Optimising Worlds to Evaluate and Influence Reinforcement Learning Agents

Extended Abstract

Richard Everett, Adam Cobb, Andrew Markham, Stephen Roberts
University of Oxford
Oxford, United Kingdom
{richard,acobb,sjrob}@robots.ox.ac.uk,andrew.markham@cs.ox.ac.uk

ABSTRACT

Training reinforcement learning agents on a distribution of procedurally generated environments has become an increasingly common method for obtaining more generalisable agents. However, this makes evaluation challenging, as the space of possible environment settings is large; simply looking at the average performance is insufficient for understanding how well - or how poorly - the agents perform. To address this, we introduce a method for strategically evaluating and influencing the behaviour of reinforcement learning agents. Using deep generative modelling to encode the environment, we propose a World Agent which efficiently generates and optimises worlds (i.e. environment settings) relative to the performance of the agents. Through the use of our method on two distinct environments, we demonstrate the existence of worlds which minimise and maximise agent reward beyond the typically reported average reward. Additionally, we show how our method can also be used to modify the distribution of worlds that agents train on, influencing their emergent behaviour to be more desirable.

KEYWORDS

Reinforcement Learning; Agent Simulation; Evaluation; Training

ACM Reference Format:
Richard Everett, Adam Cobb, Andrew Markham, Stephen Roberts. 2019. Optimising Worlds to Evaluate and Influence Reinforcement Learning Agents. 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 GENERATIVE WORLD AGENT

We propose the introduction of a generative *World Agent* into the training and evaluation process of reinforcement learning agents. By encoding the environment using a deep generative model and then searching in the model's latent space, our World Agent is able to efficiently adapt the distribution of worlds based on the performance of the reinforcement learning agents.

To help explain our method, we separate it into three phases:

(1) **Generate Worlds:** Sample worlds using our trained VAE.
(2) **Train Agents:** Train reinforcement learning agent(s) on the sampled set of generated worlds.
(3) **Optimise Worlds:** Iteratively generate and optimise worlds to maximise a given agent-based metric.

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.

Figure 1: Optimised worlds which minimise and maximise agent reward, alongside typically reported average reward.

1.1 Generating Worlds

Rather than individually optimising every aspect of a world, for example at an individual pixel level, we use a VAE [4] to compress the complex distribution over the world space $\mathbf{W}$ into a tractable distribution over the latent space $\mathbf{z}$. In Equation 1 we show how a world $\mathbf{w}_i$ can be sampled from the latent space by passing a sample $\mathbf{z}_i$ to the generator $\mathbf{G}$, where $\boldsymbol{\theta}_g$ are the generator's weights:

$$\mathbf{w}_i = \mathrm{G}(\mathbf{z}_i; \boldsymbol{\theta}_g), \qquad \mathbf{z}_i \sim q(\mathbf{z} \mid \boldsymbol{\theta}_e) \tag{1}$$

To learn $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_e$, we train our VAE on a dataset of worlds created by a handcrafted procedural generator for each environment.

1.2 Training Agents

The second phase is the training of the reinforcement learning agents which interact in the worlds produced by our World Agent.

For a given world $\mathbf{w}_i$ and its latent representation $\mathbf{z}_i$, the behaviour of the agents is summarised in their corresponding set of trajectories $\mathcal{T}_i$.

1.3 Optimising Worlds

To search the space of worlds, we use an optimiser to sample from the latent space of the generator. Samples are selected with the goal of maximising the World Agent's objective function (i.e. metric $\mathcal{M}$), where the optimisation task: $\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmax}}\, \mathcal{M}\left(\mathrm{G}(\mathbf{z}; \boldsymbol{\theta}_g), \mathcal{T}(\mathbf{z})\right)$, is over the latent space. This objective function depends on the behaviour of the agents (i.e. trajectories $\mathcal{T}$) and the generated worlds $\{\mathrm{G}(\mathbf{z}_i; \boldsymbol{\theta}_g)\}_{i=1}^{N}$, both of which are functions of the latent space.

We use two different optimisers: (1) *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) [2] and (2) *Bayesian Optimisation* (BO) [7]. We also consider a number of different metrics $\mathcal{M}$ such as minimising/maximising the agent's reward, minimising the agent's crash rate, and maximising the equality of a group of agents.

