Log-Distributional Approach for Learning Covariate Shift Ratios

Author: Guillermo Bernárdez Gil

1st Advisor: Sergio Escalera
Departament de Matemàtiques i Informàtica
Universitat de Barcelona

2nd Advisor: Carles Gelada
Google Brain, Montreal

A thesis presented for the degree of
Master in Artificial Intelligence

Facultat d'Informàtica de Barcelona (FIB)
Escola Tècnica Superior d'Enginyeria (URV)
Facultat de Matemàtiques i Informàtica (UB)

September 25, 2019

Aknowledgements

Abstract

Temporal Difference Learning algorithms classically learn value functions (sum of discounted rewards) by solving for the fixed point that relates the value of one state with the expected value of the following states. Other TD learning algorithms have been used to learn *multiplicative value functions*, which also have a fixed point solution. An example of such a function is the ratio between stationary distributions of two policies. It has been shown that learning of these multiplicative value functions suffers from higher variance and looser convergence guarantees. One potential approach to solving this issues would be to learn in log space, turning the multiplicative value function into an additive one. Unfortunately, it is easy to show with Jensen's inequality that the exponential of the log fixed point does not correspond to the multiplicative fixed point.

Distributional RL has been used to learn the return distribution (return is the sum of future discounted rewards). Although it has been shown that it mainly plays an auxiliary task role for representation learning, we propose that distributional fixed points could play a much more fundamental role to learning non additive value functions. In the case of multiplicative value distributions, learning the additive fixed point in log space and exponentiating leads to the correct solution.

In particular, we will study this approach in the case of the learning process of the Covariate Shift Ratio, which defines a multiplicative value function.

Contents

1 Introduction

- Reinforcement Learning
  - Distributional Reinforcement Learning
  - Off-Policy Learning
  - Covariate Shift Ratio

Motivation and Goals

- Relevance of logarithmic Distributional RL
  - Dessign of Distributional Covariate Shift Ratio
  - Evaluation of the logarithmich approach within the Distributional CSR setting

Structure of the Report

2 Distributional Reinforcement Learning

2.1 Reinforcement Learning Setting

Let's begin by characterizing our RL framework.

We consider an agent interacting with an environment in the standard setting[8]: at each step t, the agent selects an action a_t based on its current state s_t, to which the environment responds with a reward r_t and then moves to the next state s_{t+1}. We model this interaction as a time-homogeneous Markov Decision Process $(\mathcal{S}, \mathcal{A}, r, P, \gamma)$, where

- $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively, we assume that both are finite, with $n := |\mathcal{S}|$;
- P is the transition kernel, $s_{t+1} \sim P(\cdot|s_t, a_t)$; the Markov assumption states that $P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(s_{t+1}|s_t, a_t)$;
- $r(s, a)$ represents the immediate reward given by the environment after taking action a being in state s. These rewards are considered to be sampled from the reward function $R(s, a)$, i.e. $r_t \sim R(s_t, a_t)$;
- γ is the discount factor

A policy π maps each state to a probability distribution over the action space, $a_t \sim \pi(\cdot|s_t)$. In addition, we combine the policy π and transition function P into a state-to-state transition function $P_\pi \in \mathbb{R}^{n\times n}$, whose entries are

$$P_\pi(s'|s) := Prob_\pi(s_{t+1} = s'|s_t = s) = \sum_{a\in\mathcal{A}} \pi(a|s)P(s'|s, a) \tag{2.1}$$

In particular, powers of P_π represent the transition function across different time-steps.

Given π, the action value function is defined as the expected sum of discounted rewards from a state-action pair by following the policy:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s, a_0 = a \right] \tag{2.2}$$

The Bellman's equation can be obtained from this expression:

$$\begin{aligned}
Q^\pi(s, a) &= \mathbb{E}\left[r(s, a)\right] + \gamma \mathop{\mathbb{E}}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_{t+1}, a_{t+1}) \middle| s_0 = s, a_0 = a \right] \\
&= \mathbb{E}\left[r(s, a)\right] + \gamma \sum_{s'} P(s'|s, a) \left(\mathop{\mathbb{E}}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_{t+1}, a_{t+1}) \middle| s_1 = s' \right] \right) \\
&= \mathbb{E}\left[r(s, a)\right] + \\
&\quad \gamma \sum_{s'} P(s'|s, a) \sum_{a'} \pi(a'|s') \left(\mathop{\mathbb{E}}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_{t+1}, a_{t+1}) \middle| s_1 = s', a_1 = a' \right] \right) \\
&= \mathbb{E}\left[r(s, a)\right] + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s')} \left[Q(s', a') \right]
\end{aligned} \tag{2.3}$$

Analogously for the state value function (considering $r_\pi(s) := \mathbb{E}_{a\sim\pi(\cdot|s)}[r(s,a)]$):

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\middle| s_0=s\right] \\ &= r_\pi(s) + \gamma\sum_a \pi(a|s)\sum_{s'}P(s'|s,a)\left(\mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_{t+1},a_{t+1})\middle| s_1=s'\right]\right) \\ &= r_\pi(s) + \gamma \mathop{\mathbb{E}}_{s'\sim P_\pi(\cdot|s)}\left[V^\pi(s')\right] \end{aligned} \tag{2.4}$$

Policy Evaluation vs. Control Setting

Typically, we can distinguish two different settings among -and within- RL algorithms. On the one hand, we may be interested in computing the value function, assuming the current policy π is fixed, so as to evaluate that policy (*policy evaluation*). On the other hand, we might want to actually improve the current policy (*control setting*). Both settings have been extensively analyzed (e.g. [5]) and have associated their corresponding Bellman operators.

In the context of *policy evaluation*, considering state-action value functions Q^π as vectors in $\mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, the *Bellman operator* $\mathcal{T}^\pi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is defined as

$$\mathcal{T}^\pi Q(s,a) := \mathbb{E}[r(s,a)] + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a), a'\sim\pi(\cdot|s')}\left[Q(s',a')\right] \tag{2.5}$$

Such operator is useful to describe the expected behaviour of popular learning algorithms (e.g. Q-learning) and satisfies some interesting and desirable properties[8]:

- It is a contraction mapping (see Appendix A.3)
- The process $Q_t = \mathcal{T}^\pi Q_{t-1}$, for some initial value Q_0, converges exponentially to Q^π as $t \to \infty$.

Regarding the *control setting*, where the goal is to improve the current policy π, a *Bellman optimality operator* $\mathcal{T}^* : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is defined as follows:

$$\mathcal{T}^* Q(s,a) := \mathbb{E}[r(s,a)] + \gamma \mathop{\mathbb{E}}_{s'\sim P(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} Q(s',a')\right] \tag{2.6}$$

As it was shown in [8], $\mathcal{T}^*$ is also a contraction mapping, and the expectation of $\{Q_t\}$, $Q_t = \mathcal{T}^* Q_{t-1}$, converges exponentially to a fixed point (in this case, the optimal value function Q^*).

2.2 Towards a Distributional Reinforcement Learning

Now that we have introduced the standard expected-based RL setting, we go a step further and present its distributional counterpart, which has become the focus of many RL research from the publication of [1].

The first key of that work is to redefine the RL framework from a distributional perspective; we are not dealing with expected values anymore, but with proper distributions. Bearing that in mind, they define the following distributional Bellman equation:

$$Z^\pi(s,a) \overset{D}{=} R(s,a) + \gamma Z^\pi(S',A'), \tag{2.7}$$

where

- $Z^\pi : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$ is the *value distribution*, a mapping from state-action pairs to distributions over returns by following policy π. Its expectation is the value Q^π

$$Q^\pi(s,a) := \mathbb{E}[Z^\pi(s,a)] \tag{2.8}$$

  We will also call it the *return distribution*.

- $(S', A') \in \mathcal{S} \times \mathcal{A}$ is the next state-action random variable: $S' \sim P(\cdot|s,a)$, $A' \sim \pi(\cdot|S')$

- $R(s,a) \in \mathcal{Z}$ is the random reward, or equivalently the reward function. Note that now we are dealing with it as an explicit random variable.

- $Z^\pi(S', A') \in \mathcal{Z}$ is the random return over the random next state-action following π. This notation implies that all possible next state-action pairs need to be considered as to generate this return distribution. Thus, $Z^\pi(S', A')$ may be seen as a mixture distribution of the distributions $Z^\pi(s', a')$ where s' and a' are sampled from (S', A'):

$$f_{Z^\pi(S',A')}(z) = \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') f_{Z^\pi(s',a')}(z) \tag{2.9}$$

  The expected value, using 2.8, can be then expressed as:

$$\begin{aligned}
\mathbb{E}[Z^\pi(S', A')] &= \int_{-\infty}^{\infty} z \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') f_{Z^\pi(s',a')}(z) dz \\
&= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') \int_{-\infty}^{\infty} z f_{Z^\pi(s',a')}(z) dz \\
&= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') \, \mathbb{E}[Z^\pi(s',a')] \\
&= \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')} \Big[Q^\pi(s',a') \Big]
\end{aligned} \tag{2.10}$$

Remark. A distributional equation $U \overset{D}{:=} V$ indicates that the random variable U is distributed according to the same law as V.

Note that we can easily recover the classical Bellman's equation 2.3 for the action value Q by using 2.8 and 2.10 when taking the expected value over its distributional version 2.7:

$$\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}[Z^\pi(s,a)] \\
&= \mathbb{E}[R(s,a)] + \gamma \, \mathbb{E}[Z^\pi(S', A')] \\
&= \mathbb{E}[r(s,a)] + \gamma \mathop{\mathbb{E}}_{s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s')} \Big[Q^\pi(s',a') \Big]
\end{aligned} \tag{2.11}$$

Finally, let's try to find out what actually the random return Z represents by expanding

its density function:

$$\begin{aligned} f_{Z^\pi(s,a)}(z) &= f_{R(s,a)+\gamma Z^\pi(S',A')}(z) \\ &= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') f_{R(s,a)+\gamma Z^\pi(s',a')}(z) \\ &= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') f_{R(s,a)+\gamma(R(s',a')+\gamma Z^\pi(S'',A''))}(z) \\ &= \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') \sum_{s''} P(s''|s',a') \sum_{a''} \pi(a''|s'') \\ &\quad f_{R(s,a)+\gamma R(s',a')+\gamma^2 Z^\pi(s'',a'')}(z) \\ &= \sum_{s_1} P(s_1|s_0,a_0) \sum_{a_1} \pi(a_1|s_1) \cdots \sum_{s_t} P(s_t|s_{t-1},a_{t-1}) \sum_{a_t} \pi(a_t|s_t) \\ &\quad f_{\sum_{i=0}^t \gamma^t R(s_t,a_t)}(z) \end{aligned} \tag{2.12}$$

where we have repeatedly used property 2 of mixture distributions. In addition, note that assuming independence between the random reward R of a certain state-action pair and the return distribution Z^π of the possible next state-action (which is NOT TRUE in general), we can rewrite it in terms of convolutions as

$$\begin{aligned} f_{Z^\pi(s,a)}(z) = \sum_{s_1} P(s_1|s_0,a_0) \sum_{a_1} \pi(a_1|s_1) \cdots \sum_{s_t} P(s_t|s_{t-1},a_{t-1}) \sum_{a_t} \pi(a_t|s_t) \\ \left(f_{R(s_0,a_0)} * f_{\gamma R(s_1,a_1)} * \cdots * f_{\gamma^t R(s_t,a_t)}\right)(z) \end{aligned} \tag{2.13}$$

According to 2.12, $Z^\pi(s,a)$ can be interpreted as a convex combination of the sum of discounted reward distributions of all possible agent trajectories starting at the state-action (a,s) and following policy π from then on, each weight corresponding to the probability of that precise trajectory. It is important to note that Z^π encodes the intrinsic randomness of the agent's interactions with its environment; we should avoid considering it as a measure of uncertainty about the environment itself.

Moving to the *policy evaluation* setting, now we are interested in studying the behaviour of a distributional version of the policy evaluation operator $\mathcal{T}^\pi$. First we define the transition operator $P^\pi : \mathcal{Z} \to \mathcal{Z}$

$$P^\pi Z(s,a) \overset{D}{:=} Z(S',A') \tag{2.14}$$

Given this, the *distributional Bellman operator* $\mathcal{T} : \mathcal{Z} \to \mathcal{Z}$ is defined as

$$\mathcal{T}^\pi Z(s,a) \overset{D}{:=} R(s,a) + \gamma P^\pi Z(s,a) \tag{2.15}$$

We emphasize that three sources of randomness are involved in the compound distribution $\mathcal{T}^\pi Z$, i.e.

1. Randomness in the reward R,
2. Randomness in the transition P^π, and
3. Randomness in the next state-value distribution $Z(S',A')$,

which together make this distributional Bellman operator fundamentally different to the expected value-cased one (Equation 2.5). Authors in [1] demonstrate, under the assumption

that these three random quantities are independent, that $\mathcal{T}^\pi$ is a contraction mapping whose unique fixed point is the value distribution Z^π.

However, more difficulties arise in the *control setting* when dealing with this distributional perspective; as stated in [1], while every optimal policy attain the same value Q^* in the expected-valued case, there might be many optimal value distribiutions. Considering the set Π^* of optimal policies, an optimal value distribution is defined as the value distribution of an optimal policy. Hence, the set of optimal value distributions is $\mathcal{Z}^* := \{Z^{\pi^*} : \pi^* \in \Pi^*\}$.

Note that an optimal value distribution must match the full distribution of returns under some optimal policy, so that not all value distributions with expectation Q^* are optimal. A *distributional Bellman optimality operator* $\mathcal{T}$ is defined as well, but it doesn't behave as well as the policy evaluation operators: it is not a contraction in any usual metric between distributions, and its convergence to the set of optimal value distributions is weak.

