A control-theoretical perspective of continuous time reinforcement learning

Xin Guo
UC Berkeley
email: xinguo@berkeley.edu
23rd Winter School on Mathematical Finance, Jan 19-21, 2026

Mini-course outline

- Part I: Introduction to Q and advantage function in RL
- Part II: DPP for MFC with learning & characterization for Q function in continuous-time RL
- Part III: Regret Analysis in RL via Stability of HJB and BSDE
- Part IV: Connecting RL, LLM, and Generative Diffusion Models
- Part V: RDE for General RL

Lecture I: Q function and advantage function in RL

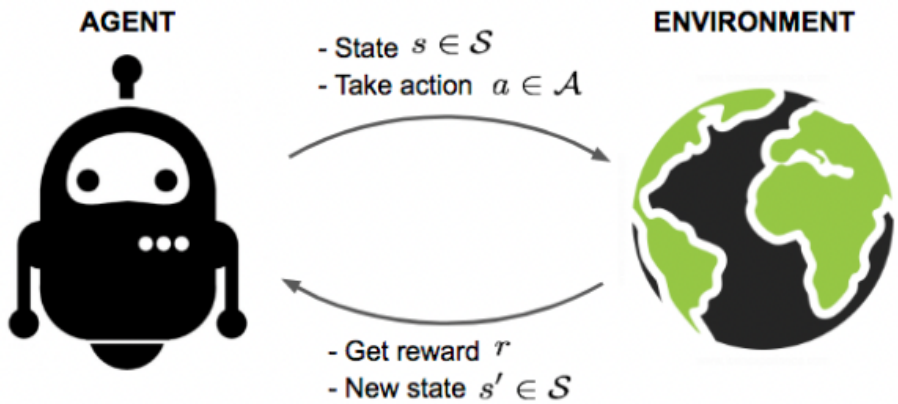


Figure: *Illustration of reinforcement learning*

- Learning the (stochastic) environment while trying to find optimal control to achieve certain objectives
- Examples: Chess and Go, video games, autonomous driving, robots
- Analytically, RL ~ MDP/control plus learning

RL in continuous-time

For a given control process $(a_t)_{t>0}$ taking values in $A \subset \mathbb{R}^k$, consider the state dynamics governed by the SDE:

$$\mathrm{d}X_t = b(t, X_t, a_t)\mathrm{d}t + \sigma(t, X_t, a_t)\mathrm{d}W_t, \quad \forall t > 0.$$

Objective: determine the optimal control that maximizes the total reward:

$$\mathbb{E}\left[\int_0^T r\big(t, X_t, a_t\big)\,\mathrm{d}t + g(X_T)\right],$$

without knowing the coefficients b, σ, r, g.

Admissible controls

- Open loop
- (*) Closed loop most relevant class in RL

Deterministic policy for classical control

Classical control

- coefficients are given
- the optimal control is characterized, under mild conditions, by

$$a_t^* = \mu^*(t, X_t^{\mu^*}), \quad \forall t \geq 0,$$

where

- $\mu^* : [0, T] \times \mathbb{R}^d \to A$ is an (optimal) deterministic policy (a.k.a. feedback control)
- μ^* is continuous/differentiable with proper regularity conditions

RL: try to learn this optimal policy without knowing the coefficients, and/or the for of payoff/reward function

Key concepts: Q function and advantage function

Q function in Markov decision process: introduction

- A tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, s_0)$

$$
\begin{aligned}
V(s) = \text{maximize}_{\pi} \quad & \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 = s\right] \\
\text{subject to} \quad & s_{t+1} \sim P(s_t, a_t), \quad a_t \sim \pi(s_t), \quad t = 0, 1, \ldots
\end{aligned}
$$

  - $s_t \in \mathcal{S}$ the state of the agent at time t
  - $a_t \in \mathcal{A}$ the action of the agent at time t
  - $a_t \sim \pi(s_t)$ where $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ is the policy of the agent
  - Instantaneous reward $R(s_t, a_t)$ sampled from some distribution
  - Time-homogeneous Markov transition kernel $P(s_t, a_t) \in \mathcal{P}(\mathcal{S})$