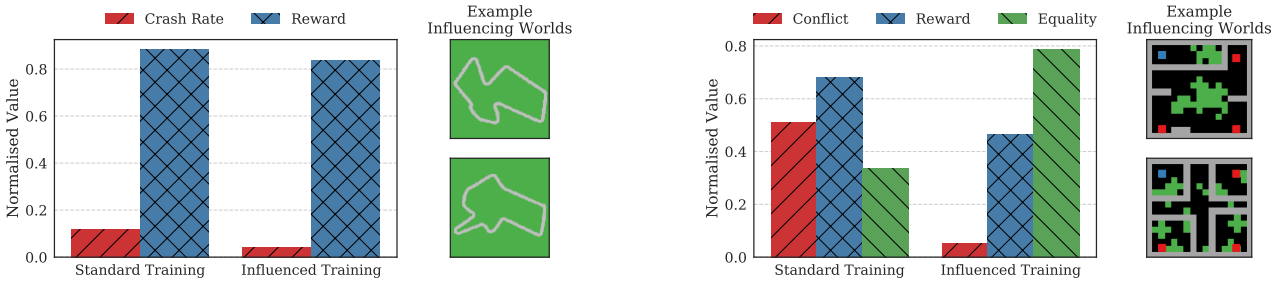

Figure 2: Comparison between training methods for (left) Particle Racing and (right) Resource Harvest. Default Training is where agents are trained on randomly sampled worlds. Influenced Training is where agents are trained on our modified training distribution. We also include two example worlds which are included in the new training distribution. Additional Metrics: *Crash Rate* is the probability of an agent crashing in an episode, *Conflict* is the average number of steps in an episode where agents are tagged out, and *Equality* is the distribution of rewards across agents calculated as 1-Gini Coefficient [6].

2 EXPERIMENTS

2.1 Environments

Particle Racing. A single-agent particle racing game based on the OpenAI Gym Car Racing environment [1]. The agent's objective is to complete loops of the track as quickly as possible, with the episode terminating early if the agent leaves the track (i.e. crashes).

Resource Harvest. A multi-agent resource gathering game based on related work [3, 5, 6]. In this environment, four self-interested agents (shown in red) are individually rewarded for harvesting resources (shown in green). Therefore, they are motivated to harvest the resources as fast as possible before the other agents are able to do so. Notably, however, resources recover based on the amount of nearby resources, meaning it is beneficial in the long run to leave several untouched so that the harvested resources recover faster.

2.2 Evaluating Agents

In this experiment, we use our method to evaluate agents by *efficiently* finding worlds where they perform worst (minimum reward) and perform best (maximum reward).

For Particle Racing (Figure 1, top), our method finds a rare world which consistently causes the agent to crash, resulting in a minimum reward of 0.13. Notably, the environment has a high number of sharp and unexpected corners. In contrast, the world the agent performs best on – obtaining a reward of 0.99 – has no surprising corners, and is instead a simple rectangle-like shape.

For Resource Harvest (Figure 1, bottom), our method finds a spatial arrangement of resources and walls such that the reward of the agents is heavily diminished - from the average of 0.61 down to a minimum of 0.05 (a 91.2% reduction). This was achieved by spreading out the resources so that they do not gain the recovery bonus from nearby resources. Conversely, reward is maximised (0.92, up from the average of 0.61) by removing walls and grouping resources so that they all benefit from the recovery bonus and therefore recover as quickly as possible.

2.3 Influencing Agents

In our next set of experiments, we use our method to adjust the training distribution of agents by sampling worlds which maximise a given agent-related property. Concretely, we create a new training distribution consisting only of sampled worlds which have the property we desire and then train new agents on this distribution. We refer to this process as *influencing* agents as our method influences the learned emergent behaviour of the agents.

Influenced agents are safer. In Particle Racing, we observe that there exists many possible worlds which causes agents to crash. To this end, we sample worlds where agents are likely to crash (i.e. high crash rate, > 0.5) to form the new training distribution. As shown in Figure 2 (left), these worlds tend to be challenging with at least one sharp corner.

In Figure 2 (left) we present the results of our influenced training regime. Notably, crash rate is significantly reduced, dropping from 0.11 to 0.04. This comes at the cost of a 5% reduction in reward due to the agent driving slower and therefore taking longer to complete the track. In the context of safety, our re-trained agent's behaviour is more desirable as it crashes less often.

Influenced agents are fairer. In Resource Harvest, we consider the situation where one agent in a multi-agent system has more power than the others. Specifically, we allow this agent to perform the TAG action which fires a beam that temporarily removes any agent hit from the episode for a number of timesteps.

Typically in this situation, the more powerful agent would frequently tag other agents (high conflict), reducing competition and therefore privatising the resources for itself (high inequality).

To counter this undesirable behaviour, we sample worlds which minimise conflict to form the new training distribution. Specifically, we sample worlds where no agents are tagged out in an entire episode and then train fresh agents on this new distribution. As shown in Figure 2 (right), these low-conflict worlds typically isolate the tagging agent (shown as blue in the top left of the world) from the other agents through the use of walls.

As can be seen in Figure 2 (right), our influenced training regime results in significantly lower conflict (0.51 to 0.05) and higher equality (0.33 to 0.78). This arises as the tagging agent never learns to associate its aggressive TAG action with increased reward, therefore reducing the probability of its use. However, this leads to reduced group reward as the resources are less likely to be privatised.

REFERENCES

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[2] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.

[3] Edward Hughes, Joel Z Leibo, Matthew G Philips, Karl Tuyls, Edgar A Duéñez-Guzmán, Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin R McKee, Raphael Koster, et al. 2018. Inequity aversion resolves intertemporal social dilemmas. *arXiv preprint arXiv:1803.08884* (2018).

[4] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[5] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 464–473.

[6] Julien Perolat, Joel Z Leibo, Vinicius Zambaldi, Charles Beattie, Karl Tuyls, and Thore Graepel. 2017. A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems*. 3646–3655.

[7] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.