Efficient Learning in Noisy Environment Using Policy Modulated PD Controller

Jiacheng Weng

***Abstract*—Reinforcement learning (RL) addresses the problem of learning optimal control policy by maximizing the expected sum of reward over a trajectory. Advancement of RL algorithms in the last decade has improved the learning efficiency and extended the capability of RL from learning in simple tabulated settings to complex, non-linear, and continuous control tasks. However, end-to-end RL is not often used in real robots due to expensive data collection and safety concerns. Instead, policy modulated controllers are often applied where the RL agent is used to modulate some other existing controllers with minimum performance and safety guarantees. In this paper, the progress of RL improvement is reviewed including classic tabulated RL, value-based RL, policy-based RL, model-based RL, and RL modulated control. The impact of RL modulation on learning efficiency is investigated empirically in the cart-pole environment. Our study suggests that policy modulated control provides an advantage in learning efficiency when the environment is noisy while achieving great performance in the noiseless environment on par with direct RL approaches.**

I. Introduction

reinforcement learning (RL) [1] is a subcategory of machine learning that addresses the optimal control problem through agent-environment interaction. An environment represents a plant which includes information about the system dynamics. An agent represents a controller of the plant. Given a state observation s_t at time step t, the agent generates an action a_t following some control policy π_θ parameterized by θ. The action a_t then drives the environment which produces the next state s_{t+1} constrained on the system dynamics $s_{t+1} = f(s_t, a_t)$ where f can be stochastic or deterministic. The environment also provides a reward signal $r_t = R(s_t, a_t)$ that evaluates the goodness of the previous action a_t at state s_t. The RL problem aims to solve for the optimal control policy $\pi^*(s)$ such that the sum of discounted reward is maximized.

Obtaining the optimal policy in RL requires many agent-environment interactions. When applying RL in expensive simulation environment and real world robotic systems, learning efficiency becomes very important as data collection is often slow and expensive. In the last decade, advances in RL algorithms and implementation strategies has shown great improvement in RL efficiency which enabled some amazing applications such as controlling neuromusculoskeletal models [2], robotic hands [3], robot arms [4, 5], and quadrupedal robots [6, 7]. When controlling real world robots, end-to-end RL is not used as it provides no safety guarantee. Instead, existing controllers are embedded in the learning loop and are modulated using RL [7, 8].

Jiacheng Weng is with the Mechanical and Mechatronics Engineering Department of the University of Waterloo, 200 University Ave W, Waterloo, ON N2L 3G1, Canada (e-mail:j5weng@uwaterloo.ca).

In this paper, the progress of RL advancement from theory to application is reviewed. In addition, the impact of using RL modulated controllers in terms of learning efficiency is analyzed empirically on the cart-pole system.

II. Background

In this section, the advancement of RL from algorithm development and real-world implementation is reviewed. This review covers RL from tabulated formulation to continuous formulation, from value-based approaches to policy-based approaches, from model-free RL to model-based RL, and from on-policy methods to off-policy methods. Relation between these different types of RL has been summarized in Table I. Additional real-world applications of RL are also reviewed at the end of this section.

A. RL with Known Model

In classical RL problems that follows bellman's principle of optimality [9], solving for optimal control policy π^* requires to estimate the optimal value function $V^*(s)$. As summarized in the planning case in Table Ia, dynamic programming (DP) can be used to obtain the optimal value estimate backward in time with greedy action. This results in the Bellman backup which is used in the value iteration (VI):

$$V_t(s) = \max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_{t+1}(s') \qquad (1)$$

where t is the time step, γ is the discount factor. $P(s'|s,a)$ is the transition probability distribution of the model where s' is the new state, s, a are previous state and action respectively. The policy is implicitly derived from V by replacing the max operation with the $argmax$ in equation (1). Another method for solving the DP problem is called policy iteration (PI) [1] which is similar to VI but update V and π alternatively. However, DP has two major constraints. First, the transition probability P is assumed a priori. Second, DP only works in practice for relative small problems with discrete state and action spaces which is also known as tabulated RL. These constraints make high-dimensional continuous state-action problem computationally intractable which is known as the curse of dimensionality [9].

