Review for a conference paper at ICLR 2021

+

Randomized Ensembled Double Q-Learning: Learning Fast Without a Model

By Jungyeon Lee

May

2021

HONGIK UNIVERSITY

Presentation Agenda

Key topics for discussion

01	02	03	04	05	06
Abstract	Introduction	REDQ	REDQ가 성공한 이유	REDQ variants & ablations	Conclusion

Abstract

Using a high Update-To-Data (UTD) ratio, model-based methods have recently achieved much higher sample efficiency than previous model-free methods for continuous-action DRL benchmarks.

In this paper, we introduce a simple model-free algorithm, Randomized Ensembled Double Q-Learning (REDQ), and show that its performance is just as good as, if not better than, a state-of-the-art model-based algorithm for the MuJoCo benchmark.

Moreover, REDQ can achieve this performance using fewer parameters than the model-based method, and with less wall-clock run time. REDQ has three carefully integrated ingredients that allow it to achieve its high performance:

(i) a UTD ratio >> 1

(ii) an ensemble of Q functions

(iii) in-target minimization across a random subset of Q functions from the ensemble

Through carefully designed experiments, we provide a detailed analysis of REDQ and related model-free algorithms. To our knowledge, REDQ is the first successful model-free DRL algorithm for continuous-action spaces using a UTD ratio >> 1

Introduction

UTD 란

MBPO와의 비교

Ensemble of Qs + in-target minimization

UTD 란

Update-To-Data

the number of updates taken by the agent compared to the number of actual interactions with the environment
에이전트가 환경과의 실제 상호 작용 횟수와 비교한 업데이트 횟수

에이전트가 업데이트한 횟수
실제 환경과 인터렉션해서 얻은 데이터

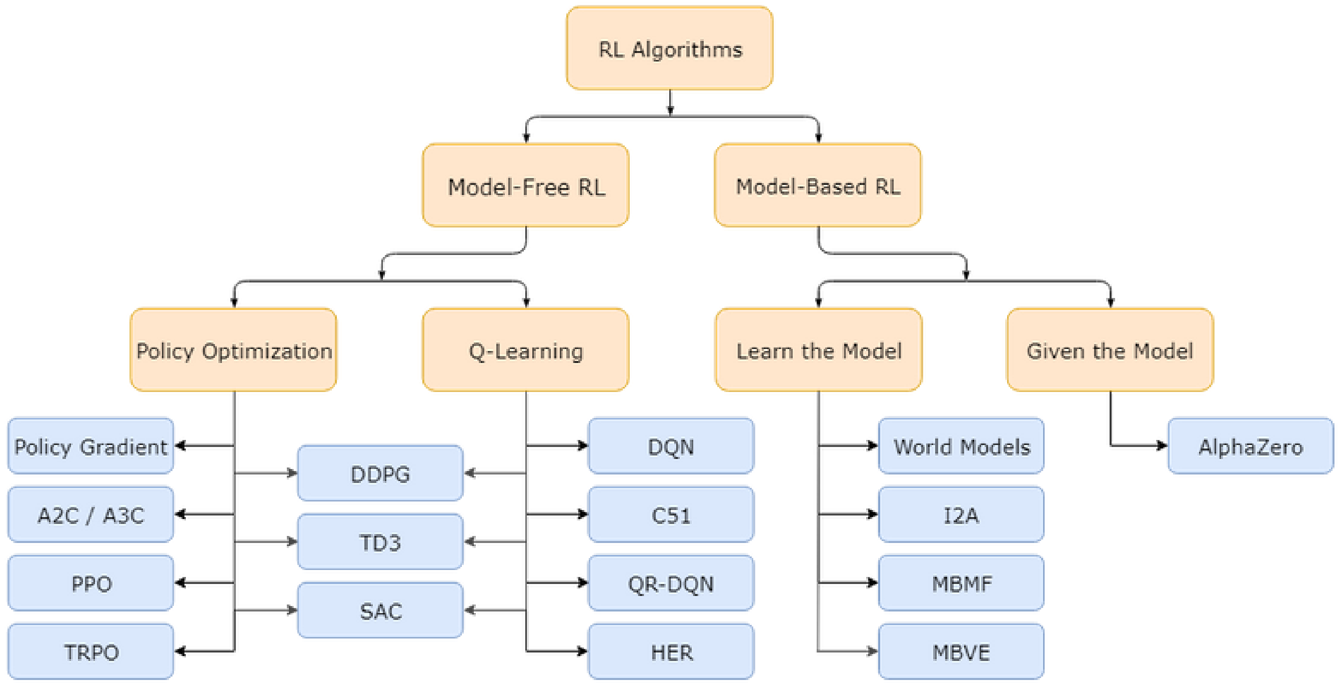
Model-based VS Model-free

based

- mix of real data from the environment and "fake" data from its model
- a large UTD ratio of 20-40
- MBPO

free

- UTD of 1
- SAC



Whether it is also possible to achieve such high performance without a model?

