CS221 Course Project Report: Automated Stock Trading Using Deep Reinforcement Learning: A TD3 Approach

Joe Wang, Danica Xiong, Tony Xie

December 6, 2023

Abstract

This project presents the development and application of a Twin Delayed Deep Deterministic Policy Gradient (TD3) agent for stock trading automation. Utilizing deep reinforcement learning, the agent operates within a complex environment comprising multiple stock tickers, with a focus on optimizing portfolio allocations to maximize returns. The agent's architecture integrates Actor and Critic networks, which process a feature vector encompassing daily closing prices and various financial indicators. A significant aspect of the project is the use of Replay Memory for efficient learning and decision-making. The model's training involves generating episodes and experiences, leveraging normalization and utility functions for effective data processing. Our results demonstrate the agent's capability to execute trading strategies, analyzed under model-based, random, and hold conditions. The project not only showcases the potential of TD3 in financial applications but also sets the stage for future enhancements in automated trading systems.

1 Introduction

In recent years, the application of Deep Reinforcement Learning (DRL) in financial markets has gained significant attention, offering innovative approaches to algorithmic trading. This project aims to harness the capabilities of DRL, particularly the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm, to automate stock trading strategies. The TD3 algorithm is known for its efficiency in handling

continuous action spaces, making it an apt choice for stock trading environments where decisions are often not binary but require a nuanced approach.

Our work revolves around constructing and training a TD3 agent capable of managing a portfolio consisting of multiple stock tickers. The heart of the agent lies in its sophisticated neural network architecture, comprising Actor and Critic networks. These networks are designed to process a comprehensive feature vector that includes daily closing prices along with an array of financial indicators, each contributing to the decision-making process.

The project's primary objective is to train the agent to optimize portfolio allocations to maximize returns, a challenging yet critical task in financial trading. The training process involves simulating a dynamic market environment, where the agent iteratively learns the most beneficial actions—buying, selling, or holding specific quantities of each stock ticker. By leveraging the principles of DRL, the agent is expected to learn complex trading strategies that go beyond traditional rule-based systems, adapting to market changes and evolving its tactics over time.

This report details the methodology employed in developing the TD3 agent, the training process, the challenges faced, and the outcomes of the project. The results are promising, demonstrating the agent's ability to formulate effective trading strategies under various market conditions, and paving the way for further advancements in automated trading systems.

2 Methodology

2.1 Data Preprocessing

2.1.1 Data Acquisition and Initial Processing

The project utilized financial data sourced from Yahoo Finance (`yfinance` library), focusing on a selection of index ETFs: QQQ, SPY, IWM, and TLT. These were chosen as a proof of concept to represent a broad spectrum of the market, circumventing the need for detailed company-specific financial information. The raw data included standard trading metrics like Open, Close, High, Low, and Volume, which underwent rigorous cleaning to eliminate missing or anomalous values.

2.1.2 Feature Space Construction

The feature space for each ETF encompassed a set of calculated financial indicators and portfolio-specific metrics:

- **Log Returns**: The natural logarithm of the ratio of consecutive closing prices.
- **Technical Indicators**: Including RSI, SMA, EMA, Stochastic Oscillator (%K), MACD, A/D Line, OBV, ROC, Williams %R, and Disparity Index.
- **Portfolio Ratios**: Daily portfolio ratios for each ETF and the cash balance ratio, providing insights into the relative weight of each asset in the portfolio.
- **Cash Balance**: An absolute figure representing the available cash for trading activities.

These features formed a multidimensional state space, capturing market trends and the current portfolio state, essential for the model's decision-making process.

2.1.3 Data Normalization and Splitting

Normalization was applied to standardize the range and scale of different features, ensuring consistent data interpretation and efficient model training. The dataset was split into two parts: 80% for training the agent, and the remaining 20% for validation to assess performance on unseen data.

2.1.4 State Representation

The state representation in the model was a fusion of the portfolio state and market signals. The portfolio state included the current cash balance and holdings of each ETF, while market signals were derived from the financial indicators. This comprehensive state vector was instrumental in empowering the agent to make informed and strategic trading decisions.

