Reinforcement Learning With Hado van Hasselt

Yaro Kazakov

September 18, 2024

Abstract

These are study notes to prepare myself for PhD applications.

1 Lecture 5 - Monte Carlo Methods

Unlike the previous chapter, here we do not assume complete knowledge of the environment. Monte Carlo methods **require only experience—sample sequences of states, actions, and rewards from actual or simulated interaction with an environment.** Learning from actual experience is striking because **it requires no prior knowledge of the environment's dynamics (transition matrix)**, yet can still attain optimal behavior. Although a model is required, **the model need only generate sample transitions**, not the complete probability distributions of all possible transitions that is required for dynamic programming (DP).

Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging sample returns. The term "Monte Carlo" is often used more broadly for any estimation method whose operation involves a significant random component.

Doesn't assume the states are Markov.

Only work for episodic MDPs (finite, terminal state). Requires each episode to terminate.

1.1 Monte Carlo Prediction

Two types - first-visit Monte Carlo and every-visit MC. . The first-visit MC method estimates $v_\pi(s)$ as the average of the returns following first visits to s, whereas the every-visit MC method averages the returns following all visits to s.

MC basically just computes empirical mean return.

1.1.1 First-visit MC on Policy Evaluation

Do it for just one state.

The simulation is under the policy π. Input: a policy π to be evaluated.

1. Initialize: V(s) $\in$ R, arbitrarily. $N(s) = 0$ (number of times visited a state). $G(s) = 0(reward, canbealist)$.
2. Loop
   (a) Sample one episode: i = $s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}...$
   (b) Define $G_t = R_{t+1,i} + \gamma * R_{t+2,i} + \cdots + \gamma^{T-1} * R_{T,i}$
   (c) For each state state s visited in episode i
      i. For **the first** time t that state s is visited in episode i

A. Increment counter of total first visits: N(s) = N(s) + 1
B. Increment total return G(s) = G(s) + $G_{i,t}$
C. Update estimate $V^{\pi}(s) = G(s)/N(s)$

Each episode estimation is iid because each sample generation is independent and you only use the return from the first time you saw that state. Therefore the value-state estimation is **unbiased**

1.1.2 Every-visit MC on Policy Evaluation

Do it for just one state.

The simulation is under the policy π. Input: a policy π to be evaluated.

1. Initialize: V(s) ∈ R, arbitrarily. $N(s) = 0$ (number of times visited a state). $G(s) = 0(reward, canbealist)$.
2. Loop
   (a) Sample one episode: i = $s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}...$
   (b) Define $G_t = R_{t+1,i} + \gamma * R_{t+2,i} + \cdots + \gamma^{T-1} * R_{T,i}$
   (c) For each state state s visited in episode i
      i. For **every** time t that state s is visited in episode i
         A. Increment counter of total first visits: N(s) = N(s) + 1
         B. Increment total return G(s) = G(s) + $G_{i,t}$
         C. Update estimate $V^{\pi}(s) = G(s)/N(s)$

The episode estimation is not iid because the rewards from the same episode are correlated. The estimation of the value-state is **biased**. However, it usually is more consistent (lower var) and has better MSE.

1.1.3 Incremental MC

Same as above but the last steps B. and C. are slightly different. After the N(s) is incremented the last step is: $V^{\pi}(s) = V^{\pi}(s) * (N(s) - 1)/N(s) + G_{i,t}/N(s) = V^{\pi}(s) + 1/N(s) * (G_{i,t} + V^{\pi}(s))$

The 1/N(s), like in Chapter 2 is the learning rate. If it is set to constant, the rate discounts the older data more (puts more emphasis on the later data.).

1.1.4 Backup Diagram of MC

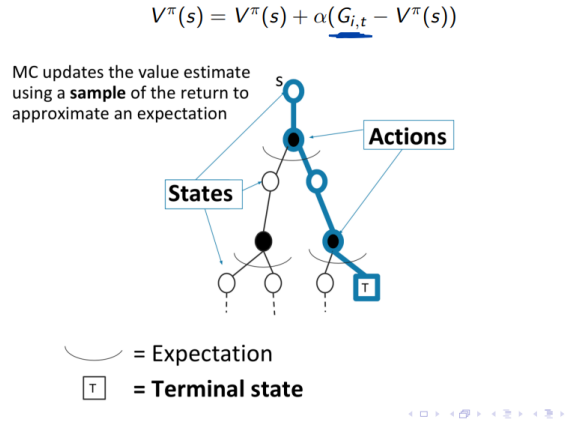

Figure 1: MC backup paths

1.1.5 Exercise 5.1

Consider the diagrams on the right in Figure 5.1. Why does the estimated value function jump up for the last two rows in the rear?

The policy is set such that you stick at 20/21 and hit below those values. If you have 20/21 you'are very likely to beat the dealer. However, if you're at 19 for example, you're very likely to lose by overdrawing and going bust. At 19, there is only one card that gets you win - 2. Same applies to the other lower numbers.

Why does it drop off for the whole last row on the left?

The dealer has a usable ace. The chances of the dealer win are larger given his ace on the table.

Why are the frontmost values higher in the upper diagrams than in the lower?

Less likely to go bust with the usable ace. 10 cards won't get you out of the game. With non usable ace, only at least 9 cards won't get you out of the game.

1.1.6 Exercise 5.2

Suppose every-visit MC was used instead of first-visit MC on the blackjack task. Would you expect the results to be very different? Why or why not?

The results would be more biased for the state-value estimations if every-visit MC was used. Since the game is simulated on every episode and is independent, the results would not be much different.

1.1.7 Exercise 5.3

What is the backup diagram for Monte Carlo estimation of q_π?

The same as on page 59 but the path would along one particular branch. MC averages over each outcome. We would have the diagram from page 95 but with both states and action values.

1.1.8 Exercise 5.4

Indeed, instead of averaging over the list a better strategy would be to estimate $Q(s, a)$ using no additional memory:

$Q(s, a) = Q(s, a) + \frac{1}{N(s,a)}(G_{i,t} - Q(s, a))$

With the fraction being the learning rate. This is also called incremental MC. The mean $Q(s, a)$ is maintained and the count is updated whenever the right Q is met in the sample.

1.2 Monte Carlo Control and Estimation of Action Values

Control - making decisions. Examples above just evaluate a policy without the knowledge of how the world works. Now we want to know how to make decisions in such environment and learn a policy.

This is called **MODEL-FREE CONTROL**.

If a model is not available, then it is particularly useful to estimate action values (the values of state–action pairs) rather than state values. With a model, state values alone are sufficient to determine a policy; one simply looks ahead one step and chooses whichever action leads to the best combination of reward and next state, as we did in the chapter on DP.

Without a model, however, state values alone are not sufficient.

The only complication is that many state–action 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**. To compare alternatives we need to estimate the value of all the actions from each state, not just the one we currently favor.

1.2.1 On-policy and Off-policy Learning

This leads us to two ways of model-free learning on- and off-policy.

1. On-policy
    (a) Direct Learning
    (b) Learn to estimate and evaluate a policy from experience obtained from following that policy
2. Off-policy
    (a) Learn to estimate and evaluate a policy from experience obtained from following a different policy. In other words, evaluate or improve policy different from that used to generate the data.