Distributional RL II

Milan Straka

March 26, 2025

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research,
Development and Education

Distributional RL

Distributional RL

Instead of an expected return $Q(s, a)$, we could estimate the distribution of expected returns $Z(s, a)$ – the *value distribution*.

The authors define the distributional Bellman operator $\mathcal{T}^\pi$ as:

$$\mathcal{T}^\pi Z(s, a) \stackrel{\text{def}}{=} R(s, a) + \gamma Z(S', A') \quad \text{for} \quad S' \sim p(s, a), A' \sim \pi(S').$$

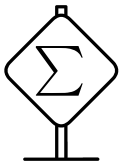
The authors of the paper prove similar properties of the distributional Bellman operator compared to the regular Bellman operator, mainly being a contraction under a suitable metric.

- For Wasserstein metric W_p, the authors define

  $$\bar{W}_p(Z_1, Z_2) \stackrel{\text{def}}{=} \sup_{s,a} W_p\big(Z_1(s, a), Z_2(s, a)\big)$$

  and prove that $\mathcal{T}^\pi$ is a γ-contraction in $\bar{W}_p$.
- However, $\mathcal{T}^\pi$ is not a contraction in KL divergence nor in total variation distance.

Wasserstein Metric



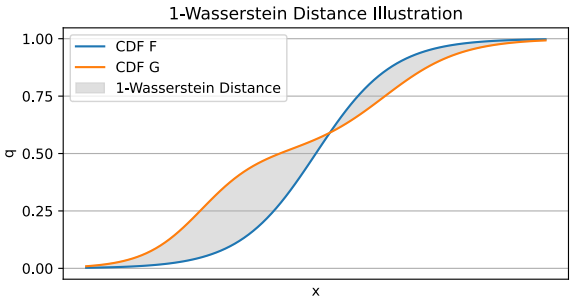
For two probability distributions μ, ν on a metric space with metric d, Wasserstein metric W_p is defined as

$$W_p(\mu,\nu) \stackrel{\text{def}}{=} \inf_{\gamma\in\Gamma(\mu,\nu)} \left(\mathbb{E}_{(x,y)\sim\gamma} d\big(x,y\big)^p\right)^{1/p},$$

where $\Gamma(\mu,\nu)$ is a set of all *couplings*, each being a joint probability distribution whose marginals are μ and ν, respectively. A possible intuition is the optimal transport of probability mass from μ to ν.

For distributions over reals with CDFs F, G, the optimal transport has an analytic solution:

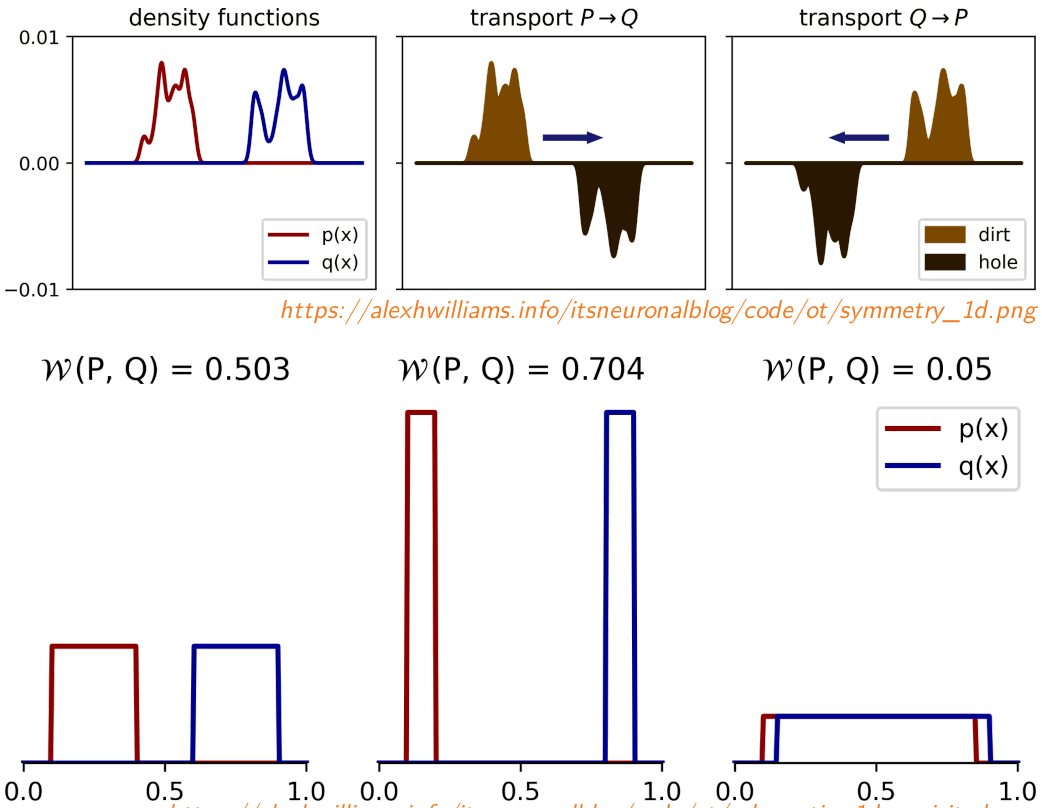
$$W_p(\mu,\nu) = \left(\int_0^1 |F^{-1}(q) - G^{-1}(q)|^p \,\mathrm{d}q\right)^{1/p},$$



where F^{-1} and G^{-1} are *quantile functions*, i.e., inverse CDFs.

For $p = 1$, the 1-Wasserstein metric correspond to area "between" F and G, and in that case we can compute it also as $W_1(\mu,\nu) = \int_x \big|F(x) - G(x)\big| \,\mathrm{d}x$.