2.3 Approximation Framework in the Distributional Setting

We should we aware that the full computation of the distributional Bellman operators on return distribution functions is generally either impossible (as we typically do not have access to the MDP dynamics, but to merely sample transitions) or infeasible (since the value distribution cannot be stored exactly in the general case).

This lead us to the design of several key approximations which are required to implement practical and scalable distributional RL algorithms[12]:

Distribution Parametrisation

Due to the fact that the full space of probability distributions, $\mathscr{P}(\mathbb{R})$, cannot be algorithmically encoded with a finite number of parameters, we need to approximate the distribution throughout a parametric family $\mathcal{P} \in \mathscr{P}(\mathbb{R})$.

Stochastic Bellman Operators

In order to evaluate the distributional Bellman operator $\mathcal{T}^\pi$, all possible next state-action-reward combinations should be taken into account. As in the expected valued case, the usual way of overcoming this practical limitation is by learning through transition samples (s, a, r, s', a') of the MDP. Hence, we can define a *stochastic distributional Bellman operator* $\widehat{\mathcal{T}}^\pi$ adapted to the randomness of these transitions, which defines a random measure whose behaviour is equal in expectation to the true Bellman operator $\mathcal{T}^\pi$.

Projection of Bellman Target Distribution

Another problem usually arises after computing the stochastic operator $\widehat{\mathcal{T}}^\pi$ over a value distribution $Z(s, a)$: the new distribution may no longer lie in the selected parametric family $\mathcal{P}$. If this is the case, we further need to apply a *projection operator* $\Pi : \mathscr{P}(\mathbb{R}) \to \mathcal{P}$ so as to map $\widehat{\mathcal{T}}^\pi Z(s, a)$ into the proper parametric family.

Gradient Updates

Finally, having computed a stochastic approximation $\widehat{Z}_k(s_k, a_k) = \Pi\widehat{\mathcal{T}}^{\pi} Z_k(s_k, a_k)$ to the full target distribution, we still have to define how to compute the next iterate Z_{k+1}. For that, the use of gradient updates seems to be appropiate[8], as it helps dissipate some noise introduced in the target by the stochastic approximation. A key aspect here, however, is the not-straightforward correspondence between the considered loss and the previously selected projection; convergence and good behaviour of the resulting algorithm have only be proven when both of them rely on the same norm-induced geometry[12, 8]

2.4 Categorical Distributional Reinforcement Learning

Although the distributional perspective is almost as old as Bellman's equations[13], it was not until the recent introduction of the Categorical Distributional Reinforcement[1] (CDRL) that it has become a central role within reinforcement learning. Their algorithm, called C51, was able to obtain state-of-the-art results in the Arcade Learning Environment[14] (ALE), outperforming the top expected-valued solutions by then.

The name 'categorical' comes from the distribution parametrisation that is used, which consists in the parametric family of categorical distributions over some fixed set of equally-spaced supports $z_1 < \cdots < z_K$:

$$\mathcal{P} = \left\{ \sum_{i=1}^{K} p_i \delta_{z_i} \middle| p_1, \ldots, p_K \geq 0, \sum_{k=1}^{K} p_k = 1 \right\} \tag{2.16}$$

Apart from that, the algorithm can be fully characterized by the rest of approximations implemented within the distributional RL framework described in the previous subsection:

- So as to learn from sampling transitions, we move to the *stochastic distributional Bellman operator* $\widehat{\mathcal{T}}^{\pi}$ and the *stochastic distributional Bellman optimality operator* $\widehat{\mathcal{T}}^{*}$. Given a sampled transition $(s_t, a_t, r, s_{t+1}, a_{t+1})$, these stochastic operators basically transform the supports of the distributions by an affine shift map $f_{r,\gamma} : \mathbb{R} \to \mathbb{R}$, defined by $f_{r,\gamma}(z) = r + \gamma z$; in our notation,

  $$\widehat{\mathcal{T}} Z_t(s_t, a_t) = (f_{r,\gamma})_{\#} Z_t(s_{t+1}, a_{t+1})$$

  with a_{t+1} either being selected by sampling the policy $\pi(\cdot|s_{t+1})$ (categorical policy evaluation, i.e. $\widehat{\mathcal{T}} = \widehat{\mathcal{T}}^{\pi}$), or being the action with the highest estimated expected returns (categorical Q-learning, $\widehat{\mathcal{T}} = \widehat{\mathcal{T}}^{*}$).

- CDRL applies the heuristic projection operator Π_C after computing the stochastic Bellman operators in order to recover a distribution within the selected parametric family P. This projection is defined for single Dirac measures as

  $$\Pi_C(\delta_y) = \begin{cases} \delta_{z_1} & y \leq z_1 \\ \frac{z_{i+1}-y}{z_{i+1}-z_i}\delta_{z_i} + \frac{y-z_i}{z_{i+1}-z_i}\delta_{z_{i+1}} & z_i < y < z_{i+1} \\ \delta_{z_K} & y > z_K \end{cases} \tag{2.17}$$

and can be easily extended to finite mixtures of Dirac measures:

$$\Pi_C \left(\sum_{i=1}^{N} p_i \delta_{y_i} \right) = \sum_{i=1}^{N} p_i \Pi_C(\delta_{y_i})$$

This projection step provides us with the target $\widehat{Z}_t(s_t, a_t) = \Pi_C \widehat{\mathcal{T}} Z_t(s_t, a_t)$.

- Finally, the original C51 performs a single step of gradient descent on the Kullback-Leibler divergence (see Appendix A.2 for the definition of this metric) of the prediction $Z_t(s_t, a_t)$ from the target $\widehat{Z}_t(s_t, a_t)$:

$$\mathrm{KL}\left(\widehat{Z}_t(s_t, a_t) || Z_t(s_t, a_t)\right)$$

with respect to the parameters of $Z_t(s_t, a_t)$; this gradient is used to generate the new estimate $Z_{t+1}(s_t, a_t) = \sum_{k=1}^{K} p_{t+1,k}(s_t, a_t)\delta_{z_k}$.

Pseudo-Algorithm 1 (provided by [12]) synthesises the steps performed by CDRL.

Algorithm 1: Categorical Distributional Reinforcement Learning

Require: $Z_t(s, a) = \sum_{k=1}^{K} p_{t,k}(s, a)\delta_{z_k}$ for each (s, a)
Input: A sample transition (s_t, a_t, r_t, s_{t+1})
`#Compute distributional Bellman target`
if Categorical Policy Evaluation **then**
$a^* \sim \pi(\cdot|s_{t+1})$
else if Categorical Q-learning **then**
$a^* \leftarrow \arg\max_a Q(s_{t+1}, a)$
end if
$\widehat{Z}_*(s_t, a_t) \leftarrow (f_{r_t,\gamma})_{\#} Z_t(s_{t+1}, a^*)$
`#Project target onto support`
$\widehat{Z}_t(s_t, a_t) \leftarrow \Pi_C \widehat{Z}_*(s_t, a_t)$
`#Compute KL loss`
Find gradient $\mathrm{KL}\left(\widehat{Z}_t(s_t, a_t) || Z_t(s_t, a_t)\right)$
Generate new estimate $Z_{t+1}(s_t, a_t) = \sum_{k=1}^{K} p_{t+1,k}(s_t, a_t)\delta_{z_k}$
Output: Estimate $Z_{t+1}(s, a)$ for each (s, a)

Despite the good results obtained by C51, all attempts to prove its convergence have failed so far. One of its most controversial steps, at least from a theoretical point of view, is the use of the KL divergence while applying the heuristic projection Π_C; as shown in [12], using the Cramér distance instead allows a useful geometrical reinterpretation of Π_C that lead us to the desired convergence results.

In fact, we recall that authors in [1] demonstrates, in the context of policy evaluation setting, that applying the Bellman operator T^π repeatedly to an initial return distribution function Z_0 guarantees convergence to the true set of return distributions Z^π in the supremum-Wasserstein metric. They did so by showing that

Lemma 2.1. *$\mathcal{T}^\pi : \mathcal{Z} \to \mathcal{Z}$ is a γ-contraction in $\bar{d}_p$*

However, after the introduction of the parametrization $\mathcal{P}$ and the projection operator Π_C, it is important to note that the operator to be analyzed is not the Bellman operator $\mathcal{T}^\pi$ itself anymore, but its composition with the projection operator, i.e. $\Pi_C \mathcal{T}^\pi$. As stated in [12], this can make contractivity break under all Wasserstein distances but $\bar{d}_1$:

Lemma 2.2. *$\Pi_C \mathcal{T}^\pi : \mathcal{Z} \to \mathcal{Z}$ is in general not a contraction in $\bar{d}_p$, for $p > 1$*

In contrast to that, Cramér distance seems to induce a useful geometric structure on the space of probability measures, allowing us to establish contractivity of the combined operator $\Pi_C \mathcal{T}^\pi$ in a much more natural way. We dedicate the following two sections to analyze CDRL algorithms together with Cramér-based metrics in the case of both Tabular Representations and Linear Function Approximations, respectively, which are the theoretical frameworks where stable behaviour and convergence properties are proven so far.

2.4.1 Tabular Representation

Our first approach to study Cramér-based CDRL algorithms will be to consider the simplest reinforcement learning framework, where we are able to store an approximate parametrization distribution for each state-action pair: the so-called tabular case.

Again, we will consider the parametric family $\mathcal{P}$ of categorical distributions over some fixed support $\{z_1, \dots, z_K\}$, as well as the heuristic projection operator Π_C defined in Equation 2.17 for mapping the backup distribution function $\mathcal{T}^\pi Z$ into $\mathcal{P}$. However, for the sake of simplicity and understandability, we will assume that no stochastic approximation is required.

First of all, we present the result of [12] that motivates the use of Cramér distance, which shows an interesting connection between this distance and the projection operator Π_C:

Proposition 2.3. *The Cramér metric ℓ_2 endows a particular subset of $\mathscr{P}(\mathbb{R})$ with a notion of orthogonal projection, and the orthogonal projection onto the subset $\mathcal{P}$ is exactly the heuristic projection Π_C. Consequently, Π_C is a non-expansion with respect to ℓ_2.*

The previous proposition directly lead us to the contractivity of the operator $\Pi_C \mathcal{T}^\pi$, which in turn ensures its convergence in the absence of stochastic approximation:

Proposition 2.4. *The operator $\Pi_C \mathcal{T}^\pi$ is a $\sqrt{\gamma}$-contraction in $\bar{\ell}_2$. Further, there is a unique distribution function $Z_C \in \mathcal{P}^{\mathcal{S}\times\mathcal{A}}$ to which the process $Z_{k+1} := \Pi_c \mathcal{T}^\pi Z_k$, given any initial distribution function $Z_0 \in \mathscr{P}(\mathbb{R})^{\mathcal{S}\times\mathcal{A}}$, converges in $\bar{\ell}_2$ as $k \to \infty$.*

Authors in [12] also address the natural question of how the limiting distribution function Z_C can differ from the true return distribution Z^π:

Proposition 2.5. *Let Z_C be the limiting return distribution of Proposition 2.4. If for all state-action pairs their corresponding true return distribution Z^π is supported on $[z_1, z_K]$, then*

$$\bar{\ell}_2(Z_C, Z^\pi) \leq \frac{1}{1-\gamma} \max_{1\leq i\leq K} (z_{i+1} - z_i).$$

Note that the true return distribution is gradually recovered as the fineness of the support increases. Hence, this result can be viewed as a way of quantifying the cost of using the parametrization $\mathcal{P}$ instead of fully non-parametric probability distributions. Finally, we also highlight the assumption that the support of the true return distributions lie on the selected support, since we don't always have access to the scale of rewards in every RL problem; for these circumstances, an analogous result can be found in Proposition 4 of [12].

2.4.2 Linear Function Approximation

In this second and last theoretical approach, we go a step beyond the tabular case by considering a framework where we make use of function approximators to compute the parametrized return distributions of each state-action pair, assuming a more realistic situation in which the state space is too large to store individual distributions.

We will restrict our analysis to the context of linear function approximation, whose convergence is already proven in [11]. To the best of our knowledge, no Distributional RL algorithm with a non-linear function approximator has been proven to converge so far, despite the many efforts in studying C51.

Regarding the rest of approximations and assumptions of the CDRL algorithms considered in this study, they are exactly the same as in the previous Section 2.4.1: we contemplate the parametric family $\mathcal{P}$ with fixed support $\{z_1, \dots, z_K\}$, the Cramér-based projection Π_C, and reject any stochastic approximation.

In our linear model, we now redefine the approximated return distributions Z as mappings from states $s \in \mathcal{S}$ to vectors defined by a linear combination of features:

$$Z_\Theta(s) := \Theta^\top \phi(s) \tag{2.18}$$

where $\phi(s) \in \mathbb{R}^m$ is the feature vector at state s and $\Theta \in \mathbb{R}^{n \times K}$ represents the weight matrix. Therefore, we are assuming that the vector $Z_\Theta(s) \in \mathbb{R}^K$ is an estimation of a distribution over the support $\{z_1, \dots, z_K\}$, although it might have negative components and it is not necessarily normalized.

Notation. In vector notation, we will write $Z_\Theta = \Phi\Theta \in \mathbb{R}^{n \times K}$, where $\Phi \in \mathbb{R}^{n \times m}$ is the feature matrix, simply formed by all feature vectors.

Before continuing, let's carefully review the three different spaces of distributions-like objects that we face in CDRL with linear function approximation:

- On the one hand, we have the probability space where true return distributions lie; without any prior knowledge, we should simply consider the full space of distributions $Z^\pi \in \mathscr{P}(\mathbb{R})$. Assuming that we know *a priori* the scale of the rewards, we can reduce it to the space of distributions with support in a certain interval; in our case, it could be $Z^\pi \in \mathscr{P}([z_1, z_K])$.

- On the other hand, the selected parametric family $\mathcal{P}$ of categorical distributions over the support $\{z_1, \dots, z_K\}$; so far, we have expected our estimated return distributions Z to belong to this space, $Z \in \mathcal{P}$.

