Attention-based Deep Reinforcement Learning for Multi-view Environments

Extended Abstract

Elaheh Barati
Wayne State University
Detroit, MI
elaheh.barati@wayne.edu

Xuewen Chen
AIWAYS AUTO
Shanghai, China
xuewen.chen@ai-ways.com

Zichun Zhong
Wayne State University
Detroit, MI
zichunzhong@wayne.edu

ABSTRACT

In reinforcement learning algorithms, it is a common practice to account for only a single view of the environment to make the desired decisions; however, utilizing multiple views of the environment can help to promote the learning of complicated policies. Since the views may frequently suffer from partial observability, their provided observation can have different levels of importance. In this paper, we present a novel attention-based deep reinforcement learning method in a multi-view environment in which each view can provide various representative information about the environment. Specifically, our method learns a policy to dynamically attend to views of the environment based on their importance in the decision-making process. We evaluate the performance of our method on TORCS racing car simulator and three other complex 3D environments with obstacles.

KEYWORDS

Reinforcement learning; Deep learning; Attention networks

ACM Reference Format:
Elaheh Barati, Xuewen Chen, and Zichun Zhong. 2019. Attention-based Deep Reinforcement Learning for Multi-view Environments. 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

Distributed reinforcement learning algorithms [2, 4] have been proposed to improve the performance of the learning algorithm by passing copies of the environment to multiple workers. The adoption of multiple workers in these works is rather to increase the training reward and earlier convergence, not to collectively increase the amount of observable information from the environment through multiple sensory inputs.

In a realistic environment, observability can be typically partial on account of occlusion from the obstacles or noise that affect the sensors such as cameras in the environment. Utilizing only one camera view can result in failure, since locating a single camera in a position that can capture both the targets as well as details of agents body is difficult [7]. On the other hands, sensory inputs with less importance can sometimes provide observations which are vital for achieving rich behavior. Therefore, it is desirable to incorporate multiple sensory inputs in the decision-making process according to their importance and their provided information at each time step. It reduces the sensitivity of policies to an individual sensor and makes the system capable of functioning despite one or more sensors malfunctioning. Since sensors can provide diverse views of the environment and they are likely to be perturbed by different noise impacts, a policy is required to attend to the views accordingly.

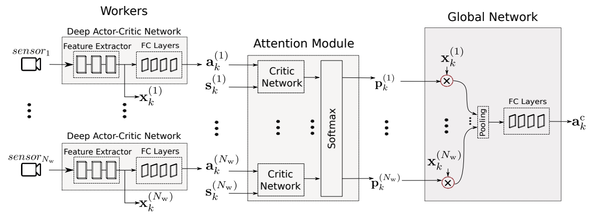


Figure 1: Architecture of the deep network that leverages attention mechanism in its global network. $\mathrm{p}_k^{(w)}$ is the weight of worker w obtained from the attention module according to the importance of its view.

In this paper, we propose an attention-based deep reinforcement learning method (depicted in Figure 1) that learns a policy to attend to different views of the environment based on their importance. Each sensory input, which provides a specific view of the environment, is assigned to a worker. We employ an extension of the actor-critic approach [6] to train the network of each worker and make the final decision through the integration of feature representations provided by the workers using an attention mechanism. Since the critic network of each worker provides a signal regarding the amount of salient information supplied by its corresponding view, we employ this signal in the attention module to estimate the amount of impact that each of the views should have in the final decision-making process.

