Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Project Plan

Learning Trading Rules with Reinforcement Learning

by

Antony Krymski

Email: antony.krymski@gmail.com
GitHub username: agk-123
Repository: https://github.com/ese-msc-2023/irp-agk123

Supervisors:

Prof. Paul Bilokon
Dr. Thomas Davison

June 2024

Abstract

The analysis of financial time series comes with unique challenges due to the stochastic, non-stationary, and often unpredictable nature of the data. This project aims to explore the application of reinforcement learning algorithms to financial time series, specifically focusing on Contrastive Difference Predictive Coding as presented by Salakhutdinov et al. Our objective is to evaluate the effectiveness of Contrastive Difference Predictive Coding in comparison with traditional models like reinforcement learning with Long Short-Term Memory networks and Deep Q-Networks. The study aims to determine the relative performance and potential advantages of Contrastive Difference Predictive Coding in developing trading strategies for non-stationary financial time series.

1 Introduction

Analysing financial time series is an integral component of stock trading, risk management and econometrics. Auto-regressive integrated moving averages and linear regression are traditionally used and serve as the foundation for many modern forecasting techniques [1]. However, these methods often falter when confronted with the highly volatile and non-stationary nature of financial datasets, where the assumptions of stationarity and linearity do not hold [1].

Advancements in machine learning have allowed for more complex temporal patterns to be captured for both analysis and forecasting of time series [2]. Reinforcement Learning (RL), allows for the learning of optimal strategies through interaction with the environment, thereby potentially improving forecasting/analysis accuracy. Notably, RL with Long Short-Term Memory (LSTM) networks and Deep Q-Networks (DQN) have been shown to be able to forecast time series accurately to a degree [3][4].

Existing methods, including RL-LSTM and DQN, often exhibit significant limitations when applied to real-world stochastic environments. These models are prone to issues such as extensive computational requirements, susceptibility to overfitting, and a reliance on substantial amounts of labeled data [5]. Salakhutdinov et al. attempt to address some of these limitations through Contrastive Difference Predictive Coding (CDPC), which introduces a temporal difference version of contrastive predictive coding (CPC) that decreases the amount of data required for the model to learn [6][5]. However, a primary limitation remains: these models have been predominantly utilised within deterministic settings.

The purpose of this project is to compare existing methods, such as RL-LSTM and DQN, with the CDPC approach within a stochastic environment, specifically focusing on financial time series data. By conducting this comparative analysis, we aim to evaluate the relative performance and potential advantages of CDPC in handling the inherent unpredictability and variability of stochastic financial time series. This will allow for CDPC's applicability beyond the controlled, deterministic environment it was evaluated on previously by Salakhutdinov et al.

2 Literature Overview

The study "Deep Reinforcement Learning for Trading" investigates the application of LSTM networks within an RL framework for stock trading [4]. The RL agent in this study uses the context vector created by the LSTM to predict stock price movements and make trading decisions. The primary advantage of LSTM lies in its ability to handle the vanishing gradient problem, ensuring that long-term dependencies are preserved [7]. However, the model's extensive computational requirements and susceptibility to overfitting highlight the need for further optimisation and

refinement. Despite these challenges, the study concludes that LSTM-based RL models have significant potential for enhancing trading strategies, contingent on further improvements.

"Representation Learning with Contrastive Predictive Coding", introduces CPC as an innovative approach for unsupervised learning [6]. CPC leverages a self-supervised learning framework that predicts future observations in a latent space while contrasting correct futures against negative samples. This approach enables the extraction of meaningful patterns from large datasets without the need for extensive labelled data. CPC's strength lies in its ability to learn compact and informative representations, making it particularly useful for applications where labelled data is scarce. However, CPC was experimented on a deterministic environment and has yet to be applied to financial time series that exhibit stochastic behaviour.

Building on the CPC framework, "Contrastive Difference Predictive Coding" by Salakhutdinov et al. addresses the limitations of the original CPC model [5]. This study enhances CPC's robustness by modifying the loss function and incorporating additional context from both past and future data points. These improvements enable CPC to better handle variability and noise, making it more suitable for real-world stochastic scenarios. While the enhanced CPC model shows improved performance in controlled and deterministic environments, the study highlights the need to evaluate these advancements in genuinely stochastic settings.

The lack of evaluating CPC and CDPC in a stochastic environment is what forms the gap that this research is based on, where CDPC will be tested in forecasting financial non-stationary time series.

3 Methods

3.1 Overview of RL

"RL is a machine learning paradigm where an agent learns to make decisions by interacting with an environment" [8]. At each time step t, the agent observes the current state s_t, selects an action a_t based on a policy $\pi(a_t|s_t)$, and receives a reward r_t from the environment. The goal is to maximize the cumulative reward $R = \sum_{t=0}^{T} \gamma^t r_t$, where $\gamma \in [0, 1]$ is the discount factor and T is the time horizon [9]. The value function $V^\pi(s)$ represents the expected return starting from state s and following policy π:

$$V^\pi(s) = E_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s \right]. \tag{1}$$

The Q-function $Q^\pi(s, a)$ represents the expected return starting from state s, taking action a, and following policy π:

$$Q^\pi(s, a) = E_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s, a_0 = a \right]. \tag{2}$$

3.2 State Representation and Action Space

For all models, the state s_t includes features derived from historical stock prices and relevant financial indicators, such as moving averages, volatility measures, and momentum indicators. Additional features may be added based on the exploration and analysis of the dataset. The action space A consists of discrete actions: *buy: 1*, *sell: -1*, and *hold: 0*.

