Distributional deep Q-learning with CVaR regression

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reinforcement learning (RL) allows an agent interacting sequentially with an environment to maximize its long-term return, in expectation. In distributional RL (DRL), the agent is also interested in the probability distribution of the return, not just its expected value. This so-called distributional perspective of RL has led to new algorithms with improved empirical performance. In this paper, we recall the atomic DRL (ADRL) framework introduced in [1] based on atomic distributions projected via the Wasserstein-2 metric. Then, we derive two new deep ADRL algorithms, namely SAD-Q-LEARNING and MAD-Q-LEARNING (both for the control task). Numerical experiments on various environments compare our approach against existing deep (distributional) RL methods.

1 Introduction

In reinforcement learning (RL), a decision-maker or *agent* sequentially interacts with an unknown and uncertain environment in order to optimize some performance criterion [21]. At each time step, the agent observes the current state of the environment, then takes an action that influences both its immediate reward and the next state. In other words, RL consists in learning through trial-and-error a strategy (or *policy*) mapping states to actions that maximizes the *long-term cumulative reward* (or *return*): this is the so-called control task. More accurately, this return is a random variable — due to the random transitions across states — and classical RL only focuses on its expected value. On the other hand, the policy evaluation task aims at assessing the quality of any given policy (not necessarily optimal as in control) by computing its expected return in each initial state, also called *value function*. For both evaluation and control, when the *model* (i.e. reward function and transition probabilities between states) is known, these value functions can be seen as fixed points of some operators and computed by dynamic programming (DP) under the Markov decision process (MDP) formalism [17]. Nevertheless, in RL the model is typically unknown and the agent can only approximate the DP approach based on empirical trajectories. The TD(0) algorithm for policy evaluation and Q-LEARNING for control, respectively introduced in [20] and [24], are flagship examples of the RL paradigm. Formal convergence guarantees were provided for both of these methods, see [10, 22, 11]. In many RL applications, the number of states is very large and thus prevents the use of the aforementioned tabular RL algorithms. In such situation, one should rather use function approximation to approximate the value functions, as achieved by the DQN algorithm [13, 14] borrowing ideas from Q-LEARNING and deep learning. More recently, the distributional reinforcement learning (DRL) framework was proposed by [3] (see also [15, 16]). In DRL, the agent is interested in the whole probability distribution of the return, not just its expectation. In this new paradigm, several distributional procedures where proposed as extensions of classic (non-distributional) RL methods, leading to improved empirical performance. In most cases, a DRL algorithm is composed of two main ingredients: i) a parametric family of distributions serving as proxies for the distributional returns, and ii) a metric measuring approximation errors between original distributions and parametric proxies. We recall a few existing DRL approaches, all of which considering mixtures of Dirac measures as their parametric family of

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

proxy distributions. Indeed, the Categorical DRL (CDRL) parametrization used in the C51 algorithm proposed in [3] was shown in [19] to correspond to orthogonal projections derived from the Cramér distance. [19] later provided the convergence analysis of CDRL. Instead of learning the categorical probabilities $p_1(x,a),\dots,p_N(x,a)$ of a distribution $\sum_{i=1}^N p_i(x,a)\delta_{z_i}$ with fixed predefined values $z_1,\dots,z_N$ as in C51, the quantile regression approach was proposed in [9], where the support $\{Q_1(x,a),\dots,Q_N(x,a)\}$ of the proxy distribution $\frac{1}{N}\sum_{i=1}^N \delta_{Q_i(x,a)}$ is learned for fixed uniform probabilities $\frac{1}{N}$ over the N atoms. In [9], these atoms result from Wasserstein-1 projections, and correspond to quantiles of the unprojected distribution. Later, [1] proposed a similar atomic method based on Wasserstein-2 projections, resulting in a more natural generalization of classic RL and leading to interpretable quantities in terms of robustness, as shown in [2]. From now on, we refer to this last approach as *atomic distributional reinforcement learning* (ADRL).

Main contribution Our main contribution is the following. We propose two new deep DRL algorithms for control (called SAD-Q-LEARNING and MAD-Q-LEARNING) whose updates for all atoms $Q_{i;\theta}(x,a)$ are obtained by minimizing (w.r.t. the deep Q-net parameters θ) the total squared loss

$$\sum_{i=1}^N \big(Q_{i;\theta}(x,a) - \mathrm{AVaR}_i(\nu)\big)^2 ,$$

where the target atoms $\mathrm{AVaR}_i(\nu)$ are easily computable quantities related to the *conditional value-at-risk* or "CVaR" of the unprojected target distribution ν. Additionally, we show that the average $\frac{1}{N}\sum_{i=1}^N \mathrm{AVaR}_i(\nu)$ is exactly equal to the Q-LEARNING update in the target network.

