s} \frac{h_\beta(a|X_s, X_T)}{\pi(a|X_s)} V(X_T).$$

The algorithm proceeds by training $V(X_s)$ to predict the usual return $Z_s = \sum_{t=s}^{T-1} \gamma^{t-s} R_t + \gamma^{T-s} V(X_T)$ and $\hat{r}(X_s, A_s)$ to predict R_s (square loss), the hindsight distribution $h_\beta(a|X_s, X_t)$ to predict A_s (cross entropy loss), and finally by updating the policy logits with $\sum_a Q^x(X_s, a) \nabla \pi(a|X_s)$. See Algorithm 1 in appendix for the detailed pseudocode.

Return-Conditional HCA Consider a parametric representation of the policy $\pi(\cdot|x)$ and the return-conditioned distribution $h_z(a|x, z)$. Generate full trajectories $\tau = (X_s, A_s, R_s)_{s \in \mathbb{N}^+}$ and compute the sampled advantage at each step:

$$A^z(X_s, A_s) = \left(1 - \frac{\pi(A_s|X_s)}{h_z(A_s|X_s, Z_s)}\right) Z_s,$$

where $Z_s = \sum_{t \geq s} \gamma^{t-s} R_t$. The algorithm proceeds by training the hindsight distribution $h_z(a|X_s, Z_s)$ to predict A_s (cross entropy loss), and updating the policy gradient with $\nabla \log \pi(A_s | X_s) A^z(X_s, A_s)$. See Algorithm 2 in appendix for the detailed pseudocode.

RL without value functions. The return-conditional version lends itself to a particularly simple algorithm. In particular, we no longer need to learn the value function V – if $h_z(a|X_s, Z_s)$ is estimated well, using complete rollouts is feasible without variance issues. This takes our idea of reversing the direction of the learning question to the extreme, it is now *entirely* in hindsight.

The result is an actor-critic algorithm, where the usual baseline $V(X_s)$ is replaced by $b_s \stackrel{\text{def}}{=} \frac{\pi(A_s|X_s)}{h_z(A_s|X_s, Z_s)} Z_s$. This baseline is strongly correlated to the return Z_s (it is proportional to it), which is desirable since we would like to remove as much of the variance (due to the dynamics of the world, or the agent’s own policy) as possible. The following proposition verifies that despite being correlated, this baseline does not introduce bias into the policy gradient.

Proposition 1. *The baseline $b_s = \frac{\pi(A_s|X_s)}{h_z(A_s|X_s, Z_s)} Z_s$ does not introduce any bias in the policy gradient:*

$$\mathbb{E}_{\tau \sim \mathcal{T}(x_0, \pi)} \left[\sum_s \gamma^s \nabla \log \pi(A_s|X_s) (Z_s(\tau) - b_s) \right] = \nabla V(x_0).$$

4.1 Learning Hindsight Distributions

We have given equivalent rewritings of the usual value functions in terms of the proposed hindsight distributions, and have motivated their properties, when they are accurate. Now, the question is if it is feasible to learn good estimates of those distributions from experience, and whether shifting the learning problem in this way is beneficial. The remainder of this section discusses this question, while the next one provides empirical evidence for the affirmative.

There are several conventional objects that could be learned to help with credit assignment: a value function, a forward model, or an inverse model over states. An accurate forward model allows one to compute value functions directly with no variance, and an accurate inverse model – to perform precise credit assignment. However, learning such generative models accurately is difficult and has been a long-standing challenge in RL, especially in high-dimensional state spaces. Interestingly, the hindsight distribution is a discriminative, rather than generative model, and is hence not required to model the full distribution over states. Additionally, the action space is usually much smaller than the state space, and so shifting the focus to actions potentially makes the problem much easier. When certain structure in the dynamics is present, learning hindsight distributions may be significantly easier still – e.g. if the transition model is stochastic or the policy is changing, a particular (x, a) can lead to many possible future states, but a particular future state can be explained by a small number of past actions. In general, learning h_z and h_β are supervised learning problems, so the new algorithms delegate some of the learning difficulty in RL to a supervised setting, for which many efficient approaches exist (e.g. [6, 22]).

