DEEP REINFORCEMENT LEARNING FOR PORTFOLIO MANAGEMENT

Ahmad Shayaan

Integrated Master of Technology Thesis

June 2019

International Institute of Information Technology, Bangalore

DEEP REINFORCEMENT LEARNING FOR PORTFOLIO MANAGEMENT

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Integrated Master of Technology

by

Ahmad Shayaan
IMT2014004

International Institute of Information Technology, Bangalore
June 2019

Dedicated to

ABC and XYZ

Thesis Certificate

This is to certify that the thesis titled **Deep Reinforcement Learning for Portfolio Management** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Integrated Master of Technology** is a bona fide record of the research work done by **Ahmad Shayaan**, **IMT2014004**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. G Srinivasaraghavan

Bangalore,

The 1st of June, 2019.

DEEP REINFORCEMENT LEARNING FOR PORTFOLIO MANAGEMENT

Abstract

This document provides information on how to write your thesis using the LaTeX document preparation system. You can use these files as a template for your own thesis, just replace the content, as necessary. You should put your real abstract here, of course.

"The purpose of the abstract, which should not exceed 150 words for a Masters' thesis or 350 words for a Doctoral thesis, is to provide sufficient information to allow potential readers to decide on relevance of the thesis. Abstracts listed in Dissertation Abstracts International or Masters' Abstracts International should contain appropriate key words and phrases designed to assist electronic searches."

— MUN School of Graduate Studies

Acknowledgements

Put your acknowledgements here...

"Intellectual and practical assistance, advice, encouragement and sources of monetary support should be acknowledged. It is appropriate to acknowledge the prior publication of any material included in the thesis either in this section or in the introductory chapter of the thesis."

— MUN School of Graduate Studies

List of Publications

[1] John Doe John Doe, and Some Guy. Journal article SWGC title. Journal of Sample Journals, 1(12):1000–1024, 2002.

Contents

Abstract **iv**

Acknowledgements **v**

List of Publications **vi**

List of Figures **ix**

List of Tables **x**

List of Abbreviations **xi**

1 Introduction **1**

1.1 Objective 2

1.2 Possible Challenges 2

1.3 Contributions 3

2 Background Material **4**

2.1 Financial Portfolio 4

2.2 Reinforcement Learning 5

2.2.1 Reinforcement Learning Problem 6

2.2.2 Elements of Reinforcement Learning Problem 9

2.2.3 Solving Reinforcement Learning Problem 12

2.3 Deep Reinforcement Learning 13

3 Dealing with Errors **14**

4 Lorem Ipsum **15**

5 Handling Citations **16**

6 Conclusions **17**

Bibliography **18**

A Appendix: How to Add an Appendix **20**

A.1 Equations 20

B Appendix: How to Add Another One **22**

List of Figures

FC2.1 The Reinforcement Learning Problem 7

FA1.1 Image of a deadlocked Petri net at 40% scaling. 20

List of Tables

TA1.1 Fall Semester Enrollment 21

List of Abbreviations

CNN	Convolutional Neural Network
DP	Dynamic Programming
GPI	Generalized Policy Iteration
IIITB	International Institute of Information Technology Bangalore
MDP	Markov Decision Process
PM	Portfolio Management
RL	Reinforcement Learning
TD	Temporal Difference

CHAPTER 1

INTRODUCTION

One of the key issues an individual faces is how to allocate his wealth among various assets with ultimate goal to optimize some relevant measure of performance of the asset allocation such as profit, return etc. This problem of adjusting investments in a group of financial products is known as the portfolio management problem

Portfolio Management is the the art and science of making decisions to continuously reallocate funds into different financial investment products with the aim to maximize the returns while restraining the risk [1].

The PM process can be divided into two stages. The first stage starts with observation and experience and ends with belief about about the performance of the various securities. The second stage starts with the relevant belief of the about the future performance of the securities and ends with the choice of portfolio weights. Our work is concerned with the second stage of PM.