The paper is organized as follows. In Section 2, we recall a few standard RL tools and notations as well as their DRL generalization. Section 3 introduces a new distributional variant of the DQN algorithm. Finally, numerical experiments are provided in Section 4 for illustration purpose.

Notations We let $\mathcal{P}_b(\mathbb{R})$ be the set of probability measures on $\mathbb{R}$ having bounded support, and $\mathcal{P}(\mathcal{E})$ the set of probability mass functions on any finite set $\mathcal{E}$, whose cardinality is denoted by $|\mathcal{E}|$. The support of any discrete distribution $q \in \mathcal{P}(\mathcal{E})$ is $\mathrm{support}(q) = \{y \in \mathcal{E} \colon q(y) > 0\}$; the supremum norm of any function $h\colon \mathcal{E} \to \mathbb{R}$ is $\|h\|_\infty = \max_{y\in\mathcal{E}} |h(y)|$. The cumulative distribution function (CDF) of a real-valued random variable Z is the mapping $F(z) = \mathbb{P}(Z \le z)$ ($\forall z \in \mathbb{R}$), and we denote its generalized inverse distribution function (a.k.a. quantile function) by $F^{-1}\colon \tau \in (0,1) \mapsto \inf\{z \in \mathbb{R}, F(z) \ge \tau\}$. For any probability measure $\nu \in \mathcal{P}_b(\mathbb{R})$ and measurable function $f\colon \mathbb{R} \to \mathbb{R}$, the *pushforward measure* $\nu \circ f^{-1}$ is defined for any Borel set $A \subseteq \mathbb{R}$ by $\nu \circ f^{-1}(A) = \nu(\{z \in \mathbb{R} \colon f(z) \in A\})$. In this article, we only need the affine case $f_{r_0,\gamma}(z) = r_0 + \gamma z$ (with $r_0 \in \mathbb{R}, \gamma \in [0,1)$) for which $\nu \circ f_{r_0,\gamma}^{-1} \in \mathcal{P}_b(\mathbb{R})$ and $\nu \circ f_{r_0,\gamma}^{-1}(A) = \nu(\{\frac{z-r_0}{\gamma} \colon z \in A\})$ if $\gamma \neq 0$, or $\nu \circ f_{r_0,\gamma}^{-1} = \delta_{r_0}$ is the Dirac measure at r_0 if $\gamma = 0$. Lastly, we recall that a function mapping a metric space to itself is called a γ-contraction if it is Lipschitz continuous with Lipschitz constant $\gamma < 1$.

2 Background on (distributional) RL

2.1 Markov decision process and RL

Throughout the paper, we consider a Markov decision process (MDP) characterized by the tuple $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$ with:

- finite state space $\mathcal{X}$,
- finite action space $\mathcal{A}$,
- transition kernel $P\colon \mathcal{X} \times \mathcal{A} \to \mathcal{P}(\mathcal{X})$,
- reward function $r\colon \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$,
- discount factor $0 \le \gamma < 1$.

If the agent takes some action $a \in \mathcal{A}$ while the environment is in state $x \in \mathcal{X}$, then the next state X_1 is sampled from the distribution $P(\cdot|x,a)$ and the immediate reward is equal to $r(x,a,X_1)$. In the

discounted MDP setting, the agent seeks a policy $\pi\colon \mathcal{X} \to \mathcal{P}(\mathcal{A})$ maximizing its expected long-term return for each pair (x, a) of initial state and action:

$$Q^\pi(x,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(X_t, A_t, X_{t+1}) \;\middle|\; X_0 = x, A_0 = a\right],$$

where $X_{t+1} \sim P(\cdot|X_t, A_t)$ and $A_{t+1} \sim \pi(\cdot|X_{t+1})$. $Q^\pi(x,a)$ and $V^\pi(x) = \sum_{a\in\mathcal{A}} \pi(a|x) Q^\pi(x,a)$ are respectively called the state-action value function and the value function of the policy π. Further, each of these functions can be seen as the unique fixed point of a so-called Bellman operator [6], that we denote by T^π for Q-functions. For any $Q\colon \mathcal{X}\times\mathcal{A} \to \mathbb{R}$, the image of Q by T^π is another Q-function given by:

$$(T^\pi Q)(x,a) = \sum_{x'} P(x'|x,a)\left[r(x,a,x') + \gamma \sum_{a'} \pi(a'|x') Q(x',a')\right]. \tag{1}$$

