Improving Generalization with Cross-State Behavior Matching in Deep Reinforcement Learning

Extended Abstract

Guan-Ting Liu
National Taiwan University
Taipei, Taiwan
f07944014@ntu.edu.com

Guan-Yu Lin
National Taiwan University
Taipei, Taiwan
r09944017@ntu.edu.tw

Pu-Jen Cheng
National Taiwan University
Taipei, Taiwan
pjcheng@csie.ntu.edu.tw

ABSTRACT

Representation learning on visualized input is an essential yet challenging task for deep reinforcement learning (RL). To help the RL agent learn more general and discriminative representation among various states, we present cross-state self-constraint (CSSC). This novel technique regularizes representation learning by comparing state embedding similarities across different state-action pairs. We test our proposed method on the OpenAI Procgen benchmark with Rainbow and PPO and demonstrate significant improvement across most Procgen environments.

KEYWORDS

Reinforcement Learning; Generalization; Behavior Matching

ACM Reference Format:
Guan-Ting Liu, Guan-Yu Lin, and Pu-Jen Cheng. 2022. Improving Generalization with Cross-State Behavior Matching in Deep Reinforcement Learning: Extended Abstract. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 3 pages.

1 INTRODUCTION

Reinforcement learning (RL) has achieved tremendous success in mastering video games [8] and the game of Go [12]. While training agents using deep reinforcement learning algorithms, we usually assume that the agent could extract appropriate features from different states and take actions accordingly. However, as more and more research works (Zhang et al. [15], Song et al. [13], Dabney et al. [3]) have pointed out, even well-trained RL agents that learn from visualized input tend to memorize spurious patterns rather than understanding the essential generic features of a given state. For example, an agent might pay more attention to the change of irrelevant background rather than noticing the obstacles or enemies [13].

In order to improve RL agent's generalization ability in the new environment, various regularization methods like stochastic policy [4] and data augmentation [6] have been proposed and tested. Data augmentation like random crop [6] or random convolution [7] have also been proposed recently and provide considerable generalization enhancement to the unseen levels of various environments (Tassa et al. [14], Cobbe et al. [2], Cobbe et al. [1]). The agent acts on multiple augmented views of the same input and learns from these prior injected data. However, modifying state information (injecting prior information to the data) may be risky or detrimental for representation learning because vital features may be altered or lost. For example, flipping state images might change the corresponding behavioral meaning, and cropping the input image might lose critical features like the position of enemies appearing on edge.

To avoid losing the informative features of the visualized input, we choose a different approach. As human learners, we try to recognize general patterns across numerous states and act accordingly. In other words, if a well-trained agent conducts the same action (or behavior) in two different states, we would infer that the agent has conceived similar feature patterns in these states. Based on this intuition, we designed a novel constraint that performs regularization directly in the representation space of the learning agent.

We test this novel constraint in combination with Rainbow [5] and PPO [11], two of the most well-known algorithms in RL. We design the proposed cross-state self-constraint (CSSC) by comparing the similarity of multiple state pairs according to the action sequences taken by the rational agent.

2 METHOD

2.1 Behavior Definition

We describe a typical Markov Decision Process(MDP) as (X, A, R, P, γ) with state space X, action space A, reward function R, state transition function P and discount factor γ. The agent would take an action a_i at state x_i and then being transited to x_{i+1} by the environment with a given step reward r_i. Here we define the behavior set $B_i^n = \{(a_i, a_{i+1}, ..., a_{i+n-1}) | a \in A\}$ as a set of action series of length n taken by the agent since state x_i. For $b_i^n \in B_i^n$ and $b_j^n \in B_j^n$, $b_i^n = b_j^n$ only if $a_{i+p} = a_{j+p}$ for $0 \le p \le n-1$. With this definition in mind, we can infer that if the agent conduct the behavior $b_i^2 = (left, fire)$ at state x_i, it would move left in state x_i and fire in the next state x_{i+1}. In the following paragraph we coin terms like unigram, bigram and trigram for behaviors of length one, two and three respectively.

