Asynchronous Methods for Deep Reinforcement Learning

Atul Balaji - EE16B002

March 5, 2019

1 Introduction

The Deep Q-Network (DQN), based on experience replay has achieved a good performance on Atari Games. But, experience replay is computationally and memory intensive. Also, it is an off-policy algorithm.
This paper proposes a lightweight framework for deep reinforcement learning. It describes four asynchronous algorithms, described in the following sections. The best method proposed in this paper is called **Asynchronous Advantage Actor-Critic** (A3C). Asynchronous methods have the following advantages:

- Instead of experience replay, multiple agents are asynchronously executed in parallel, on multiple instances of the environment. Since at any given time step the parallel agents will be experiencing a variety of different states, this will decorrelate the training samples for the agents and the distribution tends to be more stationary, thereby stabilizing the learning.
- It allows the usage of on-policy methods such as SARSA, Actor-critic.
- Also, unlike previous approaches, this paper provides an implementation that will run on a standard multi-core CPU, eliminating the need for GPUs, while achieving better results.

2 Algorithms

- **Asynchronous one-step Q-learning** - Each thread interacts with its own copy of the environment and at each step computes the gradient of the loss function. A shared target network is used. Gradients are accumulated over multiple time steps before they are applied. This reduces the chance of overwriting updates. Each thread uses a different exploratory policy resulting in more robust policies.
- **Asynchronous one-step SARSA** is very similar to asynchronous one-step Q-learning, except that the target value for $Q(s, a)$ is now $r + \gamma Q(s', a'; \theta^-)$, where a' is the action taken in state s'.
- **Asynchronous n-step Q-learning** uses n-step returns and it operates in forward view, since this is found to be better for training neural nets with momentum-based methods and backpropagation through time. The algorithm selects actions till t_{max} time-steps into the future or until a terminal state is reached and then gradients are computed for all (s, a) since the last update.

3 Asynchronous Advantage Actor-Critic

A3C is the trademark algorithm of this paper as it performs the best among the methods discussed. Here an estimate of policy (actor) as well as a value function (critic) are maintained. It uses n-step returns as in the case of n-step Q-learning. The policy and value function parameters are updated after t_{max} steps. The update rule is:

$$\nabla_{\theta'} log\pi(a_t|s_t;\theta')A(s_t, a_t;\theta,\theta_v) \tag{1}$$

$$where\ A(s_t, a_t;\theta,\theta_v)\ is\ an\ estimate\ of\ the\ advantage\ function: \sum_{i=0}^{k-1}\gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) - V(s_t;\theta_v) \tag{2}$$

The advantage function represents the advantage of taking action a_t from state s_t. Also k is upper bounded by t_{max}. In practical cases, the parameters of the policy and value functions are shared. Also, adding entropy of the policy π to the objective function encouraged more exploration and resulted in sub-optimal solutions being avoided. Three different optimization techniques - SGD with momentum, RMSProp (no stats) and RMSProp (with shared stats) were tried in this paper, with the latter being the most robust.

4 Conclusion

- It was observed that on some games, n-step methods learned faster than one-step methods because bootstrapping works better if targets span over a large temporal range. Policy-based A3C is better than the other three value-function based methods probably because the policy parameters are easily learned from the sequence of observations.
- A3C can easily be extended to continuous action spaces (MuJoCo physics simulator) and when tried on such tasks, performed well and found good solutions in a short time.
- Increasing the number of agents resulted in an increase in speed which means that it scales well with the number of parallel workers.
- Asynchronous methods are quite robust to the choice of learning rates. This is evidenced by the fact that the same range of learning rates were valid across different methods and games and resulted in solutions with similar performances.
- In this paper, experience replay has been eliminated due to the computational cost. However, it can be used to increase data efficiency, since it allows us to reuse old data.