This Bellman operator T^π has several nice properties: in particular, it is a γ-contraction in $\|\cdot\|_\infty$ and thus admits a unique fixed point (by Banach's fixed point theorem), namely $Q^\pi = T^\pi Q^\pi$. It is also well-known from [6] that there always exists at least one policy π^* that is optimal uniformly for all initial conditions (x, a):

$$Q^*(x,a) := Q^{\pi^*}(x,a) = \sup_\pi Q^\pi(x,a) \quad \text{and} \quad V^*(x) := \max_a Q^*(x,a) = V^{\pi^*}(x) = \sup_\pi V^\pi(x).$$

Similarly, this optimal Q-function Q^* is the unique fixed point of some operator T called the Bellman optimality operator and defined by:

$$(TQ)(x,a) = \sum_{x'} P(x'|x,a)\left[r(x,a,x') + \gamma \max_{a'} Q(x',a')\right], \tag{2}$$

which is also a γ-contraction in $\|\cdot\|_\infty$. Noteworthy, knowing Q^* is sufficient to behave optimally: indeed, a policy π^* is optimal if and only if in every state x, support$(\pi^*(\cdot|x)) \subseteq \arg\max_a Q^*(x,a)$.

Value-based RL Value-based reinforcement learning consists in approximating the aforementioned dynamic programming operators when the environment is unknown. More precisely, the agent do not have direct access to the reward function r or the kernel P but only observes empirical transitions $(x, a, r(x,a,x'), x')$ with $x' \sim P(\cdot|x,a)$. In this context, the temporal difference (TD) solves the policy evaluation problem through the TD(0) update [20]:

$$Q_{k+1}(x,a) \leftarrow (1-\alpha) Q_k(x,a) + \alpha\left[r(x,a,x') + \gamma \sum_{a'} \pi(a'|x') Q_k(x',a')\right], \tag{3}$$

for some step-size $\alpha \in (0,1)$. For the control task, the $(k+1)$st Q-LEARNING iterate proceeds as follows:

$$Q_{k+1}(x,a) \leftarrow (1-\alpha) Q_k(x,a) + \alpha\left[r(x,a,x') + \gamma \max_{a'} Q_k(x',a')\right]. \tag{4}$$

Both TD and Q-learning were proved to converge almost surely to respectively Q^π and Q^* under technical conditions borrowed from stochastic approximation theory; see [22, 11].

2.2 Distributional Bellman operators

In distributional RL, we replace scalar-valued functions Q by functions μ taking values that are entire probability distributions: $\mu^{(x,a)} \in \mathcal{P}_b(\mathbb{R})$ for each pair (x, a). In other words, μ is a collection of distributions indexed by states and actions. We recall below the definition of the distributional Bellman operator [3, 19], which generalizes the Bellman operator to distributions.

Definition 1. (DISTRIBUTIONAL BELLMAN OPERATOR). *Let π be a policy. The distributional Bellman operator $\mathcal{T}^\pi\colon \mathcal{P}_b(\mathbb{R})^{\mathcal{X}\times\mathcal{A}} \to \mathcal{P}_b(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ is defined for any distribution function $\mu = (\mu^{(x,a)})_{x,a}$ by*

$$(\mathcal{T}^\pi \mu)^{(x,a)} = \sum_{x',a'} P(x'|x,a)\pi(a'|x')\mu^{(x',a')} \circ f^{-1}_{r(x,a,x'),\gamma}.$$

We know from [3] that $\mathcal{T}^\pi$ is a γ-contraction in the maximal p-Wasserstein metric[1]

$$\mathcal{W}_p(\mu_1, \mu_2) = \max_{(x,a)\in\mathcal{X}\times\mathcal{A}} W_p(\mu_1^{(x,a)}, \mu_2^{(x,a)})$$

at any order $p \in [1, +\infty]$. Consequently, $\mathcal{T}^\pi$ has a unique fixed point $\mu_\pi = (\mu_\pi^{(x,a)})_{x,a}$ equal to the collection of the probability laws of the returns:

$$\mu_\pi^{(x,a)} = \text{Law}\left(\sum_{t=0}^{\infty} \gamma^t r(X_t, A_t, X_{t+1}) \,\middle|\, X_0 = x, A_0 = a\right) . \tag{5}$$

Although [3] have also proposed a DRL operator for control, unfortunately it is not a contraction and does not necessarily admits a fixed point, as stated in their Propositions 1-2. In [1], the author proposes an alternative, less ambitious, DRL approach that only renders the randomness induced by a single transition. Interestingly, this 1-step approach allows to define contractive mappings for both evaluation and control: this will be the basis for deriving one of our new ADRL algorithms.

1-step optimality operator Let us recall the 1-step DRL operator introduced in [1].

