Reward-Constrained Behavior Cloning

Zhaorong Wang[1] , **Meng Wang**[1] , **Jingqi Zhang**[2] , **Yingfeng Chen**[1*] and **Chongjie Zhang**[3]
[1]NetEase Fuxi AI Lab, Hangzhou, China
[2]School of Computer Science and Technology, Xi'an Jiaotong University
[3]MMW, Tsinghua University
{wangzhaorong, wangmeng02, chenyingfeng1}@corp.netease.com, zkkomodao@gmail.com, chongjie@tsinghua.edu.cn

Abstract

Deep reinforcement learning (RL) has demonstrated success in challenging decision-making/control tasks. However, RL methods, which solve tasks through maximizing the expected reward, may generate undesirable behaviors due to inferior local convergence or incompetent reward design. These undesirable behaviors of agents may not reduce the total reward but destroy the user experience of the application. For example, in the autonomous driving task, the policy actuated by speed reward behaves much more sudden brakes while human drivers generally don't do that. To overcome this problem, we present a novel method named Reward-Constrained Behavior Cloning (RCBC) which synthesizes imitation learning and constrained reinforcement learning. RCBC leverages human demonstrations to induce desirable or human-like behaviors and employs lower-bound reward constraints for policy optimization to maximize the expected reward. Empirical results on popular benchmark environments show that RCBC learns significantly more human-desired policies with performance guarantees which meet the lower-bound reward constraints while performing better than or as well as baseline methods in terms of reward maximization.

1 Introduction

Reinforcement Learning (RL) is successful in a range of challenging domains, especially after introducing deep learning techniques. Various deep reinforcement learning algorithms have been proposed and solved many difficult tasks including Atari games [Van Hasselt *et al.*, 2016], robot locomotion tasks [Schulman *et al.*, 2017], and the game of Go [Silver *et al.*, 2016].

However, RL methods are usually designed to solve a task through maximizing the accumulated task reward, which may not always generate desirable behaviors in the real world due to the inferior local convergence or incompetent reward design [Bohez *et al.*, 2019]. For example, in autonomous driving, the reward function is usually designed to inspire the agent to run as fast and far as possible, while some undesirable behaviors that erode the driving experience of a human could be generated under such reward settings, such as slamming on the brake and accelerator or trembling around the center of the road (as illustrated by Figure 1). In contrast, human drivers prefer a more stable driving process with a proper speed instead of an intense driving process filled with sudden brakes and sharp steerings, even though the average speed is high. Another similar problem is called bang-bang control in locomotion tasks that an agent usually switches between extreme values for the controls at a high-frequency [Bohez *et al.*, 2019]. Although these agents can achieve a high accumulated task reward and can be acceptable in simulation, they are usually not suitable for real-world applications. In fact, after the cumulative reward exceeds a certain threshold (e.g., a proper speed in autonomous driving), humans are usually more inclined to teach the agent to behave in a desirable way, rather than increase the reward endlessly.

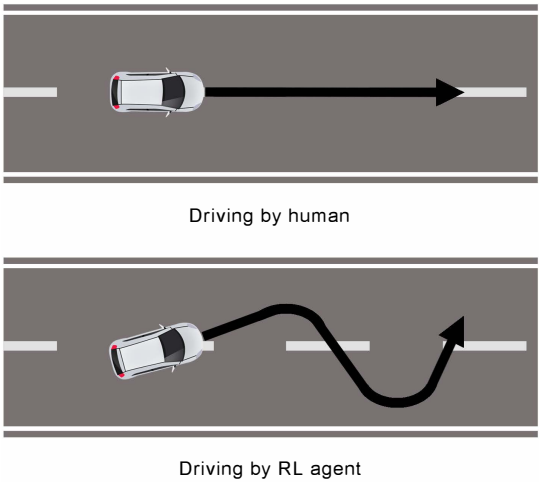


Figure 1: Driving behaviors of human vs RL agent

To induce human-desired behaviors, a widely used strategy is to add penalties for undesired behaviors in the reward

*This work is supported by the NetEase Fuxi AI Lab, Hangzhou, China. Yingfeng Chen is the corresponding author. This work was partially done while Jingqi Zhang was interning at Fuxi AI Lab.

function [Ng *et al.*, 1999]. As a result, the reward function is composed of positive rewards for achieving the goal and negative rewards or costs for executing undesirable actions. However, in many complicated tasks, manually designing appropriate cost functions can be very challenging, especially when there are conflicts between different costs or between the costs and the final goal. In addition, some works propose the human-in-the-loop reinforcement learning frameworks([Knox and Stone, 2009]) related to the idea of curriculum learning to get human-desired behaviors that the agent receives feedback signals from a human during the training process. While the setting of human-in-the-loop requires the agent can interact with a human teacher during training which is always expensive. Another solution is to use Imitation Learning (IL) algorithms, with which an agent learns to solve problems by mimicking expert demonstrations instead of maximizing the accumulated reward. By using this approach, human-like behaviors are expected to be learned from human demonstrations. However, in most cases, it requires expert demonstrations to be nearly optimal, which is difficult or even infeasible to collect in the real world, especially in complicated problems that require a mass of demonstrations.

