NMix Q-learning: Investigating overestimation bias of Q-values

Kim, Subin
21supersoo@kaist.ac.kr

Nam, Kyungwook
ruddnrdl@kaist.ac.kr

Baek, Doojin
djinb00@kaist.ac.kr

Abstract

Q-learning, a widely adopted algorithm in reinforcement learning (RL), is susceptible to the problem of overestimating action values. This can impede the learning process and result in suboptimal decision-making. To address this issue, we propose NMix Q-learning, a novel approach that effectively mitigates overestimation bias. Our method tackles overestimation bias by leveraging multiple estimators. While Maxmin also utilizes multiple estimators, it selects the minimum value among multiple maximum values, whereas NMix selects the maximum values among multiple minimum values. Through our experiments, we demonstrate that NMix Q-learning successfully addresses the challenge of Q-value overestimation across multiple environments. The code is available at `https://github.com/KSB21ST/cs492_project`.

1 Introduction

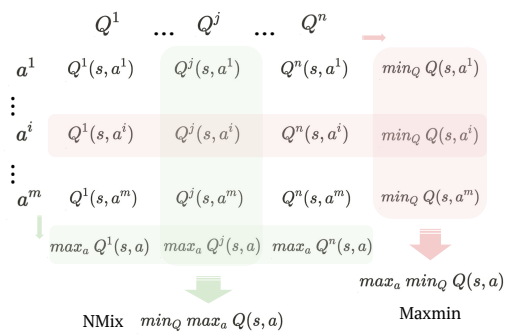
1.1 Overestimation bias in Q-Learning

Q-learning is a popular reinforcement learning algorithm due to its simplicity [1]. It updates the agent's action value estimates based on observed rewards and the estimated value of the best action in the next state. However, this simple update rule suffers from overestimation bias, where the maximum action value estimate can be misleadingly high due to stochasticity or errors in the estimation process. Imagine you are rolling a dice 100 times and take the maximum value over all throws. It would be likely to be greater than the expected value 3.5. When all action values are consistently overestimated, it does not pose a significant issue because what truly matters is the relative differences between them. However, if the overestimations are not uniform and vary across actions, it can potentially hinder the learning process by slowing down the rate at which the agent acquires accurate estimations. This can result in slower learning progress and difficulties in effectively exploring and identifying optimal states.

1.2 How can we overcome the overestimation bias?

This issue becomes more problematic when using function approximation [2] and can significantly affect the quality of the learned policy or even lead to failures of Q-learning. Recent experiments in various domains have confirmed the prevalence of this overestimation problem. In this paper, we ablate the effects of Q-value overestimation bias in diverse settings and propose a novel algorithm NMix to mitigate the overestimate bias.

Double Q-Learning involves selecting the expected action value of one estimator as the maximum action value from another estimator, ensuring that overestimation of the true maximum action value is avoided [2]. Double DQN [3], an extension of this concept to Q-learning using neural networks, has demonstrated notable performance improvements compared to traditional Q-learning. However, although this method addresses the issue of overestimation bias, it introduces the possibility of underestimation bias, which is not always preferable, as our experiments reveal.



Q-target

NMix	$r + \gamma \cdot min_Q\, max_{a'}\, Q(s', a')$
DQN	$r + \gamma \cdot max_{a'}\, Q(s', a')$
DDQN	$r + \gamma \cdot Q(s', argmax_{a'} Q(s', a';\, \theta-));\, \theta)$
Averaged DQN	$r + \gamma \cdot max_{a'}\, \frac{1}{n}\, \Sigma\, Q(s', a')$
Maxmin DQN	$r + \gamma \cdot max_{a'}\, min_Q\, Q(s', a')$

Figure 1: Our paper introduces a novel algorithm NMix to mitigate Q-value overestimation bias. The left figure is the algorithm for Maxmin Q-Learning and NMix Q-Learning. Maxmin Q-learning takes the minimum operation over multiple Q-networks for each action and then selects the maximum value among them. Unlike Maxmin, NMix takes the maximum Q-values from each N estimators and selects the minimum value. Through taking the minimum value, NMix is able to mitigate the overestimation bias. The right figure is the expression for selecting Q-target in each algorithm, where NMix, Averaged DQN, Maxmin Q-Learning all employs N estimators to mitigate both the overestimation and underestimation bias.

