Lecture 6: Model-free Control

Bolei Zhou

The Chinese University of Hong Kong

bzhou@ie.cuhk.edu.hk

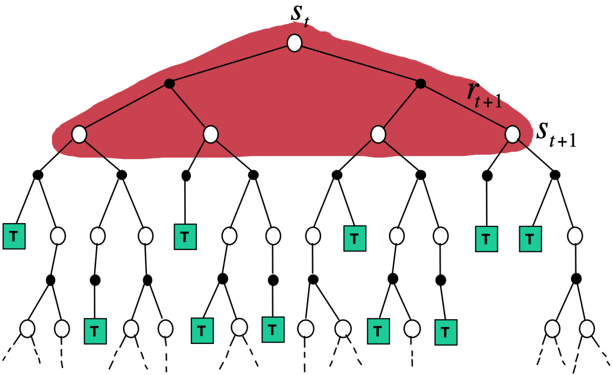
January 22, 2020

Today's Plan

1. Last Time
   1. Model-free prediction: Estimate the value function of an unknown MDP
   2. Monte-Carlo (MC) and Temporal Difference (TD)
2. This Time
   1. Model-free control: Optimize the value function of an unknown MDP
   2. Generalized Policy Iteration (GPI) with MC and TD

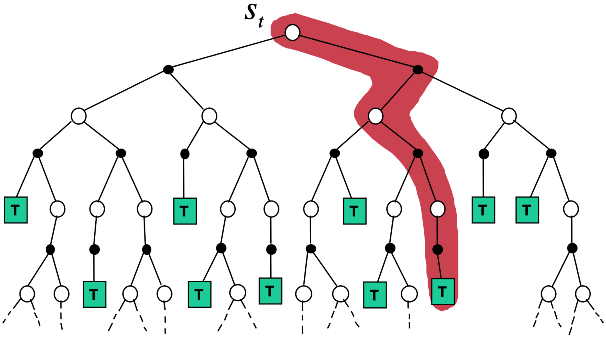
Review: Dynamic Programming Backup

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



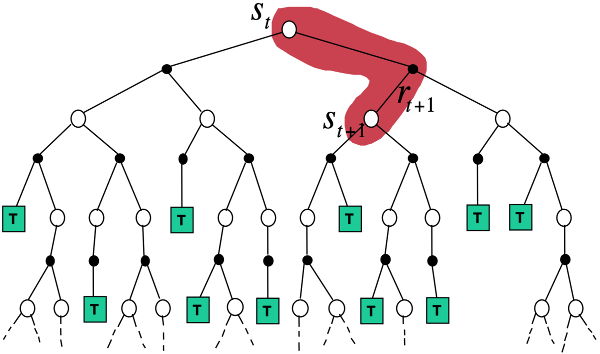
Review: Monte-Carlo Backup

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



Review: Temporal-Difference Backup

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



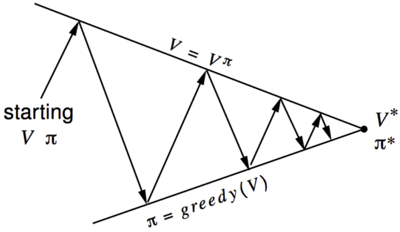
Control: Optimize the policy for a MDP

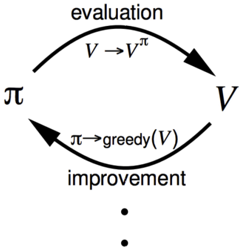
1. Model-based control: optimize the value function with known MDP
2. Model-free control: optimize the value function with unknown MDP
3. Many model-free RL examples: Go, robot locomation, patient treatment, helicopter control, Atari, Starcraft

Policy Iteration

1. Iterate through the two steps:
   1. Evaluate the policy π (computing v given current π)
   2. Improve the policy by acting greedily with respect to v_π

$$\pi' = \text{greedy}(v_\pi) \tag{1}$$





Policy Iteration for a Known MDP

1. compute the state-action value of a policy π:

$$q_{\pi_i}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) v_{\pi_i}(s')$$

