DYNAMICALLY BALANCED VALUE ESTIMATES FOR ACTOR-CRITIC METHODS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning in an actor-critic setting relies on accurate value estimates of the critic. However, the combination of function approximation, temporal difference (TD) learning and off-policy training can lead to an overestimating value function. A solution is to use Clipped Double Q-learning (CDQ), which is used in the TD3 algorithm and computes the minimum of two critics in the TD-target. We show that CDQ induces an underestimation bias and propose a new algorithm that accounts for this by using a weighted average of the target from CDQ and the target coming from a single critic. The weighting parameter is adjusted during training such that the value estimates match the actual discounted return on the most recent episodes and by that it balances over- and underestimation. Empirically, we obtain more accurate value estimates and demonstrate state of the art results on several OpenAI gym tasks.

1 INTRODUCTION

In recent years it was shown that reinforcement learning algorithms are capable of solving very complex tasks, surpassing human expert performance in games like Go (Silver et al., 2016), Starcraft (DeepMind) or Dota (OpenAI). However, usually a large amount of training time is needed to achieve these results (e.g. 45,000 years of gameplay for Dota). For many important problems (e.g. in robotics) it is prohibitively expensive for the reinforcement learning agent to interact with its environment that much. This makes it difficult to apply such algorithms in the real world.

Off-policy reinforcement learning holds the promise of being more data-efficient than on-policy methods as old experience can be reused several times for training. Unfortunately, the combination of temporal-difference (TD) learning, function approximation and off-policy training can be unstable, which is why it has been called the deadly triad (Sutton & Barto, 2018; van Hasselt et al., 2018). If the action space is discrete, solutions like Double DQN (Van Hasselt et al., 2016) are very effective at preventing divergence of the value estimates by eliminating an otherwise prevailing overestimation bias. For continuous action spaces, which characterize many tasks, it was shown that Double DQN can not solve the overestimation problem Fujimoto et al. (2018). In an actor-critic setting it is important that the value estimates of the critic are accurate in order for the actor to learn a policy from the critic. The TD3 Fujimoto et al. (2018) algorithm uses Clipped Double Q-learning (CDQ) to produce a critic without an overestimation bias, which greatly improved the performance of the algorithm. In CDQ two critics are trained at the same time and the TD target for both of them is the minimum over the two single TD targets.

While the authors note that the CDQ critic update tends to underestimate the true values, this is not further examined. We show that this underestimation bias occurs in practice and propose a method that accounts for over- and underestimation of the critic at the same time. Similarly to CDQ we train two function approximators for the Q-values, but we regress them not on the same quantity. The TD target for each of the two critics is a weighted average of the single TD target for that critic and the TD target from CDQ. The weighting parameter is learned by comparing the value estimates for the most recent state-action pairs with the observed discounted returns for these pairs. As the one term of the average has an underestimation bias while the other one has an overestimation bias, the weighted average balances these biases and we show empirically that this method obtains much more accurate estimates of the Q-values.

We verify that the more accurate critics improve the performance of the reinforcement learning agent as our method achieves state of the art results on a range of continuous control tasks from OpenAi gym Brockman et al. (2016). To guarantee reproducibility we open source our code which is easy to execute and evaluate our algorithm on a large number of different random seeds.

2 Related Work

The Deterministic Policy Gradient algorithm (DPG) Silver et al. (2014) learns a deterministic policy in an actor-critic setting. This work was extended to the Deep Deterministic Policy Gradient algorithm Lillicrap et al. (2015) by using multi-layer neural networks as function approximators. The Twin Delayed Deep Deterministic policy gradient algorithm (TD3) Fujimoto et al. (2018) adds three more components to DDPG and achieves state of the art results. First, the actor is updated less frequently than the critic, to allow for more accurate critic estimates before they are used for the actor. Second, in the critic update noise is added to the actions proposed by the actor. While these two extensions are introduced to decrease the variance in the policy gradient, the third one, Clipped Double Q-learning, aims at preventing an overestimation bias.