Averaged Q-Learning is also an extension to the Q-Learning algorithm [4] that uses the K previously learned Q-values estimates to produce the current action-value estimate. By averaging previously learned Q-value estimates, it improves the stability and performance of DRL algorithms by reducing the variance in target value approximation. Similarly, **Maxmin Q-Learning** effectively tackles the issue of overestimation bias by employing a minimization approach that considers multiple action-value estimates [5] and shows the ability to manipulate the estimation bias, ranging from positive to negative.

2 NMix Q-Learning: N-network Min-max Q-Learning

2.1 Q selection strategy

A natural question arises regarding the effectiveness of the aforementioned Q-selection strategies in mitigating the problem of overestimation. To explore this, we propose a novel strategy called NMix, taking minimum values from the maximum values of each Q-network. By initially applying the max operation over the action space for each Q-network, the resulting values are valid within the context of Q-Learning. To mitigate overestimation, we then find the minimum value among these maximum values. This process can be seen as constructing a milder Q-target, effectively preventing the detrimental effects of overestimated Q-values.

2.2 Mathematical comparison between Maxmin Q-Learning

Comparing NMix and Maxmin, we have mathematically proven that the outcome of NMix is always greater than or equal to that of Maxmin. In Figure 1, let's say the Maxmin output is the value using j^{th} Q-network with i^{th} action $Q^j(s, a^i)$, without losing generality. Since NMix takes maximum value over actions,

$$Q^j(s, a^i) \leq C^j = max_a Q^j(s, a)$$

Moreover, because Maxmin takes minimum value over Q-networks, We can derive the following inequality:

$$Q^j(s, a^i) \leq Q^k(s, a^i) \leq C^k = max_a Q^k(s, a)$$
$$Q^j(s, a^i) \leq c, \forall c \in \{C^1, C^2, \cdots, C^n\}$$

The output of NMix is $min_Q max_a Q(s, a) = min\{C^1, C^2, \cdots, C^n\}$, where we can say NMix output is always greater than or equal to Maxmin output. However, this doesn't mean that NMix cannot alleviate the overestimation problem. Compared to Q-Learning, NMix naturally handles overestimation by utilizing the min operation.

3 Experiments

In this section, we investigate the impact of q-value overestimation bias across various environments and examine the robustness of the algorithms in simplified stochastic MDP environment. To achieve this, we measure the average return over three Atari games and also estimated the mean of q-value in each step during training.

3.1 Analysis on Atari benchmarks

In order to evaluate various Q-learning algorithms, including NMix Q-learning, we selected three games from PyGame Learning Environment (PLE) [6] and MinAtar [7]: Catcher, Pixelcopter, and Asterix. We adapted the default experimental setup from Explorer [8] and ran multiple configurations to determine the best-performing hyper-parameters for each algorithm. The hyper-parameters is described in Appendix. For Averaged Q-Learning, Maxmin Q-Learning, and NMix Q-Learning, the number of estimators (target networks) N was selected among 2, 4, and 8, referring to [5].

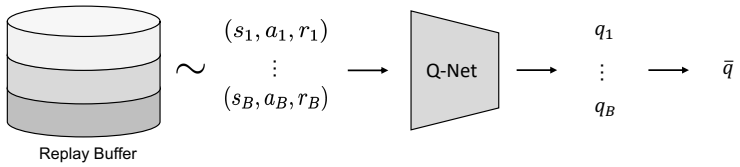
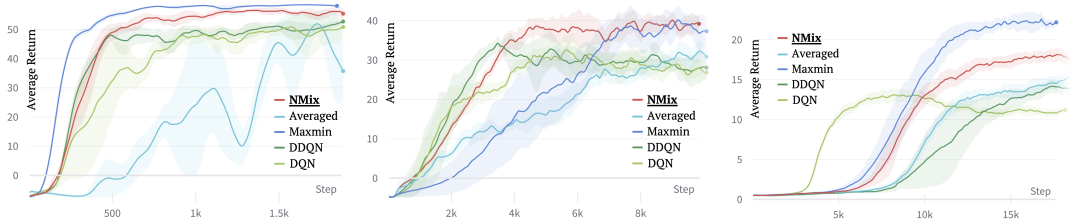
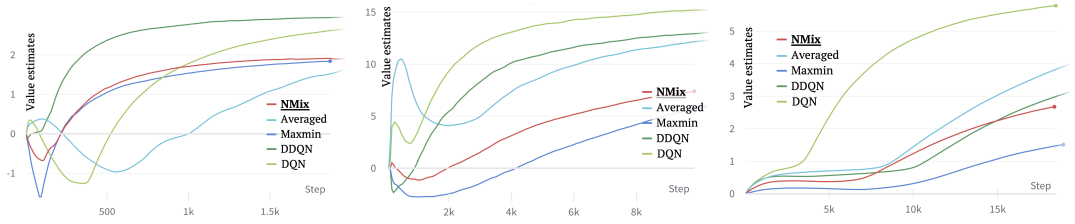