To overcome these challenges we present a novel algorithm named Reward-Constrained Behavior Cloning (RCBC), which synthesizes imitation learning and constrained reinforcement learning and aims to learn to generate human-desired behaviors while maximizing the accumulated reward. As illustrated by the autonomous driving example, in many real-world tasks, humans' preference for a high cumulative reward usually decreases after exceeding a certain threshold, and then the preference for desirable behaviors increases. Inspired by this observation, RCBC formulates imitation learning as a constrained optimization problem and introduces a reward threshold as lower-bound constraints, which ensures an agent learns desirable behaviors from human demonstrations with guaranteed performance. It is worth noting that RCBC has three important features that make it suitable for a variety of tasks. Firstly, as the lower-bounded reward constraint is guaranteed, RCBC does not require optimal demonstrations from the human. Instead, it only assumes that demonstrations contain some desirable behavior patterns so that an agent can learn such implicit patterns. Because essentially what we expected is to transfer the implicit behavior patterns in non-optimal demonstrations to RL policy with excellent performance. Secondly, RCBC provides the reward threshold as a hyper-parameter and allows users to adjust it accordingly in different situations with varied desirable behaviors (e.g., more human-like behaviors in daily life driving and higher speed only in car racing). More importantly, given such a performance threshold, RCBC can automatically balance the trade-off between reward maximization and human-desirable behavior generation. This is also an advantage of RCBC which enable easier reward design that just needs to focus on the task performance. An important point to note here is that we consider there are no longer any bad policies in the space which meet the reward constraint. More specifically, once the lower-bounded reward constraint is satisfied, we believe that the performance has been guaranteed and all policies beyond this lower-bounded reward are acceptable.

The main contributions of this paper are summarized as follows: (i) we propose a novel constrained behavior cloning optimization approach that learns with guaranteed performance to generate desired behavior patterns implicitly contained in demonstrations; (ii) using the Lagrangian Relaxation technique, the optimal trade-off between the reward maximization and the behavior constraints is obtained automatically; (iii) finally, our approach can be generalized to any Actor-Critic RL algorithm with non-optimal demonstrations, and even can be used to correct the defect of an existing model. We empirically demonstrate the effectiveness of our approach on a Grid-World environment, continuous control tasks from MuJoCo [Todorov *et al.*, 2012], and a more complex racing environment named TORCS [Wymann *et al.*, 2000].

2 Related Work

2.1 Learning from Demonstration

A popular way to learn from demonstration is imitation learning, in which the agent is trained to mimic the expert demonstrations instead of maximizing the accumulated reward from the environment. Behavioral cloning (BC) as a representative imitation learning method seeks the best policy that can minimize the action prediction error in demonstrations ([Bojarski *et al.*, 2016]). However, BC tends to have poor generalization due to the well-known distribution shift problem ([Ross and Bagnell, 2010]) and always presume near-optimal demonstrations. Inverse Reinforcement Learning (IRL, [Ziebart *et al.*, 2008]) is another popular paradigm that learns a reward model to explain the demonstrations as optimal behavior.

In addition, a fairly of approaches referred to as Reinforcement Learning from Demonstration (RLfD) have tried to combine RL with demonstrations to improve the exploration efficiency and speed up the training process. Early RLfD methods ([Brys *et al.*, 2015; Kang *et al.*, 2018]) work comparably well when the expert is optimal, while the perfect demonstration is unrealistic to meet in practice. Some recent RLfD works allow learning from imperfect demonstration. NAC ([Gao *et al.*, 2018]) uses a unified loss function to process both off-line demonstration data and online experience without assumption on the optimality of the data required. [Jing *et al.*, 2020] proposed to learn from imperfect demonstrations by applying expert guidance in a soft way. The essential goal of these methods is still to find the optimal policy which maximizes the cumulative reward with the help of demonstration and they focus on emphasizing the robustness and effectiveness of their methods on imperfect demonstrations. While our method aims to learn desirable behavior patterns from imperfect demonstration with guaranteed performance.

2.2 Constrained Reinforcement Learning

Another area related to our work is constrained reinforcement learning (Constrained RL) that the agent receives additional constraints cost signal and has to keep the expected sum of the constraint costs below a given threshold [Achiam *et al.*, 2017]. Constrained RL has been applied

in a variety of works, including safety reinforcement learning (Safety RL) where safety must be ensured for all visited states [Dalal *et al.*, 2018]. For Safety RL with large state and action spaces, [Chow *et al.*, 2018] proposes an iterative algorithm based on a novel construction of Lyapunov functions. However, their theory only holds for tabular settings. Furthermore, [Prashanth and Ghavamzadeh, 2016; Chow *et al.*, 2017] propose constrained risk-sensitive reinforcement learning using Lagrangian multipliers. Nevertheless, according to [Tessler *et al.*, 2018], the saddle-point achieved by these approaches might not be the stationary point of the original problem. Besides, [Huang *et al.*, 2018; Lacotte *et al.*, 2019] study safe reinforcement learning with demonstration data, which ensures safe behaviors of a reinforcement learning agent by imitating the expert. In addition, [Jing *et al.*, 2020] also introduces demonstrations in the learning process but it uses demonstrations to guide the exploration and the constraints from demonstrations will decay gradually, results in optimizing according to the reward functions only in the last.

3 Methodology

In this section, we start with a brief introduction to the background including Markov Decision Process (MDP) and constrained Markov decision process (CMDP). Then we provide a basic objective function of behavior cloning to learn desirable behaviors from demonstrations. However, due to the imperfectness of demonstration in terms of quality and amount and the distribution shift problem which is inherent in the behavior cloning method caused by error accumulation, the simple supervised policy from offline demonstration may have poor performance and bad generalization. And thus we introduce an additional reward constraint to encourage the policy to optimize toward the direction of attaining human-like behaviors and guaranteed performance which implicit in the lower-bound reward constraint. The whole problem is modeled as a Constrained Markov Decision Process. Lagrangian relaxation technique is used to solve the problem automatically and the implementation details are illustrated at last.

