Financial Data Science

Lecture 8
Deep Reinforcement Learning

Video tutorial:
https://www.youtube.com/watch?v=Yd3H_h-7C58

Liu Peng
liupeng@smu.edu.sg

Introduction to Deep Reinforcement Learning (DRL)

- DRL combines deep neural networks (DNNs) with reinforcement learning (RL) principles to enable agents to learn optimal actions through interaction with the environment.
- Agent: Learns and makes decisions.
- Environment: Context or system where the agent operates.
- Policy: Strategy the agent follows to select actions.
- Reward Signal: Feedback to reinforce good actions.
- Goal: Agents aim to maximize cumulative rewards by adjusting their actions through trial-and-error exploration.

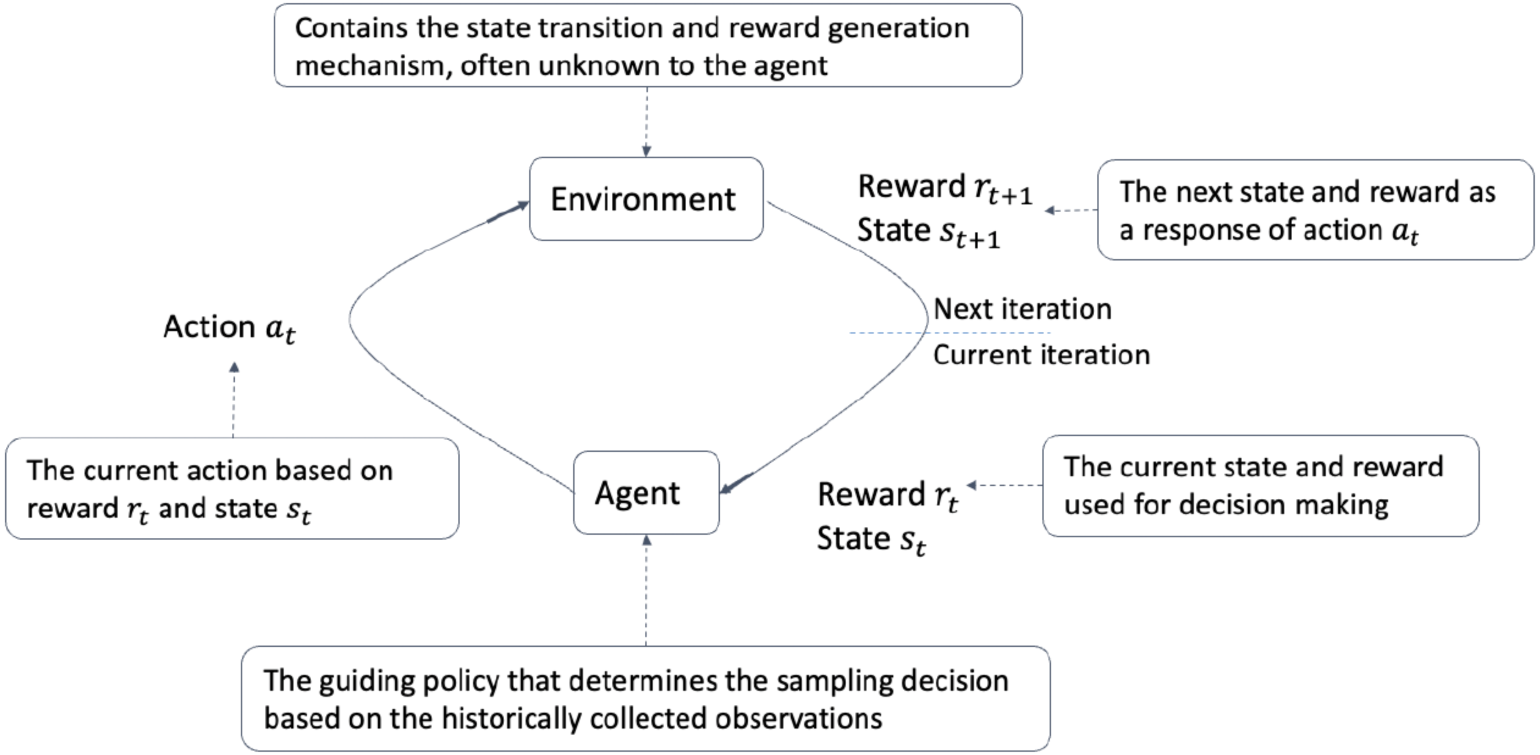


Figure 1: Iterative interaction between the agent and the environment.

