# Lecture 8: On-Policy Learning and Off-Policy Learning

Bolei Zhou

The Chinese University of Hong Kong

*bzhou@ie.cuhk.edu.hk*

February 19, 2020

# Announcement

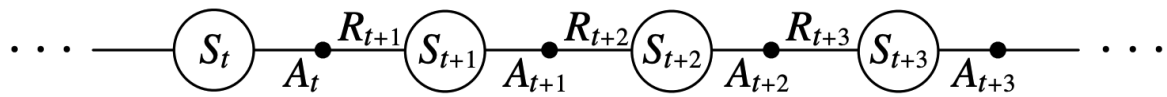
1. Course Piazza: `http://piazza.com/cuhk.edu.hk/spring2020/ierg6130`
2. Ask questions and give comments there with your name or anonymously

# Today's Content

1. On-policy learning and off-policy learning
2. Q-learning

# Review: TD Prediction

1. We collect experiences by interacting with the environment
   1. Which policy to use for the interaction? For prediction, we just use the given policy $\pi$
2. A trajectory consists of an alternating sequence of states and state–action pairs:




3. TD(0) method for estimating the value function $V_\pi(S)$

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

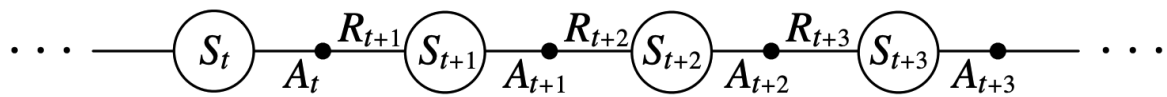
4. How about estimating action value function $Q(S)$?

# Sarsa: On-Policy TD Control

1. As it is a control task, we don't have a given policy $\pi$ and we have to learn a policy through trial and error!
2. Which policy to use for the interaction? $\epsilon$-greedy policy!
   1. All actions are tried with non-zero probability
   2. With probability $1-\epsilon$ choose the greedy action
   3. With probability $\epsilon$ choose an action at random

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

3. A collected trajectory consists of an alternating sequence of states and state–action pairs:




4. $\epsilon$-greedy policy for one step, then bootstrap the action value function:

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

# 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., $\epsilon$-greedy)
  Repeat (for each step of episode):
    Take action $A$, observe $R$, $S'$
    Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\epsilon$-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

1. In $\epsilon$-greedy policy, $\epsilon$ could be gradually decreased from 1 to 0 such that the policy follows from total random to greedy(Q), or $\epsilon$ is set as a fixed value as 0.1
2. $\epsilon$-greedy policy is improved over time, trajectories we collect become better and the policy is also being improved

# $n$-step Sarsa

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

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

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

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

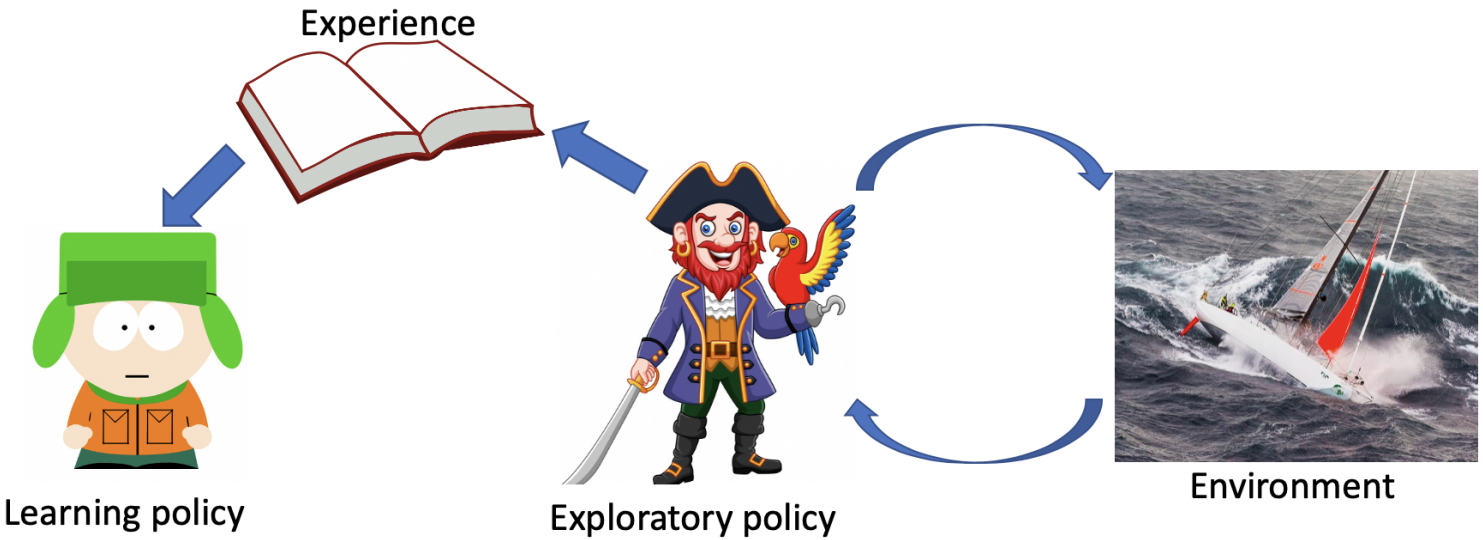
3. $n$-step Sarsa updates Q(s,a) towards the n-step Q-return:

