Offline Credit Assignment in Deep Reinforcement Learning with Hindsight Discriminator Networks

Johan Ferret jferret@google.com
Google Research, Brain Team
Paris, France

Olivier Pietquin pietquin@google.com
Google Research, Brain Team
Paris, France

Matthieu Geist mfgeist@google.com
Google Research, Brain Team
Paris, France

Abstract

Current RL approaches typically struggle in complex settings (non-Markov, partially observable, with temporal delay and stochasticity). This is related to the credit assignment problem, i.e. reinforcing actions which have indirect, noisy, delayed effects on performance. Credit assignment approaches aim at alleviating this issue in different ways: adapting the discount factor on-the-fly, redistributing rewards or values to past actions, and many others. In this work, we study a simple self-supervised method for offline credit assignment we call *Discriminator Credit Assignment*: given the realization of a future outcome, a neural network discriminator learns to tell apart the actual, played action from a fictitious action. We study its relation to existing credit assignment methods and show that it can be viewed as an alternate form of Hindsight Credit Assignment (Harutyunyan et al., 2019). We show that there are benefits to this formulation, notably for sample-efficiency and the potential of scaling to sequences of actions. We demonstrate the efficiency of the approach in difficult offline scenarios, and show that the assigned credit matches human intuitions well.

Keywords: Reinforcement Learning, Credit Assignment, Offline Learning

1. Introduction

At the heart of Reinforcement Learning (RL) lies the exploration/exploitation dilemma. On the one hand, agents should seek additional information about the task, by exploring seldom visited or novel parts of the state space. On the other, agents should seek improvement, which calls for reinforcing actions that are known to lead to larger payoffs. Credit assignment, which consists in associating actions to future outcomes (such as future payoffs), is thus key to RL. Most RL methods can be divided into two major categories: value-based methods and policy-based methods. Value-based methods learn action-values that assign a payoff to actions in a given state, and actions are selected by sampling actions according to their payoff (*e.g.* acting greedily and selecting the action with the highest payoff). Policy-based methods learn a policy, which assigns action probabilities given a current state, and values, and actions are selected by sampling from the distribution induced by the policy. In both paradigms, RL methods use a form of value estimation, which is powered by temporal differences (TD). TD consists in bootstrapping by the sum of immediate rewards and the discounted value of a future state, which can be seen as a credit assignment method based on temporal proximity to rewards. More advanced credit assignment methods aim at complementing or even bypassing TD. Such methods suffer from various flaws: they are either ad-hoc (Raposo et al., 2021), hard to scale (Harutyunyan et al., 2019) or exhibit a large complexity (Arjona-Medina et al., 2019; Hung et al., 2019; Mesnard et al., 2021). In this work, we instead go for a simple and scalable approach. Inspired from self-supervised learning successes on several modalities (), we aim at learning to assign credit from agent interactions, in a self-supervised way. We introduce Discriminator Credit Assignment (DisCA). DisCA leverages a discriminator whose main task is to distinguish real actions from fictitious ones, in the context of a given state ***and*** conditionally to a future outcome. Conditioning on the future is key to credit assignment because we want to assign greater credit to actions that are particularly more likely than other actions when the outcome is better than usual. We show that there is a connection between DisCA and another credit assignment

method called Hindsight Credit Assignment (HCA) (Harutyunyan et al., 2019). Empirically, through proper ablation studies, we also show that a few design choices are key to obtain best performances with DisCA. Additionally, we turn DisCA outputs into an auxiliary reward function and show that it matches our expert knowledge about the tasks at hand, accurately identifying key decisions and actions as such.

Our contributions can be summarized as follows: **1)** we propose DisCA, a simple and scalable method for temporal credit assignment in RL, **2)** we formalize the DisCA estimation problem and discuss connections with existing notions and work, **3)** we empirically validate the DisCA method on a synthetic task involving difficult credit assignment.

2. Background

We now introduce concepts and mathematical objects of interest that we use in further sections.

2.1 Reinforcement Learning

We use the Markov Decision Process (MDP) formalism (Puterman, 1994). An MDP is a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \gamma, \mathcal{P})$ with $\mathcal{S}$ a finite state space, $\mathcal{A}$ a discrete action space, $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ is a bounded reward function, $\gamma \in [0, 1)$ is a discount factor, and $\mathcal{P} \in \Delta_{\mathcal{S}\times\mathcal{A}}^{\mathcal{S}}$ is a transition kernel. Here and in what is next Δ_X is the simplex over the set X. A policy is an object $\pi \in \Delta_{\mathcal{A}}^{\mathcal{S}}$. The value of a policy is the quantity:

$$V_\pi(s_t) = \mathbb{E}_{A_t\sim\pi(\cdot|s_t),S_{t+1}\sim\mathcal{P}(\cdot|s_t,A_t)}\left[\sum_{k=0}^{+\infty}\gamma^k r(S_{t+k}, A_{t+k})|S_t = s_t\right].$$

