ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Is Q-learning an Ill-posed Problem?

Philipp Wissmann[1,2], Daniel Hein[1], Steffen Udluft[1], and Thomas Runkler[1,2] *

1- Siemens AG, Munich, Germany

2- TU Munich (TUM), Munich, Germany

Abstract. This paper investigates the instability of Q-learning in continuous environments, a challenge frequently encountered by practitioners. Traditionally, this instability is attributed to bootstrapping and regression model errors. Using a representative reinforcement learning benchmark, we systematically examine the effects of bootstrapping and model inaccuracies by incrementally eliminating these potential error sources. Our findings reveal that even in relatively simple benchmarks, the fundamental task of Q-learning – iteratively learning a Q-function from policy-specific target values – can be inherently ill-posed and prone to failure. These insights cast doubt on the reliability of Q-learning as a universal solution for reinforcement learning problems.

1 Introduction & related work

Q-learning [1] is one of the most fundamental reinforcement learning (RL) concepts making it the foundation of many popular RL algorithms. However, from the perspective of an industrial practitioner it often falls short in terms of learning stability. The core idea of Q-learning is to iteratively update Q-values using the Bellman equation. While this approach works well for table-based Markov Decision Processes (MDPs), many relevant MDPs involve a continuous state space, necessitating the use of a function approximator to learn the Q-function. Additionally, the frequent requirement to learn offline means that Q-learning combines bootstrapping, off-policy learning, and function approximation. This combination, known as the *deadly triad* [2, 3], presents significant challenges.

In this paper, we investigate how to mitigate the known issues of Q-learning on a representative RL benchmark with a continuous state space, and why achieving stability remains challenging. We begin by using the well-established model-free Q-learning algorithm Neural Fitted Q Iteration (NFQ) [4] as a baseline. Next, we eliminate bootstrapping by employing the model-based bootstrapping-free NFQ (BSF-NFQ) [5], and finally, we address model inaccuracies by utilizing the real environment dynamics. Throughout our study, we compare the robustness of policy learning, demonstrating significant improvements, but we are unable to completely eliminate instability.

Finally, we show that fitting the targets can result in performance variability among policies. By visualizing the true Q-function, we reveal a structure that cannot be accurately approximated using a neural network (NN), rendering the

*The project this report is based on was supported with funds from the German Federal Ministry of Education and Research under project number 01IS24087A. The sole responsibility for the report's contents lies with the authors.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Q-learning task ill-posed. Furthermore, we demonstrate that the problem is induced by the definition of the MDP and not the algorithm itself. Consequently, this issue affects not only Q-learning but also other methods that rely on sample-based evaluation of Q-values.

2 Experimental setup

This paper explores stability issues commonly encountered in continuous state MDPs like inverted pendulum, acrobot and hopper [6]. Due to illustration purposes, we perform the experiments on the iconic cart-pole balancing benchmark.

The state space is four-dimensional, *i.e.*, position x, velocity $\dot{x}$, angle θ, and angular velocity $\dot{\theta}$. The dataset D has been generated by a random policy on the gym environment *CartPole-v1* from the RL benchmark library *Gymnasium*[1]. D consists of 20,000 observation tuples of form (s_t, a_t, s_{t+1}, r_t). Depending on state s_t and action a_t, the system transitions to the next state s_{t+1} and the agent receives a real-value reward $r_t \in \mathbb{R}$.

For the reward, we define a function that assigns 1 for an upright pole with the cart in the center and decreases quadratically along cart position and pole angle relative to their termination bounds, *i.e.*, $r = (1-(x/2.4)^2+1-(\theta/0.2095)^2)/2$.

In our experiments, Q-functions were approximated using NNs in a supervised learning manner using the Adam optimizer with learning rate 0.01, mini batch size 100 and mean squared error as loss function. The dataset was split into blocks of 70% and 30% (training and validation, respectively). Like in [5], NNs with a 5-64-1 architecture were used with state-action tuples as input and ReLU non-linearity. Early stopping was employed against overfitting, halting training when no improvement of the validation error for 50 epochs was made and the best parameters found so far persisted.

