Summary - Chapter 5 - Monte Carlo Methods

Ameet Deshpande

December 28, 2017

1 Summaries

The motivation for using *Monte Carlo* methods is that we can use only experience. It is not necessary to know the environment dynamics beforehand like in *Dynamic Programming* based methods. Monte-Carlo methods use the idea of averaging sample returns. They are thus natural for episodic tasks and are used mainly for these.

1.1 Section 5.1 - Monte Carlo Policy Evaluation

- For these methods also it is important to evaluate the value functions. We are given a set of episodes that the agent has experienced. Using those, we will try and estimate the Value functions.
- There are two main ways of using the episodes, *every-visit MC* and *first-visit MC*. In the first type, each time a state s is visited, we consider all the returns there on for the state s. For *first-visit*, only the first time the state s is visited is considered for returns. Consider this example. $s_1 \rightarrow s_2 \rightarrow s_1 \dots s_n$. In *First-visit*, only the first occurrence of s_1 is considered for returns. In *Every-visit* both the occurrences of s_1 are considered while averaging over returns.

1.2 Section 5.2 - Monte Carlo Estimation of Action Values

- It is important to estimate Q values to be able to help improve the policies. Taking an action stochastically leads us to a different state with some reward. Estimating Q will allow us to choose the action which leads us to a combined best state and reward. For example, say there are 3 states, s_1, s_2, s_3, $V(s_2) > V(s_3)$ and s_2 and s_3 are successor states for s_1. When in s_1, how to decide which action to take, a_1 or a_2? Clearly just the state values are not sufficient.
- The episodes that the MC methods consider are generated by some policy π. To estimate the *action-values*, it is important that all state-value pairs have been considered in the policy π, which may not be the case if π is a deterministic policy. Note how the *exploration-exploitation* dilemma kicks in here.
- One way of maintaining exploration is *exploring starts*. One way to do this is by specifying that the first step of each episode starts at a state-action pair, and that every such pair has a nonzero probability of being selected as the start.

1.3 Section 5.3 - Monte Carlo Control

- To develop the theory, this section assumes that infinite episodes are generated. Thus, *just* using *exploring starts* we can ensure that all state-action pairs have infinite number of occurrences.
- Policy Improvement is done like in previous cases by choosing the greedy action.
- It is natural to do Policy Improvement and Policy Evaluation after each episode. After each episode, the new averages are calculated for each state visited in that episode and the action values are updated.

1.4 Section 5.4 - On-Policy Monte Carlo Control

- To avoid the unlikely assumption of *Exploring Starts*, we need to make sure we have a policy which can keep choosing all the non-greedy action. To this end, ϵ-soft policies attribute a minimum probability of $\frac{\epsilon}{|A(s)|}$ to each non-greedy action.
- Considering only ϵ-soft policies, it is easy to see that the policy that attributes $1 - \epsilon + \frac{\epsilon}{|A(s)|}$ to the *greedy-best* action, will be the policy which has the best value function V.

1.5 Section 5.5 - Evaluating One Policy While Following Another

- The methods considered so far have the property that the training data is almost being generated online. As and when we improve the policy, it is used to generate the episodes. But there might be cases when we are given episodes beforehand and we are still expected to come up with the best policies. This is the motivation of *off-policy* Monte Carlo Control.
- To use the values and returns from another policy to estimate value functions for the current policy, we need to form an unbiased estimate by taking the ratio of probabilities of the state action sequences occurring in both the policies.

1.6 Section 5.6 - Off-Policy Monte Carlo Control

- The policy that is to be improved (*estimation policy*) might in general not be the same as the policy which has been used to generate the episodes (*behavior policy*).
- The algorithm given in this section is easy to follow, but remember that π is a deterministic policy and hence the nuances. The following notes are in the PDF and attached here for convenience.
    - ⋆ We start tracking the probability part only after the two policies choose different actions. Thus, all actions after τ will be the same. Note that it says arbitrary "deterministic" policies.. Hence, there will be some τ before which the two policies will choose different actions.

- ⋆ Note here that pi is a deterministic policy and π^f will not be deterministic as it is an ϵ soft policy. In the numerator, we would technically want $\pi(s, a)$, but since it is deterministic, the probability will be exactly 1.

- It will be beneficial if the *behavior policy* π^f is ϵ-soft as the episode can be used to evaluate multiple other policies.

- A potential problem is that this method only learns from the tails of episodes, after the last non-greedy action. If non-greedy actions are frequent, then learning will be very slow, particularly for states appearing in the early portions of long episodes.

1.7 Section 5.7 - Incremental Implementation

- Like the moving averages considered, we can also have incremental implementations for weighted averages of the form $\frac{\sum w_k \times R_k}{\sum w_k}$. By keeping track of $\sum w_k$ and the weighted average, we will be able to calculate the quantities incrementally.

- Note the *Monte-Carlo* methods **do not** *bootstrap*. They use the returns at every state to form the estimates rather than using estimates of its successor states.