Wasserstein Metric



https://alexhwilliams.info/itsneuronalblog/code/ot/symmetry_1d.png

https://alexhwilliams.info/itsneuronalblog/code/ot/schematic_1d_revisited.png

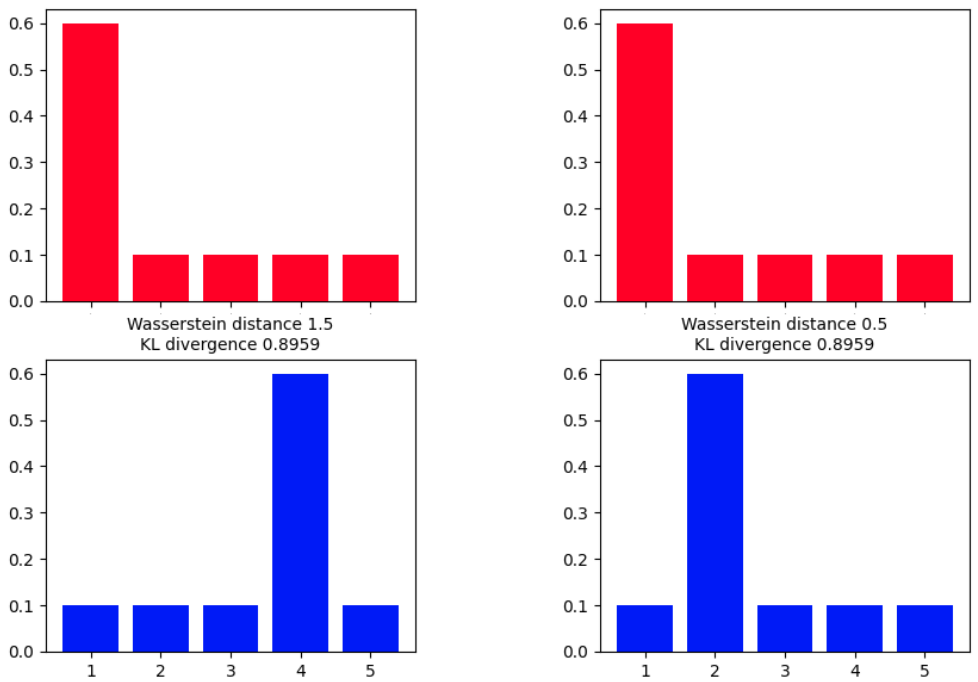


Fig. 1: Difference between Wasserstein distance and Kullbach-Leibler (KL) divergence.

Figure 1 of "WATCH: Wasserstein Change Point Detection for High-Dimensional Time Series Data", https://arxiv.org/abs/2201.07125

Distributional RL: C51 Refresh

The distribution of returns is modeled as a discrete distribution parametrized by the number of atoms $N \in \mathbb{N}$ and by $V_{\text{MIN}}, V_{\text{MAX}} \in \mathbb{R}$. Support of the distribution are atoms

$$\{z_i \stackrel{\text{def}}{=} V_{\text{MIN}} + i\Delta z : 0 \leq i < N\} \quad \text{for } \Delta z \stackrel{\text{def}}{=} \frac{V_{\text{MAX}} - V_{\text{MIN}}}{N - 1}.$$

The atom probabilities are predicted using a $\mathrm{softmax}$ distribution as

$$Z_{\boldsymbol{\theta}}(s, a) = \left\{ z_i \text{ with probability } p_i = \frac{e^{f_i(s,a;\boldsymbol{\theta})}}{\sum_j e^{f_j(s,a;\boldsymbol{\theta})}} \right\}.$$

Distributional RL: C51 Refresh

After the Bellman update, the support of the distribution $R(s,a)+\gamma Z(s',a')$ is not the same as the original support. We therefore project it to the original support by proportionally mapping each atom of the Bellman update to immediate neighbors in the original support.

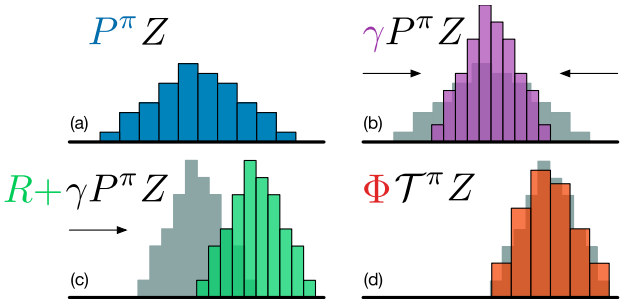


Figure 1 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

$$\Phi\big(R(s,a)+\gamma Z(s',a')\big)_i \stackrel{\text{def}}{=} \sum_{j=1}^{N}\left[1-\frac{\left|\left[r+\gamma z_j\right]_{V_{\text{MIN}}}^{V_{\text{MAX}}}-z_i\right|}{\Delta z}\right]_0^1 p_j(s',a').$$

The network is trained to minimize the Kullbeck-Leibler divergence between the current distribution and the (mapped) distribution of the one-step update

$$D_{\text{KL}}\left(\Phi\big(R+\gamma Z_{\bar{\boldsymbol{\theta}}}\big(s',\arg\max_{a'}\mathbb{E}Z_{\bar{\boldsymbol{\theta}}}(s',a')\big)\big)\middle\|Z_{\boldsymbol{\theta}}(s,a)\right).$$

Distributional RL: C51 Refresh

Algorithm 1 Categorical Algorithm

input A transition $x_t, a_t, r_t, x_{t+1}, \gamma_t \in [0, 1]$

$Q(x_{t+1}, a) := \sum_i z_i p_i(x_{t+1}, a)$

$a^* \leftarrow \arg\max_a Q(x_{t+1}, a)$

$m_i = 0, \quad i \in 0, \dots, N-1$

for $j \in 0, \dots, N-1$ **do**

Compute the projection of $\hat{\mathcal{T}} z_j$ onto the support $\{z_i\}$

$\hat{\mathcal{T}} z_j \leftarrow [r_t + \gamma_t z_j]_{V_{\text{MIN}}}^{V_{\text{MAX}}}$

$b_j \leftarrow (\hat{\mathcal{T}} z_j - V_{\text{MIN}})/\Delta z \quad \# \, b_j \in [0, N-1]$

$l \leftarrow \lfloor b_j \rfloor, u \leftarrow \lceil b_j \rceil$

Distribute probability of $\hat{\mathcal{T}} z_j$

$m_l \leftarrow m_l + p_j(x_{t+1}, a^*)(u - b_j)$

$m_u \leftarrow m_u + p_j(x_{t+1}, a^*)(b_j - l)$

end for

output $-\sum_i m_i \log p_i(x_t, a_t)$ # Cross-entropy loss

Algorithm 1 of "A Distributional Perspective on Reinforcement Learning" by Marc G. Bellemare et al.

Beware that there is a small bug in the original algorithm (on the left, taken from the paper), improperly handling one special case.

Note that by minimizing the D_{KL} instead of the Wasserstein metric W_p, the algorithm has no guarantee of convergence of any kind. However, the authors did not know how to minimize it.

Quantile Regression

Distributional RL with Quantile Regression

Although the authors of C51 proved that the distributional Bellman operator is a contraction with respect to Wasserstein metric W_p, they were not able to actually minimize it during training; instead, they minimize the KL divergence between the current value distribution and one-step estimate.

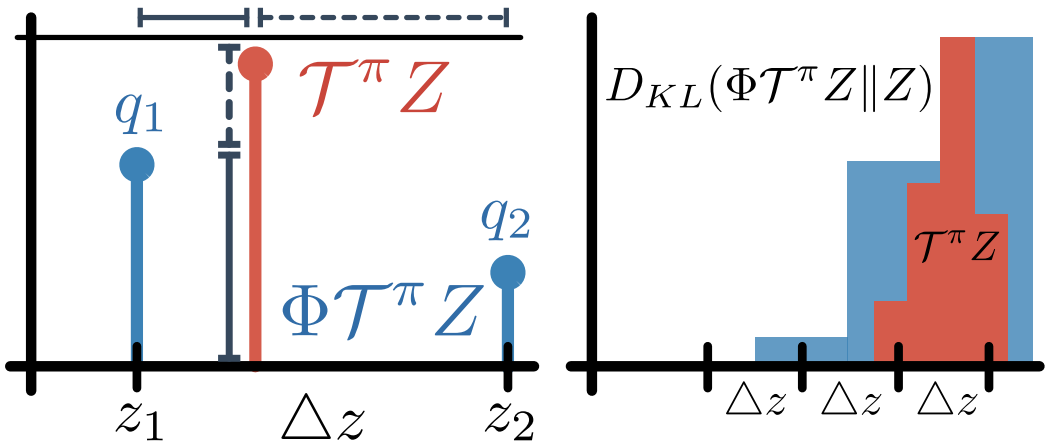


Figure 1: Projection used by C51 assigns mass inversely proportional to distance from nearest support. Update minimizes KL between projected target and estimate.

