Lecture 5: Model-free Prediction

Bolei Zhou

The Chinese University of Hong Kong

bzhou@ie.cuhk.edu.hk

January 21, 2020

This Week's Plan

1. Last Week
   1. Policy evaluation, policy iteration and value iteration for solving a known MDP
2. Today's lecture
   1. Model-free prediction: Estimate value function of an unknown MDP
3. Tomorrow's lecture
   1. Model-free control: Optimize value function of an unknown MDP

Review on previous control in MDP

1. When the MDP is known?
   1. Both R and P are exposed to the agent
   2. Therefore we can run policy iteration and value iteration
2. Policy iteration: Given a known MDP, compute the optimal policy and the optimal value function
   1. Policy evaluation: iteration on the Bellman expectation backup

$$v_i(s) = \sum_{a \in \mathcal{A}} \pi(a|s)\Big(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_{i-1}(s')\Big)$$

   2. Policy improvement: greedy on action-value function q

$$\begin{aligned} q_{\pi_i}(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v_{\pi_i}(s') \\ \pi_{i+1}(s) &= \arg\max_a q_{\pi_i}(s,a) \end{aligned}$$

Review on value iteration

1. Value iteration: Given a known MDP, compute the optimal value function
2. Iteration on the Bellman optimality backup

$$v_{i+1}(s) \leftarrow \max_{a \in \mathcal{A}} R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_i(s')$$

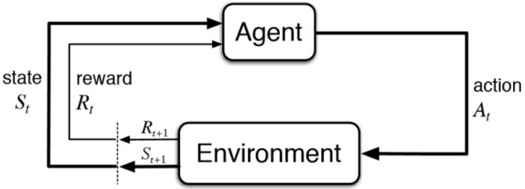
3. To retrieve the optimal policy after the value iteration:

$$\pi^*(s) \leftarrow \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v_{end}(s') \tag{1}$$

RL with knowing how the world works

1. Both of the classic policy iteration and value iteration assume known dynamics and rewards of the environment
    1. It
2. In a lot of real-world problems, MDP model is either unknown or known by too big or too complex to use
    1. Atari Game, Game of Go, Helicopter, Portfolio management, etc
3. Model-free RL can solve these problems
    1. Model-free prediction
    2. Model-free control

Model-free prediction: Policy evaluation by interaction



1. No more oracle or shortcut of the known transition dynamics and reward function
2. Trajectories/episodes are collected by the interaction of the agent and the environment
3. Each trajectory/episode contains $\{S_1, A_1, R_1, S_2, A_2, R_2, ..., S_T, A_T, R_T\}$

Model-free prediction: policy evaluation without the access to the model

1. Estimating the expected return of a particular policy if we don't have access to the MDP models
    1. Monte Carlo policy evaluation
    2. Temporal Difference (TD) learning

Monte-Carlo Policy Evaluation

1. Return: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$ under policy π
2. $v_\pi(s) = \mathbb{E}_{T\sim\pi}[G_t | s_t = s]$, thus expectation over trajectories T generated by following π
3. MC simulation: we can simply sample a lot of trajectories, compute the actual returns for all the trajectories, then average them
4. MC policy evaluation uses empirical mean return instead of expected return
5. MC does not require MDP dynamics/rewards, no bootstrapping, and does not assume state is Markov.
6. Only applied to episodic MDPs (each episode terminates)

Monte-Carlo Policy Evaluation

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

Incremental Mean

Mean from the average of samples $x_1, x_2, ...$

$$
\begin{aligned}
\mu_k =& \frac{1}{k}\sum_{j=1}^{k} x_j \\
=& \frac{1}{k}\Big(x_k + \sum_{j=1}^{k-1} x_j\Big) \\
=& \frac{1}{k}(x_k + (k-1)\mu_{k-1}) \\
=& \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})
\end{aligned}
$$

Incremental MC Updates

1. Collect one episode $(S_1, A_1, R_1, ..., S_t)$
2. For each state s_t with computed return G_t

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

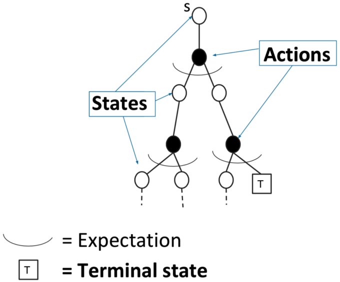
3. Or use a running mean (old episodes are forgotten). Good for non-stationary problems.