Figure 2: The mechanism of calculating the estimated Q-value. In each training step, we sample a batch of state-action pairs from the buffer and compute the Q-value from each estimator for the sampled pairs in the batch. We take the averaged Q-value as an estimate of the Q-value in each training step.



(a) Catcher (b) Copter (c) Asterix

Figure 3: Learning curves on the three benchmark environments. The depicted return is averaged over the last 100 episodes, and the curves are smoothed using an exponential average, to match previous reported results [5]. The results were averaged over 3 runs of the best-performing configurations for , with the shaded area representing one standard error.



(a) Catcher (b) Copter (c) Asterix

Figure 4: Value estimates on the three benchmark environments. The depicted returns are measured under the same condition with Figure 3. We take the averaged q-value estimates of a sampled batch in each training step, described in Figure 2. The results show that overestimation bias corresponds to the deteriorated average return.

Figure 3 shows average return over three atari games. NMix Q-learning performs well compared to DQN, DDQN and Averaged-Q-Learning, and performs similarly to Maxmin Q-Learning. Also for average Q-value as shown in Figure 4, NMix estimates Q-value lower than DQN, DDQN and Averaged Q-Learning, which corresponds to the performance evaluation in the average return. It empirically validates our theoretical hypothesis on Maxmin and NMix Q-learning.

3.2 Understanding effects of overestimation bias on simple MDP

In Section 3.1, algorithms that suffered from overestimation bias showed inferior performance. However, there can be environments where overestimation bias is helpful. To investigate whether NMix would perform moderately in such environments, we constructed a simple MDP environment as shown in Figure 5.

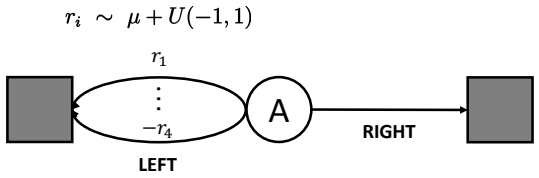
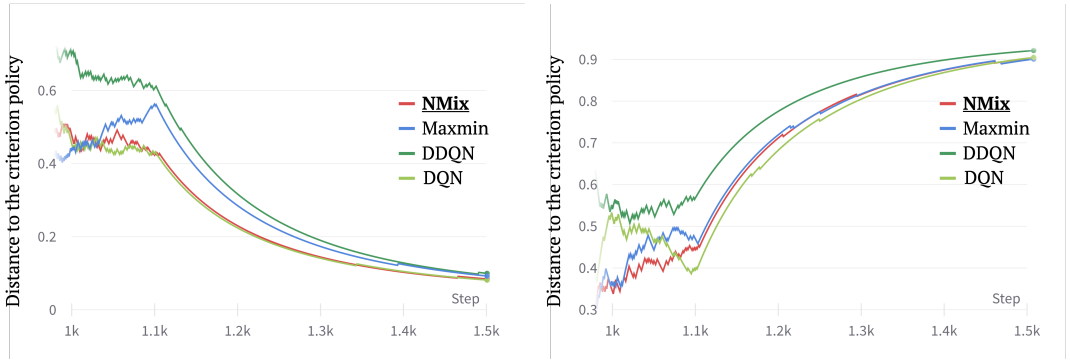


Figure 5: A simple episodic MDP, adapted from [9] which is used to highlight the difference between Double Q-learning and Q-learning. This MDP has two nonterminal states A and B. Every episode starts from A which has two actions: Left and Right. The Right action transitions to a terminal state with reward 0. The Left action transitions to state B with reward 0. From state B, there are 8 actions that all transition to a terminal state with a reward $\mu + \epsilon$, where ϵ is drawn from a uniform distribution U(1, 1). When $\mu > 0$, the optimal action in state A is Left; when $\mu < 0$, it is Right



(a) $\mu = 0.3$ (b) $\mu = -0.3$

Figure 6: Comparison of four algorithms using the simple MDP in Figure 5 with different values of μ, and thus different expected rewards. For $\mu = +0.3$, shown in (a), the optimal ϵ-greedy policy is to take the Left action with 95% probability. We set this as the criterion policy. The reported distance is the absolute difference between the probability of taking the Left action under the learned policy compared to the criterion ϵ-greedy policy.