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

# On-policy Learning vs. Off-policy Learning

1. On-policy learning: Learn about policy $\pi$ from the experience collected from $\pi$
   1. Behave non-optimally in order to explore all actions, then reduce the exploration. e.g., $\epsilon$-greedy
2. Another important approach is **off-policy learning** which essentially uses **two different polices**:
   1. the one which is being learned about and becomes the optimal policy
   2. the other one which is more exploratory and is used to generate trajectories
3. Off-policy learning: Learn about policy $\pi$ from the experience sampled from another policy $\mu$
   1. $\pi$: target policy
   2. $\mu$: behavior policy

# Off-policy Learning




1. Following behaviour policy $\mu(a|s)$ to collect data

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

Update $\pi$ using $S_1, A_1, R_2, ..., S_T$

2. It leads to many benefits:
   1. Learn about optimal policy while following exploratory policy
   2. Learn from observing humans or other agents
   3. Re-use experience generated from old policies $\pi_1, \pi_2, ..., \pi_{t-1}$

# Off-Policy Control with Q-Learning

1. The target learning policy $\pi$ is greedy on $Q(s, a)$

$$\pi(s) = \arg\max_{a'} Q(s, a')$$

2. The behavior policy could be totally random, but we let it improve, thus the behavior policy $\mu$ is $\epsilon$-greedy on $Q(s, a)$
3. Thus Q-learning target:

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

4. Thus the Q-Learning update becomes

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

# 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., $\epsilon$-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 $\epsilon$-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 $\epsilon$-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 $\epsilon$-greedy

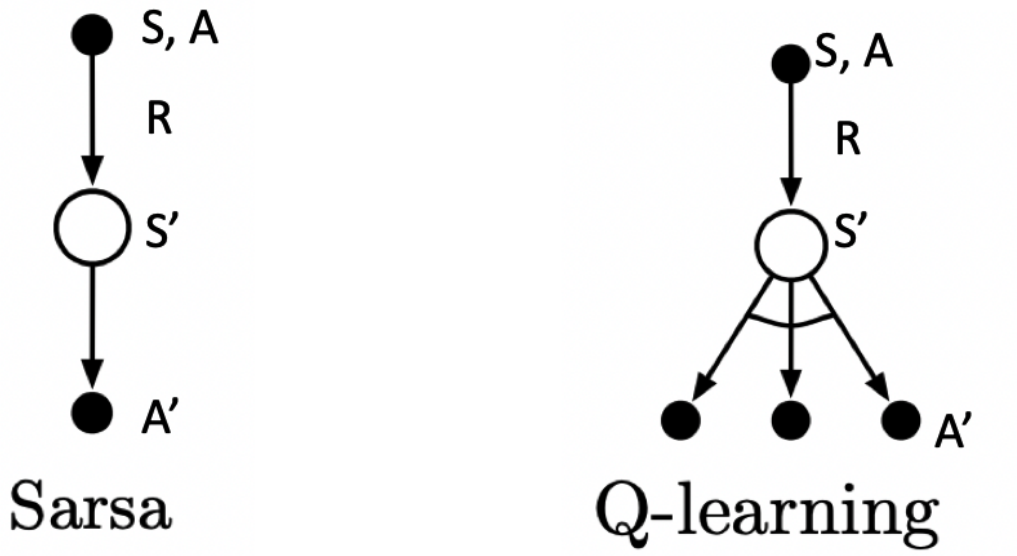
   Take action $A_t$, observe $R_{t+1}$ and $S_{t+1}$