Definition 2. (1-STEP DISTRIBUTIONAL BELLMAN OPTIMALITY OPERATOR). *The 1-step distributional Bellman optimality operator* $\mathcal{T}\colon \mathcal{P}_b(\mathbb{R})^{\mathcal{X}\times\mathcal{A}} \to \mathcal{P}_b(\mathbb{R})^{\mathcal{X}\times\mathcal{A}}$ *is defined for any* μ *by*

$$(\mathcal{T}\mu)^{(x,a)} = \sum_{x'} P(x'|x,a)\delta_{r(x,a,x')+\gamma \max_{a'} \mathbb{E}_{Z\sim\mu^{(x',a')}}[Z]} \cdot$$

As illustrated by Example 1 in [2], applying $\mathcal{T}^\pi$ may be prohibitive in terms of space complexity: if μ is discrete, then $\mathcal{T}^\pi\mu$ is still discrete, but with up to $|\mathcal{X}||\mathcal{A}|$ more atoms. The 1-step operator $\mathcal{T}$ leads to a similar issue by producing distributions with a number $|\mathcal{X}|$ of atoms, which is also too demanding in terms of space complexity for a large state space.

Projected operators Motivated by this space complexity issue, projected DRL operators were proposed to ensure a predefined and fixed space complexity budget $N \ll |\mathcal{X}|$. A projected DRL operator $\Pi \circ \mathcal{T}$ is the composition of a DRL operator $\mathcal{T}$ with a projection Π over some parametric family of distributions. The Cramér distance projection $\Pi_\mathcal{C}$ has been considered in [3, 19, 4]. In this line of work, given fixed values $z_1 \leq \cdots \leq z_N$, the proxy distributions $\sum_{i=1}^N p_i(x,a)\delta_{z_i}$ are parametrized by the categorical probabilities $p_1(x,a), \ldots, p_N(x,a)$. In [9], the Wasserstein-1 projection Π_{W_1} over atomic distributions $\frac{1}{N}\sum_{i=1}^N \delta_{Q_i(x,a)}$ is used. For this specific projection, the atoms that best approximate some CDF $F_{x,a}$ are obtained through quantile regression with $Q_i(x,a) = F_{x,a}^{-1}(\frac{2i-1}{2N})$. In this paper, we also consider such atomic distributions, but we use the Wasserstein-2 projection Π_{W_2}. Interestingly, the Wassertein-2 projection preserves the expected value [1]. Further, [2] has shown that it leads to fixed point quantities having a robust MDP interpretation. In summary, our approach is based on the projected optimality operator $\Pi_{W_2} \circ \mathcal{T}$.

2.3 The Wasserstein-2 projection

As observed in [1, 2], the Wasserstein-2 projection onto the space of atomic distributions produces quantities obtained by integrating the quantile function. In this article, we call average value-at-risk (AVaR) the local average of the quantile function over any interval included in $[0, 1]$. We point out that this is an abuse of language as the standard definition of the AVaR (also called conditional value-at-risk or CVaR, see [18]) integrates the quantile function over some interval of the form $[0, \beta]$ or $[\beta, 1]$, but never $[\beta_1, \beta_2]$ with $0 < \beta_1 < \beta_2 < 1$ as we do.

Definition 3. (AVERAGE VALUE-AT-RISK). *Let* $1 \leq i \leq N$ *and* F *be the cumulative distribution function (CDF) of a probability distribution* ν *on* $\mathbb{R}$ *with bounded support. The* i*-th average value-at-risk (AVaR) of* ν *is defined as:*

$$\textit{AVaR}_i(\nu) = N \int_{\tau=\frac{i-1}{N}}^{\frac{i}{N}} F^{-1}(\tau) d\tau ,$$

[1]For $p \geq 1$, we recall that the p-Wasserstein distance between two probability distributions ν_1, ν_2 on $\mathbb{R}$ with CDFs F_1, F_2 is defined as $W_p(\nu_1, \nu_2) = \left(\int_{\tau=0}^1 \left|F_1^{-1}(\tau) - F_2^{-1}(\tau)\right|^p d\tau\right)^{\frac{1}{p}}$. If $p = \infty$, $W_\infty(\nu_1, \nu_2) = \sup_{\tau\in(0,1)} |F_1^{-1}(\tau) - F_2^{-1}(\tau)|$.

where $F^{-1}(\tau) = \inf\{z \in \mathbb{R}, F(z) \geq \tau\}$ denotes the quantile function.

In this paper, we only consider AVaRs of discrete distributions for which F and F^{-1} are both non-decreasing staircase functions. Luckily in this specific case, the AVaR comes with a simple explicit expression given in the following lemma.