3 From bootstrapping to real Q-values

The NFQ algorithm fits with an NN iteratively the targets

$$Q_{i+1}(s_t, a_t) \leftarrow r_t + \gamma \max_{a_{t+1}} Q_i(s_{t+1}, a_{t+1}). \tag{1}$$

In [5], an alternative algorithm called BSF-NFQ was introduced. Here, the target calculation utilized model-based rollouts

$$\tilde{Q}^{\pi}_{\mathrm{MB}}(s,a) = R(s,a,\tilde{s}_1) + \sum_{k=1}^{K-1} \gamma^k R(\tilde{s}_k, \pi(\tilde{s}_k), \tilde{s}_{k+1}), \tag{2}$$

where $\tilde{s}_{k+1} = M(\tilde{s}_k, \pi(\tilde{s}_k))$, and is calculated for a transition model M and a reward model R which both can be learned from the offline dataset. BSF-NFQ increased robustness significantly by calculating targets with

$$Q_{i+1}(s_t, a_t) \leftarrow r_t + \gamma \tilde{Q}^{\pi}_{\mathrm{MB}}(s_{t+1}, \pi(s_{t+1})), \text{ with } \pi(s) = \arg\max_a Q_i(s,a). \tag{3}$$

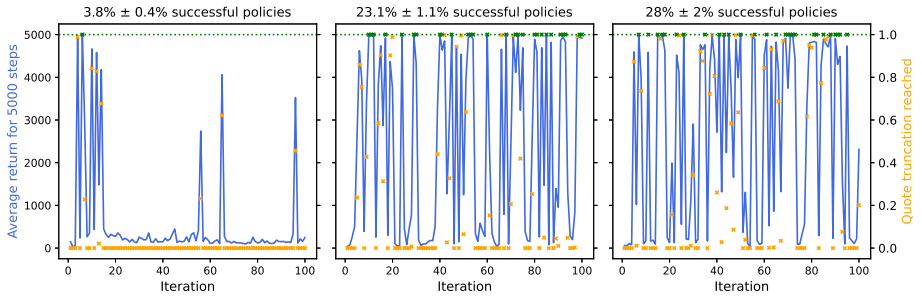
[1] https://gymnasium.farama.org

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

As a result, the instability across iterations was reduced significantly but not fully resolved (see Figure 1 (b)). After learning *successful* policies, *i.e.*, policies that balance successfully for 1,000 starting states for at least 5,000 steps, follow-up iterations yielded bad policies again. This gives rise to the question if this is due to learning imperfect transition models M on the dataset.

To eliminate this potential source of error, we replaced M with the available benchmark transition equations and calculated the true Q-value targets using Eq. (3). A rollout horizon of $K = 1,000$ and discount factor $\gamma = 0.99$ was chosen, resulting in a truncation error of $\epsilon_{\text{trunc}} < 0.005$.

Training 100 BSF-NFQ iterations on true Q-values yielded on average in 28% of the iterations successful policies (see Figure 1 (c)). Replacing the bootstrapping in NFQ by model-based policy rollout state-value estimates (BSF-NFQ) dramatically improved the robustness of the learning algorithm and using the real dynamics (BSF-NFQ-real-dyn) improved this even further. The reoccurring performance drops become rarer as depicted in Figure 1, but still persist, and thus, suggest a fundamental problem with Q-learning.



(a) NFQ (b) BSF-NFQ (c) BSF-NFQ-real-dyn

Fig. 1: Iteration-wise policy performance averaged over 1,000 gym environment episodes. Blue lines represent the average return over 1,000 episodes each with 5,000 steps. Cross markers depict the quote of episodes reaching 5,000 steps. Green markers represent iterations where *successful* policies have been found.

4 Observing policy performance instabilities

Now that we have eliminated any potential transition model error by using the true transition and reward equations, the next step is to investigate the Q-function fitting process itself. To gain insights into the observed instability, *e.g.*, the successful policy in iteration 18 yielded a bad policy in iteration 19, we save the targets calculated with Eq. (3) based on the policy in iteration i and retrain iteration $i + 1$ with NNs initialized with different seeds. The resulting policies were tested with respect to their performance and the corresponding box-plots for the first 20 iterations are depicted in Figure 2.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