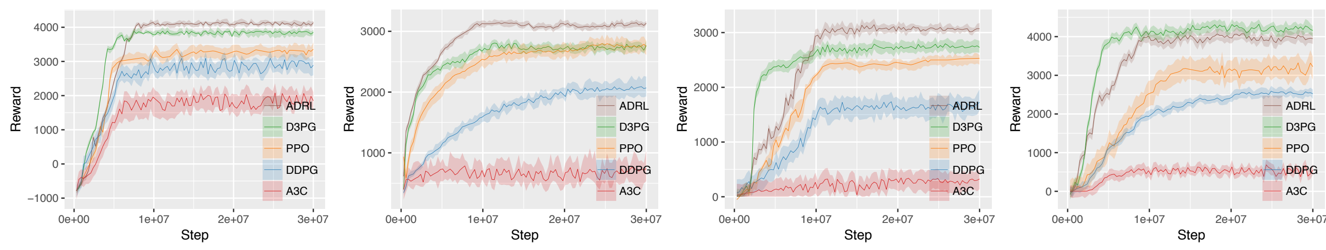
2 ATTENTION-BASED RL FRAMEWORK

By utilizing an attention weighted representation as introduced in [1], we incorporate the importance of the views in computing a unit representation of the environment ($\mathbf{x}_k$). We use a softmax gate function to learn the attention module as

$$\mathbf{x}_k = \sum_{w=1}^{N_w} \mathbf{p}_k^{(w)} \odot \mathbf{x}_k^{(w)} = \sum_{w=1}^{N_w} \frac{\exp(g_w f_w)}{\sum_l \exp(g_l f_l)} \odot \mathbf{x}_k^{(w)}\,, \qquad (1)$$

where $\odot$ stands for Hadamard product, $\mathbf{x}_k^{(w)}$ is the feature representation obtained by worker w, and g_w is a parameter of the model.

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) TORCS (b) Ant-Maze (c) Hopper-Stairs (d) Walker-Wall

Figure 2: Average reward vs. training step for the methods DDPG, D3PG, PPO, A3C, and ADRL. We obtain the rewards in these figures by averaging rewards obtained from 5 runs.

In (1), f_w is obtained from the output of critic network learned for each worker w separately, i.e., $f_w = Q(s^{(w)}, a^{(w)})$. At each time step k, the attention mechanism generates a positive weight $\mathbf{p}_k^{(w)}$ for each worker which determines the relative importance of worker w in blending the feature vectors $\{\mathbf{x}_k^{(w)} | w = 1, 2, \ldots, N_\mathrm{w}\}$.

During training of our model, the aim is to promote the behavior of each worker by comparing its selected action with the actions selected by all the other workers. To do so, we modify the reward at training of the global network by introducing a penalty term that depends on the actions selected by all the workers (A_k) as:

$$r_k^\mathrm{c} = r_k - \gamma_\mathrm{r}\delta(A_k) = r_k - \gamma_\mathrm{r}\frac{1}{N_\mathrm{w}}\sum_{w=1}^{N_\mathrm{w}} \delta^{(w)}(A_k) \tag{2}$$

where r_k is the original reward, γ_r is a constant that provides a trade-off between the deviation of actions and the original reward, and A_k is the action matrix with columns $\mathbf{a}_k^{(w)}$. In (2), we define

$$\delta^{(w)}(A_k) = \left\| \mathbf{a}_k^{(w)} - \frac{1}{N_\mathrm{w} - 1}\sum_{\substack{v=1 \\ v \neq w}}^{N_\mathrm{w}} \mathbf{a}_k^{(v)} \right\|^2 \tag{3}$$

as a deviation function that depends on the variation of action $\mathbf{a}_k^{(w)}$ of worker w from the average of actions selected by the other workers in the network. In (3), the first term is the action of worker w while the second term is the average action of other workers. In the case that all workers have trained sufficiently, the deviation $\delta(A_k)$ becomes close to zero, while by experiencing weak training performance from a worker, we get a higher $\delta(A_k)$ and a lower reward r_k^c. Therefore, the penalty term in (2) enforces improvement in the training of the workers that are yet to be trained sufficiently. We consider γ_r in (2) to be 0.1 in our experiments.

