UNIVERSITY OF TORONTO | Engineering

Reinforcement Learning

- ML for Optimal Decision Making

Chi-Guhn Lee
University of Toronto

1

UNIVERSITY OF TORONTO | Engineering

Markov Decision Processes

- Basic Components of MDP
- Optimality, Policy and Value Functions
- Optimality Equations
- Standard Solution Methods

2

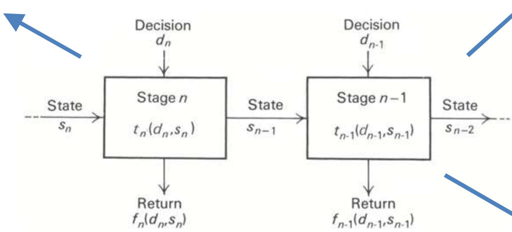
Static Optimization

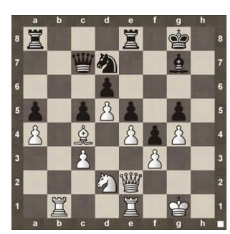
- Linear Program

$$
\begin{array}{lccccc}
\text{Minimize} & 4x_1 & + & x_2 & = & z \\
\text{Subject to} & 3x_1 & + & x_2 & \geq & 10 \\
 & x_1 & + & x_2 & \geq & 5 \\
 & x_1 & & & \geq & 3 \\
 & & x_1, & x_2 & \geq & 0.
\end{array}
$$

3

Dynamic Optimization





4

Basic Components of MDP

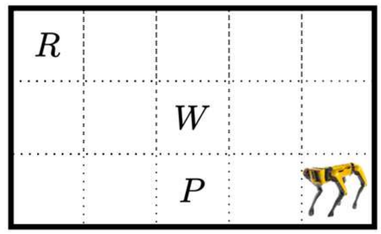
- Mathematically, MDP is a 5-tuple (S, A, R, P, γ) where
  - S is a finite set of states
  - A is a finite set of actions
  - P is a state transition probability matrix

$$P(s'|s,a) = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

  - R is a reward function

$$R(s,a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$

  - γ is a discount factor: $\gamma \in (0,1]$



UNIVERSITY OF TORONTO | Engineering

5

- Example

15	14	13	12	11
10	9	8	7	6
5	4	3	2	1

$S = \{1, 2, 3, \ldots, 15\}$

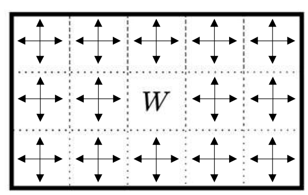
where $15 \equiv$ Goal State

$8 \equiv$ A wall (barrier)

$3 \equiv$ Pitfall (trap)

100				
		W		
		-50		

$$R(s,a) = \begin{cases} 100, & \text{if } (s,a) = (15,\ *) \\ -50, & \text{if } (s,a) = (3,\ *) \\ -1 & o/w \end{cases}$$

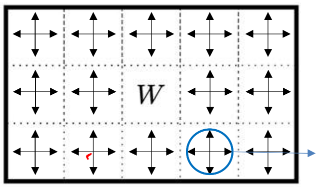


$A = \{\uparrow, \downarrow, \leftarrow, \rightarrow, \circlearrowright\}$

UNIVERSITY OF TORONTO | Engineering

6

$$P(s'|s,a) = \mathbb{P}\{S_{t+1} = s' | S_t = s, A_t = a\}$$



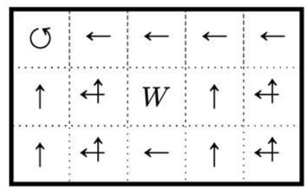
$$A = \{\uparrow, \downarrow, \leftarrow, \rightarrow, \circlearrowright\}$$

$$\begin{aligned} &\vdots \\ P(1|2,\uparrow) &= 0 \\ P(2|2,\uparrow) &= 0 \\ &\vdots \\ P(7|2,\uparrow) &= 1 \\ P(8|2,\uparrow) &= 0 \\ &\vdots \\ P(1|2,\leftarrow) &= 0 \\ P(2|2,\leftarrow) &= 0 \\ P(3|2,\leftarrow) &= 1 \\ P(4|2,\leftarrow) &= 0 \\ P(5|2,\leftarrow) &= 0 \\ &\vdots \\ &\vdots \\ P(15|15,\circlearrowright) &= 1 \end{aligned}$$

- **Policies**

A policy π is a distribution over actions given states:

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$



UNIVERSITY OF TORONTO | Engineering

7

Value Function

- Expected return when starting in state s and following π thereafter.