The use of two Q-estimators was first proposed in the Double Q-learning algorithm Hasselt (2010). The two estimates are combined in the TD target such that determining the maximizing action is decoupled from computing the value for that action. Later it was proposed to use the target network (whose parameters are periodically set to the current parameters or are an exponentially weighted moving average of them) for one of the two value estimates Van Hasselt et al. (2016). This eliminates the need to train two networks. While this works well for discrete actions, versions of double Q-learning adapted to the actor-critic setting were shown to still suffer from an overestimation bias Fujimoto et al. (2018). Other approaches that aim at preventing overestimation bias in Q-learning have averaged the Q-estimates obtained from snapshots of the parameters from different training steps Anschel et al. (2017) or used a bias correction term Lee et al. (2013).

Balancing between over- and underestimating terms in the Q-estimates has been done for a discrete action space Zhang et al. (2017). The work investigates multi-armed bandit problems and an underestimation bias is reported for Double Q-learning, while Q-learning with a single estimator is reported to overestimate the true values. Similarly to our approach a weighting is introduced in the TD target. Different to us, the weighting parameter is not learned by taking actual samples for the value estimator into account, but the parameter is set individually for each state-action pair used to train the Q-network according to a function that computes the minimum and maximum Q-value over all actions for the given state. Finding these optimal actions for every transition on which the Q-networks are trained becomes infeasible for continuous action spaces.

Divergence of Q-values has been investigated in several recent works van Hasselt et al. (2018) Achiam et al. (2019) Fu et al. (2019). Of them only in Achiam et al. (2019) the case of a continuous action space is considered. In their analysis it is investigated under which conditions a certain approximation of the Q-value updates is a contraction in the sup norm. From that an algorithm is derived that does not need multiple critics or target networks. The downside is that it is very compute intensive.

3 Preliminaries

We consider model-free reinforcement learning for episodic tasks with continuous action spaces. An agent interacts with its environment by selecting an action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ for every discrete time step t. The agent receives a scalar reward r_t and observes the new state s_{t+1}. The goal is to learn a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that selects the agents actions in order to maximize the sum of future discounted rewards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, where $\gamma \in [0, 1]$ is the discount factor.

For a given state-action pair (s, a) the value function is defined as $Q^\pi(s, a) := \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}[R_t | s, a]$, which is the expected return when executing action a in state s and following π afterwards. We write π_ϕ for the policy with parameters ϕ, that we learn in order to maximize the expected return $J(\phi) = \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}[R_0]$. The parameters can be optimized with the gradient of J w.r.t. the policy parameters ϕ. The deterministic policy gradient Silver et al. (2014) is given by

$$\nabla_\phi J(\phi) = \mathbb{E}_{s_i \sim p_\pi} \left[\nabla_a Q^\pi(s, a)|_{a=\pi(s)} \nabla_\phi \pi_\phi(s) \right]. \tag{1}$$

In practice the value function Q^π is not given and has to be approximated. This setting is called actor-critic, the policy is the actor and the learned value function has the role of a critic.

The Q-learning algorithm Watkins (1989) tries to learn the value function with TD learning Sutton (1988), which is an incremental update rule and aims at satisfying the Bellman equation Bellman (1957). Deep Q-learning Mnih et al. (2015) is a variant of this algorithm and can be used to learn the parameters θ of an artificial neural network $Q_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that approximates the value function. The network is updated by regressing its value at (s_t, a_t) to its 1-step TD targets

$$y = r + \gamma Q_{\bar{\theta}} \left(s_{t+1}, \pi_{\bar{\phi}}(s_{t+1}) \right), \tag{2}$$

where $Q_{\bar{\theta}}$, $\pi_{\bar{\phi}}$ are the target networks of Q_ϕ, π_θ and the corresponding parameters $\bar{\phi}$ and $\bar{\theta}$ are updated according to a exponential moving average: $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}$, similarly for ϕ.

