Master's thesis proposal

Krzysztof Wojdalski

Table of contents

Idea

Financial markets have shown interest in computer science methods since the 1970s. While traditional investing approaches like buy-and-hold can yield abnormal returns, more sophisticated quantitative methods are constantly on the rise. Artificial intelligence-based trading has emerged as a key category, driven by the belief that algorithms exceed human decision-making capabilities.

Often, strategies in literature focus on maximizing trading profits or risk-adjusted measures (e.g., Sharpe ratio). Researchers have sought profitable systems, drawing inspiration, for instance, from fundamental analysis, econometric modeling, and machine learning. Though some attempts succeeded, many promising ones often prove to be impractical for real markets due to significant drawbacks. These include (but are not limited to):

- Large drawdowns
- Excessive trading
- Legal/economic constraints

Market participants often view automated systems as risky compared to more traditional approaches. Even when models show good risk-return profiles in backtests, there is no guarantee of continued performance (e.g., due to overfitting). Systems fail precisely when they should adapt to changing market conditions.

The work I am going to present aims to deal with the above problems to obtain a usable, automated and "intelligent" trading system. The thesis proposes the implementation of reinforcement learning agents for ~~FX~~/equities/~~cryptocurrency~~ trading.

I am contemplating which base strategy to employ. One interesting approach could be a pairs trading strategy that utilizes signals derived from relationships between order books of related securities—for instance, between two ETFs tracking the same underlying asset.

Reinforcement learning demonstrates efficacy in domains where deterministic methodologies exhibit limitations due to structural incompatibility, implementation complexity, or computational constraints. For a detailed implementation example of a reinforcement learning agent in a trading environment, see Appendix 1.

The research will utilize multiple RL approaches including TD learning (e.g., Q-learning/DQN), Monte Carlo Control, and Policy Gradient (e.g., PPO) methods. Moreover, the implementation will incorporate algorithmic enhancements to address common challenges in RL, such as exploitation versus exploration trade-off or approximation of value function (in the context of selected algorithms).

Detailed Thesis Structure (proposal)

1. Introduction

The introductory chapter establishes the intersection between artificial intelligence and financial markets. It examines the historical evolution of computational methods in finance, tracing their development from early quantitative models to contemporary AI applications. The chapter explores the relationship between financial theory and computer science innovations, highlighting how advances in one domain have catalyzed progress in the other.

- 1.1 Foundations of Statistical Arbitrage

  This section will provide a comprehensive definition of statistical arbitrage as a trading strategy that exploits temporary price discrepancies between related securities through mathematical models. It will explain how statistical arbitrage relies on identifying mean-reverting patterns and price inefficiencies that can be systematically exploited.

  - 1.1.1 Statistical Arbitrage vs Other Types of Arbitrage (e.g., latency arbitrage)
  - 1.1.2 Critical Parameter Selection and Optimization
  - 1.1.3 Applying Reinforcement Learning to Overcome Traditional Limitations

    This section will establish why reinforcement learning presents a great framework for statistical arbitrage implementation, highlighting how RL's ability to learn optimal decision policies through trial-and-error interactions with markets makes it well-suited for adapting to changing market conditions and complex statistical relationships without requiring explicit programming of trading rules.

- 1.2 Scope and Objectives of the Research

  - 1.2.1 Hypotheses
    - AI-driven algorithms can deliver superior value by exceeding benchmark performance metrics in terms of both risk management and return generation
    - Enhanced performance is particularly evident in high-frequency trading scenarios as well as across extended time horizons
    - These algorithms demonstrate the capability to implicitly learn market patterns that would be impossible for humans to identify and efficiently select securities. In consequence, they generate alpha
  - 1.2.2 Objectives of the Research
    - Design and implementation of a basic and an extended Reinforcement Learning trading agent for pair trading
    - Test and compare both approaches on out-of-sample test data and selected performance measures

        * Interpretation and implications - Analyze how the agents evolve their learning processes and how their developed policies integrate conventional pair trading approaches and transaction cost management. Identify and suggest potential avenues for future research

- 1.3 Methodological Framework

  The methodology consists of several key steps:

  - Data pre-processing - handling data obtaining, cleaning, completion[1], and transformation
  - Feature engineering - building informative features from time series, including statistical, technical, and economic variables across different time horizons
  - State-action space construction - creating the agent's state space
  - Trading agents implementation - developing agents that maximize financial performance metrics via selected RL algorithms
  - Performance evaluation - testing both trading agents with transaction costs included, comparing approaches, and analyzing the resulting trading policies

- 1.4 Thesis Organization and Chapter Overview

  As below.

