






Understanding Failures of Deterministic Actor-Critic with Continuous Action Spaces and Sparse Rewards

Guillaume Matheron^(✉), Nicolas Perrin^{}, and Olivier Sigaud^{}

Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique,
ISIR, 75005 Paris, France
guillaume_pub@matheron.eu

Abstract. In environments with continuous state and action spaces, state-of-the-art actor-critic reinforcement learning algorithms can solve very complex problems, yet can also fail in environments that seem trivial, but the reason for such failures is still poorly understood. In this paper, we contribute a formal explanation of these failures in the particular case of sparse reward and deterministic environments. First, using a very elementary control problem, we illustrate that the learning process can get stuck into a fixed point corresponding to a poor solution, especially when the reward is not found very early. Then, generalizing from the studied example, we provide a detailed analysis of the underlying mechanisms which results in a new understanding of one of the convergence regimes of these algorithms.

1 Introduction

The Deep Deterministic Policy Gradient (DDPG) algorithm [11] is one of the earliest deep Reinforcement Learning (RL) algorithms designed to operate on potentially large continuous state and action spaces with a deterministic policy, and it is still one of the most widely used. However, it is often reported that DDPG suffers from instability in the form of sensitivity to hyper-parameters and propensity to converge to very poor solutions or even diverge. Various algorithms have improved stability by addressing well identified issues, such as the over-estimation bias in TD3 [7] but, because a fundamental understanding of the phenomena underlying these instabilities is still missing, it is unclear whether these ad hoc remedies truly address the source of the problem. Thus, better understanding why these algorithms can fail even in very simple environments is a pressing question.

To investigate this question, we introduce in Sect. 4 a very simple one-dimensional environment with a sparse reward function in which DDPG sometimes fails. Analyzing this example allows us to provide a detailed account of

This work was partially supported by the French National Research Agency (ANR), Project ANR-18-CE33-0005 HUSKI.

these failures. We then reveal the existence of a cycle of mechanisms operating in the sparse reward and deterministic case, leading to the quick convergence to a poor policy. In particular, we show that when the reward is not discovered early enough, these mechanisms can lead to a *deadlock* situation where neither the actor nor the critic can evolve anymore. Critically, this deadlock persists even when the agent is subsequently trained with rewarded samples.

The study of these mechanisms is backed-up with formal analyses in a simplified context where the effects of function approximation is ignored. Nevertheless, the resulting understanding helps analyzing the practical phenomena encountered when using actors and critics represented as neural networks.

2 Related Work

Issues when combining RL with function approximation have been studied for a long time [3, 4, 17]. In particular, it is well known that deep RL algorithms can diverge when they meet three conditions coined as the “deadly triad” [16], that is when they use (1) function approximation, (2) bootstrapping updates and (3) off-policy learning. However, these questions are mostly studied in the continuous state, discrete action case. For instance, several recent papers have studied the mechanism of this instability using DQN [12]. In this context, four failure modes have been identified from a theoretical point of view by considering the effect of a linear approximation of the deep-Q updates and by identifying conditions under which the approximate updates of the critic are contraction maps for some distance over Q-functions [1]. Meanwhile, [10] shows that, due to its stabilizing heuristics, DQN does not diverge much in practice when applied to the ATARI domain.

In contrast to these papers, here we study a failure mode specific to continuous action actor-critic algorithms. It hinges on the fact that one cannot take the maximum over actions, and must rely on the actor as a proxy for providing the optimal action instead. Therefore, the failure mode identified in this paper cannot be reduced to any of the ones that affect DQN. Besides, the formal analyses presented in this article show that the failure mode we are investigating does not depend on function approximation errors, thus it cannot be directly related to the deadly triad.

More related to our work, several papers have studied failure to gather rewarded experience from the environment due to poor exploration [5, 6, 14], but we go beyond this issue by studying a case where the reward is actually found but not properly exploited. Finally, like us the authors of [8] study a failure mode which is specific to DDPG-like algorithms, but the studied failure mode is different. They show under a batch learning regime that DDPG suffers from an *extrapolation error* phenomenon, whereas we are in the more standard incremental learning setting and focus on a deadlock resulting from the shape of the Q-function in the sparse reward case.