4 Over- and Underestimation Bias

If a learned critic Q_θ is used in the deterministic policy gradient (eq. 1), the actor π_ϕ is updated through Q_θ, which in turn is learned with the rewards obtained from the environment. This means that the actor requires a good critic to be able to learn a well performing policy. Recently, it was shown, that using Q-learning in such an actor-critic setting can lead to an overestimation of the Q-values Fujimoto et al. (2018). This is problematic if the overestimation in the critic occurs for actions that lead to low returns. To avoid this, Clipped Double Q-learning was proposed Fujimoto et al. (2018). In this approach two Q-networks $(Q_{\theta_1}, Q_{\theta_2})$ are learned in parallel. They are trained on the same TD target, which is defined via the minimum of the two Q-networks

$$y = r + \gamma \min_{i=1,2} Q_{\bar{\theta}_i} \left(s_{t+1}, \pi_{\bar{\phi}}(s_{t+1}) \right). \tag{3}$$

The authors note that this can lead to an underestimation bias for the Q-values, but argue that this is not as bad as overestimating.

A big advantage of CDQ is that the Q-estimates do not explode, which otherwise can sometimes happen and is usually followed by a breakdown in performance. Apart from that, an over- or underestimation bias would not be problematic if all values are biased by the same constant value. It becomes a problem if the bias of the value estimates for different state-action pairs differs. Then the critic might reinforces the wrong actions. If this happens and in a given state an action is erroneously given a high value by the critic, the actor is reinforced to choose the corresponding action. This increases the probability that the agent selects that action the next time when it is in that (or a similar) state. The agent will receive a low reward, which leads to a decrease of performance. But the critic can correct itself on the new experience which will eventually be propagated through to the actor. If on the other hand, the critic underestimates the Q-value of a good action, the actor is trained to never try this action. In this case the critic might never be corrected as experience opposing the critics believe is never encountered. While this is a simplistic picture of the ongoing learning dynamics, it can give a good intuition, why both cases should be prevented if possible.

It is obvious that taking the minimum over two estimates can lead to an underestimation bias. To check if this also occurs in practice, we conducted an experiment, where we examined the Q-value estimates of different agents. We trained an TD3 agent that uses CDQ as defined in eq. 3 and one TD3 agent that uses instead of CDQ the critic updates of DDPG Lillicrap et al. (2015) as defined in eq. 2. We trained on three different environments from OpenAi gym Brockman et al. (2016). Periodically, we sampled 1000 state-action pairs from the replay-buffer and computed the value estimate of the critic. We approximated the true values for each state-action pair by rolling out the current policy 50 times from that pair onwards and averaged the observed discounted return. The results for the average value of each time step are shown in the first row of Figure 1. Similarly to previous work Fujimoto et al. (2018), we observe that the DDPG-style updates of the Q-network lead to an overestimation bias. For CDQ we indeed observe an underestimation bias as the value estimates are significantly lower than the true values.

5 Algorithm

We propose Balanced Clipped Double Q-learning (BCDQ), a new algorithm to learn the critic with the goal of reducing the bias. We adopt the idea of two Q-networks, but train them on different TD