- Reinforcement learning in discrete time
  - MDP with *unknown* P and r
  - Policy
    - $\pi : \mathcal{S} \to \mathcal{A}$ pure strategy $\leftrightarrow$ strict control
    - $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ mixed strategy $\leftrightarrow$ relaxed control (to find optimal decision under exploration-exploitation context)

Q Function and Bellman Equation

- Q function, one of the basic quantities used for RL

$$\begin{aligned} Q(s,a): &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)|s_0=s,a_0=a\right] \\ &= \underbrace{\mathbb{E}r(s,a)}_{\text{reward of taking action a}} + \gamma \underbrace{\mathbb{E}_{s'\sim P(s,a)}V(s')}_{\text{reward of playing optimal afterwards}} \end{aligned}$$

- The value function (of a given policy π)

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi(s)}[Q^{\pi}(s,a)]$$

- Advantage function: quantifying the quality of a specific action a over average actions given a specific state s

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

Bellman Equation for Q Function

- The Bellman equation for the Q-function given by

$$Q^{\pi}(s,a) = \mathbb{E}[R(s,a)] + \gamma \mathbb{E}_{s' \sim P(s,a), a \sim \pi(s')} \left[Q^{\pi}(s', a') \right]$$

- $V(s) = \max_{a \in \mathcal{A}} Q(s,a)$, then $\pi^*(s) \in \arg\max_{a \in \mathcal{A}} Q(s,a)$

Q-function update for $T = \infty$

$$Q_{t+1}(s_t, a_t)$$
$$\leftarrow (1 - \beta_t(s_t, a_t)) \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \beta_t(s_t, a_t) \underbrace{\left[R(s_t, a_t) + \gamma \max_{a'} Q_t(s_{t+1}, a') \right]}_{\text{new value}}$$

- $n^i(s, a)$: i-th time to visit (s, a)
- (Watkins and Dayan 1992),

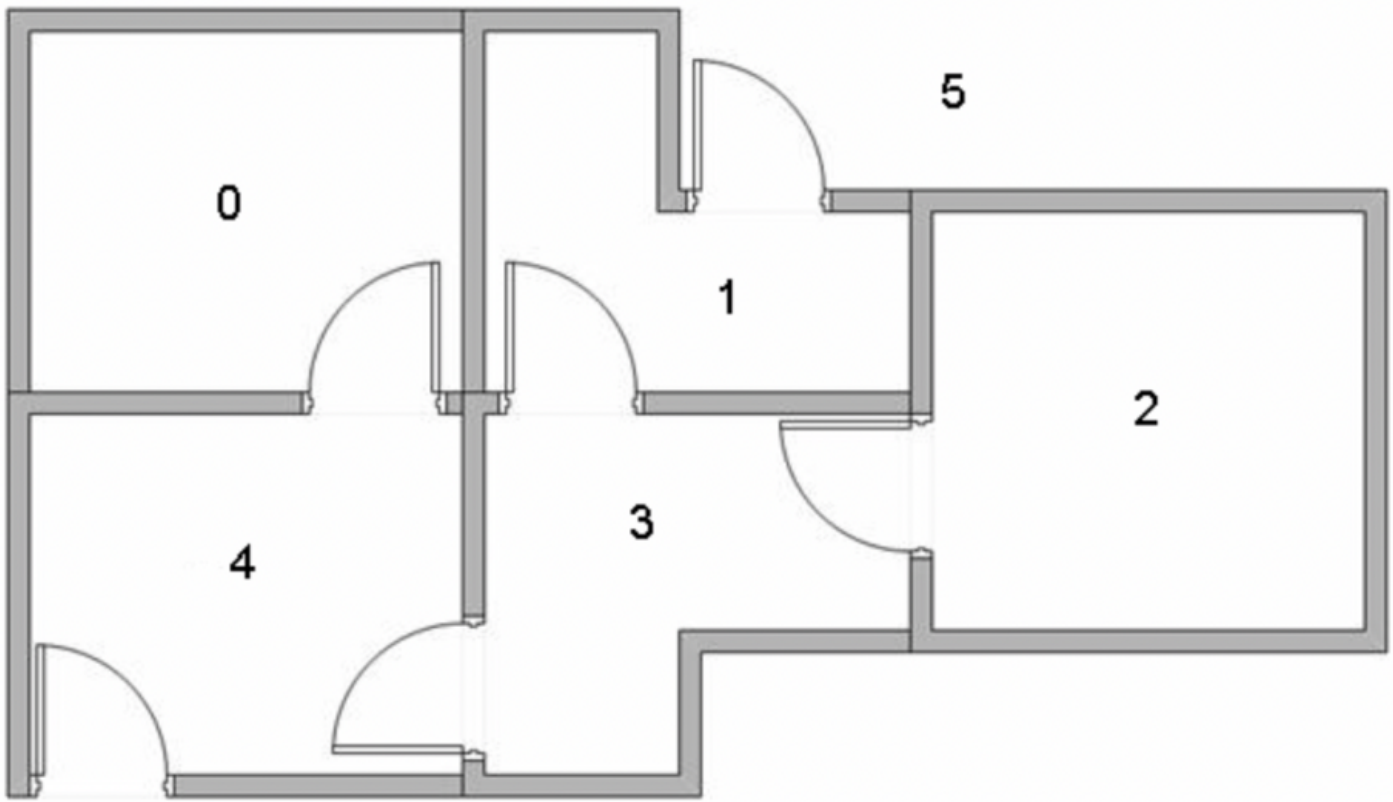
$$\sum_{i=1}^{\infty} \beta_{n^i(s,a)} = \infty, \ \sum_{i=1}^{\infty} (\beta_{n^i(s,a)})^2 < \infty,$$