3.1 Background

A standard Markov decision process (MDP) is defined by a tuple (S, A, P, R, γ), with state S, action space A, transition probability $P(s^{'}|s,a)$, reward function $R : S \times A \times S \to \mathbb{R}$, and discount factor $\gamma \in [0, 1]$. The goal of the agent is to maximize the expectation of the sum of discounted reward $J_R = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})]$ automatically by updating the parameters θ of policy $\pi(a|s;\theta)$. And a constrained Markov decision process (CMDP) represented by the tuple $(S, A, P, R, C, D, \gamma)$ is an MDP augmented with constraints that restrict a set of allowable policies for that MDP, where $C = \{c_1, c_2, ...c_m\}$ is a set of auxiliary cost functions (with each one is a function $c_i : S \times A \times S \to R$), and $D = \{d_1, ..., d_m\}$ is the set of upper bounds to the cost functions. In addition, let $C_{c_i}(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t c_i(s_t, a_t, s_{t+1})]$ denotes the expectation of the sum of discounted cost of policy π with respect to c_i. The set of feasible policies is then:

$$\prod_C \doteq \{\pi \in \prod : \forall i, C_{c_i}(\pi) \leq d_i\} \tag{1}$$

Besides, the goal of CMDP is to select a policy π that maximizes the object function $J_R(\pi)$ while satisfying the constraints. The constrained optimization problem can be expressed as:

$$\pi^* = \arg\max_{\pi \in \prod_C} J_R(\pi) \tag{2}$$

3.2 Proposed Method

Our method includes two parts: the first learns the desirable behaviors from human demonstration and the second imposes constraints on the reward to ensure acceptable performance.

Learn Desirable Behaviors

As human demonstrations implicitly contain the behavior pattern that we expected, we utilize the imitation learning method, specifically behavior cloning, to mimic the behaviors from human demonstration. The objective is to minimize the distribution discrepancy between the human demonstrations and the policy, which is formulated as follows:

$$\min_{\pi} \mathbb{D}[\rho_\pi(\cdot|s)||\rho_D(\cdot|s)] \tag{3}$$

where s represents the state of the environment and $\mathbb{D}(\cdot||\cdot)$ represents the discrepancy measure like KL divergence which is a common objective function in imitation learning, ρ_π and ρ_D depict the state action distribution induced by policy π and human demonstration D, respectively. Note that regularization techniques like $L1$ or $L2$ are always used for smoothing policy, but a smooth policy is not the only behavior pattern expected, that some behavior patterns are difficult to achieve through regularization such as arriving the goal through a specific door described in Grid-World experiment in Section 4.1.

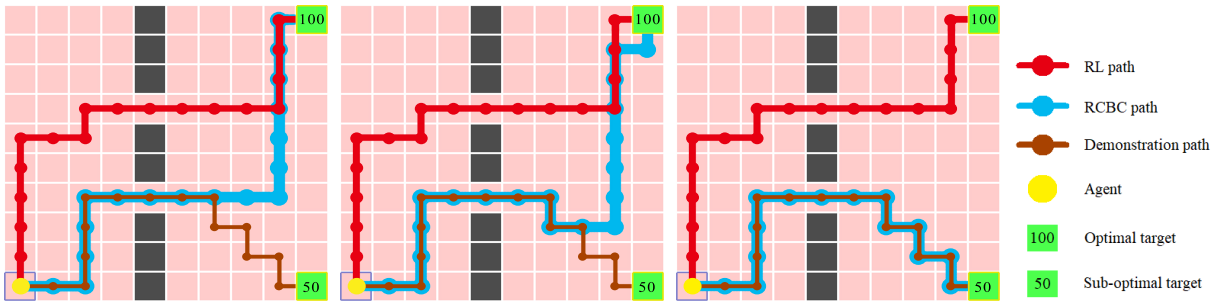
However, since the policy is completely tied up to the demonstration in this objective function, using it directly will prevent the policy from improving beyond the human demonstration. In addition, learning from non-optimal demonstrations may result in a policy with unacceptable low rewards, and small errors compounded over time during policy execution phases in the environment result in distribution mismatch that may lead the agent to new states which are not in demonstrations and the policy will make mistake. Therefore, we attach a reward constraint to the objective function and leverage the Lagrange relaxation to solve it automatically.

Constraining the Policy with a Lower-bound

We introduce a reward constraint into the objective function to attain guaranteed performance. More specifically, a lower bound on the expected return is given and denoted as $\hat{R}$. Thus, the constraint term encourages the policy to generate state-action pairs that satisfy $\mathbb{E}_{s,a\sim\pi}[Q(s,a)] \geq \hat{R}$, where $Q(s,a)$ is the Q-value for the state-action pair (s,a). Now the objective can be described as follows:

$$\min_{\pi} \mathbb{D}[\rho_\pi(\cdot|s)||\rho_D(\cdot|s)],\ s.t.\ \mathbb{E}_{s,a\sim\pi}[Q(s,a)] \geq \hat{R} \tag{4}$$

The above equation is a constrained optimization problem which can be solved by the Lagrange relaxation technique,



(a) α=0.8 (b) α=0.6 (c) α=0.4

Figure 2: The results in the Grid-World environment. The path of the original policy is represented by the red line, the path of demonstrations is represented by the brown line, and the path of our RCBC policy is represented by the blue line.

that the objective function with hard constraint will be transformed into an optimization problem without explicit constraints and the constraint exists in the way of a penalty term. Applying Lagrangian relaxation to the above equation will result in the Lagrange dual problem formulated as follows:

$$\begin{aligned} &\max_\pi \min_{\lambda\geq 0} \mathbb{E}_{s,a\sim\pi}[Q_\lambda(s,a)], \\ &Q_\lambda(s,a) = \lambda(Q(s,a) - \hat{R}) - \mathbb{D}[\rho_\pi(\cdot|s)||\rho_D(\cdot|s)] \end{aligned} \tag{5}$$

where $\lambda \geq 0$ is the Lagrange multiplier (penalty coefficient).

