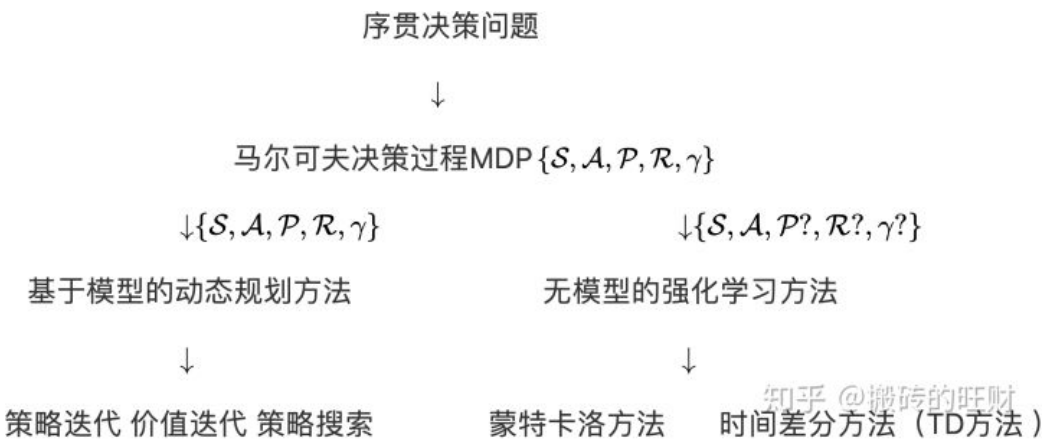
Reinforcement Learning
Model Free Prediction

Lu Hong

1 Introduction

In Dynamic Programming Lecture, we can solve a MDP using given model using DP method, by iteration to achieve the optimal value function. There exists two methods: policy gradient and value gradient.

In this chapter, we don't know the model. It means we don't know $P^a_{ss'}$and R^a_s, in order to learn from environment, agent has to interact with the environment, generating several episodes. This lecture illustrate MC method, and extends to the TD algorithm, along with TD(λ) method.



2 Monte-Carlo Reinforcement Learning

2.1 Overview

- learn directly from *episodes of experience*
- model-free: no knowledge of MDP transitions / rewards

- learns from complete episodes: **no bootsrapping**
- use the simplest possible idea: value = mean return
- all episodes must be terminated

2.2 Monte-Carlo Policy Evaluation

Given a π, an episode can be represented as follows:
$S_1, A_1, R_2, S_2, A_2, ..., S_t, A_t, R_{t+1}, ..., S_T$ π
Recall that $v_\pi(s) = \mathbb{E}[G_t|S_t = s]$, where $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$
Now, in MC method, if the state s appears multiple times in one episode, like t_1, t_2, how can we evaluate the policy?

2.2.1 First-Visit Monte-Carlo Policy Evaluation

First-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- The first time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- By law of large numbers, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

Every sampled trajectory is i.i.d.

2.2.2 Every-Visit Monte-Carlo Policy Evaluation

2.3 Incremental Monte-Carlo Updates

We compute the mean value incrementally, for each state S_t with return G_t

$$N(S_t) \leftarrow N(S_t) + 1$$
$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

Every-Visit Monte-Carlo Policy Evaluation

- To evaluate state s
- Every time-step t that state s is visited in an episode,
- Increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Value is estimated by mean return $V(s) = S(s)/N(s)$
- Again, $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$

In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes. From my perspective, it means forget $N(S_t)$ in mathmatics, turning α into a parameter likes learning rate.

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

Apparently, MC method requires episodes to be terminated and it also requires much time to

3 TD Learning
Temporal-Difference Learning

Here, bootsrapping means using TD-target to replace G_t which in TD(0), TD target $= R_{t+1} + \gamma V(S_{t+1})$

Where MC has high variance, zero bias. TD has low variance, some bias.

3.1 Batch MC and TD

We have to use limited trials to evaluate the policy, other than infinite trials. So using batch method can make us generate limited experience and evaluate the policy.

