EE675A

Lecture 21: Policy Gradient Methods

10/04/2023

Lecturer: Subrahmanya Swamy Peruru *Scribe: Chikoti Tejas and Lavesh Mangal*

From the last lecture -

- Model Based Learning (Dyna-Q)
- Function Approximation
- Deep Q-Network

In today's lecture -

- Motivation for Policy Gradient Method
- Benefits of Stochastic Policy
- Variations in Policy Gradient (Actor-Critic, Baseline version)
- Policy Gradient in model free RL setting

1 Recap

1.1 Model-based RL

Model-based RL algorithms use an explicit model of the environment to make predictions about the next state and reward. Once the model is learned (i.e. the estimation of $P^a_{ss'}$ and R^a_s from interaction data) the agent can "simulate" experience by generating trajectories using the model, which can be used to update its value estimates and improve its policy.
Dyna-Q is a model-based reinforcement learning algorithm that integrates a model of the environment within the Q-learning framework to improve learning efficiency. In Dyna-Q, we make usual Q-learning update (called learning) but 1 step of learning is followed by n steps of simulated experience (called planning).

1.2 Function Approximation

Tabular approach is feasible for small and discrete state spaces, but it becomes impractical for large or continuous state spaces. In such a situation we use Function Approximation. Function approximation represents states using its features.
Linear Function Approximation : We use stochastic gradient descent (SGD) for learning.

1. Monte Carlo for approximation (using G_t instead of $v_\pi(s)$)
2. TD for approximation (Semi-Gradient Descent)

1.3 Deep Q-Network

Deep Q-Learning is a popular algorithm for reinforcement learning. However from Deep learning perspective it faces two major issues :

1. In DQN, the target is a non-stationary. Solved by using a copy of neural net and keep it fixed for sometime to use as targets.
2. In DQN, the samples are sequences of highly correlated states. Solved by using Memory Replay Buffer.

2 Policy Gradient Methods

2.1 Motivation

1. Useful in continuous action spaces.
2. These have better convergence properties. Since, we can use Stochastic Gradient Descent instead of semi-gradient descent.
3. We can learn stochastic policies usinf Policy Gradient Methods.

2.2 Benefits of Stochastic Policies

Optimal deterministic policy is guaranteed only for Tabular methods. There may not be an optimal deterministic policy when we use function approximation. We will see this using an example (see figure 1).

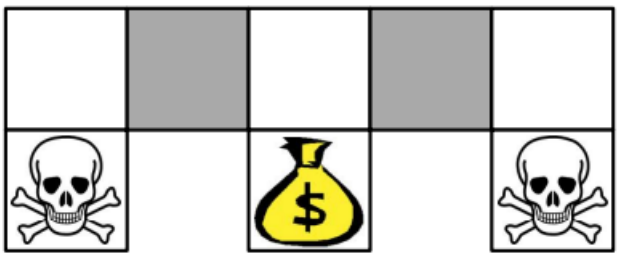

Figure 1: Aliased Gridworld

The states will be represented using some features. Here, we use 4 dimensional vector, each entry stating whether there is a wall in that direction.

$$S \rightarrow (\mathbb{1}_{left}, \mathbb{1}_{right}, \mathbb{1}_{up}, \mathbb{1}_{down}) \tag{1}$$

The leftmost state will be represented by (1,0,1,0). It is clear that the grey states in the figure have same features.

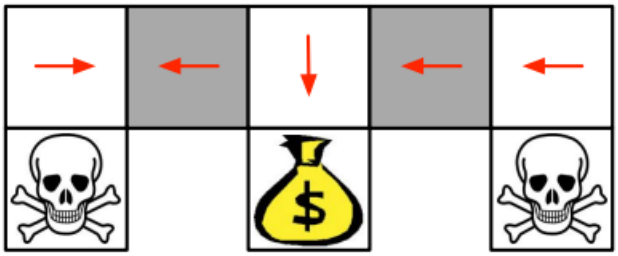

Figure 2: Deterministic Policy learnt by function approximation

A deterministic policy will have same action for both of these states. Without loss of generality, consider it as in figure 2. This will lead to sub-optimal policy. The agent will be stuck forever in the left most states.

