Improving Deep Reinforcement Learning via Transfer

Doctoral Consortium

Yunshu Du
Washington State University
Pullman, WA
yunshu.du@wsu.edu

ABSTRACT

While achieving the state-of-the-art performance in complex sequential tasks, deep reinforcement learning (deep RL) remains extremely data inefficient. Many approaches have been studied to improve the data efficiency of deep RL algorithms. This dissertation focuses on leveraging various transfer learning techniques to tackle this problem. We first show that positive transfer can be achieved cross-domains via direct weight transfer if the two agents share a certain amount of similarities. Then we look into how could the similarity between cross-domain tasks be quantified, such that we only transfer useful information from one task to another while blocking information that might have a negative effect. The third direction we studied is the human-agent transfer mechanism, which we integrate human knowledge via supervised pre-training on a set of demonstration data collected from a human then transfer to an agent. Lastly, several future directions are proposed for the remainder of this dissertation.

KEYWORDS

Reinforcement learning; Deep learning; Learning agent capabilities (agent models, communication, observation)

ACM Reference Format:
Yunshu Du. 2019. Improving Deep Reinforcement Learning via Transfer. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

In recent years, deep reinforcement learning (deep RL) has gained great attention due to its ability to learn directly from high-dimensional sensor data without needing hand-crafted features. Deep Q-network (DQN) [7] and asynchronous advantage actor-critic (A3C) [6] are the first two successful deep RL algorithms where convolutional neural networks (CNN) are used as function approximators for classic RL algorithms. Both algorithms achieved impressive results in playing 49 distinct Atari games and have become the benchmark in deep RL. The DQN algorithm combines Q-learning [11] with CNN. In Q-learning, the agent learns a *value function* $Q^{\pi}(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^{\pi}(s',a')|s,a]$ and deduce the optimal policy π^* by following actions that have the maximum Q value $Q^*(s,a) = max_{\pi} Q^{\pi}(s,a)$ at each state. In DQN, a three-layer CNN followed by two fully connected layers (parameterized as θ) are used to approximate $Q(s,a;\theta) \approx Q^*(s,a)$ since the state space of raw sensor inputs are too large to compute Q values directly.

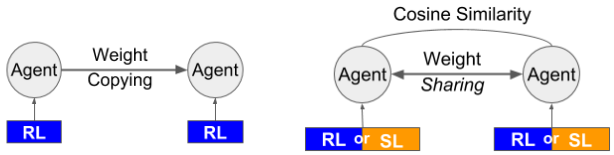
The A3C algorithm differs from DQN as it combines the actor-critic framework [9] with a CNN. A3C is a policy-based algorithm and maintains both a *policy function* (the *actor*) and a *value function* (*the critic*). Unlike DQN where only one agent is executed, A3C runs k actor-learners in parallel and each with their own copies of the environment and parameters. An update is performed using data collected from all actors.

Despite the achievements, deep RL remains extremely data inefficient. For example, both DQN and A3C algorithms need to consume millions of experiences before learning to act reasonably in a game. This thesis proposes to leverage *transfer learning (TL)* in various ways to make deep RL more efficient. Taylor and Stone [10] studied TL in the RL domain and discovered that knowledge acquired from well-trained *source tasks* could be transferred to *target tasks* to accelerate learning, under the assumption that the source and the target tasks share some degree of similarity (usually defined by a human). Following Taylor and Stone [10], three key steps need to be addressed to perform TL in deep RL: 1) how to select the appropriate source task for a given target task, 2) how to quantify the similarity between the source and the target, and 3) how to perform knowledge transfer effectively. This thesis aims to study how each step should be completed in the domain of deep RL.

2 CURRENT WORK

We first attempted to directly apply TL to improve learning speeds for the DQN algorithm in two domains: Atari games and Cart-Pole [4]. Following the three-step framework of TL, we first perform source/target selection based on intuitive task similarities. For example in Atari, we manually picked the game Breakout and Pong because they are visually similar. Since we hand-picked the tasks, the degree of task relatedness is considered as given by a human; thus we assume the second step of quantifying similarity has been implicitly fulfilled. Third, we selected the *pre-training* and *fine-tuning* methods from the deep supervised learning literature [12] as our transfer strategies. In particular, a source agent (e.g., Breakout) is first trained from scratch till convergence, and then the learned parameters are copied to a target agent (e.g., Pong) as its network initialization (instead of initializing randomly) and later fine-tuned in the target agent (Figure 1a). We also studied how transferring different layers of the network affects learning by performing layer-wise weight copying. Overall, our results show that if the source and the target are related, the more layers transferred the better the target agent performs.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.



(a) Cross-domain transfer between RL agents [4].

(b) Cross-domain transfer between RL or SL agents via cosine similarity measures [3].

(c) Same-domain transfer between SL pre-trained human demonstrations and RL agents [2].

(d) Same-domain transfer between SSL pre-trained human demonstrations and RL agents [Proposed method].

Figure 1: Overview of our current and planned work. RL: Reinforcement learning. SL: Supervised Learning. SSL: Semi-Supervised Learning.