B. Model-free Value-based RL

Section II-A discusses RL with a known model transition probability distribution. However, the model dynamics are often unknown which require empirical estimation. As listed in

TABLE I: High level summary of RL techniques from planning to control

	Prediction (predict future by estimating V given π)	**Control** (optimize future by finding the best π^*)
Planning (Known model dynamics)	Use DP assuming finite s and a - Value iteration - Policy iteration	Use DP assuming finite s and a - Policy iteration - Derive from optimal V^* with greedy action
RL (Unknown model dynamics)	Estimate model dynamics - Monte-Carlo - Temporal difference	see table below

(a)

	Model-free value-based (Learn V and Q; π is implicitly derived)	**Model-free policy-based** (Learning π without explicitly learning V and Q)	**Model-based** (Learn π, V using learned transition models)
On-policy methods (improve π using experience from π)	- Monte-Carlo control - SARSA	- PG - TRPO - PPO	- Dyna - Dreamer - MB-MPO
Off-policy methods (improve π using experience from μ)	- Q-learning - DQN - DDQN	- DDPG - TD3 - SAC	

(b)

the RL-prediction case in Table Ia, Monte-Carlo (MC) method aims to estimate true value through trajectory sampling [1]:

$$V^{\pi}(s) = V^{\pi}(s) + \alpha(G_{i,t} - V^{\pi}(s)) \tag{2}$$

where α is the update coefficient, $G_{i,t}$ is the estimated true state value from sampled trajectories. Another method is called Temporal Difference (TD) learning which aims to update V using one single transition rather than the full trajectory [1]:

$$V^{\pi}(s_t) = V^{\pi}(s_t) + \alpha([r_t + \gamma V^{\pi}(s_{t+1})] - V^{\pi}(s_t)) \tag{3}$$

where α is the update coefficient, γ is the discount factor. Note that both MC and TD estimate V conditioned on π. This means the estimated V becomes inaccurate after each update on π which makes optimization of V and π unstable in practice. Algorithm such as Monte-Carlo control, SARSA and Q-learning listed in Table Ib often have convergence issues when a non-linear value function is used [1].

To address the convergence issue, Deep-Q learning (DQN) [10] and its successor double-DQN (DDQN) [11] incorporated experience replay and fixed Q-targets during training to ensure numerical stability. Deep-Q learning also used deep neural networks (DNN) as the value function approximator which achieved human-level performance on multiple Atari games. However, DQN is still constrained by discrete action due to expensive evaluation of the state-action value function $Q(s, a)$. Although discretization can be applied to continuous action range, the resulting optimization problem is still large and computationally intractable in practice.

C. Model-free Policy-based RL

Unlike value-based model-free RL where π is not explicitly learned, policy-based model-free RL directly learns the optimal policy π^* and treat V as a function of π (shown in Table Ib). The main concept behind policy-based RL is called policy gradient (PG) [12] which involves optimization of the PG loss function [13]:

$$L^{PG}(\theta) = \mathbb{E}[\log \pi_{\theta}(a_t|s_t)(Q(s,a) - V(s))] \tag{4}$$

The PG formulation extended the capability of RL from discrete action domain to continuous action domain. Many on-policy methods (π is optimized based on the experience collected by itself) such as trust region policy optimization [14] and proximal policy optimization [15] have shown great capability of solving complex continuous state-action control problems such as humanoid walking. On-policy methods ensure that the experience is collected directly from the up-to-date policy which ensures stability during learning. However, the requirement of using up-to-date trajectories can make learning relatively inefficient as collected data are discarded after each policy update.

Off-policy methods, on the other hand, allows policy update with experience from a slightly different policy μ as noted in Table Ib which enables experience reuse from old policies. This idea is originated from [10] and is adopted by many state-of-the-art policy-based RL algorithms including deep deterministic policy gradient (DDPG) [16], TD3 [17], and soft actor-critic (SAC) [18].