2.2 Model Architecture

The architecture of the model is integral to this project, involving the development of sophisticated neural networks for effective decision-making in stock trading. The model architecture is comprised of two primary components: the Actor Network and the Critic Network, which are fundamental to the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm.

2.2.1 Actor Network

The Actor Network is responsible for deciding the actions to be taken in the trading environment. Its structure is as follows:

- **Input Layer**: Accepts the state dimension as input, which includes both the portfolio state and market signals.
- **Hidden Layers**: Consists of two hidden layers with 400 and 300 units, respectively, using ReLU activation for introducing non-linearity.
- **Output Layer**: The output layer's dimensionality matches the number of tickers being traded. It uses the tanh activation function, producing values in the range of -1 to 1, which are mapped to specific trading actions.

Mapping Actor Network Output to Trading Actions

The output from the Actor Network, ranging from -1 to 1, is mapped to real trading actions as follows:

- Values from 0 to 1 represent the ratio of the available balance to be invested in a specific ticker, capped at 5% to avoid significant investment in a single day.
- Values from -1 to 0 indicate the percentage of existing holdings to sell for each ticker. If the existing holding is 0, it indicates that there is nothing to sell.

2.2.2 Critic Network

The Critic Network evaluates the actions taken by the Actor Network. It consists of two separate but identical networks (Critic 1 and Critic 2), each with:

- **Input Layer**: Combines state and action dimensions to evaluate the action based on the given state.
- **Hidden Layers**: Similar to the Actor Network, each Critic Network has two hidden layers with 400 and 300 units, respectively, using ReLU activation.
- **Output Layer**: Outputs a single value representing the Q-value of the action-state pair, guiding the Actor Network in policy improvement.

2.2.3 TD3 Algorithm Integration

The TD3 algorithm orchestrates the interaction between the Actor and Critic Networks, introducing key improvements over the standard DDPG algorithm:

- **Clipped Double-Q Learning**: Utilizes two Critic Networks to mitigate overestimation bias.

- **Delayed Policy Updates**: Delays policy updates in the Actor Network, allowing more stable value estimates from the Critic Networks.
- **Target Policy Smoothing**: Adds noise to the target policy for enhanced learning stability.

2.2.4 Replay Memory

Replay Memory is crucial in stabilizing the training process by storing experiences in a buffer, enabling learning from a diverse range of past experiences in the dynamic stock trading environment.

This architecture equips the model to effectively balance exploration and exploitation, learning optimal trading strategies in the complex, stochastic world of financial markets.

2.3 TD3 Algorithm Implementation

2.4 Training Process

The training process of the deep reinforcement learning model is a crucial aspect of this project, involving the generation of episodes and the iterative improvement of the trading agent's decision-making capabilities. Below are the details of this process.

2.4.1 Episode Generation

An episode represents a sequence of trading days, each characterized by its unique market state and portfolio configuration. The episode generation process is as follows:

1. **Data Segmentation**: The training dataset is divided into episodes, each consisting of a fixed number of consecutive trading days, determined by the `episode_length`.
2. **State Initialization**: At the beginning of each episode, the initial state is set using the data from the first day, which includes both the portfolio state and market signals.
3. **Episode Progression**: The agent navigates through the episode, making trading decisions based on the current state for each day.

4. **Terminal State**: An episode concludes upon reaching the last day of the sequence or if a terminal condition (like a stop-loss or take-profit trigger) is met.

2.4.2 Model Training

Model training involves the following steps:

1. **Action Selection**: The Actor Network suggests an action for each state during an episode, where the action comprises trading decisions for each ticker.
2. **Experience Gathering**: Following action execution, the model observes the new state and receives a reward based on the outcome of the action. This experience (state, action, new state, reward) is stored in the Replay Memory.
3. **Batch Learning**: The model samples a batch of experiences from the Replay Memory randomly, allowing it to learn from a diverse range of experiences.
4. **Network Updates**:
    - The Critic Networks are updated to better estimate the Q-values based on the sampled experiences.
    - The Actor Network updates its parameters to improve its policy, based on the Critic Networks' Q-value estimates. This update occurs less frequently, adhering to the TD3 algorithm's delayed update strategy.
5. **Target Network Updates**: The parameters of the target networks (both Actor and Critic) are softly updated periodically to ensure stability in learning.