then $Q_t(s, a) \to Q(s, a)$ as $t \to \infty$, $\forall s, a$ with probability 1

Outline

Example of Q-function

Consider this house. You start in 2 and you want to go outside.
Quickest path? (Ok, not too hard here, but imagine 100 rooms...)



Example

Note: problem and pictures (with minor modifications) from http://mnemstudio.org/path-finding-q-learning-tutorial.htm

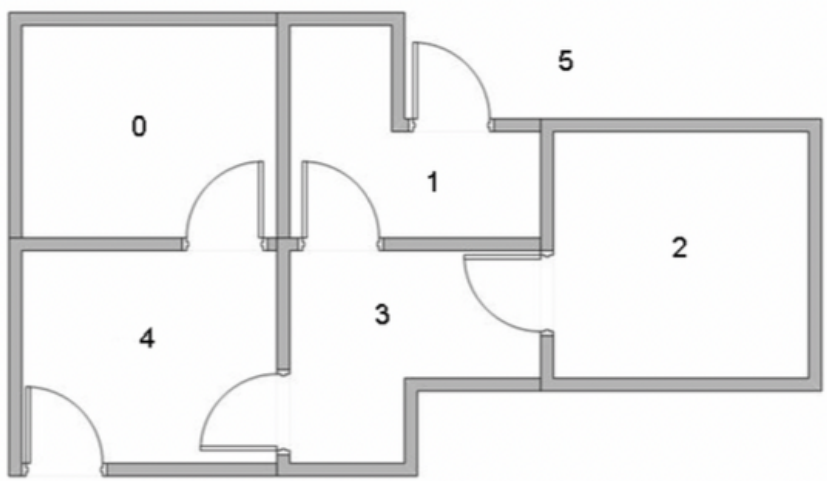
Let us solve the problem by the Q learning approach. First, let us formalize the model

- State space $\mathcal{S} = \{0, 1, 2, 3, 4, 5\}$. That is, $s_t = 5$ means that at time t, we are in room number 5
- Action space $\mathcal{A} = \{0, 1, 2, 3, 4, 5\}$. That is, $a_t = 5$ means that at time t, our action is move to room number 5
- Transition function $s_{t+1} = P(s_t, a_t) = a_t$. Note that no noise here, deterministic system
- Reward function $R =$?