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

Difference between DP and MC for policy evaluation

1. DP computes v_i by bootstrapping the rest of the expected return by the value estimate v_{i-1}
2. Iteration on Bellman expectation backup:

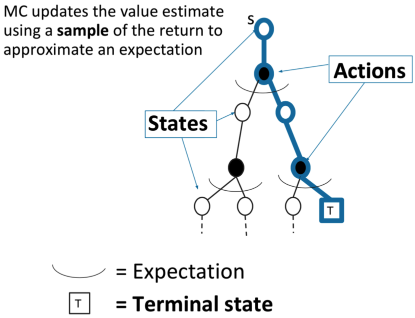
$$v_i(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s)\Big(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_{i-1}(s')\Big)$$



Difference between DP and MC for policy evaluation

1. MC updates the empirical mean return with one sampled episode

$$v(S_t) \leftarrow v(S_t) + \alpha(G_{i,t} - v(S_t))$$



Advantages of MC over DP

1. MC works when the environment is unknown
2. Working with sample episodes has a huge advantage, even when one has complete knowledge of the environment's dynamics, for example, transition probability is complex to compute
3. Cost of estimating a single state's value is independent of the total number of states. So you can sample episodes starting from the states of interest then average returns

Temporal-Difference (TD) Learning

1. TD methods learn directly from episodes of experience
2. TD is model-free: no knowledge of MDP transitions/rewards
3. TD learns from incomplete episodes, by bootstrapping

Temporal-Difference (TD) Learning

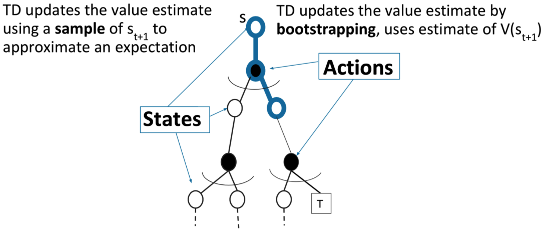
1. Objective: learn v_π online from experience under policy π
2. Simplest TD algorithm: TD(0)
   1. Update $v(S_t)$ toward estimated return $R_{t+1} + \gamma v(S_{t+1})$

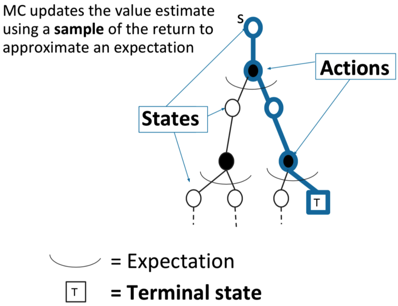
$$v(S_t) \leftarrow v(S_t) + \alpha(R_{t+1} + \gamma v(S_{t+1}) - v(S_t))$$

3. $R_{t+1} + \gamma v(S_{t+1})$ is called TD target
4. $\delta_t = R_{t+1} + \gamma v(S_{t+1}) - v(S_t)$ is called the TD error
5. Comparison: Incremental Monte-Carlo
   1. Update $v(S_t)$ toward actual return G_t given an episode i

$$v(S_t) \leftarrow v(S_t) + \alpha(G_{i,t} - v(S_t))$$

Advantages of TD over MC





Comparison of TD and MC

1. TD can learn online after every step
2. MC must wait until end of episode before return is known

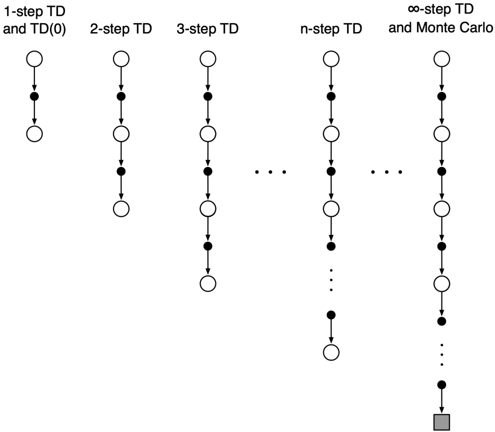
3. TD can learn from incomplete sequences
4. MC can only learn from complete sequences

5. TD works in continuing (non-terminating) environments
6. MC only works for episodic (terminating) environments

7. TD exploits Markov property, more efficient in Markov environments
8. MC does not exploit Markov property, more effective in non-Markov environments

n-step TD

1. *n*-step TD methods that generalize both one-step TD and MC.
2. We can shift from one to the other smoothly as needed to meet the demands of a particular task.



