M²DQN: A Robust Method for Accelerating Deep Q-learning Network

Zhe Zhang†
UniDT
Shanghai, China
zhe.zhang@unidt.com

Yukun Zou†
Research Institute of Intelligent Complex Systems, Fudan University
Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University
Shanghai, China
ykzou19@fudan.edu.cn

Junjie Lai†
Research Institute of Intelligent Complex Systems, Fudan University
Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University
Shanghai, China
jjlai19@fudan.edu.cn

Qing Xu
UniDT
Shanghai, China
qing.xu@unidt.com

ABSTRACT
Deep Q-learning Network (DQN) is a successful way which combines reinforcement learning with deep neural networks and leads to a widespread application of reinforcement learning. One challenging problem when applying DQN or other reinforcement learning algorithms to real world problem is data collection. Therefore, how to improve data efficiency is one of the most important problems in the research of reinforcement learning. In this paper, we propose a framework which uses the Max-Mean loss in Deep Q-Network (M²DQN). Instead of sampling one batch of experiences in the training step, we sample several batches from the experience replay and update the parameters such that the maximum TD-error of these batches is minimized. The proposed method can be combined with most of existing techniques of DQN algorithm by replacing the loss function. We verify the effectiveness of this framework with one of the most widely used techniques, Double DQN (DDQN), in several gym games. The results show that our method leads to a substantial improvement in both the learning speed and performance.

CCS CONCEPTS
• Computing methodologies → Reinforcement learning; Artificial intelligence.

KEYWORDS
depth Q network, data efficiency, mini-max, robust optimization

†Zhe Zhang, Yukun Zou, Junjie Lai contributed equally to this work

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICMLC 2023, February 17–20, 2023, Zhuhai, China
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9841-1/23/02 ...$15.00
https://doi.org/10.1145/3587716.3587735

1 INTRODUCTION
In reinforcement learning (RL), the agent explores and learns by interacting with the environment, with the ultimate goal of maximizing cumulative reward [Richard S. and Andrew G. 2018]. In recent years, a variety of different reinforcement learning algorithms have been applied to classic challenging tasks, such as Go [Silver et al. 2017], Atari Games [Mnih et al. 2013], robotics [Kober et al. 2013], etc., which have achieved excellent performance. In this process, Deep Q-Network (DQN) [Mnih et al. 2013] and its variants, such as Double DQN [Van Hasselt et al. 2016], Dueling DQN [Wang et al. 2016], play an important role.

One of the most important techniques in DQN is the use of experience replay [Lin 1992]. The experience replay address the issues of temporal correlation between transitions and data reuse. Various studies and improvements of experience replay have been proposed. To reduce the forgetting in long-time training, Zhang et al. [Zhang et al. 2019] introduce a framework of dual replay buffer. To improve the data efficiency, Schaul et al. [Schaul et al. 2015] introduce prioritized experience replay (PER), in which the transitions are prioritized by the last encountered absolute TD error. Cao et al. [Cao et al. 2019] design a kind of PER based on both reward and TD error, leading to further improvement on training efficiency. Another widely used technique in DQN is the use of target network, which stabilizes the learning of DQN by reducing variance. Experience replay and target network have become standard practice in DQN algorithms to achieve state-of-the-art performance.

In this paper, we propose a new method to improve the data efficiency. Instead of sampling one batch of experiences in the training step, we sample several batches from the experience replay
and update the parameters such that the maximum TD-error of these batches is minimized. Such maximum TD-error is called the mini-max loss, for short.

The mini-max loss is usually used in robust optimization [Delage and Ye 2010], generative adversarial learning [Goodfellow et al. 2014] and worst-case optimization [Abernethy et al. 2020]. The mini-max loss treat exceptional samples and the normal samples in an equal way, and fully consider the influence of all examples, especially the worst-case samples. Thus, the mini-max loss helps to learn a robust model.

We develop a framework which combines the Mini-Max method with Deep Q-Network and its variants (We call it M^2DQN). We evaluate our algorithm with Double DQN (DDQN) on several gym [Brockman et al. 2016] environments in this paper. The results show that the proposed method leads to a substantial improvement in both the learning speed and performance.