Lemma 4. (AVaR OF A DISCRETE DISTRIBUTION). *Let $1 \leq i \leq N$ and ν be a discrete real distribution*

$$\nu = \sum_{j=1}^{M} p_j \delta_{v_j} \ ,$$

with $M \geq 1$, $p_1, \ldots, p_M \geq 0$, $p_1 + \cdots + p_M = 1$ and sorted values $v_1 \leq \cdots \leq v_M$. Then,

$$\textit{AVaR}_i(\nu) = N \sum_{j=1}^{M} \textit{Length} \left(\left[\frac{i-1}{N}, \frac{i}{N} \right] \bigcap \left[\sum_{j' \leq j-1} p_{j'}, \sum_{j' \leq j} p_{j'} \right] \right) v_j \ .$$

From Lemma 4, we can compute exactly these discrete AVaRs by using the following formula for the length of the intersection of two intervals:

$$\text{Length}\left([L_1, R_1] \cap [L_2, R_2]\right) = \left[\min(R_1, R_2) - \max(L_1, L_2)\right]_+ \ , \tag{6}$$

where $[\cdot]_+ = \max(0, \cdot)$ denotes the positive part. Algorithm 1 fully describes the exact computation of the AVaRs of a discrete distribution; Figure 1 provides a graphic illustration of this procedure.

Algorithm 1 Discrete AVaR computation

Input: $N \geq 1$ and discrete distribution $\nu = \sum_{j=1}^{M} p_j \delta_{v_j}$ with $M \geq 1$.

Sort atoms:

$$v_{\sigma(1)} \leq \cdots \leq v_{\sigma(M)} \ \text{ with } \sigma \text{ an argsort permutation}$$

Reorder probability-atom pairs:

$$(p_j, v_j) \leftarrow (p_{\sigma(j)}, v_{\sigma(j)})$$

Compute AVaRs:

$$\text{AVaR}_i(\nu) = N \cdot \sum_{j=1}^{M} \left[\min\left(\frac{i}{N}, \sum_{j' \leq j} p_{j'} \right) - \max\left(\frac{i-1}{N}, \sum_{j' \leq j-1} p_{j'} \right) \right]_+ \cdot v_j$$

Output: $\text{AVaR}_1(\nu), \ldots, \text{AVaR}_N(\nu)$.

We are now ready to define the Wasserstein-2 atomic projection.

Definition 5. (WASSERSTEIN-2 PROJECTION). *Let $N \geq 1$ and μ be a collection of bounded distributions $\mu^{(x,a)}$.Then, the Wasserstein-2 projection $\tilde{\mu} = \Pi_{W_2}\mu$ onto the space of atomic distributions (having N uniformly weighted atoms) is given by:*

$$\tilde{\mu}^{(x,a)} = \frac{1}{N} \sum_{i=1}^{N} \delta_{\textit{AVaR}_i(\mu^{(x,a)})} \ .$$

In the next section, we describe our CVaR-based version of DQN derived from the projected operator $\Pi_{W_2} \circ \mathcal{T}$.

3 CVaR-based DQN

We now introduce our first practical deep DRL algorithm derived from the W_2-projected operator $\Pi_{W_2} \circ \mathcal{T}$. Let us denote $\mathcal{Q} = (Q_1, \ldots, Q_N)$ and $\mu_{\mathcal{Q}}^{(x,a)}$ the atomic distribution made of the N atoms

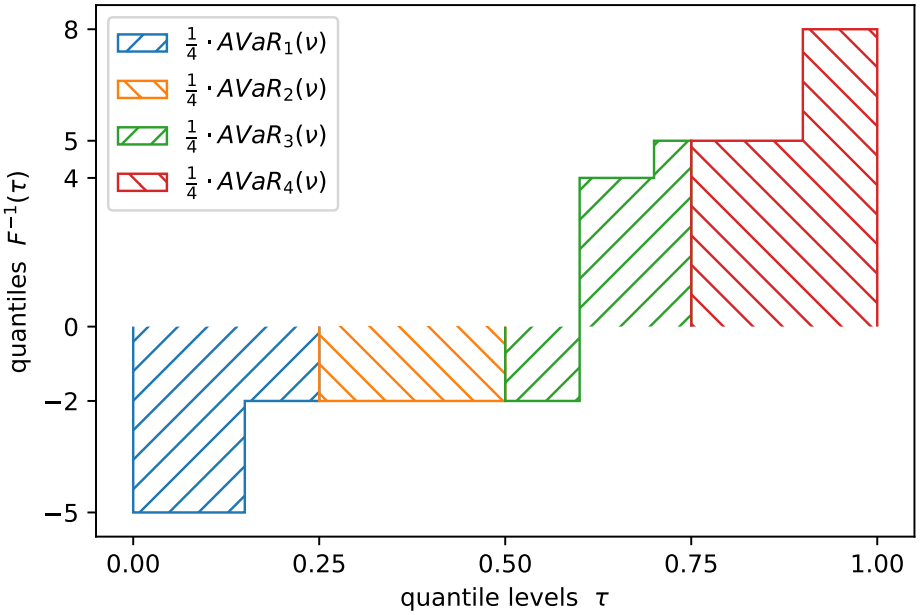

Figure 1: Wasserstein-2 projection of a discrete distribution ν (with CDF F) over $N = 4$ atoms. Each AVaR is obtained by multiplying by N the sum of the (signed) areas of some rectangles.

