Motivating Exploration in Reinforcement Learning

Bob Wei
Department of Computer Science
University of Waterloo
q25wei@edu.uwaterloo.ca

Akshay Patel
Department of Computer Science
University of Waterloo
akshay.patel@uwaterloo.ca

Samir Alazzam
Department of Computer Science
University of Waterloo
q25wei@edu.uwaterloo.ca

Abstract

Self-motivation and curiosity come second nature to humans, both of which are crucial in quickly adapting to new environments and achieving distant goals. Using this as inspiration for artificial agents trained with reinforcement learning, we present an analysis of different techniques used by the current state-of-the-art to encourage exploration in varying environments. Specifically, we focus on formulations for entropy and curiosity, which promote exploration via randomness and state-based novelty respectively. We show that such methods are invariant to task/environment specifics, such as the reward density, and we also demonstrate that curious learners train faster based on extrinsic rewards and game visualizations.

1 Introduction

Exploration is something that comes naturally to humans, almost as if there exists an innate sense of curiosity, craving to experience something new, something yet to be seen. Through observing human behaviour, it is clear that the ability to explore a domain is critical to the process of discovery and learning, regardless of the task being considered.

And yet, even with the massive computational resources available today, the most impressive artificial learning agents struggle tremendously in balancing explicit exploration of the environment and stable convergence towards a useful, learned policy. Without adequate exploration and randomness, the agent can easily become stuck and converge towards a flawed policy early on in learning; the agent's policy reflects high certainty on its understanding of the environment when, in reality, it has yet to explore the majority of the state space. On the other hand, Excessive exploration can lead to unstable rewards and updates throughout learning which are detrimental to learning a useful policy. To make matters worse, there is a huge breadth of algorithms and hand-crafted techniques used in reinforcement learning, many of them requiring method or task specific hyperparameter tuning. This in turn leads to very specialized formulations across the field for encouraging exploration.

In this work, we explore a general formulation of curiosity that aims to motivate exploration in reinforcement learning (RL) agents while remaining invariant to the algorithm and task in question. We explore the effects of including an entropy term (of the policy action probability distribution) and a curiosity module based on that of [1], which provides an intrinsic reward signal. Due to the scope of the porject, we focus specifically on the Advantage Actor-Critic (A2C) RL algorithm in the *Pong*, *Seaquest*, and *Breakout* Atari environments. These tasks were chosen due to the differences in mechanics and their respective state spaces, specifically the density and magnitude of the extrinsic reward (i.e. reward received from environment itself).

We compare the performance of the agent in the various environments with and without the mentioned exploration factors. **Add more on the experimental results once that's finalized**

2 Related Work

Our artificial agent is a model-free, on-policy reinforcement learning algorithm that leverages Advantage Actor Critic (A2C) methods, entropy-based exploration methods, and curiosity-driven exploration methods. Our artificial agent uses combination(s) of these methods to learn how to navigate different OpenAI Gym [2] environments to maximize cumulative rewards.

Actor Critic methods have been successfully applied to a variety of reinforcement learning problems, with Advantage Actor Critic methods also being used after their introduction in 2016 through Schulman et al.'s work, High Dimensional Continuous Control Using Generalized Advantage Estimation, done at the University of California, Berkeley. Schulman et al.'s work presents the Advantage Actor Critic method that our artificial agent primarily leverages to learn how to maximize cumulative rewards in a variety of environments [3]. Schulman et al. address two main challenges faced by policy gradient methods in reinforcement learning, namely the large number of samples usually required and the difficulty of gaining consistent improvement with nonstationarity of the data [3].

Curiosity-driven exploration enhancements to A2C methods have also recently been applied to a set of reinforcement learning algorithms after their introduction in 2017 through Pathak et al.'s work, Curiosity-driven Exploration by Self-supervised Prediction, which was also done at the University of California, Berkeley. Pathak et al.'s work presents the curiosity-driven exploration method that our artificial agent uses, namely the Intrinsic Curiosity Module (ICM). Pathak et al.'s work addresses a challenge faced in learning when extrinsic rewards for the agent are extremely sparse through curiosity as the error in an agent's ability to to predict the consequence of its own actions [1]. Their proposed ICM notably ignores the aspects of the environment that do not affect the agent, as well as being able to scale to high-dimensional continuous state spaces such as images, and bypasses the difficulties of directly predicting pixels [1].

Attention based enhancements to deep learning algorithms have also been used to solve reinforcement learning problems more efficiently. Reizinger et al.'s work, Attention-Based Curiosity-Driven Exploration In Deep Reinforcement Learning, done at the Budapest University of Technology and Economics, introduces an attention-based enhancement to A2C [4].

