UNIVERSITY OF MANNHEIM

ADDQ

Adaptive Distributional Double Q-learning

Leif Döring Benedikt Wille Maximilian Birr Mihail Bîrsan Martin Slowik

Institute of Mathematics
University of Mannheim

ICML 2025

Background

Setting: Model free optimization algorithms for discounted MDP problems. States $\mathcal{S}$, actions $\mathcal{A}$, state-reward transitions p, policies π.

Value functions: $V^\pi(s) = \mathbb{E}_s^\pi\left[\sum_{t=0}^\infty \gamma^t R_t\right]$ and $Q^\pi(s,a) = \mathbb{E}_{s,a}^\pi\left[\sum_{t=0}^\infty \gamma^t R_t\right]$.

Optimality: Policy π^* is called optimal if $V^{\pi^*}(s) = \sup_\pi V^\pi(s)$ for all $s \in \mathcal{S}$.

Theorem (Dynamic Programming): If Q^* is the unique solution to the Bellman optimality equation $T^*Q = Q$, then the greedy policy π^* is optimal. Here

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

Value-iteration: Banach fixed-point theorem for contraction T^*, convergence to Q^*.

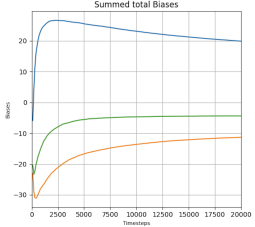
Q-learning: Stochastic approx. variant of value-iteration, a.s. convergence to Q^*.

Overestimation problem

Problem: Q-learning update

$$Q_{n+1}(s,a) = (1-\alpha)Q_n(s,a) + \alpha\Big(r + \gamma \max_{a'} Q_n(s',a')\Big)$$

does not give unbiased estimators of $Q^*(s,a)$. Estimates $Q_n(s,a)$ of $Q^*(s,a)$ are initially strongly overestimated.



Blue curve is sum of estimated Q-values in a grid world, see paper.

Explanation (in tabular setting): Estimating $\max_i \mathbb{E}[X_i]$ using the pointwise estimator $\max_i X_i$ yields a positive bias. Important: positive outliers overestimate the max, negative outliers don't.

Overestimation reduction: double Q, clipping (TD3), MaxMin, Ensemble Q, TQC, ...

Main ideas

Idea 1: Overestimation bias of $\max_i X_i$ depends strongly on outliers, e.g. large variances of random variables X_i lead to large positive biases. Thus, large variances of estimated Q-values $Q_n(s, a_1), ..., Q_n(s, a_n)$.

Idea 2: Use distributional Q-learning to learn distribution $\eta_n(s, a)$ behind $Q_n(s, a)$ and then use $\mathbb{V}(\eta(s, a))$ to decide whether need for overestimation control is large, or not.

$\rightarrow$ see paper for theoretical backup (η contains epistemic and systemic uncertainty)

Idea 3: Take some overestimation reduction method and locally (in terms of state-action pairs) adjust the correction. We built on double Q-learning (adaptive double Q-learing), other choices are possible (e.g. locally adjusting number of atoms in TQC).

Double Q-learning:

$$Q^{A/B}(s,a) \leftarrow \overbrace{(1-\alpha)Q^{A/B}(s,a) + \alpha\big(r + \gamma Q^{A/B}(s', z^*)\big)}^{Q\text{-learning update}} + \overbrace{\alpha(\gamma Q^{B/A}(s', z^*) - \gamma Q^{A/B}(s', z^*))}^{\text{bias correction}},$$

with $z^* = \operatorname{argmax} Q^{A/B}(s,a)$. Our modified bias correction:

$$\beta(s,a) \times \text{double Q-learning bias correction}$$

for locally adaptive β that depends on estimated variances ($\rightarrow$ see paper).

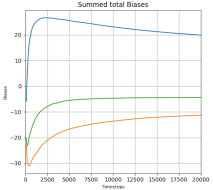
Idea: $\beta \approx 1$ gives double Q (for large variances), $\beta \approx 0$ gives Q (for small variances).

Results

Tabular example: Q and double-Q jointly suffer in examples that have local variability, ADDQ can adjust locally to the situation.

F	1	2	S
4	5	6	7
8	9	10	11
12	G	14	15

S=start,
G=goal,
F=fake goal,
gray=large variance



blue line: Q-learning (overestimation)
red line: double Q-learning (underestimation)
green line: ADDQ, local combination of Q and double Q

Deep RL: Pseudocode and experiments are provided in the paper. Experiments show improvements (RLiable metrics) over C51 and QRDQN on Atari benchmark.

Implementation: ADDQ requires the change of three lines of code in C51/QRDQN implementations. Computational overhead is minimal.

Conclusion

- We suggest to use distributional RL to optimize overestimation reduction algorithms.
- We suggest a simple to implement add-on to distributional RL algorithm. Take your overestimation method of choice and make it locally adaptive!
- Method works well on tabular examples, also in Stable-Baselines3 implementations of C51 and QRDQN.

Questions? Suggestions? See you at our poster!