Temporal-Difference Learning

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

For example, in k episodes

$$s_1^1, a_1^1, r_2^1, ..., s_{T_1}^1$$
$$...$$
$$s_1^K, a_1^K, r_2^K, ..., s_{T_K}^K$$

David Silver uses AB example to illustrate it.

By applying MC and TD method to this problem, we can get two different value of $V(A)$

3.2 Certainty Equivalence

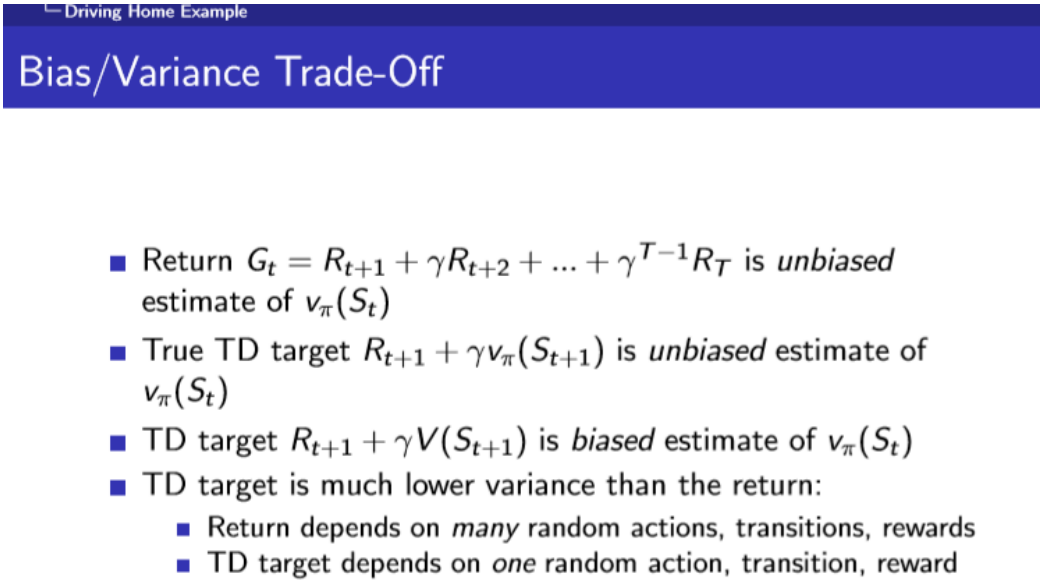
MC converges to solution with minimum mean-squared error, best fit to the observed returns.

$$\sum_{k=1}^{K}\sum_{t=1}^{T_k}(G_t^k - V(s_t^k))^2$$

TD(0) converges to solution of **max likelihood Markov model**. Solution to the MDP$< S, A, \hat{P}, \hat{R}, \gamma >$ that best fits the data

$$\hat{P}_{s,s'}^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{R}_s^a = \frac{1}{N(s,a)}\sum_{k=1}^{K}\sum_{t=1}^{T_k}\mathbf{1}(s_t^k, a_t^k = s, a)r_t^k$$



3.3 Monte-Carlo Backup

4 TD(λ)

So we use TD(λ) to bridge between TD(0) and MC method, we let TD-target to look n-step into the future in an episode.

Now, consider n-Step return. Consider the following n-step returns for n=1,2,∞:

$$\begin{aligned} n = 1 (TD) \quad & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\ n = 2 \quad & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\ ... & \\ n = \infty (MC) \quad & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T \end{aligned}$$

So we define n-step return as

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

n-step TD learning

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{(n)} - V(S_t))$$

Can we efficiently combine information from all time-steps? So we introduced $\lambda - return$

AB Example

Two states A, B; no discounting; 8 episodes of experience

$A, 0, B, 0$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 1$
$B, 0$

What is $V(A), V(B)$?

