MACHINE LEARNING RESEARCHER TASK

Title: Optimize transaction costs based on the data

Architecture Overview:

The trading model generated was based on various research papers that I reviewed. The reinforcement learning framework—specifically, Q-learning—is the foundation of the architecture. This model's primary goal is to reduce transaction costs by instantly and dynamically modifying trade activities. Inspired by (Ritter, 2017), the model includes state-action framework where Q-learning progressively learns the value of different trading actions under realistic market conditions. Instead of making big, single trades that could result in greater costs, the model learns an effective execution policy that handles slippage and market impact by structuring trades over time. The architecture enables the model to make cost-efficient trading decisions based on historical price data.

Trading Schedule Methodology:

The model generates trade schedules using Q-learning to balance execution costs with timely trade completion, splitting larger trade orders into smaller timed trades. This method is inspired by the research showing the effectiveness of reinforcement learning in reducing market impact through sequential trade decisions (Woo Jae Byun, 2023). By calculating actions based on price movements, order size, and market conditions, the model autonomously adjusts to the conditions without needing any predefined cost functions (Maochun Xu).

Research Influence and Design Decisions

The model is mainly based on the paper by (Ritter, 2017). It represents a great methodology as Q-learning as a strategy for cost-efficient trading through a utility-maximizing framework that aligns with market impact and risk tolerance. This also aligns with the further research by (Woo Jae Byun, 2023), which emphasizes on the reinforcement learning's ability to manage large order executions by splitting them across time, reducing slippage and managing real-time trading constraints.

Results of the model:

When the model was tested across the AAPL_Quotes_Data.csv dataset, the output, i.e. optimal trading schedule to minimize transaction costs was stored in a JSON format with 'time-stamp' and 'number of shares'.

Example output:


```
Optimal Trading Schedule:
[
    {
        "timestamp": "2024-07-01 13:30:00+00:00",
        "shares": 400
    },
    {
        "timestamp": "2024-07-01 13:31:00+00:00",
        "shares": 120
    },
    {
        "timestamp": "2024-07-01 13:32:00+00:00",
        "shares": 480
    }
]
```


Backtest Results:

The model was further tested against the TWAP and VWAP methods on the merged bid ask ohlcv dataset provided. Their effectiveness was measured based on Total Slippage and Total Market Impact. Based on the backtesting results, the model's hyperparameters were fine-tuned to make the performance better. The results show that the model performed better than TWAP but VWAP strategy outperformed the model's performance.

```
TWAP Total Slippage: 1325742783.2141025
TWAP Total Market Impact: 0.0023717446831659807
VWAP Total Slippage: 6598256.573898577
VWAP Total Market Impact: 0.00015594395974547336
Q-Learning Model Total Slippage: 1279449046.0000007
Q-Learning Model Total Market Impact: 0.00019435621403511008
```

Detailed comparison with a help of data frame:

```
Detailed Comparison DataFrame:
           Strategy  Total Slippage  Total Market Impact
0              TWAP    1.325743e+09             0.002372
1              VWAP    6.598257e+06             0.000156
2  Q-Learning Model    1.279449e+09             0.000194
```

Backtesting results of the model saved in JSON format on this dataset:


```
[
    {
        "timestamp": "2023-09-12 13:30:00+00:00",
        "shares": 900
    },
    {
        "timestamp": "2023-09-12 13:31:00+00:00",
        "shares": 100
    }
]
```


References

1. Ritter, G. (2017). [Machine Learning For Trading].
2. Maochun Xu, Z. L. et. all [Deep Reinforcement Learning for Quantitative Trading]
3. Woo Jae Byun, et. all. [Practical Application of Deep Reinforcement Learning to Optimal Trade Execution.](2023)