EE675A

Lecture 15: MC Control and TD Control

15/03/2023

Lecturer: Subrahmanya Swamy Peruru *Scribe: Snehil Srivastava, Himanshu Sood*

1 Recap and Overview

In the previous lecture, the Monte Carlo (MC) and Temporal Difference (TD) methods were compared. Batch MC methods always find the estimates that minimize mean square error on the training set, whereas batch TD(0) always finds the estimates that would be exactly correct for the maximum-likelihood model of the Markov process. n-Step TD and TD(λ) were also introduced.

n-Step TD:

$$V_{new}(S_t) = V_{old}(S_t + \alpha[G_t^{(n)} - V_{old}(S_t)])$$

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} .... + \gamma^{n-1} R_{t+n} + \gamma^n V_{old}(S_{t+n})$$

TD(λ):

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} G_t^{(n)} \lambda^{n-1}$$

$$G_t^{(n)} = G_t \quad \forall n \geq T-t$$

2 Challenges in Replacing Policy Evaluation with Monte Carlo Estimation

2.1 Unknown MDP Dynamics

In generalized policy iteration, the policy π is found from previously evaluated value function, for example,

$$\pi_1 = \underset{a}{\operatorname{argmax}} \, (R_s^a + \gamma \sum_{s'} P_{ss'}^a V_{\pi_0}(s'))$$

Since the MDP dynamics are unknown, the solution is to estimate Q_π rather than V_π so that we can simply find the policy as,

$$\pi_1 = \underset{a}{\operatorname{argmax}} \, Q_{\pi_0}(s,a)$$

To estimate $q_\pi(s,a)$ using MC, we simply start at state s, take action a and thereafter follow policy π. The return obtained from this episode will give one sample for $q_\pi(s,a)$.

2.2 Low Sample Efficiency in Monte Carlo estimation

One complication in MC method is that many stateaction pairs may never be visited. If π is a deterministic policy, then in following π, one will observe returns only for one of the actions from each state. With no returns to average, the Monte Carlo estimates of the other actions will not improve with experience. This is a serious problem because the purpose of learning action values is to help in choosing among the actions available in each state. For policy evaluation to work for action values, we must assure continual exploration. One way to do this is by specifying that the episodes start in a stateaction pair, and that every pair has a non-zero probability of being selected as the start. This guarantees that all state-action pairs will be visited an infinite number of times in the limit of an infinite number of episodes. Another way is to use stochastic policies instead of deterministic policies, this allows exploration not just at the starting state, but also in the later parts of the episode trajectory.

For Monte Carlo policy iteration it is natural to alternate between evaluation and improvement on an episode-by-episode basis. After each episode, the observed returns are used for policy evaluation, and then the policy is improved at all the states visited in the episode. This is accomplished in the algorithm described below, called "Monte Carlo Exploratory Start"

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Algorithm 1 Monte Carlo Exploratory Start

```
1: $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
2: $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
3: $\mathrm{Returns}(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
4: while True do
5:     Choose $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability $> 0$
6:     Generate an episode from $S_0, A_0$, following $\pi : S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$ $G \leftarrow 0$
7:     for $t$ in $T-1, T-2, ..., 0$ do
8:         $G \leftarrow \gamma G + R_{t+1}$
9:         if $S_t, A_t$ does not appear in $S_{t-1}, A_{t-1}, .., S_0, A_0$ then
10:            Append $G$ to Returns $(S_t, A_t)$
11:            $Q(S_t, A_t) \leftarrow \text{average}(\text{ Returns } (S_t, A_t))$
12:            $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$
13:        end if
14:    end for
15: end while
```

Theorem 2.1. *Policy Improvement Theorem: Any ϵ-greedy policy (π') with respect to q_π is an improvement over any ϵ-soft policy π*

Proof.

$$\begin{aligned}
q_\pi\left(s, \pi'(s)\right) &= \sum_a \pi'(a \mid s) q_\pi(s, a) \\
&= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + (1-\varepsilon) \max_a q_\pi(s, a) \\
&\geq \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + (1-\varepsilon) \sum_a \frac{\pi(a \mid s) - \frac{\varepsilon}{|\mathcal{A}(s)|}}{1-\varepsilon} q_\pi(s, a)
\end{aligned}$$

Thus, by the policy improvement theorem, $\pi' \geq \pi$ (i.e., $v_{\pi'}(s) \geq v_\pi(s)$, for all $s \in \mathcal{S}$). We now prove that equality can hold only when both π' and π are optimal among the ε-soft policies, that is, when they are better than or equal to all other ε-soft policies.