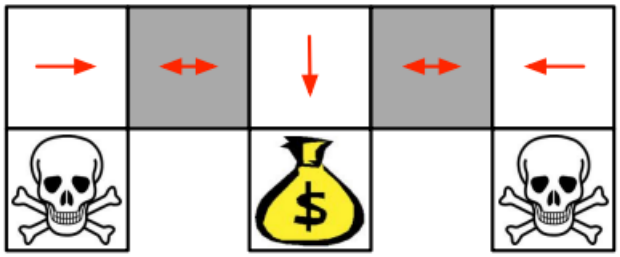

Figure 3: Stochastic Policy learnt by function approximation

A stochastic policy will have equal chance of going left or right, and will lead to the optimal policy. This is shown in figure 3. The agent will reach the goal with high probability.

2.3 Policy Gradient in Bandits settings : REINFORCE

In Lecture 8, we gave the Policy gradient algorithm for bandit setting. Following is the equation of gradient of reward function. For an elaborate derivation, refer to Lecture 8.

$$\frac{d\eta(\theta)}{d\theta_i} = \mathop{\mathbb{E}}_{\Pi,v}\left[R_t \cdot \frac{d\log(\mathbf{\Pi}(a;\theta))}{d\theta_i}\right] \tag{2}$$

The update equation is $\theta_i^{(t+1)} = \theta_i^{(t)} + \alpha_t \cdot R_t \cdot \frac{d\log(\mathbf{\Pi}(a;\theta))}{d\theta_i}$

This is also known as REINFORCE algorithm (William, 1992).

2.4 Policy Gradient for full RL setting

Now we aim to derive the best policy parameters for the arbitrary policy $\pi_\theta(s, a)$ For this, we first focus on the objective we wish to maximize; for episodic MDP we can use the average return per episode, while for continuing MDP, we use the average return per step.

Objective for an eposidic MDP:

$$J(\theta) = \mathop{\mathbb{E}}_{S_0 \sim d}[V_\pi(S_0)] \tag{3}$$

$$J(\theta) = \mathop{\mathbb{E}}_{S_0 \sim d}[\mathbb{E}_\pi[G_t | S_t = S_0]] \tag{4}$$

$$J(\theta) = \mathop{\mathbb{E}}_{S_0 \sim d, \pi_\theta}[G_t] \tag{5}$$

where d is the start-state distribution,G_t is the cumulative reward up to to the termination of the episode.

2.4.1 Analogy to the Bandits Problem

$$\tau = S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2 .....S_{T-1}, A_{T-1}, R_T, S_T \tag{6}$$

Here τ is the trajectory of the episodic MDP followed by given a starting state S_0 and taking actions according to policy π

We already have derived the formula for policy gradients in the multi-arm bandits setting and is given by

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R(S, A)] = \mathbb{E}_{\pi_\theta}[R(S, A) \cdot \nabla_\theta \log(\pi(A|S))] \tag{7}$$

Similarly we can form the Gradient for the full MDP setting using the probability distribution of trajectory in place of π(policy) and using the cumulative reward G_t in place of the reward.The final equation looks like,

$$\nabla_\theta J_\theta(\pi) = \nabla_{P(\tau)} \mathbb{E}[G(\tau)] \tag{8}$$

Now using a similar analogy to the bandit setting we can write it as :

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}[G(\tau) \cdot \nabla_\theta \log(p(\tau))] \tag{9}$$

Upon further simplification and using

$$P(\tau) = P(S_0) \cdot P(\pi_\theta(A_0|S_0)) \cdot P(S_1|S_0, A_0) \cdot P(\pi_\theta(A_1|S_1)) .....P(\pi_\theta(A_{T-1}|S_{T-1})) \cdot P(S_T|S_{T-1}, A_{T-1}) \tag{10}$$

In the above equation the state transition probabilities are not dependent on the parameterized policy π and hence on taking the logarithm of the same won't appear in the final equation.The final expression can be expressed as:

$$\nabla_\theta J_\theta(\pi) = \mathbb{E}_\pi[G(\tau) \cdot \nabla_\theta \sum_{t=0}^{T} \log(P(\pi_\theta(A_t|S_t)))] \tag{11}$$

So now we have a sample of the gradient which is equivalent to $G(\tau) \cdot \nabla_\theta \sum_{t=0}^{T} \log(P(\pi_\theta(A_t|S_t)))$ and this can be used in the updating the policy given after sampling an episode.

The material is primarily based on the content from [2] and [1]

References

[1] D. Silver. Lectures on reinforcement learning. URL: https://www.davidsilver.uk/teaching/, 2015.

[2] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.