Example

Recall: $R(s, a)$ is the reward if you are in state s and choose action a (that is, you are in room s and move to room a). We choose

- $R(s, a) = 0$ if (s, a) are linked *(alternative $= -1$)*
- $R(s, a) = -1$ if (s, a) are not linked *(alternative $= -\infty$)*
- $R(s, a) = 100$ if a is the target state



$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Example

Algorithm.

1. Initialize the matrix Q with all zeros.

2a. Randomly choose the initial state $s \in S$.

2b. Randomly choose an (admissible) action $a \in A$ and use approximated dynamic programming to update $Q(s, a)$. You are now in state $s = a$.

2c. Repeat (2a)-(2b) until you touch the target state.

3. Repeat (2a)-(2b)-(2c) for N iterations.

4. Compute the optimal control.

Example

Recall the equation to update the matrix Q:

$$Q(s,a) \leftarrow (1-\eta)Q(s,a) + \eta\left(R(s,a) + \gamma \max_{a' \in A} Q(f(s,a,\bar{e}_t), a') \right).$$

To simplify, we here take $\eta = 1$ (all the focus on the present iteration), and $\gamma = 0.8$. The update then reads

$$Q(s,a) \leftarrow R(s,a) + 0.8 \max_{a' \in A,\ a' \text{ admissible}} Q(a,a').$$

Let's see the details...

Example

We first initialize the matrix Q to zero.

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} & & & \text{Action} & & \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Example

Randomly choose an initial state, say 1. The admissible actions are $\{3, 5\}$. Randomly choose one, say 5. Then we update $Q(1, 5)$ as

$$\begin{aligned} Q^{new}(1,5) &= R(1,5) + 0.8 \max_{a' \in A} Q(5, a') \\ &= R(1,5) + 0.8 \max\{Q(5,1), Q(5,4), Q(5,5)\} \\ &= 100 + 0.8 \max\{0, 0, 0\} = 100. \end{aligned}$$

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Example

We are now in state 5. Since this is the target state, we have finished this episode.

We now run a second episode, starting from another a randomly chosen initial state...

Example

Randomly choose an initial state, say 3. The admissible actions are $\{1, 2, 4\}$. Randomly choose one, say 1. We update $Q(3, 1)$ as

$$\begin{aligned} Q^{new}(3,1) &= R(3,1) + 0.8 \max_{a' \in A} Q(1, a') \\ &= R(3,1) + 0.8 \max\{Q(1,3), Q(1,5)\} \\ &= 0 + 0.8 \max\{0, 100\} = 80. \end{aligned}$$

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Example

We are now in state 1, not the target state, so we go on. The admissible actions are $\{3, 5\}$. Randomly choose one, say 5. We update $Q(1, 5)$ as

$$\begin{aligned} Q^{new}(1,5) &= R(1,5) + 0.8 \max_{a' \in A} Q(1, a') \\ &= R(1,5) + 0.8 \max\{Q(5,1), Q(5,3), Q(5,5)\} \\ &= 100 + 0.8 \max\{0,0,0\} = 100. \quad \textit{(no update)} \end{aligned}$$

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Example

We are now in state 5. Since this is the target state, we have finished this episodes.

We repeat several times...

After several episodes, we get

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

Example

Recall that the optimal control is (s the initial state):

$$a_0^* = \arg\max_{a \in A} Q(s, a), \quad a_1^* = \arg\max_{a \in A} Q(a_0^*, a), \quad \ldots$$

Example

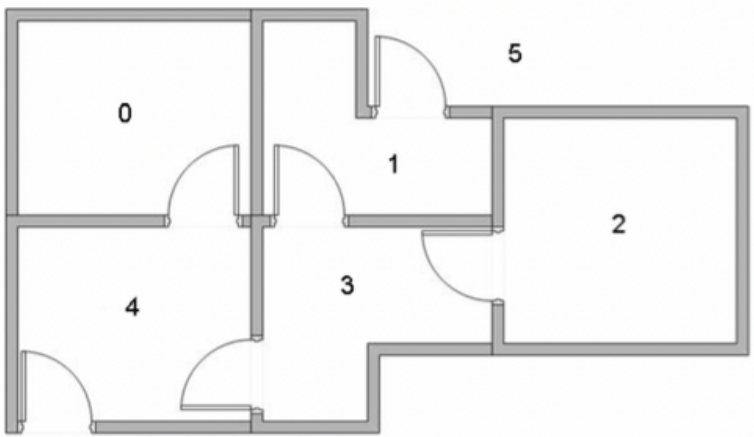
Recall that the optimal control is (s the initial state):

$$a_0^* = \arg\max_{a \in A} Q(s, a), \quad a_1^* = \arg\max_{a \in A} Q(a_0^*, a), \quad \dots$$