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

where $E_\pi[\cdot]$ denotes the expectation given π.

- Action value function
  - Expected return when taking action a from state s and follow policy π thereafter
  - This is more useful than the state value function in learning where the model is unknown

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

  - v_π is called "state-value function" of policy π, q_π is "action-value function" of policy π.

UNIVERSITY OF TORONTO | Engineering

8

Policy Evaluation

- Given

$$G_t = R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} + \gamma^3 \cdot R_{t+4} + \cdots$$
$$= R_{t+1} + \gamma \cdot (R_{t+2} + \gamma \cdot R_{t+3} + \gamma^2 \cdot R_{t+4} + \cdots)$$
$$= R_{t+1} + \gamma \cdot G_{t+1}$$

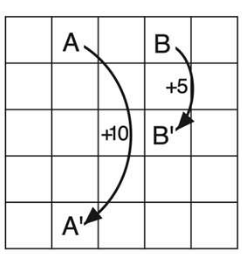
- Value functions can be computed recursively:

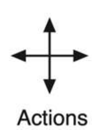
$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\Big[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\Big] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big], \quad \text{for all } s \in \mathcal{S}, \end{aligned}$$

- The last **recursive** equation is known as "the Bellman Equation" for v_π;
- Bellman equation is a system of linear equations, which can be trivially solved

9

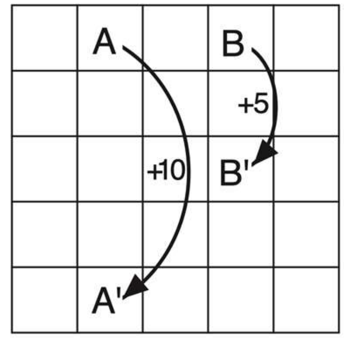
- **Example 3.5: Gridworld** Figure 3.2 (left) shows a rectangular gridworld representation of a simple finite MDP. The cells of the grid correspond to the states of the environment. At each cell, four actions are possible: `north`, `south`, `east`, and `west`, which deterministically cause the agent to move one cell in the respective direction on the grid. Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of -1. Other actions result in a reward of 0, except those that move the agent out of the special states A and B. From state A, all four actions yield a reward of $+10$ and take the agent to A$'$. From state B, all actions yield a reward of $+5$ and take the agent to B$'$.





10

- A system of equations



UNIVERSITY OF TORONTO | Engineering

11

- Value Function

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

  – $V_\pi(1, .., 25)$ when π is the equi-probable policy

Discussion:

- V(A)=8.8 $<$ R(A)
- V(B)=5.3 $>$ R(B)

UNIVERSITY OF TORONTO | Engineering

12

Policy Evaluation

- Solving a large system of equations is **not practical**
- Instead, use **iterative approximation,** which will converge to a fixed point
- The fixed point is the value of the policy (value function)

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big],
\end{aligned}
$$

$v(1) = p(a=1, s=1)(r(1,1) + \gamma \cdot v_\pi(1))$
$+ p(a=2, s=1)(r(1,2) + \gamma v_\pi(1))$
$\vdots$
$+ p(a=10, s=1)(\quad)$
$+ p(a=1, s=2)(\quad v(2))$
$+ p(a=2, s=2)(\quad v(2))$
$\vdots$
$+ p(a=10, s=20)(\quad v(20))$

- Recursion allows iterative approximation

$$
\begin{aligned}
v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big],
\end{aligned}
$$

- Contraction mapping T: $v_{k+1} \triangleq T v_k$

UNIVERSITY OF TORONTO | Engineering

13

Policy Evaluation Algorithm

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

UNIVERSITY OF TORONTO | Engineering

14

From Evaluation to Optimization

- Optimal policy

  $\pi \geq \pi_0$ if and only if v_π (s) $\geq v_{\pi_0}$ (s) for all $s \in S$. , $\forall \pi_0$

