Chapter 13 - Policy Gradient Methods

Atul Balaji - EE16B002

February 25, 2019

Unlike other methods which use action-values to determine policies, policy gradient methods learn a **parameterized policy** and do not make use of value functions. We can write:

$$\pi(a|s,\theta) = P[A_t = A|S_t = s, \theta_t = \theta] \tag{1}$$

θ is learned based on a **performance measure** $J(\theta)$.

1 Policy Approximation

If the action space is discrete, we form a **preference** $h(s,a,\theta) \in \mathbb{R}$ for each (s,a) pair. For example,

$$\pi(a|s,\theta) = \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}} \tag{2}$$

Higher preference of an action implies that there is a higher probability of selection of that action. Parameterization of policies has the following advantages:

- Softmax parameterization works better than ϵ-greedy selection since this method allows π to approach a deterministic policy, for otherwise it will end up always choosing a random action with some finite probability ϵ.
- Action-value methods have no natural way of finding stochastic optimal policies, where actions must be selected with some arbitrary probabilities. However, this is possible in with parameterization and using softmax preferences.
- Sometimes, the complexity of action-value functions maybe too high. In such cases, policy based methods like policy parameterization can provide a simpler function to approximate.
- One of the main advantages of using a policy based method is that through policy parameterization, we are enabled to inject prior knowledge about the desired form of the policy into the system.

2 Policy Gradient Theorem

This theorem is used to find the an analytic form for the gradient of J with respect to θ.
Consider $J(\theta) = v_{\pi_\theta}(s_0)$, where s_0 is the episode start state.
The policy gradient theorem (for the episodic case) states that

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta) \tag{3}$$

In the episodic case, we can take the constant of proportionality as the average length of an episode, and is 1 in the continuing case. In this equation, μ is the on-policy distribution under π.

3 REINFORCE Algorithm: Monte Carlo Policy Gradient

- Consider the above equation. This can also be written as:

$$E_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\theta) \right] \tag{4}$$

  Using this, we can perform the **stochastic gradient ascent** update:

$$\theta_{t+1} = \theta_t + \alpha \sum_a \hat{q}(S_t,a,w) \nabla \pi(a|S_t,\theta) \tag{5}$$

  where α is the step size, $\hat{q}$ is a learned approximation to q_π and w is the value function's weight vector.
  This is called the **all-actions** method since a sum of over all actions is involved in the update equation.

- In REINFORCE, we will additionally introduce the variable A_t which is the action taken at time step t. Now, the gradient equation can be written as:

$$\nabla J(\theta) = E\left[G_t \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}\right] \tag{6}$$

  Also the update equation changes to:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \tag{7}$$

  The term $\frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} = \nabla \ln(A_t|S_t,\theta$ is called the **eligibility vector**.

- As we see from these results, REINFORCE uses the complete return from time t. Also, updates are done only after the completion of an episode. This shows that it is a Monte Carlo algorithm.

- Being a Monte Carlo algorithm, REINFORCE suffers from high variance and slow learning.

3.1 REINFORCE with baseline

- The performance gradient equation changes to:

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla \pi(a|s,\theta) \tag{8}$$

- The baseline b(s) is independent of the action, but is dependent on the state s.

$$\theta_{t+1} = \theta_t + \alpha(G_t - b(S_t)) \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \tag{9}$$

- Introduction of baseline helps to reduce variance and leads to faster learning.

4 Actor Critic Methods

- Methods in which approximations to both the value function and the policy are learned in a competing and cooperative manner, are called **actor-critic** methods.

- Bootstrapping critic methods allow us to introduce bias and reduce the variance, and speed up learning. The one step actor critic update equation is:

$$\theta_{t+1} = \theta_t + \alpha \left[R_{t+1} + \gamma \hat{v}(S_t,w) - \hat{v}(S_t,w)\right] \frac{\nabla \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} = \theta_t + \alpha \delta_t \frac{\nabla \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \tag{10}$$