2 BACKGROUND

Reinforcement learning addresses the task of an agent learning to interact with the environment in order to maximize the cumulative future reward. We model this interaction as a discounted Markov Decision Process (MDP) $(S, A, T, R, γ)$, which consists of states $S$, actions $A$, a reward function $R(s, a) : S \times A \rightarrow \mathbb{R}$, a state transition function $T(s, a, s') = P(s' | s, a)$ and a discounted factor $γ \in [0, 1]$. A policy $π$ is a function that maps every state $s \in S$ to a probability distribution over the action space.

The discounted cumulative future reward (discounted return) $R_t$ at time-step $t$ is usually defined to be the discounted sum of future rewards

$$R_t = \sum_{k=0}^{\infty} γ^k r_{t+k}$$

where $r_t$ is the reward at time-step $t$. Under a given policy $π$, the action-value function is defined as the expected return starting from state $s$ and action $a$, i.e.,

$$Q^π(s, a) = \mathbb{E}_{π} \left[ R_t \mid S_t = s, A_t = a \right]$$

we define the optimal action-value function $Q^*(s, a)$ as follows

$$Q^*(s, a) = \max_π Q^π(s, a)$$

The goal of reinforcement learning is to find an optimal policy that maximizes the expected discounted return. Note that an optimal policy can be derived from the optimal action-values by selecting the the action of maximum value in each state. Therefore, the goal becomes to find an optimal action-value function $Q^*(s, a)$. The optimal action-values $Q^*(s, a)$ obeys the Bellman equation [Bellman 2010], i.e.

$$Q^*(s, a) = \mathbb{E} \left[ R(s, a) + γ \sum_{s'} P(s' \mid s, a) \max_{a'} Q^*(s', a') \right]$$

and can be learned by Q-learning [Watkins 1989]. When state and action spaces are large, it is intractable to learn action-value for each state and action pair independently. To overcome the difficulty of large state and action spaces, deep neural network $Q(s, a, \theta)$ with weights $θ$ is used to estimate the action-value function, which is called deep Q-network. Then, At each time-step $t$, the agent observes a state $s_t$ of the environment and receives a reward $r_t$ following the selected action $a_t$ and resulting a state $s_{t+1}$, the parameters $θ_t$ of Q-network $Q(s, a, θ_t)$ can be updated as follows:

$$θ_{t+1} ← θ_t + γ \cdot TD(s_t, a_t, r_t, s_{t+1}; θ_t) \nabla_θ Q(s_t, a_t, θ_t)$$

where

$$TD(s_t, a_t, r_t, s_{t+1}) = r_t + γ \max_a Q(s_{t+1}, a; θ_t) - Q(s_t, a_t; θ_t),$$

which is called TD-error.

However, the learning of DQN is unstable. One of the reasons is that consecutive samples generated in the learning of DQN are highly correlated which breaks the i.i.d. principle (dependently and identically distributed) of machine learning. Two techniques have been used to stabilize the learning of DQN. The first one is experience replay, which holds last thousands transitions. At training time, a batch of these transitions is sampled uniformly and used to update the network. The use of experience replay breaks the temporal correlation between transitions. The second one is the double Q-network, in which a separate target network is used to estimate the action-values (The target network is copied every few steps from the regular network) and the regular network is used to calculate the argmax over next states. Separating the choosing of actions and the estimation of action-values reduces the overestimation in regular DQN.

The use of the experience replay can alleviate the non-iid problem of consecutive samples to a certain extent. In this paper, we propose a new method to solve the non-iid problem in the learning of DQN based on the nonlinear expectation theory.

The nonlinear expectation theory is laid down by Peng [Peng 2005]. Nonlinear expectation, including sublinear expectation as its special case, is a functional $\mathbb{E} : H \mapsto \mathbb{R}$ satisfying monotonicity, constant preserving, sub-additivity, and positive homogeneity. It is a new framework of probability theory to characterize the uncertainty and has potential applications in some scientific fields, especially in risk management.

In statistical learning, we define maximum expectation as same concept of the sublinear expectation $\mathbb{E}$, which can be represented as the upper expectation of a subset of linear expectations, i.e.,

$$\mathbb{E}(X) = \sup_{j \in J} \mathbb{E}_j(X),$$

indexed by $j \in J$.