- Policy evaluation to optimal value functions
  - Given $v_\pi(s), \forall s \in S$, $v_*(s) \doteq \max_\pi v_\pi(s),$
  - Given $q_\pi(s,a), \forall s \in S, \forall a \in A$ $q_*(s,a) \doteq \max_\pi q_\pi(s,a),$

- State value function and action value function

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a)$$

UNIVERSITY OF TORONTO | Engineering

15

- **Optimality Equations**

$$v_*(s) \doteq \max_\pi v_\pi(s),$$

$$\begin{aligned} v_*(s) &\doteq \max_\pi v_\pi(s), \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma v_*(s')\big]. \end{aligned}$$

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a),$$

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big]. \end{aligned}$$

  - The problem is ………. we cannot find v_* or q_* without knowing v_* or q_*

UNIVERSITY OF TORONTO | Engineering

16

Optimization

- From **evaluation** to **optimization**
  - Given a value function v_π for an arbitrary policy π, we try to find a better one
  - Find a better policy by **1-step look-ahead greedy**
  - That is, from state s find antion a that maximizes the following:

$$\max_a \underbrace{R_t(s,a)}_{\text{Immediate reward}} + \gamma \cdot \underbrace{E\{v_\pi(s')\}}_{\text{1-step ahead return}}, \forall s \in S$$

  - Upon selecting a for any state differently than π, we have just improved π to a new policy

UNIVERSITY OF TORONTO | Engineering

17

Guarantee

- 1-step look-ahead greedy will always find a better policy
- Let a new policy π' such that $(a_t(s), \pi_{t+1}(s'))$. That is, do a_t and follow π thereafter

$$\left.\begin{aligned}
v_\pi(s) &\le q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\le \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \mid S_t = s] \\
&= \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \bigm| S_t = s\big] \\
&\le \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \bigm| S_t = s\big] \\
&\vdots \\
&\le \mathbb{E}_{\pi'}\big[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \bigm| S_t = s\big]
\end{aligned}\right\} \textbf{Policy Improvement Theorem}$$

$$= v_{\pi'}(s).$$

UNIVERSITY OF TORONTO | Engineering

18

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

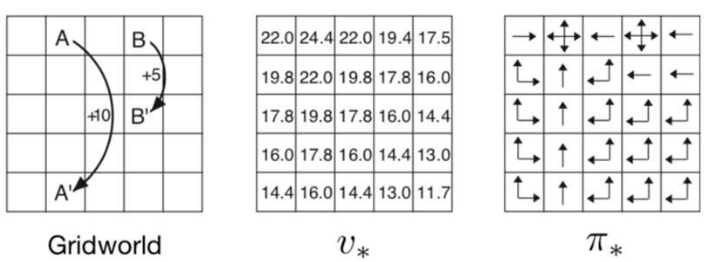
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop for each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

19

Example 3.8: Solving the Gridworld Suppose we solve the Bellman equation for v_* for the simple grid task introduced in Example 3.5 and shown again in Figure 3.5 (left). Recall that state A is followed by a reward of +10 and transition to state A′, while state B is followed by a reward of +5 and transition to state B′. Figure 3.5 (middle) shows the optimal value function, and Figure 3.5 (right) shows the corresponding optimal policies. Where there are multiple arrows in a cell, all of the corresponding actions are optimal.

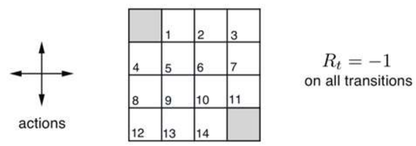


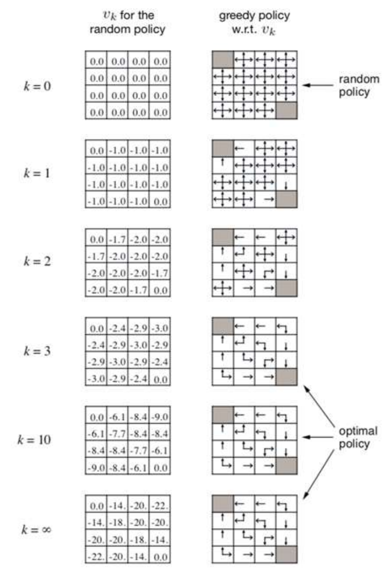
22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

Figure 3.5: Optimal solutions to the gridworld example. ■

20

- Example of Policy Iteration Algorithm





- Observations
  - Policy stops changing since k=3
  - Policy evaluation is expensive
  - Approximate evaluation?

