Deep Reinforcement Learning (Sp25)

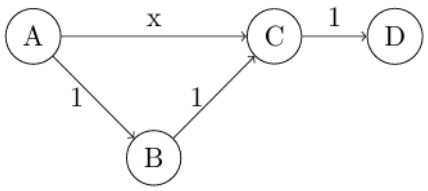
Instructor: Dr. Mohammad Hossein Rohban

Final Exam | 1404/06/09 | 150 Minutes

1 Value-Based Theory

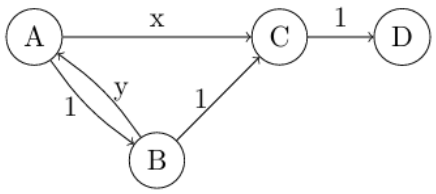
1.1 Markov Decision Process

(a) Consider the following deterministic MDP with four states A, B, C and D:



The edges designate actions between states, the weights on those edges are the rewards, and the discount factor is $\gamma = 1$. Let k be the first iteration of Value Iteration at which the value function converges for some x for a particular state (i.e. $V_k(s) = V^*(s)$). Use the convention where $V_0(s)$ is the value at initialization, $V_1(s)$ is the value after one iteration, etc. For each state A, B, C, and D, list all possible values of k. In the case a value function for a particular state never converges, set $k = \infty$ for that state.

(b) Now consider the following deterministic MDP with four states A, B, C and D:



The edges designate actions between states, the weights on those edges are the rewards, and the discount factor is again $\gamma = 1$. Furthermore, assume that $x, y \geq 0$.

(i) Let k be the first iteration of Value Iteration for some nonnegative x and y at which the value function converges for a particular state ($V_k(s) = V^*(s)$). For each state A, B, C and D list all possible values of k. In case a value for a particular state never converges set $k = \infty$ for that state.

(ii) Suppose we perform Policy Iteration and that k is the first iteration for which the policy is optimal for a particular state (i.e. $\pi_k(s) = \pi^*(s)$). On top of $x, y \geq 0$ also assume that $x + y < 1$ and that tie-breaking during policy improvement is alphabetical. The initial policy is given in the table below.

State s	Policy $\pi_0(s)$
A	C
B	C
C	D
D	D

For each state A, B, C and D, find k; if the policy never converges set $k = \infty$ for that state.

2 Policy-Based Theory

2.1 Policy Improvement Constraints in Policy Gradient

Explain in a theoretical way why we impose π_θ and $\pi_{\theta'}$ be close for consecutive updates of θ in a policy gradient algorithm, by showing that such constraint would guarantee $J(\theta')$ to increase with respect to $J(\theta)$ in a typical update of the θ in a policy gradient algorithm.

3 Advanced Theory

3.1 Best Arm Identification in Multi-armed Bandit

In many experimental settings we are interested in quickly identifying the "best" of a set of potential interventions, such as finding the best of a set of experimental drugs at treating cancer, or the website design that maximizes user subscriptions. Here we may be interested in efficient pure exploration, seeking to quickly identify the best arm for future use.

In this problem, we bound how many samples may be needed to find the best or near-optimal intervention. We frame this as a multi-armed bandit with rewards bounded in $[0, 1]$. Recall a bandit problem can be considered as a finite-horizon MDP with just one state ($|S| = 1$) and horizon 1: each episode consists of taking a single action and observing a reward. In the bandit setting – unlike in standard RL – the action (or "arm") taken does not affect the distribution of future states. We assume a simple multi-armed bandit, meaning that $1 < |A| < \infty$. Since there is only one state, a policy is simply a distribution over actions. There are exactly $|A|$ different deterministic policies. Your goal is to design a simple algorithm to identify a near-optimal arm with high probability.

We recall Hoeffding's inequality: if $X_1, \ldots, X_n$ are i.i.d. random variables satisfying $0 \leq X_i \leq 1$ with probability 1 for all i, $X = \mathbb{E}[X_1] = \cdots = \mathbb{E}[X_n]$ is the expected value of the random variables, and

$$\hat{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

is the sample mean, then for any $\delta > 0$ we have

$$\Pr\left(|\hat{X} - X| > \sqrt{\frac{\log(2/\delta)}{2n}} \right) < \delta. \tag{1}$$

Assuming that the rewards are bounded in $[0, 1]$, we propose this simple strategy: pull each arm n_e times, and return the action with the highest average payout $\hat{r}_a$. The purpose of this question is to study the number of samples required to output an arm that is at least ϵ-optimal with high probability. Intuitively, as n_e increases the empirical average of the payout $\hat{r}_a$ converges to its expected value r_a for every action a, and so choosing the arm with the highest empirical payout $\hat{r}_a$ corresponds to approximately choosing the arm with the highest expected payout r_a.