We conducted two experiments with μ values of 0.3 and -0.3, with ϵ value of 0.1. if μ is 0.3, the optimal policy is choosing left with probability of 0.95. We trained the Q-Learning algorithms and measured the distance between a trained policy and the optimal policy.

As depicted in Figure 6.a, the outcomes of each algorithm in the environment where overestimation provides assistance align with our previous analysis. Q-Learning, which does not explicitly address overestimation bias, exhibited closer proximity to the optimal policy compared to Double Q-Learning, which strongly suppresses overestimation bias. Notably, as mathematically verified, NMix showed a closer resemblance to Maxmin Q-Learning owing to its higher Q values. An intriguing observation from this experiment is that both Maxmin and NMix Q-Learning algorithms outperformed the standard Q-Learning approach.

In Figure 6.b, we examine the environment where underestimation is beneficial. Here, Double Q-Learning showed closer alignment with the optimal policy compared to Q-Learning. Conversely, NMix and Maxmin, our proposed approaches, fell between these two methods in terms of performance.

For the case that NMix using different numbers of estimators observed in Figure 8 in Appendix was most close to optimal policy than others. This result is different from [5], since they have shown Maxmin Q-Learning learns faster but reaches sub-optimal performance behaving more like Q-learning in smaller N. And for larger N learns more slowly but shows better performance.

4 Conclusion

In this study, we introduced NMix Q-learning, a novel approach that modifies the processing of Q-values among multiple estimators. Our experiments involved five Q-learning algorithms and three benchmark environments. NMix Q-learning effectively mitigated the overestimation bias and achieved comparable performance to Maxmin Q-learning, which outperformed existing algorithms. Although NMix Q-learning exhibited a larger degree of overestimation compared to Maxmin Q-learning, it was smaller than that of other algorithms, aligning with our initial hypothesis. Furthermore, we investigated the behavior of Q-learning algorithms in stochastic Markov Decision Processes (MDPs) where overestimation and underestimation can be beneficial. The stochastic nature of the environment influenced the proximity to the criterion policy of the Q-learning algorithms. The results demonstrated that NMix Q-learning, as proposed in this work, converges to the criterion policy more efficiently compared to other algorithms. In future research, we aim to explore and evaluate more complex environments that are relevant to the benefits of overestimation and underestimation. Additionally, we plan to provide a more concise mathematical convergence proof and conduct additional ablation studies to gain further insights into the nature of NMix Q-learning.

References

[1] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[2] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[3] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[4] Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*, pages 176–185. PMLR, 2017.

[5] Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

[6] Norman Tasfi. Pygame learning environment. `https://github.com/ntasfi/PyGame-Learning-Environment`, 2016.

[7] Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for more efficient reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 59:60, 2019.

[8] Qingfeng Lan. A pytorch reinforcement learning framework for exploring new ideas. `https://github.com/qlan3/Explorer`, 2019.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

5 Appendix

5.1 Contribution

Subin Kim Participated in proposing NMix Q-Learning and coding the algorithm based on Maxmin Q-Learning. Participated in coordinating the experiments and analyzed the result. Participated in overall writing process of the final report.

Doojin Baek Participated in designing NMix algorithm and mathematically proved its nature. Participated in coding MDP environment. Wrote the report and drew the figures in the report. Participated in the presentation

Kyungwook Nam participated in this project by understanding Maxmin Q-learning paper and provide references and code examples. Also he wrote the report and presentation slides, took a role on presentation of the result and discussion part. Also he resolved git related issues which team faced.