5 Experiments

To empirically validate our proposal in a controlled way, we devised a set of diagnostic tasks that highlight issues 1-4, while also being representative of what occurs in practice (Fig. 2). We then systematically verify the intuitions developed throughout the paper. In all cases, we learn the hindsight distributions in tandem with the control policy. For each problem we compare HCA with state and return conditioning to standard baseline policy gradient, that is: n -step advantage actor critic (with $n = \infty$ for Monte Carlo). All the results are an average of 100 independent runs, with the plots depicting means and standard deviations. For simplicity we take $\gamma = 1$ in all of the tasks.

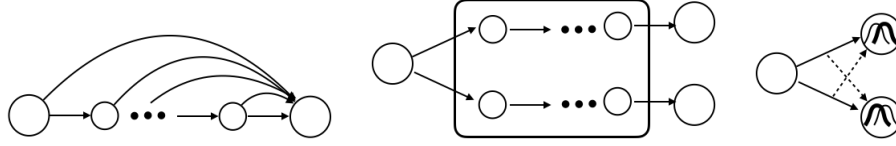


Figure 2: **Left:** Shortcut. Each state has two actions, one transitions directly to the goal, the other to the next state of the chain. **Center:** Delayed effect. Start state presents a choice of two actions, followed by an aliased chain, with the consequence of the initial choice apparent only in the final state. **Right:** Ambiguous bandit. Each action transitions to a particular state with high probability, but to the other action's state with low probability. When the two states have noisy rewards, credit assignment to each action becomes challenging.

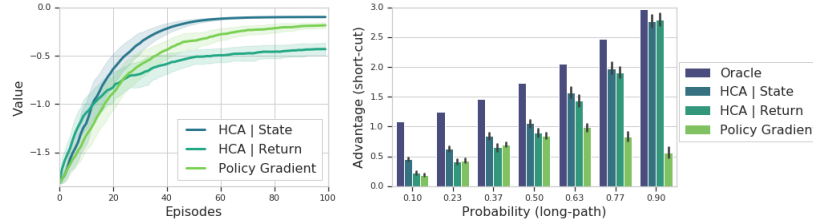


Figure 3: **Shortcut.** **Left:** learning curves for $n = 5$ with the policy between long and short paths initialized uniformly. Explicitly considering the likelihood of reaching the final state allows state-conditioned HCA to more quickly adjust its policy. **Right:** the advantage of the shortcut action estimated by performing 1000 rollouts from a fixed policy. The x -axis depicts the policy probabilities of the actions on the long path. The oracle is computed analytically without sampling. When the shortcut action is unlikely and rarely encountered, it is difficult to obtain an accurate estimate of the advantage. HCA is consistently able to maintain larger (and more accurate) advantages.

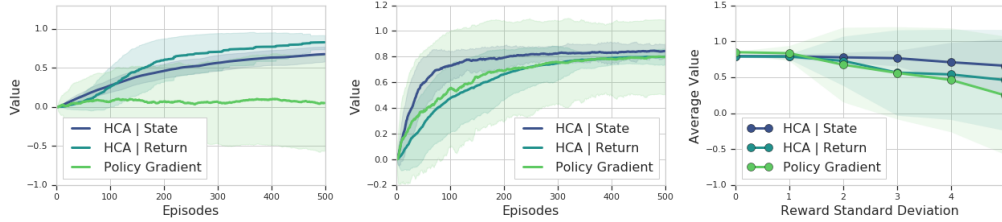


Figure 4: **Delayed effect.** **Left:** Bootstrapping. The learning curves for $n = 5$, $\sigma = 0$, and a 3-step return, which causes the agent to bootstrap in the partially observed region. As expected, naive bootstrapping is unable to learn a good estimate. **Middle:** Using full Monte Carlo returns (for $n = 3$) overcomes partial observability, but is prone to noise. The plot depicts learning curves for the setting with added white noise of $\sigma = 2$. **Right:** The average performance w.r.t. different noise levels – predictably, state HCA is the most robust.