The final action ($\mathbf{a}_k^\mathrm{c}$) is determined by the global network from an aggregation of feature representations ($\mathbf{s}_k^\mathrm{c} = \mathbf{x}_k$) obtained from the workers. To train the parameters of global network ($\theta_k^{\mu^\mathrm{c}}, \theta_k^{Q^\mathrm{c}}$), at each time step k, we stack the modified reward r_k^c on the replay buffer, forming a tuple $<\mathbf{s}_k^\mathrm{c}, \mathbf{a}_k^\mathrm{c}, r_k^\mathrm{c}, \mathbf{s}_{k+1}^\mathrm{c}>$. Then, we sample a random mini-batch from the replay buffer to train critic network ($\theta_k^{Q^\mathrm{c}}$) by minimizing the following loss function [3]:

$$\mathcal{L}(\theta_k^{Q^\mathrm{c}}) = \mathbb{E}\left(r_k^\mathrm{c} + \gamma\overline{Q}(\mathbf{s}_{k+1}^\mathrm{c}, \overline{\mu}^\mathrm{c}(\mathbf{s}_{k+1}^\mathrm{c}; \theta^{\overline{\mu}^\mathrm{c}}); \theta^{\overline{Q}^\mathrm{c}}) - Q(\mathbf{s}_k^\mathrm{c}, \mathbf{a}_k^\mathrm{c}; \theta_k^{Q^\mathrm{c}})\right)^2 .$$

We update the actor network at each time step with respect to the set of parameters $\theta_k^{\mu^\mathrm{c}}$ by using sampled policy gradient:

$$\nabla_{\theta_k^{\mu^\mathrm{c}}} J = \mathbb{E}\left(\nabla_\mathbf{a} Q(\mathbf{s}, \mathbf{a}; \theta_k^{Q^\mathrm{c}})|_{\mathbf{s}=\mathbf{s}_k^\mathrm{c}, \mathbf{a}=\mu^\mathrm{c}(\mathbf{s}_k^\mathrm{c})} \nabla_{\theta_\mu^\mathrm{c}} \mu^\mathrm{c}(\mathbf{s}; \theta_k^{\mu^\mathrm{c}})|_{\mathbf{s}=\mathbf{s}_k^\mathrm{c}}\right) . \tag{4}$$

3 EXPERIMENTS

We compare the performance of our method with the baselines on tasks, Ant-Maze, Hopper-Stairs, and Walker-Wall developed in the MuJoCo physics simulator [8]. We also verify our method for autonomous driving on an open-source platform for car racing called TORCS [9]. The baselines D3PG [2], PPO [5], DDPG [3], and A3C [4] are state-of-the-art actor-critic based methods that are designed to work on continuous action spaces. Figure 2 shows the training speed of our method, Attention-based Deep RL (ADRL), and four of its baselines in terms of average reward per step. At the first stage in training of ADRL, multiple workers are trained separately given multiple views of the environment, and the reward of ADRL is the average reward of all these workers. Since D3PG uses all its workers to train the network given a single view of the environment, at the initial steps of training, the reward of ADRL is less than D3PG for all the tasks other than Ant-Maze. However, after convergence, the reward of ADRL is either higher or comparable to D3PG in all the tasks which is an indication of the advantage of attending to multiple various views of the environment. In Ant-Maze task, different camera views provide significantly diverse information about the environment, and leveraging these views via the attention mechanism in ADRL leads to a significantly higher reward for ADRL in comparison to its baselines. In Walker-Wall task, the reward of ADRL is slightly less than D3PG which is due to having a low diversity between the camera views provided from the environment.

4 CONCLUSION

Our method, ADRL, takes advantage of multiple views of the environment to obtain a stabilized training policy. ADRL dynamically attends to views according to their importance in the final decision-making process. To measure the importance of each view, ADRL uses the output of the critic network designated for that view. Through the experiments, we observed that ADRL outperforms its baselines.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR-15)*.

[2] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. 2018. Distributed Distributional Deterministic Policy Gradients. In *Proceedings of International Conference on Learning Representations (ICLR-18)*.

[3] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. In *Proceedings of International Conference on Learning Representations (ICLR-15)*.

[4] 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 *International Conference on Machine Learning (ICML-16)*. 1928–1937.

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

[6] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *Proceedings of International Conference on Machine Learning (ICML-14)*. 387–395.

[7] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. 2018. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690* (2018).

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

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