D. Model-based RL

Unlike model-free RL where modeling environment dynamics are not considered during learning, model-based RL aims to estimate the environment dynamics or the transition model, and then use the learned dynamics to optimize V and π. This led to the classic formulation of Dyna architecture [19]. The goal of approximating the environment dynamics is to learn optimal policy based on imaginary data which reduces the need for real experience. Methods that incorporate the Dyna architecture includes world models [20], Dreamer [21] and model-based meta policy optimization (MB-MPO) [22]. These methods have shown great improvement in sample efficiency when compared to the model-free counterpart. However, studies have shown that model-based approaches suffer from accumulating error during trajectory rollout using transition models and cause limited performance on challenging motor control tasks [23]. Another study also indicated that with careful latent feature tuning, model-free RL can perform as well as model-based RL [24].

E. RL on Real Robots

Despite remarkable advances in RL algorithms, almost no real-world applications use RL for end-to-end control, especially safety-critical applications such as robotic systems. However, RL is often used along with existing controllers in a policy modulated manner which constrains the performance lower bound and safety through existing controllers [4, 5, 8]. Policy modulating control is also used in quadrupedal robots along with existing trajectory generators and stability controllers to achieve state-of-the-art control performance in complex and dynamically changing real environments [7, 8]. The inclusion of existing controllers injects prior control knowledge to the system, which essentially reduces the difficulty of the learning tasks and improves learning efficiency.

III. Methods

In this section, we explain the experimental setup for validating the learning efficiency by using policy modulation with existing controllers. We first cover the SAC which is the RL algorithm used in this study. and then the experimental setup.

A. Soft Actor-Critic

Our RL algorithm follows the SAC implementation as explained in [6] which is one of the most sample efficient model-free RL algorithms. The algorithm consists of four neural networks to approximate $V_\psi(s)$ parameterized by ψ, $Q_\phi(s,a)$ paramerized by ϕ, and $\pi_\theta(s)$ parameterized by θ. The state value function $V(s)$ is updated by minimizing the state value objective in (5):

$$J_V(\psi) = \mathbb{E}_\mathcal{D}[\frac{1}{2}(V(s_t) - \mathbb{E}_\pi[Q(s_t,a_t) - \log\pi(a_t|s_t)])^2] \quad (5)$$

where $\mathcal{D}$ is the replay buffer, and $\log\pi_\theta(a_t|s_t)$ captures the entropy of π. The state-action value function Q is updated by minimizing the objective in (6):

$$J_Q(\phi) = \mathbb{E}_\mathcal{D}[\frac{1}{2}(Q(s_t,a_t) - r(s_t,a_t) - \gamma\mathbb{E}_{s_{t+1}\sim p}[V(s_{t+1})])^2] \quad (6)$$

where γ is the discount factor, p is the state probability distribution. The objective aims to minimize the error of the output from Q and the estimate using the true reward r and value estimate V for the next step. In actual implementation, two Q networks are used to reduce the overestimation effect as stated in [11]. The policy in SAC is a stochastic policy which outputs the mean and standard deviation of all actions. The objective function for policy learning is shown in (7):

$$J_\pi(\theta) = \mathbb{E}_\mathcal{D}[\mathbb{E}_\pi[\alpha\log(\pi(a_t|s_t)) - Q(s_t,a_t)]] \quad (7)$$

where α is some scaling constant. The objective for π is derived from minimization of the Kullback–Leibler (KL) divergence between π and Q which aims to give higher probability to actions that have higher Q value similar to Q learning.

In our study, we used the same feed forward network architecture for V, Q, and π which includes two 64-neuron fully connected hidden layers between the input and output layers.

