Machine learning and stochastic control

Lecture 1

Jyväskylä Summer School
August 4-8, 2025

Huyên PHAM [1]

1. Ecole Polytechnique, CMAP

Lecturer : Huyên PHAM

- Full Professor at Ecole Polytechnique (2024-), Department of Applied Mathematics (CMAP)

- Research areas : Stochastic Analysis and Control, Mathematical Finance, Numerical Probabilities, Data Science (Deep learning, Reinforcement learning, Generative modeling)

- https://sites.google.com/site/phamxuanhuyen/

- Email : huyen.pham@polytechnique.edu

Planning and organization

- Five sessions on Monday-Friday 12h-14h : August 4-8, 2025
- **Passing** : Attending at least 4 of the 5 lectures (attendance list to be circulated)
- Documents and slides on : MA1

Objectives of the course

- Explore the synergy between machine learning and stochastic control :
  - Address challenges in decision-making problems under uncertainty : curse of dimensionality, complex nonlinear dynamics, unknown environment
  - Combine theoretical insights with computational tools for solving real-world problems
  - Alignment with current research trends

Outline

1. Foundations
    - Machine learning (ML) meets stochastic control
    - Basics of reinforcement learning (RL)
    - Continuous time stochastic control RL
2. Machine learning techniques for control
    - Neural networks algorithms for PDEs and HJB equations
    - Policy gradient methods in continuous time
    - Q-learning and approximations in continuous time

Some references

1. **Part I**

🗎 Hambly B., Xu R., Yang H. : Recent advances in reinforcement learning in finance, *Mathematical Finance*, 2023, 33(3), 437-503, arXiv:2112.04553

🗎 Sutton R., Barto A. : Reinforcement learning, and introduction, 2nd ed. MIT Press, pdf file

🗎 Wang H., Zariphopoulou T., Zhou XY : Reinforcement learning in continuous time and space : a stochastic control approach, *JMLR*, 2020, 21, 1-34, pdf file

2. **Part II**

🗎 Beck C., Hutzenthaler M., Jentzen A., Kuckuck B. : "An overview on deep learning-based approximation methods for partial differential equations". arXiv:2012.12348, *Discrete and continuous dynamical systems, B*, 2023, 28(6), 3697-3746.

🗎 Germain M., Pham H., Warin X. : Neural networks-based algorithms for stochastic control and PDEs, *Machine learning and data sciences for financial markets*, CUP, 2023, arXiv:2101.08068

🗎 Jia Y., Zhou XY : Policy gradient and actor-critic learning in continuous time and space : theory and algorithms, *JMLR*, 2022, 23, 1-50, pdf file

🗎 Jia Y., Zhou XY : q-learning in continuous time, *JMLR*, 2023, 24, 1-61, pdf file

Outline

Part I: Foundations

Stochastic control in a nutshell

- **Stochastic control** : optimization of dynamical systems in a random environment
- Key components :
  - State dynamics modeled e.g. in continuous time by stochastic differential equations (SDEs) :

$$\mathrm{d}X_t = b(X_t, \alpha_t)\mathrm{d}t + \sigma(X_t, \alpha_t)\mathrm{d}W_t \tag{1}$$

  - Objective : maximize/minimize over control/action/decision $\alpha = (\alpha_t)_t$ performance/cost over time under uncertainty :

$$J(\alpha) = \mathbb{E}\Big[\int_0^T f(X_t, \alpha_t)\mathrm{d}t + g(X_T) \Big].$$

▶ Resolution/characterization by dynamic programming or maximum principle

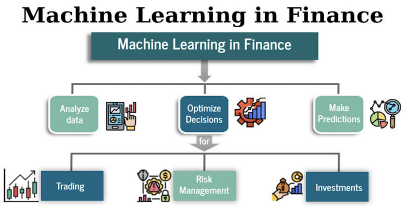
- Hamilton-Jacobi-Bellman equation
- Forward-backward SDE

Applications in finance and beyond

- **Portfolio optimization** : allocate assets dynamically to maximize returns
  - X_t capital/wealth, P&L at time t
  - α_t : number of shares (or amount) invested in assets at time t
  - f, g : utility functions $\leftrightarrow$ preferences criteria of the investor
- **Risk management** : mitigate exposure to adverse market conditions
- Many more applications :
  - Robotics and automation : autonomous vehicles
  - Energy systems : smart grids and energy storage
  - Heathcare : personalized treatment strategies

Why machine learning in stochastic control ?