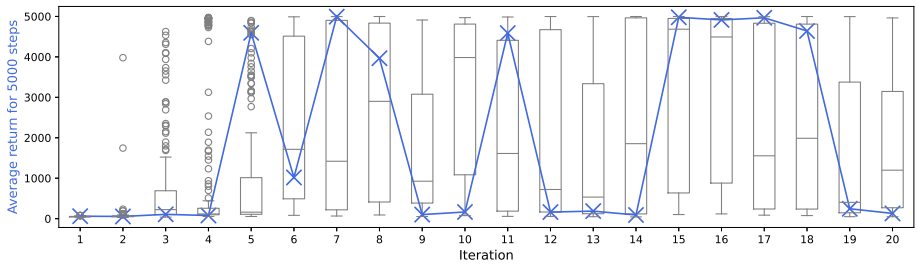


Fig. 2: Iteration-wise policy performance averaged over 1,000 episodes. Average return of the original iterations are depicted in blue. The boxplots visualize policy performance results for retraining with saved Q-value targets on 100 seeds.

Surprisingly, even within one iteration, the retrained policies show a large spread in performance although they use the same Q-value targets. Examining potential indicators throughout the training, such as training and validation error, showed no correlation with the resulting policy performance. Similarly, adjusting hyperparameters like total epochs to train or patience for early stopping also showed no significant impact on the performance variability. Furthermore, the large spread is observed for nearly all iterations. This points to an inherent issue with the Q-value targets themselves since they do not necessarily seem to be useful to learn if we want to learn a better policy.

5 An ill-posed learning task

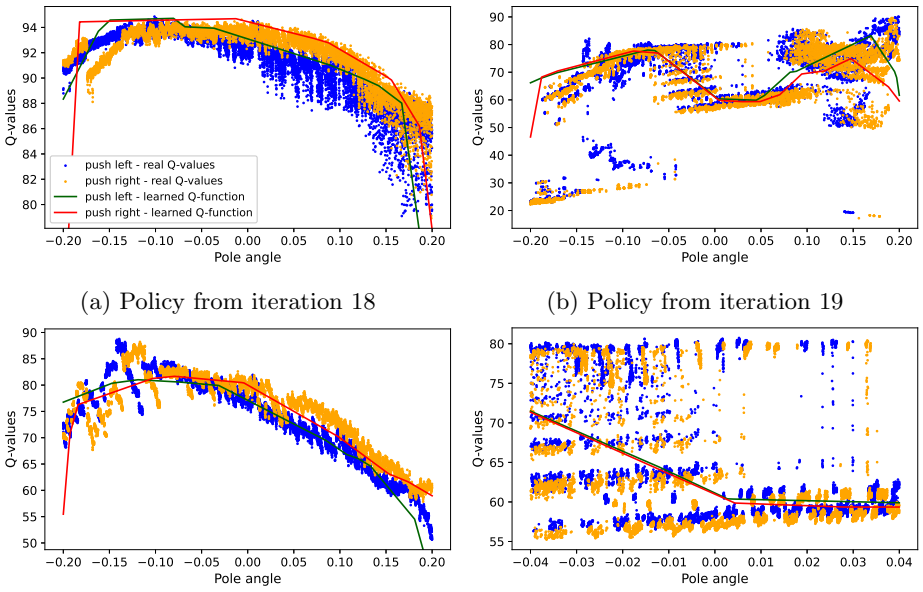
To explore why learning Q-values even for a relatively simple benchmark like cart-pole leads to vastly different Q-function approximations in terms of balancing performance, we conducted an in-depth analysis of the structure of the actual Q-values. Figure 3 illustrates slices through the true Q-function by holding the three state space variables x, $\dot{x}$, and $\dot{\theta}$ constant at 0, and plotting the Q-values across 10,000 gridded values of the pole angle θ.

Figures 3 (a) and (b) display the Q-values from iterations 18 and 19 of the Q-learning run shown in Figure 2, respectively. Notably, the policy resulting from iteration 18 successfully balanced the pole for all starting states. However, in the subsequent iteration 19, the policy's performance completely collapsed.

The figures reveal the highly discontinuous structure of the true Q-values which is particularly evident in the magnified view depicted in Figure 3 (d). We want to emphasize that the substantial differences in the true Q-values between adjacent angle values are not caused by noise, as the policy rollouts in Figures 3 (a), (b), and (d) are entirely deterministic.