$Q_1(x, a), \dots, Q_N(x, a)$:

$$\mu_{\mathrm{Q}}^{(x,a)} = \frac{1}{N} \sum_{i=1}^{N} \delta_{Q_i(x,a)} \,.$$

Based on an empirical transition $(x, a, r(x, a, x'), x')$, we naturally consider the *unbiased* stochastic approximation $\hat{\mathfrak{T}}$ of the (unknown) operator $\mathfrak{T}$:

$$\left(\hat{\mathfrak{T}}\mu_{\mathrm{Q}}\right)^{(x,a)} = \delta_{r(x,a,x') + \gamma \max_{a'} Q(x',a')} \,,$$

where $Q = \frac{1}{N} \sum_{i=1}^{N} Q_i$. Then, as in the CDRL algorithm analysed in [19], we do a mixture update between our current atomic distribution and this stochastic target:

$$\nu \leftarrow (1 - \alpha)\mu_{\mathrm{Q}}^{(x,a)} + \alpha \left(\hat{\mathfrak{T}}\mu_{\mathrm{Q}}\right)^{(x,a)} \,, \tag{7}$$

which results in a distribution containing $N + 1$ atoms. Finally, in order to satisfy our fixed space complexity budget equal to N, we project ν with Π_{W_2} by computing the AVaRs:

$$Q_i(x, a) \leftarrow \mathrm{AVaR}_i(\nu) \,, \qquad \text{for all } 1 \leq i \leq N \,. \tag{8}$$

In a deep RL context, we rather minimize the squared distance between the atoms $Q_i(x, a)$ (that are parameterized by a deep neural network) and the targets $\mathrm{AVaR}_i(\nu)$: we call this method SAD-DQN (Algorithm 2) as it relies on Eq. 7 that mixes $\mu_{\mathrm{Q}}^{(x,a)}$ with a *single* Dirac measure. Similarly, we propose the Algorithm 3 called MAD-DQN, using a mixture update with *multiple* Dirac measures, which can be derived by approximating the DRL operator $\mathcal{T}^{\pi_Q}$ with π_Q a greedy policy w.r.t. Q.

Algorithm 2 SAD-DQN update

Input: $(Q_{1;\theta}, \ldots, Q_{N;\theta})$ with deep Q-net parameters θ, target network θ^-, transition $(x, a, r(x, a, x'), x')$, mixing ratio $\alpha \in (0, 1)$ and learning rate $\eta > 0$.
Target state-action value function:

$$Q_{\theta^-} \leftarrow \frac{1}{N} \sum_{i=1}^{N} Q_{i;\theta^-}$$

Target atomic distribution in (x, a):

$$\mu_{\theta^-}^{(x,a)} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \delta_{Q_{i;\theta^-}(x,a)}$$

Mixture update:

$$\nu \leftarrow (1-\alpha)\mu_{\theta^-}^{(x,a)} + \alpha\delta_{r(x,a,x')+\gamma \max_{a'} Q_{\theta^-}(x',a')}$$

Perform a gradient descent step w.r.t. θ on the squared Wasserstein-2 loss function:

$$\theta \leftarrow \theta - \eta\nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \left(Q_{i;\theta}(x, a) - \text{AVaR}_i(\nu)\right)^2$$

Output: Updated parameters θ.

Algorithm 3 MAD-DQN update

Input: $(Q_{1;\theta}, \ldots, Q_{N;\theta})$ with deep Q-net parameters θ, target network θ^-, transition $(x, a, r(x, a, x'), x')$, mixing ratio $\alpha \in (0, 1)$ and learning rate $\eta > 0$.
Target state-action value function:

$$Q_{\theta^-} \leftarrow \frac{1}{N} \sum_{i=1}^{N} Q_{i;\theta^-}$$

Target atomic distribution in (x, a):

$$\mu_{\theta^-}^{(x,a)} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \delta_{Q_{i;\theta^-}(x,a)}$$

Mixture update:

$$\nu \leftarrow (1-\alpha)\mu_{\theta^-}^{(x,a)} + \frac{\alpha}{N} \sum_{i=1}^{N} \delta_{r(x,a,x')+\gamma Q_{i;\theta^-}(x',a^*)}\,,$$

where $a^* \leftarrow \arg\max_{a'} Q_{\theta^-}(x', a')$
Perform a gradient descent step w.r.t. θ on the squared Wasserstein-2 loss function:

$$\theta \leftarrow \theta - \eta\nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \big(Q_{i;\theta}(x, a) - \text{AVaR}_i(\nu)\big)^2$$

Output: Updated parameters θ.