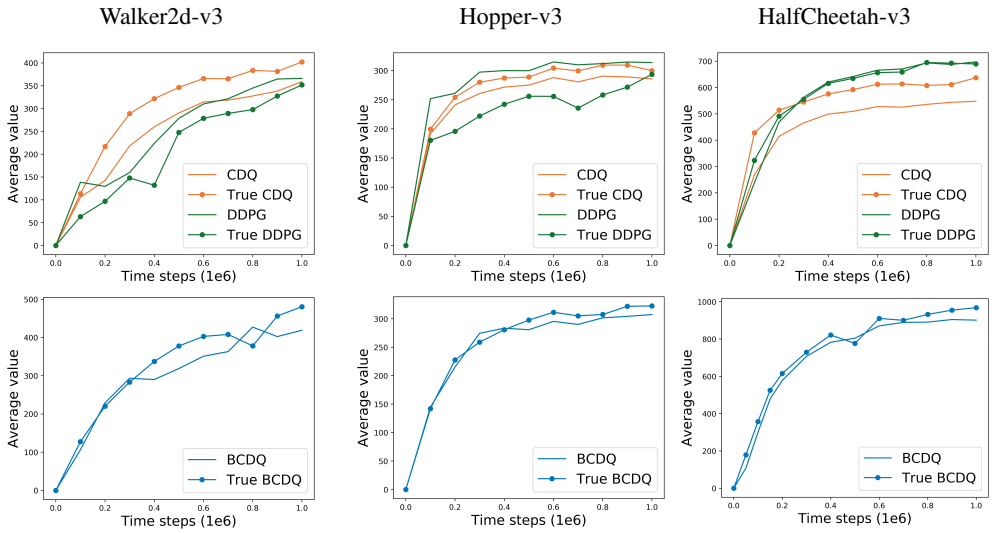

Figure 1: Measuring estimation bias in the Q-value estimates of DDPG, CDQ and Balanced Clipped Double Q-learning (BCDQ) on three different OpenAI gym environments. The first row shows the estimates of DDPG and CDQ and it can be seen that DDPG leads to an overestimation bias, while CDQ leads to an underestimation bias. In the second row the value estimates of BCDQ are shown. It can be observe that the BCDQ estimates are more accurate and do not exhibit a clear bias in any direction.

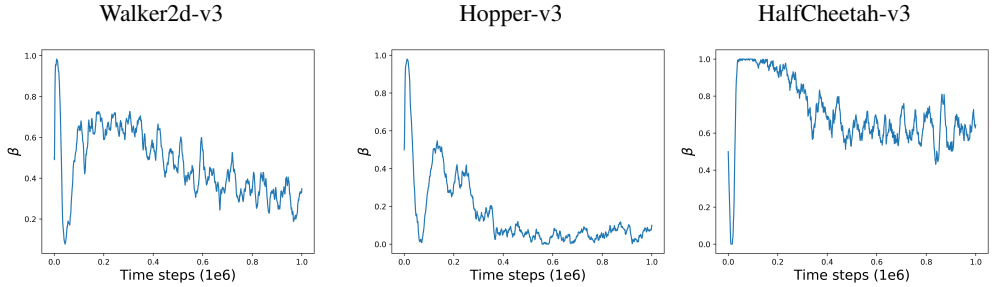

Figure 2: The plots show the average over 10 runs of the weighting parameter β for three OpenAI gym environments.

targets. The TD target y_k for the k-th Q-network $Q_{\theta_k}, k \in \{1, 2\}$ is defined as a weighted average of the network itself and the minimum of both networks

$$y_k = r + \gamma \left(\beta \, Q_{\bar{\theta}_k}(s_{t+1}, \pi_{\bar{\phi}}(s_{t+1})) + (1 - \beta) \min_{i=1,2} Q_{\bar{\theta}_i}(s_{t+1}, \pi_{\bar{\phi}}(s_{t+1})) \right), \tag{4}$$

where $\beta \in [0, 1]$. The first term corresponds to the TD target according to DDPG and second term corresponds to the TD target of CDQ. While the first term tends to overestimate, the second term tends to underestimate the true Q-values. Correctly weighting between them can correct for this bias.

However, setting β manually is difficult. The perfect β that maximally reduces bias may change from environment to environment and also over the time of the training process. Consequently, we adjust β over the course of the training. As the goal is to minimize bias and since β controls in which direction more bias is introduced, we use samples of the Q-values to learn β. After every episode we compute for every seen state-action pair (s_t, a_t) the actual discounted future return from that pair onwards $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, which is a sample for the quantity the Q-networks $Q_{\theta_k}(s_t, a_t)$ try to estimate. If the Q-estimates are higher than R_t, they overestimated and β should be decreased to give the "min" term in eq. 4 more weight. If on the other hand the Q-estimates are lower, we observe the case of underestimation and β should be increased. This behaviour can be achieved by

Algorithm 1 BTD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$, $\bar{\phi} \leftarrow \phi$, set $k = 0$ and $\beta \in [0, 1]$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** *total timesteps* **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r, new state s' and binary value d indicating if the episode ended
  Store transition tuple (s, a, r, s', d) in $\mathcal{B}$ and set $k \leftarrow k + 1$
  if $k \geq$ *beta update rate* and $d = 1$ **then**
    // Update β
    Get recent tuples $(s_i, a_i, r_i, s'_i, d_i)_{i=t-k}^{t}$ from $\mathcal{B}$
    Compute discounted return $R_t = \sum_{i=t}^{d_i=1} \gamma^{i-t} r_i$ and store it in batch $(s_i, a_i, R_i)_{i=t-k}^{t}$
    $\nabla_\beta = \frac{1}{k} \sum_{i=t-k}^{t} (Q^\pi(s_i, a_i) - R_i)$
    $\beta \leftarrow \text{clip}\left(\beta - \alpha \nabla_\beta, 0, 1\right)$
    $k = 0$
  end if