3 Background: Deep Deterministic Policy Gradient

The DDPG algorithm [11] is a deep RL algorithm based on the Deterministic Policy Gradient theorem [15]. It borrows the use of a replay buffer and target networks from DQN [13]. DDPG is an instance of the Actor-Critic model. It learns both an actor function π_ψ (also called policy) and a critic function Q_θ , represented as neural networks whose parameters are respectively noted ψ and θ .

The deterministic actor takes a state $s \in S$ as input and outputs an action $a \in A$. The critic maps each state-action pair (s, a) to a value in \mathbb{R} . The reward $r : S \times A \rightarrow \mathbb{R}$, the termination function $t : S \times A \rightarrow \{0, 1\}$ and the discount factor $\gamma < 1$ are also specified as part of the environment.

The actor and critic are updated using stochastic gradient descent on two losses L_ψ and L_θ . These losses are computed from mini-batches of samples $(s_i, a_i, r_i, t_i, s_{i+1})$, where each sample corresponds to a transition $s_i \rightarrow s_{i+1}$ resulting from performing action a_i in state s_i , with subsequent reward $r_i = r(s_i, a_i)$ and termination index $t_i = t(s_i, a_i)$.

Two target networks $\pi_{\psi'}$ and $Q_{\theta'}$ are also used in DDPG. Their parameters ψ' and θ' respectively track ψ and θ using exponential smoothing. They are mostly useful to stabilize function approximation when learning the critic and actor networks. Since they do not play a significant role in the phenomena studied in this paper, we ignore them in our formal analyses.

Equations (1) and (2) define L_ψ and L_θ :

$$L_\psi = - \sum_i Q_\theta(s_i, \pi_\psi(s_i)) \quad (1)$$

$$\begin{cases} \forall i, y_i = r_i + \gamma(1 - t_i)Q_{\theta'}(s_{i+1}, \pi_{\psi'}(s_{i+1})) \\ L_\theta = \sum_i [Q_\theta(s_i, a_i) - y_i]^2. \end{cases} \quad (2)$$

Training for the loss given in (1) yields the parameter update in (3), with α the learning rate:

$$\psi \leftarrow \psi + \alpha \sum_i \frac{\partial \pi_\psi(s_i)}{\partial \psi}^T \nabla_a Q_\theta(s_i, a)|_{a=\pi_\psi(s_i)}. \quad (3)$$

As DDPG uses a replay buffer, the mini-batch samples are acquired using a behavior policy β which may be different from the actor π . Usually, β is defined as π plus a noise distribution, which in the case of DDPG is either a Gaussian function or the more sophisticated Ornstein-Uhlenbeck noise.

Importantly for this paper, the behavior of DDPG can be characterized as an intermediate between two extreme regimes:

- When the actor is updated much faster than the critic, the policy becomes greedy with respect to this critic, resulting into a behavior closely resembling that of the Q-LEARNING algorithm. When it is close to this regime, DDPG can be characterized as off-policy.

- When the critic is updated much faster than the actor, the critic tends towards $Q^\pi(s, a)$. The problems studied in this paper directly come from this second regime.

4 A New Failure Mode

In this section, we introduce a simplistic environment which we call 1D-TOY. It is a one-dimensional, discrete-time, continuous state and action problem, depicted in Fig. 1.