In the following lemma, we state that our approach satisfies a very natural property: the average target of our deep ADRL methods coincide with the standard Q-LEARNING update.

Lemma 6. (AVERAGING PROPERTY). *At the end of Algorithms 2-3, the average of the AVaR targets is equal to the* Q-LEARNING *update in the target network:*

$$\frac{1}{N}\sum_{i=1}^{N} \mathit{AVaR}_i(\nu) = (1-\alpha)Q_{\theta^-}(x,a) + \alpha\left[r(x,a,x') + \gamma \max_{a'} Q_{\theta^-}(x',a')\right] .$$

Proof. At the end of Algorithms 2-3, we have

$$\frac{1}{N}\sum_{i=1}^{N} \mathrm{AVaR}_i(\nu) = \frac{1}{N}\sum_{i=1}^{N} N \int_{\tau=\frac{i-1}{N}}^{\frac{i}{N}} F^{-1}(\tau)d\tau = \int_{\tau=0}^{1} F^{-1}(\tau)d\tau ,$$

where F denotes the CDF of ν. Then, recalling that the expectation is equal to the integral of the quantile function over the interval $(0,1)$ (see e.g. [7] or Lemma 3 in [2]), the desired result follows:

$$\frac{1}{N}\sum_{i=1}^{N} \mathrm{AVaR}_i(\nu) = (1-\alpha)Q_{\theta^-}(x,a) + \alpha\left[r(x,a,x') + \gamma \max_{a'} Q_{\theta^-}(x',a')\right] .$$

□

Lemma 6 suggests that our methods are natural as they extend classic deep Q-learning. Further, this averaging property paves the way for future theoretical analysis of the tabular version of our algorithms (with hard AVaR update in Eq. 8 instead of minimizing the squared distance). Indeed, a similar property is used in the proof of Theorem 2 in [19] for the convergence of categorical Q-learning.

4 Numerical experiments

In this section, we report preliminary experimental results. We compare our SAD-DQN and MAD-DQN algorithms against the classic DQN [13] as well as the distributional QR-DQN method [9] based on quantile regression. For SAD-DQN, MAD-DQN and QR-DQN, we take $N = 10$ atoms (additional plots for $N = 3$ are provided in the Supplementary Material), and for all of these three methods, we use the Adam optimzer [12] with learning rate set to 10^{-3} and batch size equal to 32. We run our experiments on four different OpenAI Gym environments [8]: Acrobot-v1, CartPole-v0, CartPole-v1 and LunarLander-v2. The discount factor is $\gamma = 0.99$ for all environments. Lastly, we choose our mixture ratio parameter $\alpha = 0.8$. In Figure 4, we observe convergence for both of our proposed methods, with even a slight advantage for SAD-DQN on Acrobot-v1 and LunarLander-v2. From a theoretical perspective, we may expect MAD-DQN to perform better than SAD-DQN as it learns richer information about the distributions. Nevertheless, MAD-DQN displays a more oscillating behaviour and takes longer to converge. In addition, Figure 4 shows encouraging results for SAD-DQN over three different Atari games [5] (Pong, Breakout and Robotank) with the following parameters: $\gamma = 0.99$, $N = 51$, $\alpha = 0.8$, Adam with $lr = 5 \cdot 10^{-5}$ and batch size of 32. We compare to QR-DQN ($N = 51$), DQN and DOUBLE DQN [23]. On the three games, we reach almost the same performance as QR-DQN.