To simplify, we note the previous expectation $\mathbb{E}_\pi$ when there is no ambiguity. The action-value (or Q-value) of a policy is the quantity $Q_\pi(s_t, a_t) = \mathbb{E}_\pi\big[\sum_{k=0}^{+\infty}\gamma^k r(S_{t+k}, A_{t+k})|S_t = s_t, A_t = a_t\big]$. A policy π^* is optimal iff $V_{\pi^*}(s) \geq V_\pi(s)\,\forall s \in \mathcal{S}$. Value functions verify the *Bellman equation*:

$$V_\pi(s_t) = \mathbb{E}_\pi\Big[r(s_t, a_t) + \gamma V_\pi(s_{t+1})\Big],$$

with a right-hand term that we call *Bellman target*. When we have full access to the MDP (*i.e.* to the reward function and to the transition kernel), we can use Dynamic Programming (DP) (Howard, 1960) to compute optimal policies. For instance, policy iteration alternates steps of policy evaluation (*i.e.* estimate V_{π_k}) and policy improvement (*i.e.* update π_k to π_{k+1} so that $V_{\pi_{k+1}} \geq V_{\pi_k}\,\forall s \in \mathcal{S}$). Policy evaluation recursively applies the Bellman equation starting from arbitrary initial values to obtain V_{π_k}. Policy improvement then computes the greedy policy wrt V_{π_k}, that it is $\pi_{k+1} = \mathcal{G}(V_{\pi_k})$, with $\mathcal{G}(V_{\pi_k})(a_t|s_t) = \mathbb{1}\{a_t = \arg\max_a Q_{\pi_k}(s_t, a)\}$, and $Q_{\pi_k}(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_\pi[V_\pi(s_{t+1})]$. Typically, models of the MDP (*i.e.* r and $\mathcal{P}$) are not available so DP methods are replaced by approximate methods. Temporal differences (TD) (Sutton, 1984) use samples of the Bellman target (called *TD targets*) as means of supervision for estimated value functions.

2.2 Credit Assignment

We start by presenting the principal families of credit assignment methods in RL, and then present in more details methods that are closest to our work, notably those that involve policy functions conditioned on the future.

General directions There are many distinct ways to improve credit assignment in RL. Most methods perform a redistribution of the rewards (Ferret et al., 2020; Raposo et al., 2021), returns (Liu et al., 2019; Gangwani et al., 2020) or values (Hung et al., 2019) towards past actions. This requires identifying which previous actions contribute to later effects, which is generally done by having a learned linear combination (*e.g.* attention mechanisms, soft memory retrieval) as part of a predictive task or by learning an explicit decomposition of the quantity redistributed. Another line of research seeks to adapt important parameters of RL methods, such as the discount factor γ or the bias-variance trade-off parameter λ in TD(λ), which is either done by meta-learning (Xu et al., 2018) or via bandit algorithms (Badia et al., 2020b). Emphatic TD learning (Sutton et al., 2016) incorporates a notion of state importance inside TD, with the goal of increasing the visitation of important states. Finally, approaches that condition the value function on the

future (Guez et al., 2020; Mesnard et al., 2021) are also useful for credit assignment since they can reduce update variance and pinpoint action contributions in a more narrow way. In such methods, the future is represented as a learned embedding of the subsequent observations.

Policies conditioned on the future Conditioning policies on a desirable future (either a discounted sum of rewards achieved or a particular state) is an idea that is used a lot in goal-conditioned RL (Andrychowicz et al., 2017), where the agent has to reach a goal and passing the goal as input helps it adapt to the current objective. For credit assignment, intuitively, it appears useful to be able to act according to an outcome to achieve, given that we know which outcomes should be pursued. An advantage is that this formulation reduces credit assignment to a supervised problem: predicting the action to be played in order to reach the chosen outcome. The resulting future-conditioned policy can be used to sample actions conditionally to desirable outcomes. The notion of desirability is not trivial to define, though. Several methods explore this idea. Reward-Conditioned Policies (RCP) (Kumar et al., 2019) learn a policy conditioned on an outcome (either the return or the advantage) and a generator to define which outcome to condition on at inference time. Upside-Down RL (Schmidhuber, 2019; Srivastava et al., 2019) takes a similar approach but conditions on a temporal horizon along the return, essentially asking the estimator to predict the action to take in order to reach the specified return within the temporal window defined. Decision Transformer (DT) (Chen et al., 2021) uses a similar approach, but with a Transformer (*i.e.* a sequential architecture) as estimator, which can memorize and synthesize information from the past. It makes the assumption that the maximal return is of one and conditions on this return value when sampling actions. Predictive Information (Lee et al., 2020) estimates the mutual information between past and future interaction data as an auxiliary loss. They also propose an alternative version that replaces the state by this estimate for control.

Hindsight predictions Hindsight Credit Assignment (HCA) (Harutyunyan et al., 2019) introduces the notion of hindsight prediction, which is also equivalent to a policy conditioned on an outcome. Hindsight predictions use a neural network estimator to predict observed actions given a state and the value of a future outcome (either a future state or a trajectory return). The explicit credit for a given action is formulated as the ratio $c_\pi(s, a|s') = h_\pi(a|s, s')/\pi(a|s)$, in accordance with the intuition that actions that are much more probable given a future outcome are direct contributors. The credit is used in a modified policy gradient method, which updates all actions at once: $g_{\theta,HCA} = \sum_{a\in\mathcal{A}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a|s_t)\phi_{HCA}(s_t, a)$, where $\phi_{HCA} = r_\theta(s_t, a)+\sum_{t'=t}^{T} \gamma^{t'-t}\big(h_{\theta,\beta}(a|s_t, s_{t'})/\pi_\theta(a|s_t)\big)r(s_{t'}, a_{t'})$.

2.3 Self-supervised Learning

We briefly review the state of self-supervised learning approaches, since we build on those.