2. Classical Finance Context and Literature Review

This chapter provides a comprehensive review of relevant literature in quantitative finance and machine learning, establishing the theoretical foundation for the research. It begins with an examination of classical financial models such as CAPM and progresses to critical analysis of them. The last subchapter discusses works that correspond thematically to machine learning in trading systems and the current work.

This one might change as it's not a good idea to dump a bunch of classic papers, especially if they are not that important in the context of the research

- 2.1 Quantitative Finance Models
  - 2.1.1 Classical Models (CAPM, APT, Factor Models, e.g., Fama-French)
  - 2.1.2 Modern Approaches in Algorithmic Trading
  - 2.1.3 Critical Analysis of Traditional Financial Models
- 2.2 Criticism of the EMH
  - 2.2.1 Efficient Market Hypothesis (EMH)
  - 2.2.2 Empirical Evidence Against Perfect Market Efficiency

[1] Data completion is a process of filling in missing values in the data. It is a crucial step in the data preprocessing pipeline as it can significantly impact the performance of the trading agent.

  - 2.2.3 Market Anomalies and Exploitable Patterns
    * For instance, DeBondt and Thaler's (1985) and Jegadeesh and Titman's (1993) findings
- 2.3 Machine Learning in Financial Markets
  - 2.3.1 Supervised Learning Applications
  - 2.3.2 Unsupervised Learning Approaches
  - 2.3.3 Reinforcement Learning Foundations
    * 2.3.3.1 Key Concepts from Reinforcement Learning
    * 2.3.3.2 Seminal Works in RL (Sutton and Barto's book)
    * 2.3.3.3 RL Applications in Trading Systems

3. Algorithmic Trading Systems

This chapter serves as a foundational introduction to subsequent sections of the research. It will examine critical components essential for effective HFT systems, including data processing mechanisms, normalization techniques, co-located server infrastructure, optimization and execution layer.

Given the theoretical nature of this section, it will also address the structure of equity markets (participant categorization, on-exchange versus off-exchange), market microstructure, participant categorization, and emerging developments such as "private rooms." These elements constitute theoretical context appropriate for a master's thesis rather than a concise working paper.

I don't consider this chapter as absolutely essential and it could be written once the research part is done.

- 3.1 Algorithmic Trading Systems
  - 3.2.1 Components of an Algorithmic Trading System
    * 3.2.1.1 Data Handlers
    * 3.2.1.2 Reference Data
    * 3.2.1.3 Optimization
    * 3.2.1.4 Execution
    * 3.2.1.5 UI / UX - only mentioned with little details

  This one could be changed as Rule-Based Trading is possibly small enough to merge it somehow with the next subsection

  - 3.2.2 Rule-Based Trading
  - 3.2.3 Model-Based Trading
    * 3.2.3.1 Trading based on Forecasts
    * 3.2.3.2 Training a Trading System on Labelled Data
    * 3.2.3.3 Direct Optimization of Performance (RL)

4. Reinforcement Learning

This part provides a comprehensive overview of reinforcement learning theory and methodology, establishing the foundation for the trading agent implementation. It explores the fundamental components (4.1), timeline of new RL algorithms (4.2), taxonomies (4.3), and optimization approaches that will be applied to financial market trading. In this chapter, I would like to clearly state distinction between RL algorithms so in the further parts of the thesis the reader can understand the design choices and why they were made.

- 4.1 Components of a Reinforcement Learning Agent
- 4.2 Timeline of New RL Algorithms In this part, I would like to outline the timeline of new RL algorithms, to show where the novelty comes from

  This section could be restructured; however, I intended to provide concise historical context regarding algorithmic developments. For example, the method I want to implement PPO was published in 2017 so it's fairly new.

  - 4.2.1 1980s - 1990s
  - 4.2.2 2000s
  - 4.2.3 2010s
  - 4.2.4 2020s

- 4.3 Reinforcement Learning - Categories

  - 4.3.1 Model-Free vs Model-Based
  - 4.3.2 Value-Based vs Policy-Based vs Actor-Critic
  - 4.3.3 On-Policy vs Off-Policy
  - 4.3.4 Single-Agent vs Multi-Agent - **This one is not relevant for financial market applications so might be dropped**
  - 4.3.5 Discrete vs Continuous

- 4.4 Solving Sequential Decision Making Problems
- 4.5 Policy Optimization

5. Design of the Trading Agent

- 5.1 Action Space

  A vector of actions (either discrete from the set $\{-1, 0, 1\}$ or continuous[2])

[2]Number of options here is broader. Yet-to-be-determined.