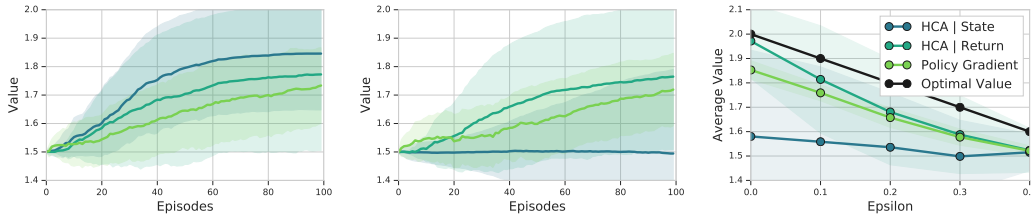


Figure 5: **Ambiguous bandit with Gaussian rewards of means 1, 2, and standard deviation 1.5.** **Left:** The state identity is observed. Both HCA methods improve on PG. **Middle:** The state identity is hidden, handicapping state HCA, but return HCA continues to improve on PG. **Right:** Average performance w.r.t. different ϵ -s with Gaussian rewards of means 1, 2, and standard deviation 0.5. Note that the optimal value itself decays in this case.

Shortcut. We begin with an example capturing the intuition from Fig. 1 (left). Fig. 2 (left) depicts a chain of length n with a rewarding final state. At each step, one action takes a shortcut and directly transitions to the final state, while the other continues on the longer path, which may be more likely according to the policy. There is a per-step penalty (of -1), and a final reward of 1. There is also a chance (of 0.1) that the agent transitions to the absorbing state directly.

This problem highlights two issues: (1) the importance of counter-factual credit assignment (issue 4); when the long path is taken more frequently than the shortcut path, counter-factual updates become increasingly effective (see Fig. 3, right) (2) the use of time as a proxy for relevance (issue 3) is shown to be only a heuristic, even in a fully-observable MDP. The relevance for the states along the chain is not accurately reflected in the long temporal distance between them and the goal state. In Fig. 3 we show that HCA is more effective at quickly adjusting the policy towards the shortcut action.

Delayed Effect. The next task instantiates the example from Fig. 1 (right). Fig. 2 (middle) depicts a POMDP, in which after the first decision, there is aliasing until the final state. This is a common case of partial observability, and is especially pertinent if the features are being learned. We show that (1) Bootstrapping naively is inadequate in this case (issue 2), but HCA is able to carry the appropriate information;¹ and (2) While Monte Carlo is able to overcome the partial observability, its performance deteriorates when intermediate reward noise is present (issue 1). HCA on the other hand is able to reduce the variance due to the irrelevant noise in the rewards.

Additionally, in this example the first decision is the most relevant choice, despite being the most temporally remote, once again highlighting that using temporal proximity for credit assignment is a heuristic (issue 3). One of the final states is rewarding (with $r = 1$), the other penalizing (with $r = -1$), and the middle states contain white noise of standard deviation σ . Fig. 4 depicts our results. In this task, the return-conditional HCA has a more difficult learning problem, as it needs to correctly model the noise distribution to condition on, which is as difficult as learning the values naively, and hence performs similarly to the baseline.

Ambiguous Bandit. Finally, to emphasize that credit assignment can be challenging, even when it is not long-term, we consider a problem without a temporal component. Fig. 2 (right) depicts a bandit with two actions, leading to two different states, whose reward functions are similar (here: drawn from overlapping Gaussian distributions), with some probability ϵ of crossover. The challenge here is due to variance (issue 1) and a lack of counter-factual updates (issue 4). It is difficult to tell whether an action was genuinely better, or just happened to be on the tail end of the distribution. This is a common scenario when bootstrapping with similar values. Due to the explicit aim at modeling the distributions, the hindsight algorithms are more efficient (Fig. 5 (left)).

To highlight the differences between the two types of hindsight conditioning, we introduce partial observability (issue 2), see Fig. 5 (right). The return-conditional policy is still able to improve over policy gradient, but state-conditioning now fails to provide informative conditioning (by construction).

6 Related Work

Hindsight experience replay (HER) [1] introduces the idea of off-policy learning about many goals from the same trajectory. The intuition is that regardless of what goal the trajectory was pursuing originally, *in hindsight* it, e.g., successfully found the one corresponding to its final state, and there is something to be learned. Rauber et al. [14] extend the same intuition to policy gradient algorithms, with goal-conditioned policies. Goyal et al. [4] also use goal conditioning and learn a backtracking model, which predicts the state-action pairs occurring on trajectories that end up in goal states. These works share our intuition of in hindsight using the same data to learn about many things, but in the context of goal-conditioned policies, while we essentially contrast conditional and unconditional policies, where the conditioning is on the extra outcome (state or return). Note that we never act w.r.t. the conditional policy, and it is used solely for credit assignment. Prioritized sweeping can be viewed as changing the sampling distribution with hindsight knowledge of the TD errors [12].

