Lecture 15 - Trial and Error

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning
Course Page: [Click]

Model-Free Control

In this lecture, we will discuss model-free control where we learn good policies with only interactions, no knowledge of reward structure or transition probabilities). This framework is important in two types of domains:

1. When the MDP model is unknown, but we can sample trajectories from the MDP;
2. When the MDP model is known but computing the value function via our model-based control methods is infeasible due to the size of the domain, but we can sample trajectories from the MDP.

In this lecture, we will still restrict ourselves to the setting of discrete RL.

Model-Free Control

Generalized Policy Iteration (with a known model):

Algorithm 1: Policy iteration

Input: $\mathcal{M}, \epsilon$

$\pi \leftarrow$ Randomly choose a policy $\pi \in \Pi$

while *true* **do**

  $V^\pi \leftarrow$ POLICY EVALUATION $(\mathcal{M}, \pi, \epsilon)$

  $\pi^*(s) \leftarrow \arg\max_{a \in A} \mathbb{E}[R(s,a)] + \gamma \sum_{s' \in S} \mathbb{P}(s'|s,a) V^\pi(s')$, $\forall\, s \in S$

  if $\pi^*(s) = \pi(s)$ **then**

    break

  else

    $\pi \leftarrow \pi^*$

$V^* \leftarrow V^\pi$

return $V^*(s)$, $\pi^*(s)$ for all $s \in S$

Model-Free Control

Model-free Generalized Policy Iteration

Algorithm 2: Model-free generalized policy iteration

Input: ϵ

$\pi \leftarrow$ Randomly choose a policy $\pi \in \Pi$

while *true* **do**

  $Q^\pi \leftarrow$ MODEL-FREE POLICY EVALUATION (π, ϵ)

  $\pi^*(s) \leftarrow \arg\max_{a \in A} Q^\pi(s, a)\,,\ \forall\, s \in S$

  if $\pi^*(s) = \pi(s)$ **then**

    break

  else

    $\pi \leftarrow \pi^*$

$Q^* \leftarrow Q^\pi$

return $Q^*(s, a)$, $\pi^*(s)$ for all $s \in S$, $a \in A$

Model-Free Control

There are a few caveats to this algorithm due to the substitution we made in line 5.

1. If policy π is deterministic or does not take every action a with some positive probability, then we cannot actually compute the *argmax* in line 5.
2. The policy evaluation algorithm gives us an estimate of Q^{π}, so it is not clear whether line 5 will monotonically improve the policy like in the model-based case.

The policy π needs to explore actions, even if they might be suboptimal with respect to our current Q-value estimates.

Exploration

- ε-greedy policies: Take a random action with a small probability and take the greedy action the rest of the time. This type of exploration strategy is called an ε-greedy policy. Mathematically, an ε-greedy policy with respect to the state-action value $Q^\pi(s,a)$ takes the form

$$\pi(a \mid s) = \begin{cases} \text{Uniform}(\mathcal{A}) & \text{with probability } \varepsilon \\ \arg\max_a Q^\pi(s,a) & \text{with probability } 1-\varepsilon\,. \end{cases}$$

Exploration

- Monotonic ε-greedy policy improvement: the policy improvement for the ε-greedy policy can be shown as:

Lemma (Monotonic ε-greedy policy improvement)

Let π_i be an ε-greedy policy. Then, the ε-greedy policy with respect to Q^{π_i}, denoted π_{i+1}, is a monotonic improvement on policy π. In other words, $V^{\pi_{i+1}} \geq V^{\pi_i}$.

Exploration

- Greedy in the limit of exploration: balance the exploration of new actions with the exploitation of current knowledge by introducing a new class of exploration strategies that allows convergence guarantees of our algorithms.

Exploration

Definition (Greedy in the limit of infinite exploration)

A policy π is greedy in the limit of infinite exploration if it satisfies the following:

- 1. All state-action pairs are visited for infinitely many times, i.e., for all $s \in \mathcal{S}, a \in \mathcal{A}$,

  $$\lim_{k \to \infty} N_k(s, a) \to \infty \text{ with probability } 1\,,$$

  where $N_k(s, a)$ is the number of times action a is taken at state s up to episode k.