정말 모델 없이 좋은 퍼포먼스를 기대하기 힘들까?

http://wiki.hash.kr/index.php/%EB%AA%A8%EB%8D%B8_%ED%94%84%EB%A6%AC_%EA%B0%95%ED%99%94%ED%95%99%EC%8A%B5
http://wiki.hash.kr/index.php/%EB%AA%A8%EB%8D%B8_%EA%B8%B0%EB%B0%98_%EA%B0%95%ED%99%94%ED%95%99%EC%8A%B5

MBPO와의 비교

free vs based

REDQ	MBPO
• the performance • UTD ratio 1이상	
• model-free • fewer parameters • less wall-clock run time • no rollout; • updates with real data	• model-based • more parameters • more wall-clock run time • rollout • updates with real data and fake data

Ensemble of Qs + in-target minimization

another ingredients

Ensembles with in-target minimization

- reduce the std of the Q-function bias (close to zero)
- control the average Q-function bias
- (SAC와의 비교) much lower std of Q-function bias while maintaining an average bias that is negative but close to zero throughout most of training → better learning performance
- very robust to choices of hyperparameters
- work well with a small ensemble and a small number of Q functions in the in-target minimization

REDQ + OFENet

REDQ의 성능을 더 끌어올리기 위해 Online Feature Extractor Network라는 알고리즘과 합침

- Ant와 Humanoid에서 사용

REDQ

Key parameter와 특징

Algorithm

Experimental results

Key parameter와 특징

REDQ

REDQ can be used with any standard off-policy model-free algorithm, such as SAC, SOP, TD3, or DDPG

(i) a UTD ratio >> 1
(ii) an ensemble of Q functions
(iii) in-target minimization across a random subset of Q functions from the ensemble

Key parameter

G

UTD ratio

To improve sample efficiency,

the UTD ratio G is much greater than one

N

ensemble size

To reduce the variance in the Q-function estimate,

use an ensemble of N Q-functions, with each Q-function randomly and independently initialized
but updated with the same target

M

in-target minimization parameter

To reduce over-estimation bias,

the target for the Q-function includes a minimization over a random subset M of the N Q-functions
The size of the subset M is kept fixed
default : M = 2

Key parameter와 특징

REDQ

REDQ can be used with any standard off-policy model-free algorithm, such as SAC, SOP, TD3, or DDPG

(i) a UTD ratio >> 1
(ii) an ensemble of Q functions
(iii) in-target minimization across a random subset of Q functions from the ensemble

Key parameter	Examples
G UTD ratio	**G=1, N=M=2** the underlying off-policy algorithm such as SAC
N ensemble size	**G=1, N=M>2** similar to, but not equivalent to, Maxmin Q-learning
M in-target minimization parameter	**G=20, N=10, M=2** works well!

Algorithm

Algorithm 1 Randomized Ensembled Double Q-learning (REDQ)

Initialize policy parameters θ, N Q-function parameters ϕ_i, $i = 1, \ldots, N$, empty replay buffer $\mathcal{D}$. Set target parameters $\phi_{\text{targ},i} \leftarrow \phi_i$, for $i = 1, 2, \ldots, N$ ensemble size

repeat

  Take one action $a_t \sim \pi_\theta(\cdot|s_t)$. Observe reward r_t, new state s_{t+1}.

  Add data to buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$

  for G updates **do** Update ratio

    Sample a mini-batch $B = \{(s, a, r, s')\}$ from $\mathcal{D}$

    Sample a set $\mathcal{M}$ of M distinct indices from $\{1, 2, \ldots, N\}$ in-target minimization

    Compute the Q target y (same for all of the N Q-functions):

$$y = r + \gamma \left(\min_{i \in \mathcal{M}} Q_{\phi_{\text{targ},i}}\left(s', \tilde{a}'\right) - \alpha \log \pi_\theta\left(\tilde{a}' \mid s'\right) \right), \quad \tilde{a}' \sim \pi_\theta\left(\cdot \mid s'\right)$$

    for $i = 1, \ldots, N$ **do** Update all Qs

      Update ϕ_i with gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \left(Q_{\phi_i}(s, a) - y\right)^2$$

      Update target networks with $\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i$