Figure 1 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Distributional RL with Quantile Regression

The same authors later proposed a different approach, which actually manages to minimize the 1-Wasserstein distance.

In contrast to C51, where $Z(s,a)$ is represented using a discrete distribution on a fixed "comb" support of uniformly spaces locations, we now represent it as a *quantile distribution* – as quantiles $\theta_i(s,a)$ for a fixed probabilities $\tau_1, \dots, \tau_N$ with $\tau_i = \frac{i}{N}$.

Formally, we can define the quantile distribution as a uniform combination of N Diracs:

$$Z_\theta(s,a) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} \delta_{\theta_i(s,a)},$$

so that the cumulative density function is a step function increasing by $\frac{1}{N}$ on every quantile θ_i.

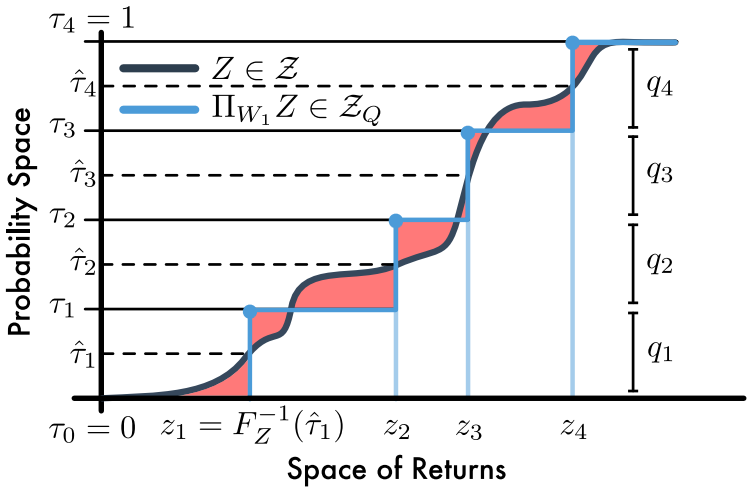


Figure 2: 1-Wasserstein minimizing projection onto $N = 4$ uniformly weighted Diracs. Shaded regions sum to form the 1-Wasserstein error.

Modified Figure 2 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Distributional RL with Quantile Regression

The quantile distribution offers several advantages:

- a fixed support is no longer required;
- the projection step Φ is not longer needed;
- this parametrization enables direct minimization of the Wasserstein loss.

Distributional RL with Quantile Regression

Recall that 1-Wasserstein distance between two distributions μ, ν can be computed as

$$W_1(\mu, \nu) = \int_0^1 \left| F_\mu^{-1}(q) - F_\nu^{-1}(q) \right| \mathrm{d}q,$$

where F_μ, F_ν are their cumulative density functions.

For arbitrary distribution Z, the we denote the most accurate quantile distribution as

$$\Pi_{W_1} Z \stackrel{\text{def}}{=} \arg\min_{Z_\theta} W_1(Z, Z_\theta).$$

In this case, the 1-Wasserstein distance can be written as

$$W_1(Z, Z_\theta) = \sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} \left| F_Z^{-1}(q) - \theta_i \right| \mathrm{d}q.$$

Distributional RL with Quantile Regression

It can be proven that for continuous F_Z^{-1}, $W_1(Z, Z_\theta)$ is minimized by (for proof, see Lemma 2 of Dabney et al.: Distributional Reinforcement Learning with Quantile Regression, or consider how the 1-Wasserstein distance changes in the range $[\tau_{i-1}, \tau_i]$ when you move θ_i):

$$\left\{ \theta_i \in \mathbb{R} \middle| F_Z(\theta_i) = \frac{\tau_{i-1} + \tau_i}{2} \right\}.$$

We denote the *quantile midpoints* as

$$\hat{\tau}_i \stackrel{\text{def}}{=} \frac{\tau_{i-1} + \tau_i}{2}.$$

In the paper, the authors prove that the composition $\Pi_{W_1} \mathcal{T}^\pi$ is γ-contraction in $\bar{W}_\infty$, so repeated application of $\Pi_{W_1} \mathcal{T}^\pi$ converges to a unique fixed point.

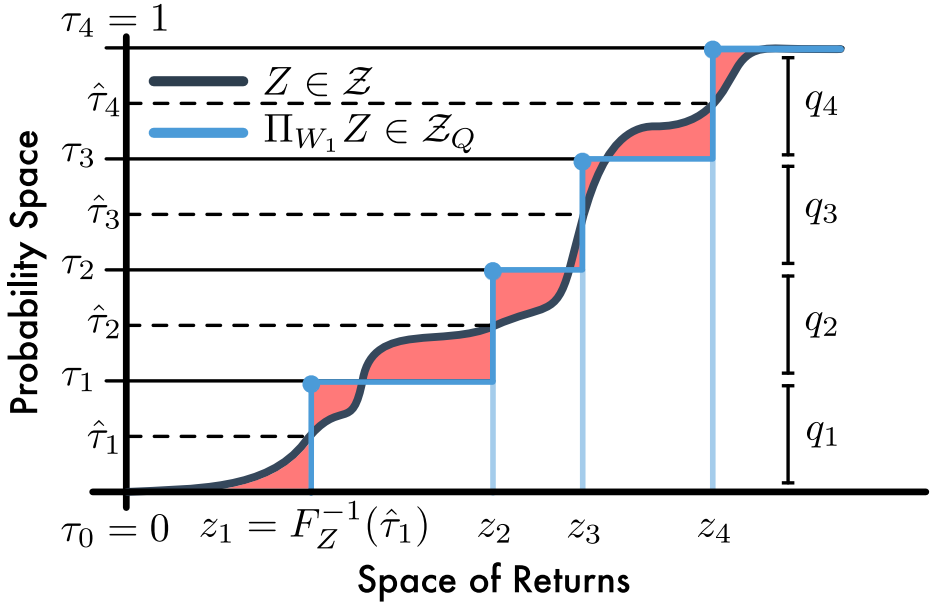


Figure 2: 1-Wasserstein minimizing projection onto $N = 4$ uniformly weighted Diracs. Shaded regions sum to form the 1-Wasserstein error.

Modified Figure 2 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Quantile Regression

Our goal is now to show that it is possible to estimate a quantile $\tau \in [0, 1]$ by minimizing a loss suitable for SGD.

Assume we have samples from a distribution P.