- The $\lambda - return$ G_t^λ combines all n-step return $G_t^{(n)}$
- Using weight $(1-\lambda)\lambda^{n-1}$, as $G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$
- Forward-view TD(λ), as $V(S_t) \leftarrow V(S_t) + \alpha(G_t^\lambda - V(S_t))$

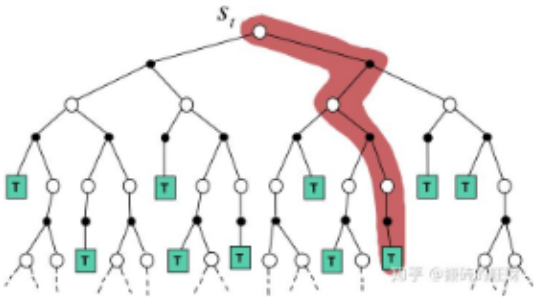
4.1 Back-view TD(λ)

4.1.1 Eligibility Traces

- Frequency heuristic: assign credit to most frequent states
- Recency heuristic: assign credit to most recent states
- Eligibility traces combine both heuristics

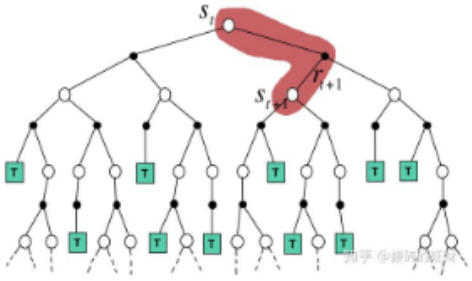
$$E_0(s) = 0$$
$$E_t(s) = \gamma\lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$

MC $V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$



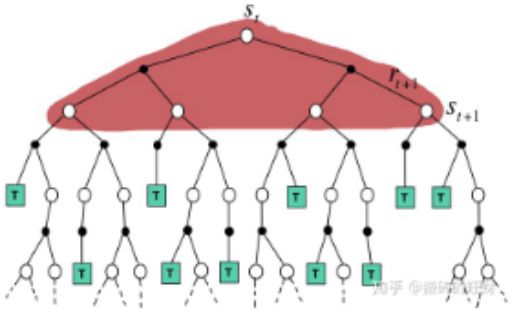
采样，一次完整经历，用实际收获更新状态预估价值

TD $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$

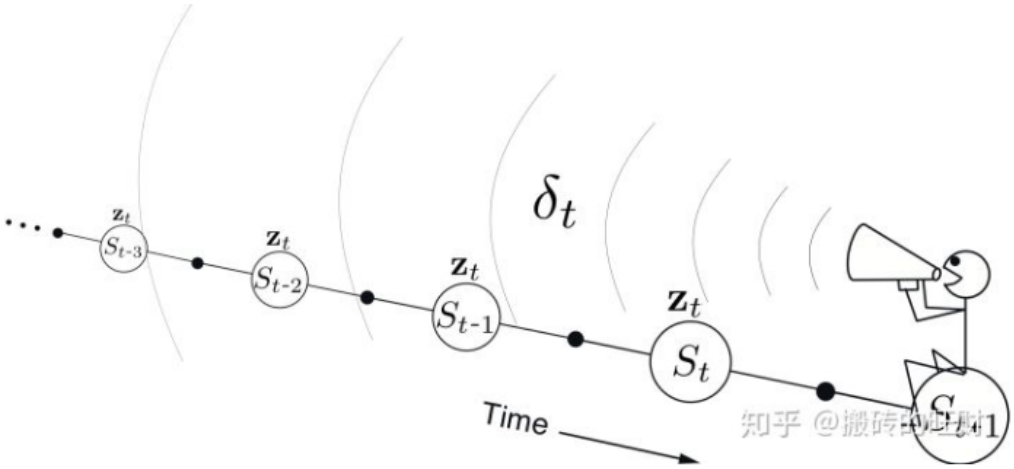


采样，经历可不完整，用后续状态的预估状态价值预估收获再更新预估价值

DP $V(S_t) \leftarrow \mathbb{E}_\pi\left[R_{t+1} + \gamma V\left(S_{t+1}\right)\right]$



没有采样，根据完整模型，依靠预估数据更新状态价值



后向视角使用了我们刚刚定义的资格迹，每个状态 s 都保存了一个资格迹。我们可以将资格迹理解为一个权重，状态 s 被访问的时间离现在越久远，其对于值函数的影响就越小，状态 s 被访问的次数越少，其对于值函数的影响也越小。