  Update policy parameters θ with gradient ascent using Update policy

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\frac{1}{N} \sum_{i=1}^{N} Q_{\phi_i}\left(s, \tilde{a}_\theta(s)\right) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s) | s\right) \right), \quad \tilde{a}_\theta(s) \sim \pi_\theta(\cdot \mid s)$$

Experimental results

REDQ vs MBPO vs SAC

MBPO와의 공정한 비교를 위해

- MBPO 저자 코드 사용
- G=20과 같은 MBPO의 하이퍼파라미터 똑같이 사용
- REDQ : G=20, N=10, M=2
- 5번 indep. trials의 average return
- standard deviation : 5 seeds
- MBPO에서 환경과 인터렉션한 횟수와 똑같이 인터렉션

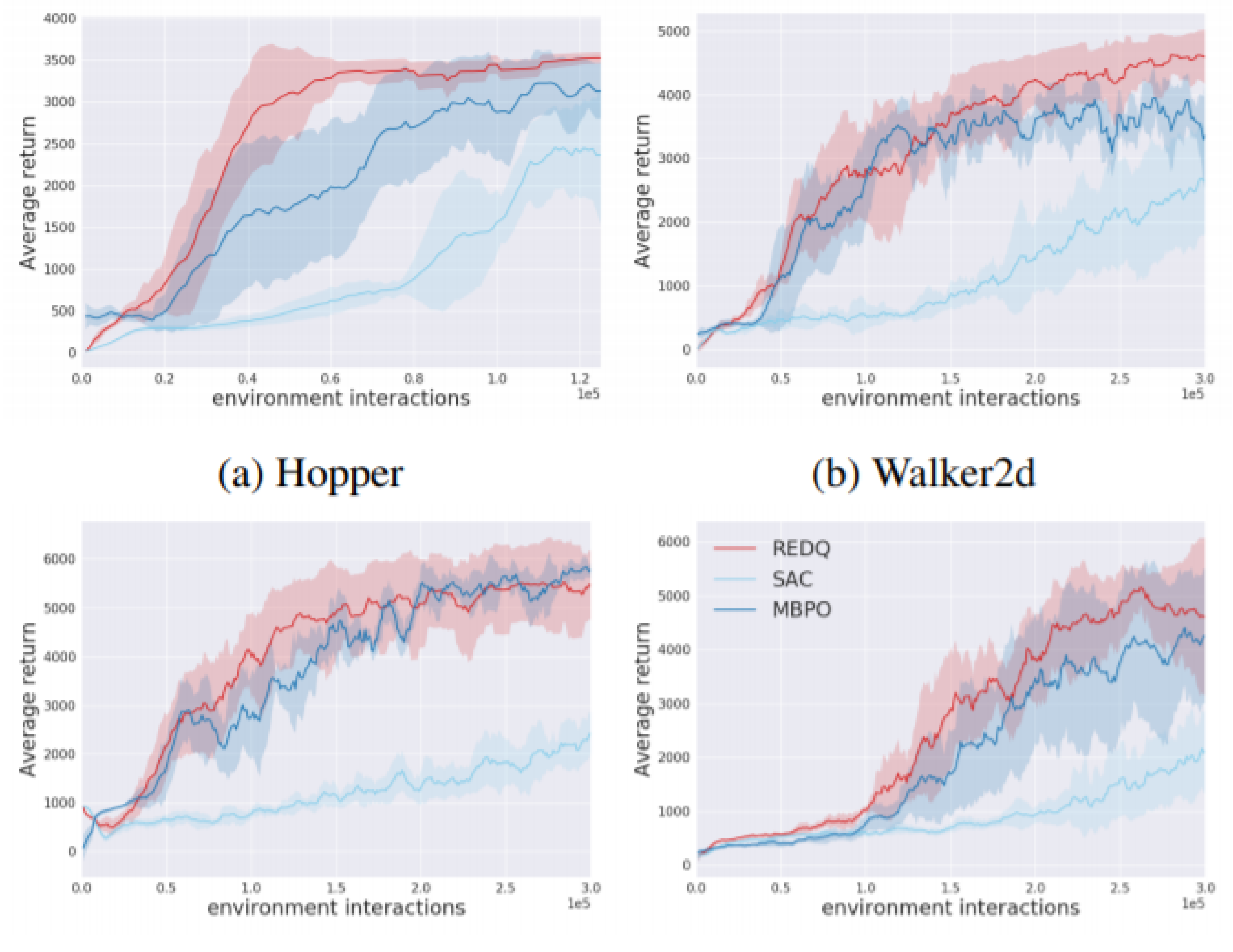
REDQ vs MBPO vs SAC

- REDQ와 MBPO가 SAC 보다 훨씬 빠름
- Hopper에서 매우 빠름
- REDQ performs 1.4x better than MBPO half-way through training and 1.1x better at the end of training

Computational resource 비교

Does REDQ achieve its sample efficiency using more computational resources than MBPO?