UNIVERSITY OF TORONTO | Engineering

21

Value Iteration Algorithm

- PI to an extreme by taking **just 1 iteration per evaluation**

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

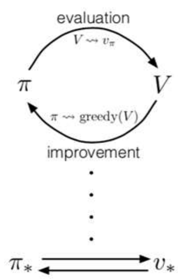
Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

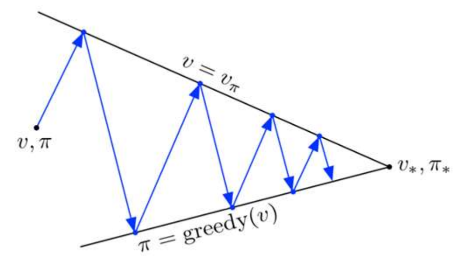
UNIVERSITY OF TORONTO | Engineering

22

Generalized Policy Iteration

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$





23

Monte Carlo Methods

24

Learning the Value Function

- Exact Method (via Dynamic Programming)
  - Requires a complete knowledge of MDP: dynamics and reward
  - Computationally expensive known as the curse of dimensionality
- Monte Carlo Learning
  - Estimation using the sample returns (Monte Carlo)
- Reinforcement Learning
  - Sampling and bootstrapping

25

Monte Carlo Prediction/Estimation

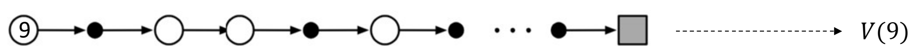
- **Averaging** to predict **value function**
- A trajectory (or episode) is given,

  - The root is a state node shown as a hollow circle
  - Solid circles are for actions
  - Square is for termination (or terminal state)
  - Arrow shows sequential relationship: A and then B, followed by C, etc.

26

- Prediction of value function

$V(1) = R_0 + R_1 + \cdots$



$V(9) = R_0 + R_1 + \cdots$

$V(1) = R_0 + R_1 + \cdots$

$\vdots$

$V(3) = R_0 + R_1 + \cdots$

UNIVERSITY OF TORONTO | Engineering

27

- (First-visit) MC prediction

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:
    $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
    $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
    Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
            Append G to $Returns(S_t)$
            $V(S_t) \leftarrow \text{average}(Returns(S_t))$

UNIVERSITY OF TORONTO | Engineering

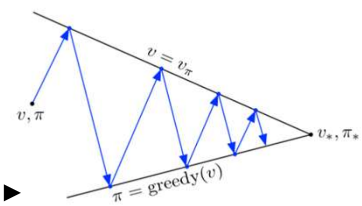
29

What's Next to Evaluation?

- **Improve the given policy**
  - by **1-step look-ahead greedy**

- Afterward, **repeat the following steps** until convergence
  - Evaluation of the new policy
  - Given the value function, improve the new policy

GPI (Generalized Policy Iteration) ►



UNIVERSITY OF TORONTO | Engineering

30

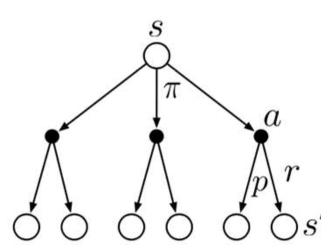
Monte Carlo vs. Dynamic Programming

- A trajectory (or episode) based ,

$$v_k(s) = \frac{1}{c_s}\sum v_i(s)$$

where $v_i(s) = \{R_t(s) + \gamma R_{t+1}(S_{t+1}) + \cdots\}$

- As opposed to Dynamic Programming



$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_k(s')\Big] \end{aligned}$$

- See the dramatic difference in computation

UNIVERSITY OF TORONTO | Engineering

31

Bootstrapping

- Samples in MC are independent
  - That is, the estimate for a state is not built upon that of other states
  - That is, MC uses no **bootstrapping**
  - As a result, **un**biased estimation but sample **in**efficient
- DP uses the maximum bootstrapping

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_k(s')\Big], \end{aligned}$$

- Bootstrapping improves **sample efficiency** a lot

UNIVERSITY OF TORONTO | Engineering

32

Which Value Function to Predict?