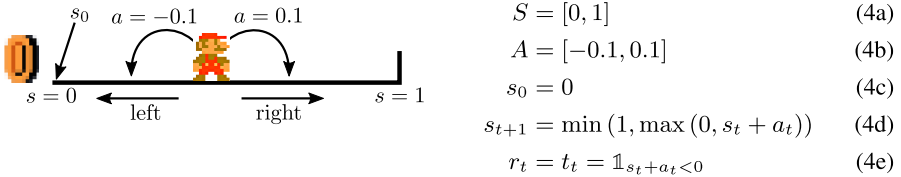


Fig. 1. The 1D-TOY environment

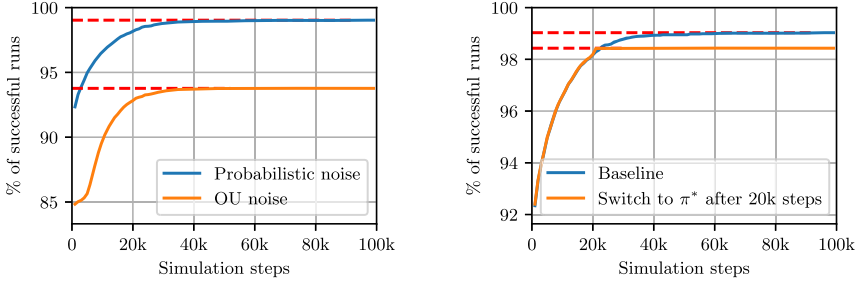
Despite its simplicity, DDPG can fail on 1D-TOY. We first show that DDPG fails to reach 100% success. We then show that if learning a policy does not succeed soon enough, the learning process can get stuck. Besides, we show that the initial actor can be significantly modified in the initial stages before finding the first reward. We explain how the combination of these phenomena can result into a deadlock situation. We generalize this explanation to any deterministic and sparse reward environment by revealing and formally studying a undesirable cyclic process which arises in such cases. Finally, we explore the consequences of getting into this cyclic process.

4.1 Empirical Study

In all experiments, we set the maximum episode length N to 50, but the observed phenomena persist with other values.

Residual Failure to Converge Using Different Noise Processes. We start by running DDPG on the 1D-TOY environment. This environment is trivial as one infinitesimal step to the left is enough to obtain the reward, end the episode and succeed, thus we might expect a quick 100% success. However, the first attempt using an Ornstein-Uhlenbeck (OU) noise process shows that DDPG succeeds in only 94% of cases, see Fig. 2a.

These failures might come from an exploration problem. Indeed, at the start of each episode the OU noise process is reset to zero and gives little noise in the first steps of the episode. In order to remove this potential source of failure, we replace the OU noise process with an exploration strategy similar to ϵ -greedy



(a) Success rate of DDPG with Ornstein-Uhlenbeck (OU) and probabilistic noise. Even with probabilistic noise, DDPG fails on about 1% of the seeds. (b) Comparison between DDPG with probabilistic noise and a variant in which the behavior policy is set to the optimal policy π^* after 20k steps.

Fig. 2. Success rate of variants of DDPG on 1D-TOY over learning steps ($N = 10k$).

which we call “probabilistic noise”. For some $0 < p < 1$, with probability p , the action is randomly sampled (and the actor is ignored), and with probability $1 - p$ no noise is used and the raw action is returned. In our tests, we used $p = 0.1$. This guarantees at least a 5% chance of success at the first step of each episode, for any policy. Nevertheless, Fig. 2a shows that even with probabilistic noise, about 1% of seeds still fail to converge to a successful policy in 1D-TOY, even after 100k training steps. All the following tests are performed using probabilistic noise.

We now focus on these failures. On all failing seeds, we observe that the actor has converged to a saturated policy that always goes to the right ($\forall s, \pi(s) = 0.1$). However, some mini-batch samples have non-zero rewards because the agent still occasionally moves to the left, due to the probabilistic noise applied during rollouts. The expected fraction of non-zero rewards is slightly more than 0.1%¹. Fig. 3a shows the occurrence of rewards in minibatches taken from the replay buffer when training DDPG on 1D-TOY. After each rollout (episode) of n steps, the critic and actor networks are trained n times on minibatches of size 100. So for instance, a failed episode of size 50 is followed by a training on a total of 5000 samples, out of which we expect more than 5 in average are rewarded transitions.

The constant presence of rewarded transitions in the minibatches suggests that the failures of DDPG on this environment are not due to insufficient exploration by the behavior policy.

Correlation Between Finding the Reward Early and Finding the Optimal Policy.

We have shown that DDPG can get stuck in 1D-TOY despite finding the reward regularly. Now we show that when DDPG finds the reward early in the training session, it is also more successful in converging to the optimal policy. On the

¹ 10% of steps are governed by probabilistic noise, of which at least 2% are the first episode step, of which 50% are steps going to the left and leading to the reward.