- Finally, and in addition to the previous two, we now have a new vector space spanned by the features $\Phi \in \mathbb{R}^{n \times m}$, i.e. $\mathcal{L} := \{\Phi\Theta : \Theta \in \mathbb{R}^{m \times K}\}$. This is, in fact, the crucial space in our development, as every approximated return distribution $\mathcal{Z}_\Theta$ lies in it. Ideally, if all Z_Θ were proper probability distributions over $\{z_1, \dots, z_K\}$, we would have $\mathcal{L} \subset \mathcal{P}$ (assuming some loss of expressiveness by our linear model).

Let's review the process. First, our linear function model provides us with return distribution estimates

$$Z_\Theta \in \mathcal{L}$$

for each state $s \in \mathcal{S}$. In order to get better approximations, CDRL algorithm applies the distributional Bellman operator, so we get

$$\mathcal{T}^\pi Z_\Theta \in \mathscr{P}([z_1, z_K]).$$

Then we would need to apply a first projection operator $\Pi_1 : \mathscr{P}([z_1, z_K]) \to \mathcal{P}$ (like the Cramér projection Π_C) so as to recover a distribution within our selected parametric family, so

$$\Pi_1 \mathcal{T}^\pi Z_\Theta \in \mathcal{P}.$$

However, for the algorithm to compare distributions and improve its linear representations, now we need a further projection onto the space $\mathcal{L}$, which defines all the probability distributions that our linear model can actually express. Thus, the idea is that a second projection operator $\Pi_2 : \mathcal{P} \to \mathcal{L}$ is required so that

$$\Pi_2 \Pi_1 \mathcal{T}^\pi Z_\Theta \in \mathcal{L}.$$

As shown in [11], the (re)definition of both projections plays again a very important role for proving the convergence of the process $Z_{\Theta_{k+1}} = \Pi_2 \Pi_1 \mathcal{T}^\pi Z_{\Theta_k}$.

So far, we demonstrated in the tabular case that the combined operator $\Pi_C \mathcal{T}^\pi$ is a contraction mapping in the supremum-Cramér metric $\bar{\ell}_2$; our task now is to achieve a similar result for the combination $\Pi_2 \Pi_1 \mathcal{T}^\pi$. For doing so, authors in [11] introduce a *generalized Cramér distance* which is able to deal with our return distribution estimates (we recall they are not necessarily probability distributions).

We denote by $C \in \mathbb{R}^{K \times K}$ the lower-triangular matrix of ones, and by $e \in \mathbb{R}^K$ the vector of ones. Let $e_K = (1/\sqrt{K})e^\top$ and $\Pi_{e_K^\top} = I_K - e_K e_K^\top$. For any two discrete value distributions Z_1, Z_2 over the fixed support $\{z_1, \dots, z_K\}$, the generalized Cramér distance is defined as

$$\ell_\lambda^2(Z_1, Z_2) = (Z_1 - Z_2)^\top \Pi_{e_K^\top} C C^\top \Pi_{e_K^\top} (Z_1 - Z_2) + \lambda((Z_1 - Z_2)^\top e_K)^2 \tag{2.19}$$

We will simplify the notation by denoting $C_\lambda = \Pi_{e_K^\top} C C^\top \Pi_{e_K^\top} + \lambda e_K e_K^\top$, so we can rewrite the distance definition as

$$\ell_\lambda^2(Z_1, Z_2) = (Z_1 - Z_2)^\top C_\lambda (Z_1 - Z_2) = \|Z_1 - Z_2\|_{C_\lambda}$$

Looking at Equation 2.19, the idea is that the first term on the right-hand side penalizes the difference in cumulative probabilities of both distributions, whereas the second term accounts for the difference in mass; we refer to [11] to get more details.

But, in addition to the Cramér distance generalization, authors in [11] also take into account the distribution ξ according to which the states to be updated are sampled; they use it to define a *ξ-weighted Cramér distance* over value distributions as

$$\ell_{\xi,\lambda}^2(Z_1, Z_2) := \sum_{s \in \mathcal{S}} \xi(s) \ell_\lambda^2(Z_1(s), Z_2(s)). \tag{2.20}$$

The two presented distances are precisely the key to all further results shown in [11], and therefore to prove the desired convergence of CDRL algorithm with linear function approximation.

In particular, the previously commented projections of value distributions Π_1 and Π_2 are defined through ℓ_λ^2 and $\ell_{\xi,\lambda}^2$, respectively. The former, which we will denote by $\Pi_{\lambda,\mathcal{P}}$, projects any value distribution Z onto the subspace $\mathcal{P}$ according to

$$\Pi_{\lambda,\mathcal{P}} Z := \arg\min_{Z' \in \mathcal{P}} \ell_\lambda^2(Z, Z'); \tag{2.21}$$

it can be seen that the previously considered Cramér projection Π_C satifies this minimization expression[11]. The latter, which in our notation will be expressed as $\Pi_{\xi,\lambda,\Phi}$, performs a ξ-weighted projection of any value distribution Z onto the set of value distributions Φ by:

$$\Pi_{\xi,\lambda,\Phi} Z := \arg\min_{\Phi\Theta,\Theta\in\mathbb{R}^{m\times K}} \ell^2_{\xi,\lambda}(Z, \Phi\Theta) \tag{2.22}$$

The following Lemma 2.6 shows an intewresting result regarding the good behaviour of this projection $\Pi_{\xi,\lambda,\Phi}$

Lemma 2.6. *$\Pi_{\xi,\lambda,\Phi}$ is a non-expansion in $\ell^2_{\xi,\lambda}$; for every pair of return distributions Z, Z', we have*

$$\ell^2_{\xi,\lambda}(\Pi_{\xi,\lambda,\Phi} Z, \Pi_{\xi,\lambda,\Phi} Z') \leq \ell^2_{\xi,\lambda}(Z, Z')$$

At this point, before facing the proof of convergence of the whole CDRL algorithm, we would need to rewrite distances ℓ^2_λ and $\ell^2_{\xi,\lambda}$ into two separate components: along dimension e_K and along the subspace A orthogonal to e_K, $A := \Pi_{e_K^\top} C$:

$$\begin{cases} \ell^2_\lambda(Z_1, Z_2) & = \|Z_1 - Z_2\|^2_{AA^\top} + \lambda \|Z_1 - Z_2\|^2_{e_K e_K^\top} \\ \ell^2_{\xi,\lambda}(Z_1, Z_2) & = \|Z_1 - Z_2\|^2_{\xi,AA^\top} + \lambda \|Z_1 - Z_2\|^2_{\xi,e_K e_K^\top} \end{cases} \tag{2.23}$$

The reason relies in Lemma 2.7, which states that the combined operator $\Pi_{\lambda,\mathcal{P}} \mathcal{T}^\pi$ behaves differently in each of these dimensions: while contracting all dimensions orthogonal to e_K -i.e. the subspace A- by a factor $\gamma^{1/2}$, it is only a non-expansion along e_K.

Lemma 2.7. *Let d_π the stationary distribution induced by policy π. For any two return distributions Z_1, Z_2, we have*

$$\begin{array}{rcl} \|\Pi_{\lambda,\mathcal{P}} \mathcal{T}^\pi Z_1 - \Pi_{\lambda,\mathcal{P}} \mathcal{T}^\pi Z_2\|^2_{d_\pi,AA^\top} & \leq & \gamma \|Z_1 - Z_2\|^2_{d_\pi,AA^\top} \\ \|\Pi_{\lambda,\mathcal{P}} \mathcal{T}^\pi Z_1 - \Pi_{\lambda,\mathcal{P}} \mathcal{T}^\pi Z_2\|^2_{d_\pi,e_K E_K^\top} & \leq & \|Z_1 - Z_2\|^2_{d_\pi,e_K E_K^\top} \end{array}$$

Note that, instead of the more general notation ξ, we are now specifying the stationary distribution d_π (i.e. $d_\pi \in \Delta(\mathcal{S})$ such that $d_\pi = P_\pi d_\pi$; see a more detailed definition in Section 3.1). This is simply due to the fact that in our CDRL algorithm we are actually following policy π to describe the agent trajectories we are learning from.

Finally, we present the theorem of [11] that provides us with the desired convergence guarantee for CDRL with Linear Function Approximation:

Theorem 2.8. *Let d_π the stationary distribution induced by policy π. The process*

$$Z_0 := \Phi\Theta_0, \quad Z_{k+1} := \hat{\Pi}_{d_\pi,\lambda,\Phi} T^\pi Z_k$$

converges to a set S such that, for any two $Z, Z' \in S$, there is a $\mathcal{S}$-indexed vector of constants α such that

$$Z(s) = Z'(s) + \alpha(s) e_K.$$

If $\lambda > 0$, S consists of a single point $\hat{Z}$ which is the fixed point of the process. Moreover, we can bound the error of this fixed point with respect to the true return distribution Z^π by

$$\ell^2_{d_\pi,\lambda}(\hat{Z}, Z^\pi) \leq \frac{1}{1-\gamma} \ell^2_{d_\pi,\lambda}(\Pi_{d_\pi,\lambda,\Phi} Z^\pi, Z^\pi) - \frac{\gamma\lambda}{1-\gamma} \|\hat{Z} - Z^\pi\|_{d_\pi,e_K e_K^\top},$$

where the second terms measures the difference in mass between $\hat{Z}$ and Z^π.

Note, however, that in order to prove convergence Theorem 2.8 do not make use of the projection $\Pi_{d_\pi,\lambda,\Phi}$, but the operator $\hat{\Pi}_{d_\pi,\lambda,\Phi}$, which is based on the loss

$$\hat{\ell}^2_\lambda(Z_1, Z_2) = (Z_1 - Z_2)^\top \Pi_{e_K^\top} CC^\top \Pi_{e_K^\top}(Z_1 - Z_2) + \lambda(Z_2 e - 1)^2 \tag{2.24}$$

instead of on the ℓ^2_λ distance. In fact, we refer to $\hat{\ell}^2_\lambda$ as a loss because it is not a distance, as it contains the explicit normalization penalty $(Z_2 e - 1)^2$ that encourages return distributions to have unit mass. Given expression 2.24, the corresponding ξ-weighted Cramér loss $\hat{\ell}^2_{\xi,\lambda}$ is accordingly derived, and operator $\hat{\Pi}_{d_\pi,\lambda,\Phi}$ is defined so that, for a return distribution Z, it finds the value distribution in the span of Φ which minimizes $\hat{\ell}^2_{\xi,\lambda}(Z, \cdot)$:

$$\hat{\Pi}_{d_\pi,\lambda,\Phi} Z = \Phi\Theta^* \quad \text{where} \quad \Theta^* = \arg\min_\Theta \hat{\ell}^2_{\xi,\lambda}(Z, \Phi\Theta) \tag{2.25}$$

For bounding the approximation error we recover the well-defined distance $\ell^2_{d_\pi,\lambda}$, though.

As pointed out in [11], the parameter λ plays an important role in the theorem, both to guarantee convergence and to bound the approximation error. At a high level, this makes sense: a high value of λ forces the algorithm to output something close to a distribution, at the expense of actual predictions. On the other hand, taking $\lambda = 0$ yields a process which may not converge to a single point.

3 Covariate Shift Ratio

3.1 Off-Policy Learning Setting

As stated in [3], here we move to the *policy evaluation* problem within *off-policy learning*, where we want to learn the value function V^π of a *target policy* π from samples drawn from P and a *behaviour policy* μ.

We first introduce some useful notation:

- The Bellman equation for the state value function can be expressed in vector notation as $V^\pi = r_\pi + \gamma P_\pi V^\pi$, where $V^\pi \in \mathbb{R}^n$, $r_\pi \in \mathbb{R}^n$ and $P_\pi \in \mathbb{R}^{n\times n}$. The value function is in fact the fixed point of the *Bellman operator* $\mathcal{T}_\pi : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, defined as $\mathcal{T}_\pi V := r_\pi + \gamma P_\pi V$. It defines a single step of *bootstrapping*: the process $V^{k+1} := \mathcal{T}_\pi V^k$ converges to V^π.
- Let $d \in \mathbb{R}^n$; we write $D_d \in \mathbb{R}^{n\times n}$ for the corresponding diagonal matrix, and consider the weighted squared seminorm notation of vectors $x \in \mathbb{R}^n$ $||x||_A^2 := ||Ax||^2 = x^T A^T A x$, $||x||_d^2 := ||x||_{D_d}^2 = \sum_{i=1}^n d(i)^2 x(i)^2$.
- $e \in \mathbb{R}^n$ accounts for the vector of all ones, and $\Delta(\mathcal{S})$ for the simplex over states: $d \in \Delta(\mathcal{S}) \implies d^T e = 1, d \geq 0$.
- $d \in \Delta(\mathcal{S})$ is the stationary distribution of a transition function P if and only if $d = d \cdot P$. This distribution is unique when P defines a Markov chain with a single recurrent class[4].

In this particular setting we distinguish between two different state-to-state transition functions, P_π and P_μ, one for each policy; their respective stationary distributions will be represented by d_π and d_μ.