Example: if you start in 0, then $0 \to 4 \to 5$.

Example: if you start in 2, then either $2 \to 3 \to 4 \to 5$ or $2 \to 3 \to 1 \to 5$.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$



Lecture II: MFC with learning & characterization of Q-function in continuous-time RL

Outline

Extension to McKean-Vlasov Control

Consider the state dynamics

$$\begin{aligned} \mathrm{d}X_s^{t,\nu} &= b(s, X_s^{t,\nu}, \mathbb{P}_{X_s^{t,\nu}})\mathrm{d}s + \sigma(s, X_s^{t,\nu}, \mathbb{P}_{X_s^{t,\nu}})\mathrm{d}W_s, \\ X_t^{t,\nu} &\sim \nu, \end{aligned}$$

and the associated value function

$$\mathbb{E}\left[\int_t^T r\left(s, X_s^{t,\nu}, \mathbb{P}_{X_s^{t,\nu}}\right)\mathrm{d}s + g\left(X_T^{t,\nu}, \mathbb{P}_{X_T^{t,\nu}}\right)\right].$$

Dynamic programming principle for MFC

- MFC is time-inconsistent
  - Strict controls/without learning: Pham and Wei (2016, 2017), Bayraktar, Cosso, Pham (2018), Wu and Zhang (2019), Djete, Possamai, and Tan (2022), Talbi, Touzi, and Zhang (2023)
  - Relaxed control for MFC with learning: Gu, G., Wei, and Xu (2023)
- Time consistency property is foundation for MDP, RL, and MARL
  - Value-based method: Q learning
  - Policy-based method: Actor-Critic algorithm

DPP for MFC with learning (Gu, G. Wei, and Xu, (2023))

Idea:

- Identify the "correct" Q function: with a wrong form of Q function, Q-table will converge to different values with different initial population distribution
- Need to work with the correct "state-action" space
- Similar to the MFC theory, "lift" the state-action space into their probability measure space

Q function on the probability measure space

Define Integrated Q (IQ) function for MFC $\quad Q(s,a)$ No $\quad Q(\mu,h)$ Yes

$$Q(\mu,h) := \underbrace{\mathbb{E}\left[r(s_0,a_0,\mu)\middle| s_0\sim\mu, a_0\sim h\right]}_{\text{reward of taking } a_0\sim h}$$
$$+ \underbrace{\mathbb{E}_{s_1\sim P(s_0,a_0,\mu)}\left[\sum_{t=1}^{\infty}\gamma^t r(s_t,a_t,\mu_t)\middle| a_t\sim\pi_t^*\right]}_{\text{reward of playing optimal afterwards } a_t\sim\pi_t^*}$$

- $\mathcal{H}$ is the set of local policies $h:\mathcal{S}\to\mathcal{P}(\mathcal{A})$
- $V(\mu)=\sup_h Q(\mu,h)\to$ to find optimal policy (if exists)

$$\pi^*(\mu)\in\arg\max_h Q(\mu,h)$$

DPP for IQ Function

Bellman Equation for IQ Function (Gu, G., Wei, Xu, 2023)

For any $\mu \in \mathcal{P}(\mathcal{S})$ and $h \in \mathcal{H}$,

$$Q(\mu, h) \quad = \quad \hat{r}(\mu, h) + \gamma \sup_{h' \in \mathcal{H}} Q(\Phi(\mu, h), h').$$