Intuitively, the value of λ affects the degree of the penalty and can be adjusted by updating λ with gradient descent, and the update of λ alternates with policy optimization until the constraints are satisfied. Note that the constraints may be unsatisfied for a long time, which results in an overall large magnitude of λ and makes the learning process unstable. To solve this problem, we re-weight the relative importance of distribution discrepancy and constraints, which gives us the following objective:

$$\begin{aligned} &\max_\pi \min_{\lambda\geq 0} \mathbb{E}_{s,a\sim\pi}[Q_\lambda(s,a)], \\ &Q_\lambda(s,a) = \tfrac{\lambda}{\lambda+1}(Q(s,a) - \hat{R}) - \tfrac{1}{\lambda+1}\mathbb{D}[\rho_\pi(\cdot|s)||\rho_D(\cdot|s)] \end{aligned} \tag{6}$$

The function will converge to a saddle point when $\nabla_\lambda \mathbb{E}[Q_\lambda(s,a)] = 0$, and the λ at this point is the expected trade-off between minimizing the distribution discrepancy and satisfying the constraints.

Furthermore, RCBC requires Q-value to calculate the constraint term in objective function hence it cannot be applied to Actor-Critic methods directly and we use some techniques to make it compatible with Actor-Critic methods. And our method can also be used to correct some behaviors of an existing model. These contents are described in detail in Appendix C.

4 Experiments

In this section, we aim at investigating (i) whether RCBC can learn from demonstration and avoid undesirable behaviors, and (ii) whether RCBC can obtain high returns while preserving the human-like style. To evaluate our method, we conduct extensive experiments on several widely used environments: a navigation task in Grid-World, two environments in MuJoCo [Todorov *et al.*, 2012], and finally, a more complex racing environment named TORCS [Wymann *et al.*, 2000]. Detailed environment descriptions of the last two experiments are given in Appendix D and E.

4.1 Grid-World

Experimental Setup

We use a 10*10 Grid-World environment as shown in Figure 2, the agent (yellow point) starts from the bottom left corner and is required to arrive at the goal grid (green squares) placed at the bottom right corner (+50 reward) or the top right corner (+100 reward). Besides, there is a wall (black squares) in the vertical direction and the agent can only pass through it from the two doors. In this environment, although it is equivalent to pass through any of the two doors to reach the goal, while the agent trained with PPO converges to pass through the upper door, and the path is shown with the red line in Figure 2. Thus in order to evaluate RCBC's ability to learn the desirable behaviors from sub-optimal demonstration, we provide a path shown with brown line in Figure 2 that ends at the bottom right corner which is the sub-optimal goal with 50 reward. Note that the demonstration path passes through the bottom door represents the desirable behavior. We conduct a set of experiments and intend to verify the following issues: (i) Whether the agent can learn to go through the bottom door and finally arrive at the optimal goal even the demonstration is sub-optimal? (ii) Whether the agent is sensitive to the reward threshold and can adapt to different threshold values to provide different trade-off points between reward maximization and human-desirable behavior generation? (iii) Whether RCBC is effective to modify an existing deficient model?

For the first issue, we train an agent with the sub-optimal demonstration using RCBC and hope the agent can learn to walk through the bottom door as well as reach the top right corner. For the second issue, we train the agent with three different threshold values. Here we reparameterize the reward constraint by α which represents the ratio of threshold and the max reward (100). We set α to be 0.8, 0.6, 0.4 in the following experiments corresponding to 80, 60, 40 total rewards, respectively. For the third issue, we load a pre-trained PPO model which can arrive at the optimal goal but pass through the top door and continue optimizing the policy with RCBC to encourage the agent to reach the optimal goal through the bottom door. We also test three different α values (0.8, 0.6, 0.4) in this experiment according to Eq.(7). The results are shown in Figure 2 and our analysis is in the next section.

$$\hat{R} = \alpha * Q_{old}(s,a) \tag{7}$$

Experimental Analysis

First of all, from the results in 2(a) and 2(b), we can find that the agent trained by RCBC (blue lines) learns the desirable behaviors (through the bottom door) and reaches the optimal goal, even if the demonstration is non-optimal. Furthermore, as shown in Figure 2, policies are different under different constraint values. In Figure 2(a), RCBC guides the agent to move along the path in the demonstration at first and realizes that the last six steps in demonstration violate the reward constraint then it guides the agent to move towards the top right goal. As depicted in Figure 2(b), the agent walks along with the demonstration for two more steps and finally moves to the top right corner too. Compared with the results in Figure 2(a), we can see that these two steps satisfy the constraint $\alpha = 0.6$ but violate the constraint $\alpha = 0.8$. The result path of $\alpha = 0.4$ is visualized in Figure 2(c). It is worth noting that all (s, a) pairs in demonstration satisfy the constraint when $\alpha = 0.4$ and the policy's behaviors are consistent with the demonstration which finally reaches the sub-optimal goal. At last, we test our method in the setting of correcting an existing model. Given the same α, RCBC can identically achieve the performance as shown in Figure 2, which proves that our method is also effective to amend an existing model.