Linear Function Approximation

In most RL problems, it is unfeasible to learn the value of each state separately due to the large state spaces involved; in these cases, we get value function estimates through *function approximators*. Among them, it is common to consider a *linear function approximation*, which uses a mapping from states to features $\phi : \mathcal{S} \to \mathbb{R}^k$. In these cases, the approximate value function at a state s can be expressed as the inner product of a feature vector with a vector of weights $\theta \in \mathbb{R}^k$:

$$\hat{V}(s) = \phi(s)^T \theta \tag{3.1}$$

which can be written as $\hat{V} = \Phi\theta$ in vector notation, being $\Phi \in \mathbb{R}^{n\times k}$ the matrix of row vectors. The *semi-gradient update rule* for TD learning[5] learns an estimation of V^π from sample transitions. Given a starting state $s \in \mathcal{S}$, a successor state $s' \sim P_\pi(\cdot|s)$, and a step-size parameter $\alpha > 0$, this update is

$$\theta \leftarrow \theta + \alpha \left[r_\pi(s) + \gamma \phi(s')^T \theta - \phi(s)^T \theta \right] \phi(s) \tag{3.2}$$

The expected behaviour of this update rule is described by the *projected Bellman operator* $\Pi_d \mathcal{T}_\pi$, a combination of the usual Bellman operator with a projection Π_d in norm $||\cdot||_d$ -for some $d \in \Delta(\mathcal{S})$- onto the span of Φ[6]. In fact, the stationary point of 3.2, if it exists, is the solution of the *projected Bellman equation* $\hat{V}^\pi = \Pi_d \mathcal{T}_\pi \hat{V}^\pi$. If the projection is performed

under d_π, we can guarantee convergence to a fixed point[8]; otherwise -i.e. $d \neq d_\pi$- it may lead to divergence.

We want to note that the proof of convergence when $d = d_\pi$ is done by showing that the combined operator $\Pi_{d_\pi} T^\pi$ is a contraction in the d_π-weighted norm, i.e. for any two value functions $V, V' \in \mathbb{R}^n$

$$\|\Pi_{d_\pi} T^\pi V - \Pi_{d_\pi} T^\pi V'\|_{d_\pi} \leq \gamma \|V - V'\|_{d_\pi}$$

From this, it only remains to apply Banach's fixed point theorem. Furthermore, as presented in [3], not only $d = d_\pi$ provides convergence, but this choice also seems optimal in terms of the quality of the fixed point under off-policy data with linear function approximation.

3.2 Covariate Shift Approach

Supposing that stationary distributions d_π and d_μ are known, and that states are updated according to $s \sim d_\mu$, the covariate shift approach presented in [2] uses importance sampling to redefine 3.2 so that the semi-gradient update rule can be considered *under the sampling distribution* d_π:

$$\theta \leftarrow \theta + \alpha \frac{d_\pi(s)}{d_\mu(s)} \left[r(s,a) + \gamma \phi(s')^T \theta - \phi(s)^T \theta \right] \phi(s) \tag{3.3}$$

with $a \sim \mu(\cdot|s)$, $s' \sim P(\cdot|s,a)$ as before.

Hence, the Consistent Off-Policy Temporal Difference[2] (COP-TD) algorithm seek to learn that covariate shift ratio d_π/d_μ from samples by bootstrapping from a previous prediction, similar to temporal difference learning. Given a step size $\alpha > 0$, a ratio vector $c \in \mathbb{R}^n$ and a sample transition $(s_t, a_t, s_{t+1}) = (s, a, s')$ drawn from d_μ, $\mu(\cdot|s)$ and $P(\cdot|s,a)$, respectively, the COP-TD update is

$$c(s') \leftarrow c(s') + \alpha \left[\frac{\pi(a|s)}{\mu(a|s)} c(s) + c(s') \right] \tag{3.4}$$

The expected behaviour of this learning rule, which learns "in reverse" compared to TD learning, is captured by the *COP operator* Y:

$$(Yc)(s') := \mathop{\mathbb{E}}_{s \sim d_\mu, a \sim \mu(\cdot|s)} \left[\frac{\pi(a|s)}{\mu(a|s)} c(s) \middle| s' \right] \tag{3.5}$$

Note that the condition $s_{t+1} = s'$ in the expectation of 3.5 forces to take into account the distribution of previous state-action pairs (s,a) according to policy μ. The distribution of the possible previous states s is given by the time-reversal transition function $\bar{P}_\mu$, whose entries are:

$$\begin{aligned} \bar{P}_\mu(s|s') &:= Prob_\mu(s_t = s | s_{t+1} = s') \\ &= \frac{Prob_\mu(s_{t+1} = s' | s_t = s) Prob_\mu(s_t = s)}{Prob_\mu(s_{t+1} = s')} \\ &= \frac{P_\mu(s'|s) d_\mu(s)}{d_\mu(s')} \end{aligned} \tag{3.6}$$

Or, equivalently, $\bar{P}_\mu = D_{d_\mu}^{-1} P_\mu^T D_{d_\mu}$ in vector notation. Regarding the distribution of the possible actions that lead to s' from a certain state s by following policy μ, it will be represented

by function $\bar{\mu}$:

$$\begin{aligned}\bar{\mu}(a|s,s') &:= Prob_\mu(a_t = a|s_t = s, s_{t+1} = s') \\ &= \frac{Prob_\mu(a_t = a, s_t = s, s_{t+1} = s')}{Prob_\mu(s_t = s, s_{t+1} = s')} \\ &= \frac{Prob_\mu(s_{t+1} = s'|a_t = a, s_t = s)Prob_\mu(a_t = a|s_t = s)Prob_\mu(s_t = s)}{Prob_\mu(s_{t+1} = s'|s_t = s)Prob_\mu(s_t = s)} \\ &= \frac{P(s'|s,a)\mu(s|a)}{P_\mu(s'|s)}\end{aligned} \tag{3.7}$$

Bearing in mind the introduced notation, the expectation in 3.5 can be rewritten and expanded:

$$\begin{aligned}\mathop{\mathbb{E}}_{s\sim d_\mu, a\sim\mu(\cdot|s)}\left[\frac{\pi(a|s)}{\mu(a|s)}c(s)\middle|s'\right] &= \mathop{\mathbb{E}}_{s\sim\bar{P}_\mu(\cdot|s'), a\sim\bar{\mu}(\cdot|s,s')}\left[\frac{\pi(a|s)}{\mu(a|s)}c(s)\right] \\ &= \sum_s \bar{P}_\mu(s|s')\sum_a \bar{\mu}(a|s,s')\frac{\pi(a|s)}{\mu(a|s)}c(s) \\ &= \sum_s \left(\frac{P_\mu(s'|s)d_\mu(s)}{d_\mu(s')}\right)\sum_a \left(\frac{P(s'|s,a)\mu(s|a)}{P_\mu(s'|s)}\right)\frac{\pi(a|s)}{\mu(a|s)}c(s) \\ &= \frac{1}{d_\mu(s')}\sum_s d_\mu(s)c(s)\sum_a \pi(a|s)P(s'|s,a) \\ &= \frac{1}{d_\mu(s')}\sum_s P_\pi(s'|s)d_\mu(s)c(s)\end{aligned} \tag{3.8}$$

Thus, according to the last term of 3.8, the COP operator Y can be expressed in vector notation as

$$Yc = D_{d_\mu}^{-1}P_\pi^T D_{d_\mu}c \tag{3.9}$$

Regarding the behaviour of the operator, the following result of [3] guarantees that the process $c^{k+1} := Yc^k$ converges, and that any multiple of d_π/d_μ is a fixed point:

Theorem 3.1. *Suppose that P_π defines an ergodic Markov chain on $\mathcal{S}$, and let $c^0 \in \Delta(\mathcal{S})$. Then the process $c^{k+1} := Yc^k$ converges to $C\frac{d_\pi}{d_\mu}$, $C \in \mathbb{R}^+$*

Furthermore, we can reduce the set of fixed points to the covariate shift ratio exclusively by considering a normalized version of the COP operator:

Corollary 3.2. *Suppose conditions of Theorem 3.1 are satisfied. Let*

$$(\bar{Y}c)(s') := \frac{(Yc)(s')}{\sum_s (Yc)(s)}$$

be the normalized COP operator. Then the unique fixed point of $\bar{Y}$ is the ratio d_π/d_μ, to which $c^{k+1} = \bar{Y}c^k$ converges.

COP-TD with Linear Function Approximation

Whenever the value function is approximated, we should expect to learn a ratio $\hat{c}$ estimate as well. In our analysis, we consider a linear approximation of the form

$$\hat{c}(s) = \phi(s)^\top w$$

where again ϕ defines a map from states to features, $\phi : \mathcal{S} \to \mathbb{R}^k$, and now $w \in \mathbb{R}^k$ is the vector of weights we are interested in learning. Note that in such a model negative ratio values are allowed; this can be avoid in practice by clipping those values at zero.

The introduced linear model, given a sample transition (s, a, s') drawn from d_μ, $\mu(\cdot|s)$ and $P(\cdot|s, a)$, respectively, induces the following semi-gradient update

$$\tilde{w} \leftarrow w + \alpha \left[\frac{\pi(a|s)}{\mu(a|s)} \phi(s)^\top w - \phi(s')w \right] \phi(s'), \tag{3.10}$$

which can be thought as a d-weighted projection Π_d for some $d \in \Delta(\mathcal{S})$[3]. However, an additional step is required for granting that the resulting ratio estimate $\hat{c}$ corresponds to some proper distribution ratio d/d_μ for that $d \in \Delta(\mathcal{S})$, just as we want to. This is solved in [2] by following the update rule by a projection onto the d_μ-weighted simplex Δ_{Φ,d_μ}

$$w \leftarrow \arg\min_{u \in \mathcal{W}_{\Phi,d_\mu}} \|u - \tilde{w}\| \tag{3.11}$$

where $\mathcal{W}_{\Phi,d_\mu} := \{u \in \mathbb{R}^k : \sum_{s\in\mathcal{S}} d_\mu(s)\phi(s)^\top u = 1, \phi(s)^\top u \geq 0\}$. Looking at the definition of $\mathcal{W}_{\Phi,d_\mu}$, we can identify this second projection $\Pi_{\Delta_{\Phi,d_\mu}}$ as a normalization step as well.

The following Lemma 3.3 of [3] shows that, in fact, the normalization component of operator $\Pi_{\Delta_{\Phi,d_\mu}}$ is not only a convenience but also a requisite to get a good convergence guarantee.

Lemma 3.3. *Let Y be a symmetric COP-TD operator and Π the projection onto Φ in L_2 norm. If d_π/d_μ is not in the span of Φ, then $c = 0$ is the only fixed point of the process $c^{k+1} = \Pi Y c^k$.*

Hence, in order to get a meaningful approximation ratio, we must consider the repeated application of the combined operator $\Pi_{\Delta_{\Phi,d_\mu}}\Pi_d Y$, so that iterating $\hat{c}^{k+1} := \Pi_{\Delta_{\Phi,d_\mu}}\Pi_d Y \hat{c}^k$ provides the estimate.

In practice, however, that combination of the COP operator with the projection onto the d_μ-weighted simplex can be really hard to implement; authors in [2] presented a method for approximating the projection step in an online, sample-based setting for linear function approximation, but to the best of our knowledge there are no analogous implementation designs for other kind of function approximations, like neural networks.

3.3 Discounted COP-TD

Despite the good properties shown of COP-TD, there are two practical limitations of the algorithm that specially motivated the authors in [3] to work in an improvement. On the one hand, we have the previously commented difficulties to practically implement the projection operator $\Pi_{\Delta_{\Phi,d_\mu}}$ with function approximations other than linear ones. On the other hand, we note that the COP operator Y lacks a result ensuring it is a contraction mapping, which can make the algorithm converge at a slow rate or with high variance, or even be unstable together with function approximation.

They address these two issues in [3] through the *$\hat{\gamma}$-discounted COP-TD* learning rule; given a step size $\alpha > 0$, a ratio vector $c \in \mathbb{R}^n$ and a sample transition $(s_t, a_t, s_{t+1}) = (s, a, s')$ drawn from d_μ, $\mu(\cdot|s)$ and $P(\cdot|s, a)$, respectively, it modifies the previous COP-TD update 3.4 by

$$c(s') \leftarrow c(s') + \alpha \left[\hat{\gamma} \frac{\pi(a|s)}{\mu(a|s)} c(s) + (1 - \hat{\gamma}) - c(s') \right] \tag{3.12}$$

Again, the expected behaviour of this rule is captured by the corresponding $\hat{\gamma}$-discounted COP operator $Y_{\hat{\gamma}}$

$$(Y_{\hat{\gamma}}c)(s') := \mathop{\mathbb{E}}_{s\sim d_\mu, a\sim\mu(\cdot|s)}\left[\hat{\gamma}\frac{\pi(a|s)}{\mu(a|s)}c(s) + (1-\hat{\gamma})\middle| s'\right] = \hat{\gamma}(Yc)(s') + (1-\hat{\gamma}), \tag{3.13}$$

where Y is the COP operator (3.5). In vector notation we have

$$Y_{\hat{\gamma}}c = \hat{\gamma}D_{d_\mu}^{-1}P_\pi^T D_{d_\mu}c + (1-\hat{\gamma})e \tag{3.14}$$

From the above expressions, it is straightforward to notice that we recover COP-TD for $\hat{\gamma} = 1$, i.e. $Y_1 = Y$.

So as to characterize the discounted COP operator authors in [3] introduce the discounted reset transition function $\hat{P}_\pi$, which for a given $\hat{\gamma} \in [0, 1]$ is defined as

$$\hat{P}_\pi := \hat{\gamma}P_\pi + (1-\hat{\gamma})ed_\mu^\top,$$

denoting by $\hat{d}_\pi$ its stationary distribution (i.e. $\hat{d}_\pi = \hat{d}_\pi\hat{P}_\pi$). We can interpret that $\hat{P}_\pi$ encodes a stochastic process in which either transitions occur as usual with probability $\hat{\gamma}$, or resets to the stationary distribution d_μ with the remainder probability. Lemma 3.4 gives us more insights about how operator $Y_{\hat{\gamma}}$ is intimately connected to this process:

Lemma 3.4. *For $\hat{\gamma} < 1$, the ratio $\hat{d}_\pi/d_\mu$ is the unique fixed point of the operator $Y_{\hat{\gamma}}$, where $\hat{d}_\pi$ is the stationary distribution of the transition function $\hat{P}_\pi$ corresponding to the given $\hat{\gamma}$*

So, basically, the discounted reset transition function encodes the transition dynamics associated to the use of the DCOP operator. Furthermore, as shown in the following result of [3], we can also get convergence guarantees for the repeated application of $Y_{\hat{\gamma}}$ for $\hat{\gamma} < 1$ without requiring positive initial values or any normalization.