- $\mathcal{H} := \{h : \mathcal{S} \to \mathcal{P}(\mathcal{A})\}$: set of local policies $\underbrace{\mathcal{P}(\mathcal{A}) \times \ldots \mathcal{P}(\mathcal{A})}_{|\mathcal{S}|}$
- Aggregated reward: $\hat{r}(\mu, h) := \sum_{s,a} \mu(s) h(s, a) r(s, a, \mu)$
- Aggregated dynamics
  - $\Phi(\mu, h) := \sum_{s,a} P(s, \mu, a) \mu(s) h(s, a)$: aggregated dynamics
  - $\mu_{t+1} = \Phi(\mu_t, h)$: distribution at time $t + 1$, flow property
- $\mathcal{H}$ is key for DPP of the IQ function

Insight of DPP

- Centralized training with decentralized execution: when rewards can be decomposed additively across agent observations

$$Q^{\pi}_{\text{global}}(\mu, h) = \sum_{s,a} Q^{\pi}_{\text{local}}\Big(\mu(s), h(s,a)\Big)\mu(s)h(s,a)$$

Two key quantities in continuous time RL

- Q-function?
- Advantage function?

First approach: stochastic policy with entropy regularization

Exploratory formulation [Wang, Zariphopoulou and Zhou (2020)]

Consider stochastic policies $\pi : [0, T] \times \mathbb{R}^d \to \mathcal{P}(A)$, and the corresponding exploratory state dynamics:

$$\mathrm{d}X_t^\pi = \tilde{b}^\pi(t, X_t^\pi)\mathrm{d}t + \tilde{\sigma}^\pi(t, X_t^\pi)\mathrm{d}W_t,$$

where

$$\tilde{b}^\pi(t,x) = \int_A b(t,x,a)\pi(\mathrm{d}a|t,x),\ \tilde{\sigma}^\pi(t,x) = \sqrt{\int_A \sigma^2(t,x,a)\pi(\mathrm{d}a|t,x)}.$$

Maximize the entropy-regularized objective function over π:

$$\mathbb{E}\left[\int_0^T \left(\tilde{r}^\pi(t, X_t^\pi) - \gamma \mathrm{KL}(\pi(\cdot|t, X_t^\pi))\right)\mathrm{d}t + g(X_T^\pi)\right],$$

where $\tilde{r}^\pi(t,x) = \int_A r(t,x,a)\pi(a|t,x)\mathrm{d}a$, and $\gamma > 0$ is a given parameter.

Stochastic policy with entropy regularization

- Stochastic policy and exploratory dynamics capture exploration: at each state, sampling actions according to a distribution.

$$\mathrm{d}X_t = b(t, X_t, a_t)\mathrm{d}t + \sigma(t, X_t, a_t)\mathrm{d}W_t, \quad a_t \sim \pi(\mathrm{d}a|t, X_t).$$

- Optimal exploratory policy is Gaussian in LQ-RL
- Based on this framework, many discrete-time RL algorithms are extended to continuous-time, e.g., temporal difference learning [Jia and Zhou (2022a)], q-learning [Jia and Zhou (2022b)], policy gradient and actor-critic learning [Jia and Zhou (2023)].

Issues with stochastic policy approach

- Inconsistent with classical control framework with deterministic policy
- Recovering a deterministic policy from a learned stochastic one is challenging (except the LQ setting).
- Exploratory dynamics requires continuously sampling independent actions at different states, which is infeasible both theoretically and practically [Jia, Ouyang and Zhang (2025)].
- Frequent sampling makes actions discontinuous over time, in contrast to deterministic policy.

New approach: directly characterize a deterministic policy!

Issues with stochastic policy approach

- Inconsistent with classical control framework with deterministic policy
- Recovering a deterministic policy from a learned stochastic one is challenging (except the LQ setting).
- Exploratory dynamics requires continuously sampling independent actions at different states, which is infeasible both theoretically and practically [Jia, Ouyang and Zhang (2025)].
- Frequent sampling makes actions discontinuous over time, in contrast to deterministic policy.

New approach: directly characterize a deterministic policy!

General idea (Cheng, G., Zhang (2025))

Approximate the (optimal) deterministic policy (DPG) in a parametric form $\{\mu_\phi\}_{\phi\in\mathbb{R}^n}$, and optimize the policy parameterization by gradient methods.