- State value function V vs. Action value function Q

  - State value function $V(s), \forall s \in S$
  - Action value function $Q(s,a), \forall s \in S, \forall a \in A$
- Why do we want to learn action value function?
  - One step look ahead greedy optimization requires **knowledge of dynamics**
  - No access to the model in learning problem

UNIVERSITY OF TORONTO | Engineering

33

On-policy vs. Off-policy Learning

online learning
offline learning

	On-policy	Off-policy
Bias	Unbiased	Biased
Variance	Low	High
Generality	Low	High

- On-policy
  - Sampling = learning
  - Policy to sample is the policy to be learned
  - No bias and more stable
  - Simpler
- Off-policy
  - Sampling ≠ learning
  - Behavior policy to sample (generate data): exploration
  - Target policy is to be evaluated and to be improved (optimization): exploitation
  - Unstable due to higher variance, hence slower convergence

UNIVERSITY OF TORONTO Engineering

34

Off-Policy Prediction via Importance Sampling

- Importance Sampling
  - A general technique for estimating $E[\cdot]$ under one distribution given samples from another
  - Weighing samples according to the relative probability

Given a starting state S_t, the probability of the subsequent state–action trajectory, $A_t, S_{t+1}, A_{t+1}, \ldots, S_T$, occurring under any policy π is

$$\begin{aligned}&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\}\\&\quad= \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1})\\&\quad= \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),\end{aligned}$$

The relative probability of the trajectory under the target and behavior policies is

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}.$$

UNIVERSITY OF TORONTO Engineering

35

- The expected return from policy π based on data generated by policy b is

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s).$$

The mean $\mu_1, \mu_2, \ldots$ of a sequence $x_1, x_2, \ldots$ can be computed incrementally,

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

- Cumulative computation of return:

$$V_n \equiv \frac{\sum_{k=1}^{n-1} \rho_k G_t}{n-1}, \qquad n \geq 2$$

$$V_{n+1} \leftarrow V_n + \frac{1}{n}(\rho_n G_n - V_n), \qquad n \geq 2$$

UNIVERSITY OF TORONTO | Engineering

36

Off-Policy Algorithms

- Off-policy MC Prediction

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \in \mathbb{R}$ (arbitrarily)
  $C(s,a) \leftarrow 0$
Loop forever (for each episode):
  $b \leftarrow$ any policy with coverage of π
  Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $A_t = \pi(S_t)$
    $G \leftarrow \gamma G + R_{t+1}$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{W} \cdot \left[\frac{\pi(A_t|S_t)}{b(A_t|S_t)} G - Q(S_t, A_t)\right]$
    $W \leftarrow W + 1$

- Off-policy MC GPI

Off-policy MC control, for estimating $\pi \approx \pi_*$

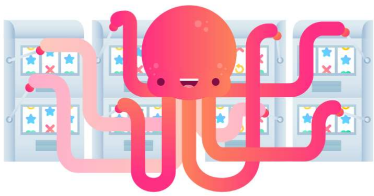
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s,a) \in \mathbb{R}$ (arbitrarily)
  $C(s,a) \leftarrow 0$
  $\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)
Loop forever (for each episode):
  $b \leftarrow$ any soft policy
  Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{W} \cdot \left[\frac{\pi(A_t|S_t)}{b(A_t|S_t)} G - Q(S_t, A_t)\right]$
    $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
    If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
    $W \leftarrow W + 1$

UNIVERSITY OF TORONTO | Engineering

37

Exploration vs. Exploitation

- Multi-armed Bandit



- Randomization is necessary
  - ϵ-greedy
  - Bayesian learning
  - Energy-based policy (Boltzmann policy)

- ϵ-Greedy Policy

$$A^* \leftarrow \arg\max_a Q(S_t, a) \qquad \text{(wit}$$

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$