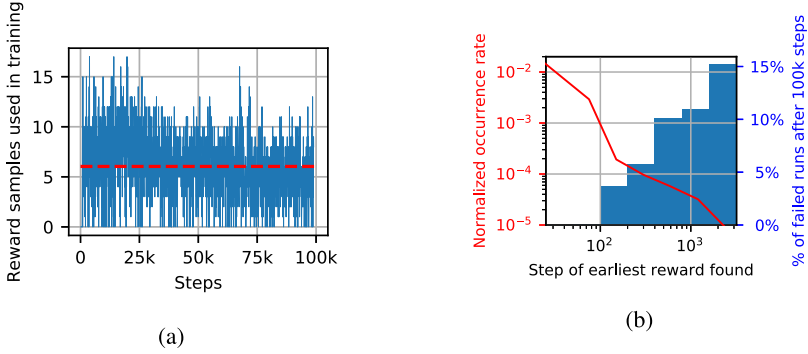


Fig. 3. (a) Number of rewards found in mini-batches during training. After a rollout of n steps, the actor and critic are both trained on n minibatches of size 100. The red dotted line indicates an average of 6.03 rewarded transitions present in these n minibatches. (b) In red, normalized probability of finding the earliest reward at this step. In blue, for each earliest reward bin, fraction of these episodes that fail to converge to a good actor after 100k steps. Note that when the reward is found after one or two episodes, the convergence to a successful actor is certain. (Color figure online)

other hand, when the first reward is found late, the learning process more often gets stuck with a sub-optimal policy.

From Fig. 3b, the early steps appear to have a high influence on whether the training will be successful or not. For instance, if the reward is found in the first 50 steps by the actor noise (which happens in 63% of cases), then the success rate of DDPG is 100%. However, if the reward is first found after more than 50 steps, then the success rate drops to 96%. Figure 3b shows that finding the reward later results in lower success rates, down to 87% for runs in which the reward was not found in the first 1600 steps. Therefore, we claim that there exists a critical time frame for finding the reward in the very early stages of training.

Spontaneous Actor Drift. At the beginning of each training session, the actor and critic of DDPG are initialized to represent respectively close-to-zero state-action values and close-to-zero actions. Besides, as long as the agent does not find a reward, it does not benefit from any utility gradient. Thus we might expect that the actor and critic remain constant until the first reward is found. Actually, we show that even in the absence of reward, training the actor and critic triggers non-negligible updates that cause the actor to reach a saturated state very quickly.

To investigate this, we use a variant of 1D-TOY called DRIFT where the only difference is that no rewarded or terminal transitions are present in the environment. We also use a stripped-down version of DDPG, removing rollouts and using random sampling of states and actions as minibatches for training.

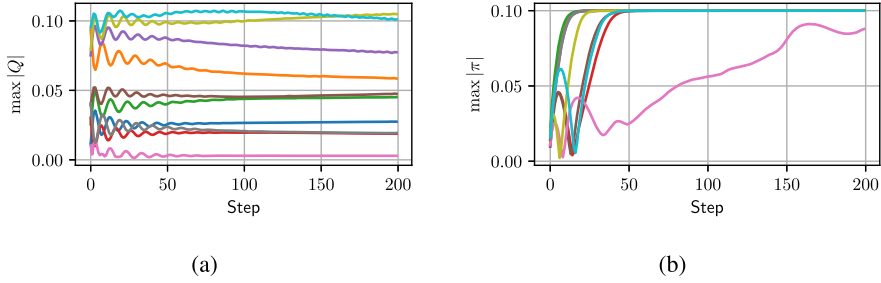
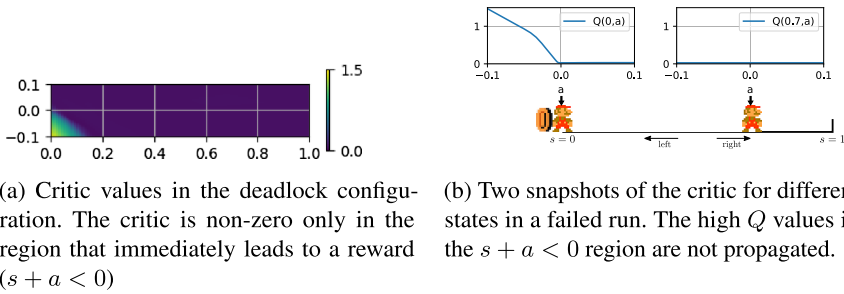


Fig. 4. Drift of $\max|Q|$ and $\max|\pi|$ in the DRIFT environment, for 10 different seeds. In the absence of reward, the critic oscillates briefly before stabilizing. However, the actor very quickly reaches a saturated state, at either $\forall s, \pi(s) = 0.1$ or -0.1 .