- measured the runtime on a 2080-Ti GPU and found that MBPO roughly takes 75% longer



(a) Hopper (b) Walker2d

(c) Ant (d) Humanoid

Figure 1: REDQ compared to MBPO and SAC. Both REDQ and MBPO use $G = 20$.

Algorithm	Hopper	Walker2d	Ant	Humanoid
MBPO	1.106M	1.144M	1.617M	7.087M
REDQ $N = 10$	0.769M	0.795M	1.066M	1.840M

REDQ가 성공한 이유

Estimation error

REDQ vs SAC-20 vs AVG

Theoretical Analysis

Estimation error

Q추정 편향에 대해 분석

(i) a UTD ratio >> 1
(ii) an ensemble of Q functions
(iii) in-target minimization across a random subset of Q functions from the ensemble

Key to REDQ's sample efficiency is using a UTD >>1

Why is it that SAC and ordinary ensemble averaging (AVG) cannot do as well as REDQ by simply increasing the UTD?

그 전에,

Q추정 편향 분석을 위한 몇가지 notation 정리

$Q^{\pi}(s,a)$ the action-value function for policy π using the standard infinite-horizon discounted return definition

$Q_{\phi}(s,a)$ 추정값 $Q_{\phi_i}(s,a),\ i=1,\ldots,N$

$Q_{\phi}(s,a)-Q^{\pi}(s,a)$ the bias of an estimate → $\dfrac{(Q_{\phi}(s,a)-Q^{\pi}(s,a))}{|E_{\bar{s},\bar{a}\sim\pi}[Q^{\pi}(\bar{s},\bar{a})]|}$

How estimation error accumulates in the training process
→ calculate the average and std of these bias values

- Average : whether Qφ is in general overestimating or underestimating
- Std : how uniform the bias is across different state-action pairs

REDQ vs SAC-20 vs AVG

Why is it that SAC and ordinary ensemble averaging (AVG) cannot do as well as REDQ by simply increasing the UTD?

SAC-20

- SAC but with G increased from 1 (as in standard SAC) to 20

AVG

- ensemble
- when computing the Q target, we take the average of all Q values without any in-target minimization

(i) a UTD ratio >> 1
(ii) an ensemble of Q functions
(iii) in-target minimization across a random subset of Q functions from the ensemble

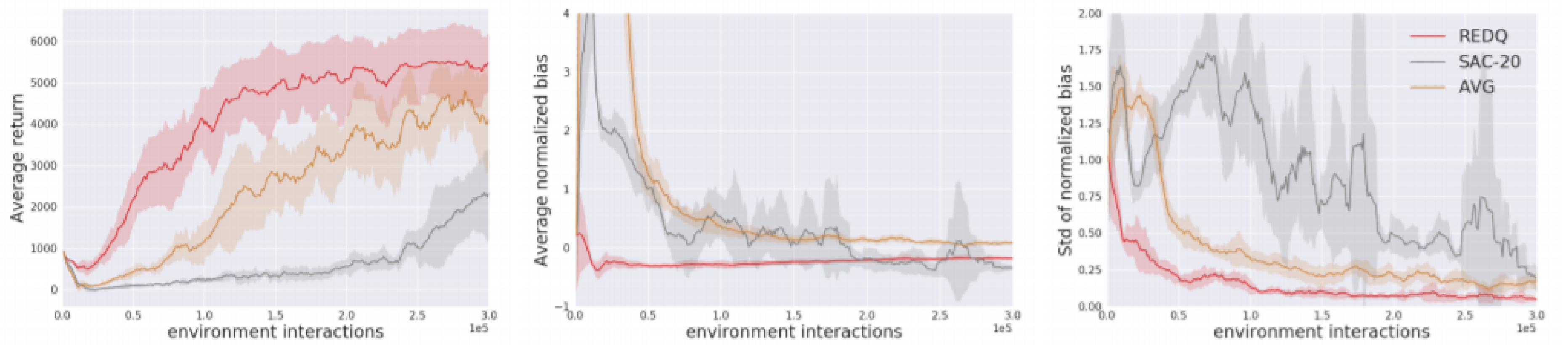
REDQ vs SAC-20 vs AVG

Why is it that SAC and ordinary ensemble averaging (AVG) cannot do as well as REDQ by simply increasing the UTD?