2.4.3 Episode Iteration and Model Evaluation

The training iterates over multiple episodes, with the model continually learning and adjusting its parameters. The model's performance is evaluated on the validation set after a certain number of episodes to assess its ability to generalize to unseen data, guiding any necessary adjustments in training or hyperparameters.

3 Model Performance

3.1 Data Summary

Our validation dataset encompasses a series of trading actions and the corresponding portfolio metrics for a curated selection of tickers: IWM, SPY, QQQ, and TLT. Initial holdings, cash balance, and the closing price for each ticker are meticulously documented alongside the executed actions. These actions are represented as vectors, where each component correlates with the traded shares for the respective ticker. Additionally, the dataset captures the total portfolio value (`total_port`), the ensemble return (`reward_ensemble`), and the date of each transaction.

action	total_port	reward_ensemble	date	cashbalance	IWM_holding	SPY_holding	QQQ_holding	TLT_holding	IWM_closeprice	QQQ_closeprice	SPY_closeprice	TLT_closeprice
[0.0, 0.0, 0.0, -16.0]	1284544.451	0.014574487	20230123	15627.61621	2	1527	1922	542	187.3300018	288.9599915	400.6300049	105.6999969
[0.0, 0.0, 0.0, -10.0]	1283615.803	-0.00072294	20230124	17318.81641	2	1527	1922	526	186.9299927	288.3699951	400.2000122	107.2200012
[0.0, 0.0, 0.0, -5.0]	1283062.036	-0.000431411	20230125	18391.01563	2	1527	1922	516	187.4499969	287.730011	400.3500061	107.4800034
[0.0, 0.0, 0.0, -15.0]	1299831.992	0.013070261	20230126	18928.41602	2	1527	1922	511	188.5599976	293.3399963	404.75	106.9800034
[0.0, 1.0, 0.0, -24.0]	1305946.416	0.004704011	20230127	20533.11523	2	1527	1922	496	189.5800018	296.2600098	405.6799927	106.7099991
[0.0, 0.0, 1.0, -9.0]	1286821.331	-0.014644616	20230130	22797.89453	2	1528	1922	472	186.9400024	290.269989	400.5899963	106.3199997
[0.0, 1.0, 1.0, -4.0]	1305197.266	0.0142801	20230131	23354.18359	2	1528	1923	463	191.4799957	294.6199951	406.480011	107.1699982
[0.0, 1.0, 1.0, -13.0]	1323611.241	0.014108193	20230201	23081.76367	2	1529	1924	459	194.4900055	300.9200134	410.7999878	108.1800003
[0.0, 1.0, 1.0, -40.0]	1351716.842	0.021234031	20230202	23776.38281	2	1530	1925	446	198.3200073	311.7200012	416.7799988	108.3199997
[0.0, 1.0, 1.0, -32.0]	1334042.543	-0.013075446	20230203	27380.68359	2	1531	1926	406	196.9900055	306.1799927	412.3500061	106.6999969
[0.0, 1.0, 1.0, -22.0]	1324917.433	-0.006840194	20230206	30076.55273	2	1532	1927	374	194.1399994	303.5899963	409.8299866	105.9100037
[0.0, 1.0, 1.0, -45.0]	1344597.804	0.014854036	20230207	31693.15234	2	1533	1928	352	195.5800018	309.8800049	415.1900024	105.0599976
[0.0, 1.0, 1.0, -15.0]	1327535.551	-0.012689483	20230208	35695.78125	2	1534	1929	307	192.7200012	304.3699951	410.6499939	105.5599976
[0.0, 1.0, 1.0, -14.0]	1316238.021	-0.008510152	20230209	36564.16016	2	1535	1930	292	189.9299927	301.6799927	407.0899963	104.5599976
[0.0, 1.0, 1.0, 0.0]	1314706.746	-0.001163373	20230210	37319.23047	2	1536	1931	278	190.3099976	299.7000122	408.0400085	103.3899994
[0.0, 1.0, 1.0, 0.0]	1331585.007	0.012838041	20230213	36611.49219	2	1537	1932	278	192.6000061	304.5	412.8299866	104.2600021
[0.0, 1.0, 1.0, -5.0]	1334611.391	0.002272767	20230214	35894.16406	2	1538	1933	278	192.5099945	306.75	412.6400146	104.0199966
[0.0, 1.0, 1.0, -10.0]	1340558.696	0.004456207	20230215	35694.875	2	1539	1934	273	194.4600067	309.1000061	413.980011	103.0500031
[0.0, 1.0, 1.0, 0.0]	1320209.445	-0.015179681	20230216	36002.29688	2	1540	1935	263	192.6000061	303.2999878	408.2799988	101.5899963
[0.0, 1.0, 1.0, 2.0]	1315145.863	-0.00383544	20230217	35290.71875	2	1541	1936	263	193.1300049	301.1600037	407.2600098	102.3799973
[0.0, 1.0, 2.0, 3.0]	1287787.33	-0.020802661	20230221	34377.53906	2	1542	1937	265	187.4299927	294.0299988	399.0899963	100.3899994
[0.0, 1.0, 2.0, 4.0]	1287307.946	-0.000372255	20230222	32984.16016	2	1543	1939	268	187.9400024	294.25	398.5400085	101.3099976
[0.0, 0.0, 1.0, 4.0]	1295662.909	0.00649026	20230223	31487.58984	2	1544	1941	272	189.2799988	296.8200073	400.6600037	102.3000031