Interaction between Agent and Environment

Value functions

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots$$

$$G_t = R_{t+1} + \gamma G_{t+1}$$

$$V^\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

$$V^\pi(s) = \mathbb{E}_\pi[R_{t+1} | S_t = s] + \gamma \mathbb{E}_\pi[G_{t+1} | S_t = s]$$

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma V^\pi(s')]$$

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

$$Q^\pi(s, a) = \sum_{s',r} p(s', r | s, a)[r + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s', a')]$$

DRL for Portfolio Optimization

Objective:

- Dynamically allocate assets to maximize returns, manage risk, or achieve specific financial goals through learned investment policies.

DRL Formulation:

- **Agent:** Portfolio manager making sequential investment decisions.
- **Environment:** Market conditions represented by financial indicators.
- **Actions:** Adjustments to asset weights within a portfolio.
- **Reward:** Risk-adjusted returns, e.g., Sharpe ratio or cumulative returns minus transaction costs.

Key Advantages:

- Adapts to changing market environments.
- Captures complex, nonlinear market dynamics.
- Integrates various constraints (e.g., risk, transaction costs, liquidity).

DRL in Portfolio Optimization

$$(S, A, P, R, \gamma)$$

$$s_t = \begin{bmatrix} P_t \\ I_t \\ \boldsymbol{w}_{t-1} \end{bmatrix}$$

$$a_t = \boldsymbol{w}_t = \begin{bmatrix} w_{1,t} \\ w_{2,t} \\ \vdots \\ w_{N,t} \end{bmatrix}, \quad \text{subject to} \quad \sum_{i=1}^{N} w_{i,t} = 1.$$

$$r(s_t, a_t) = \log\left(\sum_{i=1}^{N} w_{i,t} \frac{P_{i,t}}{P_{i,t-1}}\right).$$

$$p(s_{t+1}|s_t, a_t) = \mathbb{P}(S_{t+1} = s_{t+1}|S_t = s_t, A_t = a_t),$$

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, \pi(s_t))|s_0\right],$$

DRL for Hedging

Goal: Manage risk in derivative transactions (e.g., options).

Traditional Approach: Delta hedging (BSM model). Assumes frictionless, continuous-time markets.

Real-World Problems:

- **Discrete Time:** Hedging occurs periodically (e.g., daily), not continuously.
- **Transaction Costs:** Buying/selling the underlying asset incurs costs.
- **Imperfect Replication:** Leads to tracking errors and mis-hedging risk.
- **Need:** A strategy that balances minimizing hedging errors and reducing transaction costs, adapting to market dynamics.

Applying RL to Derivative Hedging

Framework: Model the hedging problem as a Markov Decision Process.

State (s_t): Can include:

- Time to maturity (t)
- Underlying asset price (S_t)
- Current hedge position (N_t)
- Option price (C_t), Delta (Δ_t) (optional, sometimes learned)
- Strike price (K)

Action (a_t): The new hedge position (N_{t+1}) or the change in position (δN_t). Can be discrete or continuous.

Reward (r_t): Designed to achieve the hedging objective. Common forms:

- Minimize P&L variance: Penalize the difference between the hedge portfolio's value change and the option's value change.
- Mean-Variance Optimization: $r_t = \mathbb{E}[\delta\Pi_t] - \frac{\lambda}{2}Var[\delta\Pi_t]$ (Balances expected P&L change and its variance, factoring in transaction costs).
- Accounting P&L: Immediate reward based on step-by-step change in net portfolio value.

Key RL Methods in Derivative Hedging

QLBS (Halperin, 2017):

- Applied Q-learning to option hedging in a Black-Scholes world.
- Minimized terminal variance.
- Used discrete states/actions, no transaction costs initially.

Deep Hedging (Buehler et al., 2019):

- Used neural networks as function approximators.
- Incorporated transaction costs and convex risk measures.

Automated Hedging (Ritter & Kolm, 2019):

- RL agent learns optimal strategy considering transaction costs.
- Used Q-learning/SARSA with continuous states, discrete actions.
- Showed RL comparable to delta hedging variance but lower cost.

