Backward Curriculum Reinforcement Learning

KyungMin Ko[1]

Abstract— **The current reinforcement learning algorithms use forward-generated trajectories to train the agent. The forward-generated trajectories give the agent little guidance, so the agent can explore as much as possible. While the appreciation of reinforcement learning comes from enough exploration, this gives the trade-off of losing sample efficiency. The sampling efficiency is an important factor that decides the performance of the algorithm. Past tasks use reward shaping techniques and changing the structure of the network to increase sample efficiency, however, these methods require many steps to implement. In this work, we propose novel backward curriculum reinforcement learning. Backward curriculum learning starts training the agent using the backward trajectory of the episode rather than the original forward trajectory. This gives the agent a strong reward signal, so the agent can learn in a more sample-efficient manner. Moreover, our method only requires a minor change in the algorithm, which is reversing the order of trajectory before training the agent. Therefore, it can be simply applied to any state-of-art algorithm.**

I. INTRODUCTION

In recent years, the development of GPU computation brought tremendous advances in deep learning. As a result, reinforcement learning could be applied to various fields. Reinforcement learning (RL) is a powerful technique that trains the agent to maximize the reward gain, so the agent can efficiently solve the problem. There are many applications that reinforcement learning is being used such as robotic manipulation tasks [1], Atari games [2], multiagent systems [3], and Alpha go [4]. However, reinforcement learning's natural reward function is usually sparse since a reward is given to the agent only when the agent completes the task. Moreover, conventional reinforcement learning problems are formulated with the agent blind to the goal of the task. This property will help the agent to discover optimal policy without any guidance from the human, but it often takes a large amount of computational power and samples to train.

In our work, we introduce novel backward curriculum reinforcement learning, which uses trajectories of the agent in reverse order to train the agent. This enables our agent to start the training process by recognizing the goal of the task. As a result, agents will be trained in a sample efficient way by utilizing strong reward signals during "reverse learning." Our method does not need to modify any of the neural network structures and requires any other previous knowledge to train the agent. Moreover, reverse learning can be simply applied to any state-of-the-art algorithm such as PPO [5], A3C [6], and SAC [7] within only two simple steps. First, collect the trajectory of the agent as usual, then reverse the order to train our agent. Our reverse learning is using the concept of curriculum learning, which manually changes the order of the training process to train more efficiently. We empirically tested our reverse learning method using REINFORCE [8] and REINFORCE with Baseline [9] algorithms on CartPole-v1 and Lunar Lander-v2 environments in an Open AI gym [10], which both use discrete action spaces. Moreover, we further investigated the effects of the return normalization technique [11], variation of the learning rate, and structure of neural network on the performance of reverse learning.

II. RELATED WORKS

There are some approaches published currently that share similar features with our reverse learning method. First, imitation learning [12] requires the demonstration of experts who explains to the agent how to reach the goal of the task efficiently. This approach can use samples efficiently to train our agents. However, imitation learning limits agents to take the advantages of exploration. Moreover, it requires lots of work since we need to find experts who can demonstrate completing the task, then record the process to train our agent. Instead, our reverse learning doesn't need extra work, but we only need to reverse the trajectory to train. In this case, we are using our backward trajectory as an expert in imitation learning. Curriculum learning, which modifies the schedule of the learning process, has been applied to various machine learning tasks. The curriculum learning trains the agent on easier examples first, then constantly increasing the level of difficulty to solve the problem [13]. In our reverse learning, we start training the agent using from the end of the episode to the beginning of the episode, so we are following the concept of curriculum learning since it is easier for the agent to complete the task from near the goal state rather than start state. Previously, curriculum learning has been applied to pre-specified tasks such as shooting the ball into a goal [14]. Similar work was proposed by Barnes that solved difficult robotic problems [15]. However, their method requires partitioning the full task space, which limits application to various problems. Instead, our reverse learning can be applied to various machine learning tasks since this method doesn't require previous knowledge to train the agent.

III. PRELIMINARIES

In this work, we consider finite discrete time horizon Markov decision process (MDP) by a tuple $M =$

*This work is supported by Georgia Tech's summer undergraduate in engineering program (SURE), which is part of the National Science Foundation (NSF). Special thanks to professor Maguluri and graduate mentor Sajad Khodadadian for advising during the project.