Consider a new environment that is just like the original environment, except with the requirement that policies be ε-soft "moved inside" the environment. The new environment has the same action and state set as the original and behaves as follows. If in state s and taking action a, then with probability $1-\varepsilon$ the new environment behaves exactly like the old environment. With probability ε it re-picks the action at random, with equal probabilities, and then behaves like the old environment with the new, random action. The best one can do in this new environment with general policies is the same as the best one could do in the original environment with ε-soft policies. Let $\widetilde{v}_*$ and $\widetilde{q}_*$ denote the optimal value functions for the new environment. Then a policy π is optimal among ε-soft policies if and only if $v_\pi = \widetilde{v}_*$. From the definition of $\widetilde{v}_*$ we know that it is the unique solution to

$$\begin{aligned}
\widetilde{v}_*(s) =&(1-\varepsilon) \max_a \widetilde{q}_*(s, a) + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a \widetilde{q}_*(s, a) \\
=&(1-\varepsilon) \max_a \sum_{s', r} p\left(s', r \mid s, a\right)\left[r + \gamma \widetilde{v}_*\left(s'\right)\right] \\
&+ \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a \sum_{s', r} p\left(s', r \mid s, a\right)\left[r + \gamma \widetilde{v}_*\left(s'\right)\right]
\end{aligned}$$

When equality holds and the ε-soft policy π is no longer improved, then we also know that

$$\begin{aligned}
v_\pi(s) =&(1-\varepsilon) \max_a q_\pi(s, a) + \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) \\
=&(1-\varepsilon) \max_a \sum_{s', r} p\left(s', r \mid s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right] \\
&+ \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a \sum_{s', r} p\left(s', r \mid s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right]
\end{aligned}$$

However, this equation is the same as the previous one, except for the substitution of v_π for $\widetilde{v}_*$. Because $\widetilde{v}_*$ is the unique solution, it must be that $v_\pi = \widetilde{v}_*$. $\square$

3 TD methods for Control Problem

In the use of TD prediction methods for the control problem, the same pattern of generalized policy iteration (GPI) is followed, only using TD methods for the evaluation or prediction part. An action-value function is learnt rather than a state-value function. In particular, we must estimate $q_\pi(s,a)$ for the current behavior policy π and for all states s and actions a.

3.1 TD for V_π

$$\begin{aligned} v_\pi(s) &= R_s^\pi + \gamma \sum_{s'} v_\pi(s') p_{ss'}^\pi \\ &= R_s^\pi + \gamma \mathbb{E}_\pi[v_\pi(s_{t+1}) | s_t = s] \end{aligned}$$

And the sample version,

$$V_{new}(S_t) = V_{old}(S_t) + \alpha[R_{t+1} + \gamma V_{old}(S_{t+1}) - V_{old}(S)]$$

3.2 TD for q_π

$$\begin{aligned} q_\pi(s,a) &= R_s^a + \gamma \sum_{s'} v_\pi(s') p_{ss'}^a \\ &= R_s^a + \gamma \sum_{s'} \left(\sum_{a'} \pi(a'|s') q_\pi(s',a') \right) p_{ss'}^a \\ &= R_s^a + \gamma \sum_{s',a'} p_{ss'}^a \pi(a'|s') q_\pi(s',a') \end{aligned}$$

And the sample version,

$$Q_{new}(S_t, A_t) = Q_{old}(S_t, A_t) + \alpha[R_{t+1} + \gamma Q_{old}(S_{t+1}, A_{t+1}) - Q_{old}(S_t, A_t)]$$

This update is done after every transition from a non-terminal state S_t. If S_{t+1} is terminal, then $Q(S_{t+1}, A_{t+1})$ is defined as zero. This rule uses every element of the quintuple of events, $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, that make up a transition from one state-action pair to the next. This quintuple gives rise to the name *Sarsa* for the algorithm.

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

References

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2020.