3 Methods

In this section, we present our baseline RL framework using the A2C algorithm which is based on the previous works of [3]. We then describe the formulations of entropy and curiosity based learning factors to encourage environment exploration.

3.1 Advantage Actor-Critic (A2C)

Our baseline is built around A2C, which is an on-policy learning algorithm. The core decision making of the agent stems from the policy network (π), which is also referred to as the *Actor*. The policy network is learned and its weights (θ_π) are updated via the standard policy gradient equation, wherein the rewards r_t for a trajectory t are weighted by the negative log likelihood of that trajectory. Minimizing this loss $\mathcal{L}_P$ is equivalent to updating π such that the probability of high reward trajectories are maximized.

$$\mathcal{L}_P = \sum_t -\log \pi(s_t; \theta_\pi) \times r_t \tag{1}$$

A2C also uses the value network (V) or *critic* which predicts the accumulated, discounted rewards R_i over the episode timesteps i. The advantages can be computed as $A_i = R_i - V_i$, where V_i are the predictions from V, and are used in place of the actual rewards in the policy update 1, reducing the variance of $\mathcal{L}_P$, which is a common downfall of on-policy methods. We train the value network V with update equation 2.

$$\mathcal{L}_V = \frac{1}{n} \sum_i (R_i - V_i)^2 \tag{2}$$

The base A2C algorithm does not explicitly seek random exploration as it learns to decide on the action at a https://arxiv.org/pdf/1601.06733.pdf state purely from the policy update. The most commonly used method to introduce randomness to the agent is through maximizing the entropy of the next-action sampling distribution. This distribution is categorical with probabilities defined by the policy network logits.

$$\mathcal{L}_E = \sum_a (-\log \pi_a \times \mu_\pi) \tag{3}$$

Instead of minizing the loss function in 3, we maximize the value so that the individual probabilities of the sampling distribution π_a are as similar as possible; thus, inducing a degree of randomness in the policy network's state to action mapping.

3.2 Intrinsic Curiosity

Entropy encourages the agent to explore but in a purely random manner, which has been shown to produce exceptional results (refer to experiments). However, we also compare with a more targeted approach by leveraging the Intrinsic Curiosity Module (ICM) introduced in [1], which provides an intrinsic reward as a learning signal. ICM consists of two linear networks referred to as the forward and inverse models. The forward model f_{Fwd} is trained to predict the next state ($\phi(s_{t+1})$) given the current state ($\phi(s_t)$) and the action (a_t) decided by the policy net. Note that we use ϕ to denote a feature extractor network used to encode the raw states.

$$r_i = \sum_t (\phi(s_{t+1}) - f_{Fwd}(\phi(s_t); a_t))^2 \tag{4}$$

$$r = r_e + \beta r_i \tag{5}$$

The new intrinsic reward r_i is defined as the mean-squared-error of the predicted next state and the actual next state, which is then combined with the extrinsic reward to obtain the total reward used in the policy update 1. Thus, our agent learns to maximize the likelihood of following trajectories that maximize the intrinsic reward, which means that the agent is encouraged to enter new, unseen states that f_{Fwd} is unable to predict accurately. The forward model itself is trained using the loss below.

$$\mathcal{L}_{Fwd} = \frac{1}{n} \sum_t (\phi(s_{t+1}) - f_{Fwd}(\phi(s_t); a_t))^2 \tag{6}$$

Further, the purpose of the inverse model f_{Inv} is to train the feature extractor ϕ to encode observed features from the raw state that are most impacted by the agent's actions. Specifically, the inverse model takes the current and next states and infers the corresponding action taken: $f_{Inv}(\phi(s_t), \phi(s_{t+1})) = \hat{p}$, where $\hat{p}$ is a vector of probabilities for each possible action. This is trained with the following,

$$\mathcal{L}_{Inv} = -\log\left(\frac{e^{\hat{p}[a_t]}}{\sum_j e^{\hat{p}[j]}}\right) = -\hat{p}[a_t] + \log\left(\sum_j e^{\hat{p}[j]}\right) \tag{7}$$

The described curiosity model is incorporated into the base A2C model and can be trained jointly:

$$\mathcal{L} = \mathcal{L}_P + \lambda_V \mathcal{L}_V - \lambda_E \mathcal{L}_E + \mathcal{L}_{Fwd} + \mathcal{L}_{Inv} \tag{8}$$

where we use $\lambda_V = 0.5$ and $\lambda_E = 0.02$.

4 Experiments

Below, we provide details on our implementation and setup for training the agent. We also include plots of the smoothed, extrinsic rewards from full episodes throughout the training, where we compare the effectiveness of motivating random or targeted exploration. Finally, we show some qualitative results across various environments that demonstrate interesting agent behaviour.