[1]Kyung Min Ko is with the department of electrical engineering, Purdue University, Indiana, USA, ko120@purdue.edu

$(S, A, P, r, \gamma, \rho_0, T)$. We define S as a state set, A as an action set, $P : S \times A \times S \to R$ as a transition probability. r is a bounded reward function, γ is a discounted reward factor, ρ_0 is the initial state distribution, and T is the trajectory of the moving agent. In the Markov decision process(MDP), at each time step, the agent takes an action, receives a reward, and moves to the next state using transition probability P. The goal of reinforcement learning is to find the optimal policy $\pi_\theta(a_t|s_t)$ that maximizes reward gain. In our reverse learning, we reverse the order of the agent's trajectory T and call it T_b, and use it to train the agent from goal state s_g to initial state s_0. Since our agent will start the training near the goal state s_g, it will give the agent a strong reward signal which gives meaningful guidance to the goal.

IV. Backward Curriculum Learning

A. Backward Curriculum Learning in REINFORCE

REINFORCE algorithm is one of the basic policy gradient algorithms that gave insights to build state-of-art policy gradient algorithms. We first applied backward curriculum learning to this algorithm since it is a basic policy gradient algorithm, which is easy to implement and intuitive. The REINFORCE algorithm has a simple structure that uses cumulative discounted returns and log probability of choosing an action to compute its gradient. However, the original REINFORCE algorithm has a sparse reward function that lowers the performance on complicated tasks. Backward curriculum learning can solve this problem by letting the agent start updating its gradient from the end of the episode. Therefore, our agent will recognize the goal at the beginning of the training process, which will replace the natural sparse reward function with a strong reward signal. Our detailed algorithm is explained in Algorithm 2. We first collect the sample trajectory using the policy network, then we simply flip the order of episodes to compute the loss function in reverse order. The major difference between the original REINFORCE algorithm is that it reverses the order of the episode before starting to take its gradient.

Algorithm 1 REINFORCE

```
Collect the sample trajectories following $\pi_\theta(a_t|s_t)$
$T = eqepisode\ length$
for $t = 1$ to $t = T - 1$ do
    Compute Return $G$
    $\theta \leftarrow \theta_{old} + G\nabla(\log \pi_\theta(a_t|s_t))$
    Update $Optimizer$
end for
```

B. Backward Curriculum Learning in REINFORCE with Baseline

The main problem associated with the REINFORCE algorithm is dealing with a high variance that may cause a divergence of policy network parameters. The common way to reduce the variance is to subtract the baseline [9]. In the REINFORCE algorithm, a value function is an appropriate baseline to subtract from return G. Agent will have both policy network and value network, and our loss function will be $\theta \leftarrow \theta_{old} + (G - V)\nabla(\log \pi_\theta(a_t|s_t))$. This baseline will decrease variance by reducing the step size of the gradient. In our backward REINFORCE with the baseline algorithm, we first collect the trajectories using our policy, then flip the order of the episode to compute the loss function as shown on algorithm 4.

Algorithm 2 Backward Curriculum REINFORCE

```
Collect the sample trajectories following $\pi_\theta(a_t|s_t)$
$T = eqepisode\ length$
for $t = T - 1$ to $t = 1$ do
    Compute Return $G$
    $\theta \leftarrow \theta_{old} + G\nabla(\log \pi_\theta(a_t|s_t))$
    Update $Optimizer$
end for
```

Algorithm 3 REINFORCE with Baseline

```
Collect the sample trajectories following $\pi_\theta(a_t|s_t)$
$T = eqepisode\ length$
for $t = 1$ to $t = T - 1$ do
    Compute Return $G$
    $\theta \leftarrow \theta_{old} + (G - V)\nabla(\log \pi_\theta(a_t|s_t))$
    Update $Optimizer$
end for
```

Algorithm 4 Backward Curriculum REINFORCE with Baseline

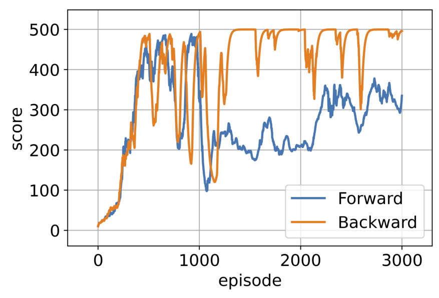
```
Collect the sample trajectories following $\pi_\theta(a_t|s_t)$
$T = eqepisode\ length$
for $t = T - 1$ to $t = 1$ do
    Compute Return $G$
    $\theta \leftarrow \theta_{old} + (G - V)\nabla(\log \pi_\theta(a_t|s_t))$
    Update $Optimizer$
end for
```

V. Experimental Result

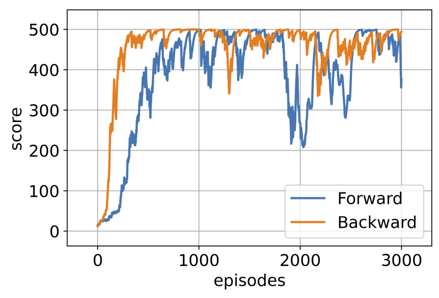
We used a reinforcement learning testing environment from Open AI Gym [10] to test the performance of our backward curriculum learning algorithm both on REINFORCE and REINFORCE with baseline algorithms. For experiments, we used Cart pole-v1 and Lunar Lander-v2 environments. The default structure of the network is a multi-layer perceptron network with 2 layers and 128 neurons as a baseline. Moreover, we analyzed the effect of return normalization as well as modifying the structure of networks.