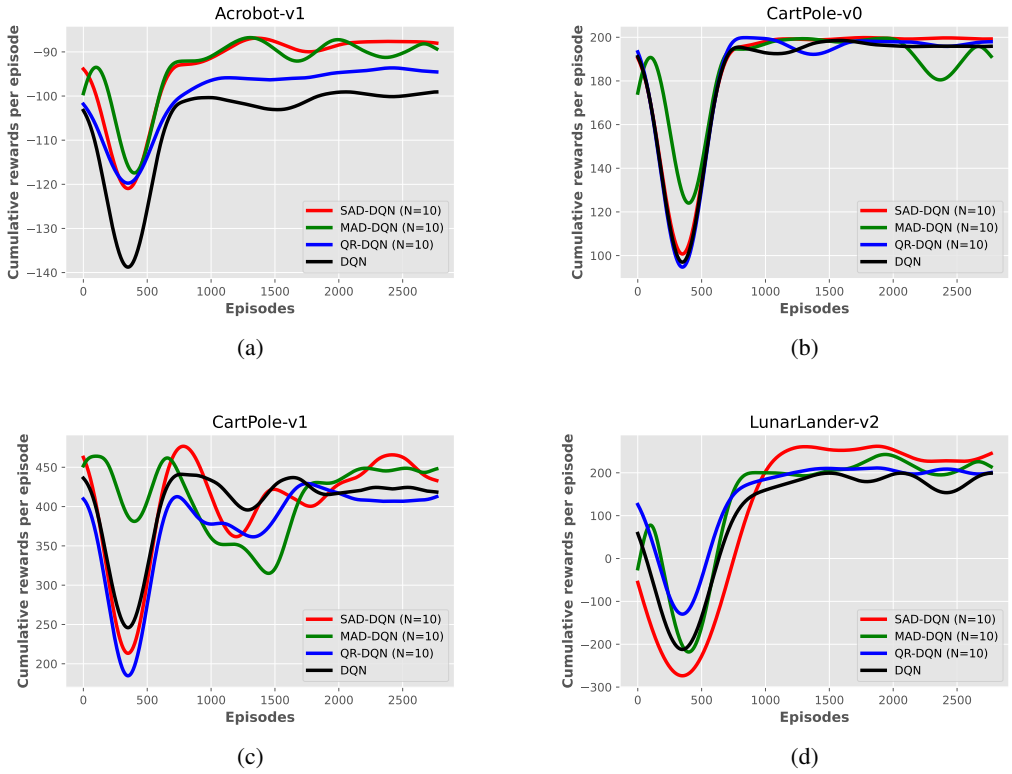

Figure 2: Comparison of the SAD-DQN and MAD-DQN algorithms against DQN and QR-DQN.

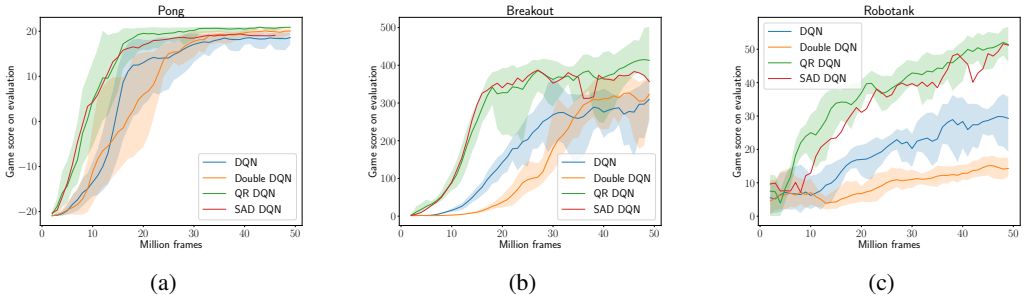

Figure 3: Performance on three Atari games.

5 Concluding remarks

We proposed two new deep and distributional RL algorithms that naturally extend DQN. In particular, they are both based on the squared loss akin to the original DQN loss. Moreover, the atomic targets satisfy a natural averaging property stated in Lemma 6, which paves the way for future theoretical convergence analysis. We empirically observed the convergence on various environments. Further research could investigate the performance gain on challenging environments such as Atari games [5], as well as the theoretical guarantees of the tabular algorithms described in Section 3.

References

[1] Achab, M. (2020). *Ranking and risk-aware reinforcement learning*. PhD thesis, Institut polytechnique de Paris.

[2] Achab, M. and Neu, G. (2021). Robustness and risk management via distributional dynamic programming. *arXiv preprint arXiv:2112.15430*.

[3] Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR.

[4] Bellemare, M. G., Le Roux, N., Castro, P. S., and Moitra, S. (2019). Distributional reinforcement learning with linear function approximation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2203–2211. PMLR.

[5] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

[6] Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

[7] Billingsley, P. (2013). *Convergence of probability measures*. John Wiley & Sons.

[8] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

[9] Dabney, W., Rowland, M., Bellemare, M., and Munos, R. (2018). Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[10] Dayan, P. (1992). The convergence of td (λ) for general λ. *Machine learning*, 8(3):341–362.

[11] Jaakkola, T., Jordan, M., and Singh, S. (1993). Convergence of stochastic iterative dynamic programming algorithms. *Advances in neural information processing systems*, 6.

[12] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[14] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

[15] Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010). Nonparametric return distribution approximation for reinforcement learning. In *ICML*.

[16] Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2012). Parametric return density estimation for reinforcement learning. *arXiv preprint arXiv:1203.3497*.

[17] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[18] Rockafellar, R. T., Uryasev, S., et al. (2000). Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42.

[19] Rowland, M., Bellemare, M., Dabney, W., Munos, R., and Teh, Y. W. (2018). An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 29–37. PMLR.

[20] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44.

[21] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[22] Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202.

[23] Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

[24] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 2.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]