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\bar{\phi}}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y_{\min} \leftarrow r + \gamma \min_{i=1,2} Q_{\bar{\theta}_i}(s', \tilde{a})$
  $y_i \leftarrow \beta\big(r + \gamma Q_{\bar{\theta}_i}(s', \tilde{a})\big) + (1 - \beta) y_{\min}, \quad i = 1, 2$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} \frac{1}{N} \sum (y_i - Q_{\theta_i}(s, a))^2$
  if t mod *actor delay rate* $= 0$ **then**
    // Update ϕ by the deterministic policy gradient:
    $Q(s, a) \leftarrow \frac{1}{2}(Q_{\theta_1}(s, a) + Q_{\theta_2}(s, a))$
    $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_a Q(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    // Update target networks:
    $\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau)\bar{\theta}_i, \quad \bar{\phi} \leftarrow \tau \phi + (1 - \tau)\bar{\phi}$
  end if
end for

minimizing the following objective w.r.t. β:

$$\beta \left(\sum_{j=1}^{E} T_j \right)^{-1} \sum_{j=1}^{E} \sum_{t=0}^{T_j - 1} \left(Q^\pi \left(s_t^{(j)}, a_t^{(j)} \right) - R_t^{(j)} \right), \tag{5}$$

where we restrict β to be in the interval $[0, 1]$, E is the number of episodes we optimize over, $s_t^{(j)}$ is the t-th state in the j-th considered episode (similarly for the actions $a_t^{(j)}$), T_j is the number of time steps in episode j and $R_t^{(j)}$ are the future discounted returns. The parameter β is updated every time the sum over all time steps in the episodes since the last update, $\sum_{j=1}^{E} T_j$, exceeds a fixed threshold. We set this threshold to be the maximum number of episode steps that are possible. To optimize β we use stochastic gradient descent. We note that learning β increases the computational complexity only minimal, as it is just one parameter that has to be optimized. To evaluate the objective in eq. 5, a further forward path through the Q-network is performed, but no backward path is needed in the training.

We evaluated the accuracy of the value estimates of BCDQ and report the results in the second row of Figure 1. It can be seen, that compared to the other methods BCDQ approximates the true Q-values much better. This indicates that the weighting parameter β can indeed be adjusted over the course of the training such that the two opposing biases cancel each other out.

The behaviour of β is visualized in Figure 2 as an average over 10 runs per environment. For the Hopper task it can be observed that after some time the parameter β gets very close to zero, which corresponds to using only CDQ to update the critic. For HalfCheetah, the CDQ term is not weighted very high. This is explainable as in Figure 1 it can be seen that CDQ induces a large bias on this

Table 1: Average over the Max episode reward of 10 trials of 1 million time steps (2 million steps for Humanoid-v3). Maximum value for each task is bolded. $\pm$ corresponds to a single standard deviation over trials.

Environment	BTD3	TD3	DDPG	SAC
HalfCheetah-v3	**11487 $\pm$ 614**	10009 $\pm$ 1849	7840 $\pm$ 574	9508 $\pm$ 1068
Ant-v3	**5437 $\pm$ 610**	4955 $\pm$ 971	994 $\pm$ 3	4793 $\pm$ 722
Walker2d-v3	**4939 $\pm$ 559**	4668 $\pm$ 487	3491 $\pm$ 926	4410 $\pm$ 364
Hopper-v3	**3637 $\pm$ 114**	3615 $\pm$ 95	2970 $\pm$ 10701	3288 $\pm$ 258
Humanoid-v3	**5651 $\pm$ 88**	5228 $\pm$ 42	1898 $\pm$ 1810	5555 $\pm$ 122
Reacher-v2	**-2.66 $\pm$ 0.17**	-2.68 $\pm$ 0.16	-6.75 $\pm$ 0.48	-3.61 $\pm$ 0.23

task. Adjusting β over time allows to put more weight on the term that currently gives a more accurate estimate. This prevents the accumulation of errors introduced by bootstrapping in the TD target. From the plots it can also be seen that treating β as an hyperparameter might be difficult, as it would have to be tuned for every environment. Furthermore, leaving β fixed could not account for the changing learning dynamics over the course of training. In Figure 2 it can be seen that different drifts exist in each environment. For example in Walker2d the learned β decreases on average after an initial stabilization period.