A. Cart Pole Environment

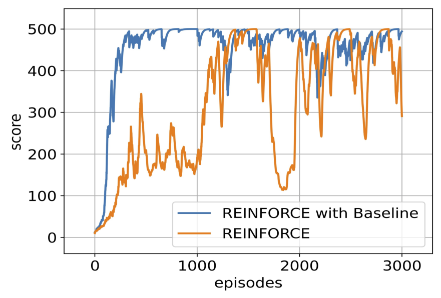
In this section, we first applied our backward curriculum method to the REINFORCE algorithm. The goal of the Cartpole-v1 algorithm is to balance the pendulum on the cart from falling by applying a discrete action domain with the force of +1 or -1 to the cart. The environment is considered



(a) REINFORCE

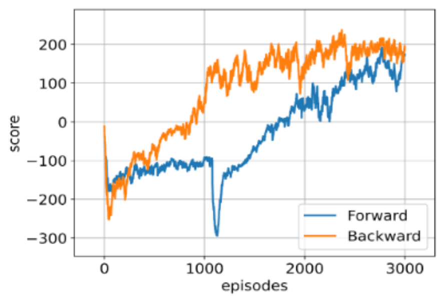


(b) REINFORCE with Baseline

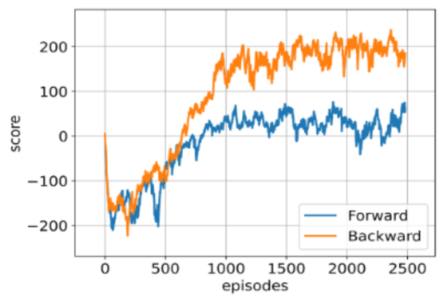


(c) Comparing REINFORCE and REINFORCE with Bassline

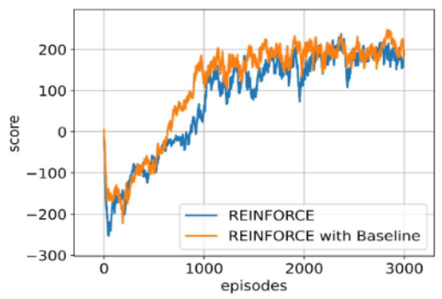
Fig. 1. Experimental Results on Cart Pole Environment



(a) REINFORCE



(b) REINFORCE with Baseline



(c) Comparing REINFORCE and REINFORCE with Bassline

Fig. 2. Experimental Results on Lunar Lander Environment

solved when the agent reaches a score of 500 without falling down the pole from the cart.

The comparison of the backward curriculum algorithm and the original REINFORCE algorithm is shown in figure 1(a). Until 1000 episodes, there is no conspicuous difference, but after that point, the backward curriculum algorithm clearly outperforms the original algorithm and solves the environment within 3000 episodes.

Next, we took experiments using REINFORCE with a baseline algorithm to analyze the effect of the backward curriculum learning algorithm. In figure 1(b), the backward curriculum algorithm solved the environment within 250 episodes, but the original REINFORCE with the baseline algorithm not only took longer episodes but also has a high variance that disturbs the agent from remaining in the goal state. We can further investigate that adding a baseline to the REINFORCE algorithm effectively solved a high variance problem that also improved the performance of the algorithm since it took about 1500 episodes to solve the environment using REINFORCE algorithm, but adding baseline results into solving the Cart pole environment less than 500 episodes as shown on figure 1(c).

