FinCatch
Coding task – hard
August 9, 2024

Context

In the rapidly evolving landscape of financial technology, artificial intelligence plays a crucial role in forecasting, decision-making, and risk management. This assessment is designed to evaluate your ability to design, implement, and critically analyze AI systems in a financial context. You have 48 hours to complete this task, allowing you to showcase both your technical skills and your ability to manage a complex project within a realistic timeframe.

Problem statement

You are tasked with developing an AI system for a quantitative trading firm. The system consists of three main components: a Large Language Model (LLM) for market analysis, an Agent for decision-making, and an Environment that simulates market conditions. Your goal is to implement these components, analyze their performance, and propose improvements.

Q1 MLP Network for LLM Simulation (Market Analyzer)

Implement a Multi-Layer Perceptron (MLP) to simulate an LLM that analyzes market conditions.

- Input: A number between 1 and 100, representing various market indicators.
- Output: A number $k \in [1, 10000]$, representing the LLM's market analysis.
- Special cases:
  - When input is 1-9, output should be 1111-9999 respectively.
  - For inputs 10-100, output should be any other value in the range.

Q2 Agent Policy Network (Decision Maker)

Design a network that takes the LLM output and decides on a trading action $\alpha \in [1, 1000]$. Implement the following reward structure:

- If LLM output is XXXX and action α contains:
  - One X: 10 points
  - Two X's: 20 points
  - Three X's: 100 points
  - All other cases: 0 points

For example:

- Input 1 $\rightarrow$ LLM $\rightarrow$ state = 1111 $\rightarrow$ Agent $\rightarrow$ Action = 111 $\rightarrow$ returns reward = 100;

- Input 2 → LLM → state = 3333 → Agent → Action = 313 → returns reward = 20.
- Input 3 → LLM → state = 3333 → Agent → Action = 124 → returns reward = 0.

Q3 Environment simulation (Market simulator)

In real-world scenarios, you may not have a predefined reward function, and the Agent needs to interact with the environment multiple times before receiving a reward. Design an environment function, a reward function, and use reinforcement learning to complete the following test cases:

Test Case 1:

Input 1 → LLM → state = 1111 → Agent → Action = 111 → Environment: returns the number of matching digits → State = 3 → LLM → state = 3333 → Agent → Action = 333 → Environment: returns the number of matching digits → State = 3 → Done: state matches the previous result → If state = 3, return reward = 100/2, where 2 represents the number of interactions with the environment.

Test Case 2:

Input 1 → LLM → state = 1111 → Agent → Action = 199 → Environment: returns the number of matching digits → State = 2 → LLM → state = 2222 → Agent → Action = 213 → Environment: returns the number of matching digits → State = 1 → LLM → state = 1111 → Agent → Action = 111 → Environment: returns the number of matching digits → State = 3 → LLM → state = 3333 → Agent → Action = 333 → Environment: returns the number of matching digits → State = 3 →→Done: state matches the previous result → If state = 3, return reward = 100/4, where 4 represents the number of interactions with the environment.

- If the interaction with the environment reaches 10 rounds without stopping, return reward = 0.
- If at the end, the state is not equal to 3, return reward = 0.

In shorts, the episode should end when:

- The state matches the previous state (successful prediction) 10 rounds of interaction are reached
- If final state is 3: 100 / (number of interactions)
- Otherwise: 0

Discussion on the question

If the policy network above is not "pre-trained", you might find it difficult to converge during training. Why do you think this happens? Can you suggest clever techniques/methods to efficiently train the Agent's Policy network? Provide analysis and code results.

Future directions and open-ended exploration

What insights into AI agent design did you gain from the above tasks? Please use any open-source finance-related projects, dataset, and packages to make the agent more realistic. Provide your attempts and analysis.

Reminder

Provide well-commented code for all implementations, and include a comprehensive report describing your approach, analysis, and findings.

There are no strictly right or wrong answers for the open-ended sections. We're interested in your thought process, creativity, and ability to tackle complex problems in AI and finance.

Good luck!