Week 4: Off-policy Learning, Connection between Optimal Control and RL

Bolei Zhou

The Chinese University of Hong Kong

September 28, 2020

This Week

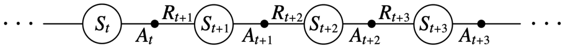
1. On-policy learning and off-policy learning
2. Importance sampling
3. Introduction on the connection between optimal control and RL

On-policy Learning vs. Off-policy Learning

1. On-policy learning: Learn about policy π from the experience collected from π
   1. Behave non-optimally in order to explore all actions, then reduce the exploration. e.g., ϵ-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 π from the experience sampled from another policy μ
   1. π: target policy
   2. μ: behavior policy

On-policy Control: SARSA

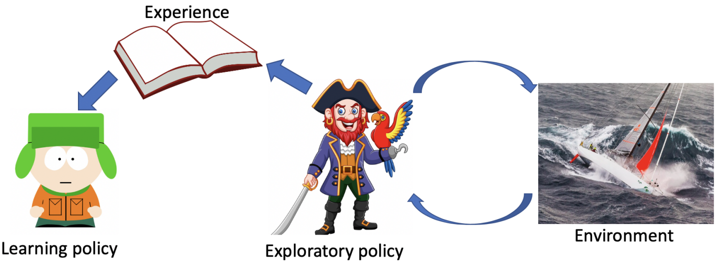
1. Trajectory is collected by the ϵ-greedy policy that depends on $Q(S_t, A_t)$,



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

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

Off-policy Learning



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

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

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 π 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 μ is ϵ-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

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

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

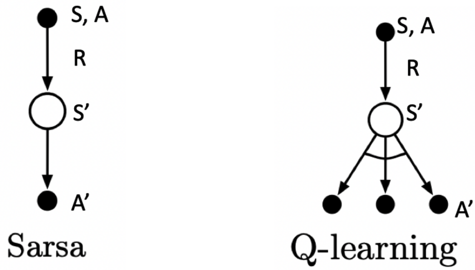
   Take action A_t, observe R_{t+1} and S_{t+1}

   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

Off-policy Learning with Importance Sampling

What is importance sampling?

1. Estimate the expectation of a function $f(x)$

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

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

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

Off-policy Learning with Importance Sampling

1. Expected return: $\mathbb{E}_{\tau\sim\pi}[r(\tau)]$ where $r(.)$ is the reward function and π is the policy
2. Estimate the expectation of return using trajectories τ_i sampled from another policy (behavior policy μ)

$$
\begin{aligned}
\mathbb{E}_{\tau\sim\pi}[r(\tau)] &= \int P_\pi(\tau) r(\tau) d\tau \\
&= \int P_\mu(\tau) \frac{P_\pi(\tau)}{P_\mu(\tau)} r(\tau) d\tau \\
&= \mathbb{E}_{\tau\sim\mu}\left[\frac{P_\pi(\tau)}{P_\mu(\tau)} r(\tau)\right] \\
&\approx \frac{1}{n} \sum_i \frac{P_\pi(\tau_i)}{P_\mu(\tau_i)} r(\tau_i)
\end{aligned}
$$

Off-Policy Monte Carlo with Importance Sampling

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

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

3. Update value towards correct return

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

Off-Policy TD with Importance Sampling

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

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

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

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

1. Off-policy TD

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

2. Short answer: 1,Q-learning uses a deterministic policy so no action probability. 2,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 update of value iteration, except instead of using the expected value over the transition dynamics, we use the sample collected from the environment

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

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

Reinforcement Learning and Optimal Control

1. Dimitri P. Bertsekas on reinforcement learning and optimal control
2. https://web.mit.edu/dimitrib/www/RLbook.html

Next Week

1. Value function approximation
2. Deep Q Learning