   Then 'imagine' $A_{t+1}$ as $\arg\max Q(S_{t+1}, a')$ in the update target

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

# Comparison of Sarsa and Q-Learning

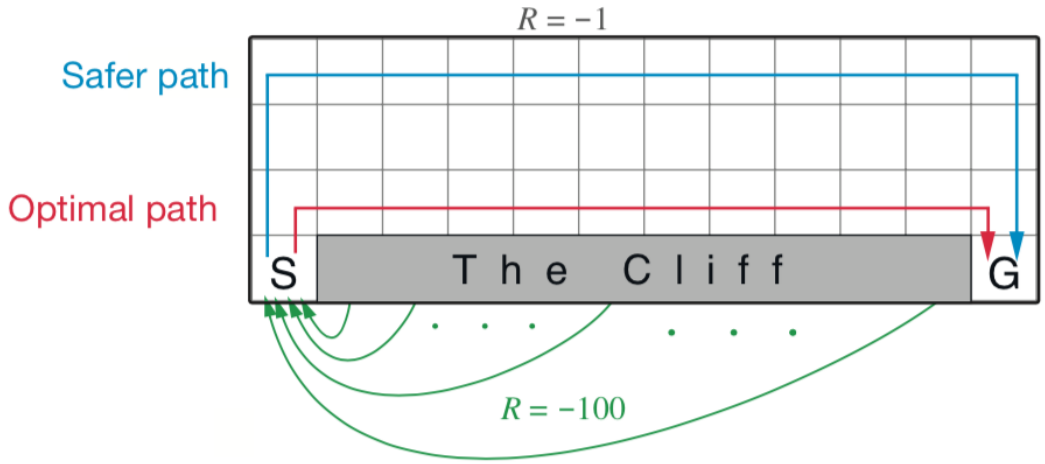
1. Backup diagram for Sarsa and Q-learning




2. In Sarsa, A and A' are sampled from the same policy so it is on-policy
3. In Q Learning, A and A' are from different policies, with A being more exploratory and A' determined directly by the max operator

# Example on Cliff Walk (Example 6.6 from Textbook)

https://github.com/cuhkrlcourse/RLexample/blob/master/modelfree/cliffwal

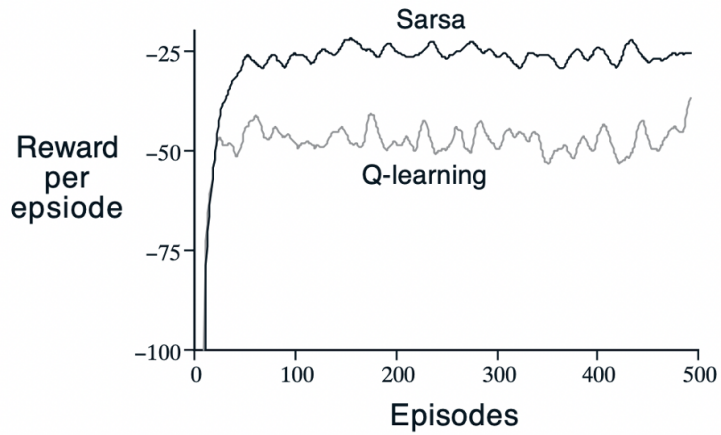



| 0 | 0 | 0 | 0 | R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|---|---|---|---|
| R | R | R | R | R | 0 | 0 | 0 | 0 | 0 | 0 | R |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R |
| R | * | * | * | * | * | * | * | * | * | * | G |

Result of Sarsa

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | R | R | R | R | R | R | R | R | R | R | R |
| R | * | * | * | * | * | * | * | * | * | * | G |

Result of Q-Learning




On-line performance of Q-learning is worse than that of Sarsa

# 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

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

# 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 $\mu$ 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

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

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 $\mu$ to evaluate $\pi$
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

# Why don't use importance sampling on Q-Learning?

1. Off-policy TD

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

2. Short answer: Because Q-learning does not make expected value estimates over the policy distribution. For the full answer click here
3. Remember bellman optimality backup from value iteration

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a'} Q(s',a') \quad (2)$$

   1. Q-learning can be considered as sample-based version of value iteration, except instead of using the expected value over the transition dynamics, we use the sample collected from the environment

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a') \quad (3)$$

   Q-learning is over the transition distribution, not over policy distribution thus no need to correct different policy distributions

# Comparison of DP and TD in Update

| **Expected update (DP)** | **Sample update for model-free RL** |
|---|---|
| Iterative Policy Evaluation<br>$V(s) \leftarrow \mathbb{E}[R + \gamma V(S')\|s]$ | TD Learning<br>$V(S) \leftarrow^{\alpha} R + \gamma V(S')$ |
| Policy Iteration<br>$Q(S, A) \leftarrow \mathbb{E}[R + \gamma Q(S', A')\|s, a]$ | Sarsa<br>$Q(S, A) \leftarrow^{\alpha} R + \gamma Q(S', A')$ |
| Value Iteration<br>$Q(S, A) \leftarrow \mathbb{E}[R + \gamma \max_{a' \in \mathcal{A}} Q(S', A')\|s, a]$ | Q-Learning<br>$Q(S, A) \leftarrow^{\alpha} R + \gamma \max_{a' \in \mathcal{A}} Q(S', a')$ |

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

# Sarsa and Q-Learning Example

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

# Next Week

1. Value function approximation
2. Deep Q Learning