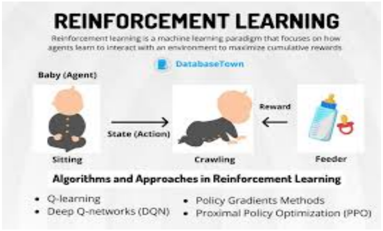
- Challenges :
  - Explicit solutions are rarely available $\rightarrow$ need efficient numerical methods
  - Curse of dimensionality in large state/action spaces
  - Unknown (or misspecified) dynamics or reward functions
- Solutions with ML :
  - Neural networks (deep learning) for approximating value function
  - Data-driven learning of policies



ML meets stochastic control

• Approximation of policies and value functions using supervised and unsupervised ML

• Reinforcement learning (RL) : a data-driven approach to stochastic control by interacting with the unknown environment by *trial and error*


REINFORCEMENT LEARNING

Reinforcement learning is a machine learning paradigm that focuses on how agents learn to interact with an environment to maximize cumulative rewards

Algorithms and Approaches in Reinforcement Learning

- Q-learning
- Deep Q-networks (DQN)
- Policy Gradients Methods
- Proximal Policy Optimization (PPO)

Figure – A toy illustration !

Outline

Part I: Foundations

Markov decision process (MDP)

MDP is defined by a quadruple $(\mathcal{X}, A, P, r = (f, g))$

- $\mathcal{X}$: state space in which the discrete-time process $(X_t)_{t\in\mathbb{N}}$ is valued
- A : action space in which the control/decision $(\alpha_t)_t$ is valued
- State dynamics determined by

$$X_{t+1} \quad \sim \quad P_t(X_t, \alpha_t)$$

  with a transition probability $(t, x, a) \in \mathbb{N} \times \mathcal{X} \times A \mapsto P_t(x, a, \mathrm{d}x') \in \mathcal{P}(\mathcal{X})$.

- Reward (reinforcement) :
  - running reward $f(x, a)$ obtained in state x when choosing action a
  - terminal reward (case of finite horizon) $g(x)$
  - Discount factor $\beta \in [0, 1]$ (case of infinite horizon)

Policy

- A (Markovian) policy π is a sequence $(\pi_t)_t$ which can be either
  - **deterministic** when $\pi_t : \mathcal{X} \to A$, i.e., $\pi_t(x)$ represents the action value chosen at time t in state x
  - **randomized/stochastic** when $\pi_t : \mathcal{X} \to \mathcal{P}(A)$, i.e., $\pi_t(x, \mathrm{d}a)$ represents the probability of choosing an action at time t in state x
- We say that a control α is drawn (or follows) a policy π, denoted $\alpha \sim \pi$, when
  - $\alpha_t = \pi_t(X_t)$ (case of deterministic policy), for all t
  - $\alpha_t \sim \pi_t(X_t)$ (case of randomized policy), for all t

▶ Goal : learn a policy/control that maximizes the sum of rewards

Evaluation of a policy : value functions

• **State value function** (over a finite horizon $T \in \mathbb{N}^*$) of a policy $\pi = (\pi_t)_t$:

$$V_t^\pi(x) \quad = \quad \mathbb{E}_\pi\Big[\sum_{s=t}^{T-1} f(X_s, \alpha_s) + g(X_T) | X_t = x \Big], \quad t \in \{0, \ldots, T\}, x \in \mathcal{X},$$

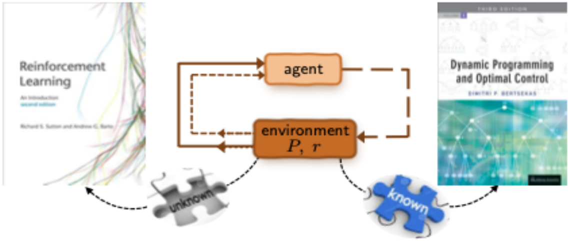
Here, $\mathbb{E}_\pi$ is expectation w.r.t. trajectory X when $\alpha \sim \pi$.

• Q-**function** of π :

$$Q_t^\pi(x, a) \quad = \quad \mathbb{E}_\pi\Big[\sum_{s=t}^{T-1} f(X_s, \alpha_s) + g(X_T) | X_t = x, \alpha_t = a \Big], \quad t \in \{0, \ldots, T\}, (x, a) \in \mathcal{X} \times A.$$

Remark : $V_t^\pi(x) = \mathbb{E}_{a \sim \pi_t(x)}\big[Q_t^\pi(x, a) \big]$.

Searching for optimal policy



Goal : find an **optimal policy** π^* that maximize V^π

$\rightarrow$ Optimal value function/Q function :

$$V := \sup_\pi V^\pi, \qquad Q := \sup_\pi Q^\pi$$

Remark : $V_t(x) = \sup_{a\in A} Q_t(x,a)$.