Consider maximizing the reward over $\phi \in \mathbb{R}^n$:

$$V^\phi(t,x) := \mathbb{E}\left[\int_t^T r(s, X_s^\phi, \mu_\phi(s, X_s^\phi))\mathrm{d}s + g(X_T^\phi) \,\middle|\, X_t^\phi = x\right],$$

where

$$\mathrm{d}X_s^\phi = b(s, X_s^\phi, \mu_\phi(s, X_s^\phi))\mathrm{d}s + \sigma(s, X_s^\phi, \mu_\phi(s, X_s^\phi))\mathrm{d}W_s.$$

Deterministic policy gradient in continuous-time RL

Theorem 1 (Cheng, G., Zhang (2025))

Under suitable regularity conditions,

$$\partial_\phi V^\phi(t,x) = \mathbb{E}^{t,x}\left[\int_t^T \partial_\phi \mu_\phi(s, X_s^\phi)^\top \partial_a A^\phi(s, X_s^\phi, \mu_\phi(s, X_s^\phi))\mathrm{d}s\right],$$

where $A^\phi(t,x,a) := \mathcal{L}[V^\phi](t,x,a) + r(t,x,a)$ *is the Hamiltonian, with*

$$\mathcal{L}[\varphi](t,x,a) := \partial_t\varphi(t,x) + b(t,x,a)^\top \partial_x\varphi(t,x) + \tfrac{1}{2}\,\mathrm{tr}(\sigma\sigma^\top(t,x,a)\partial_{xx}^2\varphi(t,x)).$$

- A^ϕ is analogous to the advantage function in discrete-time DPG [Silver et.al (2014)], and the q-function for stochastic policy gradient [Jia and Zhou (2023)].

Limit of discrete-time DPG

$\partial_\phi V^\phi(t,x) = \mathbb{E}^{t,x}\left[\int_t^T \partial_\phi \mu_\phi(s, X_s^\phi)^\top \partial_a A^\phi(s, X_s^\phi, \mu_\phi(s, X_s^\phi))\mathrm{d}s\right]$

Let $\beta = 0$, and consider the time discretized objective

$$J_{\Delta t}(\phi) := \mathbb{E}\left[\sum_{i=0}^{N-1} r(t_i, X_{t_i}^{\Delta t,\phi}, \mu_\phi(t_i, X_{t_i}^{\Delta t,\phi}))\Delta t + g(X_T^{\Delta t,\phi})\right],$$

where

$$X_{t_{i+1}}^{\Delta t,\phi} = X_{t_i}^{\Delta t,\phi} + b(t_i, X_{t_i}^{\Delta t,\phi}, \mu_\phi(t_i, X_{t_i}^{\Delta t,\phi}))\Delta t + \sigma(t_i, X_{t_i}^{\Delta t,\phi}, \mu_\phi(t_i, X_{t_i}^{\Delta t,\phi}))\sqrt{\Delta t}\omega_{t_i}.$$

The discrete-time DPG in [Silver et.al (2014)] gives

$$\partial_\phi J_{\Delta t}(\phi) = \mathbb{E}\Big[\sum_{i=0}^{N-1} \partial_\phi \mu_\phi(t_i, X_{t_i}^{\Delta t,\phi})^\top \partial_a A^{\Delta t,\phi}(t_i, X_{t_i}^{\Delta t,\phi}, \mu_\phi(t_i, X_{t_i}^{\Delta t,\phi}))\Delta t\Big],$$

where $A^{\Delta t,\phi}(t,x,a) := \frac{Q^{\Delta t,\phi}(t,x,a) - V^{\Delta t,\phi}(t,x)}{\Delta t}$ is the discrete-time advantage rate function.

How to learn A^ϕ?