- Minimizing the MSE of $\hat{x}$ and the samples of P,

$$\tilde{x} = \arg\min_{\hat{x}} \mathbb{E}_{x \sim P}\left[(x - \hat{x})^2\right],$$

  yields the *mean* of the distribution, $\tilde{x} = \mathbb{E}_{x \sim P}[x]$.

  To show that this holds, we compute the derivative of the loss with respect to $\hat{x}$ and set it to 0, arriving at

$$0 = \mathbb{E}_x[2(\hat{x} - x)] = 2\mathbb{E}_x[\hat{x}] - 2\mathbb{E}_x[x] = 2\big(\hat{x} - \mathbb{E}_x[x]\big).$$

Quantile Regression

Assume we have samples from a distribution P with cumulative density function F_P.

- Minimizing the mean absolute error (MAE) of $\hat{x}$ and the samples of P,

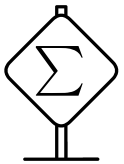
$$\tilde{x} = \arg\min_{\hat{x}} \mathbb{E}_{x \sim P}\big[|x - \hat{x}|\big],$$

yields the *median* of the distribution, $\tilde{x} = F_P^{-1}(0.5)$.

We prove this again by computing the derivative with respect to $\hat{x}$, assuming the functions are nice enough that the Leibnitz integral rule can be used:

$$\begin{aligned}
\frac{\partial}{\partial \hat{x}} \int_{-\infty}^{\infty} P(x)|x - \hat{x}|\,\mathrm{d}x &= \frac{\partial}{\partial \hat{x}}\left[\int_{-\infty}^{\hat{x}} P(x)(\hat{x} - x)\,\mathrm{d}x + \int_{\hat{x}}^{\infty} P(x)(x - \hat{x})\,\mathrm{d}x\right] \\
&= \int_{-\infty}^{\hat{x}} P(x)\,\mathrm{d}x - \int_{\hat{x}}^{\infty} P(x)\,\mathrm{d}x \\
&= 2\int_{-\infty}^{\hat{x}} P(x)\,\mathrm{d}x - 1 = 2F_P(\hat{x}) - 1 = 2\big(F_P(\hat{x}) - \tfrac{1}{2}\big).
\end{aligned}$$

Leibniz integral rule



The Leibniz integral rule for differentiation under the integral sign states that for $-\infty < a(x), b(x) < \infty$,

$$\frac{\partial}{\partial x}\left[\int_{a(x)}^{b(x)} f(x,t)\,\mathrm{d}t\right] =$$

$$= \int_{a(x)}^{b(x)} \frac{\partial}{\partial x} f(x,t)\,\mathrm{d}t + \left(\frac{\partial}{\partial x} b(x)\right) f\big(x, b(x)\big) - \left(\frac{\partial}{\partial x} a(x)\right) f\big(x, a(x)\big).$$

Sufficient condition for the Leibnitz integral rule to hold is that the $f(x,y)$ and its partial derivative $\frac{\partial}{\partial x} f(x,y)$ are continuous in both x and t, and $a(x)$ and $b(x)$ are continuous and have continuous derivatives.

If any of the bounds is improper, additional conditions must hold, notably that the integral of the partial derivatives of f must converge.

Assume we have samples from a distribution P with cumulative density function F_P.

- By generalizing the previous result, we can show that for a quantile $\tau \in [0, 1]$, if

$$\tilde{x} = \arg\min_{\hat{x}} \mathbb{E}_{x \sim P}\Big[(x - \hat{x})(\tau - [x \leq \hat{x}])\Big],$$

then $\tilde{x} = F_P^{-1}(\tau)$. Let $\rho_\tau(x - \hat{x}) \stackrel{\text{def}}{=} (x - \hat{x})(\tau - [x \leq \hat{x}]) = |x - \hat{x}| \cdot |\tau - [x \leq \hat{x}]|$. This loss penalizes overestimation errors with weight $1 - \tau$, underestimation errors with τ.

$$\frac{\partial}{\partial \hat{x}} \int_{-\infty}^{\infty} P(x)(x - \hat{x})(\tau - [x \leq \hat{x}])\,\mathrm{d}x =$$

$$= \frac{\partial}{\partial \hat{x}} \left[(\tau - 1) \int_{-\infty}^{\hat{x}} P(x)(x - \hat{x})\,\mathrm{d}x + \tau \int_{\hat{x}}^{\infty} P(x)(x - \hat{x})\,\mathrm{d}x\right]$$

$$= (1 - \tau) \int_{-\infty}^{\hat{x}} P(x)\,\mathrm{d}x - \tau \int_{\hat{x}}^{\infty} P(x)\,\mathrm{d}x = \int_{-\infty}^{\hat{x}} P(x)\,\mathrm{d}x - \tau = F_P(\hat{x}) - \tau.$$

Quantile Regression

Using the quantile regression, when we have a value distribution Z, we can find the most accurate quantile distribution by minimizing

$$\sum_{i=1}^{N} \mathbb{E}_{z\sim Z}\left[\rho_{\hat{\tau}_i}(z-\theta_i)\right].$$

However, the quantile loss is not smooth around zero, which could limit performance when training a model. The authors therefore propose the **quantile Huber loss**, which acts as an asymmetric squared loss in interval $[-\kappa,\kappa]$ and fall backs to the standard quantile loss outside this range.

Specifically, let

$$\rho_\tau^\kappa(z-\theta) \stackrel{\text{def}}{=} \begin{cases} \left|\tau - [z\le\theta]\right| \cdot \frac{1}{2}\left(z-\theta\right)^2 & \text{if } |z-\theta|\le\kappa, \\ \left|\tau - [z\le\theta]\right| \cdot \kappa\left(|z-\theta| - \frac{1}{2}\kappa\right) & \text{otherwise.} \end{cases}$$

Distributional RL with Quantile Regression

To conclude, in DR-DQN-κ, the network for a given state predicts $\mathbb{R}^{|\mathcal{A}| \times N}$, so N quantiles for every action.

The following loss is used:

Algorithm 1 Quantile Regression Q-Learning

Require: N, κ

input $x, a, r, x', \gamma \in [0, 1)$

Compute distributional Bellman target

$Q(x', a') := \sum_j q_j \theta_j(x', a')$

$a^* \leftarrow \arg\max_{a'} Q(x', a')$

$\mathcal{T}\theta_j \leftarrow r + \gamma \theta_j(x', a^*), \quad \forall j$

Compute quantile regression loss (Equation 10)

output $\sum_{i=1}^{N} \mathbb{E}_j \left[\rho^{\kappa}_{\hat{\tau}_i} (\mathcal{T}\theta_j - \theta_i(x, a)) \right]$

Modification of Algorithm 1 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

The q_j is just $\frac{1}{N}$.