Next, we studied how to quantify task similarity during transfer (i.e., the second step of the TL framework). Our recent work in [3][1] proposed a new algorithm to tackle the problem of negative transfer due to potentially inaccurate or even false similarity measures. We found that the cosine similarity between task gradients can be used as an elegant measure for quantifying task similarity, thus knowing when is one task helpful for another and for how long. We form a particular type of transfer learning: transferring knowledge of an *auxiliary task* (T_{aux}) to a *main task* (T_{main}) where only the performance of T_{main} is of interest, even though they are trained simultaneously. The two tasks share a subset of parameters θ and also have their own parameters ϕ_{main} and ϕ_{aux} respectively. We devised an algorithm that can automatically 1) leverage T_{aux} when it is helpful to T_{main}, and 2) block negative transfer when T_{aux} hinders T_{main}. Our objective minimizes L_{main} at each time step t

$$\underset{\lambda^t}{\text{argmin}}\, L_{main}\Big(\theta^t - \alpha\nabla_\theta(L_{main} + \lambda^t L_{aux}), \phi^t_{main} - \alpha\nabla_{\phi_{main}} L_{main}\Big)$$

where $\lambda^t = (sign(cos(\nabla_\theta L_{main}, \nabla_\theta L_{aux})) + 1)/2$ is an adaptive weight based on the cosine similarity between the gradients of L_{main} and L_{aux}. Intuitively, when the gradients of both tasks are pointing at similar directions (i.e., cosine similarity is non-negative), we leverage L_{aux} to minimize L_{main}; when the two tasks disagree (i.e., cosine similarity is negative), we ignore L_{aux} and minimize L_{main} alone. Despite its simplicity, our algorithm showed empirical success in detecting and blocking potential negative transfer in various domains: deep supervised learning on subsets of ImageNet, RL on gridworlds, and deep RL on Atari games (Figure 1b).

Thus far we have studied cross-domain agent-to-agent transfer (Figure 1a and 1b). However, positive transfer is hard to guarantee when a domain shift presents. Thus, we consider a different mechanism that integrates human demonstrations as the source and performs human to agent transfer within the same domain; the complication of cross-domain task selection and similarity measurement can be avoided. Our latest work [2] studied leveraging *non-expert* human demonstrations to improve the A3C algorithm in the Atari domain (Figure 1c). Unlike the *learning from demonstration* literature (e.g., [1, 5]) which assume *expert* demonstrations are available, our method does not make this assumption. This makes our method of higher practical utility.

The first step of our method is to ask a non-expert human player to play a game for less than 20 minutes and stored all state-action observations. Then, we performed supervised pre-training on the collected demonstrations using the same network architecture as in A3C. We assume that the action demonstrated by the human is the ground truth label for a given state. The classifier learns a mapping between the state and the action which can be viewed as a feature learner that captures important regions of the game. After pre-training, we performed weight copying and transferred parameters of the classifier to an A3C agent. As expected, agents initialized with human knowledge outperformed baseline agents in all six Atari games tested.

The most significant contribution of this work is that we provided the first empirical analysis of what features are learned from supervised pre-training and why pre-training on human demonstration helps. In particular, we proposed a visualization method modified from the Gradient-weighted Class Activation Mapping (Grad-CAM) [8] and using which we were able to observe similarities between features learned during pre-training versus that of an RL agent, indicating why pre-training could be helpful. For example in the game of Breakout, after pre-training the network learned to pay attention to the paddle since it is associated with the action; a converged A3C agent also pays attention to similar regions around the paddle and also learned to track the movement of the ball—knowing where the paddle is indeed was a useful prior for the agent. Visualization results are available at https://sites.google.com/view/pretrain-deeprl.

3 FUTURE WORK

We are interested in several directions in this dissertation. First, we have an immediate research plan to look into other approaches to leverage human knowledge. While we have shown that a small amount of noisy demonstration data is sufficient for improving deep RL, a larger amount of better quality data could be more helpful. Suppose we do not add a further burden to the human demonstrator (e.g., increase demonstration time), we want the agent to self-generate extra useful information. For this purpose, we propose to leverage *semi-supervised learning* during the pre-training stage (Figure 1d). In addition to the collected human demonstrations, we can execute an arbitrary policy to randomly explore the environment and save all observations. The demo data can be viewed as the labeled data while the agent-generated observations are the unlabeled data—this resembles the setting of a semi-supervised learning problem, and existing techniques can be applied.

Another direction we will explore is to combine pre-training with the cosine similarity measure. For example, an agent might want to avoid negative transfer from a lousy demonstration. We might be able to measure the gradient cosine similarity between the demonstration and the agent such that we disregard demonstrations that are too distinct from the optimization goal of the agent.

[1] Work done during an internship at DeepMind

REFERENCES

[1] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A Survey of Robot Learning from Demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.

[2] Gabriel V. de la Cruz, Jr., Yunshu Du, and Matthew E Taylor. 2018. Pre-training with Non-expert Human Demonstration for Deep Reinforcement Learning. *arXiv preprint arXiv:1812.08904* (2018).

[3] Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan. 2018. Adapting Auxiliary Losses using Gradient Similarity. In *Proceedings of the Continual Learning Workshop at NeurIPS 2018*. Montreal, Canada.

[4] Yunshu Du, Gabriel V. de la Cruz, Jr., James Irwin, and Matthew E. Taylor. 2016. Initial Progress in Transfer for Deep Reinforcement Learning Algorithms. In *Proceedings of the Deep Reinforcement Learning: Frontiers and Challenges Workshop at IJCAI 2016*. New York City, NY, USA.

[5] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. 2018. Deep Q-learning from Demonstrations. In *The Thirty-second AAAI Conference on Artificial Intelligence (AAAI)*.

[6] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *The Thirty-third International Conference on Machine Learning (ICML)*. 1928–1937.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level Control Through Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529.

[8] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual Explanations from Deep Networks via Gradient-based Localization. In *IEEE-International Conference on Computer Vision (ICCV)*. IEEE, 618–626.

[9] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction*. MIT press.

[10] Matthew E Taylor and Peter Stone. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research* 10, Jul (2009), 1633–1685.

[11] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[12] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable are Features in Deep Neural Networks?. In *The Twenty-eighth Conference on Neural Information Processing Systems (NeurIPS)*. 3320–3328.