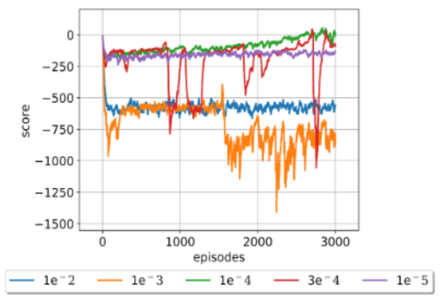
B. Lunar Lander Environment

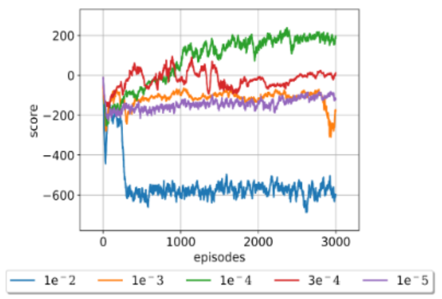
The next environment that we used to test our algorithms is the LunarLander-v2 environment from OpenAI gym. The goal of this environment is to make the agent safely land down on the goal region from the sky without crashing down the agent. The action space consists of 4 actions, which are resting, firing left, right, and main engines. When an agent reaches 200 scores without crashing, this environment is considered solved. In this section, we not only compared the performance of the backward curriculum algorithm and the original algorithm but also further experimented with the effect of return normalization.

As shown in figure 2(a), our backward curriculum REINFORCE algorithm reaches the goal state with an average of 200 scores within 2000 episodes, but REINFORCE algorithm solves the environment within 3000 episodes, which clearly shows that letting the agent know the goal of the environment at the beginning of the training process significantly helps the agent to reach the goal using fewer samples. Moreover, in figure 2(b), the REINFORCE with baseline using the backward curriculum learning algorithm finishes the task even faster with 1000 episodes using a reverse method, however, the original algorithm took more than 2500 episodes. Finally, we compared the performance of backward curriculum REINFORCE and backward curriculum REINFORCE with the baseline in figure 2(c). As we can observe in figure 2(c), both methods successfully reach the goal score, but backward curriculum learning on REINFORCE with baseline has less variance and uses fewer samples to solve the task.

C. Return Normalization

Even though reverse learning on the REINFORCE algorithm gradually optimized sample efficiency, it still has


(a) Without Return Normalization


(b) Return Normalization

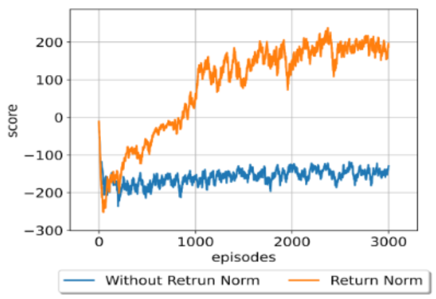
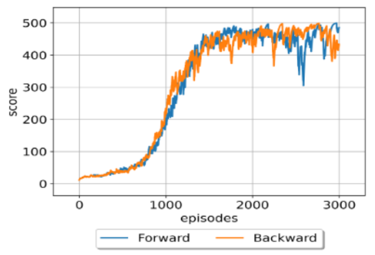

(c) Comparing Return Normalization and Without Return Normalization

Fig. 3. Effect of Return Normalization on Backward Curriculum Learning on Lunar Lander Environment


(a) Comparing Backward and Forward Method on Deeper Network

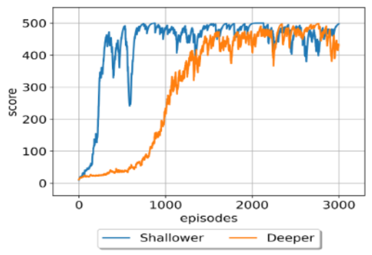

(b) Comparing Shallower Network and Deeper Network

Fig. 4. Effect of Depth of Deep Neural Network on Backward Curriculum Learning on CartPole Environment

a problem with high variance. If we take a close look at the original REINFORCE loss function $\theta \leftarrow \theta_{old} + G\nabla(\log \pi_\theta(a_t|s_t))$, the step size of the gradient mainly depends on G and log $\log \pi_\theta(a_t|s_t)$. The log of action probability is usually acceptable, but the return has a high magnitude which will dynamically increase the step size of the gradient. The choice of step size is crucial in reinforcement learning since a small step size slows down the convergence rate, while a large step size may cause oscillations or divergence of policy networks due to overshooting [16]. Therefore, we can apply return normalization to our method to stabilize the high variance. The basic idea of return normalization is making the mean of return to 0 and variance with 1, so we can scale down the magnitude of the return to avoid overshooting problems. We can do this by subtracting the average of return and dividing it by the standard deviation of return by following equation 1 [11]. As a result, applying return normalization will make the backward curriculum REINFORCE algorithm stable, which will increase the performance of the algorithm. We have applied return normalization both on the original algorithm and our backward curriculum algorithm simulated on the Lunar Lander environment.