- 5.2 State Space / Environment

  Environment - consists of raw market data and additional features relevant in the context of the research. Enhancements to the environment will be made with various features identified through literature review, such as technical indicators, order book-related variables, and autoregressive features, which form the foundation for the algorithm's decision-making process.

  States - a vector of environment in time t

  - 5.2.1 Current Position (Recurrence)
  - 5.2.2 Order Book Related Variables
    - 5.2.2.1 Book Pressure-Related Variables
    - 5.2.2.2 Volume-Related Variables
    - 5.2.2.3 Last Trade-Related Variables
    - 5.2.2.4 Tick-Related Variables
  - 5.2.3 Technical Indicators
  - 5.2.4 Autoregressive Features

- 5.3 Reward Function

  - 5.3.1 Differential Sharpe Ratio
    - Rewards - in the form of Differential Sharpe Ratio (see Appendix 2)
  - 5.3.2 Other Reward Functions
    - Rewards - in the form of absolute return, %return, etc.

- 5.4 Value Function

  - 5.4.1 Value Function Approximation

- 5.5 Policy

  - A set of rules for agents to take actions in the environment. It's a function that maps states to actions (output of the algorithm)
    - 5.5.1 Argmax Policy
    - 5.5.2 Proximal Policy Optimization

- 5.6 Exploration Rate
- 5.7 Step Size
- 5.8 Summary

6. Implementation of the Trading Agent

To ensure realistic market simulation, the methodology will incorporate transaction costs, financing costs (if applicable), and slippage effects.

- 6.1 Data Preparation
  - 6.1.1 Data Collection

    For this research, I aim to maximize practical applicability by utilizing (tick-by-tick) L3 order book data. Currently, I'm evaluating several data source options:

    ~~* Leveraging API handlers for cryptocurrency exchanges (with Binance being a particularly suitable candidate)~~
    - Utilizing Databento for historical equity market data[3] (see Appendix 3 for the raw data structure)

    ~~* Exploring foreign exchange (FX) market data (though this presents challenges due to the predominance of OTC trading)~~[4]

    ~~* Implementing data handlers for specific market segments (the least likely scenario)~~
  - 6.1.2 Data Preprocessing
  - 6.1.3 Feature Engineering
- 6.2 Code
  - 6.2.1 Code Structure
  - 6.2.2 Code Implementation

7. Empirical Evaluation and Performance Analysis

Benchmarks:

- Random actions - this part of the algorithm will generate random values in a domain of $\{-1, 0, 1\}$. These values will serve as a position in the underlying pairs. The benchmark will not include any transactional costs as it is obvious that this extreme case would have an enormous cumulated transactional cost (position would change in $\frac{2}{3}$ of states).
- Buy-and-hold strategy which means holding long-position in selected currency pairs.
- Risk-free rate, or just zero - depending on the time horizon which is yet to be decided.

  Not much to say here as of now, but I think some standardized measures/methods should be used.

[3] This looks promising and pretty easy to obtain. I've managed to construct order books out of MBO data.

[4] The foreign exchange market is a global decentralized market for the trading of currencies. There's no central exchange (with some exceptions like CME Group for FX futures or MOEX).

- 7.1 Statistical Validation (e.g., through Bootstrapping)
- 7.2 Robustness Assessment (e.g., through Out-of-sample Testing)
- 7.3 Comprehensive Performance Evaluation and Findings

8. Conclusions and Future Work

- 8.1 Summary of Findings
- 8.2 Limitations and Future Research
  - What could be additionally implemented?
  - What were limitations and what must be done to overcome them in future works?
- 8.3 Implications for Trading Systems
  - What are the implications of the research for the future of the trading systems?
- 8.4 Recommendations for Practitioners
  - There are a lot of issues that practitioners would need to consider that I'm unlikely to cover in the thesis. For instance, computation time - it's very likely that persistence of certain trades will be so short that only highly optimized and deterministic algorithms will be able to take advantage of them (e.g., with use of FPGAs).
- 8.5 Conclusion

(Possible) Obstacles in the Research

- Find the right framework for the research. As the problem is not trivial, I would like to avoid coding everything from scratch.
  - This part has been researched. The working solution would be based on the following libraries/frameworks:
    * torch - a popular deep learning framework
    * torchrl - pytorch-based framework for reinforcement learning
    * Gym - a toolkit for developing and comparing reinforcement learning algorithms
    * Gym Trading Env - a framework specifically designed for trading environments
- Computational resources:
  - I have access to a single machine with 8 cores and 18GB of RAM
  - Depending on the scope of the research, this might be insufficient
- Scope of the research which is demanding in various aspects:

- Novelty of the approach (algorithms are relatively new, not trivial, and it takes effort to understand them)
- Agent design
- Feature engineering - preparing meaningful features, especially working ones, is a time-consuming process due to several issues, e.g., alpha decay.