(a) Critic values in the deadlock configuration. The critic is non-zero only in the region that immediately leads to a reward ($s + a < 0$)

(b) Two snapshots of the critic for different states in a failed run. The high Q values in the $s + a < 0$ region are not propagated.

Fig. 5. Visualization of the critic in a failing run, in which the actor is stuck to $\forall s, \pi(s) = 0.1$.

Figure 4b shows that even in the absence of reward, the actor function drifts rapidly (notice the horizontal scale in steps) to a saturated policy, in a number of steps comparable to the “critical time frame” identified above. The critic also has a transitive phase before stabilizing.

In Fig. 4a, the fact that $\max_{s,a} |Q(s, a)|$ can increase in the absence of reward can seem counter-intuitive, since in the loss function presented in Eq. (2), $|y_i|$ can never be greater than $\max_{s,a} |Q(s, a)|$. However, it should be noted that the changes made to Q are not local to the minibatch points, and increasing the value of Q for one input (s, a) may cause its value to increase for other inputs too, which may cause an increase in the global maximum of Q . This phenomenon is at the heart of the over-estimation bias when learning a critic [7], but this bias does not play a key role here.

4.2 Explaining the Deadlock Situation for DDPG on 1D-TOY

Up to now, we have shown that DDPG fails about 1% of times on 1D-TOY, despite the simplicity of this environment. We have now collected the necessary elements to explain the mechanisms of this deadlock in 1D-TOY.

Figure 5 shows the value of the critic in a failed run of DDPG on 1D-TOY. We see that the value of the reward is not propagated correctly outside of the region in which the reward is found in a single step $\{(s, a) \mid s + a < 0\}$. The key of the deadlock is that once the actor has drifted to $\forall s, \pi(s) = 0.1$, it is updated according to $\nabla_a Q_\theta(s, a)|_{a=\pi_\psi(s)}$ (Eq. (3)). Figure 5b shows that for $a = \pi(s) = 0.1$, this gradient is zero therefore the actor is not updated. Besides, the critic is updated using $y_i = r(s_i, a_i) + \gamma Q(s'_i, \pi(s'_i))$ as a target. Since $Q(s'_i, 0.1)$ is zero, the critic only needs to be non-zero for directly rewarded actions, and for all other samples the target value remains zero. In this state the critic loss given in Eq. (2) is minimal, so there is no further update of the critic and no further propagation of the state-action values. The combination of the above two facts clearly results in a deadlock.

Importantly, the constitutive elements of this deadlock do not depend on the batches used to perform the update, and therefore do not depend on the experience selection method. We tested this experimentally by substituting the behavior policy for the optimal policy after 20k training steps. Results are presented in Fig. 2b and show that, once stuck, even when it is given ideal samples, DDPG stays stuck in the deadlock configuration. This also explains why finding the reward early results in better performance. When the reward is found early enough, $\pi(s_0)$ has not drifted too far, and the gradient of $Q(s_0, a)$ at $a = \pi(s_0)$ drives the actor back into the correct direction.

Note however that even when the actor drifts to the right, DDPG does not always fail. Indeed, because of function approximators the shape of the critic when finding the reward for the first time varies, and sometimes converges slowly enough for the actor to be updated before the convergence of the critic.