3.3 RL-LSTM

RL-LSTM combines reinforcement learning with LSTM networks to handle temporal dependencies in financial time series [7]. The architecture is designed to predict the next action in a sequence of stock trading decisions.

3.3.1 Policy Learning

The agent uses reinforcement learning algorithms to update its policy. The LSTM network helps maintain sequence information, ensuring that the policy incorporates long-term dependencies in the financial data [7]. This is done through a context vector that combines all features extracted by the LSTM of the data up to a timestamp. The Q-learning update rule is used to learn the optimal policy:

$$Q(s_t, a_t) = r_t + \gamma \max_{a'} Q(s_{t+1}, a'). \tag{3}$$

3.4 DQN

DQN apply Q-learning with deep neural networks to handle high-dimensional state spaces [10]. The same Q-learning update rule is used to learn the optimal policy. The key difference is the neural network used to approximate this Q-function, will not be an LSTM but in fact a simple Multi-Layer Perceptron.

3.5 CDPC

CDPC extends the CPC framework by incorporating temporal difference learning. This approach is designed to handle stochastic environments more effectively, reducing the data requirements and enhancing sample efficiency [5].

3.5.1 Predictive Coding

The model predicts future latent representations z_{t+k} based on the current state using an autoregressive model g_{ar}. This ensures that the latent representations are informative and capture relevant temporal dependencies.

$$c_t = g_{ar}(z_{\leq t}) \tag{4}$$

This autoregressive model constructs a context c_t that summarizes all the latent variables up to time t, which is used to predict the future latent variables z_{t+k}. This prediction helps in capturing the temporal structure of the data effectively [6].

3.5.2 Contrastive Learning

CDPC employs a temporal difference version of the InfoNCE loss, known as TD InfoNCE, to maximize the mutual information between the predicted and actual future latent representations[5]. This approach enhances sample efficiency and enables the model to stitch together pieces of different time series data.

$$\mathcal{L}_{\text{TDInfoNCE}}(f) E_{(s,a)\sim p(s,a), s_{t+}^{(2:N)}\sim p(s_{t+})} \left[(1-\gamma) E_{s_{t+}^{(1)}\sim p(s'|s,a)} \left[\log \frac{e^{f(s,a,s_{t+}^{(1)})}}{\sum_{i=1}^{N} e^{f(s,a,s_{t+}^{(i)})}} \right] \right.$$

$$+ \gamma E_{s' \sim p(s'|s,a), a' \sim \pi(a'|s')} \left[\left\lfloor w(s', a', s_{t+}^{(1:N)}) \right\rfloor_{\text{sg}} \log \frac{e^{f(s,a,s_{t+}^{(1)})}}{\sum_{i=1}^{N} e^{f(s,a,s_{t+}^{(i)})}} \right] \quad (5)$$

The policy is informed by the latent representations, enabling the agent to make more informed trading decisions.

3.6 Experiment Pipeline

To apply these models to our task, we follow these steps:

1. **Data Collection**: Gather historical stock prices and relevant financial indicators for a specific stock (set of stocks). This data will serve as the input for training and testing the models.
2. **Feature Engineering**: Extract meaningful features from the raw data, such as moving averages, volatility measures, and momentum indicators. This will be done with the neural networks defined earlier along with some manual feature engineering. These features will be used to represent the state in each model.
3. **Model Training**: Train the RL-LSTM, DQN, and CDPC models on the historical data. Each model will learn a policy that maps the state to actions (buy, sell, hold) to minimise their respective losses.
4. **Validation and Testing**: Validate the models using a separate validation dataset to tune hyperparameters and prevent overfitting. Then backtest the models on out-of-sample data accounting for possible transaction costs to evaluate their performance in "real-world" trading.
5. **Performance Comparison**: Compare the performance of RL-LSTM, DQN, and CDPC in terms of the following metrics: profitability, risk-adjusted returns, and Sharpe ratio.

4 Deliverables

The outcomes of this project will be the following:

- A Jupyter notebook performing data analysis, cleaning and preprocessing.
- A Jupyter notebook with the RL-LSTM model architecture and the model trained on historical stock prices.
- A Jupyter notebook with the DQN model architecture and the model trained on historical stock prices.
- A Jupyter notebook with the CDPC model architecture and the model trained on historical prices.
- A Jupyter notebook comparing the performance of these models.
- A final report culminating all findings and comparing the performance of all ML models on described task.

The main limitation of this project is the possibility of the RL algorithms not working in a stochastic environment due to the inherent interaction the agent has with the environment/state.