Appendices

Appendix 1 - Tic Tac Toe

Consider the canonical Tic Tac Toe game as an example. A conventional approach to policy optimization for maximizing win probability would require defining a deterministic policy across 3^9 possible states (acknowledging some configurations are infeasible). The factor 3 represents the three possible cell states (player X, player O, or empty), while 9 corresponds to the standard 3×3 matrix. This deterministic approach presents challenges in implementation robustness and generalizability. Reinforcement learning offers an elegant alternative, circumventing exhaustive conditional logic while achieving comparable performance. The methodology leverages dynamic programming principles to derive a policy that optimizes the value function, effectively maximizing the probability of favorable outcomes.

The superiority hypothesis of reinforcement learning rests on the premise that a properly calibrated agent can identify variable relationships that may elude human perception. I contend that, in some contexts, predefined rule-based systems are significantly less effective, or even incapable, of providing protection against adverse outcomes under specific market conditions compared to adaptive learning approaches.

In conclusion, my master's thesis will evaluate the potential advantages and practical benefits that reinforcement learning methods could provide to organizations and entities that implement them in trading contexts.

As a demonstration of reinforcement learning principles, I have developed an implementation for the Tic Tac Toe game using R programming language:

- Krzysztof Wojdalski - GitHub / Tic Tac Toe

Appendix 2 - Formula for the Differential Sharpe Ratio

- The differential Sharpe ratio is a dynamic extension of the traditional Sharpe ratio that captures the marginal impact of returns at time t on the overall Sharpe Ratio. The computation begins with two recursive formulas:

$$A_n = \frac{1}{n} R_n + \frac{n-1}{n} A_{n-1}$$

$$B_n = \frac{1}{n} R_n^2 + \frac{n-1}{n} B_{n-1}$$

At initialization ($t = 0$), both values are set to 0. These formulas serve as the foundation for calculating the exponentially moving Sharpe ratio on an η time scale:

$$S_t = \frac{A_t}{K_\eta \sqrt{B_t - A_t^2}}$$

where:

- $A_t = \eta R_t + (1 - \eta) A_{t-1}$
- $B_t = \eta R_t^2 + (1 - \eta) B_{t-1}$
- $K_\eta = \frac{1 - \frac{\eta}{2}}{1 - \eta}$

The differential Sharpe ratio offers several advantages for algorithmic trading systems:

- Recursive updating: There's no need to recalculate the mean and standard deviation of returns with each evaluation. The formulas for A_t and B_t enable straightforward calculation of the exponential moving Sharpe ratio by simply incorporating the latest returns R_t and R_t^2.
- Computational efficiency: The formula structure facilitates rapid calculation through simple updates of recent values, making it ideal for online optimization.
- Clear interpretation: The differential Sharpe ratio provides an intuitive measure of how recent returns affect the risk-reward profile captured by the Sharpe ratio.

Appendix 3 - Data Structure

Field	Description
ts_recv	The capture-server-received timestamp expressed as the number of nanoseconds since the UNIX epoch.
size	The order quantity.
ts_event	The matching-engine-received timestamp expressed as the number of nanoseconds since the UNIX epoch.
channel_id	The channel ID assigned by Databento as an incrementing integer starting at zero.
rtype	The record type. Each schema corresponds with a single rtype value.

Field	Description
order_id	The order ID assigned at the venue.
publisher_id	The publisher ID assigned by Databento, which denotes dataset and venue.
flags	A bit field indicating event end, message characteristics, and data quality.
instrument_id	The numeric instrument ID.
ts_in_delta	The matching-engine-sending timestamp expressed as the number of nanoseconds before ts_recv.
action	The event action. Can be Add, Cancel, Modify, Clear book, Trade, Fill, or None.
sequence	The message sequence number assigned at the venue.
side	The side that initiates the event. Can be Ask for a sell order (or sell aggressor in a trade), Bid for a buy order (or buy aggressor in a trade), or None where no side is specified by the original source.
symbol	The requested symbol for the instrument.
price	The order price expressed as a signed integer where every 1 unit corresponds to 1e-9, i.e. 1/1,000,000,000 or 0.000000001.