Advanced Algorithms (Du et al., 2020):

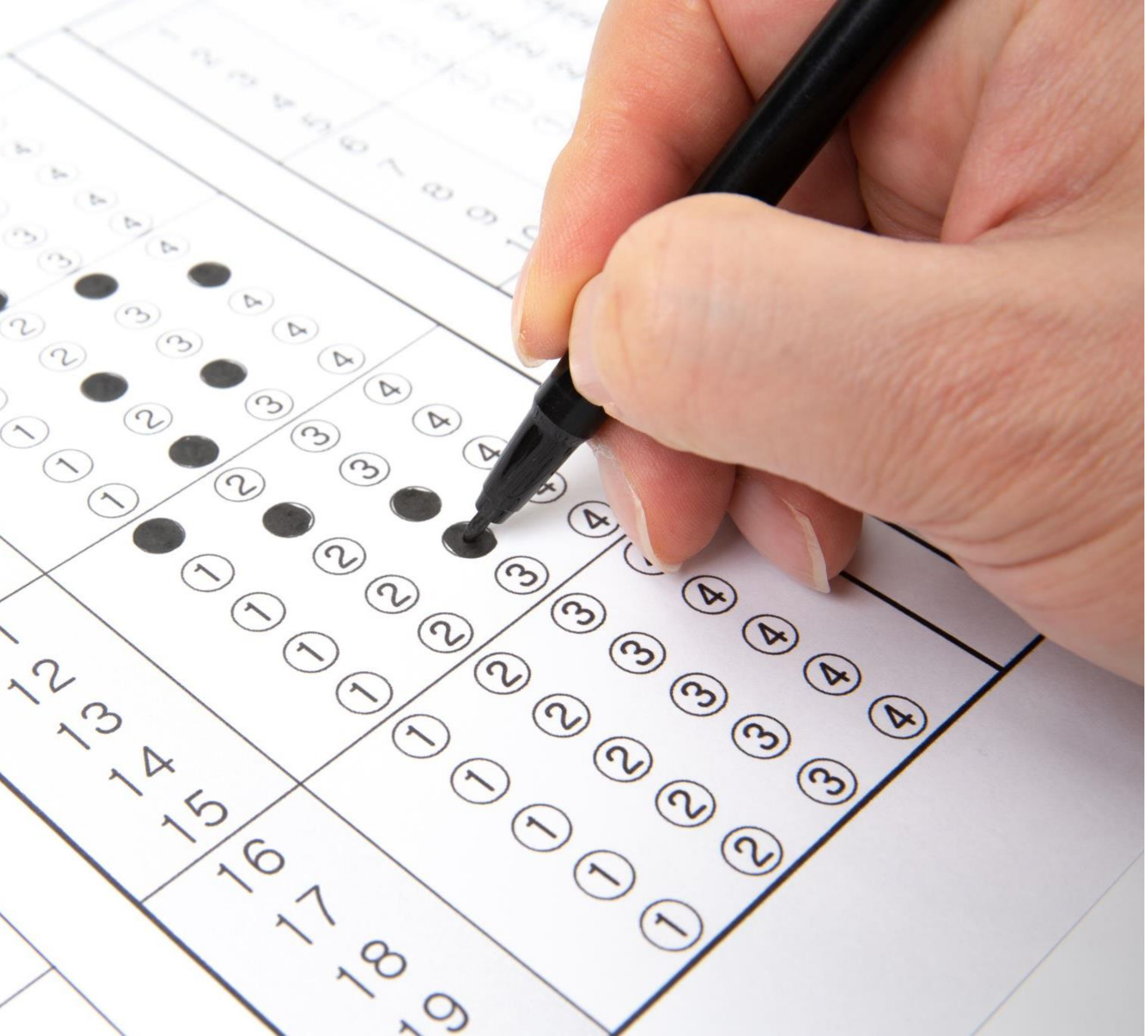
- Applied DQN, PPO to learn policy/action function directly.
- PPO showed strong performance (delta neutrality, training time/data).
- Learned option price/delta implicitly.

Continuous Actions (Cao et al., 2021):

- Used DDPG for continuous state and action spaces, reducing discretization error.
- Introduced decoupled reward (expectation & std dev of cost).
- Favored accounting P&L for faster learning.
- Outperformed traditional delta hedging with transaction costs & stochastic volatility.

Beyond Delta (Cao et al., 2023):

- Hedging Gamma & Vega using options as instruments.
- Used Distributional RL for more nuanced risk management (VaR, CVaR).
- Included Greeks in the state space.



In-class quiz