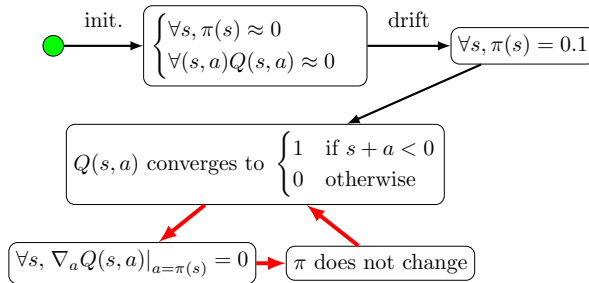


Fig. 6. Deadlock observed in 1D-TOY, represented as the cycle of red arrows. (Color figure online)

Figure 6 summarizes the above process. The entry point is represented using a green dot. First, the actor drifts to $\forall s, \pi(s) = 0.1$, then the critic converges to

Q^π which is a piecewise-constant function, which in turn means that the critic provides no gradient, therefore the actor is not updated (as seen in Eq. 3)².

4.3 Generalization

Our study of 1D-TOY revealed how DDPG can get stuck in this simplistic environment. We now generalize to the broader context of more general continuous action actor critic algorithms, including at least DDPG and TD3, and acting in any deterministic and sparse reward environment. The generalized deadlock mechanism is illustrated in Fig. 7 and explained hereafter in the idealized context of perfect approximators.

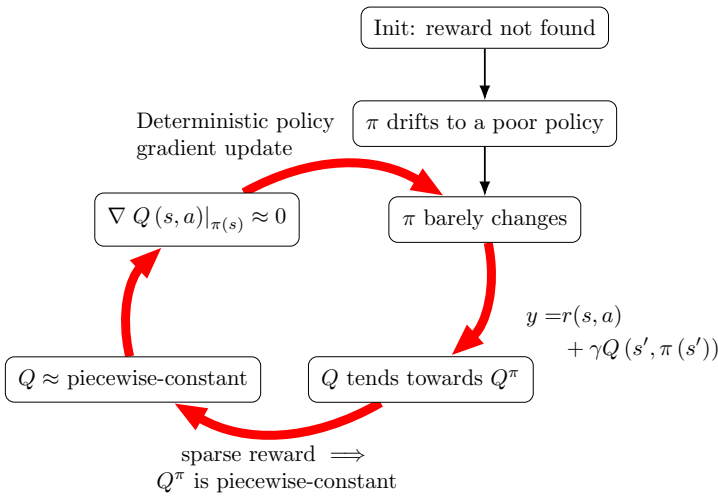


Fig. 7. A cyclic view of the undesirable convergence process in continuous action actor-critic algorithms, in the deterministic and sparse reward case. (Color figure online)

Entry Point: As shown in the previous section, before the behavior policy finds any reward, training the actor and critic can still trigger non-negligible updates that may cause the actor to quickly reach a poor state and stabilize. This defines our entry point in the process.

Q Tends Towards Q^π : A first step into the cycle is that, if the critic is updated faster than the policy, the update rule of the critic Q given in Eq. (2) makes Q converge to Q^π . Indeed, if π is fixed, Q is updated regularly via approximate dynamic programming with the Bellman operator for the policy π . Under strong assumptions, or assuming exact dynamic programming, it is possible to prove

² Note that Fig. 5 shows a critic state which is slightly different from the one presented in Fig. 6, due to the limitations of function approximators.

that the iterated application of this operator converges towards a unique function Q^π , which corresponds to the state-action value function of π as defined above [9].

Q^π is Piecewise-Constant: In a deterministic environment with sparse terminal rewards, Q^π is piecewise-constant because $V^\pi(s')$ only depends on two things: the (integer) number of steps required to reach a rewarded state from s' , and the value of this reward state, which is itself piecewise-constant. Note that we can reach the same conclusion with non-terminal rewards, by making the stronger hypothesis on the actor that $\forall s, r(s, \pi(s)) = 0$. Notably, this is the case for the actor $\forall s, \pi(s) = 0.1$ on 1D-TOY.

Q is Approximately Piecewise-Constant and $\nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})|_{\mathbf{a}=\pi(\mathbf{s})} \approx \mathbf{0}$: Quite obviously, from Q^π is piecewise-constant and Q tends towards Q^π , we can infer that Q progressively becomes almost piecewise-constant as the cyclic process unfolds. Actually, the Q function is estimated by a function approximator which is never truly discontinuous. The impact of this fact is studied in Sect. 4.5. However, we can expect Q to have mostly flat gradients since it is trained to match a piecewise-constant function. We can thus infer that, globally, $\nabla_a Q(s, a)|_{a=\pi(s)} \approx 0$. And critically, the gradients in the flat regions far from the discontinuities give little information as to how to reach regions of higher values.