4.2 MuJoCo

Experimental Setup

This set of experiments are based on the classic continuous control tasks named Inverted-Pendulum and Inverted-Double-Pendulum in MuJoCo[Todorov *et al.*, 2012]. As shown in Figure 6 (provided in Appendix D), both of the two tasks need to apply a horizontal force to the car to keep the pendulum balanced as long as possible, while Inverted-Double-Pendulum is more complicated than Inverted-Pendulum hence the pendulum is doubled. A more detailed description of state action space and reward setting can be seen in Appendix D.

In this experiment, the policy-based RL method usually makes the pendulum stand at the middle of the pole. We assume that keeping the pendulum stay at a specific location on the horizontal bar (+0.3 distance to center in Inverted-Double-Pendulum and +0.5 distance to center in Inverted-Pendulum) is the desirable behavior, shown in Figure 6(b) and Figure 6(d) (provided in Appendix D). We provide one demonstration trajectory that satisfies the constraint in each environment by a policy trained with PPO and reward shaping that giving a bonus when the pendulum reaches the target position. In the following experiments, RCBC is trained with the single trajectory and is hoped to generate desirable behaviors. To validate the robustness of our method, we add another set of experiments in which noise is injected into the demonstration by selecting 50 (s, a) pairs randomly from demonstration and changing actions to be the opposite, e.g., the action +0.1 will be changed to -0.1. We introduce PPO with reward shaping (denoted as PPO+RS), and GAIL-based PPO (denoted as PPO+GAIL). Furthermore, to figure out the relationship between the original objective and the constraint in Eq.(4), we also add another baseline (denoted as Constraint2) whose objective function is:

$$\max_{\pi} E_{s,a\sim\pi}[Q(s,a)],\ s.t.\ \mathbb{D}[\rho_\pi(\cdot|s)||\rho_D(\cdot|s)] \leq \epsilon \quad (8)$$

where ϵ is a threshold for behavior cloning constraint with 1.0×10^{-10} in the following experiments. The pseudo-code for Constrain2 is provided in Appendix B. Finally, it is worth noting that we erase the additional location-related reward in PPO+GAIL, Constrain2, and RCBC to ensure that desirable behavior patterns can only be learned from the demonstration.

Experimental Analysis

The results of Inverted-Pendulum and Inverted-Double-Pendulum are demonstrated in Figure 3-4, where RCBC, PPO+RS, PPO+GAIL, and Constrain2 are represented by blue, green, black, and red lines, respectively. For Inverted-Pendulum, it's obvious that RCBC finds a feasible solution and exhibits superior performance. From Figure 3(a), we can see that RCBC converges to the target location (average distance to the target position converges to near 0) in a much faster and more stable way. While PPO+GAIL tries to approach the target location in the early stage but crashed later. Constrain2 and PPO+RS learn the desirable behavior finally but the learning speed is slower than RCBC and their learning process is not quite stable. It is worth noting that RCBC even outperforms the PPO+RS which includes the target location in the reward function explicitly. On the other hand, as shown in Figure 3(b), we can see that RCBC also has the best performance in the reward evaluation which proves that our method can learn the desirable behaviors while achieving a high reward. For the experiments with noises in the demonstration, RCBC still surpasses other methods significantly in learning desirable behaviors and obtaining high total rewards, despite that the fluctuations in the training process will be a bit larger. The results are shown in Figure 3(c) and 3(d).

Moreover, Figure 4 shows the learning processes in the Inverted-Double-Pendulum environment. As it is harder to control the two individual pendulums simultaneously, the performance of all four methods has declined. Under the original demonstrations without noises (Figure 4(a)-(b)), RCBC finally found a policy that can both arrive at the target location and achieve a high total reward. The other three methods failed in learning the implicit behavior pattern in the demonstration although they learn faster in the reward metric. In the noisy situation (Figure 4(c)-(d)), RCBC converges to the target location with fluctuation while the other three methods failed to the target location again. Moreover, the learning speed of RCBC is also the fastest one in this situation.

4.3 TORCS

Experimental Setup

To demonstrate the effectiveness of RCBC in complex control problems, the last experiment is carried out on TORCS ([Wymann *et al.*, 2000], The Open Racing Car Simulator), The Open Racing Car Simulator), which is a popular simulation environment in autonomous driving. The target of the task is to safely drive along the track as fast as possible, more details about the task are provided in Appendix E.

In this experiment, we first train a policy with PPO according to the reward function formulated as Eq.(11) provided in Appendix E and observe that the car controlled by the policy jitters frequently during driving, which destroys human's driving experience. This result is consistent with the previous

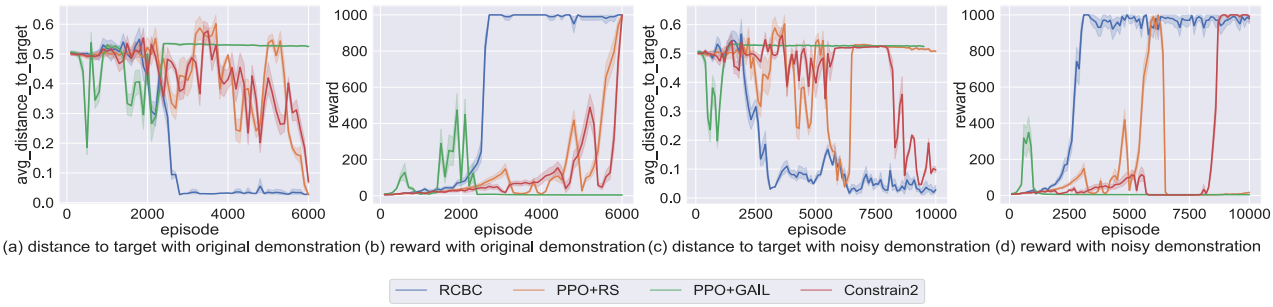


Figure 3: Experiment results on Inverted-Pendulum. The distance from the starting position of the pendulum to the target is 0.5.