A key tool for MDP and RL : Dynamic programming (DP)

• MDP is a global dynamic optimization

• Dynamic programming is a method to reduce the optimization problem to local/static optimization problem

Bellman's principle of optimality (57) :

"An optimal policy has the property that, whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

Bellman equation for value functions : Backward recursion from one-step transition

- For a fixed policy $\pi = (\pi_t)_t$,

$$\begin{cases} V_T^\pi(x) = \; g(x) \\ V_t^\pi(x) = \; \mathbb{E}_{a\sim\pi_t(x)}\Big[\underbrace{f(x,a) + \mathbb{E}_{x'\sim P_t(x,a)}\big[V_{t+1}^\pi(x')\big]}_{Q_t^\pi(x,a)}\Big], \quad t = 0,\ldots,T-1, \end{cases} \tag{2}$$

and

$$\begin{cases} Q_T^\pi(x,a) = \; g(x) \\ Q_t^\pi(x,a) = \; f(x,a) + \mathbb{E}_{\substack{x'\sim P_t(x,a)\\ a'\sim\pi_{t+1}(x')}}\Big[Q_{t+1}^\pi(x',a')\Big] \end{cases} \tag{3}$$

- For optimal value and Q functions

$$\begin{cases} V_T(x) = \; g(x) \\ V_t(x) = \; \sup\limits_{a\in A} \underbrace{\Big\{f(x,a) + \mathbb{E}_{x'\sim P_t(x,a)}\big[V_{t+1}(x')\big]\Big\}}_{Q_t(x,a)} \end{cases} \tag{4}$$

and

$$\begin{cases} Q_T(x,a) = \; g(x) \\ Q_t(x,a) = \; f(x,a) + \mathbb{E}_{x'\sim P_t(x,a)}\big[\sup\limits_{a'\in A} Q_{t+1}(x',a')\big] \end{cases} \tag{5}$$

$\to$ Optimal (deterministic) policy : $\pi^* = (\pi_t^*)$ from the Q-value function :

$$\pi_t^*(x) \quad \in \; \arg\max_{a\in A} Q_t(x,a).$$

Proof of Bellman equations

- From definition of Q^π, and law of conditional expectation, we get :

$$Q_t^\pi(x,a) = \mathbb{E}_\pi\Big[f(X_t,\alpha_t) + \underbrace{\mathbb{E}_\pi\big[\sum_{s=t+1}^{T-1} f(X_s,\alpha_s) + g(X_T)|X_{t+1}\big]}_{V_{t+1}^\pi(X_{t+1})}\Big|X_t = x, \alpha_t = a\Big]$$

$$= f(x,a) + \mathbb{E}_{x' \sim P_t(x,a)}\big[V_{t+1}^\pi(x')\big]. \tag{6}$$

By using the relation $V_s^\pi(x) = \mathbb{E}_{a\sim\pi_s(x)}[Q_s^\pi(x,a)]$ for $s = t$, this yields the Bellman equation (2) for V^π, and for $s = t+1$, this gives the Bellman equation (3) for Q^π.

- For the Q-value function, and using (6), we have

$$Q_t(x,a) = \sup_\pi Q_t^\pi(x,a) = \sup_\pi \Big\{f(x,a) + \mathbb{E}_{x'\sim P_t(x,a)}\big[V_{t+1}^\pi(x')\big]\Big\}$$

$$= f(x,a) + \mathbb{E}_{x'\sim P_t(x,a)}\big[V_{t+1}(x')\big], \tag{7}$$

by definition of $V = \sup_\pi V^\pi$.

Proof of Bellman equations (Ctd)

- For the optimal value function, and using (2), we have

$$\begin{aligned} V_t(x) \;=\; \sup_\pi V_t^\pi(x) \quad &= \quad \sup_\pi \mathbb{E}_{a\sim\pi_t(x)}\Big[f(x,a) + \mathbb{E}_{x'\sim P_t(x,a)}\big[V_{t+1}^\pi(x')\big]\Big] \\ &\leq \quad \sup_{a\in A}\Big[f(x,a) + \mathbb{E}_{x'\sim P_t(x,a)}\big[V_{t+1}(x')\big]\Big] \;=\; \sup_{a\in A} Q_t(x,a), \end{aligned} \tag{8}$$

by (7). To prove the reverse inequality, let us consider the (deterministic) policy π^* defined by $\pi_t^*(x) \in \arg\max_a Q_t(x,a)$ for all t, x. Then, we prove by backward induction that for all $t = 0,\ldots,T$,

$$V_t^{\pi^*}(x) \;\geq\; \sup_{a\in A} Q_t(x,a) \;\geq\; V_t(x), \quad x \in \mathcal{X}.$$