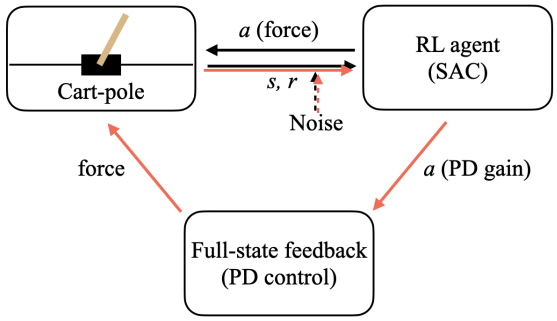

Fig. 1: Interaction between the cart-pole environment and the agent; back arrows show the interaction for direct RL; red arrows show the interaction for policy modulated PD control. Optional noises can be added to the states.

B. Experimental Setup

The environment used in this study is the classical cart-pole setup as shown in Fig. 1. The control goal is to balance a inverted pole by sliding a cart horizontally. The cart-pole system has four states including cart linear position x and velocity $\dot{x}$, and the pole angle θ and angular velocity $\dot{\theta}$. At initialization, all state values are drawn randomly from a uniform distribution with a limit of $\pm$ 0.1. The input to the cart-pole system is a horizontal force u. The reward function of the cart-pole system is expressed in (8):

$$R = \cos(\theta) - x^2 - 10^{-4}u^2 \quad (8)$$

The objective aims to minimize the deviation from the upright angle, zero horizontal location, and the control effort.

In our study, the learning efficiency of the SAC algorithm is tested in four settings. The first setting uses direct RL without state noise as shown in Fig. 1 with black arrows. In this setting, the agent outputs control force u directly with limit of $\pm$ 10. The second setting uses policy modulated control without state noise as shown in Fig. 1 with red arrows. Instead of interacting with the environment directly, a proportional-derivative (PD) controller is used to provide prior knowledge of the control architecture. The agent generates actions that correspond to the gains of the PD controller. The PD controller then generates the control force u based on the specified gains from the agent. Alternatively, the second setting can be seen as an automatic tuning problem of a variable PD controller using RL. The output of the gain values from the RL agent is limited to $\pm$ 300 which does not guarantee the stable region for gain selection. Last, the remaining two settings include state noise with Gaussian distribution $\mathcal{N}(0, 0.05^2)$ for both direct RL and policy modulated control.

For each experimental setting, three agents are trained separately with different random seeds to avoid arbitrary good or bad performance. The training progress for each setting is analyzed by averaging the reward over training steps across all three agents. In the noiseless settings, the training is limited to 100,000 steps, while in the noisy settings, the training is limited to 200,000 steps. After training, the final performance (accumulated reward) is evaluated by running ten tests with different cart-pole initial configurations.

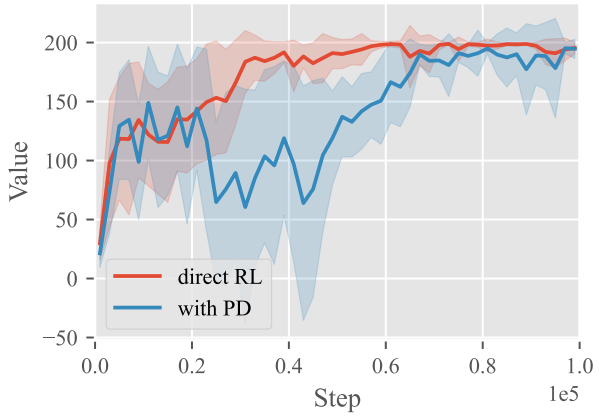

Fig. 2: Reward plot for agent performance ± SD in noiseless environment

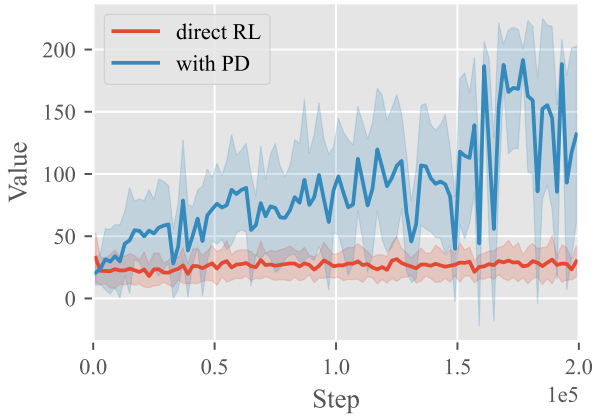

Fig. 3: Reward plot for agent performance ± SD in environment with Gaussian observation noise.