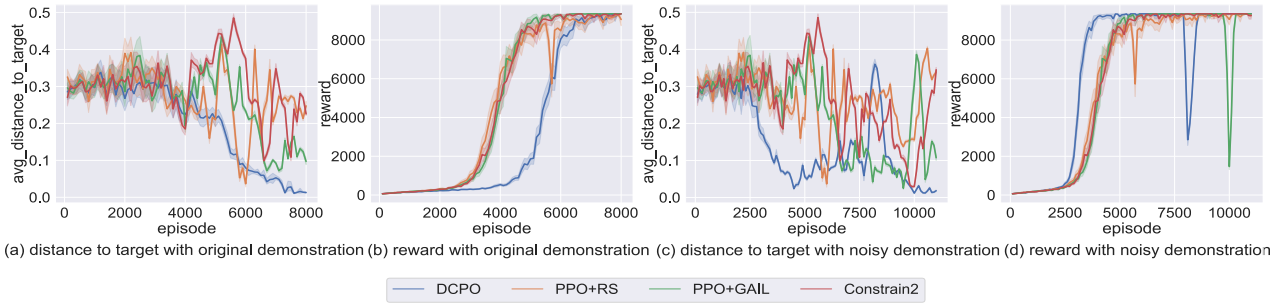


Figure 4: Experiment results on Inverted-Double-Pendulum. The distance from the starting position of the pendulum to the target is 0.3.

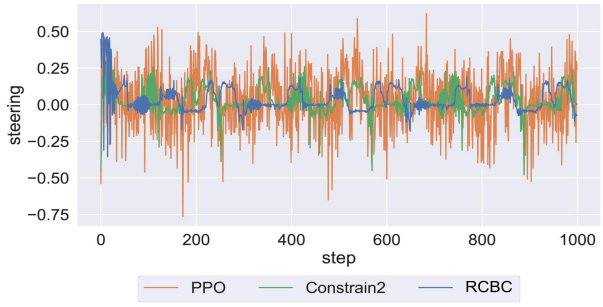


Figure 5: Steering values of the executed policies in TORCS.

study [Bohez *et al.*, 2019] which names the problem as bang-bang control. To solve it we collect some demonstration data in which the vehicle speed is still high but the driving process is more stable. We compare the result of RCBC according to this demonstration with Constrain2 as a baseline. PPO+GAIL is not included because it struggles in learning a stable policy.

Experimental Analysis

Results are provided in Figure 5, where the horizontal axis represents the number of steps and the vertical axis represents the steering values in action vectors. The results of RCBC, Constrain2, and PPO are marked with the blue, green, and orange lines, respectively. When optimized with PPO, we can observe that the absolute value of the steering action is larger and switches rapidly between negative and positive values. Although the speed of the car is high, this kind of driving style is not desirable for real-world driving tasks. The results of Constrain2 and RCBC are much better as they learned the smooth driving style from the demonstration, but there are still several high-frequency steering switches for Constrain2. The policy learned with RCBC is much smoother and consistent with the driving style of humans, where the two driving modes of turning and driving straight appear alternately. Besides, we provide the average speed and reward of each method in Table 1 provided in Appendix F. The speed of RCBC is slightly lower than PPO and we attribute this to the additional optimization process needed for the reward and constraint trade-off. We recorded the driving process of PPO and RCBC with a video and provide it in the supplementary materials with the name "torcs-driving.mp4".

5 Conclusion

Reinforcement learning aims to maximize the expected reward may generate undesirable behaviors in the real world due to the inferior local convergence or incompetent reward design. In this paper, we propose an approach to balance maximizing the accumulated reward and generating the desirable behaviors through combining imitation learning and constrained reinforcement learning together. Imitation learning is used to learn the desirable behaviors from demonstration and the constrained reinforcement learning sets a lower bound on the reward to ensure the guaranteed performance. We validate our approach in several environments and show that RCBC can learn human-like behaviors from non-optimal demonstration while maintaining a high reward. In future work, we seek to evaluate the performance of our method with off-policy algorithms whose sample efficiency is better. Besides, extending our method with transfer learning algorithms is also worth researching in the future.

References

[Achiam *et al.*, 2017] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 22–31, 2017.

[Bohez *et al.*, 2019] Steven Bohez, Abbas Abdolmaleki, Michael Neunert, Jonas Buchli, Nicolas Heess, and Raia Hadsell. Value constrained model-free continuous control. *arXiv preprint arXiv:1902.04623*, 2019.

[Bojarski *et al.*, 2016] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[Brys *et al.*, 2015] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[Chow *et al.*, 2017] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research*, 18(1):6070–6120, 2017.

[Chow *et al.*, 2018] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *Advances in neural information processing systems*, pages 8092–8101, 2018.

[Dalal *et al.*, 2018] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

[Gao *et al.*, 2018] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[Haarnoja *et al.*, 2017] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

[Huang *et al.*, 2018] Jessie Huang, Fa Wu, Doina Precup, and Yang Cai. Learning safe policies with expert guidance. In *Advances in Neural Information Processing Systems*, pages 9105–9114, 2018.

[Jing *et al.*, 2020] Mingxuan Jing, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Chao Yang, Bin Fang, and Huaping Liu. Reinforcement learning from imperfect demonstrations under soft expert guidance. In *AAAI*, pages 5109–5116, 2020.

[Kang *et al.*, 2018] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pages 2469–2478, 2018.