Theorem 3.5. *Given $\hat{\gamma} < 1$, the process $c^{k+1} := Y_{\hat{\gamma}}c^k$ converges to $\hat{d}_\pi/d_\mu$ for any $c^0 \in \mathbb{R}^n$.*

DCOP-TD with Linear Function Approximation

Now it is time to analyze how DCOP-TD algorithm behaves when it is combined with (linear) function approximation., and in particular whether it provides us with good convergence guarantees in a practical online application.

As in COP-TD, we consider a linear ratio estimate $\hat{c}(s) = \phi(s)^\top w$, and assume that sample transitions (s, a, s') are drawn from d_μ, $\mu(\cdot|s)$ and $P(\cdot|s, a)$, respectively. This induces the semi-gradient update

$$\tilde{w} \leftarrow w + \alpha\left[\left(\hat{\gamma}\frac{\pi(a|s)}{\mu(a|s)}\phi(s)^\top w + (1-\hat{\gamma})e\right) - \phi(s')w\right]\phi(s'), \tag{3.15}$$

which again can be interpreted as a d-weighted projection for some $d \in \Delta(\mathcal{S})$. Nevertheless, now the additional, hard-to-implement projection-normalization step is not required for the process to converge. In this context, authors in [3] argue that $s' \sim d_\mu$ -since d_μ is the stationary distribution that ens up ruling the sampled agent trajectories-, and therefore that the induced d-weighted projection should be in fact considering $d = d_\mu$. As a result, the process we are analyzing can be described by the projected DCOP operator

$$\Pi_{d_\mu}Y_{\hat{\gamma}}$$

Before presenting any convergence result, however, we need to introduce the concentration coefficient $K_{\pi,\mu,n}$ defined in Lemma 3.6; at a high level, it measures the discrepancy in stationary distributions -$d\pi$ and d_μ- between a pair of states that can be considered 'close' according to policy π, in the sense that one of these states is reachable from the other in n steps; it simplifies to 1 when $\pi = \mu$.

Lemma 3.6. *The induced operator norm of the COP operator Y^n is upper bounded by a constant $\sqrt{K_{\pi,\mu,n}}$ in the sense that*

$$\|Y^n\|_{d_\mu}^2 \leq K_{\pi,\mu,n} := \sup_{s' \in \mathcal{S}} \sum_{s \in \mathcal{S}} \frac{d_\mu(s)}{d_\mu(s')} P_\pi^n(s'|s).$$

Further, the series can be bounded by a constant,

$$K_{\pi,\mu,n} \leq K_{\pi,\mu} := \left\| \frac{d_\mu(s)}{d_\pi(s)} \right\|_\infty \left\| \frac{d_\pi(s)}{d_\mu(s)} \right\|_\infty$$

Theorem 2.8 provides us with the reason why this concentration coefficient is required in our study: authors in [3] make use of it to obtain a safe value of $\hat{\gamma}$ below which the DCOP operator is a contraction mapping, and from this they can prove convergence of the combined operator $\Pi_{d_\mu} Y_{\hat{\gamma}}$ in the linear function approximation case.

Theorem 3.7. *Consider the n-step DCOP operator $\hat{Y}_{\hat{\gamma}}$. For any $c \in \mathbb{R}^n$,*

$$\left\| \hat{Y}_{\hat{\gamma}} c - \frac{\hat{d}_\pi}{d_\mu} \right\|_{d_\mu} \leq \hat{\gamma}^n \sqrt{K_{\pi,\mu,1}} \left\| c - \frac{\hat{d}_\pi}{d_\mu} \right\|_{d_\mu}$$

and in particular $\hat{Y}_{\hat{\gamma}}$ is a contraction mapping for $\hat{\gamma} < (K_{\pi,\mu,n})^{-(2)^{-1}}$. Since $K_{\pi,\mu,1}$ is a bounded series, the exponential factor is guaranteed to dominate, so there exists a value of $\hat{\gamma} < 1$ for which the projected DCOP operator $\Pi_{d_\mu} \hat{Y}_{\hat{\gamma}}$ is a contraction mapping.

Remark. We note that the actual Theorem 4 of paper [3], from which previous Theorem 3.7 is extracted, provides a slightly more general result for n-step updates; we refer to [5] to get more details about these methods. For simplicity, we have reduced it to the case of 1-step updates, as we have been considering in the rest of our analysis.

So, provided we take a sufficiently small $\hat{\gamma}$, Theorem 3.7 guarantees that DCOP-TD overcomes the divergence issues of COP-TD by granting that $\Pi_{d_\mu} \hat{Y}_{\hat{\gamma}}$ is a contraction mapping. In addition, the empirical evaluation performed on [3] of this discounted version suggests that it is unlikely to be in the worst-case scenario achieving $K_{\pi,\mu,1}$ strictly, as the algorithm avoid divergence even with large $\hat{\gamma}$ values.

4 Distributional Covariate Shift Approach

Now that we have introduced both Distributional RL and the Covariate Shift Approach for Off-policy Learning, we are ready to present our original work.

Similarly to what was done in [1] by going beyond the notion of value within the reinforcement learning setting, we first wanted to analyze a distributional perspective of the covariate shift approach. Our starting point could be the following distributional equation

$$X(s') \overset{D}{=} \frac{\pi(A^{\mu}_{s,s'}|S^{\mu}_{s'})}{\mu(A^{\mu}_{s,s'}|S^{\mu}_{s'})} X(S^{\mu}_{s'}) \tag{4.1}$$

where

- X is the random ratio between distributions of a certain state.
- $(S^{\mu}_{s'}, A^{\mu}_{s,s'})$ is the previous state-action random variable:
  - The random variable $S^{\mu}_{s'}$ represents the states from which state s' is achievable by following policy μ; $S^{\mu}_{s'} = s$ with probability $\bar{P}_{\mu}(s|s')$
  - $A^{\mu}_{s,s'}$ encodes the random action that can be taken to get state s' from a state $s \sim S^{\mu}_{s'}$ according to policy μ, so $A^{\mu}_{s,s'} = a$ with probability $\bar{\mu}(a|S^{\mu}_{s'}, s')$

Thus, paying attention to equation 4.1, we note that it intrinsically expresses the random ratio of a state $X(s_{t+1})$ as a mixture distribution with the 'corrected' previous state random ratios $\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} X(s_t)$ as mixing components, and the previous state-action random variable $(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t,s_{t+1}})$ as the mixing distribution (see Appendix A.1 for more details about mixture distributions):

$$f_{X(s_{t+1})}(x) = \sum_{s_t} \bar{P}_{\mu}(s_t|s_{t+1}) \sum_{a_t} \bar{\mu}(a_t|s_t, s_{t+1}) f_{\frac{\pi(s_t,a_t)}{\mu(s_t,a_t)} X(s_t)}(x) \tag{4.2}$$

Notation. So as to reduce the complexity and increase the readability of the formulation, we introduce the following notation:

- Let define ρ the policy ratio π/μ:

$$\rho(a, s) := \frac{\pi(a|s)}{\mu(a|s)}$$

- Note in equation 4.2 that there are as many mixture components as state-action pairs (s_t, a_t); thus, we can iterate the summation over all these possible pairs and define each corresponding mixture weight α as

$$\alpha(s_t, a_t|s_{t+1}) = \bar{P}_{\mu}(s_t|s_{t+1}) \bar{\mu}(a_t|s_t, s_{t+1})$$

4.1 Distributional DCOP-TD

We will directly develop the basis of a Distributional $\hat{\gamma}$-Discounted COP-TD, which has provided us with better convergence guarantees in the expected-valued case; in addition, we recall that it's more general in the sense that we can recover the undiscounted setting by simply considering $\hat{\gamma} = 1$.

Definition 1. We define the *distributional DCOP operator* $Y_{\hat{\gamma}}^D : \mathscr{P}(\mathbb{R}) \to \mathscr{P}(\mathbb{R})$ as

$$(Y_{\hat{\gamma}}^D X)(s_{t+1}) :\overset{D}{=} \hat{\gamma}\ \rho(S_{s_{t+1}}^{\mu}, A_{s_t,s_{t+1}}^{\mu}) X(S_{s_{t+1}}^{\mu}) + 1 - \hat{\gamma}$$

so that

$$f_{(Y^D X)(s_{t+1})}(x) = \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\hat{\gamma}\rho(s_t,a_t)X(s_t)+1-\hat{\gamma}}(x)$$

In addition, we can recover the distributional undiscounted setting by simply considering $\hat{\gamma} = 1$, with the corresponding *distributional COP operator* $Y^D = Y_1^D$.

Lemma 4.1. *Let $(Y_{\hat{\gamma}}^D X)(s) \in \mathscr{P}(\mathbb{R})$ be the resulting distribution of applying the distributional DCOP operator over a random ratio $X(s)$, for any state $s \in \mathcal{S}$. Then we have that*

$$\mathbb{E}[(Y_{\hat{\gamma}}^D X)(s)] = Y_{\hat{\gamma}}(\mathbb{E}[X(s)])$$

where $Y_{\hat{\gamma}}$ is the original value-based DCOP operator defined in Equation 3.13.

Proof. Let's consider $s_{t+1} \in \mathcal{S}$. The result follows from expanding the expectation of $(Y_{\hat{\gamma}}^D X)(s_{t+1})$ by using Property 3 of Mixture Distributions (Appendix A.1):

$$\begin{aligned}
\mathbb{E}[(Y_{\hat{\gamma}}^D X)(s_{t+1})] &= \int_{-\infty}^{\infty} x f_{(Y_{\hat{\gamma}}^D X)(s_{t+1})}(x) dx \\
&= \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot \int_{-\infty}^{\infty} x f_{\hat{\gamma}\rho(s_t,a_t)X(s_t)+1-\hat{\gamma}}(x) dx \\
&= \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot \mathbb{E}\left[\hat{\gamma}\rho(s_t, a_t)X(s_t) + 1 - \hat{\gamma}\right] \\
&= 1 - \hat{\gamma} + \hat{\gamma} \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot \rho(s_t, a_t) \cdot \mathbb{E}\left[X(s_t)\right] \\
&= 1 - \hat{\gamma} + \hat{\gamma} \mathop{\mathbb{E}}_{s_t \sim \bar{P}_\mu(\cdot|s_{t+1}), a_t \sim \bar{\mu}(\cdot|s_t,s_{t+1})} [\rho(s_t, a_t)\, \mathbb{E}[X(s_t)]] \\
&= 1 - \hat{\gamma} + \hat{\gamma} Y(\mathbb{E}[X(s)]) = Y_{\hat{\gamma}}(\mathbb{E}[X(s)])
\end{aligned} \tag{4.3}$$

where the last but one step holds taking into account the second term of the expansion 3.8 of the COP operator Y. □

In the previous Lemma 4.1 we show that the expectation of the learned ratio distributions coincide with the values we would learn by applying the usual DCOP operator to the expected values of the ratio distributions. This helps us prove in the following Corollary 4.2 that these expectations $\mathbb{E}[(Y^D X)(s)]$ converge to a fixed point, which in turn allows us to estimate the Covariate Shift Ratio values in this Distributional DCOP-TD setting as simply

$$\frac{d_\pi}{d_\mu}(s) = \mathbb{E}[X(s)]$$

Corollary 4.2. *Consider the process $X^{k+1} := Y_{\hat{\gamma}}^D X^k$ for any initial ratio distribution $X^0 \in \mathcal{P}(\mathbb{R})^n$. Then $\mathbb{E}[X^k]$ converges to some fixed point $c \in \mathbb{R}^n$ as $k \to \infty$.*

Proof. Lemma 4.1 tells us that:

$$\mathbb{E}[X^{k+1}] = \mathbb{E}[Y_{\hat{\gamma}}^D X^k] = Y_{\hat{\gamma}}(\mathbb{E}[X^k])$$

Defining $c^k := \mathbb{E}[X^k]$, we note that the above expression defines the process

$$c^{k+1} = Y_{\hat{\gamma}} c^k$$

Invoking the convergence result (Theorem 3.5) of the DCOP operator $Y_{\hat{\gamma}}$, we conclude that $c^k = \mathbb{E}[X^k]$ converges to some fixed point $c \in \mathbb{R}^n$. □

However, don't have any convergence guarantee for the iterates $\{X_k\}$ to some ratio distribution X^C, as we would like to. That is precisely what motivated us to develop the following logarithmic approach, which provides us with the desired distributional convergence.

4.2 Log-Distributional Covariate Shift Approach

Note that the Distributional Covariate Shift Equation 4.1 is purely multiplicative, and so it is the associated update rule in the learning setting. This complicates our analysis, and we suspect it is the main reason why we couldn't find any convergence guarantee for the distributional DCOP-TD algorithm.