As we can observe in figure 3(a), before applying return normalization, the learning curve has high variance and some of the learning rates result in divergence. Instead, applying return normalization decreased the variance significantly, and an agent with a learning rate e^{-4} solved the environment within 3000 episodes as shown in figure 3(b). This result demonstrates that return normalization not only optimized the performance but also minimized the variance as shown in figure 3(c), which shows the comparison between applying return normalization and without return normalization with a best-performing learning rate e^{-4} .

$$Return\ Norm = \frac{return - average\ return}{standard\ deviation\ of\ return} \tag{1}$$

D. Comparison between Deep and Shallow Network

The depth of the network is one of the important factors that affect the performance of the algorithm. In general, a deep neural network requires more computation power since increasing the number of layers will yield more parameters to compute the gradient. Instead, a shallow network has the benefit of quick computation since there are fewer parameters on fewer layers. Both shallow and deep networks have different benefits, but we discovered that for simple environments like Cartpole, the shallow network works better than the deep network. For a simple environment, the agent can solve the problem by applying simple heuristics without wasting computation efforts [17]. If we use deeper networks to solve the simple environment, it may lead to overthinking, which causes computational waste, and eventually leads to a misclassification [17].

We took experiments to compare the performance in the Cartpole environment using both shallow and deep networks. Our baseline network structure is two layers with 128 neurons, and we compared our baseline network to a deeper netork, which has 3 layers with 256 neurons. We used backward curriculum REINFORCE with baseline and original REINFORCE with baseline method to observe the difference between them on deeper networks. The result was interesting since our backward curriculum method didn't improve the performance as we expected as shown in figure 4(a). We concluded that on a deeper network, the network already has an accurate approximation, so letting the agent start the training process while recognizing the goal of the task has little effect on the performance. Moreover, we compared the performance between shallow and deep networks in the Cartpole environment. As shown in figure 4(b), the agent with shallow networks could finish the task using under 500 episode samples, but the agent with deeper networks took more than 1500 episodes to reach the goal state. Therefore, we can conclude that backward curriculum learning works best in a shallow network to solve simple tasks by maximizing sample efficiency.

VI. Conclusion

In this paper, we proposed a novel backward curriculum reinforcement learning that addresses the natural sparse reward function problem. Our method simply reverses the order of the episode before starting the training process, so it will let the agent recognize the goal of the task in the beginning. Therefore, it replaces the natural sparse reward function with a strong reward signal, which will optimize the sample efficiency. Unlike the previously proposed method, backward curriculum learning doesn't require many steps to modify the structure of the code, so we can directly apply our method to current state-of-art algorithms. We chose REINFORCE and REINFORCE with a baseline algorithm to test our method since it is the baseline of state-of-art algorithms. We empirically proved that backward curriculum learning uses fewer samples to solve the given tasks. Moreover, we tested the effect of return normalization and depth of the network on our reverse learning algorithm. We discovered that backward curriculum learning best fits while solving simple tasks like Cartpole with a shallow network. In future work, we will apply our backward curriculum learning on current state-of-art algorithms to test on various environments.

References

[1] T.-H. S. Li, Y.-T. Su, S.-W. Lai, and J.-J. Hu, "Walking motion generation, synthesis, and control for biped robot by using pgrl, lpi, and fuzzy logic," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 3, pp. 736–748, 2010.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[3] K.-S. Hwang, S.-W. Tan, and C.-C. Chen, "Cooperative strategy based on adaptive q-learning for robot soccer systems," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 569–576, 2004.

[4] S. D. Holcomb, W. K. Porter, S. V. Ault, G. Mao, and J. Wang, "Overview on deepmind and its alphago zero ai," in *Proceedings of the 2018 international conference on big data and education*, 2018, pp. 67–71.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[6] M. Sewak, "Actor-critic models and the a3c," in *Deep Reinforcement Learning*. Springer, 2019, pp. 141–152.

[7] 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.

[8] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[9] L. Weaver and N. Tao, "The optimal reward baseline for gradient-based reinforcement learning," *arXiv preprint arXiv:1301.2315*, 2013.

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[11] T. Schaul, G. Ostrovski, I. Kemaev, and D. Borsa, "Return-based scaling: Yet another normalisation trick for deep rl," *arXiv preprint arXiv:2105.05347*, 2021.

[12] K. Ciosek, "Imitation learning by reinforcement learning," *arXiv preprint arXiv:2108.04763*, 2021.

[13] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *arXiv preprint arXiv:2003.04960*, 2020.

[14] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning," *Machine learning*, vol. 23, no. 2, pp. 279–303, 1996.

[15] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, 2013.

[16] M. Pirotta, M. Restelli, and L. Bascetta, "Adaptive step-size for policy gradient methods," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[17] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *International conference on machine learning*. PMLR, 2019, pp. 3301–3310.