The usual estimation of expected error is based on the Law of Large Numbers. When the i.i.d. condition breaks, LLN is not applicable. However, the nonlinear LLN still holds, i.e., the limit distribution of

$$\frac{X_1 + X_2 + \cdots + X_n}{n}$$

is a maximal distribution (see [Peng 2005]). The parameter of the maximal distribution can be estimated using a max-mean statistics (see [Jin and Peng 2021]).

Motivated by this work, Xu et al. (2019) [Xu and Xuan 2019] consider a class of nonlinear regression problems without the i.i.d. assumption. They split the training set into $N$ groups such that in
each group the i.i.d assumption can be satisfied. Then, the following max-mean loss is used:

$$\max_{1 \leq j \leq N} \frac{1}{n_j} \sum_{k=1}^{n_j} (g^\theta (x_{jk}) - y_{jk})^2,$$

where $n_j$ is the number of samples in group $j$, $(x_{jk}, y_{jk})$ is the $k$-th sample in group $j$ and $g^\theta$ is the model function parameterized by $\theta$ which we want to learn.

We want to learn $\lambda$, the parameter of the dual problem. Hence we turn to the dual problem

$$\min_{\lambda} \left( \frac{1}{2} a^T G G^T \lambda - f^T \lambda \right)$$

subject to

$$\lambda_i \geq 0,$$  \hspace{1cm} (5)

where $G = \nabla f, f = (f_1(\theta_1), \ldots, f_N(\theta_N))^T$ is the Jacobian matrix. Note that the dimension of the dual problem (4)-(5) is $N \times (n - n)$ which is independent of $n$ (number of parameters). Let $\lambda$ be a solution of the dual problem (4)-(5). Then, $d_k = -G^T \lambda$ is the solution of problem (2)-(3), which is also a descent direction. Therefore, we update $\theta_t$ as follows

$$\theta_{t+1} \leftarrow \theta_t - \alpha G^T \lambda$$

where $\alpha$ is the learning rate.

The max-mean loss function can be combined with most of existing techniques of DQN algorithm by simply replacing the loss function. Algorithm 1 shows an example of the case of DDQN.

### 3 METHOD

In this section, we formulate the Max-Mean Deep Q-learning Network (M²DQN). Instead of sampling only one batch at each training step, we sample $N$ batches from the experience replay at each time-step $t$. Then the loss function of max-mean TD-error can be defined as follows

$$L(N; \theta) = \max_{1 \leq j \leq N} \frac{1}{n_j} \sum_{k=1}^{n_j} TD(s_{t+jk}, a_{t+jk}, r_{t+jk}, s_{t+1+jk}; \theta)^2$$

where $n_j$ is the sample size and $(s_{t+jk}, a_{t+jk}, r_{t+jk}, s_{t+1+jk})$ is the transition of $j$-th batch.

Since the experience replay breaks the temporal correlations of transitions, the i.i.d condition is satisfied (or considered satisfied) in each group (batch). Therefore, to minimize the max-mean loss, we follow the algorithm of Xu et al. (2019) [Xu and Xuan 2019]. Denote

$$f_j(\theta) = \frac{1}{n_j} \sum_{k=1}^{n_j} TD(s_{t+jk}, a_{t+jk}, r_{t+jk}, s_{t+1+jk}; \theta)^2$$

and

$$\Phi(\theta) = \max_{1 \leq j \leq N} f_j(\theta)$$

To find the descent direction at each time-step $t$, we linearize $f_j$ and $\theta_t$ and obtain the convex approximation of $\Phi$ as

$$\hat{\Phi}(\theta) = \max_{1 \leq j \leq N} \{ f_j(\theta_t) + (\nabla f_j(\theta_t), \theta - \theta_t) \}$$

By adding a regularization term and setting $d = \theta - \theta_t$, the minimization problem becomes

$$\min_d \max_{1 \leq j \leq N} \left\{ f_j(\theta_t) + (\nabla f_j(\theta_t), \theta - \theta_t) + \frac{1}{2} \| d \|^2 \right\},$$

which is equivalent to

$$\min_d \frac{1}{2} \| d \|^2 + a$$

subject to

$$f_j(\theta_t) + (\nabla f_j(\theta_t), d) \leq a, \forall 1 \leq j \leq N.$$  \hspace{1cm} (3)