Our method builds on Reversibility-Aware RL (RARL) (Grinsztajn et al., 2021), notably its reversibility estimation component. RARL estimates how easily reversible actions are via a self-supervised objective: that of predicting which of two observations came first in an observed agent trajectory (which is theoretically motivated). On a high-level, if the estimator is incapable of telling which observation comes first, then the transition is deemed reversible. On the contrary, if the estimator is very confident that the first observation comes first, the transition is deemed irreversible. Then, at inference, how reversible an action is gets estimated as a function of the probability assigned by the estimator, for a pair of subsequent observations. Similar ideas can be applied to assign credit in the context of RL: given a real action and a fictitious action along with the observed outcome of a trajectory, a discriminator can be trained to tell apart the real action from the fake one, and its confidence can be used as a proxy for credit assignment. This is the idea that we explore in the next section.

3. Discriminator Credit Assignment

We now present in more details our discriminative approach to credit assignment and formalize it first.

Formalism. We note the outcome of choice Z. As stated before, it can take different forms: the return of the trajectory, whether a particular state is reached, etc. Generally speaking, it can be any function of the history: $Z_t = f(H_t)$. In what is next, we suppose it is a function of the entire trajectory (*e.g.* $Z_T = f(H_T)$) without loss of generality. We note

the space of outcomes $\mathcal{Z}$. We introduce the following function $\psi_\pi : \mathcal{A}^2 \times \mathcal{S} \times \mathcal{Z} \to [0,1]$[1]:

$$\begin{aligned}\psi_\pi(a, a'|s, z) &:= \frac{P_\pi(A_t = a|S_t = s, Z_T = z)}{P_\pi(A_t = a|S_t = s, Z_T = z) + P_\pi(A_t = a'|S_t = s, Z_T = z)}, \\ &:= \frac{P_\pi(A_t = a|S_t = s, Z_T = z)}{P_\pi(A_t \in \{a, a'\}|S_t = s, Z_T = z)}, \\ &:= \frac{P_\pi(A_t = a|S_t = s, Z_T = z)}{U_\pi(a, a'|s, z)}.\end{aligned}$$

It roughly measures how much the single action a is responsible for the probability U_π of the union of the actions $A_t \in \{a\} \cup \{a'\}$, conditionally to the value of the state S_t and outcome Z_T. Interestingly, this quantity maximizes the following objective for a discriminator $D : \mathcal{A}^2 \times \mathcal{S} \times \mathcal{Z} \to [0,1]$:

$$J(D) = \mathbb{E}_{\substack{(s,a,z)\sim\mathcal{D} \\ \bar{a}\sim\mathcal{U}(\mathcal{A}\setminus\{a\}) \\ (a_1,a_2)=\texttt{shuffle}(a,\bar{a})}} \Big[\mathbb{1}\{a_1 = a\} \log D(a_1, a_2|s, z) + \mathbb{1}\{a_2 = a\}\Big(1 - \log D(a_1, a_2|s, z)\Big)\Big],$$

Proof We can equivalently express the `shuffle` operation as a Bernoulli random variable:

$$\begin{aligned}J(D) &= \mathbb{E}_{\substack{(s,a,z)\sim\mathcal{D} \\ \bar{a}\sim\mathcal{U}(\mathcal{A}\setminus\{a\}) \\ (a_1,a_2)=\texttt{shuffle}(a,\bar{a})}} \Big[\mathbb{1}\{a_1 = a\} \log D(a_1, a_2|s, z) + \mathbb{1}\{a_2 = a\}\Big(1 - \log D(a_1, a_2|s, z)\Big)\Big], \\ &= \mathbb{E}_{\substack{(s,a,z)\sim\mathcal{D} \\ \bar{a}\sim\mathcal{U}(\mathcal{A}\setminus\{a\}) \\ p=\text{Bernoulli}(\frac{1}{2})}} \Big[\mathbb{1}\{p = 0\} \log D(a, \bar{a}|s, z) + \mathbb{1}\{p = 1\}\Big(1 - \log D(\bar{a}, a|s, z)\Big)\Big], \\ &= \frac{1}{2} \sum_{s,a,z} \sum_{\bar{a}\neq a} P_\mathcal{D}(s, a, \bar{a}, z)\Big(\log D(a, \bar{a}|s, z) + 1 - \log D(\bar{a}, a|s, z)\Big).\end{aligned}$$

We then have the decomposition:

$$\begin{aligned}P_\mathcal{D}(s, a, \bar{a}, z) &= P_\mathcal{D}(\bar{a}|s, a, z) P_\mathcal{D}(a|s, z) P_\mathcal{D}(s, z) \\ &= \frac{1}{|\mathcal{A}| - 1} \cdot P_\mathcal{D}(a|s, z) P_\mathcal{D}(s, z)\end{aligned}$$

By injecting this back in the previous equation we get:

$$J(D) = \frac{1}{2\big(|\mathcal{A}| - 1\big)} \sum_{s,a,z} \sum_{\bar{a}\neq a} P_\mathcal{D}(a|s, z) P_\mathcal{D}(s, z)\Big(\log D(a, \bar{a}|s, z) + 1 - \log D(\bar{a}, a|s, z)\Big).$$

Due to the equality $\sum_i \sum_{j\neq i} f(i,j) = \sum_{\substack{i,j \\ i<j}} f(i,j) + f(j,i)$ we can rewrite the double sum as:

1. Note that this quantity is indefinite if $U_\pi(a, a'|s, z) = 0$, which happens if both actions have probability zero given a starting state and an outcome. We can ensure definiteness if at least one of the two actions was observed from the current policy in the same context (state and future outcome).