Figure 1: Sample of result

3.2 Portfolio Performance Analysis

The trading model's efficacy was appraised through three distinct strategies: model-based actions, random actions, and a buy-and-hold strategy. The model-based actions were determined by the neural network's learned policy, while random actions were chosen indiscriminately within the trading constraints, and the buy-and-hold strategy embodied a static investment approach post-initial asset allocation.

A comparative analysis of these strategies was conducted by plotting their aggregate returns over time. The emergent trends are as follows:

- **Model-Based Actions**: Exhibiting a volatile yet generally upward trajectory in total portfolio return, suggesting the model's adeptness at leveraging market movements.

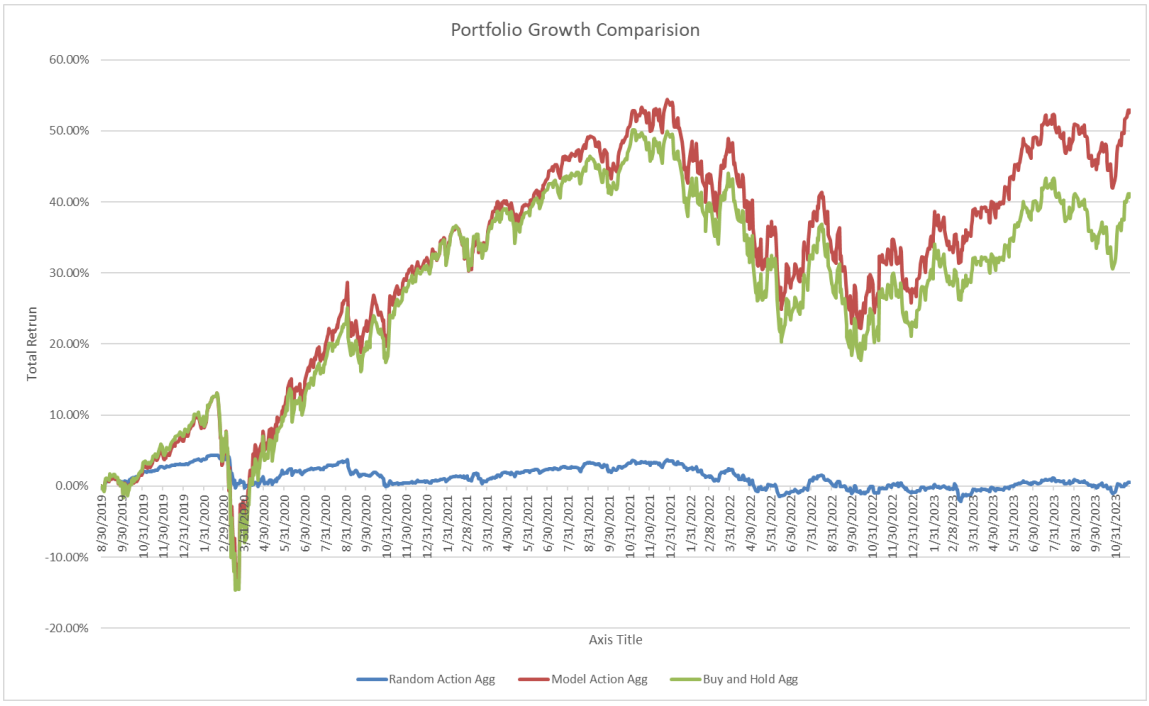

Figure 2: Sample of result

- **Random Actions**: The performance is relatively stagnant, showcasing minimal growth and highlighting the deficiency of random actions in intricate market scenarios.
- **Buy-and-Hold Strategy**: Reflects a steady increase in total return, aligning with typical market growth expectations over extended periods.

The plotted returns underscore the instances where the model-based strategy surpassed the passive buy-and-hold strategy, especially during exploitable market trends. Conversely, the underperformance of the random action strategy underscores the significance of informed and strategic decision-making in portfolio management.

3.3 Enhanced Observations

The initial phase of the validation period was characterized by a consistent market uptrend, with all strategies starting from a zero cash balance. This restriction led to limited selling by the model, resulting in constrained capital for purchases.

Nonetheless, the model's portfolio demonstrated a modest improvement compared to the buy-and-hold strategy, even when the overall market trend was bullish.

In the latter half of the validation set, marked by increased market volatility, the model's ability to execute trades and accumulate cash reserves facilitated the identification of optimal investment opportunities. This period showcased a clear differentiation in performance from the buy-and-hold strategy, emphasizing the model's capability to adapt to and exploit fluctuating market conditions.

3.4 Concluding Insights

The validation phase elucidates the model's potential in active portfolio management, adapting its strategy to both bullish trends and volatile movements. The insights gained indicate a promising avenue for enhancing model performance by integrating a more extensive set of economic and policy data. Such advancements could potentially yield superior strategic decisions and a more comprehensive understanding of financial market dynamics.