It will be very useful if a automated system can assist an investor by correctly indicating the trading action by utilizing the information available to the investor. In recent the application of artificial intelligence techniques for trading and portfolio management has seen significant growth [2]. Deep reinforcement learning has gained a lot of attention due to its remarkable achievements in playing video games and board games. It is also gaining popularity in the area of algorithmic trading. In our work we explore the applications of deep RL in portfolio management in the India equity market.

Objective

The popularity if algorithmic trading systems has been steadily growing and many major financial institutions have started to replace their traditional human traders with their electronic counterparts [3]. Developing these trading systems involves a significant amount of trial and error. The scope of this project is to build a trading system that uses RL to effectively manage financial portfolios. Being a fully machine learning solution our system is not restricted to any particular market. To examine the validity and profitability of the system we test it in the Indian equity market. We will examine the existing use of RL in PM and then use this established information to build our own novel approach to the problem of PM.

Possible Challenges

There are numerous existing systems and techniques that have been developed, proven to provide good return rates. However when developing a new idea from scratch, method of trial and error as well as market intuition and experience is often required. Several of the RL techniques that we will be using have only been recently established and thus the optimal choices of the parameters is still an open problem. We aim to to experiment with different parameter choices, investigate how they might the trading system and report the best returns.

Contributions

There are many existing deep learning systems which trade in the financial market. Most of these systems only try and predict the movements or trends in the financial market like Niaki and Hoseinzade [4], Fretias et al [5] etc. These systems predict the price of assets for the next period by extracting the necessary information form the history of prices [6]. These price predictions are not market actions, converting them into market actions will require an additional layer of logic. RL can convert these market prediction into market actions.

In this work we propose a RL trading system designed for the task of portfolio management. The core of the trading system is the combination of neural network predictors, which predict the prices of the assets of the portfolio, with a neural network whose job is to inspect the history of the assets and evaluate its potential for growth.

The former predicts the price value of asset sometime in the future from historical data. We use minimization of the mean square error as the criterion for prediction. The second neural network is used to generate a trading signal based on these predicted value, historical price data and an investment strategy. The evaluation score of each asset is discounted by the size of the intentional weight change for the asset in the portfolio and is presented to the softmax layer, whose outcome will be the new portfolio weights for the coming trading period. The portfolio weights define the market action of the RL agent.

CHAPTER 2

BACKGROUND MATERIAL

Financial Portfolio

A portfolio is a grouping of financial assets such as stocks, bonds, commodities, currencies and cash equivalents, as well as their fund counterparts, including mutual, exchange-traded and closed funds. Portfolios are held directly by investors or managed by financial professionals. These investor manage the distributions of funds into the various financial assets of the portfolio [7].

Portfolio management can be either active or passive. In active management a team of managers or a single manager actively manage the funds of a portfolio. The portfolio managers rely on analytical research, forecast of market and their own knowledge of the behaviour of the market to make investment decision. Whereas passive portfolio management refers to a strategy where investors stress on minimizing the investor fee and avoid unpleasant results of failing to correctly predict the future. Passive portfolio management involves imitating the performance of a particular market index.

Portfolio managers use a lot of terminologies in the process of managing a portfolio and we therefore provide a background on the terminologies that we will use

- **Asset allocation:** The practice of optimizing the risk/return profile of an investor by investing in a mix of asset that have low correlation to each other [8]. It is based on the understating that the different types of assets do not move in concert, and some of the assets are more volatile than other.

- **Diversification:** It is the practice of spreading funds among different asset classes. The practice of diversification provides a broad exposure to all the asset classes and thus spreads the risk and reward across asset classes [8]. It is difficult to know which particular subset of an asset class or sector is likely to outperform another, diversification seeks to capture the returns of all of the sectors over time but with less volatility at any one time.

- **Rebalancing:** It is the practice of adjusting the weights of the portfolio to return it to its original goal [8]. Rebalancing is done to retain an asset mix that best reflects an investor's risk/return profile. If the weights of the portfolio are not rebalanced the movement of the market can expose the investor to greater risks or reduce the opportunities of return. Rebalancing also ensures that the weights of the portfolio do not overweight a asset category.