Since the two Q-networks are not trained on the same target as it is the case for CDQ, the difference between the predictions of the two Q-networks will be higher. This suggest that - similarly to ensemble methods - the average of the two predictions might be an even better estimator. Following that rationale, in our algorithm the critic that teaches the actor is the average of the predictions of the two Q-networks.

As a result of the above discussion, we propose the Balanced Twin Delayed Deep Deterministic policy gradient algorithm (BTD3), which builds on TD3 Fujimoto et al. (2018). Differently to TD3, our algorithm uses BTDQ instead of CDQ to update the critics. For the learning of the actor the predictions of the two critics are averaged instead of using only the first critic. The BTD3 algorithm is shown in Algorithm 1.

6 Experiments

We evaluate our algorithm on a range of challenging continuous control tasks from OpenAI Gym Brockman et al. (2016), which makes use of the physics engine MuJoCo Todorov et al. (2012) (version 2.0). To guarantee an apples-to-apples comparison with TD3, we extended the original source code of TD3 with our method and evaluate our algorithm with the default hyperparameters of TD3 for all tasks except for Humanoid-v3. We observed that TD3 does not learn a successful policy on Humanoid-v3 with the default learning rate of 0.001, but we found that TD3 does learn if the learning rate for both actor and critic is reduced. Consequently, we set it to 0.0001 for this task and did the same for BTD3. We set the learning rate for the weighting parameter β to 0.05 and initialize $\beta = 0.5$ at the beginning of the training for all environments.

As is done in TD3, we reduce the dependency on the initial parameters by using a random policy for the first 10000 steps for HalfCheetah-v3, Ant-v3, Humanoid-v3 and the first 1000 steps for the remaining environments. After that period we add Gaussian noise $\mathcal{N}(0, 0.1)$ to each action in order to ensure enough exploration.

During training the policy is evaluated every 5000 environment steps by taking the average over the episode reward obtained by rolling out the current policy without exploration noise 10 times. For each task and algorithm we average the results of 10 trials each with a different random seed, except for Humanoid-v3, where we used 5 trials.

We compare our algorithm to the state of the art continuous control methods SAC Haarnoja et al. (2018a) (with learned temperature parameter Haarnoja et al. (2018b)), TD3 Fujimoto et al. (2018) and to DDPG Lillicrap et al. (2015). For both, SAC and TD3, we used the source code published by