3.5 Future Enhancement Potential

The current model performance is derived from a pure simulation environment. Incorporating a broader spectrum of data—encompassing macroeconomic indicators, policy changes, and federal data—could significantly augment the model's predictive accuracy and overall financial performance. This expansion of the dataset is anticipated to facilitate better informed and potentially more profitable trading decisions.

4 Challenges and Limitations with Future Work

4.1 Data Quality and Availability

The model's performance was partly constrained by the limited scope of financial data. High-frequency trading data, more nuanced economic indicators, and comprehensive sentiment analysis could significantly refine the model's predictive capabilities. Expanding the dataset to include these factors is a critical area for future development.

4.2 Model Overfitting

Overfitting remains a perennial concern, with the model exhibiting a strong fit to the training data but potentially lacking generalization to new data. Incorporating regularization techniques, cross-validation, and live market testing could address this limitation in future iterations of the project.

4.3 Market Volatility and Black Swan Events

Financial markets are inherently volatile and subject to unpredictable 'Black Swan' events. Enhancing the model to anticipate and adapt to such conditions is vital. Future models could incorporate scenario analysis and stress testing to better withstand market shocks.

4.4 Simplified Assumptions

The model's foundational assumptions, such as zero slippage and no market impact, are not reflective of real trading environments. Future enhancements should aim to model these aspects more accurately, possibly by integrating microstructural elements of financial markets into the simulation framework.

4.5 Systematic Risk Management

Currently, the model lacks a robust risk management framework, focusing primarily on return maximization. Future work should integrate risk-adjusted return metrics and strategies to balance the trade-off between risk and reward, ensuring a comprehensive and sustainable trading strategy.

4.6 Future Enhancement Potential

To address these challenges, future work will focus on improving computational efficiency, enriching the dataset, refining the model to guard against overfitting, and incorporating advanced risk management strategies. The goal is to develop a more robust, accurate, and market-responsive trading agent that can navigate the complexities of the financial markets with enhanced adaptability and foresight.