When the dimension of $d$, i.e., the number of parameters of Q-network, is large, solving Problem (2)-(3) is time-consuming. Hence we turn to the dual problem

| CartPole | LunarLander | MountainCar | Acrobat |
|----------|-------------|-------------|---------|
| HiddenLayer | (128,64,64) | (128,64,64) | (64,32,32) |
| LearningRate | 5e-4 | 5e-4 | 5e-4 |
| MaxStep | 200000 | 1000000 | 1000000 |
| ReplaySize | 10000 | 50000 | 50000 |
| BatchSize | 128 | 128 | 128 |
| Gamma | 0.99 | 0.99 | 0.99 |

Table 1: Experimental Parameters

To measure the efficiency of training process, we focus on two evaluation metric. Our main metric is learning speed, in terms of the training step before the environment is considered solved. The learning speed shows whether the proposed method improves data efficiency or not. Also, we need to ensure that the performance does not decrease when we use the max-mean loss. Therefore, the second metric is quality of the best policy, in terms of the maximum evaluation score of 50 random games during the training process. The gym leaderboard\(^1\) defines "solving" as getting average evaluation score of a specific value over 100 random games. In this paper, we follow the definition of gym leaderboard, but due to the limit of calculation resources, we defines "solving" as getting average evaluation score of a specific value over 50 random games.

\(^1\)https://github.com/openai/gym/wiki/Leaderboard

\(^2\)https://github.com/Myyura/Minimax-DQN
Algorithm 1 Double DQN with max-mean loss (M²DDQN)

Input: batch size $K$; max step $T$; group size $N$; discount rate $\gamma$.

Initialize replay (experience) memory $\mathcal{H} = \emptyset$
Initialize action-value network $Q_{\theta}$ with random weights $\theta$
Initialize target action-value network $Q'_{\theta'}$ with weights $\theta' = \emptyset$
observe $s_0$
for $i = 0$ to $T$
choose action $a_i = \arg \max_{a} Q(s_i, a; \theta)$
execute action $a_i$ and observe reward $r_i$ and state $s_{i+1}$
store transition $(s_i, a_i, r_i, s_{i+1})$ in $\mathcal{H}$
for $j = 0$ to $N$
sample a batch of $K$ transitions $(s_{tj}, a_{tj}, r_{tj}, s_{tj+1})$ from $\mathcal{H}$
compute mean TD-error $f_j = \frac{1}{K} \sum_{k=1}^{K} (r_{tjk} + \gamma Q'(s_{tjk+1}, \arg \max_{a} Q(s_{tjk+1}, a; \theta') - Q(s_{tjk}, a_{tjk}; \theta))^2$
end for
compute the Jacobian matrix $G = \nabla f$, where $f = (f_1, \ldots, f_K)^T$
build and solve the quadratic programming (4)-(5) and get a solution $\lambda$
update $\theta \leftarrow \theta - \alpha G^T \lambda$
from time to time copy weights into target network $\theta' \leftarrow \theta$
end for

The normalized max score and the normalized training step before a environment is solved are showed in Table 2, with a more detailed learning curves and cumulative mean score curves on Figure 1 ~ Figure 2.

| Environment  | Method     | MaxScore | StepToSolve |
|--------------|------------|----------|-------------|
| CartPole-v1  | DDQN       | 100%     | 100%        |
|              | M²DDQN ($N = 5$) | 100%     | 75.73%      |
|              | M²DDQN ($N = 10$) | 100%     | 89.07%      |
| LunarLander-v2| DDQN        | 100%     | 100%        |
|              | M²DDQN ($N = 5$) | 100%     | 67.11%      |
|              | M²DDQN ($N = 10$) | 100%     | 62.34%      |
| MountainCar-v0| DDQN        | 100%     | 100%        |
|              | M²DDQN ($N = 5$) | 100%     | 52.77%      |
|              | M²DDQN ($N = 10$) | 99.86%   | 33.48%      |
| Acrobat-v1   | DDQN       | 100%     | -           |
|              | M²DDQN ($N = 5$) | 100%     | 106.04%     |
|              | M²DDQN ($N = 10$) | 100%     | 104.86%     |

Table 2: Normalized max score and normalized training step before a environment is solved (maximum evaluation score and steps to solve of DDQN is 100%) on CartPole-v1, LunarLander-v2, MountainCar-v0 and Acrobat-v1. Note that Acrobat-v1 is an unsolved environment, which means it does not have a specified reward threshold at which it's considered solved.