2.2 Implementation of Cross-State Self-Constraint in combination with Rainbow and PPO

For each sample of state triples$(x_p, x_q, x_r) \in X$ with $(b_p^n, b_q^n, b_r^n) \in B^n$ and $b_p^n = b_q^n \neq b_r^n$, we decompose the estimator $\hat{x}_{pqr}$ and define it as:

$$\hat{x}_{pqr} := \hat{x}_{pq} - \hat{x}_{pr} = e_\theta(x_p) \cdot e_\theta(x_q) - e_\theta(x_p) \cdot e_\theta(x_r) \quad (1)$$

where e_θ is the encoder function with weight θ that maps the pixel input to 1-D array feature vector with shape like [*element 0, element*

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

1, ..., element n-1] as the state representation. We directly perform inner-product on $e_\theta(x_p)$ and $e_\theta(x_q)$ to calculate the representation similarity $\hat{x}_{pq}$ between x_p and x_q. Please note that the encoder function is the same part of the original base model used for feature extraction on visualized input. To sample these state triplets without modifying the base DQN or PPO algorithm, we collect these state triples in the same batch of transitions used to calculate Bellman loss or policy loss. For policy-based algorithms and DQN without prioritized replay [10], we pair up the state triples for each transition in the sample and calculate the CSSC loss as follows:

$$\mathcal{L}_{CSSC} = -\frac{1}{N_{D_s}} \sum_{(p,q,r)\in D_s} \ln \sigma(\hat{x}_{pqr}) \tag{2}$$

where D_s represents the sample batch of size N_{D_s} from the replay buffer and σ is the sigmoid function. Then, we train the base algorithm in combination with the auxiliary CSSC loss as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{base} + \beta_{cssc} \cdot \mathcal{L}_{CSSC} \tag{3}$$

where β_{cssc} is the hyperparameter to control the contribution of CSSC during training.

In the case of Rainbow with CSSC, we design the loss function in combination with the importance sampling (IP) weight as follows:

$$\widehat{\mathcal{L}_{total}} = (\widehat{\mathcal{L}_{Rainbow}} \bigoplus (\beta_{cssc} \cdot \widehat{\mathcal{L}_{CSSC}})) \odot \widehat{W_{IP}} \tag{4}$$

where $\bigoplus$ and $\odot$ are element-wise add and multiplication respectively.

We find that setting β_{cssc} to 0.01 with Rainbow and 0.1 with PPO works well in most cases, so we stick with these settings for all the experiment conducted in the next section.

3 EXPERIMENT

Our primary goal for CSSC is to enhance the generalization capability of the RL algorithm to unseen levels. Thankfully, OpenAI Procgen[1] presents a collection of procedurally generated environments where the training and testing environments differ in visual appearance and layout structure. Therefore, we evaluate CSSC in the following ways: (i) generalization improvement on 16 easy-mode games (ii) generalization improvement on 16 easy-mode games with PPO on OpenAI Procgen.

We use the IMPALA CNN architecture recommended by Cobbe et al. [1] for the Rainbow and PPO model on the Procgen benchmark in the following experiments. All the experiments in this paper are conducted in the single-agent setting. We use “unigram CSSC" or “bigram CSSC" to indicate the behavior length used to pair up the state triplet for CSSC.

3.1 Generalization on Procgen with Rainbow

As Cobbe et al. [1] have suggested, we conduct 25 million timesteps of training on 200 levels and evaluate on full distribution of testing levels across 16 Procgen games in easy mode. We test unigram CSSC in comparison with vanilla Rainbow and plot the result in Figure 1.

3.2 Generalization on Procgen in Easy mode with PPO

We test CSSC with PPO on all 16 games of the Procgen benchmark in Easy mode. We conduct 25 million timesteps of training on 200 levels and evaluate on full testing levels across all 16 Procgen games. From normalized score shown in Table 1 we can tell that CSSC helps reduce the gap between training and testing performance. Because our proposed method is orthogonal to other approaches, we test CSSC in combination with UCB-DrAC, the state-of-the-art data augmentation method proposed by Raileanu et al. [9].