[Knox and Stone, 2009] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16, 2009.

[Lacotte *et al.*, 2019] Jonathan Lacotte, Mohammad Ghavamzadeh, Yinlam Chow, and Marco Pavone. Risk-sensitive generative adversarial imitation learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2154–2163, 2019.

[Ng *et al.*, 1999] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, page 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[Prashanth and Ghavamzadeh, 2016] LA Prashanth and Mohammad Ghavamzadeh. Variance-constrained actor-critic algorithms for discounted and average reward mdps. *Machine Learning*, 105(3):367–417, 2016.

[Ross and Bagnell, 2010] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Silver *et al.*, 2016] 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):484–489, 2016.

[Tessler *et al.*, 2018] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. In *International Conference on Learning Representations*, 2018.

[Todorov *et al.*, 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[Wymann *et al.*, 2000] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 4(6):2, 2000.

[Ziebart *et al.*, 2008] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

A. General Algorithm of RCBC

The general outline of the optimization procedure for RCBC is listed in Algorithm 1. We use the actor-critic structure as the overall framework.

Algorithm 1 RCBC

Input: demonstrations $D = \{(s_1, a_1), ..., (s_d, a_d)\}$, reward threshold $\hat{R}$, Lagrange multipliers $\lambda_0 = 0$, Lagrange learning rate η, actor parameters θ, critic parameters ϕ

for $k = 0 \rightarrow M$ **do**
  for $t = 1 \rightarrow T$ **do**
    Collect and store the tuple (s_t, a_t, r_t, s_{t+1}) in B
  end for
  Sample N transitions (s_i, a_i, r_i, s_{i+1}) from B
  Sample N state-action pairs (s_i^d, a_i^d) from D
  Set $y_i = r_t + \gamma V(s_{t+1})$
  Update the critic by minimizing the loss:
  $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - V(s_i))^2$
  Update the actor by minimizing the loss:
  $L = \frac{1}{N}\sum_{i=1}^{N}\mathbb{D}[\rho_\pi(\cdot|s_i^d)||\rho(\cdot|s_i^d)]$
  $+\frac{1}{N}\sum_{i=1}^{N}\lambda_k(\hat{R} - \log\pi(s_i, a_i) - V(s_i))$
  Lagrange multiplier update:
  $\lambda_{k+1} \leftarrow \lambda_k - \eta\frac{1}{N}\sum_{i=1}^{N}(\log\pi(s_i, a_i) + V(s_i) - \hat{R})$
end for
return policy parameters θ, critic parameters ϕ =0

B. General Algorithm of Constrain2

The pseudo-code for Constrain2 is listed in Algorithm 2. $A(s, a)$ is an estimator of the advantage function, $H(\pi)$ is the entropy of policy π and β is a hyper-parameter.

Algorithm 2 Constrain2

Input: demonstrations $D = \{(s_1, a_1), ..., (s_d, a_d)\}$, KL threshold ϵ, Lagrange multipliers $\lambda_0 = 0$, Lagrange learning rate η, actor parameters θ, critic parameters ϕ

for $k = 0 \rightarrow M$ **do**
  for $t = 1 \rightarrow T$ **do**
    Collect and store the tuple (s_t, a_t, r_t, s_{t+1}) in B
  end for
  Sample N transitions (s_i, a_i, r_i, s_{i+1}) from B
  Sample N state-action pairs (s_i^d, a_i^d) from D
  Set $y_i = r_t + \gamma V(s_{t+1})$
  Update the critic by minimizing the loss:
  $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - V(s_i))^2$
  Update the actor by minimizing the loss:
  $L = -\frac{1}{N}\sum_{i=1}^{N}\log\pi(a_i|s_i)A(s_i, a_i) - \beta H(\pi)$
  $+\frac{1}{N}\sum_{i=1}^{N}\lambda_k(\mathbb{D}[\rho_\pi(\cdot|s_i^d)||\rho(\cdot|s_i^d)] - \epsilon)$
  Lagrange multiplier update:
  $\lambda_{k+1} \leftarrow \lambda_k + \eta\frac{1}{N}\sum_{i=1}^{N}(\mathbb{D}[\rho_\pi(s_i, a_i)||\rho(s_i^d, a_i^d)] - \epsilon)$
end for
return policy parameters θ, critic parameters ϕ =0

C. Implementation Details

C.1 Compatible with Actor-Critic Method

To make our approach compatible with Actor-Critic based algorithms, we utilize some techniques in previous studies. According to [Haarnoja *et al.*, 2017], the policy π and the value function $V(s)$ can be defined by $Q(s, a)$ as follows:

$$\pi(a|s) = \frac{\exp\left(Q(s,a)\right)}{\sum_{a' \in A}\exp(Q(s,a'))} \quad V(s) = \log\sum_{a\in A}\exp(Q(s,a)) \tag{9}$$

Then $Q(s, a)$ can be defined by $V(s)$ and $\pi(a|s)$ based on above equation as follows:

$$Q(s, a) = V(s) + \log\pi(a|s) \tag{10}$$

Thus we can compute $Q(s, a)$ which is needed in Eq. (6) through $V(s)$ and $\pi(a|s)$, and our algorithm can be applied to Actor-Critic methods.

C.2 Extend to Amend an Existing Model

Our method can also be used to correct some behaviors of an existing model. Note that in reality, training a model costs lots of time as well as computation resources. If there are some terrible behaviors from an existing model, it is usually expensive to retrain it. In this situation, we can replace the constant threshold $\hat{R}$ in Eq.(6) to a dynamic variable, which is formulated as Eq.(7), where Q_{old} is the Q-value function of the existing policy π_{old}, and α is the hyper-parameter that controls the value of reward threshold. Intuitively, α represents the lower bound of the expected return ratio between π and π_{old}. Then we can start the training process by taking π_{old} as a pre-trained model and RCBC will try to fix the bad behaviors in π_{old} referring to the demonstrations while ensuring the expected return above the lower bound related to π_{old}.