Another line of work that aims to propagate credit efficiently backward in time is the temporal value transport algorithm [10], in which an attention mechanism over memory is used to jump over parts of a trajectory that are irrelevant for the rewards obtained. While demonstrated on challenging problems,

¹See the discussion in Appendix F

that method is biased; a promising direction for future research would be to apply our unbiased hindsight mechanism with past states chosen by such an attention mechanism.

A large number of variance reduction techniques have been applied in RL, e.g. using learned value functions as critics, and other control variates [e.g. 23]. When a model of the environment is available, it can be used to reduce variance. Rollouts from the same state fill the same role in policy gradients [17]. Differentiable system dynamics allow low-variance estimates of the Q-value gradient by using the pathwise derivative estimator, effectively backpropagating the gradient of the objective along trajectories [e.g. 16, 8, 9]. In stochastic systems this requires knowledge of the environment noise. To bypass this, Heess et al. [8] *infer* the noise given an observed trajectory. Buesing et al. [2] apply this idea to POMDPs, where it can be viewed as reasoning about events in hindsight. They use a structural causal model of the dynamics and infer the posterior over latent causes from empirical trajectories. Using an empirical rather than a learned distribution over latent causes can reduce bias and, together with the (deterministic) model of the system dynamics, allows exploring the effect of alternative action choices for an observed trajectory.

Inverse models similar to the ones we use appear, for instance, in variational intrinsic control [5] (see also e.g. [7]). However, in our work, the inverse model serves as a way of determining the influence of an action on a future outcome, whereas the work in [5, 7] aims to use the inverse model to derive an intrinsic reward for training policies in which actions influence the future observations.

7 Closing

We proposed a new family of algorithms that explicitly consider the question of credit assignment as a part of, or instead of, estimating the traditional value function. The proposed estimators come with new properties, and as we validate empirically, are able to address some of the key issues in credit assignment. Investigating the scalability of these algorithms in the deep reinforcement learning setting is an exciting problem for future research.

Acknowledgements

The authors thank Joseph Modayil for reviews of earlier manuscripts, Theo Weber for several insightful suggestions, and the anonymous reviewers for their useful feedback.

References

- [1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.
- [2] Lars Buesing, Theophane Weber, Yori Zwols, Sébastien Racanière, Arthur Guez, Jean-Baptiste Lespiau, and Nicolas Heess. Woulda, coulda, shoulda: Counterfactually-guided policy search. *CoRR*, abs/1811.06272, 2018.
- [3] Igor Vladimirovich Girsanov. On transforming a certain class of stochastic processes by absolutely continuous substitution of measures. *Theory of Probability & Its Applications*, 5(3):285–301, 1960.
- [4] Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.
- [5] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.
- [6] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.
- [7] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations (ICLR)*, 2018.

- [8] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems* 28, pages 2944–2952. Curran Associates, Inc., 2015.
- [9] Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.
- [10] Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *arXiv preprint arXiv:1810.06721*, 2018.
- [11] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.
- [12] Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13(1):103–130, 1993.
- [13] Martin Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.
- [14] Paulo Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. In *International Conference on Learning Representations (ICLR)*, 2019.
- [15] Gerardo Rubino, Bruno Tuffin, et al. *Rare event simulation using Monte Carlo methods*, volume 73. Wiley Online Library, 2009.
- [16] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. *CoRR*, abs/1506.05254, 2015.
- [17] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.
- [18] Satinder Singh, Tommi Jaakkola, and Michael I. Jordan. Learning without state estimation in partially observable environments. In *International Conference on Machine Learning (ICML)*, 1994.
- [19] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.
- [20] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018.
- [21] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.
- [22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [23] Théophane Weber, Nicolas Heess, Lars Buesing, and David Silver. Credit assignment techniques in stochastic computation graphs. *CoRR*, abs/1901.01761, 2019.
- [24] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.