**Background**

**Policy Gradient**

Recall that the primary goal in reinforcement learning is maximize the sum of expected rewards by learning an optimal policy, , which an agent employs to determine the next action based on the current state of its world. In other words, this policy maps states to actions. Because the state space of a given problem may be huge, we are primarily interested in parametrized representations of policies. While there are many algorithms to learn such a policy, one popular method is the policy gradient algorithm.

Consider a deterministic policy, , in a deterministic environment. If we let be a closed form expression representing the reward when is executed, then we can use standard optimization procedures to optimize our reward function. We then define the policy gradient vector as . If is differentiable, we can express the policy gradient in closed form. Otherwise, we can follow the empirical gradient by hill climbing to achieve an approximation for the policy gradient. In the case of a stochastic policy, which maps states to a probability distribution, and stochastic environment, we can represent the policy gradient as

In which for trajectory or rollout , we can consider the state and action at the th step of the trajectory according to and receive reward . We can use the policy gradient to tell us how we should improve our policy such that we might converge to a local minima. Observe how this algorithm is very similar to gradient descent in structure. We define the policy gradient algorithm (Williams 1992) as the following

POLICY-GRADIENT:

1. Sample from

**Actor Critic**

The two primary reinforcement learning methods are value-based methods, such as Q-Learning or Deep Q-Learning, and policy-based methods, such as policy gradients. Recall that value-based methods learn a value function that map each state action pair to a value, and we calculate our policy by taking the action that with maximum value. Policy-based methods directly learn such a policy without a value function. The actor-critic algorithm is a hybrid method that uses both value-based and policy-based methods. More specifically, the actor-critic algorithm uses two controllers: a policy-based actor that controls how an agent behaves in its environment, and a value-based critic that measures how good the action taken is and how the actor should adjust its policy. We can define an advantage function that defines how much better an action was than expected

The actor-critic algorithm trains the actor and critic in parallel. The actor’s parameters are updated in a very similar way to the vanilla policy gradient algorithm, but the update step uses the advantage function as a scalar multiplier of the policy gradient. The critic is trained to directly minimize the advantage function. The actor-critic paradigm helps reduce the variance of policy gradient.

**DDPG**

The traditional policy gradient algorithm suffers from high variance and slow convergence. Furthermore, it may be difficult to choose an appropriate learning rate for the given problem. Additionally, while algorithms such as DQN can solve problems with high-dimensional observation spaces, they cannot handle high-dimensional action spaces. Lilicrap et. all introduce a model-free off-policy actor-critic algorithm called Deep Deterministic Policy Gradient (DDPG) that addresses such challenges.

DDPG is a policy gradient algorithm that uses a stochastic policy for exploring the state space to estimate a deterministic target policy. The actor network, , receives the current state space as the input and outputs a single real value from a continuous action space. The critic network, , receives the current state space and the actor’s choice of action as input and outputs the estimated Q-value of the provided action. DDPG also borrows the ideas of a replay buffer and target Q network from DQN. DDPG randomly samples from the replay buffer to calculate approximations of the gradients as an attempt to decorrelate training samples and DDPG also uses target networks to create more stable training rollouts. Finally, DDPG adds a noise parameter to the action space; one common choice for this noise parameter is the Ornstein-Uhlenbeck Random Process.

**A3C**

Although utilizing a replay buffer decorrelates updates, it limits methods to off-policy reinforcement learning algorithms because the data must be batched or randomly sampled. Furthermore, the replay buffer uses significant memory and computation. Mnih et all. introduce an asynchronous advantage actor critic algorithm (A3C) to help overcome these problems.

Recall that in DQN, a single agent represented by a single neural network controller interacts with a single environment. A3C is asynchronous: it utilizes multiple agents via a single global neural network controller and each agent has its own set of network parameters. Each agent is independent of one another, which decorrelates the actors and increases the diversity of training the networks because each agent is likely to be exploring a different part of the environment. Asynchronous agents eliminate the need for a replay buffer, which allows models to pursue on-policy learning algorithms to train neural networks, such as the actor-critic algorithm. A3C employs an actor-critic paradigm to train two neural networks: the actor that estimates policy and the critic that estimates a value function. A3C also utilizes the advantage function while performing updates. The resulting architecture and training of A3C agents yields much more stable neural networks with both value-based and policy-based methods, on-policy and off-policy methods, as well as discrete and continuous domains. Because of its extreme flexibility, fast training time, and model stability, A3C has become the start of the art, go-to starting point for many deep reinforcement learning problems.

Citations:

<http://www-anw.cs.umass.edu/~barto/courses/cs687/williams92simple.pdf>

<https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf>

<https://arxiv.org/pdf/1509.02971.pdf>

<https://arxiv.org/pdf/1602.01783.pdf>