For $t = T$, this is clear since $V_T^{\pi^*} = Q_T = V_T = g$. Suppose now that $V_{t+1}^{\pi^*} \geq \sup_{a\in A} Q_{t+1}(\cdot,a) \geq V_{t+1}$. From the Bellman equation (2) for V^π, we then have

$$\begin{aligned} V_t^{\pi^*}(x) \quad &= \quad f(x,\pi_t^*(x)) + \mathbb{E}_{x'\sim P_t(x,\pi_t^*(x))}\big[V_{t+1}^{\pi^*}(x')\big] \\ &\geq \quad f(x,\pi_t^*(x)) + \mathbb{E}_{x'\sim P_t(x,\pi_t^*(x))}\big[V_{t+1}(x')\big] \;=\; Q_t(x,\pi_t^*(x)) \;=\; \sup_{a\in A} Q_t(x,a) \geq V_t(x), \end{aligned}$$

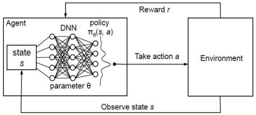
by (7), (8), hence the required relation at t. Since $V^{\pi^*} \leq V$, this shows the equality $V_t(x) = \sup_{a\in A} Q_t(x,a) = V_t^{\pi^*}(x)$ for all t, x, hence the optimality of π^*. Together with (7), this yields the Bellman equation (4) for V, and also the Bellman equation (5) for Q.

From MDP to RL

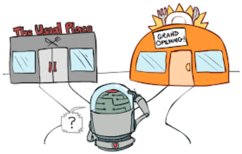
• Model-based approach : DP algorithms require to know the environment, i.e., transition probability P and reward function $r = (f, g)$

• When P, r are unknown (model-free setting), the MDP becomes a RL problem

- **Agent-environment interface.**



- **Exploration vs Exploitation, Randomized vs Deterministic Policy.**



Classification of RL algorithms

In a model-free setting, the different learning framework based on samples and observations of state/reward are

- **Online learning** : the agent is in state X_t, take an action α_t, observes the reward $f_t = f(x_t, \alpha_t)$, is in the new state $X_{t+1} \sim P_t(X_t, \alpha_t)$, and proceeds forward again.
- **Episodic learning** : Sequence of episodes where on each episode one follows a policy during a certain period for exploration, and one initializes again the state and starts a new episode.
- **Learning with generative model** : Simulator of the environment that takes as input at any time t state/action (x, a), and generates as output the reward/next state (f_t, x').

▶ In this context, there are mainly two classes of RL methods :

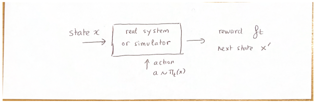
- Value-based methods
  - Learn a representation of the value and Q functions
  - Policy is then derived implicitly from the value function
- Policy-based methods
  - Learn directly a representation of the policy
  - Value function is estimated afterwards from the learnt policy.

Temporal-difference (TD) learning

For a policy π, recall the Bellman equation satisfied by V^π :

$$V_t^\pi(x) = \mathbb{E}_{\substack{a \sim \pi_t(x) \\ x' \sim P_t(x,a)}} \Big[f(x,a) + V_{t+1}^\pi(x') \Big].$$

- Estimate V^π by stochastic approximation (Robbins-Monro algorithm) :
  - Initialize at episode $k = 0$ an estimate $\hat{V}^\pi$ of V^π
  - Generate (from real system or simulator) episodic trajectories/reward samples (t, x, a, f_t, x') from policy π, $t = 0, \ldots, T-1$, f_T final reward



  - Update estimate $\hat{V}^\pi$ at the next episode by

$$\hat{V}_t^\pi(x) \longleftarrow (1-\eta) \underbrace{\hat{V}_t^\pi(x)}_{\text{current estimate}} + \eta \Big[\underbrace{f_t + \hat{V}_{t+1}^\pi(x')}_{\text{prediction at next state}} \Big] \qquad (9)$$

with $\hat{V}_{t+1}^\pi(x') = f_T$ when $t = T-1$, and where $\eta = \eta_t(x,a) \in (0,1)$ is the learning rate.
Updating rule (9) is also written as

$$\hat{V}_t^\pi(x) \longleftarrow \hat{V}_t^\pi(x) + \eta \big[\underbrace{f_t + \hat{V}_{t+1}^\pi(x') - \hat{V}_t^\pi(x)}_{\text{TD error } \delta_t} \big],$$

and usually referred to as TD(0) algorithm.

Variations of TD

- TD(1) (Monte-Carlo method) : update value function estimate according to

$$\hat{V}_t^\pi(x) \longleftarrow \hat{V}_t^\pi(x) + \eta \sum_{s=t}^{T-1} \delta_s$$

- More generally TD(λ) for $\lambda \in [0,1]$:

$$\hat{V}_t^\pi(x) \leftarrow \hat{V}_t^\pi(x) + \eta \sum_{s=t}^{T-1} \lambda^{s-t} \delta_s$$

Remark : Large λ close to 1 permits a more accurate estimation of value function, but with higher variance.