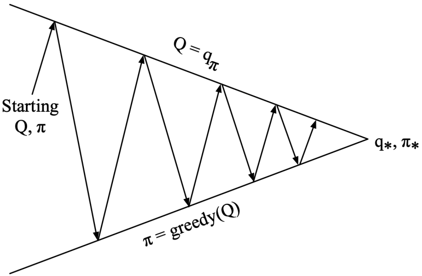
2. Compute new policy π_{i+1} for all $s \in \mathcal{S}$ following

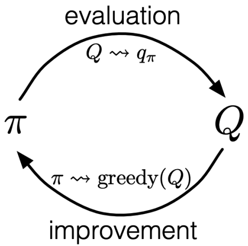
$$\pi_{i+1}(s) = \arg\max_a q_{\pi_i}(s, a) \tag{2}$$

3. Problem: What to do if there is neither $R(s, a)$ nor $P(s'|s, a)$ known/available?

Generalized Policy Iteration with Action-Value Function

Monte Carlo version of policy iteration





1. Policy evaluation: Monte-Carlo policy evaluation $Q = q_\pi$
2. Policy improvement: Greedy policy improvement?

$$\pi(s) = \arg\max_a q(s, a)$$

Monte Carlo with Exploring Starts

1. One assumption to obtain the guarantee of convergence in PI: Episode has exploring starts
2. Exploring starts can ensure all actions are selected infinitely often

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
  $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
  $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
  Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
  Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \dots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \dots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Monte Carlo with ϵ-Greedy Exploration

1. Trade-off between exploration and exploitation (we will talk about this in later lecture)
2. ϵ-Greedy Exploration: Ensuring continual exploration
   1. All actions are tried with non-zero probability
   2. With probability $1 - \epsilon$ choose the greedy action
   3. With probability ϵ choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/|\mathcal{A}| + 1 - \epsilon & \text{if } a^* = \arg\max_{a \in \mathcal{A}} Q(s, a) \\ \epsilon/|\mathcal{A}| & \text{otherwise} \end{cases}$$

Monte Carlo with ϵ-Greedy Exploration

1. Policy improvement theorem: For any ϵ-greedy policy π, the ϵ-greedy policy π' with respect to q_π is an improvement, $v_{\pi'}(s) \geq v_\pi(s)$

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\
&= \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \max_a q_\pi(s, a) \\
&\geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \frac{\epsilon}{|\mathcal{A}|}}{1 - \epsilon} q_\pi(s, a) \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s)
\end{aligned}
$$

Therefore, $v_{\pi'}(s) \geq v_\pi(s)$ from the policy improvement theorem

Monte Carlo with ϵ-Greedy Exploration

Algorithm 1

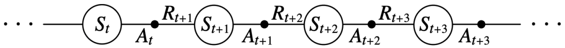
Initialize $Q(S, A) = 0, N(S, A) = 0, \epsilon = 1, k = 1$
$\pi_k = \epsilon$-greedy(Q)
loop
    Sample k-th episode $(S_1, A_1, R_2, ..., S_T) \sim \pi_k$
    for each state S_t and action A_t in the episode **do**
        $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$
    end for
    $k \leftarrow k + 1, \epsilon \leftarrow 1/k$
    $\pi_k = \epsilon$-greedy(Q)
end loop

MC vs. TD for Prediction and Control

1. Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
    1. Lower variance
    2. Online
    3. Incomplete sequences
2. So we can use TD instead of MC in our control loop
    1. Apply TD to $Q(S, A)$
    2. Use ϵ-greedy policy improvement
    3. Update every time-step rather than at the end of one episode

Recall: TD Prediction

1. An episode consists of an alternating sequence of states and state–action pairs:



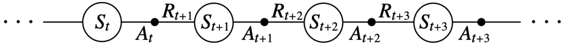
2. TD(0) method for estimating the value function $V(S)$

$$A_t \leftarrow \text{action given by } \pi \text{ for S}$$
$$\text{Take action } A_t\text{, observe } R_{t+1} \text{ and } S_{t+1}$$
$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

3. How about estimating action value function $Q(S)$?

Sarsa: On-Policy TD Control

1. An episode consists of an alternating sequence of states and state–action pairs:



2. ϵ-greedy policy for one step, then bootstrap the action value function:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big]$$

3. The update is done after every transition from a nonterminal state S_t
4. TD target $\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$