There was, though, one way of transforming that multiplicative behaviour into a better-suited aditive one: working in the logarithmic space. To do so, we first define the corresponding log-ratio distributions:

Definition 2. Consider any state $s_{t+1} \in \mathcal{S}$, and the corresponding ratio distribution $X(s_{t+1})$. We define the log-ratio distribution $W(s_{t+1}) \in \mathscr{P}(\mathbb{R})$ as

$$W(s_{t+1}) := \log\left(X(s_{t+1})\right),$$

so that

$$\begin{aligned} f_{W(s_{t+1})}(x) &= \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\log(\rho(s_t,a_t)X(s_t))}(x) \\ &= \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\log(\rho(s_t,a_t))+W(s_t)}(x) \end{aligned}$$

Having these new distribution objects lying in the logarithmic space, our proposal is to learn an estimate of the true log-ratio distribution W^C in that space. Hence, in the undiscounted case, the corresponding learning rule would be captured by the following operator:

Definition 3. We define the Log-Distributional COP operator $G : \mathscr{P}(\mathbb{R}) \to \mathscr{P}(\mathbb{R})$ as

$$(GW)(s_{t+1}) := \log\left(\rho(S^\mu_{s_{t+1}}, A^\mu_{s_t,s_{t+1}})\right) + W(S^\mu_{s_{t+1}}) \tag{4.4}$$

so that

$$f_{(GW)(s_{t+1})}(x) = \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\log(\rho(s_t,a_t))+W(s_t)}(x)$$

At this point, one might wonder why this logarithmic approach wasn't considered in the original value-based (D)COP-TD development; the reason is that, in general, Jensen's inequality prevents the correspondence between the multiplicative fixed point and the exponential of the log fixed point:

$$\mathbb{E}[c(s)] \neq \exp(\mathbb{E}[\log c(s)]) \quad \text{since} \quad \mathbb{E}[\log c(s)] \leq \log(\mathbb{E}[c(s)])$$

In contrast to that, Proposition 4.3 shows that we can circumvent this limitation in the distributional setting:

Proposition 4.3. *For any state $s \in \mathcal{S}$, let define $U(s) := \exp(W(s))$. Then we have that*

$$U(s) :\overset{D}{=} X(s)$$

In particular, this implies that

$$\mathbb{E}[\exp(GW(s))] = \mathbb{E}[Y^D X(s)]$$

Proof. We can simply show that density functions of $U(s)$ and $X(s)$ are the same, which is straightforward using Mixture Properties (Appendix A.1):

$$\begin{aligned} f_{U(s_{t+1})}(x) &= \sum_{(s_t,a_t)} \alpha(s_t,a_t|s_{t+1}) \cdot f_{\exp(\log(\rho(s_t,a_t)X(s_t)))}(x) \\ &= \sum_{(s_t,a_t)} \alpha(s_t,a_t|s_{t+1}) \cdot f_{\rho(s_t,a_t)X(s_t)}(x) \\ &= f_{X(s_{t+1})}(x) \end{aligned} \tag{4.5}$$

□

Note that the previous result ensures that learning the additive fixed distribution W^C in log space and exponentiating it would lead to a proper ratio distribution X^C. However, we emphasize that, so far, we don't have any guarantee that these distributional fixed point exists, neither in log space nor in the original multiplicative one.

What we do know, though, is that the expectation of the ratio distributions that result from the repeated application of the distributional (D)COP operator converge (Corollary 4.2); the following Corollary of Proposition 4.3 allows us to extend this result to the exponentiated log-ratio distributions, so that, again, we can estimate the Covariate Shift Ratio values in the Log-Distributional COP-TD setting as simply

$$\frac{d_\pi}{d_\mu}(s) = \mathbb{E}[\exp(W(s))]$$

Corollary 4.4. *Consider the process $W^{k+1} := GW^k$ for any initial log-ratio distribution $W^0 \in \mathcal{P}(\mathbb{R})^n$. Then $\mathbb{E}[\exp(W^k)]$ converges to some fixed point $c \in \mathbb{R}^n$ as $k \to \infty$.*

Proof. In particular, Proposition 4.3 implies that

$$\mathbb{E}[\exp(GW(s))] = \mathbb{E}[Y^D X(s)]$$

From that, the result follows by applying Corollary 4.2. □

Our goal, nonetheless, is to get a more robust convergence result beyond expectations: we would like to prove the existence -and uniqueness- of some fixed point W^C. In practice, as we have shown in previous Sections, the discounted version is potentially more suitable to attain that objective, but now that we work in a logarithmic space we face the problem of how exactly apply the $\hat{\gamma}$-discounting in order to get a practical and reasonable algorithm.

We finally decided to implement the discounting so that the resulting operator belongs to the class of Bellman operators already detailed and analyzed in the RL literature. As we shall see, this will greatly simplify our study, and will provide us with strong convergence results.

Definition 4. We define the Log-Distributional DCOP operator $G_{\hat{\gamma}} : \mathscr{P}(\mathbb{R}) \to \mathscr{P}(\mathbb{R})$ as

$$(G_{\hat{\gamma}}W)(s_{t+1}) := \log\left(\rho(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t,s_{t+1}})\right) + \hat{\gamma}W(S^{\mu}_{s_{t+1}}) \tag{4.6}$$

so that

$$f_{(G_{\hat{\gamma}}W)(s_{t+1})}(x) = \sum_{(s_t,a_t)} \alpha(s_t, a_t|s_{t+1}) \cdot f_{\log(\rho(s_t,a_t))+\hat{\gamma}W(s_t)}(x)$$

The similarities of $G_{\hat{\gamma}}$ with the usual distributional Bellman operator (Expression 2.15) are quite evident; in fact, the Log-Distributional DCOP operator defines a distributional Bellman update where

- the policy that drives the agent trajectory is μ
- the state-to-state transition function is $\bar{P}_{\mu}$, the time reversal transition function defined in 3.6.
- the immediate reward is $\log\left(\rho(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t,s_{t+1}})\right)$
- the discount factor is $\hat{\gamma}$
- the value distribution W is expressed as a sum of $\hat{\gamma}$-discounted rewards

The benefits of identifying $G_{\hat{\gamma}}$ as a Bellman operator are hugely relevant, since all results previously presented of the Bellman operator $\mathcal{T}^{\pi}$ of Distributional Reinforcement Learning hold for our Log-Distributional DCOP operator. In particular, we can directly re-write the contraction result of Lemma 2.1 to

Lemma 4.5. *$G_{\hat{\gamma}} : \mathcal{P}(\mathbb{R}) \to \mathcal{P}(\mathbb{R})$ is a $\hat{\gamma}$-contraction in $\bar{d}_p$.*

Proof. Proof of Lemma 2.1. □

Hence, we can at last prove the desired convergence of log-ratio distributions to some fixed point in the general distributional setting when using the supremum-Wasserstein metric:

Corollary 4.6. *The process $W^{k+1} := G_{\hat{\gamma}}W^k$, for any initial log-ratio distribution W^0, converges to some fixed point W^C in $\bar{d}_p$.*

Proof. Using Lemma 4.5, we conclude applying Banach's fixed point theorem that $G_{\hat{\gamma}}$ has a unique fixed point W^C. As we assume all moments are bounded, this is sufficient to conclude that the sequence $\{W^k\}$ converges to W^C in $\bar{d}_p$ for $1 < p < \infty$. □

4.3 Categorical Log-Distributional DCOP-TD

The idea is to design a Categorical framework for Log-Distributional DCOP-TD analogous to those of CDRL algorithms whose convergence has been proven. Hence, we will make use of the following approximations:

- We consider the parametric family of categorical distributions over some fixed set of equally-spaced supports $w_1 < \cdots < w_K$:

$$\mathcal{P} = \left\{ \sum_{i=1}^{K} p_i \delta_{w_i} \middle| p_1, \ldots, p_K \geq 0, \sum_{k=1}^{K} p_k = 1 \right\}$$

- For any state $s_{t+1} \in \mathcal{S}$, we assume $\rho(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t,s_{t+1}})$ can be computed, and we will denote this value as simply ρ.

- Given a state $s_{t+1} \in \mathcal{S}$ and its corresponding ρ, the Log-Distributional DCOP operator $G_{\hat{\gamma}}$ transforms the support $\{w_1, \dots, w_K\}$ of log-ratio distributions by an affine shift map $f_{\log(\rho),\hat{\gamma}} : \mathbb{R} \to \mathbb{R}$ defined by $f_{\log(\rho),\hat{\gamma}}(w_i) := \log(\rho) + \hat{\gamma} w_i$. Hence, we can write

$$G_{\hat{\gamma}} W(s) := (f_{\log(\rho),\hat{\gamma}})_{\#} W(s)$$

- We apply the heuristic projection operator Π_C defined in 2.17 so as to recover a ratio distribution within $\mathcal{P}$ after applying the Log-Distributional DCOP operator $G_{\hat{\gamma}}$.

- We denote by $W^{\mu} \in \mathscr{P}(\mathbb{R})$ the true log-ratio distribution which we aim to estimate.

In the following subsections we present the corresponding convergence results for Categorical Log-Distributional DCOP-TD algorithms when using Tabular Representation and Linear Function Approximation, respectively.

4.3.1 Tabular Representation

Let us first consider the simplest setting, where we assume we are able to store an approximate parametrized log-ratio distribution for each state, and no stochastic approximation is required during the learning process. Given the above categorical framework, the operator to be analyzed is now $\Pi_C G_{\hat{\gamma}}$.

However, due to the fact that $G_{\hat{\gamma}}$ is a Bellman operator, we can re-use the results of Section 2.4.1 for CDRL algorithms with Tabular Representation. This dramatically simplifies our study, as we can directly show a strong convergence result in the supremum-Cramér distance:

Proposition 4.7. *The operator $\Pi_C G_{\hat{\gamma}}$ is a $\sqrt{\hat{\gamma}}$-contraction in $\bar{\ell}_2$. Further, there is a unique distribution function $W^C \in \mathcal{P}^{\mathcal{S}}$ to which the process $W^{k+1} := \Pi_c G_{\hat{\gamma}} W^k$, given any initial log-ratio distribution function $W^0 \in \mathscr{P}(\mathbb{R})^{\mathcal{S}}$, converges in $\bar{\ell}_2$ as $k \to \infty$.*

Proof. Proof of Proposition 2.4. □

Moreover, when we know a priori that log-ratio distributions lie in a certain interval, we can easily bound the error of our estimate W^C with respect to the true log-ratio distribution W^{μ}:

Proposition 4.8. *Let W^C be the limiting return distribution of Proposition 4.7. If for all states their corresponding true log-ratio distribution W^{μ} is supported on $[w_1, w_K]$, then*

$$\bar{\ell}_2(W^C, W^{\mu}) \leq \frac{1}{1-\hat{\gamma}} \max_{1 \leq i \leq K} (w_{i+1} - w_i).$$

Proof. Proof of Proposition 2.5. □

This difference can be interpreted as the cost of using the parametrization $\mathcal{P}$; we observe how W^{μ} can be recovered by increasing the fineness of the support.

4.3.2 Linear Function Approximation

In our second and last framework, we consider the use of linear function approximation to estima te the parametrized log-ratio distribution of each state. We also make use of the Categorical setting described above, and avoid any stochasticity in the following analysis.

We will introduce the same notation that was presented in Section 2.4.2 for CDRL with Linear Function Approximation. Thus, the linear model redefines the approximated log-ratio distributions W as mappings from states $s \in \mathcal{S}$ to vectors defined by a linear combination of features:

$$W_\Theta(s) := \Theta^\top \phi(s) \tag{4.7}$$

where $\phi(s) \in \mathbb{R}^m$ is the feature vector at state s and $\Theta \in \mathbb{R}^{n\times K}$ represents the weight matrix. In vector notation, we have $W_\Theta = \Phi\Theta \in \mathbb{R}^{n\times K}$, where $\Phi \in \mathbb{R}^{n\times m}$ is the feature matrix.

Remark. We recall the assumption that vector $W_\Theta(s) \in \mathbb{R}^K$ is an estimation of a distribution over the support $\{w_1, \dots, w_K\}$, although it might have negative components and it is not necessarily normalized.

Again, all the theory developed in Section 2.4.2 with Linear Function Approximation is valid for our Categorical Log-Distributional DCOP-TD due to the class equivalence of operators $G_{\hat\gamma}$ and $\mathcal{T}^\pi$, and the equivalence of the Categorical framework. So, in order not to repeat step by step the study presented in that Section 2.4.2, we can directly show the final convergence result that applies in our case:

Theorem 4.9. *Let d_μ the stationary distribution induced by policy μ. The process*

$$W^0 := \Phi\Theta^0, \quad W^{k+1} := \hat\Pi_{d_\mu,\lambda,\Phi} G_{\hat\gamma} W^k$$

converges to a set S such that, for any two $W, W' \in S$, there is a $\mathcal{S}$-indexed vector of constants α such that

$$W(s) = W'(s) + \alpha(s) e_K.$$

If $\lambda > 0$, S consists of a single point W^C which is the fixed point of the process. Moreover, we can bound the error of this fixed point with respect to the true log-ratio distribution W^μ by

$$\ell^2_{d_\mu,\lambda}(W^C, W^\mu) \le \frac{1}{1-\hat\gamma}\ell^2_{d_\mu,\lambda}(\Pi_{d_\mu,\lambda,\Phi} W^\mu, W^\mu) - \frac{\hat\gamma\lambda}{1-\hat\gamma}\|W^C - W^\mu\|_{d_\mu, e_K e_K^\top},$$

where the second terms measures the difference in mass between $\hat W$ and W^μ.

We highlight the previous Theorem 4.9, as it provides us with a convergence result independent of the value of $\hat\gamma$ in the distributional setting with Linear Function Approximation; if we recall Theorem 3.5 in the value-based setting, convergence was conditioned to a sufficiently small value of the discounting $\hat\gamma$.

Hence, we note that our log-distributional approach improves the theoretical convergence guarantees of the process of learning the true ratio distributions. We can perform the learning process in log space, where we can find the additive fixed point W^C, and get our estimate X^C of the true ratio distribution as simply

$$X^C := \exp(W^C)$$

5 Implementation

In the previous sections, the theoretical framework has been presented and analyzed in detail. Our goal now is to describe how we have implemented that framework in practice

First of all, we highlight the use of Dopamine[7], a research framework for fast prototyping of reinforcement learning algorithms, to implement our Distributional Covariate Shift approach; all the development can be found at our Github repository. Basically, our baseline is the `RainbowAgent`[17] already implemented in Dopamine with the C51 configuration file. The resulting `CovariateShiftAgent` is contained in two files:

- The agent class in `dopamine/agents/covariate_shift/covariate_shift_agent.py`, inheriting from `rainbow_agent.RainbowAgent`.
- The replay buffer in `dopamine/replay_memory/cs_replay_buffer.py`, inheriting from `prioritized_replay_buffer.WrappedPrioritizedReplayBuffer`.