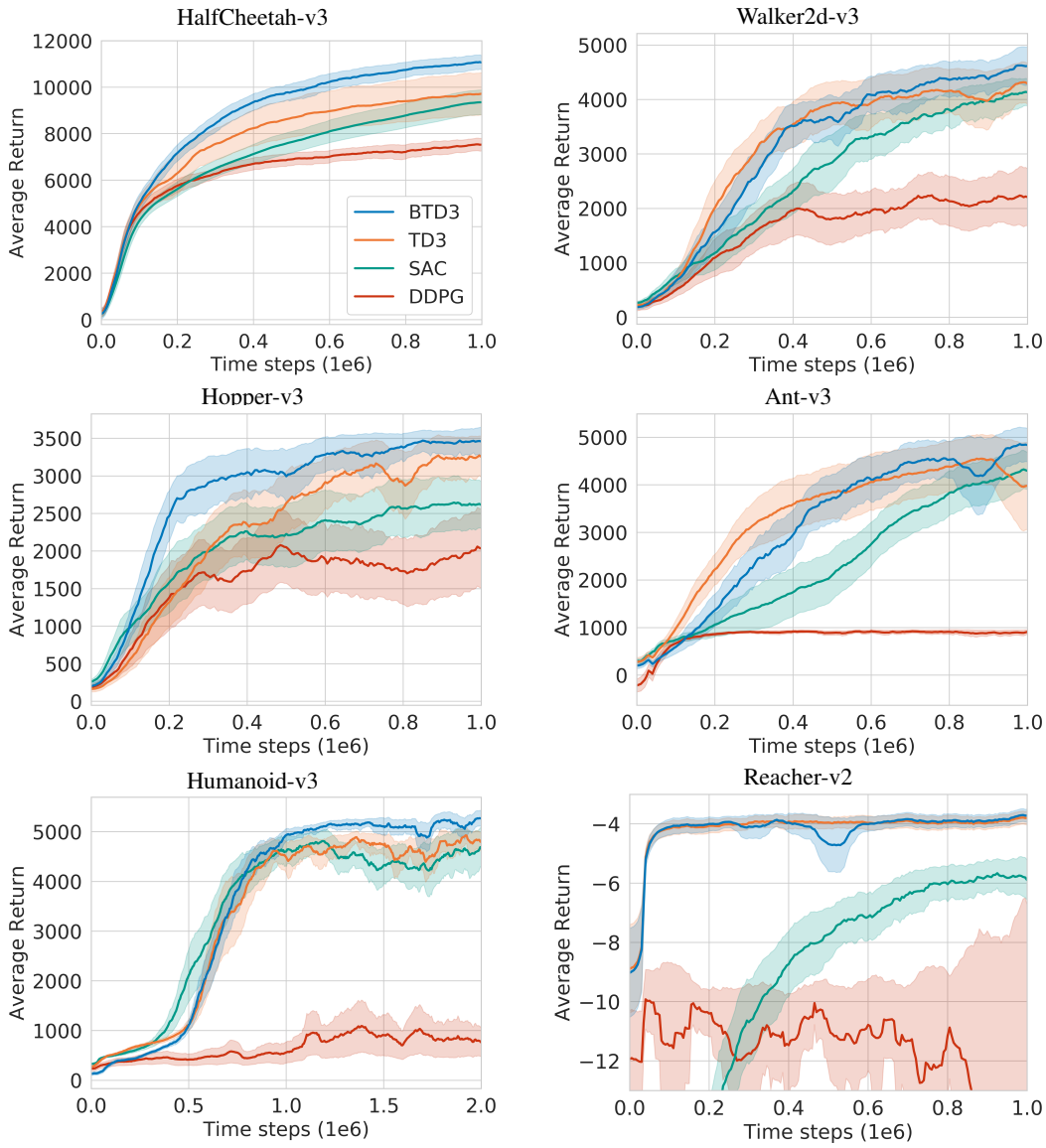

Figure 3: Learning curves for six different continuous control tasks from OpenAi gym. The shaded area represents half a standard deviation over the 10 trials (5 for Humanoid-v3). For readability the curves showing the mean are filtered with a uniform filter of size 15.

the respective authors. Code to reproduce the results of BTD3 can be found at `https://gofile.io/?c=AQFK3j`.

The learning curves are shown in Figure 3. For all tasks BTD3 matches or outperforms TD3. Furthermore, it performs significantly better than SAC and DDPG.

In Table 1 the results are presented in terms of the average maximum episode reward. In order to compute that statistic, for each trial we computed the maximum over the evaluations that were executed all 5000 time steps, where the evaluations are itself the average over 10 rollouts of the current policy. Afterwards, we computed the average of this value over the different trials. The results show that the best policies of BTD3 achieve significantly higher episode rewards than the best policies of the other methods.

To further understand the influence of the dynamic weighting scheme we trained BTD3 with a fixed value for β. We evaluated for the values $\beta \in \{0.00, 0.25, 0.50, 0.75, 1.00\}$, where $\beta = 0.00$

corresponds to TD3 and $\beta = 1.00$ corresponds to DDPG. The averaged results over 10 runs are shown in Figure 4. From the plots we can make two observations. First, it is essential that β is adjusted during the training. For any of the considered values of β leaving it fixed leads to a worse performance compared to BTD3 and in most cases also worse than TD3. In Figure 2 it was shown that the adjusted weighting parameter is on average over many runs attracted to different values depending not only on the environment but also on the timestep during training. The dynamic adjustment to prevent accumulating errors is not possible when β is fixed. Second, it is surprising that fixed values for β that would seem promising to try from Figure 2 can perform worse than other fixed values. For example inspecting the plots in Figure 2 the value $\beta = 0.75$ seems a good fit for the HalfCheetah environment. But the evaluation shows that $\beta = 0.25$ and $\beta = 0.50$ perform better. This further supports the hypothesis that the most important part about BCDQ is the dynamic adjustment of the weighting parameter.