Sarsa algorithm

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

n-step Sarsa

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

$$\begin{aligned} n = 1 (Sarsa) q_t^{(1)} =& R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \\ n = 2 \qquad\qquad q_t^{(2)} =& R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}, A_{t+2}) \\ &\vdots \\ n = \infty (MC) \quad q_t^{\infty} =& R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-t-1} R_T \end{aligned}$$

2. Thus the n-step Q-return is defined as

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, A_{t+n})$$

3. n-step Sarsa updates Q(s,a) towards the n-step Q-return:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big(q_t^{(n)} - Q(S_t, A_t) \Big)$$

On-policy Learning and Off-policy learning

1. On-policy learning: Learn about policy π from the experience sampled from π
   1. Behave non-optimally in order to explore all actions, then reduce the exploration. e.g., ϵ-greedy
2. Another solution is to use two different polices:
   1. one is learned about and becomes the optimal policy
   2. the other one is more exploratory and is used to generate behavior
3. Off-policy learning: Learn about policy π from the experience sampled from another policy b
   1. π: target policy
   2. b: behavior policy

Off-policy Learning

1. Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$, while following behaviour policy $\mu(a|s)$

$$S_1, A_1, R_2, ..., S_T \sim \mu$$
$$\text{Update } \pi \text{ using } S_1, A_1, R_2, ..., S_T$$

2. Why is this important?
   1. Learn from observing humans or other agents
   2. Re-use experience generated from old policies $\pi_1, \pi_2, ..., \pi_{t-1}$
   3. Learn about optimal policy while following exploratory policy

Importance Sampling

1. Estimate the expectation of a function

$$E_{x\sim P}[f(x)] = \int f(x)P(x)dx \approx \frac{1}{n}\sum_i f(x_i)$$

2. But sometimes it is difficult to sample x from $P(x)$, then we can sample x from another distribution $Q(x)$, then correct the weight

$$\begin{aligned}\mathbb{E}_{x\sim P}[f(x)] &= \int P(x)f(x)dx \\ &= \int Q(x)\frac{P(x)}{Q(x)}f(x)dx \\ &= \mathbb{E}_{x\sim Q}\left[\frac{P(x)}{Q(x)}f(x)\right] \approx \frac{1}{n}\sum_i \frac{P(x_i)}{Q(x_i)}f(x_i)\end{aligned}$$

Importance Sampling for Off-Policy RL

1. Estimate the expectation of return using trajectories sampled from another policy (behavior policy)

$$
\begin{aligned}
\mathbb{E}_{T\sim\pi}[g(T)] &= \int P(T)g(T)dT \\
&= \int Q(T)\frac{P(T)}{Q(T)}g(T)dT \\
&= \mathbb{E}_{T\sim\mu}\Big[\frac{P(T)}{Q(T)}g(T)\Big] \\
&\approx \frac{1}{n}\sum_i \frac{P(T_i)}{Q(T_i)}g(T_i)
\end{aligned}
$$

Importance Sampling for Off-Policy Monte Carlo

1. Generate episode from behavior policy μ and compute the generated return G_t

$$S_1, A_1, R_2, ..., S_T \sim \mu$$

2. Weight return G_t according to similarity between policies
   1. Multiply importance sampling corrections along whole episode

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

3. Update value towards correct return

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\pi/\mu} - V(S_t))$$

Importance Sampling for Off-Policy TD

1. Use TD targets generated from μ to evaluate π
2. Weight TD target $R + \lambda V(S')$ by importance sampling
3. Only need a single importance sampling correction

$$V(S_t) \leftarrow V(S_t) + \alpha\Big(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}\big(R_{t+1} + \lambda V(S_{t+1})\big) - V(S_t)\Big)$$

4. Policies only need to be similar over a single step

Q Learning

1. Off-policy learning of action values $Q(s,a)$
2. No importance sampling is needed
3. Next action is chosen using behavior policy $A_{t+1} \sim \mu(.|S_t)$. However, we consider alternative action $A' \sim \pi(.|S_t)$
4. update $Q(S_t, A_t)$ towards value of alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t)\big)$$