In addition, the configuration file

`dopamine/agents/covariate_shift/configs/covariate_shift.gin`

allows us to easily control all hyper-parameter values of the agent. For more technical information about the implementation, we refer to the API Documentation of our repository, in which we detail each module, class and function that conforms this agent.

In the following subsections, we will analyze the key points of our practical framework at a higher level of abstraction. First, we will describe the function approximation that we use to get our distribution estimates. In the following section, we give account of the sampling process and the replay memory where we store them. Thirdly, we detail how we dealt with the different approximations associated to distributional settings; this facilitates us to finally present in the last two subsections the Categorical Distributional and Categorical Log-Distributional DCOP-TD algorithms, respectively, that we have implemented in our `CovariateShiftAgent`.

5.1 Non-linear Function Approximation

In our attempt to achieve the best experimental performance, we go beyond the defined theoretical framework by implementing a non-linear function approximation for getting ratio -or log-ratio- distribution estimates. This is a choice that prevent us from having any convergence guarantee, but it is supported by the outstanding experimental behaviour of C51[1], which makes use of a non-linear neural network. In fact, in our implementation we augment the original C51 network by adding an extra head, the distributional ratio model, to the final convolutional layer, whose role is to predict the corresponding (log-)ratio distribution.

We can see in Figure 1 the model network architecture in detail when working with the ALE environment[14]. Game frames are selected, grouped and preprocessed in the standard way, generating inputs which consist of a $84 \times 84 \times 4$ image. The following convolutional layers are exactly as the ones already defined in the DQN implementation[15]:

- The first hidden layer convolves 32 filters of 8×8 with stride 4 with the input image and applies a rectifier non-linearity -in particular, a ReLu[16].

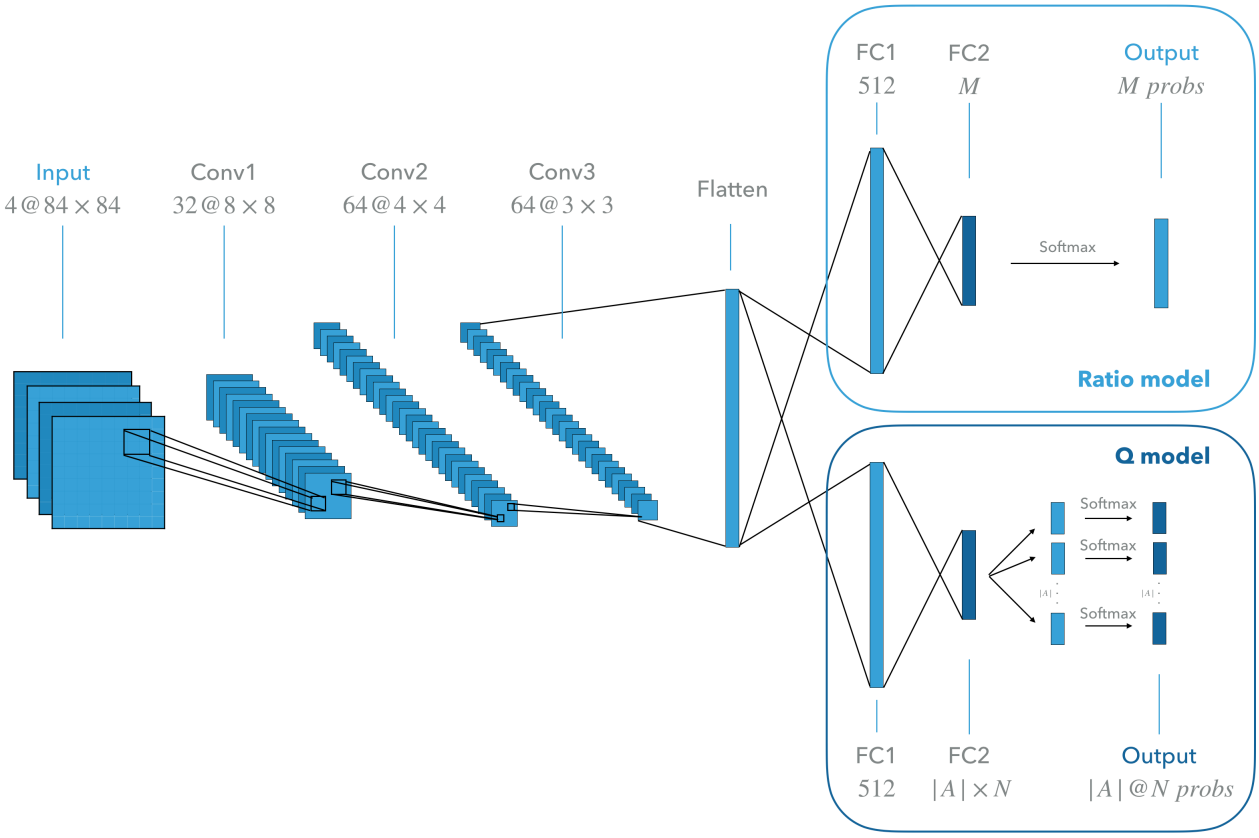

Figure 1: Visual representation of the Neural Network implemented in `CovariateShiftAgent` working in the ALE environment.

- The second hidden layer convolves 64 filters of 4×4 with stride 2, again followed by a ReLu activation function.
- The third and last convolutional layer convolves 64 filters of 3×3 with stride 1 and applies a ReLu rectifier as well, and the result is flattened.

At that point, our network is forked in two separate heads, each connected to the previous flattened layer:

- On the one hand, we have the part responsible of predicting the Q value distributions, which we call the Q model. Just as the original distributional RL models[1], the final hidden layer is fully-connected and consists of 512 rectifier units, followed by a fully-connected linear layer with $|\mathcal{A}| \times N$ outputs, being $|A|$ the number of actions of the played game[1] and N the number of fixed supports considered for the categorical Q distributions. This is re-arranged in $|A|$ groups of N outputs, and applying to each of these groups of logits a Softmax layer we finally obtain the N probabilities for each state-action pair.
- On the other hand, we have added the previously mentioned ratio model, which provides us with estimates of the covariate shift (log-)ratio distributions. As in the Q model, the final hidden layer is fully-connected and consists of 512 rectifier units. This is followed by a fully-connected linear layer that produces as mani outputs as the number of atoms M of the ratio parametric model. A final softmax layer transforms the resulting logits into probabilities.

[1]The number of valid actions varied between 4 and 18 along the games of ALE environment.

Finally, we notice that we work with double networks[18] -i.e. the processes of selection and evaluation of the bootstrap action are decoupled, with online and target weights respectively; this is a usual method in Deep Reinforcement Learning that addresses a natural overestimation bias of Q-learning[19]. In our implementation, the update period for the target network is controlled by `target_update_period` parameter, whose value is set to 8000 iterations just like the `RainbowAgent`[17].

5.2 Replay Memory

The implemented replay memory in Dopamine[7] is responsible of storing past transitions of the agent, acting as a windowed buffer from which training samples are drawn continuously. Consequently, there is always some degree of off-polyceness during the learning process given that the learned policy is being constantly updated.

In order to evaluate our model, however, we are interested in achieving a degree of off-polyceness as high as possible; thus, we consider the very hard setting proposed for testing the Discounted COP-TD algorithm[3], in which:

- We have a fixed behaviour policy μ, the uniformly random policy;
- At each step, the target policy π is the ϵ-greedy policy with respect to the predicted Q estimates.

On ALE[14], this particular setup guarantees that the generated data is significantly different from any learned policy, as we want to.

Taking the work of [3] as reference, to reweight sample transitions we have implemented a prioritized replay memory[20] where priorities are simply covariate shift estimates. Nonetheless, this increases the risk of overfitting, since it reduces the effective size of the data set -those samples unlikely under policy π, which might be the vast majority due to the randomness of μ, end up mostly ignored; to compensate this effect, the `replay_capacity` is increased to 10M frames.

Authors of [3] also explain that stability issues arise when we learn the ratio with prioritized sampling. To overcome this, at each training step two independent transition batches are sampled from the replay buffer: prioritized for the value distribution, and uniform for the covariate shift ones. In both cases, the `batch_size` is set to 32 in our experiments.

Last, but not least, we note that initial states cannot be updated through transition samples as the training of the ratio distributions is done 'backwards'. In these cases, given that the distribution of any initial state is policy-independent, their ratio is 1; in our implementation, we take this into account by replacing the bootstrapping target with a Dirac delta centered at 1.

5.3 Distributional Setting

We recall Section 2.3, where we described the approximation framework defined in [12] to implement reproducible and scalable distributional-based RL algorithms. Let's see how we have dealt in practice with each of the key points of that framework:

Distribution Parametrisation

Just as we considered along the theoretical framework, we select the parametric family of categorical distributions over some fixed set of supports $z_1 < \cdots < z_K$:

$$\mathcal{P} = \left\{ \sum_{i=1}^{K} p_i \delta_{z_i} \middle| p_1, \ldots, p_K \geq 0, \sum_{k=1}^{K} p_k = 1 \right\}$$

The only difference is that our implementation allows to define supports which are not necessarily equally-spaced; in fact, apart from linear divisions of the support, in our experiments we have also taken into account exponential bins.

Stochastic Learning

Despite the convergence results have been obtained without stochasticity, in practice it is difficult to find an example where it can be rejected. Hence, supported with the work done in [12, 11] regarding stochastic updates, our implementation learns to predict covariate shift ratio estimates through transition samples of the MDP.

In particular, the practical samples for the ratio model are simply of the form (s_t, a_t, s_{t+1}); neither the immediate reward r_t nor the next action a_{t+1} are required. However, in order to define the learning rule, the ratio of policies $\rho(S^{\mu}_{s_{t+1}}, A^{\mu}_{s_t, s_{t+1}})$ must be computed; in our stochastic setting, given a transition (s_t, a_t, s_{t+1}), it is estimated as $\rho_t := \rho(s_t, a_t)$. Since

- the *behavioural policy* μ is simply the uniformly random policy, i.e.

$$\mu(a_t|s_t) = \frac{1}{|\mathcal{A}|} \quad \forall a \in \mathcal{A};$$

- the *target policy* π is the ϵ-greedy policy with respect to the estimated state-action q-values of the model, i.e.

$$\pi_\theta(a_t|s_t) = \begin{cases} (1-\epsilon) + \epsilon \frac{1}{|\mathcal{A}|} & \text{if} \quad a_t = \arg\max_a Q_\theta(s_t, a) \\ \epsilon \frac{1}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

we can easily obtain a computable expression:

$$\rho_t = \frac{\pi_\theta(a_t|s_t)}{\mu(a_t|s_t)} = \begin{cases} |\mathcal{A}|(1-\epsilon) + \epsilon & \text{if} \quad a_t = \arg\max_a Q_\theta(s_t, a) \\ \epsilon & \text{otherwise} \end{cases} \tag{5.1}$$

Once we have the ratio computed for each sample, it is straightforward to define both the *stochastic Distributional DCOP operator* $\hat{Y}^D_{\hat{\gamma}}$ and the *stochastic Log-Distributional DCOP operator* $\hat{G}_{\hat{\gamma}}$, which modify the supports of ratio distributions and log ratio distributions, respectively, according to the 1-step information contained in each sample.

Projection of Bellman Target Distribution

In order to recover a ratio distribution within $\mathcal{P}$ after applying the stochastic operator -either $\hat{Y}^D_{\hat{\gamma}}$ or $\hat{G}_{\hat{\gamma}}$, depending on the case-, we also use the heuristic projection operator Π_C defined in 2.17; in particular, we coded it extended to finite mixtures of Dirac measures:

$$\Pi_C \left(\sum_{i=1}^{N} p_i \delta_{y_i} \right) = \sum_{i=1}^{N} p_i \Pi_C(\delta_{y_i})$$

We note, however, that our implementation of the function `project_distribution(...)`[2], responsible of performing this projection Π_C, differs from the already implemented in Dopamine; we have slightly modified it so as to allow non-equally spaced supports, thus enabling the option of running experiments with exponential bins as well.

Gradient Updates

-A decision had to be taken about how to compute the next iterate of our stochastic approximation. Attending to what we have shown in our theoretical framework, defining a gradient update based on a Cramér loss seem to be the most appropriate solution given that we have implemented the Cramér projection Π_C. However, we finally decided to imitate the original C51 agent and perform a single step of gradient descent on the Kullback-Leibler divergence of the predicted distribution from the target one with respect to the parameters of the prediction.

The reasons behind our decision are purely practical; despite there is no convergence guarantee of C51 due to this KL divergence step, so far its experimental results[1] are much better than those that strictly rely on the theoretical framework[11]. Hence, as we now seek for the best performance possible of our Distributional Covariate Shift approach, we kind of ignore that part of the theory and implement the KL gradient update for generating the new estimates of our `CovariateShiftAgent`.