REDQ vs SAC-20 vs AVG

- All : UTD of G = 20
- REDQ learns significantly faster than both SAC-20 and AVG
- REDQ has a very low normalized std of bias for most of training
- REDQ has a small and near-constant under-estimation bias
- The shaded areas for mean and std of bias are also smaller, indicating that REDQ is robust to random initial conditions
- AVG performs significantly better than SAC-20 in Ant and Humanoid
  - explained by the bias: due to ensemble averaging, AVG can achieve a lower std of bias; and when it does, its performance improves significantly faster than SAC-20

the success of REDQ is largely due to a careful integration of both of these critical components.



(a) Performance, Ant (b) Average bias, Ant (c) Std of bias, Ant

Figure 2: Performance, mean and std of normalized Q bias for REDQ, AVG, and SAC for Ant. Figures for the other three environments have similar trends and are shown in the Appendix.

Theoretical Analysis

Relation between the estimation error, the in-target minimization parameter M and the size of the ensemble N

Tabular version of REDQ

the target for $Q^i(s,a)$ for each $i = 1, \ldots, N$ is:

$$r + \gamma \max_{a' \in \mathcal{A}} \min_{j \in \mathcal{M}} Q^j(s', a') \tag{1}$$

where $\mathcal{A}$ is the finite action space, (s, a, r, s') is a transition, and $\mathcal{M}$ is again a uniformly random subset from $\{1, \ldots, N\}$ with $|\mathcal{M}| = M$.

How the bias changes after an update
How this change is effected by M and N

the post-update estimation bias

$$\begin{aligned} Z_{M,N} &\triangleq r + \gamma \max_{a' \in \mathcal{A}} \min_{j \in \mathcal{M}} Q^j(s', a') - (r + \gamma \max_{a' \in \mathcal{A}} Q^{\pi}(s', a')) \\ &= \gamma (\max_{a' \in \mathcal{A}} \min_{j \in \mathcal{M}} Q^j(s', a') - \max_{a' \in \mathcal{A}} Q^{\pi}(s', a')) \end{aligned}$$

Theoretical Analysis

Relation between the estimation error, the in-target minimization parameter M and the size of the ensemble N

Random approximation error

$$Q^i(s,a) = Q^\pi(s,a) + e^i_{sa}$$

for each fixed s

- zero-mean independent random variables
- identically distributed across i for each fixed (s, a) pair

Due to the zero-mean assumption,
the expected pre-update estimation bias is $\mathbb{E}[Q^i(s,a) - Q^\pi(s,a)] = 0$

$\mathbb{E}[Z_{M,N}] > 0$ over-estimation accumulation

$\mathbb{E}[Z_{M,N}] < 0$ under-estimation accumulation

Theoretical Analysis

Relation between the estimation error, the in-target minimization parameter M and the size of the ensemble N

Bias control

Theorem 1. 1. *For any fixed* M, $\mathbb{E}[Z_{M,N}]$ *does not depend on* N.

2. $\mathbb{E}[Z_{1,N}] \geq 0$ *for all* $N \geq 1$.

3. $\mathbb{E}[Z_{M+1,N}] \leq \mathbb{E}[Z_{M,N}]$ *for any* $M < N$.

4. *Suppose that* $e^i_{sa} \leq c$ *for some* $c > 0$ *for all* s, a *and* i. *Then there exists an* M *such that for all* $N \geq M$, $\mathbb{E}[Z_{M,N}] < 0$.

$\mathbb{E}[Z_{M,N}]$

(2~4) can control the expected post-update bias, bringing it from above zero (over estimation) to under zero (under estimation) by increasing M

(1) the expected bias only depends on M

→ control the post-update bias with M

separately control the variance of the average of the ensemble with N

Theoretical Analysis

Relation between the estimation error, the in-target minimization parameter M and the size of the ensemble N

Variants:instead of choosing a random set of size M

$$Y_{M,N} = r(s,a) + \gamma \frac{1}{\binom{N}{M}} \sum_{\substack{B \subset \mathcal{N} \\ |B| = M}} \max_{a'} \min_{j \in B} Q^j(s', a')$$

calculate the target by taking the expected value over all possible subsets of size M

this variant of REDQ = "Weighted"
since we can efficiently calculate the target as a weighted sum of a re-ordering of the N Q-function

Let $v_M := \mathrm{Var}(\max_{a'} \min_{j \in B} Q^j(s', a'))$ for any subset $B \subset \mathcal{N}$ where $|B| = M$. (It is easily seen that v_M only depends on M and not only the specific elements of B.)

