The Role of Target Update Frequencies in Q-Learning

Simon Weissmann[1] Tilman Aach[1] Benedikt Wille[1] Sebastian Kassing[2]
Leif Döring[1]

[1]Institute of Mathematics
University of Mannheim

[2]Department of Mathematics & Informatics
University of Wuppertal

ICML 2026

Background

Setting: Discounted MDP with states $\mathcal{S}$, actions $\mathcal{A}$, transitions p, policies π.

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

Theorem (Dynamic Programming): If Q^* is the unique solution to the Bellman optimality equation $T^*Q = Q$, the derived greedy policy 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: $Q_{n+1} = T^*Q_n$ converges to Q^* since T^* is a contraction (BFT)

Approximate dynamic programming: In model-free RL, T^*Q_n cannot be computed exactly. ADP algorithms approximate Bellman updates in value iteration from samples.

Periodic Q-Learning

Periodic Q-Learning: Keep a fixed copy Q_n^- for K_n steps, nested update rule:

Inner Loop: $$Q_n(s,a) \leftarrow Q_n(s,a) + \alpha\big(r + \gamma \max_{a'} Q_n^-(s',a') - Q_n(s,a)\big)$$

Outer Loop: $$Q_{n+1}, Q_{n+1}^- = Q_n.$$

We call K_n cycle length and $1/K_n$ target update frequency (TUF).

Special cases: Constant K_n: DQN target update rule, constant $K_n = 1$: Q-Learning.

Classical DQN view: Target freezing is employed to stabilize bootstrapping.

Periodic Q-Learning view: Target freezing is ADP; the inner loop approximates

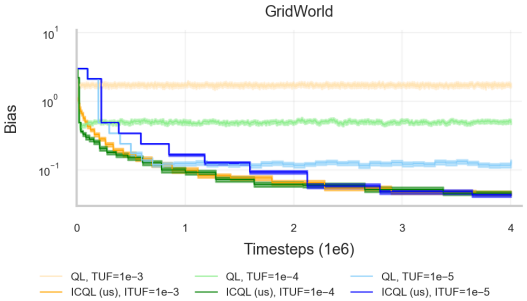
$$(T^*Q)(s,a) = \arg\min_{\theta \in \mathbb{R}} \mathbb{E}\big[(r + \gamma \max_{a'} Q(s',a') - \theta)^2\big]$$

Intuition

Problem: How should we choose the target update frequencies?

ADP: Inner cycle length K_n steers accuracy of Bellman operator approximation

Idea: Early on, contraction dominates; close to Q^*, approximation error dominates.



Small K_n: Cheap but inaccurate updates;
Fast decrease, but plateaus at a high level.

Large K_n: Expensive but accurate updates;
Slow decrease but plateaus at a low level.

ICQL: Start with short cycles and decrease TUF over training.

Optimal TUF

Goal: Choose (K_n) minimizing sample complexity while achieving target accuracy ε:

$$\min_{N,(K_n)} \sum_{j=0}^{N-1} K_j \qquad \text{s.t.} \qquad e_N \leq \varepsilon, \qquad \text{where} \qquad e_N := \mathbb{E}\big[\|Q_N^{(0)} - Q^*\|_\infty\big].$$

Error decomposition: $e_N \leq \mu^N e_0 + \sum_{j=0}^{N-1} \mu^{N-1-j} \sqrt{\frac{c}{K_j}}$ with $\mu = \frac{1+\gamma}{2} < 1$.

Interpretation: Errors are exponentially discounted by μ^{N-1-j}, thus Bellman updates should become increasingly accurate over time; geometric balancing attains optimality.

Optimal Fixed TUF: $$\mathcal{O}\left(\frac{\log((1-\gamma)^{-1}\varepsilon^{-1})}{\xi^2(1-\gamma)^5\varepsilon^2}\right)$$

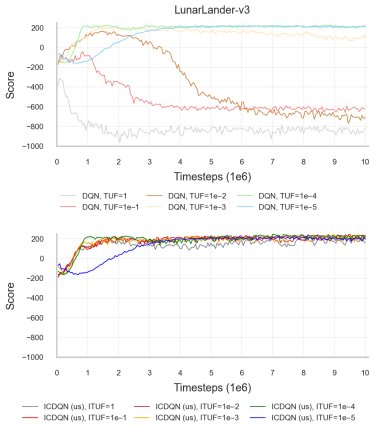
Optimal geometrically decreasing TUF: $$\mathcal{O}\left(\frac{1}{\xi^2(1-\gamma)^5\varepsilon^2}\right)$$

Experiments

Tabular: High TUFs reduce error quickly, plateau higher. Optimal TUFs fast initial learning, continue improving.

Deep RL with SGD: ICDQN robust across tested initial TUFs, DQN highly sensitive to choice of TUF.

Beyond theory: Preliminary experiments on ADAM and other ADP algos promising; Additional suggestion: Accuracy-Triggered Target Updates.



Implementation: Straightforward to implement, no computational overhead.

Conclusion

- Target freezing is a principled ADP design problem.
- Target Update Frequencies should decrease geometrically.
- Experiments suggest higher stability across initial TUFs
- Simple to implement, no computational overhead, applicable in many ADP algos.

Questions? Suggestions? See you at our poster!