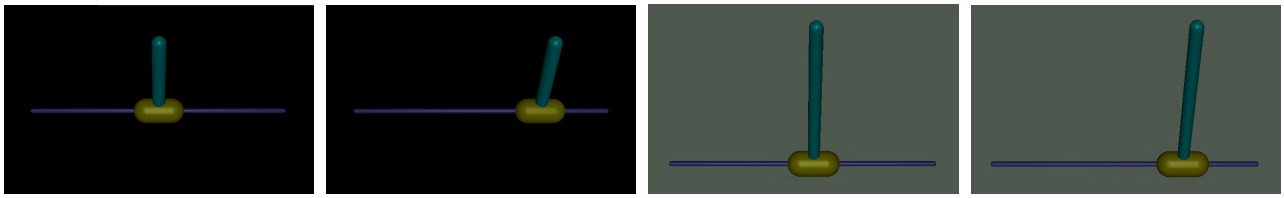
D. MuJoCo Environments

D.1 Environmental Description

In the Inverted-Pendulum task, a cart on which a pendulum is fixed through the pivot point can move in the horizontal direction along a finite-length bar. A horizontal force can be used to keep the pendulum balanced as long as possible. The observation is defined as a four-dimensional vector and the action is defined as the horizontal force added to the cart. The agent is given a positive reward +1 if the angle θ between the pendulum and the vertical line is kept within the desired degree at each time step. An episode is terminated if the cart moves out of the bar or θ is larger than the target. The task of the Inverted-Double-Pendulum is very similar to Inverted-Pendulum except that the pendulum is doubled. Besides, The observation of the Inverted-Double-Pendulum is expanded to an 11-dimension vector and the action are consistent with Inverted-Pendulum.

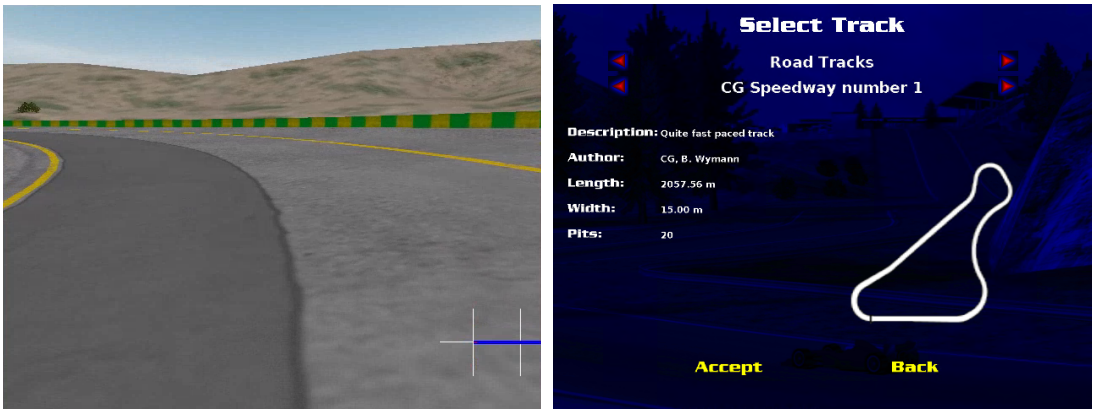
D.2 Modified MuJoCo environments

We modify the Inverted-Pendulum environment and the Inverted-Double-Pendulum environment to add a location-related constraint. Figure 6(a) and Figure 6(c) show the original environments in which the pendulum can stand anywhere



(a) Inverted-Pendulum (b) Modified Inverted-Pendulum (c) Inverted-Double-Pendulum (d) Modified Inverted-Double-Pendulum

Figure 6: The continuous control environments provided by MuJoCo.



(a) torcs (b) track

Figure 7: The continuous control environments provided by Torcs.

along the bar, in contrast, Figure 6(b) and Figure 6(d) show the modified environment in which the pendulum is required to stand in a specific location in the right part of the bar.

E. TORCS Environment

As shown in Figure 7, in this experiment we choose the CG SpeedWay track and the target is to safely drive along the track as fast as possible. The observation is defined as a 35-dimension vector, including the angle between the car and the track axis, the longitudinal and transverse speed of the car, and so on. Besides, the action is a 3-dimension vector with each dimension meaning steering, acceleration, and deceleration correspondingly. At last, the reward function is designed as follows:

$$R(t) = V_x cos(\theta) - V_x sin(\theta) - V_x |trackPos| \qquad (11)$$

where θ is the angle between the vehicle's direction and the track axis, V_x is the vehicle's speed in the track axis direction and $trackPos$ is the distance between the vehicle and the center of the road. The reward encourages the agent to move along the road and penalizes the agent if it moves across the road or deviates from the center of the road.

TORCS (The Open Racing Car Simulator) is an open-source 3D car racing simulator that has been adopted as a benchmark for many researchers. It provides the first-person view when driving as shown in Figure 7(a) and we choose the widely used track named CG SpeedWay whose shape is shown in Figure 7(b).

F. Average Reward and Speed of Methods in TORCS

	Avg. Speed (km/h)	Avg. Reward (per step)
Demonstration	109.70	-
PPO	108.92	94.86
Constrain2	95.13	78.98
RCBC	103.82	90.49

Table 1: Average reward and speed of each method.