Theorem 2.

$$\mathrm{Var}(Y_{M,N}) \le G_M(N)$$

for some function $G_M(N)$ satisfying

$$\lim_{N \to \infty} \frac{G_M(N)}{M^2 v_M / N} = 1$$

Consequently,

$$\lim_{N \to \infty} \mathrm{Var}(Y_{M,N}) = 0$$

REDQ variants & ablations

by ensemble size N

by in-target minimization parameter M

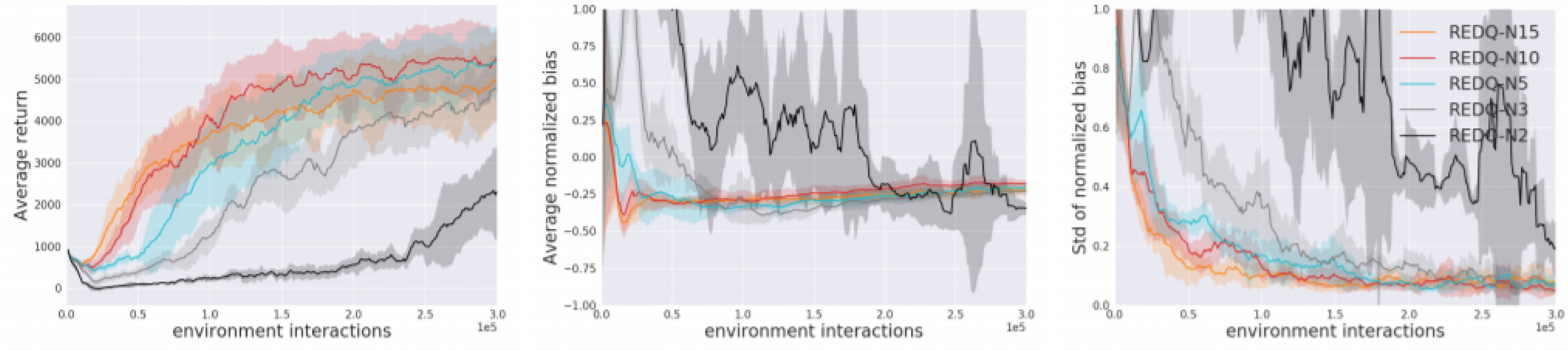
by target computation methods

Improving REDQ
with Auxiliary Feature Learning

by ensemble size N

How the ensemble size N affects REDQ

- when we increase the ensemble size, we generally get a more stable average bias, a lower std of bias, and stronger performance
- even a small ensemble (e.g., N = 5) can greatly help in stabilizing bias accumulation when training under high UTD

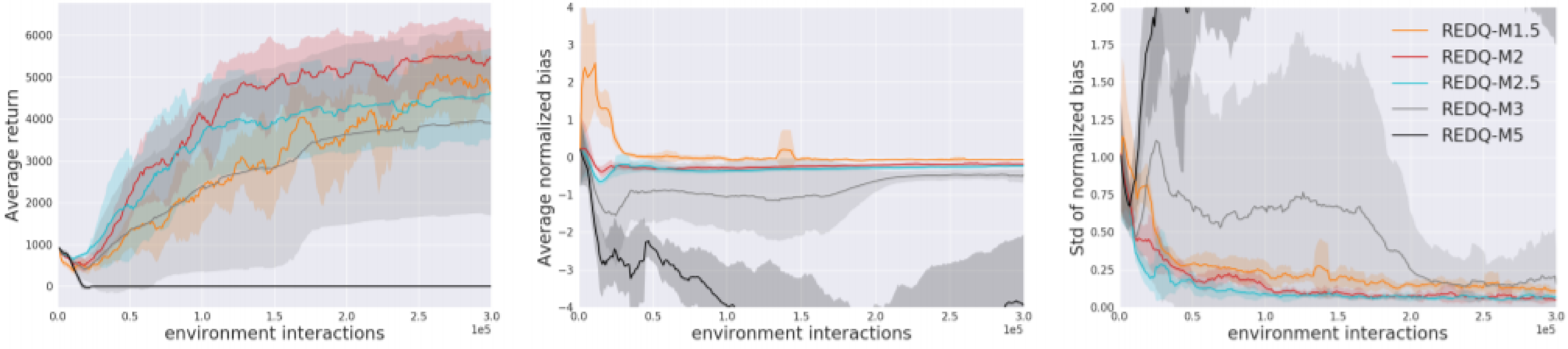


(a) Performance, Ant (b) Average bias, Ant (c) std of bias, Ant

by in-target minimization parameter M

How M, the in-target minimization parameter, can affect performance

- By increasing M we lower the average bias
- When M gets too large, the Q estimate becomes too conservative and the large negative bias makes learning difficult
- M = 2, which has the overall best performance, strikes a good balance between average bias (small underestimation during most of training) and std of the bias (consistently small)