4.1 Implementation Details

Our work is implemented in Pytorch and we iterate our agents in the Atari environments provided through OpenAI Gym [5, 2], namely `PongNoFrameskip-v0`, `SeaquestNoFrameskip-v0`, and

`BreakoutNoFrameskip-v0` (using consistent random seeds for each of the environments). Specifically, we train using 4 parallel environments at a time and with each state consisting of the current frame concatenated with 3 previous frames. We use the Adam optimizer [6] with a learning rate of 0.0001, a rollout size of 5, and a varying number of update steps (due to the limited scope of the project). All of our models are trained synchronously on Nvidia GTX 1080Ti (11GB VRAM) and Tesla V100 (16GB VRAM) gpus with batch size equivalent to the number of parallel environments.

The main actor-critic network consists of a feature extractor, followed by the actor and critic headers, which are simply single linear layers that map the extracted features to the desired next-action logits and predicted discounted rewards respectively. The feature extractor consists of four stride-2 convolutional layers followed by a LSTM layer [7], which is needed to leverage past information. We note that the LSTM memory is cleared at the end of every episode. The encoded features are flattened to $B \times 288$ tensors before being forwarded to the actor-critic. The initial state is resized to a 48×48 image before being passed to the feature extractor.

The curiosity module being used consists of a forward and inverse model as well as another feature extractor net (this one being purely convolutional). Both forward and inverse models are simple multi-layer perceptrons comprised of two stacked linear layers with 288 hidden neurons.

4.2 Quantitative Results

Here, we show our training results in the three environments: `Pong`, `Seaquest`, and `Breakout`. Specifically, we plot the extrinsic reward (objective measure of agent performance) as training progresses for three different variants of A2C:

- **A2C**: baseline A2C algorithm without entropy term
- **A2C + Entropy**: A2C with entropy term to encourage randomized exploration
- **A2C + Entropy + Curiosity**: A2C with ICM and entropy to encourage exploration of novel states

4.2.1 Sparse Extrinsic Reward

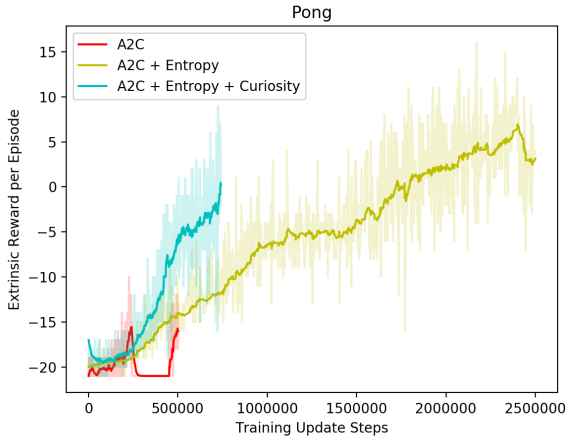

Figure 1: Sample figure caption.

4.2.2 Dense Extrinsic Reward

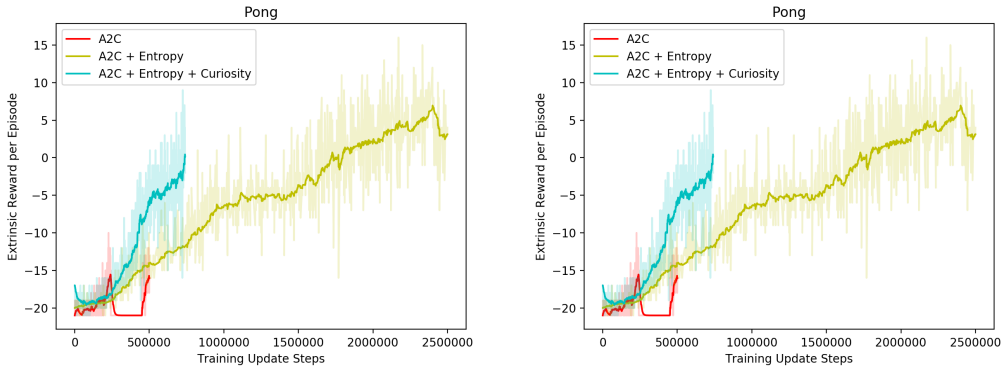

Figure 2: A figure with two subfigures

4.3 Qualitative Results

5 Discussion

References

[1] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 488–489, 2017.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *ArXiv*, vol. abs/1606.01540, 2016.

[3] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *CoRR*, vol. abs/1506.02438, 2015.

[4] P. Reizinger and M. Szemenyei, "Attention-based curiosity-driven exploration in deep reinforcement learning," *ArXiv*, vol. abs/1910.10840, 2019.

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.