5.4 Categorical Distributional DCOP-TD

With all the main ingredients of our implementation already detailed in previous sections, we simply want to summarize the steps that conform our distributional DCOP-TD algorithms. To begin with, we show in Algorithm 2 the Categorical Distributional $\hat{\gamma}$-Discounted COP-TD main process, where

- $\{x_K\}$ is the set of atoms that define our categorical parametric family $\mathcal{P}$
- we denote by θ and $\bar{\theta}$ the weights of the online and target networks, respectively;
- $\{p_{\theta_t,K}(s)\}$ and $\{p_{\bar{\theta}_t,K}(s)\}$ are, respectively, the set of probability outputs of the online and target networks for a state $s \in \mathcal{S}$ at a training time step t.
- $\hat{\epsilon} \in [0,1]$ comes from the definition of the ϵ-greedy policy $\pi_{\hat{\theta}}$; in practice, however, it can be controlled apart by an extra hyper-parameter (`quotient_epsilon`).
- $(f_{a,b})_{\#}$ represents the application to each element of the support of an affine shift map $f_{a,b} : \mathbb{R} \to \mathbb{R}$ defined by $f_{a,b}(y) := a+by$; in particular, note that $Y^D_{\hat{\gamma}} X(s) := (f_{\cdot,\hat{\gamma}\rho})_{\#} X(s)$

[2]This function can be found in dopamine/agents/covariate_shift/covariate_shift_agent.py

Algorithm 2: Categorical Distributional DCOP-TD

Require: Estimates $X_{\theta_t}(s) = \sum_{k=1}^{K} p_{\theta_t,k}(s)\delta_{x_k}$ and $X_{\bar{\theta}_t}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_t,k}(s)\delta_{x_k}$ for each $s \in \mathcal{S}$

Input: A sample transition (s_i, a_i, s_{i+1})

`#Compute policy ratio`

if $a_t = \arg\max_a Q_{\bar{\theta}}(s_i, a)$ **then**

$\rho_i \leftarrow |\mathcal{A}|(1 - \hat{\epsilon}) + \hat{\epsilon}$

else

$\rho_i \leftarrow \hat{\epsilon}/|\mathcal{A}|$

end if

`#Compute distributional ratio target`

$\widehat{X}_*(s_{i+1}) \leftarrow Y^D_{\hat{\gamma}} X_{\bar{\theta}_t}(s_i)$

`#Project target onto support`

$\widehat{X}_t(s_{i+1}) \leftarrow \Pi_C \widehat{X}_*(s_{i+1})$

`#Compute KL loss`

Find gradient KL $\left(\widehat{X}_t(s_{i+1}) || X_{\theta_t}(s_{i+1})\right)$

Update online weights θ_{t+1}

`#Update target network if required`

$\bar{\theta}_{t+1} \leftarrow \theta_{t+1}$ **if** t % `target_update_period` $= 0$ **else** $\bar{\theta}_t$

Output: New estimates $X_{\theta_{t+1}}(s) = \sum_{k=1}^{K} p_{\theta_{t+1},k}(s)\delta_{x_k}$ and $X_{\bar{\theta}_{t+1}}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_{t+1},k}(s)\delta_{x_k}$ for each $s \in \mathcal{S}$

Ratio estimates $c_{\bar{\theta}_{t+1}}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_{t+1},k}(s)x_k$

5.5 Categorical Log-Distributional DCOP-TD

Finally, we present in Algorithm 3 the sketch of our Categorical Log-Distributional DCOP-TD algorithm. The observations made in previous Section apply here, where now $\{w_K\}$ is the support and $G_{\hat{\gamma}}W(s) = (f_{\log(\rho),\hat{\gamma}})_{\#}W(s)$. In fact, as one can easily see, the steps are equivalent to those of the multiplicative case.

Algorithm 3: Categorical Log-Distributional DCOP-TD

Require: Estimates $W_{\theta_t}(s) = \sum_{k=1}^{K} p_{\theta_t,k}(s)\delta_{x_k}$ and $W_{\bar{\theta}_t}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_t,k}(s)\delta_{x_k}$ for each $s \in \mathcal{S}$

Input: A sample transition (s_i, a_i, s_{i+1})

`#Compute the log policy ratio`

if $a_t = \arg\max_a Q_{\bar{\theta}}(s_i, a)$ **then**

$\log(\rho_i) \leftarrow \log\left(|\mathcal{A}|(1-\hat{\epsilon}) + \hat{\epsilon}\right)$

else

$\log(\rho_i) \leftarrow (\hat{\epsilon}/|\mathcal{A}|)$

end if

`#Compute distributional log-ratio target`

$\widehat{W}_*(s_{i+1}) \leftarrow (f_{\log(\rho_i),\hat{\gamma}})_{\#}W_{\bar{\theta}_t}(s_i)$

`#Project target onto support`

$\widehat{W}_t(s_{i+1}) \leftarrow \Pi_C \widehat{W}_*(s_{i+1})$

`#Compute KL loss`

Find gradient $\mathrm{KL}\left(\widehat{W}_t(s_{i+1}) || W_{\theta_t}(s_{i+1})\right)$

Update online weights θ_{t+1}

`#Update target network if required`

$\bar{\theta}_{t+1} \leftarrow \theta_{t+1}$ **if** t % `target_update_period` $= 0$ **else** $\bar{\theta}_t$

Output: New estimates $W_{\theta_{t+1}}(s) = \sum_{k=1}^{K} p_{\theta_{t+1},k}(s)\delta_{x_k}$ and $W_{\bar{\theta}_{t+1}}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_{t+1},k}(s)\delta_{x_k}$ for each $s \in \mathcal{S}$

Ratio estimates $c_{\bar{\theta}_{t+1}}(s) = \sum_{k=1}^{K} p_{\bar{\theta}_{t+1},k}(s)\exp(w_k)$

We note that we can easily get covariate shift estimates -which we recall are used as priorities for training the Q model- by exponentiating log-ratio distributions and taking its mean; in practice, as the support is fixed, we simply compute the exponentiated support at the beginning, and compute its weighted mean by the corresponding probabilities of log-ratio distributions.

6 Evaluation of the Proposal

Importance of Ratio Intervals

Linear vs Exponential Bins

Log-Distributional Approach

Qualitative Evaluation of the Learned Ratios

7 Conclusions

Appendix

A.1 Mixture Distributions

Remark. All random variables presented in this document are considered to be real-valued, i.e. their measurable space is $E = \mathbb{R}$.

A random variable Y is a mixture distribution if it is derived from a collection of other random variables $\{X_i\}$, $i \in \{1, \dots, N\}$, (named mixture components) in such a way that the combination of these parent distributions is driven according to a certain distribution A (called mixing distribution). A encapsulates the mixture weights $\alpha_i \sim A$, $i \in \{1, \dots, N\}$, which represent the probabilities of each individual mixture component X_i.

The mixture distribution Y can be defined in terms of its density function f_Y, which is the resulting α-convex combination of the mixture components' density functions:

$$f_Y(x) = \sum_{i=1}^{N} \alpha_i f_{X_i}(x) \tag{A.1}$$

Let's present some interesting properties of mixture distributions:

Property 1. *The expectation of the mixture distribution Y is the convex combination of expectations of each mixture component:*

$$\begin{aligned} \mathbb{E}[Y] &= \int_{-\infty}^{\infty} x f_Y(x) dx = \int_{-\infty}^{\infty} x \sum_{i=1}^{N} \alpha_i f_{X_i}(x) dx \\ &= \sum_{i=1}^{N} \alpha_i \int_{-\infty}^{\infty} x f_{X_i}(x) dx \\ &= \sum_{i=1}^{N} \alpha_i \, \mathbb{E}[X_i] \end{aligned} \tag{A.2}$$

Property 2. *Let be Z a mixture distribution with mixture components $\{g_i(X_i)\}$, $i \in \{1, \dots, N\}$ and mixing weights $\alpha_i \sim A$*

$$\mathbb{E}[Z] = \sum_{i=1}^{N} \alpha_i \, \mathbb{E}[g_i(X_i)] \tag{A.3}$$

Property 3. *Let be $Z = g(Y)$, being Y a mixture distribution with mixture components $\{X_i\}$, $i \in \{1, \dots, N\}$, and g a monotonic, invertible and differentiable function. Then we have that Z is a mixture distribution whose expectation is*

$$\begin{aligned} \mathbb{E}[Z] &= \int_{-\infty}^{\infty} g(x) f_Y(x) dx \\ &= \int_{-\infty}^{\infty} g(x) \left(\sum_{i=1}^{N} \alpha_i f_{X_i}(x) \right) dx = \sum_{i=1}^{N} \alpha_i \int_{-\infty}^{\infty} g(x) f_{X_i}(x) dx \\ &= \sum_{i=1}^{N} \alpha_i \, \mathbb{E}[g(X_i)] \end{aligned} \tag{A.4}$$

Note that in both the first and last steps the so-called Law of the Unconscious Statistician has been applied, which states that

$$\mathbb{E}[g(X)] = \int_{-\infty}^{\infty} y f_{g(X)}(y) dy = \int_{-\infty}^{\infty} g(x) f_X(x) dx \tag{A.5}$$

We emphasize the relevance of **Property 3**. In distributional TD learning, the distribution mixture plays the role of the expectation in expected TD. But while $\mathbb{E}[g(x)] \neq g(\mathbb{E}[x])$, we can interchange mixtures and functions. This allows us to circumvent Jensen's inequalities.

A.2 Metrics over Distributions

A.2.1 Kullback-Leibler Divergence

Definition 1. Let be $\nu_1, \nu_2 \in \mathscr{P}(\mathbb{R})$ two probability distributions. The Kullback-Leibler (KL) divergence of ν_1 from ν_2 is defined as

$$D_{KL}(\nu_1, \nu_2) = \int_{-\infty}^{\infty} \nu_1(x) \log \frac{\nu_1(x)}{\nu_2(x)} dx$$

A.2.2 Wasserstein

Definition 2. The Wasserstein distance d_p, for $p \in [1, \infty)$, between two distributions $\nu_1, \nu_2 \in \mathscr{P}(\mathbb{R})$, with cumulative distribution functions F_{ν_1}, F_{ν_2} respectively, can be defined by:

$$d_p(\nu_1, \nu_2) = \left(\int_{\mathbb{R}} |F_{\nu_1}^{-1}(u) - F_{\nu_2}^{-1}(u)|^p du \right)^{1/p}$$

Further, the supremum-Wasserstein metric $\bar{d}_p$ is defined between two value distribution functions $Z, Z' \in \mathcal{Z}$ by

$$\bar{d}_p(Z, Z') = \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} d_p(Z(s,a), Z'(s,a))$$

A.2.3 Cramér

Definition 3. The Cramér distance ℓ_2 between two distributions $\nu_1, \nu_2 \in \mathscr{P}(\mathbb{R})$, with cumulative distribution functions F_{ν_1}, F_{ν_2} respectively, is defined by:

$$\ell_2(\nu_1, \nu_2) = \left(\int_{\mathbb{R}} (F_{\nu_1}(x) - F_{\nu_2}(x))^2 dx \right)^{1/2}$$

Further, the supremum-Cramér metric $\bar{\ell}_2$ is defined between two value distribution functions $Z, Z' \in \mathcal{Z}$ by

$$\bar{\ell}_2(Z, Z') = \sup_{(s,a) \in \mathcal{S} \times \mathcal{A}} \ell_2(Z(s,a), Z'(s,a))$$

A.3 Contraction Mappings

A.4 Implementation Details

TODO!!!

Our baseline is the C51 distributional reinforcement learning agent[1] within Dopamine framework[7]. We use published hyperparameters unless otherwise noted. We augment the C51 network by adding an extra head, the distributional ratio model $X(s)$, to the final convolutional layer, whose role is to predict the distribution of the ratio d_π/d_μ . This model consists of a two-layer fully-connected network, with as many outputs as the number of atoms M of the parametric model. A final softmax layer transforms the resulting logits into probabilities.

References

[1] M. G. Bellamare, W. Dabney and R. Munos. A Distributional Perspective on Reinforcement Learning.

[2] A. Hallak and S. Mannor. Consistent On-Line Off-Policy Evaluation.

[3] C. Gelada and M. G. Bellamare. Off-Policy Deep Reinforcement Learning by Bootstrapping the Covariate Shift.

[4] S. P. Meyn and R. L. Tweedie. Markov chains and stochastic stability. 2012.

[5] Sutton, R. S., and Barto, A. G. 2018. Reinforcement learning: An introduction. MIT Press, 2nd edition.

[6] Tsitsiklis, J. N., and Van Roy, B. 1997. An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control 42(5):674–690.

[7] Dopamine: A Research Framework for Deep Reinforcement Learning. https://github.com/google/dopamine

[8] Bertsekas, D. and Tsitsiklis, J. Neuro-Dynamic Programming. Athena Scientific, 1996.

[9] Sutton, R. S., Mahmood, A. R., and White, M. An emphatic approach to the problem of off-policy temporal- difference learning. arXiv:1503.04269, 2015.

[10] Hallak, Assaf, Tamar, Aviv, Munos, Remi, and Mannor, Shie. Generalized emphatic temporal difference learning: Bias-variance analysis. arXiv preprint arXiv:1509.05172, 2015.

[11] Marc G. Bellemare, Nicolas Le Roux, Pablo Samuel Castro and Subhodeep Moitra. Distributional reinforcement learning with linear function approximation

[12] Rowland, Mark, Bellemare, Marc G, Dabney, Will, Munos, Rémi, and Teh, Yee Whye. An analysis of categorical distributional reinforcement learning. In Proceedings of the International Conference on Artificial Intelligence and Statistics, 2018.

[13] Jaquette, Stratton C. Markov decision processes with a new opti- mality criterion: Discrete time. The Annals of Statistics, 1(3): 496–505, 1973.

[14] Bellemare, Marc G, Naddaf, Yavar, Veness, Joel, and Bowling, Michael. The arcade learning environment: An evaluation plat- form for general agents. Journal of Artificial Intelligence Re- search, 47:253–279, 2013.

[15] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. Nature 518(7540):529–533.

[16] Nair, V. and Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. Proc. Int. Conf. Mach. Learn. 807–814 (2010).

[17] Hessel, Matteo, Modayil, Joseph, van Hasselt, Hado, Schaul, Tom, Ostrovski, Georg, Dabney, Will, Hor- gan, Dan, Piot, Bilal, Azar, Mohammad, and Silver, David. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[18] van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep re- inforcement learning with double Q-learning. In Proc. of AAAI, 2094–2100.

[19] van Hasselt, H. 2010. Double Q-learning. In Advances in Neural Information Processing Systems 23, 2613–2621.

[20] Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized experience replay. In International Conference on Learning Representations.