$$
\begin{aligned}
J(D) = \frac{1}{2\big(|\mathcal{A}|-1\big)} \sum_{\substack{s,a,\bar{a},z \\ a<\bar{a}}} & P_{\mathcal{D}}(a|s,z)P_{\mathcal{D}}(s,z)\Big(\log D(a,\bar{a}|s,z) + 1 - \log D(\bar{a},a|s,z)\Big) + \\
& P_{\mathcal{D}}(\bar{a}|s,z)P_{\mathcal{D}}(s,z)\Big(\log D(\bar{a},a|s,z) + 1 - \log D(a,\bar{a}|s,z)\Big), \\
= \frac{1}{2\big(|\mathcal{A}|-1\big)} \sum_{\substack{s,a,\bar{a},z \\ a<\bar{a}}} P_{\mathcal{D}}(s,z)\Big(& \underbrace{P_{\mathcal{D}}(a|s,z)\log D(a,\bar{a}|s,z) + P_{\mathcal{D}}(\bar{a}|s,z)(1-\log D(a,\bar{a}|s,z))}_{=A} + \\
& \underbrace{P_{\mathcal{D}}(\bar{a}|s,z)\log D(\bar{a},a|s,z) + P_{\mathcal{D}}(a|s,z)(1-\log D(\bar{a},a|s,z))}_{=B}\Big),
\end{aligned}
$$

Since $f(x) = a\log(x) + b\log(1-x)$ is maximal for $x = \frac{a}{a+b}$ when restrained to $[0,1]$ and for a, b positive, it follows that both A and B, and thus the last equality, are maximal for:

$$
\begin{aligned}
D(a,\bar{a}|s,z) &= \frac{P_{\mathcal{D}}(a|s,z)}{P_{\mathcal{D}}(a|s,z) + P_{\mathcal{D}}(\bar{a}|s,z)} \\
&= \psi(a,\bar{a}|s,z).
\end{aligned}
$$

■

This is the objective for the discriminative task we mentioned previously: telling apart a real action from a fictitious action from an observed state, action and corresponding outcome sampled from a dataset of interactions $\mathcal{D}$[2].

Link to HCA. We now explicit the link to HCA, and more precisely to hindsight distributions. We recall that the hindsight distribution is the quantity $h_\pi(a|s,z) := P_\pi(A_t = a|S_t = s, Z_T = z)$. Given an arbitrary state, action and outcome (s_t, a_t, z_T) and a fictitious action $\bar{a} \in \mathcal{A} \setminus \{a_t\}$ we have the following relations between ψ_π and h_π:

$$
h_\pi(a_t|s_t,z_T) = \psi_\pi(a_t,\bar{a}|s_t,z_T)U_\pi(a_t,\bar{a}|s_t,z_T).
$$

We can use this identity to derive an upper bound on h_π:

$$
\begin{aligned}
h_\pi(a_t|s_t,z_T) &= \frac{1}{|\mathcal{A}|-1} \sum_{\bar{a}\in\mathcal{A}\setminus\{a_t\}} \psi_\pi(a_t,\bar{a}|s_t,z_T)U_\pi(a_t,\bar{a}|s_t,z_T), \\
&\leq \frac{1}{|\mathcal{A}|-1} \sum_{\bar{a}\in\mathcal{A}\setminus\{a_t\}} \psi_\pi(a_t,\bar{a}|s_t,z_T).
\end{aligned}
$$

More generally, for any scalar set $\{w_i\}$ verifying $\sum_i w_i = 1$ and counterfactual action set $\big\{\bar{a}_i$ such that $\bar{a}_i \in \mathcal{A}\setminus\{\bar{a}\}\big\}$, we have:

$$
\begin{aligned}
h_\pi(a_t|s_t,z_T) &= \sum_i w_i h_\pi(a_t|s_t,z_T), \\
&= \sum_i w_i \psi_\pi(a_t,\bar{a}_i|s_t,z_T)U_\pi(a_t,\bar{a}_i|s_t,z_T).
\end{aligned}
$$

Method. We now present the DisCA method as a way to approximate the above quantities. DisCA works as pictured in Fig. 1. From an existing partial transition $\{s_t, a_t\}$ and corresponding outcome z_T, a pair of actions $(a_t, \bar{a})$ is created, with $\bar{a} \sim \mathcal{U}\big(\mathcal{A}\setminus\{a_t\}\big)$. The discriminator D_θ learns to distinguish which of the actions $(a_1, a_2) = \texttt{shuffle}(a_t, \bar{a})$ is real after a probabilistic shuffle. We also present an overview of the architecture of the Hindsight Discriminator Networks we use in DisCA in Fig. 2. Each input is mapped to an embedding via a specific, learned component: **1)**

2. In that context, $U_\pi(a_t, \bar{a}|s_t, z)$ is definite since after the shuffling operation exactly one of the two actions is a_t.

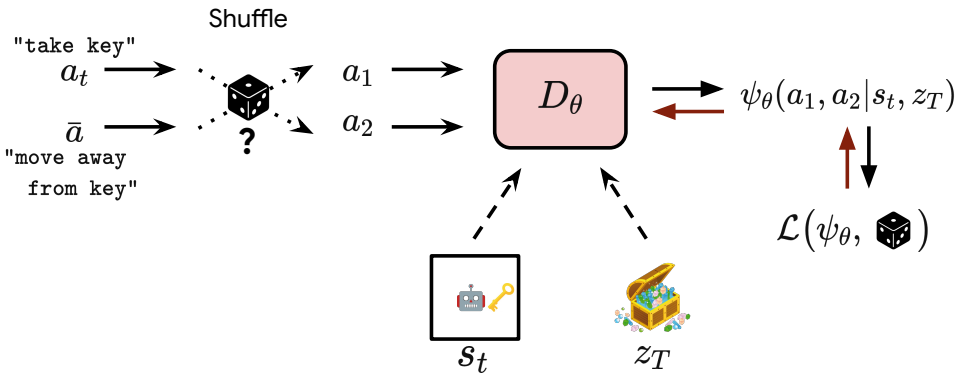

Figure 1: A high-level overview of how single-action DisCA works. From an existing partial transition $\{s_t, a_t\}$ and corresponding outcome z_T, a pair of actions $(a_t, \bar{a})$ is created, with $\bar{a} \sim \mathcal{U}(\mathcal{A} \setminus \{a_t\})$. The discriminator D_θ learns to distinguish which of the actions $(a_1, a_2) = \texttt{shuffle}(a_t, \bar{a})$ is real after a probabilistic shuffle. Red arrows correspond to gradients.