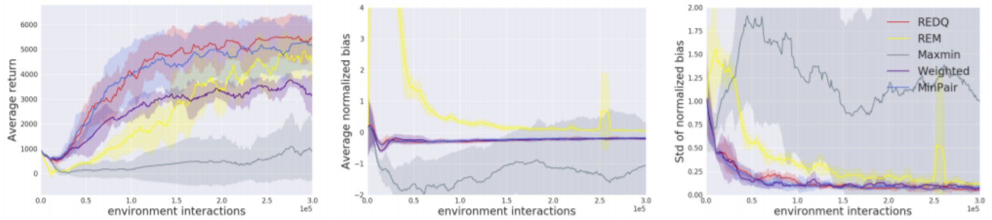
(d) Performance, Ant

(e) Average Q bias, Ant

(f) std of Q Bias, Ant

by target computation methods

REDQ, REM, Maxmin, Weighted, MinPair



(g) Performance, Ant (h) Average bias, Ant (i) std of bias, Ant

1.Maxmin

- min of all the Q networks in the ensemble is taken to compute the Q target
- As the ensemble size increases, Maxmin Q-learning shifts from overestimation to underestimation
- a large N value will cause even more divergence of the Q values
- When we increase the ensemble size to be larger than 3, Maxmin starts to reduce the bias so much that we get a highly negative Q bias, which accumulates quickly, leading to instability in the Q networks and poor performance
- continuous action Q-learning based methods such as DDPG & SAC suffer much more from Q bias accumulation compared to discrete action methods

2. REM

- use the random convex combination of Q values to compute the target
- similar to ensemble average (AVG), but with more randomization

3. Weighted

- the target is computed as the expectation of all the REDQ targets, where the expectation is taken over all N-choose-2 pairs of Q-functions
- This leads to a formula that is a weighted sum of the ordered Q-functions, where the ordering is from the lowest to the highest Q value in the ensemble

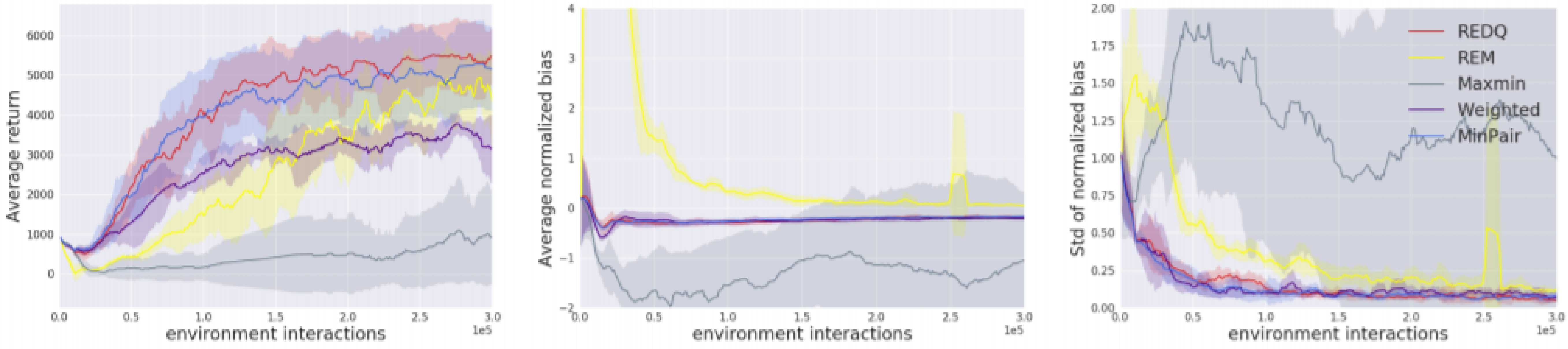
4. MinPair

- divide the 10 Q networks into 5 fixed pairs, and during an update we sample a pair of Q networks from these 5 fixed pairs

by target computation methods

REDQ, REM, Maxmin, Weighted, MinPair

- REDQ & MinPair: the best and similar performance
- For Ant, the performance of Weighted is much lower than REDQ
- However, Weighted & REDQ have similar performance for the other three environments (Appendix)
- In terms of the Q bias, REM has a positive average bias, while REDQ, MinPair, and Weighted all have a small negative average bias
- Overall these results indicate that the REDQ algorithm is robust across different mechanisms for choosing the functions, and that randomly choosing the Q functions can sometimes boost performance