  2. The behavior policy converges to the policy that is greedy with respect to the learned Q-function, i.e., for all $s \in \mathcal{S}, a \in \mathcal{A}$,

  $$\lim_{k \to \infty} \pi_k(a \mid s) = \arg\max_a Q(s, a) \text{ with probability } 1\,.$$

Exploration

- Example of a GLIE strategy:

  An ε-greedy policy where ε is decayed to zero with $\varepsilon_k = O(1/k)$, where k is the episode number. We can see that since $\sum_{k=1}^{K} \varepsilon_k = O(\log K)$, we will explore each action for infinitely many times, hence satisfying the first GLIE condition (we leave the rigorous proof to the reader). Since $\varepsilon_k \to 0$ as $k \to \infty$, we also have that the policy is greedy in the limit, hence satisfying the second GLIE condition.

Monte-Carlo Control

Online Monte-Carlo Control:

Algorithm 3: Online Monte-Carlo control

Initialize $Q(s,a) = 0$, $Returns(s,a) = 0$ for all $s \in S, a \in A$
Set $k \leftarrow 1$
while *true* **do**
  Sample k-th episode $\{s_{t,k}, a_{t,k}, r_{t,k}\}_{t \in [H]}$ under policy π
  for $t = 1, \ldots, H$ **do**
    if *First visit to* (s,a) *in episode* k **then**
      Append $\sum_{t'=t}^{H} r_{t',k}$ to $Returns(s_t, a_t)$
      $Q(s_t, a_t) \leftarrow \text{average}(Returns(s_t, a_t))$
  $k \leftarrow k+1$, $\epsilon = \frac{1}{k}$
  $\pi_k = \epsilon$-greedy with respect to Q
Return Q, π_k

Monte-Carlo Control

GLIE strategies can help us arrive at convergence guarantees for our model-free control methods. In particular, we have the following statement.

Lemma

GLIE Monte-Carlo control converges to the optimal state-action value function. That is $Q(s, a) \to Q^*(s, a)$.

Temporal-Difference (TD) Methods for Control

Online Temporal-difference Methods for Control:

Algorithm 4: SARSA

Input: ϵ, α_t

Initialize $Q(s,a)$ for all $s \in S, a \in A$ arbitrarily except $Q(terminal, \cdot) = 0$

$\pi \leftarrow \epsilon$-greedy policy with respect to Q

for *each episode* **do**

- $t \leftarrow 1$
- Set s_1 as the starting state
- Choose action a_1 from policy $\pi(s_1)$
- **while** *until episode terminates* **do**
  - Take action a_t and observe reward r_t and next state s_{t+1}
  - Choose action a_{t+1} from policy $\pi(s_{t+1})$
  - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  - $\pi \leftarrow \epsilon$-greedy with respect to Q
  - $t \leftarrow t + 1$

Return Q, π

Temporal-Difference (TD) Methods for Control

SARSA gets its name from the parts of the trajectory used in the update equation.

- We can see that to update the Q-value at state-action pair (s, a), we need the reward, next state and next action, thereby using the values (s, a, r, s', a').
- SARSA is an on-policy method because the actions a and a' used in the update equation are both derived from the policy that is being followed at the time of the update.

Temporal-Difference (TD) Methods for Control

Lemma

SARSA for finite-state and finite-action MDPs converges to the optimal action-value, i.e., $Q(s,a) \to Q^*(s,a)$, if the following two conditions hold:

1. *The sequence of policies π from is GLIE*
2. *The step-sizes α_t satisfy the Robbins-Munro sequence such that*

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty.$$

Importance sampling for off-policy TD

Recall that our TD update took the form

$$V(s) \to V(s) + \alpha(r + \gamma V(s') - V(s)).$$

Suppose that like in off-policy Monte-Carlo policy evaluation, we have data from a policy π_b, and we want to estimate the value of policy π_e. This new update will be:

$$V^{\pi_e}(s) \to V^{\pi_e}(s) + \alpha\left(\frac{\pi_e(a \mid s)}{\pi_b(a \mid s)}(r + \gamma V^{\pi_e}(s') - V^{\pi_e}(s))\right).$$

Importance sampling for off-policy TD

- Off-Policy TD uses one trajectory sample instead of sampling the entire trajectory like in Monte Carlo, so we only incorporate the likelihood ratio from one step, so Off-Policy TD also has a significantly lower variance than Monte Carlo.
- π_b does not need to be the same at each step, but we do need to know the probability for every step. As is in Monte Carlo, we need the two policies to have the same support. That is, if $\pi_e(a \mid s) \cdot V^{\pi_e}(s') > 0$, then $\pi_b(a \mid s) > 0$.