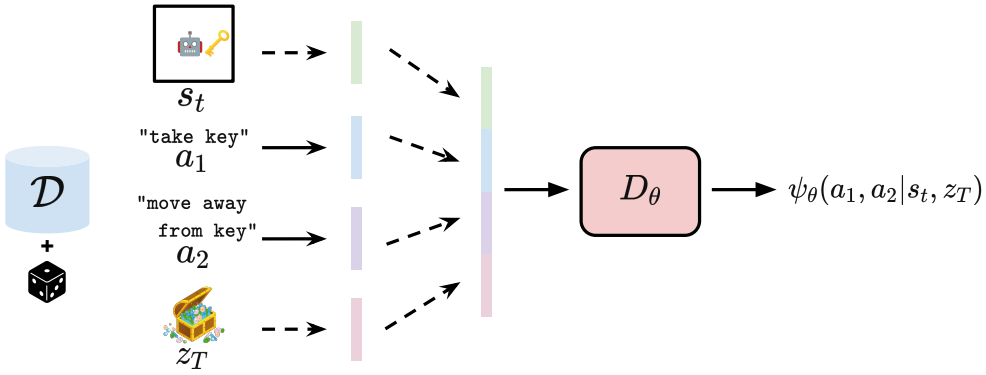

Figure 2: Hindsight Discriminator Networks used for DisCA. Each input is mapped to an embedding via a specific, learned component. Embeddings are then concatenated and passed as input to the discriminator network D_θ, which is itself an MLP and outputs a prediction probability. This probability corresponds to the approximate likelihood that a_1 is played instead of a_2.

The visual state or observation is passed into a Conv-MLP network. **2)** *Discrete actions* are represented as one-hot vectors, and passed into an MLP. **3)** *The outcome* is represented as a scalar vector of size 1 and passed into an MLP. Embeddings are then concatenated and passed as input to the discriminator network D_θ, which is itself an MLP and outputs a prediction probability. This probability corresponds to the approximate likelihood that a_1 is played instead of a_2 in the context of a starting state and future outcome.

Sampling strategies. A particularity of the estimation problem of DisCA is that it tends to be heavily imbalanced. First, with a suboptimal policies, bad outcomes are much more frequent than good outcomes, which are mostly due to chance. To alleviate this issue, we split the dataset of interaction data DisCA uses into two datasets: the first stores transitions leading to regular outcomes, the second those that lead to rare outcomes, as is common practice for credit assignment approaches (Arjona-Medina et al., 2019; Liu et al., 2019). When sampling data to train the DisCA discriminator, we sample equally from both datasets, effectively *oversampling data from trajectories featuring good outcomes*. We illustrate the approach in Fig. 3. Second, even with oversampling, there might be a significant number of states in which discriminating actions conditionally to any outcome is hard. As an example, one can think about a toy navigation task, where the agent can move across all cardinal directions. If the agent is in the middle of a large room, it

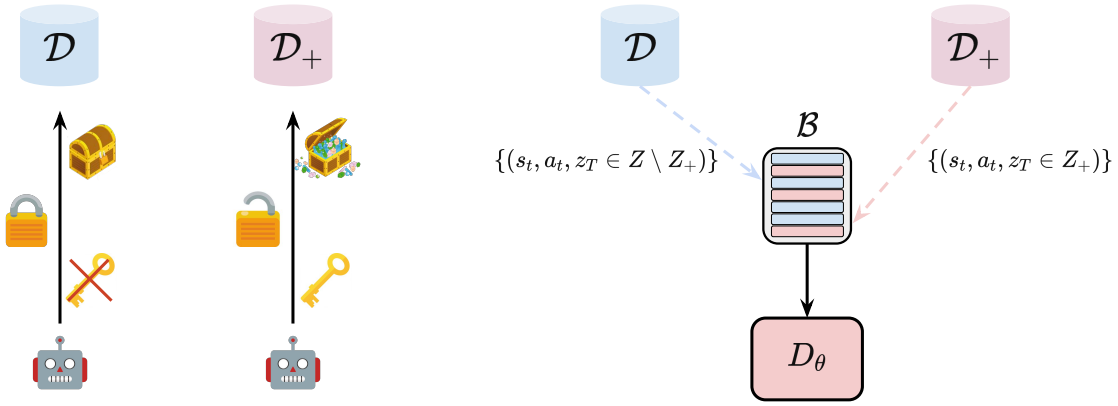

Figure 3: **Left:** Trajectories with outstanding returns are saved to the $\mathcal{D}_+$ dataset. Regular trajectories go to $\mathcal{D}$. **Right:** Batches of interaction data are sampled with equal proportion from $\mathcal{D}$ and $\mathcal{D}_+$ and fed to D_θ.