Off-Policy Control with Q-Learning

1. We allow both behavior and target policies to improve
2. The target policy π is greedy on $Q(s,a)$

$$\pi(S_{t+1}) = \arg\max_{a'} Q(S_{t+1}, a')$$

3. The behavior policy μ is ϵ-greedy on $Q(s,a)$
4. Thus Q-learning target:

$$\begin{aligned} R_{t+1} + \gamma Q(S_{t+1}, A') =& R_{t+1} + \gamma Q(S_{t+1}, \arg\max Q(S_{t+1}, a')) \\ =& R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') \end{aligned}$$

5. Thus the Q-Learning update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

Q-learning algorithm

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Comparison of Sarsa and Q-Learning

1. Sarsa: On-Policy TD control

   Choose action A_t from S_t using policy derived from Q with ϵ-greedy

   Take action A_t, observe R_{t+1} and S_{t+1}

   Choose action A_{t+1} from S_{t+1} using policy derived from Q with ϵ-greedy

   $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big]$$

2. Q-Learning: Off-Policy TD control

   Choose action A_t from S_t using policy derived from Q with ϵ-greedy

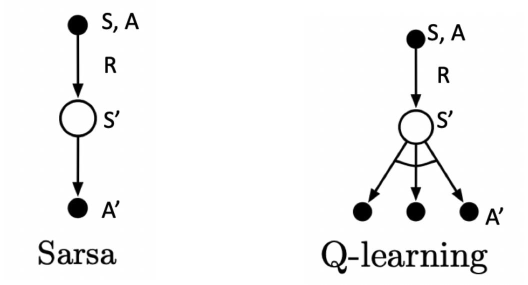
   Take action A_t, observe R_{t+1} and S_{t+1}

   Take action A_{t+1} from S_{t+1} using policy derived from Q with greedy

   $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

Comparison of Sarsa and Q-Learning

1. Backup diagram for Sarsa and Q-learning

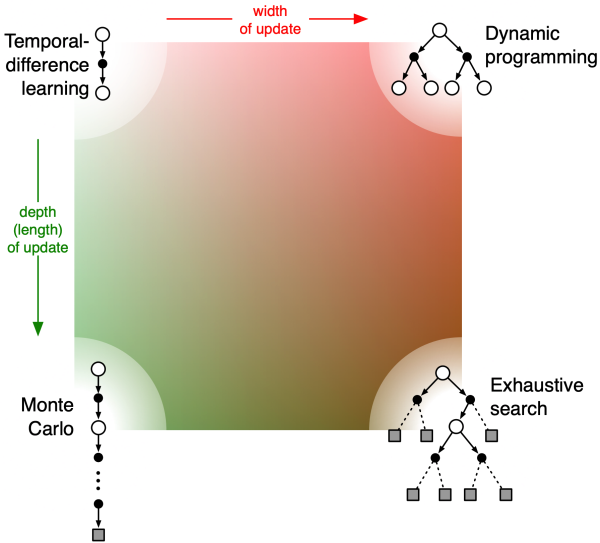


Summary of DP and TD

Expected Update (DP)	**Sample Update (TD)**
Iterative Policy Evaluation $V(s) \leftarrow \mathbb{E}[R + \gamma V(S')\|s]$	TD Learning $V(S) \xleftarrow{\alpha} R + \gamma V(S')$
Q-Policy Iteration $Q(S, A) \leftarrow \mathbb{E}[R + \gamma Q(S', A')\|s, a]$	Sarsa $Q(S, A) \xleftarrow{\alpha} R + \gamma Q(S', A')$
Q-Value Iteration $Q(S, A) \leftarrow \mathbb{E}[R + \gamma \max_{a' \in \mathcal{A}} Q(S', A')\|s, a]$	Q-Learning $Q(S, A) \xleftarrow{\alpha} R + \gamma \max_{a' \in \mathcal{A}} Q(S', a')$

where $x \xleftarrow{\alpha} y$ is defined as $x \leftarrow x + \alpha(y - x)$

Unified View of Reinforcement Learning



Sarsa and Q-Learning Example

https://github.com/cuhkrlcourse/RLexample/tree/master/modelfree

To Do for the CNY

1. Reinforce yourself with Chapter 1 to Chapter 8 (done)
    1. We will get into Part II: Approximate Solution Methods after CNY
2. Finish your Assignment 1