TABLE II: Final performance of the learned controller

	Direct RL	With PD	LQR
Noiseless	196.2 ± 3.0	196.1 ± 1.3	199.2 ± 0.08
Gaussian noise	30.0 ± 14.4	165.6 ± 46.9	199.2 ± 0.05

IV. Results and Discussion

In this section, we discuss the sample efficiency of the direct RL and policy modulated control in the cart-pole environment with and without state noise. Final agent performance is also evaluated.

First, two sets of SAC agents are trained directly with the environment and with the PD controller respectively in the noiseless environment. The reward plot is summarized in Fig. 2. The direct RL with SAC showed more efficient convergence to the maximum available reward of 200. For RL modulated PD controller, the initial learning phase has a large variance and was slower than the direct RL approach. This behaviour indicates potential challenges of learning stable regions for feedback gain selection. After 70,000 steps, the performance of the two agents converged with similar final performances as summarized in Table II.

Similarly, another two sets of SAC agents are trained directly with the environment and with the PD controller respectively but in the noisy environment as explained in III-B. The reward plot is summarized in Fig. 3. The direct RL with SAC performed poorly in the noisy environment with no noticeable improvement in accumulative reward through the training process. However, the RL modulated PD controller showed relatively steady improvement over time. The final performances of these agents are summarized in Table II with the RL modulated PD controller having five times the performance of the agents with direct RL. During the experiment with the RL modulated PD controller. It is found that the initialization of the agent parameters had a large impact on the final performance in the policy modulation setting. Two out of three trained agents reached a final performance of 197.6 ± 0.15, while the other one stagnated at around 101.5 ± 15.6.

To compare to classical control methods, a reference controller using infinite-horizon, time-invariant linear quadratic regulator (LQR) is obtained by simplifying the cart-pole system dynamics with the assumption of the small angle and angular velocity. The ground truth of the system parameters is assumed a priori which means the added state observation noise does not affect the identification of the system parameters. The optimal LQR feedback gain is obtained using the MATLAB lqr solver. The optimal LQR controller showed better performance than all other learning based controllers in both the noiseless and the noisy environment. This is expected as rule-based controller often provides smoother control which is difficult to achieve in controllers with neural networks. The superior performance of the LQR controller in the noisy environment also indicated less sensitivity of the PD controllers to observation noise, which makes the RL modulated PD controllers easier to learn in noisy environments.

V. Conclusion

A review of the recent development of RL for improving learning efficiency has been covered in terms of algorithmic improvement and practical implementation on real robotic systems. The use of RL modulation with PD controller has been tested against direct RL approach and LQR on the cart-pole control problem. The RL modulated PD controller showed slightly lower learning efficiency compared to direct RL in noiseless environment, while showed significant improvement in learning efficiency in noisy environment, while LQR still shows the best performance in this experiment.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] J. Weng, E. Hashemi, and A. Arami, "Natural walking with musculoskeletal models using deep reinforcement learning," *IEEE Robotics and Automation Letters*, 2021.

[3] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[4] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," *arXiv preprint arXiv:1906.08880*, 2019.

[5] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Learning multimodal representations for contact-rich tasks," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 582–596, 2020.

[6] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[8] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," in *Conference on Robot Learning*. PMLR, 2018, pp. 916–926.

[9] B. R. Ernest, "Dynamic programming," 2003.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[11] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[12] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPs*, vol. 99. Citeseer, 1999, pp. 1057–1063.

[13] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[17] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[19] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM Sigart Bulletin*, vol. 2, no. 4, pp. 160–163, 1991.

[20] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[21] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[22] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in *Conference on Robot Learning*. PMLR, 2018, pp. 617–629.

[23] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.

[24] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," *arXiv preprint arXiv:1910.01741*, 2019.