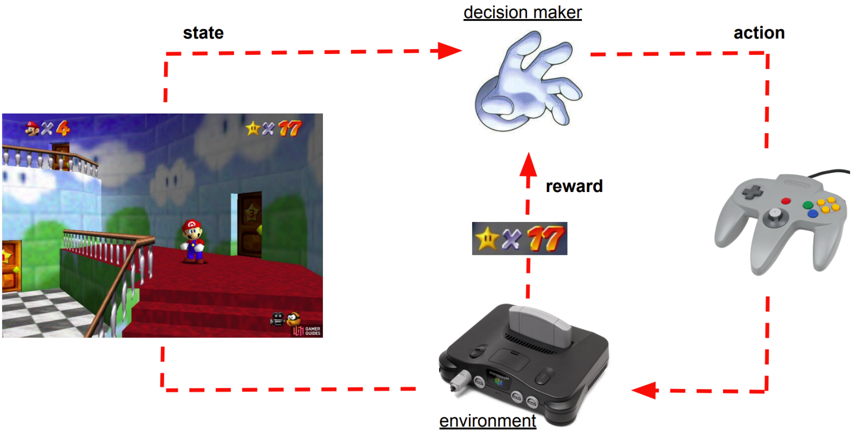
Distributional deep Q-learning with CVaR regression

M. Achab, R. Alami, Y. A. Dahou Djilali, K. Fedyanin, E. Moulines, M. Panov

Technology Innovation Institute, Masdar City, Abu Dhabi, United Arab Emirates

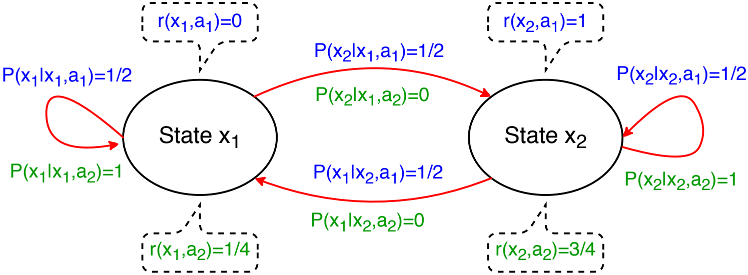
Deep Reinforcement Learning Workshop, NeurIPS 2022

Context: Sequential decision-making



Markov decision process (MDP)

An MDP [Puterman, 2014] is characterized by: states x, actions a, rewards $r(x, a, x')$ and transition probabilities $P(x'|x, a)$.



The control task

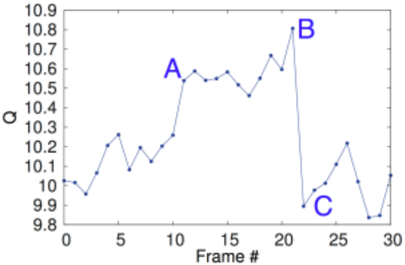
Optimality. Find a strategy π (mapping any state x to an action $\pi(x)$) that is optimal in terms of *expected* cumulative discounted return (for some discount factor $0 \leq \gamma < 1$):

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

with states $X_{t+1} \sim P(\cdot|X_t, A_t)$ and actions $A_{t+1} = \pi(X_{t+1})$.
Reinforcement learning (RL). Learn an optimal strategy without knowing the transitions probabilities $P(x'|x,a)$ or the reward function: an RL agent only observes empirical transitions (x_t, a_t, r_t, x_{t+1}).

Deep Q-Network (DQN)

The DQN agent [Mnih et al., 2013] learns Q^* with a deep neural net Q_θ with parameters θ: successfully plays Atari games!