Distributional RL with Quantile Regression

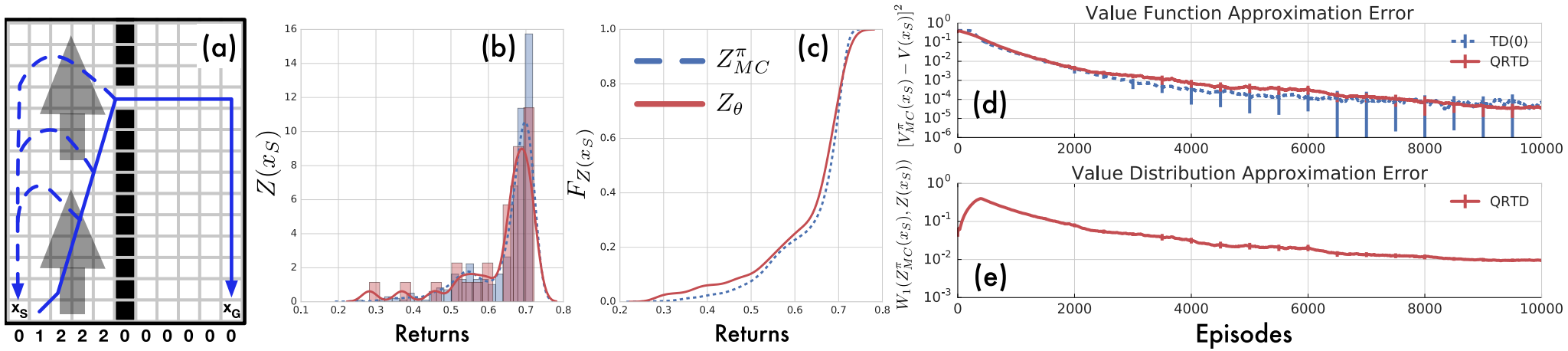


Figure 3: (a) Two-room windy gridworld, with wind magnitude shown along bottom row. Policy trajectory shown by blue path, with additional cycles caused by randomness shown by dashed line. (b, c) (Cumulative) Value distribution at start state x_S, estimated by MC, Z^π_{MC}, and by QRTD, Z_θ. (d, e) Value function (distribution) approximation errors for TD(0) and QRTD.

Figure 3 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Each state transition has probability of 0.1 of moving in a random direction.

Distributional RL with Quantile Regression

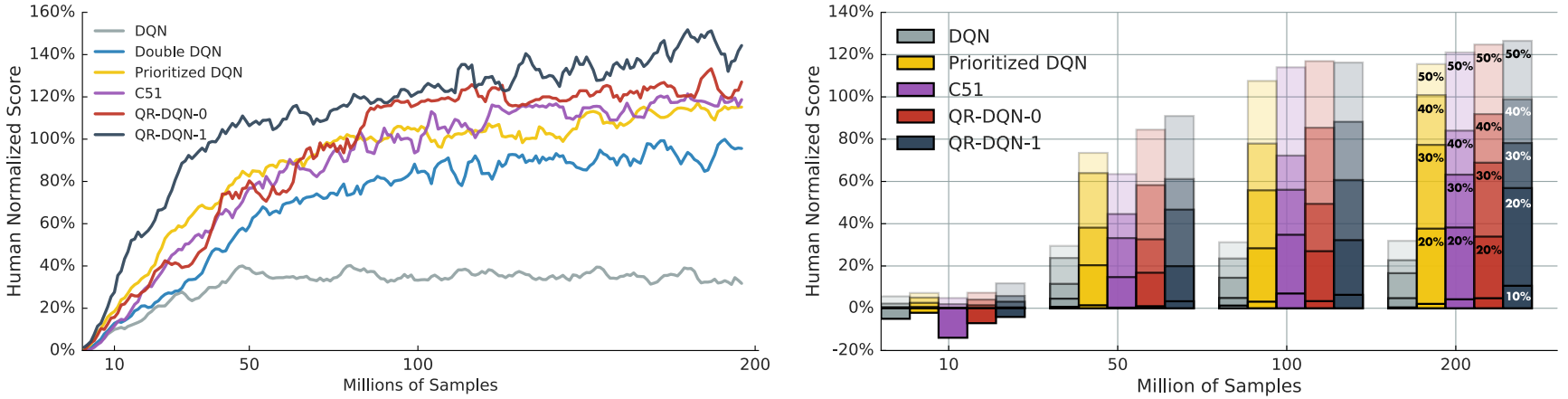


Figure 4: Online evaluation results, in human-normalized scores, over 57 Atari 2600 games for 200 million training samples. (Left) Testing performance for one seed, showing median over games. (Right) Training performance, averaged over three seeds, showing percentiles (10, 20, 30, 40, and 50) over games.

Figure 4 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

	Mean	Median	>human	>DQN
DQN	228%	79%	24	0
DDQN	307%	118%	33	43
DUEL.	373%	151%	37	50
PRIOR.	434%	124%	39	48
PR. DUEL.	592%	172%	39	44
C51	701%	178%	40	50
QR-DQN-0	881%	199%	38	52
QR-DQN-1	**915%**	**211%**	**41**	**54**

Table 1 of "Distributional Reinforcement Learning with Quantile Regression", https://arxiv.org/abs/1710.10044

Hyperparameter	Value
learning rate	0.00005
quantiles N	200

N chosen from $(10, 50, 100, 200)$ on 5 training games.

Implicit Quantile Regression

Implicit Quantile Networks for Distributional RL

In IQN (implicit quantile regression), the authors (again the same team as in C51 and DR-DQN) generalize the value distribution representation to predict *any given quantile* τ.

- The $\psi(s)$ is a convolutional stack from DQN, composed of
  - CNN 8×8, stride 4, 32 filters, ReLU;
  - CNN 4×4, stride 2, 64 filters, ReLU;
  - CNN 3×3, stride 1, 64 filters, ReLU.
- The f is an MLP:
  - fully connected layer with 512 units, ReLU;
  - output layer, 1 unit.