Learning a subsequent Q-function on samples of these true Q-values fails to capture the structure completely and results in the NN approximating an average over the samples, which is illustrated by the line plots in Figure 3. This loss of information is the primary source of error for the policy performance instability.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.



(a) Policy from iteration 18

(b) Policy from iteration 19

(c) ϵ-greedy policy from iteration 18

(d) Policy from iteration 19 - magnified

Fig. 3: Q-function values along 10,000 different pole angle values with cart position, cart velocity and pole velocity fixed at 0.0. To calculate the rollouts, policies that are greedy with respect to their learned Q-value approximation were used, or for (c) ϵ-greedy with an $\epsilon = 0.05$. Note that the plots display only a single Q-value for each corresponding angle. Therefore, the significant differences between neighboring angle values indicate function discontinuities.

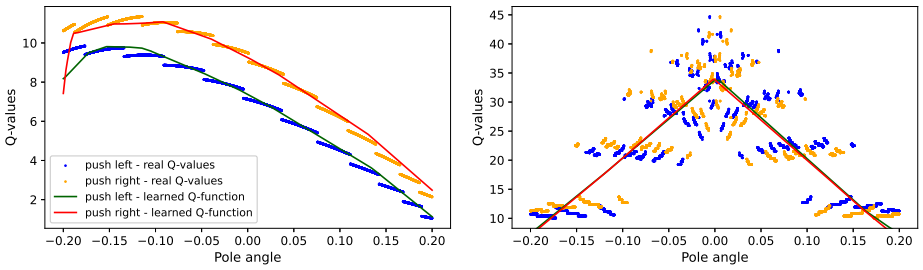
The subsequent policy can encounter states where the true Q-value for the optimal action was underestimated while the true Q-value for the other action was overestimated. This effect can result in the subsequent policy to take a suboptimal action and in case of the cart-pole benchmark, taking several suboptimal actions consecutively can result in a quickly terminating trajectory. Thus, the policy in iteration $i+1$ can be significantly worse than the policy from iteration i that was used to calculate the targets.

The effect of calculating the expected rollout stochastically with an ϵ-greedy policy, as depicted in Figure 3 (c), shows that the randomness reduces the size of discontinuities, but does not remove them.

To demonstrate the observed discontinuities are fundamental and not solely due to specific policies in Q-learning, Figure 4 shows the Q-function for two simple policies. On the left, the Q-function for the policy that pushes left is shown, on the right, for the policy acting against the pole's angle is depicted. In both cases, the discontinuities manifest, indicating they are a fundamental characteristic of the underlying MDP rather than an artifact of the policy design.

These results suggest that the discontinuities can already occur if the state

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.



(a) Policy that always pushes left

(b) Policy against pole angle

Fig. 4: Q-function values along 10,000 different pole angle values with cart position, cart velocity and pole velocity fixed at 0.0.

space of the MDP is continuous. We argue that discontinuities in the Q-function or return values affect any method that uses them in a sample-based manner. This includes all methods using function approximators that have been derived from Q-learning or are based on the evaluation of the return on a sample, as well as that of offline policy evaluation. Although Q-learning can yield desired policies, discontinuities can lead to performance collapse and divergent behavior. Trying to fit an NN with samples from a discontinuous function makes the problem ill-posed in the first place.

6 Conclusion

In this paper, we identified a fundamental issue with estimating Q-values and return values. While estimating Q-values at isolated points can be relatively effective, using a function approximator to learn them presents significant challenges. We demonstrated the dramatic impact that discontinuities in the Q-function can have on Q-learning, potentially causing a collapse in the quality of the learned policies. Our findings illustrate that this problem can emerge even in simple MDPs with continuous state spaces.

References

[1] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 1992.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press Cambridge, 2018.

[3] Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad, 2018.

[4] Martin Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *ECML*, number 6, 2005.

[5] Philipp Wissmann, Daniel Hein, Steffen Udluft, and Volker Tresp. Why long model-based rollouts are no reason for bad Q-value estimates. In *ESANN*, 2024.

[6] Tao Wang, Sylvia Herbert, and Sicun Gao. Fractal landscapes in policy optimization. In *Advances in Neural Information Processing Systems*, volume 36, pages 4277–4294, 2023.