5 Future Plan

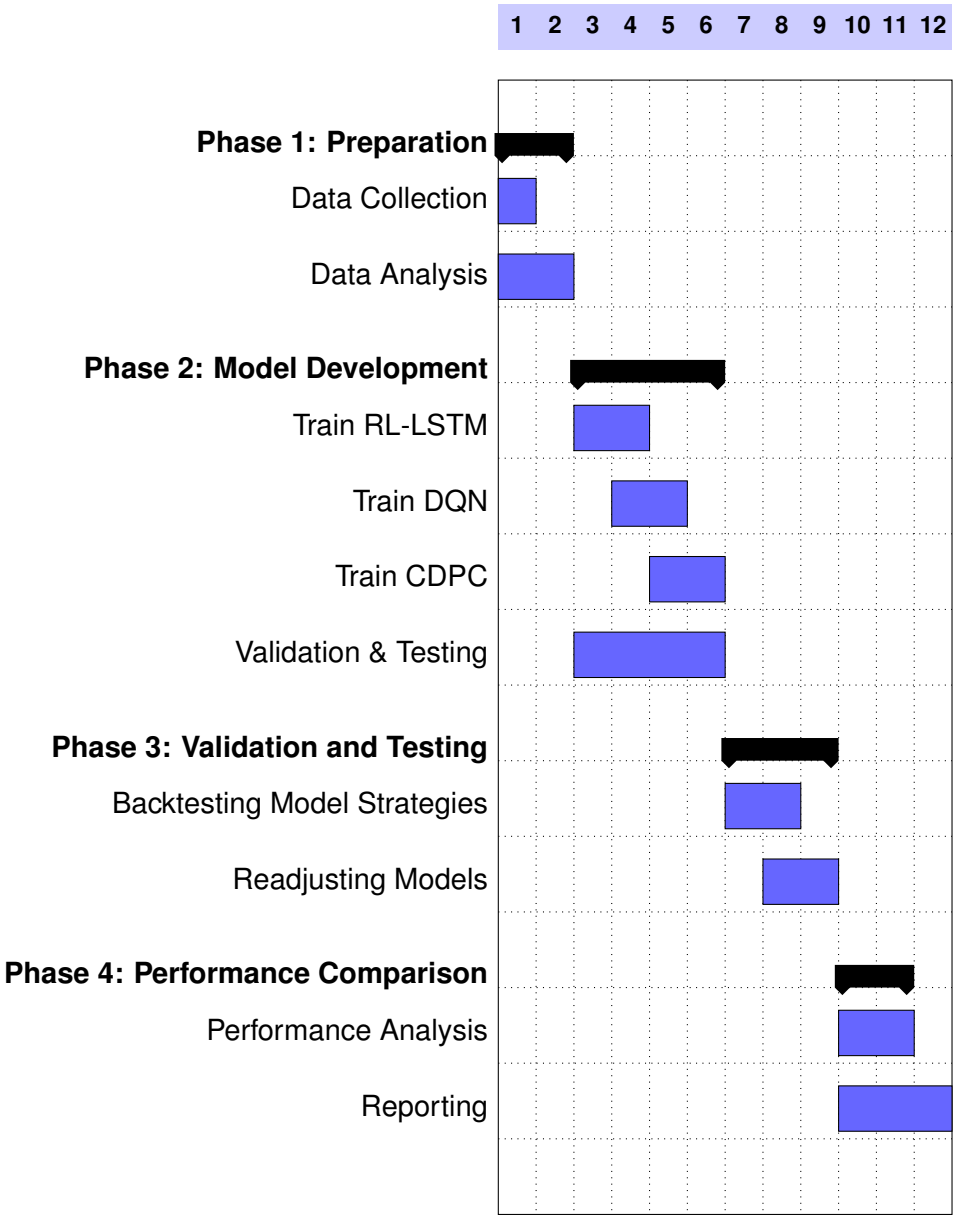

Figure 1: Gantt chart showing the future project phases.

In Figure 1, one can see a Gantt chart depicting the overall schedule for the project.

In summary, the first two weeks are focused on preparing the data, and the following four weeks are for developing the reinforcement learning models. The last six weeks focus on validating, testing, analysing and reporting findings.

Currently, data preparation has been partially completed and model architectures have been researched.

References

[1] Ping-Feng Pai and Chih-Sheng Lin. A hybrid arima and support vector machines model in stock price forecasting, 2005.

[2] Angelo Casolaro, Vincenzo Capone, Gennaro Iannuzzo, and Francesco Camastra. Deep learning for time series forecasting: Advances and open problems. *Information*, 14(11), 2023.

[3] Bram Bakker. Reinforcement learning with long short-term memory. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, page 1475–1482, Cambridge, MA, USA, 2001. MIT Press.

[4] Xiang Gao. Deep reinforcement learning for time series: playing idealized trading games, 2018.

[5] Chongyi Zheng, Ruslan Salakhutdinov, and Benjamin Eysenbach. Contrastive difference predictive coding, 2024.

[6] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] Neta Glazer, Aviv Navon, Aviv Shamsian, and Ethan Fetaya. Multi task inverse reinforcement learning for common sense reward, 2024.

[9] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.