might be hard to tell if going up or left is the action that was performed, conditionally to a given goal being reached later on. The consequence is that batches that only feature states in which actions are impossible to discriminate might have low learning potential. To reduce this effect, we use *large batch sizes* when possible. This increases the probability of having one or several states in which the discriminator can learn how to meaningfully tell apart actions conditionally to the outcome. Another aspect of the estimation problem of DisCA is its symmetries and invariances. The symmetry stems from the `shuffle` operation. Fundamentally, we have the equality:

$$\psi_\pi(a_t, \bar{a}|s_t, z) = 1 - \psi_\pi(\bar{a}, a_t|s_t, z).$$

This equality is not obtained by construction in the DisCA discriminator, which means that we can pass both action pairs as inputs and average the results. A useful invariance is that on the counterfactual action. If the real action is truly crucial to reach the specified outcome, then any of the counterfactual actions can be used as a negative action with identical effects. This means that we can create several action pairs, with different counterfactual actions as input, and average the results:

$$\psi_{avg}(a_t|s_t, z_T) = \frac{1}{|\mathcal{A}| - 1} \sum_{\bar{a} \in \mathcal{A} \setminus \{a_t\}} \psi(a_t, \bar{a}|s_t, z_T).$$

Marginalizing over fictitious actions gives a sense of how much more likely the action is compared to those on average, and ψ_{avg} is the quantity we use to assign credit directly. Note that this quantity is the previously introduced upper-bound to $h(a_t|s_t, z_T)$. Of course, we can combine both ensembling strategies, which is what we do in practice.

Incorporating credit to RL. There are several options to integrate assigned credit in RL, that we review here:

- **auxiliary reward function:** one way to incorporate external credit into an RL algorithm is to modify the extrinsic reward from the environment. This is easy to do but generally optimal behaviors are not conserved, unless optimality-preserving reward modifications like potential-based reward shaping (Ng et al., 1999) or look-ahead/look-back advice (Wiewiora et al., 2003) are used.

- **modified policy gradients:** a theoretically sound way to incorporate credit in RL is to modify the policy gradients. This can lead to unbiased updates and reduced variance, but it is only applicable to policy-based RL approaches (*i.e.* that have an explicit policy network).

- **modified temporal differences:** since TD is the culprit for lack of credit assignment capabilities, a natural solution consists in modifying TD and introduce feedback loops from outcomes to past actions. A potential drawback is that, usually, this results in biased updates, though potentially reducing variance.

- **prioritization:** another approach is to prioritize transitions that get assigned important credit for updates. Unfortunately, this is mostly useful in off-policy and offline RL, where one has the ability to revisit past interaction data.

In practice, we opt for the *auxiliary reward function* approach. Though heuristic, we find empirically that the auxiliary rewards match relatively well our expert intuition about the tasks. Additionally, we observe significant performance gains, both in terms of sample-efficiency and asymptotical performance, when using auxiliary rewards from DisCA over the extrinsic reward. To define the rewards, we use the positive part difference between the discriminator output (post-ensembling) when conditioned on a good outcome and its output when conditioned on a bad outcome. The reason to use the difference is that predictions, especially in the low data regime, tend to be imprecise and close to 0.5, and the difference behaves better.

Value separation. Value estimation can suffer from evolving auxiliary reward functions. To avoid issues of scale and instability in value estimation, we adopt the value separation approach from Badia et al. (2020a). In a nutshell, value separation consists in having two separate value networks: one for the extrinsic rewards, and the other for intrinsic rewards:

$$Q_\theta(s,a) = Q_{\theta_e}(s,a) + \alpha Q_{\theta_i}(s,a).$$

Each set of Q-value weights is updated to minimize Bellman residuals for its own reward. Optimizers are kept separate too. This is helpful because separate networks (and optimizers) can adapt to the potentially distinct scale and variance of their allocated reward. We also suspect that it could help in the low data regime, when the DisCA component is not particularly precise and is being updated quickly: value separation ensures that fast updates do not mess with the value estimation from the extrinsic rewards (which are stationary). They provide a proof of the convergence of the corresponding Bellman operator, and how to proceed in practice with value-based RL agents. We propose a simple implementation of value separation for policy-based RL agents, which make use of a value function instead of a Q-value function:

$$V_\theta(s) = V_{\theta_e}(s) + \alpha V_{\theta_i}(s).$$

As with Q-values, each set of value weights minimizes Bellman residuals for its reward. A crucial difference is that policy gradients are calculated with a modified advantage:

$$\begin{aligned} A_\theta(s,a) &= A^{V_{\theta_e}}(s,a) + \alpha A^{V_{\theta_i}}(s,a), \\ &= A^{V_{\theta_e}+\alpha V_{\theta_i}}(s,a). \end{aligned}$$

A common addition to policy-based RL agents is generalized advantage estimation (GAE) (Schulman et al., 2016). We provide a simple proof that the advantage equivalence holds true for GAE advantages in Appendix A.1.

4. Experiments

4.1 Delayed Catch

We investigate how pertinent DisCA is in Delayed Catch (Raposo et al., 2021). Delayed Catch is a task in which an agent can move from left to right and has to collect balls that appear at random locations at the top of the screen and fall in a straight line. The delay comes from the fact that the agent is not rewarded directly for catching balls: instead, it gets a reward that corresponds to the total number of balls caught at the very last timestep of the episode. In Fig. 4 we display the credit assigned by DisCA along an episode of interaction played following a policy that takes a uniform random action when possible (*i.e.* when the ball is not too low or the agent not too far off), and an optimal action otherwise. This policy is optimal according to the undiscounted objective. We see that the credit assigned by DisCA is imperfect but still quite informative, since catches or actions closely preceding catches are rewarded most of the time. We then compare a standard PPO agent (Schulman et al., 2017) with the exact same agent benefitting from DisCA rewards. DisCA improves the sample-efficiency and asymptotical performance of PPO.

