Reinforcement Learning - Lecture 1

ENS M2

October 1, 2025

Contents

1 Machine Learning and Reinforcement Learning

1.1 Key Problems in Machine Learning

1.1.1 Key Problem 1: Pattern Recognition

MNIST Dataset: Supervised classification

- **Input**: Dataset $(X_1, Y_1), \ldots, (X_n, Y_n)$
- **Output**: Decision rule $\mathbb{X} \to y$

1.1.2 Key Problem 2: Unsupervised Classification

Definition

Goal: Find a classification of the input space.
Find: $\phi : \mathbb{X} \to \{1, \ldots, K\}$ K classes such that $\sum_{i=1}^{n} d(X_i, X_j)$ is minimized.

1.1.3 Key Problem 3: Reinforcement Learning

Example

Stock Management Problem:
Capacity M slots for bikes.
On Monday:

- Command and receive the number of bikes you want: for a bikes you pay $f(a) = C_0 + C \cdot a$
- You pay for stocking the bikes in stock: for s bikes in stock you pay $g(s) = l \cdot s$

Problem: How many bikes should you command on Monday?
Model:

- State space: $\mathcal{S} = \{0, \ldots, M\}$ number of bikes in stock at the beginning of the week
- Action space: $\mathcal{A} = \{0, \ldots, M\}$ number of bikes to order
- Transition: $s_{t+1} = \min(s_t + a_t, M) - D_{t+1}$ where D_t is the demand at week t
- Reward: $r_t = -f(a_t) - g(s_t)$ cost at week t
- Policy: $\pi : \mathcal{S} \to \mathcal{A}$
- Objective: $\max_\pi \mathbb{E}[\sum_{t=1}^{T} r_t]$
- Model: D_t are i.i.d. random variables with known distribution

Tip

In reinforcement learning, the agent interacts with an environment with a random variable A_t (action) that influences the state S_t of the environment and receives a reward R_t.

2 Markov Decision Process (MDP)

Definition

A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, P, r)$ where:

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action set
- P is the transition kernel of the MDP: $P = (P_a(s, s'))_{s,s' \in \mathcal{S}, a \in \mathcal{A}}$ where $\forall a \in \mathcal{A}, s, s' \in \mathcal{S}$, P_a is a transition matrix with $\sum_{s' \in \mathcal{S}} P_a(s, s') = 1$ and $0 \leq P_a(s, s') \leq 1$
- $P_a(s, s')$ is the probability to transition from state s to state s' given that you choose action a
- r is the reward kernel of the MDP: $r = (r(s, a, s'))_{s,s' \in \mathcal{S}, a \in \mathcal{A}}$ where $r(s, a, s')$ is the reward received associated to a transition from state s to state s' given that you choose action a

Example

Windy Cliff Labyrinth:
There is an entry and an exit and a cliff and a sink.

- $\mathcal{S} = [1, M] \times [1, N]$
- $\mathcal{A} = \{\text{N, S, E, W}\}$
- Transition: $P_a(s, s')$ with specific probabilities according to wind
- Reward: $r(s, a, s') = 1$ if s' is the exit, 0 otherwise

2.1 Running a Policy

A run: Horizon T = number of actions to take

Start: State $S_0 = (s, 1)$

For $t = 0, \ldots, T-1$:

1. Choose action A_t in $\mathcal{A}$
2. Move to random state S_{t+1} with the probability law $P_{A_t}(S_t, \cdot)$
3. Receive reward $R_t = \mathbf{1}_{S_{t+1} \text{ is the exit } (M,1)}$

2.2 Policies

A sequence of mappings $\phi_t : (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^t \to \mathcal{A}$

$A_t = \phi_t(S_0, A_0, R_0, \ldots, S_{t-1}, A_{t-1}, R_{t-1})$ may be deterministic or random.

2.3 Cumulated Reward

$W_T = \sum_{t=0}^{T-1} R_t = \sum_{t=0}^{T-1} r(S_t, A_t, S_{t+1})$ is a random quantity

Goal: $\max_{(\phi_t)_t} \mathbb{E}[W_T]$ - Classical RL problem. We might want to **choose quantiles of W_T instead of the expectation.**

Example

Stock Management:

- State space: $\mathcal{S} = \{0, \dots, M\}$ number of bikes in stock at the beginning of the week. S_t is the number of bikes in stock at the beginning of week t
- Action space: $\mathcal{A} = \{0, \dots, M\}$ number of bikes to order. A_t is the number of bikes ordered at the beginning of week t
- Reward: $r(s, a, s') = -f(a) - g(s) + h(\min(s + a, M) - s')$ where h is the gain function
- Transition: $P_a(s, s') = \mathbb{P}(U_1 = (s + a \wedge M) - s')$ Probability of the demand (the $\wedge$ symbol means minimum)

Two main approaches:

- **Planning**: find the optimal policy if the dynamics (i.e. P and r) are known
- **Learning**: find the optimal policy if the dynamics are unknown but we can observe the system