$A^\phi(t,x,a) := \mathcal{L}[V^\phi](t,x,a) + r(t,x,a)$

Theorem 2 (Cheng, G., Zhang (2025))

Let $\hat{V} \in C^{1,2}([0,T]\times\mathbb{R}^d)$ *and* $\hat{q} \in C([0,T]\times\mathbb{R}^d \times A)$ *satisfy for all* $(t,x)\in[0,T]\times\mathbb{R}^d$,

$$\hat{V}(T,x) = g(x), \quad \hat{q}(t,x,\mu_\phi(t,x)) = 0,$$

and there exists a neighborhood $\mathcal{O}_{\mu_\phi(t,x)}$ *of* $\mu_\phi(t,x)$ *s.t.* $\forall a \in \mathcal{O}_{\mu_\phi(t,x)}$,

$$\left(\hat{V}(s,X_s^{t,x,a}) + \int_t^s (r-\hat{q})(u,X_u^{t,x,a},\alpha_u)\mathrm{d}u\right)_{s\in[t,T]} \text{ is a martingale,}$$

where

$$\mathrm{d}X_s^{t,x,a} = b(s,X_s^{t,x,a},\alpha_s)\mathrm{d}s + \sigma(s,X_s^{t,x,a},\alpha_s)\mathrm{d}W_s, \quad X_t^{t,x,a} = x,$$

and $(\alpha_s)_{s\geq t}$ *is a control process with* $\lim_{s\searrow t}\alpha_s = a$.

Then $\hat{V} = V^\phi$, *and* $\hat{q} = A^\phi$ *in the neighborhood of* μ_ϕ.

Key insight from deterministic policy

- Explore the neighborhood of current actions, rather than the entire action space
- Use data from the original state, rather than the exploratory state
- The Bellman condition only evaluates at the current action, while the stochastic policy counterpart requires

$$\int_A (\hat{q}(t,x,a) - \gamma \log \pi(a|t,x))\, \pi(\mathrm{d}a|t,x) = 0,$$

which needs an additional Monte Carlo step to approximate the integral.

Extending DPG to continuous-time MFC with learning

Theorem 3 (Cheng, G., Pham, Zhang (2026))

$$\partial_\phi V^\phi(t,x,\nu) = \int_t^T \partial_\phi A[V^\phi](s, \mathbb{P}_{X_s^{t,\nu,\phi}}, \phi)\mathrm{d}s,$$

where $A[\varphi](t,\nu,\phi) := \mathcal{L}^\phi[\varphi](t,\nu) + \langle r(t,\cdot,\nu,\phi),\nu\rangle$, *and*

$$\begin{aligned}\mathcal{L}^\theta[\varphi](t,\nu) :=& \partial_t\varphi(t,\nu) + \mathbb{E}_{\xi\sim\nu}\Big[b(t,\xi,\nu,\theta)\cdot\partial_\mu\varphi(t,\nu)(\xi) \\ &+ \frac{1}{2}\Sigma(t,\xi,\nu,\theta)\colon\partial_v\partial_\mu\varphi(t,\nu)(\xi)\Big].\end{aligned}$$

- A model-free characterization holds for $A[V^\phi](t,\nu,\phi)$.

References

Today's lecture is based on the following materials and the references therein.

- Gu, Guo, Wei, Xu. Dynamic programming principle for mean field controls with learning. Operations Research, 71 (4), 1040-1054, 2023. (Arxiv: 1911.07314).
- Gu, Guo, Wei, Xu. Mean-Field controls with Q-learning for cooperative MARL: convergence and complexity analysis. SIAM Journal on Mathematics of Data Science. (4), 1168-1196, 2021. (ArXiv:2002.04131).
- Guo, Xu, Zariphopoulou. "Entropy regularizations for mean field games with learning." Mathematics of Operations Research, 47 (4), 3239-3260, 2022. (ArXiv 2010.00145 and SSRN 3702956).
- Gu, Guo, Wei, Xu. Multi-agent reinforcement learning, a decentralized network approach." Mathematics of Operations Research. 50 (1), 506-536, 2025.
- Cheng, Guo, Zhang. Bridging discrete and continuous RL: stable deterministic policy gradient with martingale characterization. arXiv:2509.23711.
- Cheng, Guo, Pham, Zhang, *Model-free sensitivity formula for McKean-Vlasov SDEs with application to policy gradient learning, (2026), working paper.*