Lecture 14

Offline Approximate Dynamic Programming

In Chapter 2, we learned about techniques for dynamic programming for the case of discrete states and discrete actions. However, in these situations we are limited not only to discrete state space and action sets, but we are also limited by the "curse of dimensionality". That is as our state and action spaces grow, it becomes increasingly intractable to use dynamic programming. To overcome some of these limitations, researchers have leveraged function approximation to represent the value function (as well as the policy). These learning methods are typically applied offline, via supervised-learning/regression.

14.1. Fitted Value Iteration

Consider the optimization problem

$$J^*(x_0) = \min_u \sum_{k=0}^{\infty} g(x_k, u_k) \tag{14.1}$$

$$s.t. \quad x_{k+1} = f(x_k, u_k). \tag{14.2}$$

We have seen how we can use the principle of optimality to write the value iteration algorithm

$$\hat{J}^{*,k+1} = \min_u \left[g(x,u) + \hat{J}^{*,k}(x') \right]. \tag{14.3}$$

However, now we have continuous states and a potentially high dimensional state space. If we are able to sample our system at states x_i we can compute J_i from samples as

$$J_i^d = \min_u [g(x_i, u) + \hat{J}_\theta^*(f(x_i, u))] \tag{14.4}$$

$$\hat{\theta} = \arg\min_\theta \sum_i (\hat{J}_\theta^*(x_i) - J_i^d)^2. \tag{14.5}$$

Convergence is not guaranteed for general function approximators. However, linear function approximators are actually known to converge to globally optimal θ^*. Consider the linear function approximator

$$J_\theta^*(x) = \sum_i \theta_i \psi_i(x) = \psi(x)^T \theta \tag{14.6}$$

where

$$\theta = \begin{bmatrix} \psi^T(x_1) \\ \vdots \\ \psi^T(x_i) \end{bmatrix}^+ \begin{bmatrix} J_1^d \\ \vdots \\ J_i^d \end{bmatrix}. \tag{14.7}$$

14.2. Fitted Q-Iteration

What happens if you don't know the process model? Remember, to get the policy we use

$$\pi^*(x) = \arg\min_u \left[g(x,u) + J^*(x') \right]. \tag{14.8}$$

The Q-function, or quality function, combines both actions and values into a single function. For the deterministic discrete action, discrete time case we have

$$\begin{aligned} Q^*(s_k, a_k) &= g(s_k, a_k) + J^*(s_{k+1}) && (14.9)\\ &= g(s_k, a_k) + \min_a \left[g(s_{k+1}, a_{k+1}) + J^*(s_{k+2}) \right] && (14.10)\\ &= g(s_k, a_k) + \min_a Q^*(s_{k+1}, a_{k+1}). && (14.11) \end{aligned}$$

We can then write the algorithm for Q-Iteration as

$$Q^{*,i+1}(s,a) = g(s,a) + \min_a Q^{*,i}(s', a'). \tag{14.12}$$

Both the value function and the policy can be retrieved from the Q function as

$$\begin{aligned} V^*(s) &= \min_a Q^*(s,a) && (14.13)\\ \pi^*(s_i) &= \arg\min_a Q^*(s,a). && (14.14) \end{aligned}$$

You can apply the same approximate dynamic programming technique with a fitted Q-function. For continuous states and actions, we have

$$Q^{*,i+1}(x,a) = g(x,a) + \min_a Q^{*,i}(x,a). \tag{14.15}$$

In vector form, where we sample from the state space, x_i, and action space, u_i, we have

$$Q_i^d = g(x_i, a_i) + \min_a \hat{Q}_\theta^*(x_i, a) \tag{14.16}$$

and we can estimate $\hat{\theta}$ as

$$\hat{\theta} = \arg\min_\theta \sum_i (\hat{Q}_\theta^*(x_i, a_i) - Q_i^d)^2. \tag{14.17}$$

14.3. Approximate Policy Iteration

A similar approximation strategy can be used in an approximate policy iteration approach. We can write

$$\begin{aligned} Q^\pi(s_k, \pi(s_k)) &= g(s_k, \pi(s_k)) + J^\pi(s_{k+1}) && (14.18)\\ &= g(s_k, \pi(s_k)) + g(s_{k+1}, \pi(s_{k+1})) + J^\pi(s_{k+2}) && (14.19)\\ &= g(s_k, \pi(s_k)) + Q^\pi(s_{k+1}, \pi(s_{k+1})) && (14.20)\\ &= g(s, \pi(s)) + Q^\pi(s', \pi(s')). && (14.21) \end{aligned}$$

Policy Iteration with a Q-function becomes:

$$\begin{aligned} Q^{\pi,j+1}(s,a) &= g(s, \hat{\pi}^{*,i}(s)) + Q^{\pi,j}(s', \hat{\pi}^{*,i}(s')) && (14.22)\\ \hat{\pi}^{*,i+1}(s) &= \arg\min_a Q^\pi(s,a). && (14.23) \end{aligned}$$

Approximate policy iteration uses function approximators to represent the Q-function. However, approximate policy iteration can suffer from the same convergence issues exhibited by fitted value iteration. Even though the policy evaluation step is no longer dependent a minimization step, the policy update can step can still erroneously exploit an imperfect value function estimate.

Bibliography

[1] Russ Tedrake. *Underactuated Robotics*. 2023.

[2] Dimitri Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[3] Drew Bagnell, Byron Boots, and Sanjiban Choudhury. *Modern Adaptive Control and Reinforcement Learning*. 2022.