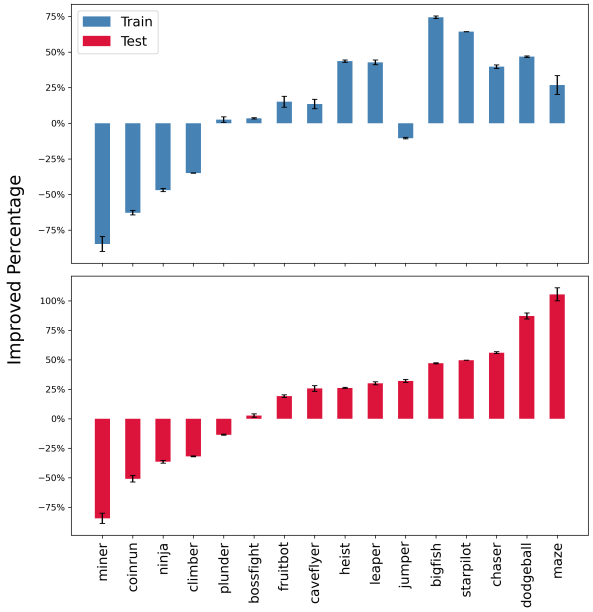

Figure 1: Improved percentage of Rainbow with unigram CSSC on 16 Procgen games in easy mode. Each bar represents mean and STD improved percentage from 3 independent seeds.

Table 1: Mean normalized score of PPO in OpenAI Procgen Easy Mode

Model	PPO			
Mode	Easy (16 env.)			
Regularization	No	unigram CSSC	UCB − DrAC	unigram CSSC + UCB − DrAC
Train	0.5678	0.5529	0.5594	0.5626
Test	0.3331	0.3506	0.3927	0.4085
Improve. (test)	0.0%	5.26%	17.92%	22.64%

REFERENCES

[1] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. 2020. Leveraging Procedural Generation to Benchmark Reinforcement Learning. arXiv:1912.01588 [cs.LG]

[2] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. 2018. Quantifying Generalization in Reinforcement Learning. arXiv:1812.02341 [cs.LG]

[3] Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G. Bellemare, and David Silver. 2020. The Value-Improvement Path: Towards Better Representations for Reinforcement Learning. arXiv:2006.02243 [cs.LG]

[4] Matthew Hausknecht and Peter Stone. 2015. The Impact of Determinism on Learning Atari 2600 Games. In *AAAI Workshop on Learning for General Competency in Video Games*. Austin, Texas, USA. http://www.cs.utexas.edu/users/ai-lab?hausknecht:aaai15

[5] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2017. Rainbow: Combining Improvements in Deep Reinforcement Learning. arXiv:1710.02298 [cs.AI]

[6] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. 2020. Reinforcement Learning with Augmented Data. arXiv:2004.14990 [cs.LG]

[7] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. 2019. Network Randomization: A Simple Technique for Generalization in Deep Reinforcement Learning. arXiv:1910.05396 [cs.LG]

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (February 2015), 529—533. https://doi.org/10.1038/nature14236

[9] Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. 2020. Automatic Data Augmentation for Generalization in Deep Reinforcement Learning. *arXiv preprint arXiv:2006.12862* (2020).

[10] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized Experience Replay. arXiv:1511.05952 [cs.LG]

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[12] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (October 2017), 354—359. https://doi.org/10.1038/nature24270

[13] Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. 2019. Observational Overfitting in Reinforcement Learning. arXiv:1912.02975 [cs.LG]

[14] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. 2018. DeepMind Control Suite. arXiv:1801.00690 [cs.AI]

[15] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. 2018. A Study on Overfitting in Deep Reinforcement Learning. arXiv:1804.06893 [cs.LG]