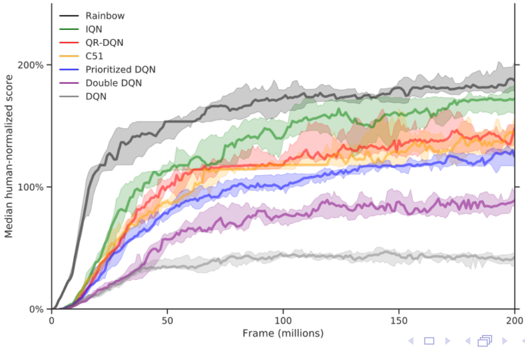


Distributional RL [Bellemare et al., 2017]

In distributional RL, the agent learns the whole probability distribution of the total return:

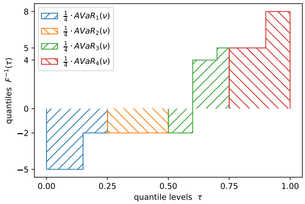
$$\text{Law}\left(\sum_{t\geq 0}\gamma^t r(X_t, A_t, X_{t+1}) \,\middle|\, X_0 = x, A_0 = a; \pi\right).$$

In contrast, RL only focuses on the expected value $Q^\pi(x,a)$ of this distribution. On Atari games, distributional RL outperforms RL!



Our approach: Wasserstein-2 projection

We use the W_2-projection to approximate distributions by a fixed number of values called “AVaRs” [Achab and Neu, 2021].



Algorithm 3 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:

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

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

Distributional DQN with AVaRs

We propose two new deep and distributional RL algorithms based on AVaR targets.

Algorithm 2 SAD-DQN update

Input: $(Q_{1;\theta}, \dots, Q_{N;\theta})$ 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:

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

Output: Updated parameters θ.

Algorithm 3 MAD-DQN update

Input: $(Q_{1;\theta}, \dots, Q_{N;\theta})$ 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:

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

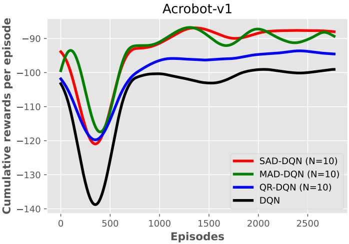
where $a^* \leftarrow \arg\max_{a'} Q_{\theta^-}(x', a')$

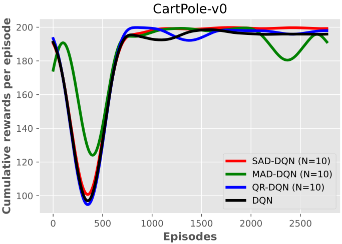
Perform a gradient descent step w.r.t. θ on the squared Wasserstein-2 loss function:

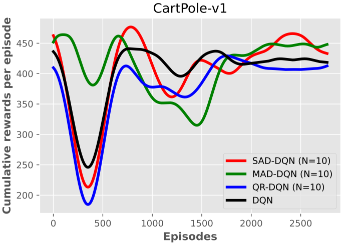
$$\theta \leftarrow \theta - \eta \nabla_\theta \frac{1}{N} \sum_{i=1}^{N} \left(Q_{i;\theta}(x,a) - \mathrm{AVaR}_i(\nu)\right)^2$$

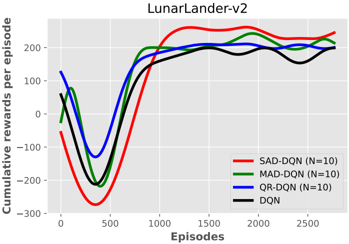
Output: Updated parameters θ.

Experiments - OpenAI Gym

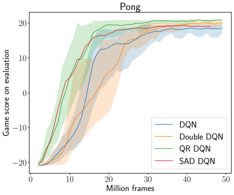




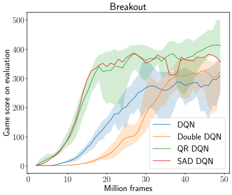




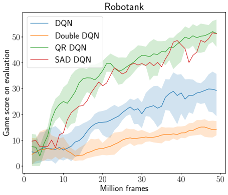
Experiments - Atari games



(a)



(b)



(c)

Figure: Performance on three Atari games.

References

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

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.

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.

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