5.2 Hyper-parameters

Table 1: Hyperparameter Setting on Three Benchmark Environments

Env	Catcher	PixelCopter	Asterix
Model Hidden Layers	[[64, 64]]	[[64, 64]]	[[64, 64]]
Memory Size	1e4	1e4	1e5
Exploration Steps	1e3	1e3	5e3
Epsilon Start, End	(1.0, 0.01)	(1.0, 0.01)	(1.0, 0.1)
Epsilon Decay	0.999	0.999	0.999
Loss	MSELoss	MSELoss	SmoothL1Loss
Optimizer	RMSProp	RMSProp	RMSProp
Batch Size	32	32	32
Discount Factor	0.99	0.99	0.99

Table 2: Learning Rate Settings for each environment and algorithm

	Catcher	PixelCopter	Asterix
DQN	1e-4	1e-4	3e-4
DDQN	1e-4	1e-4	1e-4
Averaged-DQN	1e-5	3e-5	1e-4
Maxmin Q-learning	3e-4	1e-4	1e-3
NMix Q-learning	1e-4	2e-4	1.2e-3

5.3 Ablation of NMix on different number of estimators

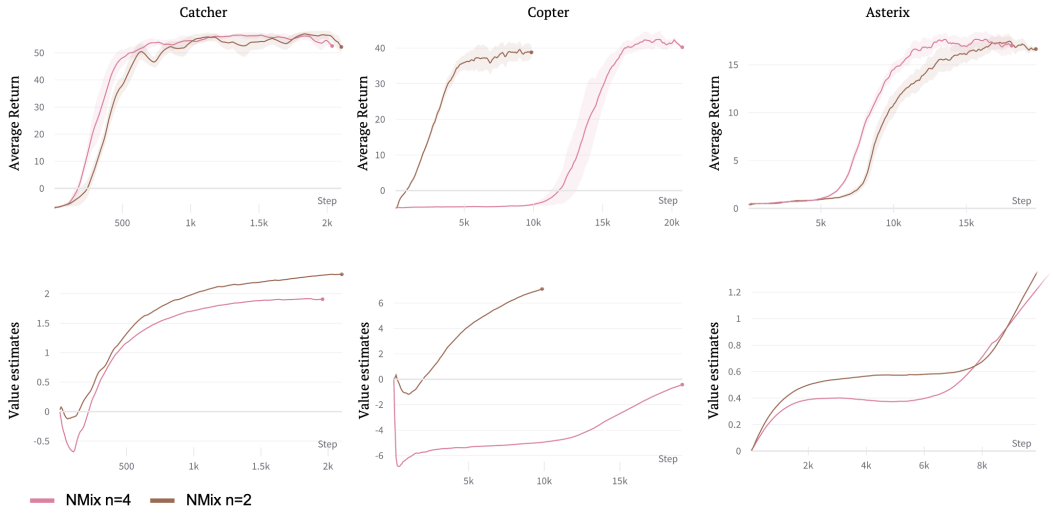
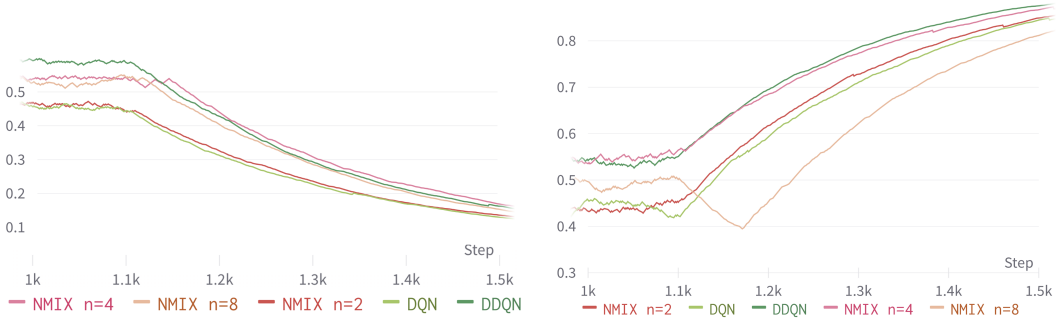


Figure 7: Learning curves of NMix using different numbers of estimators on the three benchmark environments. The depicted returns are measured under the same condition with Figure 3. Using 4 estimators performs slightly better then using 2 estimators.



(a) $\mu = 0.3$ (b) $\mu = -0.3$

Figure 8: Comparison of DQN, DDQN, and NMix using 2, 4, 8 number of estimators in the simple MDP in Figure 5. The depicted returns are measured under the same condition with Figure 7. We can see that using 4 estimators tend underestimate, and using 2 estimators tend to overestimate compared to DQN, DDQN. Meanwhile, using 8 estimators is showing less bias in both situations.