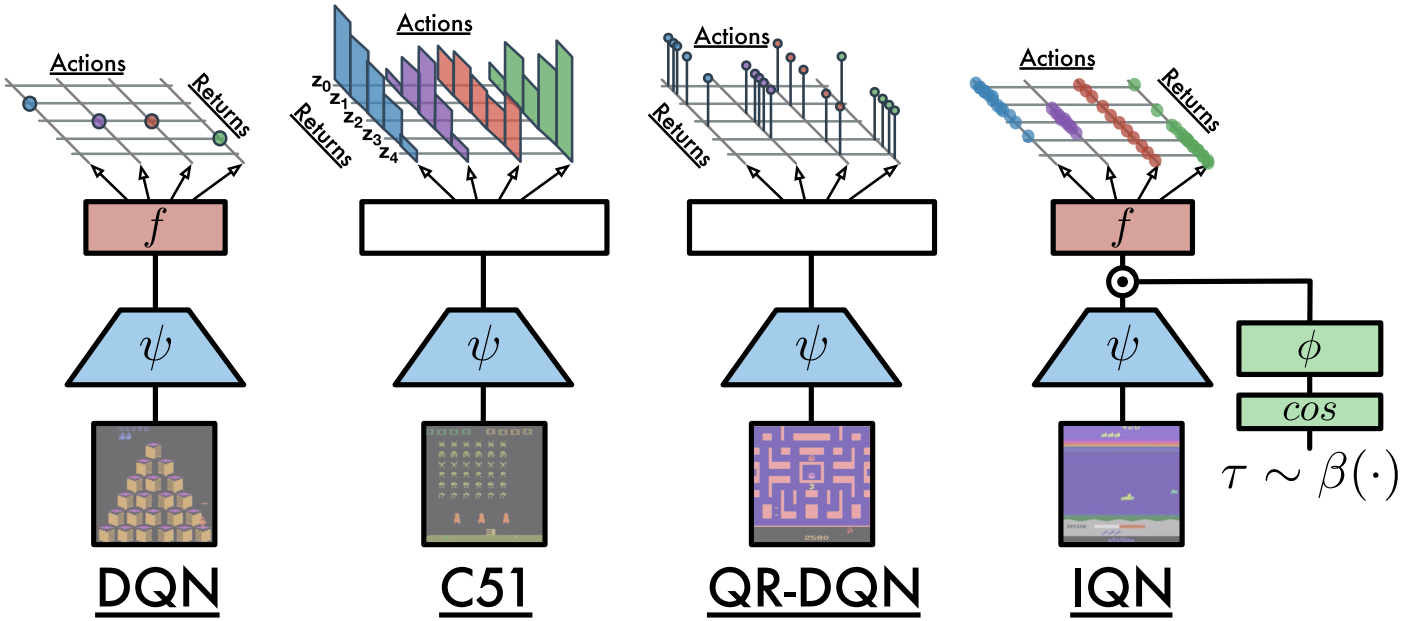


Figure 1. Network architectures for DQN and recent distributional RL algorithms.

Figure 1 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

Implicit Quantile Networks for Distributional RL

The quantile τ of the value distribution, $Z_\tau(s, a)$, is modeled as

$$Z_\tau(s, a) \approx f\big(\psi(s) \odot \varphi(\tau)\big)_a.$$

- Other ways than multiplicative combinations were tried (concatenation, or residual computation $\psi(s) \odot (1 + \varphi(\tau))$), but the multiplicative form delivered the best results.
- The quantile τ is represented using trainable cosine embeddings with dimension $n = 64$:

$$\varphi_j(\tau) \stackrel{\text{def}}{=} \text{ReLU}\left(\sum_{i=0}^{n-1} \cos(\pi i \tau) w_{i,j} + b_j\right).$$

- The target policy is greedy with respect to action-value approximation computed using K samples $\tilde{\tau}_k \sim U[0, 1]$:

$$\pi(x) \stackrel{\text{def}}{=} \arg\max_a \frac{1}{K} \sum_{k=1}^{K} Z_{\tilde{\tau}_k}(x, a).$$

  - As in DQN, the exploration is still performed by using the ε-greedy policy.

Implicit Quantile Networks for Distributional RL

The overall loss is:

Algorithm 1 Implicit Quantile Network Loss

Require: N, N', K, κ and functions β, Z

input $x, a, r, x', \gamma \in [0, 1)$

Compute greedy next action

$a^* \leftarrow \arg\max_{a'} \frac{1}{K} \sum_k^K Z_{\tilde{\tau}_k}(x', a'), \quad \tilde{\tau}_k \sim \beta(\cdot)$

Sample quantile thresholds

$\tau_i, \tau'_j \sim U([0, 1]), \quad 1 \leq i \leq N, 1 \leq j \leq N'$

Compute distributional temporal differences

$\delta_{ij} \leftarrow r + \gamma Z_{\tau'_j}(x', a^*) - Z_{\tau_i}(x, a), \quad \forall i, j$

Compute Huber quantile loss

output $\sum_{i=1}^N \mathbb{E}_{\tau'} \left[\rho_{\tau_i}^{\kappa}(\delta_{ij})\right]$

Algorithm 1 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

Note the different roles of N and N'.

Implicit Quantile Networks for Distributional RL

The authors speculate that:

- large N may increase sample complexity (faster learning because we have more loss terms),
- larger N' could reduce variance (like a minibatch size).

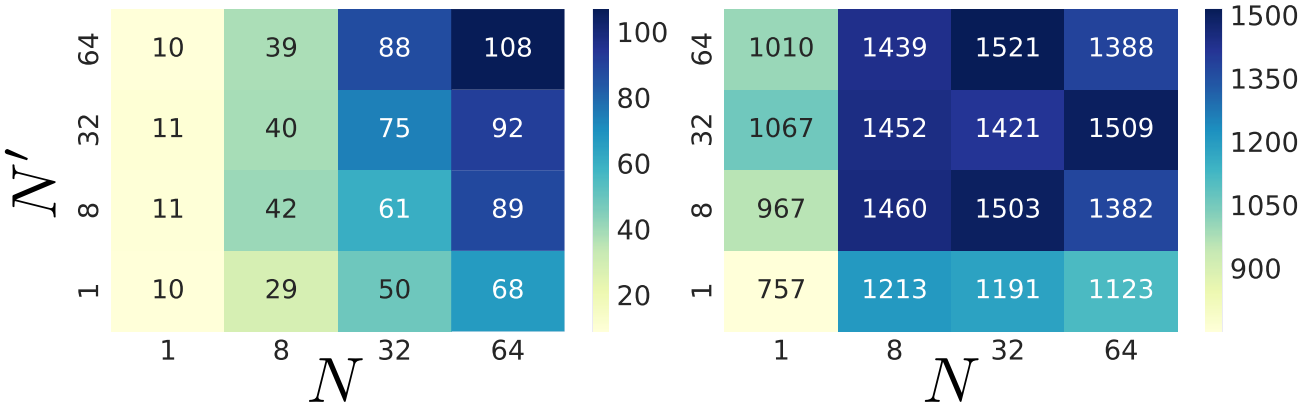


Figure 2. Effect of varying N and N', the number of samples used in the loss function in Equation 3. Figures show human-normalized agent performance, averaged over six Atari games, averaged over first 10M frames of training (left) and last 10M frames of training (right). Corresponding values for baselines: DQN $(32, 253)$ and QR-DQN $(144, 1243)$.