(a) We start by bounding the probability of the "bad event" in which the empirical mean of some arm differs significantly from its expected return. Starting from Hoeffding's inequality with n_e samples allocated to every action, show that:

$$\Pr\left(\exists a \in A \text{ s.t. } |\hat{r}_a - r_a| > \sqrt{\frac{\log(2/\delta)}{2n_e}} \right) < |A|\delta. \tag{2}$$

Note that, depending on your derivation, you may come up with a tighter upper bound than $|A|\delta$. This is also acceptable (as long as you argue that your bound is tighter), but showing the inequality above is sufficient.

(b) After pulling each arm (action) n_e times our algorithm returns the arm with the highest empirical mean:

$$a^{\dagger} = \arg\max_a \hat{r}_a. \tag{3}$$

Notice that $a^{\dagger}$ is a random variable. Let $a^{\star} = \arg\max_a r_a$ be the true optimal arm. Suppose that we want our algorithm to return at least an ϵ-optimal arm with probability at least $1 - \delta'$, as follows:

$$\Pr\left(r_{a^{\dagger}} \geq r_{a^{\star}} - \epsilon\right) \geq 1 - \delta'. \tag{4}$$

How accurately do we need to estimate each arm in order to pick an arm that is ϵ-optimal? Then derive how many total samples we need (across all arms) to return an ϵ-optimal arm with probability at least $1 - \delta'$ (that satisfies Equation (4)). Express your result as a function of the number of actions, the required precision ϵ, and the failure probability δ'.

4 Exploration Methods

4.1 Implicit Density Modeling and Optimal Classifier

In implicit density modeling, where each observed state s is assigned positive class while all previously seen states are assigned as the negative class, prove that the density of the states can be modeled as $\frac{(1-D(s))}{D(s)}$ with $D(s)$ being the trained optimal classifier. (Hint: write the training loss function to obtain the optimal classifier).

5 Imitation & Inverse RL

5.1 Compounding Errors in Behavior Cloning

Consider an MDP with finite horizon T. Suppose $\pi^\star$ is an expert policy and π is a learned deterministic policy. The per-step cost is defined as $C(s, a) \in [0, 1]$, and the expected total cost of a policy is:

$$J(\pi) \;=\; \sum_{t=1}^{T} \mathbb{E}[C(s_t, a_t) \mid a_t \sim \pi(\cdot \mid s_t)]\,.$$

The state distribution at time t under π is denoted by d_π^t, and the time-averaged distribution is:

$$d_\pi = \tfrac{1}{T} \sum_{t=1}^{T} d_\pi^t.$$

To measure imitation of the expert, we use a binary surrogate loss:

$$\ell(s, \pi) \;\geq\; \mathbf{1}\{\pi(s) \neq \pi^\star(s)\}.$$

Suppose π is trained only using supervised learning on expert states such that

$$\mathbb{E}_{s \sim d_{\pi^\star}}[\ell(s, \pi)] \;\leq\; \varepsilon.$$

Show that in the worst case, the performance gap can grow quadratically with the horizon:

$$J(\pi) \;\leq\; J(\pi^\star) \;+\; T^2\,\varepsilon.$$

6 Offline RL

6.1 Conservative Q-Learning Optimization

(a) Conservative Q-Learning involves a two-level optimization of the Q-function. Explain what parameters are optimized in each level of this optimization.

(b) In reality, one level of this optimization is solved analytically. Explain how that happens and give explicit formula for that.

7 Multi-Agent RL

7.1 Independent Q-Learning and Non-Stationarity

Write down the update equation of independent Q-Learning in multi-agent RL, and explain why the rewards might be non-stationary? How does that hurt convergence?

7.2 Fictitious Play in Q-Learning

Explain what fictitious play is and how using it changes the Q-Learning updates, i.e. write down the update equations.

8 Meta RL

8.1 Hidden State Persistence in Meta-RL

Why in RNN implementation of meta-RL, the hidden state is not reset between the episodes?

8.2 Adapting to Unknown Friction with MAML

Suppose that you want to train a robot to walk on a 2d surface with different frictions. The problem is that at the test time, the robot does not know what the friction of the surface is. What happens if you train the robot on union of various friction environments? Explain in details how you would solve this problem using Model-Agnostic Meta-RL.