4.2 Key-to-Door

We evaluate the credit assigned by DisCA in Key-to-Door (Harutyunyan et al., 2019). The objective is to open a door, which only happens if the agent grabs a key in a previous sequence, with a consequent delay between important

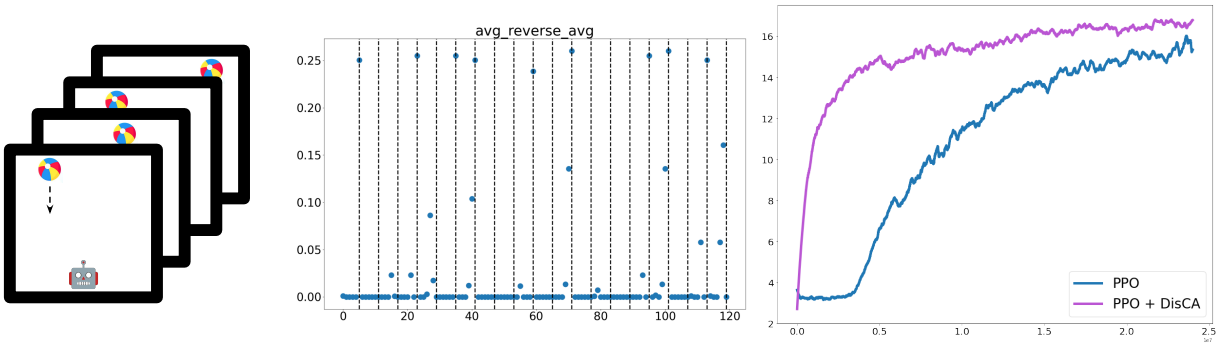

Figure 4: **Left:** A schematic illustration of Delayed Catch. In the standard setting the episode terminates after 20 tries (120 timesteps in total). In each try a single ball can be caught by the agent. The agent only receives rewards at the last timestep. **Mid:** the credit assigned by DisCA along a single episode of Delayed Catch. The x-axis corresponds to episode timesteps, the y-axis corresponds to the amount of credit assigned. Dotted vertical lines correspond to ball catches by the agent. **Right:** the agent performance of PPO and PPO with DisCA rewards on Delayed Catch.

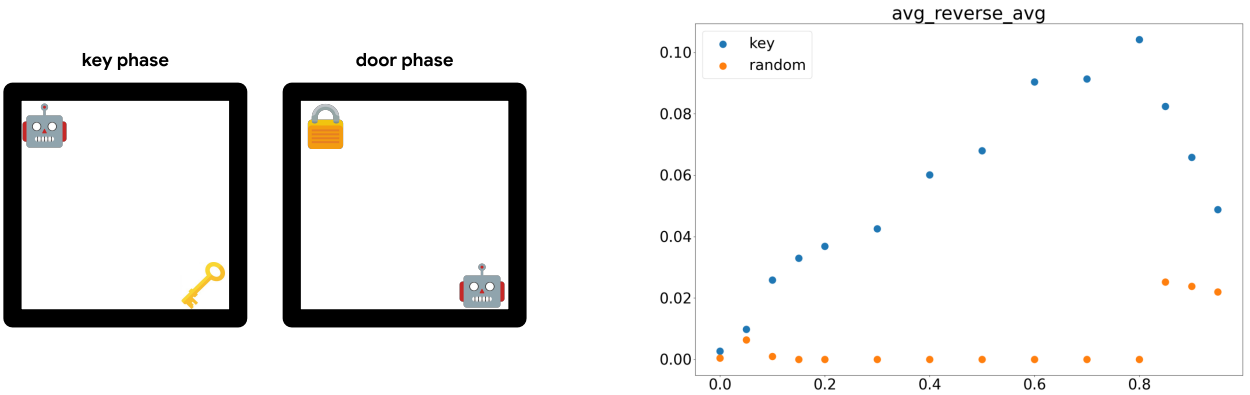

Figure 5: **Left:** The Key-to-Door environment. **Right:** the average credit assigned by DisCA for various policies. The x-axis corresponds to the mixture parameter α (controlling the mixture between a random and an optimal policy), the y-axis to the amount of credit assigned. Blue dots correspond to grabbing the key, orange to random actions.

actions (*i.e.* grabbing the key) and outcomes (*i.e.* opening the door). In Fig. 5 we display the average credit assigned to grabbing the key (in blue) and other, randomly sampled actions in randomly sampled states (in orange). We repeat the procedure with several distinct policies generating the training data for DisCA, parameterized by α such that $\pi_{\text{generator}} = \alpha\pi_{\text{random}} + (1-\alpha)\pi^*$. We show that there is a clear distinction between both types of actions in all settings except the hardest one, *i.e.* α close to 0. In that setting, the agent is close to optimal, which means that the agent almost always manages to open the door and collect rewards, which makes the discrimination task difficult. These results show that DisCA is robust to varying degrees of diversity in the training data.