Figure 2 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

Implicit Quantile Networks for Distributional RL

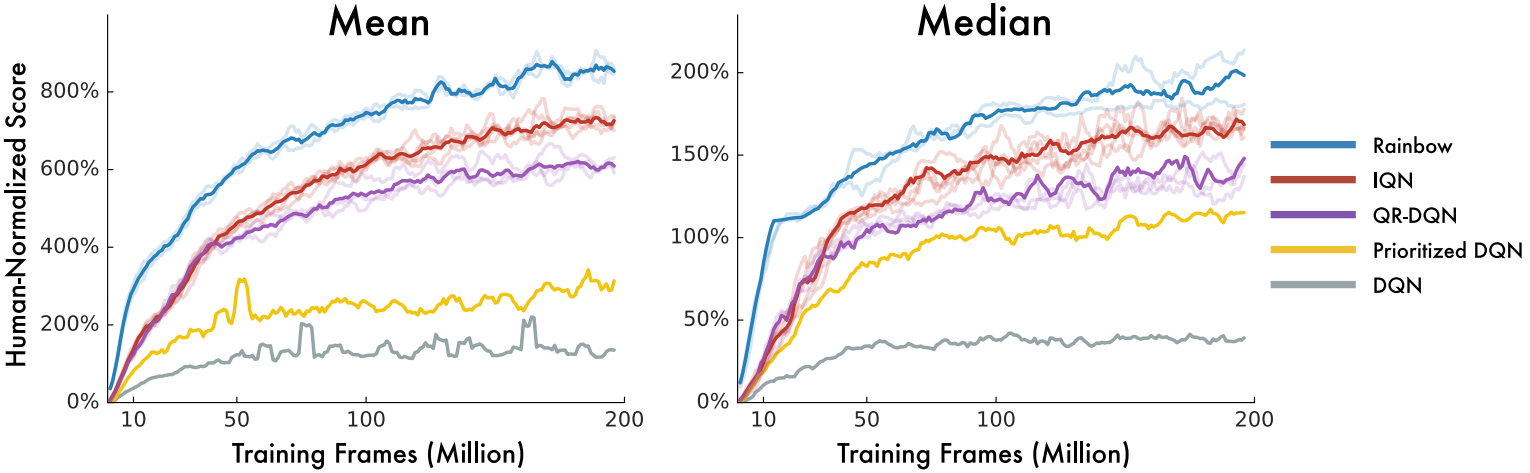


Figure 4. Human-normalized mean (left) and median (right) scores on Atari-57 for IQN and various other algorithms. Random seeds shown as traces, with IQN averaged over 5, QR-DQN over 3, and Rainbow over 2 random seeds.

Figure 4 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

	Mean	Median	Human Gap	Seeds
DQN	228%	79%	0.334	1
PRIOR.	434%	124%	0.178	1
C51	701%	178%	0.152	1
RAINBOW	**1189%**	**230%**	0.144	2
QR-DQN	864%	193%	0.165	3
IQN	1019%	218%	**0.141**	5

Table 1. Mean and median of scores across 57 Atari 2600 games, measured as percentages of human baseline (Nair et al., 2015). Scores are averages over number of seeds.

Table 1 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

Human-starts (median)					
DQN	PRIOR.	A3C	C51	RAINBOW	IQN
68%	128%	116%	125%	153%	**162%**

Table 2. Median human-normalized scores for human-starts.

Table 2 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

Implicit Quantile Networks for Distributional RL

The ablation experiments of the quantile representation. A full grid search with two seeds for every configuration was performed, with the black dots corresponding to the hyperparameters of IQN; six Atari games took part in the evaluation.

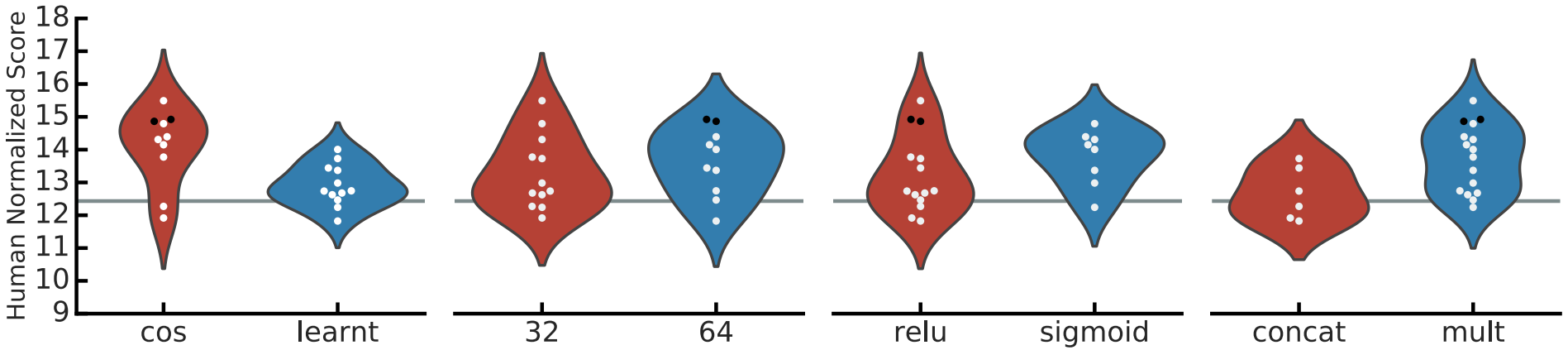
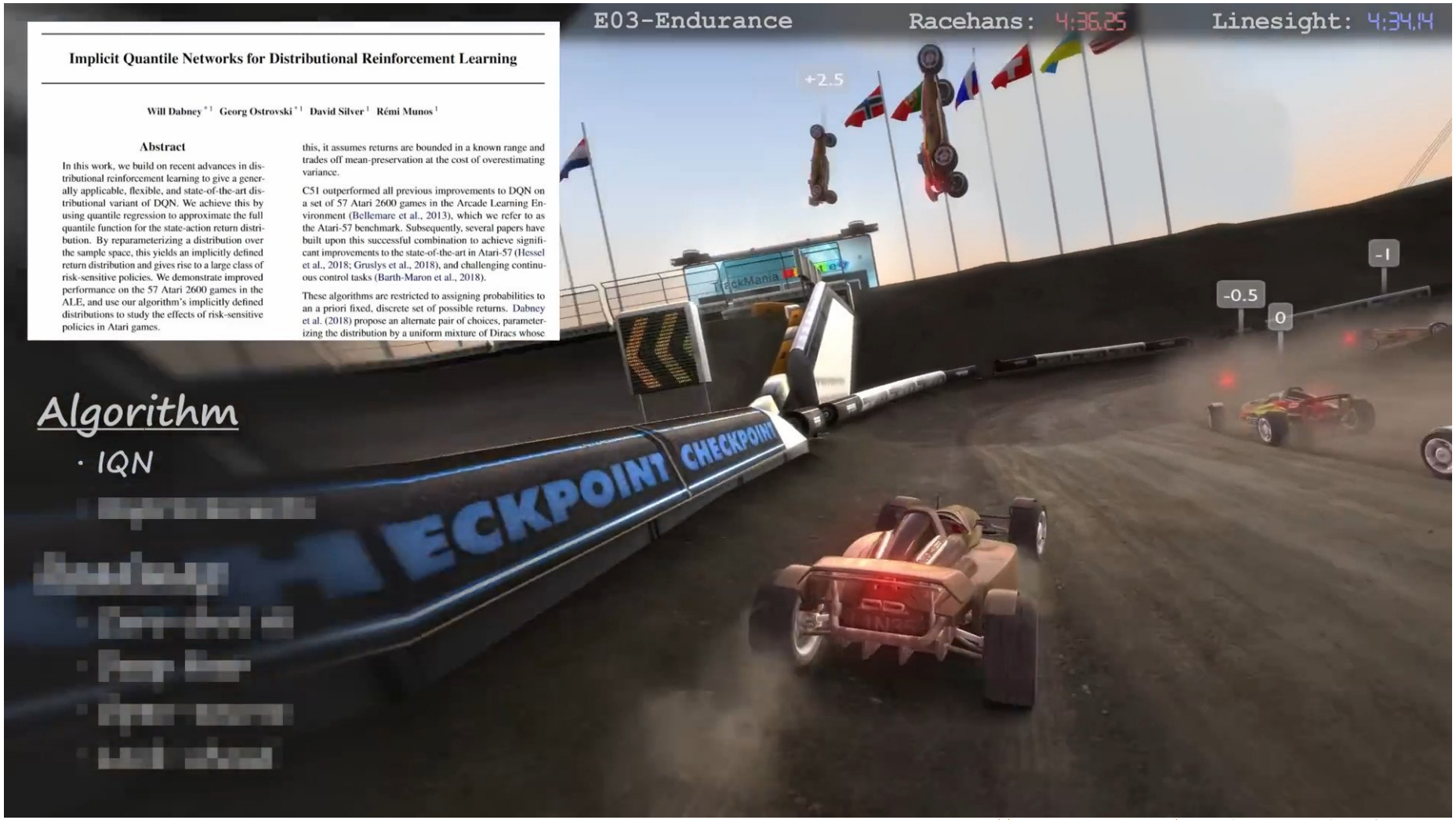