π Barely Changes: DDPG uses the deterministic policy gradient update, as seen in Eq. (3). This is an analytical gradient that does not incorporate any stochasticity, because Q is always differentiated exactly at $(s, \pi(s))$. Thus the actor update is stalled, even when the reward is regularly found by the behavior policy. This closes the loop of our process.

4.4 Consequences of the Convergence Cycle

As illustrated with the red arrows in Fig. 7, the more loops performed in the convergence process, the more the critic tends to be piecewise-constant and the less the actor tends to change. Importantly, this cyclic convergence process is triggered as soon as the changes on the policy drastically slow down or stop. What matters for the final performance is the quality of the policy reached before this convergence loop is triggered. Quite obviously, if the loop is triggered before the policy gets consistently rewarded, the final performance is deemed to be poor.

The key of this undesirable convergence cycle lies in the use of the deterministic policy gradient update given in Eq. (3). Actually, rewarded samples found by the exploratory behavior policy β tend to be ignored by the conjunction of two reasons. First, the critic is updated using $Q(s', \pi(s'))$ and not $Q(s, \beta(s))$, thus if π differs too much from β , the values brought by β are not properly propagated. Second, the actor being updated through (3), i.e. using the analytical gradient of the critic with respect to the actions of π , there is no room for considering other actions than that of π . Besides, the actor update involves only the state s of the sample taken from the replay buffer, and not the reward found from this sample $r(s, a)$ or the action performed. For each sample state s , the actor update

is intended to make $\pi(s)$ converge to $\operatorname{argmax}_a \pi(s, a)$ but the experience of different actions performed for identical or similar states is only available through $Q(s, \cdot)$, and in DDPG it is only exploited through the gradient of $Q(s, \cdot)$ at $\pi(s)$, so the process can easily get stuck in a local optimum, especially if the critic tends towards a piecewise-constant function, which as we have shown happens when the reward is sparse. Besides, since TD3 also updates the actor according to (3) and the critic according to (2), it is susceptible to the same failures as DDPG.

4.5 Impact of Function Approximation

We have just explained that when the actor has drifted to an incorrect policy before finding the reward, an undesirable convergence process should result in DDPG getting stuck to this policy. However, in 1D-TOY, we measured that the actor drifts to a policy moving to the right in 50% of cases, but the learning process only fails 1% of times. More generally, despite the issues discussed in this paper, DDPG has been shown to be efficient in many problems. This better-than-predicted success can be attributed to the impact of function approximation.

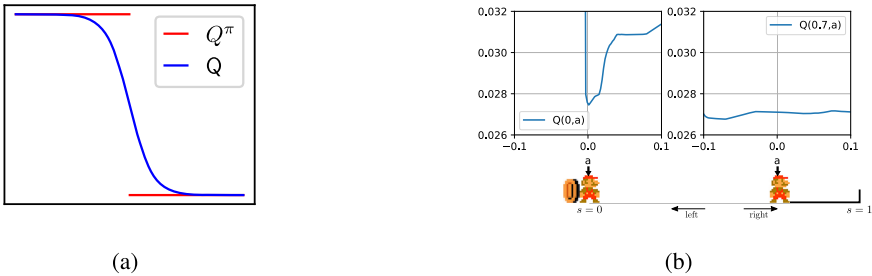


Fig. 8. (a) Example of a monotonous function approximator. (b) Simply changing the vertical scale of the graphs presented in Fig. 5b reveals that the function approximator is not perfectly flat, and has many unwanted local extrema. Specifically, continuously moving from $\pi(0) = 0.1$ to $\pi(0) < 0$ requires crossing a significant valley in $Q(0, a)$, while $\pi(0) = 0.1$ is a strong local maximum.