(g) Performance, Ant　(h) Average bias, Ant　(i) std of bias, Ant

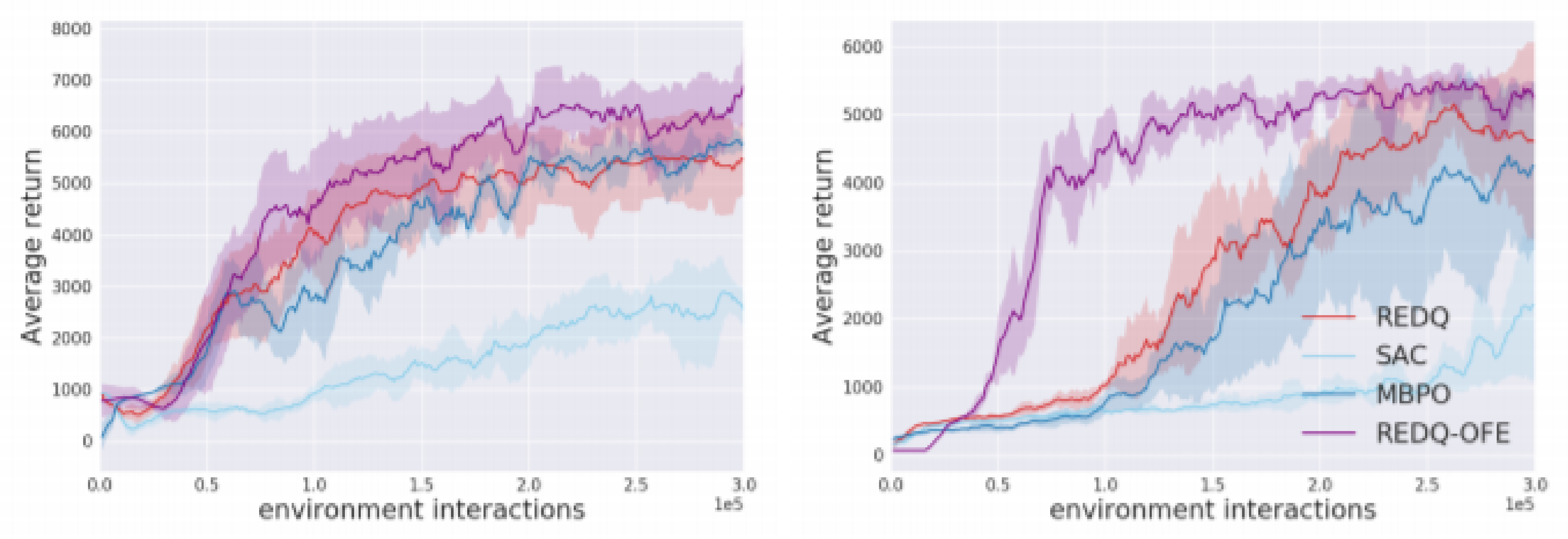
Improving REDQ with Auxiliary Feature Learning

whether we can further improve the performance of REDQ by incorporating better representation learning?

OFENet

learn representation vectors from environment data, and provides them to the agent as additional input, giving significant performance improvement

- Hopper and Walker2d에서는 효과 x
- Ant and Humanoid에서는 효과o
- 7x the sample efficiency of SAC to reach 5000 on Ant and Humanoid
- outperforms MBPO with 3.12x and 1.26x the performance of MBPO at 150K and 300K data, respectively



(a) Performance, Ant (b) Performance, Humanoid

Figure 4: Performance of REDQ, REDQ with OFE, and SAC.

Conclusion

The contributions of this paper are as follows.

1. We propose a simple model-free algorithm that attains sample efficiency that is as good as or better than SOTA model-based algorithms for the MuJoCo benchmark. This result indicates that, at least for the MuJoCo benchmark, **models may** ***not be necessary*** for achieving high sample efficiency.
2. Using carefully designed experiments, we explain *why REDQ succeeds when other model-free algorithms* with high UTD ratios fail.
3. Finally, we combine *REDQ with OFE*, and show that REDQ-OFE can learn extremely fast for the challenging environments Ant and Humanoid.

Reference

참고 코드와 자료

Official Code

https://github.com/watchernyu/REDQ

Medium

https://medium.com/analytics-vidhya/randomized-ensembled-double-q-learning-learning-fast-without-a-model-11b25e2fc3a8

Reference Code

https://github.com/BY571/Randomized-Ensembled-Double-Q-learning-REDQ-