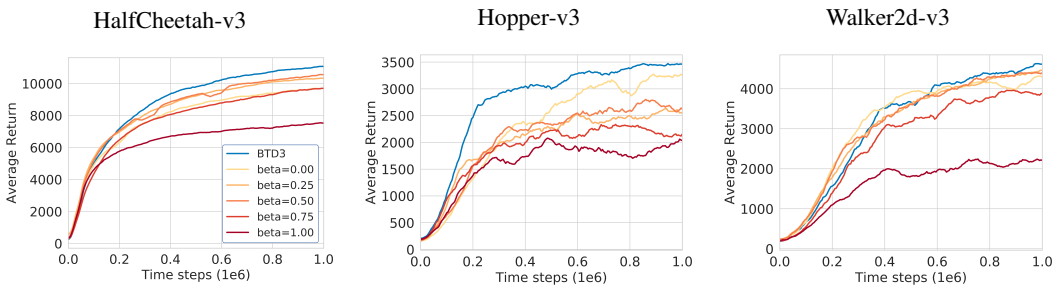

Figure 4: Learning curves for four different continuous control tasks from OpenAi gym over 10 random seeds each. The show algorithms are BTD3 and versions of it with a fixed value of β. For each algorithm the curves show the mean over 10 runs with different random seeds and are filtered with a uniform filter of size 15.

7 Conclusion

We showed that Clipped Double Q-learning (CDQ) induces an underestimation bias in the critic, while an overestimation bias occurs if just one Q-network is used. From that we derived the Balanced Clipped Double Q-learning algorithm (BCDQ) that updates the critic through a weighted average of the two mentioned update mechanisms. The weighting parameter is adjusted over the course of training by comparing the Q-values of recently visited state-action pairs with the actual discounted return observed from that pair onwards.

It was shown that BCDQ achieves much more accurate value estimates by adjusting the weighting parameter. Replacing CDQ with BCDQ leads to the Balanced Twin Delayed Deep Deterministic policy gradient algorithm (BTD3). Our method achieves state of the art performance on a range of continuous control tasks. Furthermore, BCDQ can be added to any other actor-critic algorithm while it only minimally increases the computational complexity compared to CDQ. It is also be possible to use BCDQ for discrete action spaces. Evaluating that approach is an interesting area for future research.

References

Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards characterizing divergence in deep q-learning. *CoRR*, abs/1903.08894, 2019. URL `http://arxiv.org/abs/1903.08894`.

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 176–185, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

DeepMind. Alphastar: Mastering the real-time strategy game starcraft ii. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/. Accessed: 2019-09-23.

Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep q-learning algorithms. *CoRR*, abs/1902.10250, 2019. URL http://arxiv.org/abs/1902.10250.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1582–1591, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018b. URL http://arxiv.org/abs/1812.05905.

Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pp. 2613–2621, 2010.

Donghun Lee, Boris Defourny, and Warren B Powell. Bias-corrected q-learning to control max-operator bias in q-learning. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 93–99. IEEE, 2013.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

OpenAI. Openai five description. https://openai.com/blog/how-to-train-your-openai-five/. Accessed: 2019-09-23.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 387–395, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, Aug 1988.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2018. ISBN 78-0262039246.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033. IEEE, 2012.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *CoRR*, abs/1812.02648, 2018.

C.J.C.H. Watkins. Learning from delayed rewards. *PhD thesis, Cambridge University*, 1989.

Zongzhang Zhang, Zhiyuan Pan, and Mykel J. Kochenderfer. Weighted double q-learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pp. 3455–3461. AAAI Press, 2017. ISBN 978-0-9992411-0-3. URL http://dl.acm.org/citation.cfm?id=3172077.3172372.