n-step TD prediction

1. Consider the following n-step returns for $n = 1, 2, \infty$

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

2. Thus the n-step return is defined as

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

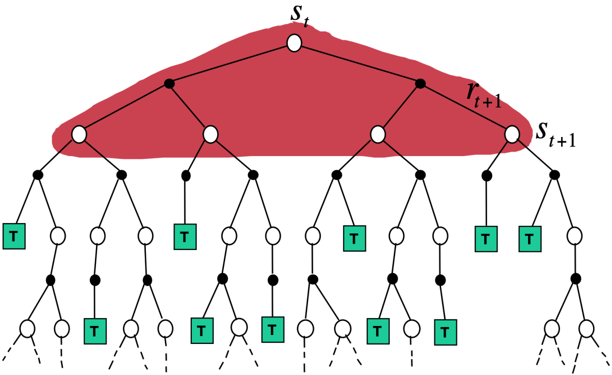
3. n-step TD: $v(S_t) \leftarrow v(S_t) + \alpha\Big(G_t^n - v(S_t)\Big)$

Bootstrapping and Sampling for DP, MC, and TD

1. Bootstrapping: update involves an estimate
    1. MC does not bootstrap
    2. DP bootstraps
    3. TD bootstraps
2. Sampling: update samples an expectation
    1. MC samples
    2. DP does not sample
    3. TD samples

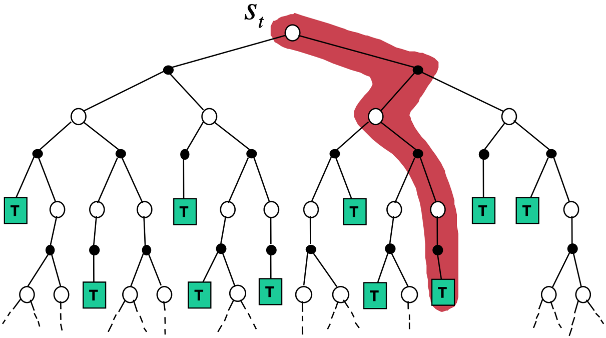
Unified View: Dynamic Programming Backup

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



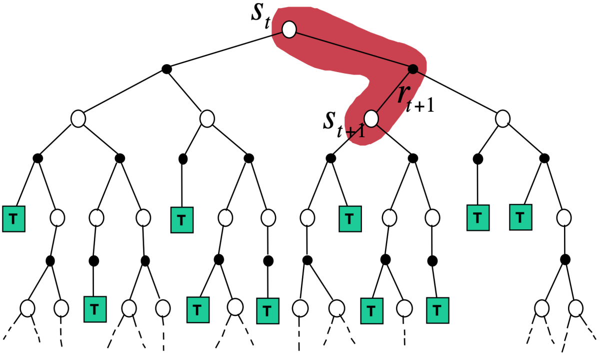
Unified View: Monte-Carlo Backup

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

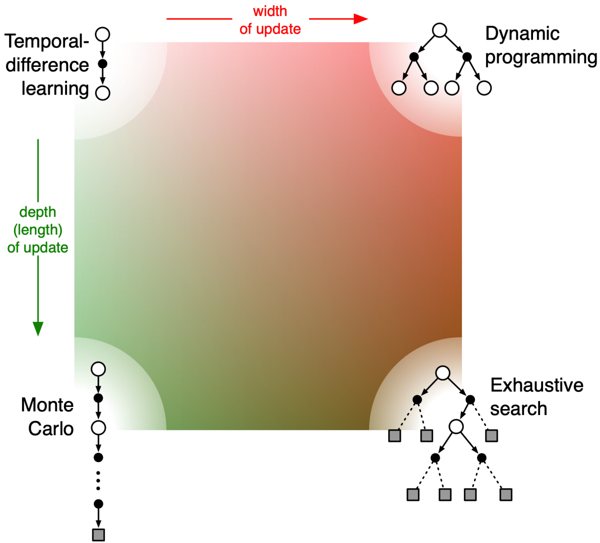


Unified View: Temporal-Difference Backup

$$TD(0) : v(S_t) \leftarrow v(S_t) + \alpha(R_{t+1} + \gamma v(s_{t+1}) - v(S_t))$$



Unified View of Reinforcement Learning



Summary

1. This time: Model-free prediction
    1. Evaluate the state value without knowing the MDP model, by only interacting with the environment
2. Next time: Model-free control