MS&E338 Reinforcement Learning Lecture 10 - May 2, 2018

Lecture 10: Policy Gradient Methods

Lecturer: Ben Van Roy *Scribe: Vikranth Dwaracherla, Jacob Perricone*

In this lecture we will review some of the performance bounds previously discussed and introduce policy gradient methods.

1 Review

1.1 Performance bounds for Aggregation case:

Approximate Value Function (AVI): Let $\tilde{Q}$ be the fixed point with respect to a fixed distribution μ for approximate value iteration in case of state aggregation.

$$\tilde{Q} = \Pi_\mu F \tilde{Q} \tag{1}$$

Error bound:

$$||Q^* - \tilde{Q}||_\infty \leq \frac{1}{1-\alpha} ||Q^* - \Pi_\mu Q^*||_\infty \tag{2}$$

This says that if the projection of Q^* onto our subspace is close to optimal, then AVI gives us something which is close to optimal. Further, if $\tilde{\pi}$ is the greedy policy with respect to $\tilde{Q}$, then

$$||Q^* - Q_{\tilde{\pi}}||_\infty \leq \frac{2\alpha}{(1-\alpha)^2} ||Q^* - \Pi_\mu Q^*||_\infty \tag{3}$$

This result is a bit vacous since the $(1-\alpha)^2$ could be tiny.

Trajectory distribution $\tilde{\mu}$ - Real time reinforcement learning: The idea is that instead of coming up with an arbitrary $\tilde{\mu}$, we have an algorithm generate the $\tilde{\mu}$. If this process converges it solves the equation $\tilde{Q} = \Pi_{\tilde{\mu}} F \tilde{Q}$. Here, we have $\tilde{Q}$ also influening $\tilde{\mu}$, since $\tilde{Q}$ determines $\tilde{\pi}$, which in term determines $\tilde{\mu}$. In this case we have performance bound,

$$\tilde{\mu}^\top (Q^* - Q_{\tilde{\pi}}) \leq \frac{\alpha}{1-\alpha} ||Q^* - \Pi_{\tilde{\mu}} Q^*||_\infty \tag{4}$$

The left hand sides of (3) and (4) are different. But if we multiply by $(1-\alpha)$ and let $\alpha \to 1$, they are the same, which is the difference between the average reward of optimal to the greedy policy. In (4), difference between average rewards of optimal and greedy policy does not blow up, unlike the case (3). This was the main point of the last lecture that there is some magic here in how the reinforcement learning algorithms efficiently use data to learn from actual experience.

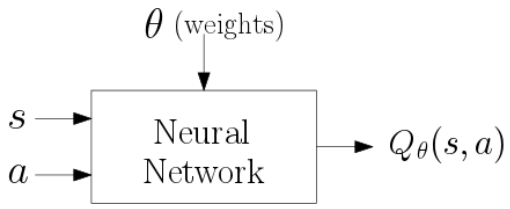
2 Policy Gradient Methods

Aside from being conceptually simple, Policy Gradient Methods are nice in that they always work. That is, they improve the policy by moving in the direction of improvement. They essentially perform hill climbing in parametric space. Nevertheless, they are extremely data inefficient and can get stuck at local optima.

2.1 Policy Learning

We want a parameterized representation of a policy - $\pi_\theta(a|s)$ i.e. some way to choose an action based on a state. A popular way to parameterize is by a parametrized Q function, $Q_\theta(s, a)$ which may not be an actual Q function and just some way to represent a policy.

One example is a Deep neural network which takes (s, a) as input and produces $Q_\theta(s, a)$. Here, weights of the neural network acts as parameters, θ.



One way to define a policy is by a using a distribution. We want policies to be continuous and differentiable with respect to theta for convenience. An example of policy is

$$\pi_\theta(a|s) = \frac{e^{\beta Q_\theta(s,a)}}{\sum_{a' \in A} e^{\beta Q_\theta(s,a')}} \tag{5}$$

As β increases we concentrate more on the maximum value, i.e. become more greedy. That is we assign higher probabilities to actions with higher Q_θ. The extent to which the probability concentrates on the largest Q_θ depends on β. Another advantage of (5) is that Q_θ is unconstrained.

Main ideas of policy gradient algorithms are

- Using chain rule to find an update expression for θ
- Writing this update rule as an approximation and solving for steady state expectation

Consider episodic framework. i.e, in each episode, we terminate with probability 1 at a random instant.

Lets define,

$$V(\pi) = \sum_{s \in S} \rho(s) V_\pi(s) \tag{6}$$

which is a weighted average of the rewards for an episode. So this is a function of π, but we want it in terms of θ, define $V(\theta) = V(\pi_\theta)$. Also note that, $V_\pi(s) = \Sigma_{a \in A} Q_\pi(s, a)\pi_\theta(a|s)$ as $\pi(a|s)$ is a distribution.

$$\mu_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\tau-1} \mathbb{1}_{s_t = s} \mid \pi\right] \tag{7}$$

be the expected number of times we visit state, s. We also have $\mu_\theta(s) = \mu_{\pi_\theta}(s)$. After applying the chain rule to the gradient of $V(\theta)$,

$$\nabla_\theta V(\theta) = \sum_{s \in S} \mu_\theta(s) \sum_{a \in A} \nabla_\theta \pi_\theta(a|s) Q_{\pi_\theta}(s, a) \tag{8}$$

We are averaging over states, and then for all actions, we have $\nabla_\theta \pi_\theta(a|s)$ as the direction to nudge θ in order to improve the value of our estimate at that action.

Stochastic Gradient Descent Algorithm

Algorithm 1: Stochastic Gradient descent

```
begin
    initialize θ;
    for l = 1, 2, 3, ... do
        Apply π_θ for episode;
        compute stochastic gradient d_l from the trajectory (s_0, a_0, r_1, ..., s_τ);
        θ := θ + γ_l d_l
    end
end
```

Here, d_l is an estimate of gradient of $V(\theta)$ with respect to θ. There are several ways to define d_l. One proposal for d_l is to first define

$$\hat{Q}_t = \sum_{k=t}^{\tau-1} r_{k+1} \tag{9}$$

Then

$$d_l = \sum_{t=0}^{\tau-1} \left(\frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \right) \hat{Q}_t \tag{10}$$

$\mathbb{E}[d_l]$ for d_l in (10) gives us the original expression, (8) for $\nabla_\theta V(\theta)$.

Next we will discuss **Actor Critic Methods**, which learn both value function and policy at the same time. This is more efficient in terms of samples required.