Q-learning

We do not need to leverage importance sampling, instead, we can maintain the Q estimates and bootstrap the value of the best future action. Recall our SARSA update:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right),$$

but we can instead bootstrap the Q value at the next state to get the following update:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right).$$

The select action is not necessarily the same as the one we would derive from the current policy. Therefore, Q-learning is an off-policy algorithm.

Q-learning

Algorithm 5: Q-learning with ϵ-greedy exploration

Input: ϵ, α, γ

Initialize $Q(s, a)$ for all $s \in S, a \in A$ arbitrarily except $Q(terminal, \cdot) = 0$

$\pi \leftarrow \epsilon$-greedy policy with respect to Q

for *each episode* **do**

    $t \leftarrow 1$

    Set s_1 as the starting state

    while *until episode terminates* **do**

        Sample action a_t from policy $\pi(s_t)$

        Take action a_t and observe reward r_t and next state s_{t+1}

        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$

        $\pi \leftarrow \epsilon$-greedy policy with respect to Q

        $t \leftarrow t + 1$

return Q, π

Q-learning

Boltzmann Exploration. Alternatively, one can perform Boltzmann exploration. Instead of utilizing the ε-greedy exploration, we can use:

$$\pi(a|s) = \frac{\exp Q(s,a)}{\sum_{a'} \exp Q(s,a')} \tag{1}$$

This is a "soft" policy representation based on the action-value Q function.

Maximization bias

Example: Game of coins Suppose there are two identical fair coins, but we do not know that they are fair or identical. If a coin lands on heads, we get one dollar and if a coin lands on tails, we lose a dollar. We ask the following two questions.

1. Which coin will yield more money for future flips?
2. How much can we expect to win/lose per flip using the coin from question 1?

Maximization bias

In an effort to answer this question:

- we flip each coin once. We then pick the coin that yields more money as the answer to question 1.
- We answer question 2 with however much that coin gave us.

For example, if coin 1 landed on heads and coin 2 landed on tails, we would answer question 1 with coin 1, and question 2 with one dollar.

Maximization bias

We examine the possible scenarios for the outcome of this procedure.

- If at least one of the coins is heads, then our answer to question 2 is one dollar.
- If both coins are tails, then our answer is negative one dollar.

Thus, the expected value of our answer to question 2 is $\frac{3}{4} \times (1) + \frac{1}{4} \times (-1) = 0.5$. This gives us a higher estimate of the expected value of flipping the better coin than the true expected value of flipping that coin.

Maximization bias

This problem comes from the fact that we are using our estimate to both choose the better coin and estimate its value. We can alleviate this by separating these two steps.

- Flip the coin to choose the better coin,
- Flip the better coin again and use this value as your answer for question 2.

The expected value of this answer is now 0, which is the same as the true expected value of flipping either coin.

Double Q-learning

Double Q-learning:

- We can maintain two independent unbiased estimates, Q_1 and Q_2 and when we use one to select the maximum, we can use the other to get an estimate of the value of this maximum.
- The ε-greedy policy with respect to $Q_1 + Q_2$ indicates that the ε-greedy policy where the optimal action at state s is equal to $\arg\max_a Q_1(s,a) + Q_2(s,a)$.

Double Q-learning

Algorithm 6: Double Q-learning

Input: ϵ, α, γ

Initialize $Q_1(s,a), Q_2(s,a)$ for all $s \in S, a \in A$ arbitrarily

$t \leftarrow 0$

$\pi \leftarrow \epsilon$-greedy policy with respect to $Q_1 + Q_2$

while *true* **do**

  Sample action a_t from policy π at state s_t

  Take action a_t and observe reward r_t and next state s_{t+1}

  if *with 0.5 probability* **then**

    $Q_1(s_t, a_t) \leftarrow Q_1(s_t, a_t) + \alpha(r_t + \gamma Q_2(s_{t+1}, \arg\max_{a'} Q_1(s_{t+1}, a')) - Q_1(s_t, a_t))$

  else

    $Q_2(s_t, a_t) \leftarrow Q_2(s_t, a_t) + \alpha(r_t + \gamma Q_1(s_{t+1}, \arg\max_{a'} Q_2(s_{t+1}, a')) - Q_2(s_t, a_t))$

  $\pi \leftarrow \epsilon$-greedy policy with respect to $Q_1 + Q_2$

  $t \leftarrow t + 1$

return $\pi, Q_1 + Q_2$

Question and Answering (Q&A)