Figure 8a shows a case in which the critic approximates Q^π while keeping a monotonous slope between the current policy value and the reward. In this case, the actor is correctly updated towards the reward (if it is close enough to the discontinuity). This is the most often observed case, and naturally we expect approximators to smooth out discontinuities in target functions in a monotonous way, which facilitates gradient ascent. However, the critic is updated not only in state-action pairs where $Q^\pi(s, a)$ is positive, but also at points where $Q^\pi(s, a) = 0$, which means that the bottom part of the curve also tends to flatten. As this happens, we can imagine phenomena that are common when trying to approximate

discontinuous functions, such as the overshoot observed in Fig. 8b. In this case, the gradient prevents the actor from improving.

5 Conclusion and Future Work

In RL, continuous action and sparse reward environments are challenging. In these environments, the fact that a good policy cannot be learned if exploration is not efficient enough to find the reward is well-known and trivial. In this paper, we have established the less trivial fact that, if exploration does find the reward consistently but not early enough, an actor-critic algorithm can get stuck into a configuration from which rewarded samples are just ignored. We have formally characterized the reasons for this situation, and we believe our work sheds new light on the convergence regime of actor-critic algorithms.

Our study was mainly built on a simplistic benchmark which made it possible to study the revealed deadlock situation in isolation from other potential failure modes such as exploration issues, the over-estimation bias, extrapolation error or the deadly triad. The impact of this deadlock situation in more complex environments is a pressing question. For this, we need to sort out and quantify the impact of these different failure modes. Using new tools such as the ones provided in [2], recent analyses of the deadly triad such as [1] as well as simple, easily visualized benchmarks and our own tools, for future work we aim to conduct deeper and more exhaustive analysis of all the instability factors of DDPG-like algorithms, with the hope to contribute in fixing them.

References

1. Achiam, J., Knight, E., Abbeel, P.: Towards characterizing divergence in deep q-learning. [arXiv:1903.08894](https://arxiv.org/abs/1903.08894) (2019)
2. Ahmed, Z., Roux, N.L., Norouzi, M., Schuurmans, D.: Understanding the impact of entropy on policy optimization. [arXiv:1811.11214](https://arxiv.org/abs/1811.11214) (2019)
3. Baird, L.C., Klopff, A.H.: Technical Report WL-TR-93-1147. Wright-Patterson Air Force Base, Ohio, Wright Laboratory (1993)
4. Boyan, J.A., Moore, A.W.: Generalization in reinforcement learning: safely approximating the value function. In: Advances in Neural Information Processing Systems, pp. 369–376 (1995)
5. Colas, C., Sigaud, O., Oudeyer, P.Y.: GEP-PG: Decoupling Exploration and Exploitation in Deep Reinforcement Learning Algorithms. [arXiv:1802.05054](https://arxiv.org/abs/1802.05054) (2018)
6. Fortunato, M., et al.: Noisy Networks for Exploration. [arXiv:1706.10295](https://arxiv.org/abs/1706.10295) (2017)
7. Fujimoto, S., van Hoof, H., Meger, D.: Addressing Function Approximation Error in Actor-Critic Methods. ICML (2018)
8. Fujimoto, S., Meger, D., Precup, D.: Off-Policy Deep Reinforcement Learning without Exploration. [arXiv:1812.02900](https://arxiv.org/abs/1812.02900) (2018)
9. Geist, M., Pietquin, O.: Parametric value function approximation: a unified view. In: ADPRL 2011, Paris, France, pp. 9–16 (2011)
10. van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., Modayil, J.: Deep Reinforcement Learning and the Deadly Triad. [arXiv:1812.02648](https://arxiv.org/abs/1812.02648) (2018)

11. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. [arXiv:1509.02971](https://arxiv.org/abs/1509.02971) (2015)
12. Mnih, V., et al.: Playing Atari with Deep Reinforcement Learning. [arXiv:1312.5602](https://arxiv.org/abs/1312.5602) (2013)
13. Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
14. Plappert, M., et al.: Parameter space noise for exploration. arXiv preprint [arXiv:1706.01905](https://arxiv.org/abs/1706.01905) (2017)
15. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: International Conference on Machine Learning, pp. 387–395 (2014)
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
17. Tsitsiklis, J.N., Van Roy, B.: Analysis of temporal-difference learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1075–1081 (1997)