In CartPole-v1 task, we define "solving" as getting average evaluation score of 495.0 over 50 random games. For the quality of the best policy, all of three algorithms get a full evaluation score policy. For the learning speed, the training steps before the environment is solved of DDQN, M²DDQN ($N = 5$) and M²DDQN ($N = 10$) are 100%, 75.73% and 89.07%, respectively. Proposed method learns slightly faster than standard DDQN.

In LunarLander-v2 task, we define "solving" as getting average evaluation score of 200.0 over 50 random games. For the quality of the best policy, the maximum score of DDQN, M²DDQN ($N = 5$) and M²DDQN ($N = 10$) are 100%, 105.13% and 105.16%, respectively. For the learning speed, the training steps before the environment is solved of DDQN, M²DDQN ($N = 5$) and M²DDQN ($N = 10$) are 100%, 67.11% and 62.34%, respectively. Proposed method outperforms both in the quality of policy and learning speed.

In MountainCar-v0 task, we define "solving" as getting average evaluation score of −110.0 over 50 random games. For the quality of the best policy, the maximum score of DDQN, M²DDQN ($N = 5$) and M²DDQN ($N = 10$) are 100%, 98.92% and 99.86%, respectively. The maximum score of the three algorithms are almost the same. For the learning speed, the training steps before the environment is solved of DDQN, M²DDQN ($N = 5$) and M²DDQN ($N = 10$) are 100%, 52.77% and 33.48%, respectively. Proposed method leads to a great improvement in the learning of this sparse reward environment.

In Acrobat-v1 task, there is no definition of "solving" in the gym leaderboard. Therefore, we only focus on the quality of the best policy, the maximum score of DDQN, M²DDQN ($N = 5$) and M²DDQN ($N = 10$) are 100%, 106.04% and 104.86%, respectively. Proposed method get a better policy than standard DDQN.

In summary, the max-mean loss leads to a substantial improvement in all of the four tasks, especially in learning speed. Also, the choice of group size $N$ may affect the performance of proposed method, in our experiments, larger group size of higher performance.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a new framework of DQN to learn a policy more efficiently, by taking the max-mean loss instead of the standard loss. This method can be combined with most of existing
We thank Xuli Shen for useful discussions of the implementation (Acrobot-v1). The result shows that max-mean loss speeds up learning (SAC) [Haarnoja et al. 2018]. Also, more detailed experiments on different group size $N$ should be performed, in order to find the relation between the performance and group size $N$.

ACKNOWLEDGMENTS

We thank Xuli Shen for useful discussions of the implementation of mini-max method.

REFERENCES

Jacob Abernethy, Pranjal Awasthi, Matthias Klenkdesner, Jamie Morgenstern, Chris Russell, and Jie Zhang. 2020. Active sampling for min-max fairness. arXiv preprint arXiv:2008.06879 (2020).

Richard E Bellman. 2010. Dynamic programming. Princeton university press.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. arXiv preprint arXiv:1606.01540 (2016).

Xi Cao, Huaiyu Wan, Youfang Lin, and Sheng Han. 2019. High-value prioritized experience replay for off-policy reinforcement learning. In 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 1510–1514.

Erick Delage and Yinyu Ye. 2010. Distributionally robust optimization under moment uncertainty with application to data-driven problems. Operations research 58, 3 (2010), 595–612.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. (2014). arXiv:1406.2661 [stat ML].

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning. PMLR, 1861–1870.

Hanqing Jin and Shige Peng. 2005. Optimal unbiased estimation for maximal distributional robustness. Probability, Uncertainty and Quantitative Risk 6, 3 (2021), 189–198.

Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research 32, 11 (2013), 1238–1274.

Timothy P Lillicrap, Jonathan H Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015).

Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning 8 (1992), 293–321.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).

Shige Peng. 2005. Nonlinear expectations and nonlinear Markov chains. Chinese Annals of Mathematics 26, 02 (2005), 159–184.

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

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015).