Figure 5. Comparison of architectural variants.

Figure 5 of "Implicit Quantile Networks for Distributional Reinforcement Learning", https://arxiv.org/abs/1806.06923

- the gray horizontal line is the QR-DQN baseline;
- "learn" is a learnt MLP embedding with a single hidden layer of size n;
- "concat" combines the state and quantile representations by concatenation, not $\odot$.

TrackMania using Implicit Quantile Networks



https://www.youtube.com/watch?v=cUojVsCJ51I&t=1342s

Policy Gradient Methods

Policy Gradient Methods

Instead of predicting expected returns, we could train the method to directly predict the policy

$$\pi(a|s;\boldsymbol{\theta}).$$

Obtaining the full distribution over all actions would also allow us to sample the actions according to the distribution π instead of just ε-greedy sampling.

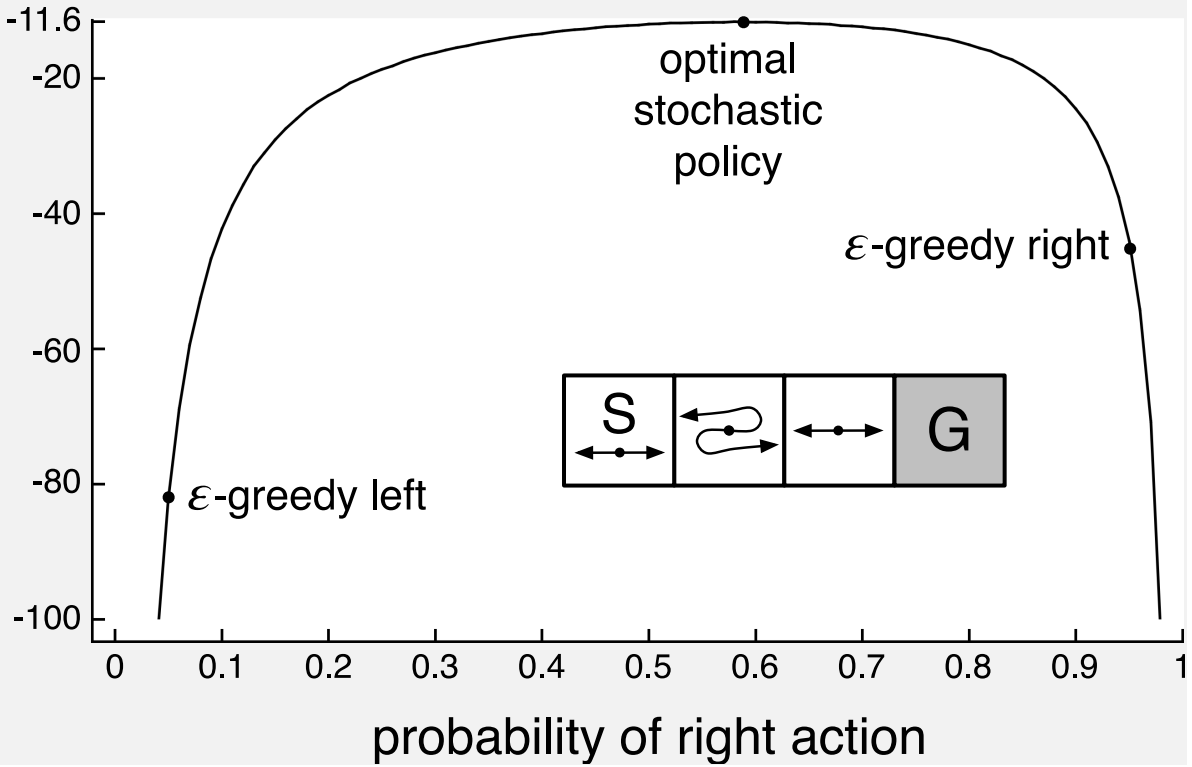
However, to train the network, we maximize the expected return $v_\pi(s)$ and to that account we need to compute its *gradient* $\nabla_{\boldsymbol{\theta}} v_\pi(s)$.

Policy Gradient Methods

In addition to discarding ε-greedy action selection, policy gradient methods allow producing policies which are by nature stochastic, as in card games with imperfect information, while the action-value methods have no natural way of finding stochastic policies (distributional RL might be of some use though).

In the example, the reward is -1 per step, and we assume the three states appear identical under the function approximation.

$$J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(\mathrm{S})$$



Example 13.1 of "Reinforcement Learning: An Introduction, Second Edition".

Policy Gradient Theorem

Let $\pi(a|s;\boldsymbol{\theta})$ be a parametrized policy. We denote the initial state distribution as $h(s)$ and the on-policy distribution under π as $\mu(s)$. Let also $J(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \mathbb{E}_{s\sim h} v_\pi(s)$.

Then

$$\nabla_{\boldsymbol{\theta}} v_\pi(s) \propto \sum_{s'\in\mathcal{S}} P(s\to\ldots\to s'|\pi) \sum_{a\in\mathcal{A}} q_\pi(s',a)\nabla_{\boldsymbol{\theta}}\pi(a|s';\boldsymbol{\theta})$$

and

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \sum_{s\in\mathcal{S}} \mu(s) \sum_{a\in\mathcal{A}} q_\pi(s,a)\nabla_{\boldsymbol{\theta}}\pi(a|s;\boldsymbol{\theta}),$$

where $P(s\to\ldots\to s'|\pi)$ is the probability of getting to state s' when starting from state s, after any number of 0, 1, ... steps. The γ parameter should be treated as a form of termination, i.e., $P(s\to\ldots\to s'|\pi) \propto \sum_{k=0}^{\infty} \gamma^k P(s\to s' \text{ in } k \text{ steps }|\pi)$.