5. Discussion

In this work, we presented DisCA, a simple yet promising approach for improved temporal credit assignment in RL using offline data. Central to DisCA, a neural network discriminator learns to distinguish observed actions from fictitious actions when possible, conditionally to a specified outcome. Actions that are crucial to reaching the future outcome are identified with a high confidence and can then be reinforced directly. All in all, DisCA is a straightforward approach to the temporal credit assignment problem that is conceptually simple, and has the potential to scale to sequences of actions.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, et al. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, 2019.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, 2020a.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, et al. Never give up: Learning directed exploration strategies. *International Conference on Learning Representations*, 2020b.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Johan Ferret, Raphaël Marinier, Matthieu Geist, and Olivier Pietquin. Self-attentional credit assignment for transfer in reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2020.

Tanmay Gangwani, Yuan Zhou, and Jian Peng. Learning guidance rewards with trajectory-space smoothing. In *Advances in Neural Information Processing Systems*, 2020.

Nathan Grinsztajn, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. There is no turning back: A self-supervised approach for reversibility-aware rl. In *Advances in Neural Information Processing Systems*, 2021.

Arthur Guez, Fabio Viola, Théophane Weber, Lars Buesing, Steven Kapturowski, Doina Precup, David Silver, and Nicolas Heess. Value-driven hindsight modelling. In *Advances in Neural Information Processing Systems*, 2020.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, and Remi Munos. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, 2019.

Matt Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

Ronald A Howard. Dynamic programming and markov processes. 1960.

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. *Nature Communications*, 2019.

Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019.

Kuang-Huei Lee, Ian Fischer, Anthony Liu, Yijie Guo, Honglak Lee, John Canny, and Sergio Guadarrama. Predictive information accelerates learning in RL. In *Advances in Neural Information Processing Systems*, 2020.

Yang Liu, Yunan Luo, Yuanyi Zhong, Xi Chen, Qiang Liu, and Jian Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420*, 2019.

Thomas Mesnard, Théophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Tom Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. In *International Conference on Machine Learning*, 2021.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, 1999.

Martin L. Puterman. *Markov Decision Processes*. 1994.

David Raposo, Sam Ritter, Adam Santoro, Greg Wayne, Theophane Weber, Matt Botvinick, Hado van Hasselt, and Francis Song. Synthetic returns for long-term credit assignment. *arXiv preprint arXiv:2102.12425*, 2021.

Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR 2016)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

Richard Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *Journal of Machine Learning Research*, 2016.

Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *International Conference on Machine Learning*, 2003.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Advances in neural information processing systems*, 2018.

Appendix A. Mathematical elements and proofs

A.1 Generalized advantage of a mixture of values

We provide a short proof that $A^{V_1+\alpha V_2}_{GAE,\lambda} = A^{V_1}_{GAE,\lambda} + \alpha A^{V_2}_{GAE,\lambda}$. We introduce the Bellman residual $\delta_t V = r_t + \gamma V(s_{t+1}) - V(s_t)$. We recall that the generalized advantage can be calculated as $A^{V}_{GAE,\lambda}(s_t, a_t) = \sum_{k=0}^{+\infty} (\gamma\lambda)^k \delta_{t+k} V$ (Schulman et al., 2016). Due to the linearity of the Bellman residual, we thus get that the generalized advantage is also linear in V, which concludes the proof.

Appendix B. Implementation details

B.1 Delayed Catch

Delayed Catch is implemented as a 7x7, fully-observable grid with balls falling from uniformly sampled positions on the top of the screen. Each phase consists in a single ball falling and lasts 6 timesteps, for a total of 120 timesteps for whole episodes with 20 balls. The action space is discrete with a cardinality of 3 (moving left, staying put, moving right). The environment rewards, due to the introduced delay, are non-Markov, so it is best seen as a POMDP. We use a PPO agent (Schulman et al., 2017) with a recurrent core to deal with the partial observability. We use the Acme implementation (Hoffman et al., 2020) of PPO with default parameters. We explain the changes brought to implement the value separation in Appendix B.4.

B.2 Key-to-Door

The Key-to-Door environment is implemented as a two-phase environment: in the first phase the agent moves on a grid and can collect or not a key, in the second phase the agent can use this key to open a door that is the only source of rewards. In both phases observations come as 5x5 frames with objects reduced to a single colored pixel. We use a timeout of 7 timesteps for the key stage and 40 timesteps for the door stage. As in Delayed Catch, the environment rewards, due to the fact the agent does not access whether it got the key or not in the first phase, are non-Markov, so the environment is best seen as a POMDP.

B.3 DisCA

The observation embedding is created by forwarding normalized observations into two convolutional layers of kernel size 3x3, with 32 feature maps, and with respective strides of 2 and 1. Action embeddings are created by forwarding action scalars into a two-layer perceptron with 32 neurons per layer and ReLU nonlinearities. Outcome embeddings are created the same way. Both actions and outcomes are normalized so that they belong to $[0, 1]$. All embeddings are concatenated and fed to a two-layer perceptron, the first layer being a 64 neuron layer with a ReLU nonlinearity, the second being a 1 neuron layer with a sigmoid nonlinearity. We use Adam as the optimizer with a learning rate of $3 \cdot 10^{-4}$ and otherwise standard parameters. We use a simple heuristic to split interaction data among the two replay buffers of DisCA. Episodes are added to $\mathcal{D}_+$ if their return is superior to the average return so far. They are added to $\mathcal{D}$ otherwise.

B.4 PPO with value separation

For the intrinsic value, we use a neural network that is identical to the standard value network of PPO except that it does not share weights with the policy or extrinsic value networks. We use separate Adam optimizers for the standard PPO components and the intrinsic value weights. We use the same coefficient for the extrinsic and intrinsic value losses. We normalize the advantage after calculating the mixture of advantages (*i.e.* not separately normalizing the extrinsic and intrinsic advantages).