- **Position:** It is the amount of security owned by the investor. The position can be of two types *long or* short. If the investor buys the asset with the hopes of the value of the asset increasing in the future and later sell it for a profit. The investor are said to take a long position. If the investor sells the asset with the hopes for decrease in price so that they can buy the assets back. The investors are said to take a short position.

Reinforcement Learning

Reinforcement learning (RL) is an a actively researched branch of machine learning that differs from the conventional methods. Supervised and unsupervised learning seem to exhaustively classify machine learning paradigm, but they do not. Supervised learning learns from a set of labelled examples provided by a domain expert. The objective of supervised learning is to extrapolate or generalize information from the labelled data to situations present not in the training data. Unsupervised learning finds structure hidden in collection of unlabelled data. Many people think of RL as a kind of unsupervised learning as it does not rely on any examples of right behaviour. But it is not the case RL tries to maximize the expected reward rather than finding a hidden structure. Reinforcement learning differs from both supervised and unsupervised learning and is third machine learning paradigm alongside the other too [9].

In RL the model is explicitly not told what actions should be taken rather the model is expected to experience the environment, evaluate the outcome of its action and learn what is the optimal choice [9]. The agent is not only expected to autonomously learn which action to take in order to maximize a numerical reward signal from the environment, but also the expected subsequent rewards that may be obtained from taking this action. The concepts of exploration, exploitation, the ability of the agent to recognize the importance of delayed rewards and learning from both successes and failures form the core of RL and differentiate it from other machine learning methods.

RL is best suited for interactive problems as it is almost impossible to decide the optimal behaviour and it is much easier to allow the agent learn by interacting with the environment. RL represents a general problem that can be solved using a variety of methods and there problems are known as reinforcement learning problem.

Reinforcement Learning Problem

The Reinforcement Learning problem is about learning from interaction how to behave in order to achieve a certain goal. The model is called the *agent* and the agent takes decision on which action to take. Upon taking the action the agent alters the state

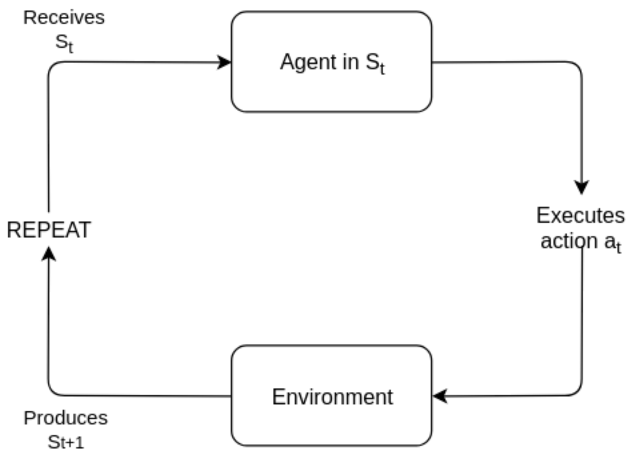

Figure FC2.1: The Reinforcement Learning Problem

of its surroundings. The surroundings, which comprise of everything outside the agent is known as the *environment*. For every action taken by the agent the environment responds by advancing the to a new state and generating a reward signal. They ultimate goal of the agent is to maximize the expected cumulative rewards [10].

The definition of the state in a RL problem is very important for the success of the agent. The state should contain enough information so that the agent can fully utilize the information and make a decision. On the other hand it is unrealistic and unhelpful to define a state with more information than the agent should know at an expected point in time. This can be summarized to say that an agent is not expected to know everything important to it at all time points but is expected to remember anything that it has already experienced [9]

Mathematically this is very similar to the Markov property which is commonly used in stochastic processes. Informally the Markov property states that conditioned on the present value, the future value of the process is independent of the past values. For RL problems this means that the reward and the state signal that the agent will receive at time $t+1$ is only depends on the the state and action at time t.

The Markov property is important as it allows one to calculate the dynamics of th transition to the next state purely based on the current state and action. A Markov system with a reward signal for transition from one state to another is known as a Markov Decision Process(MDP). The MDP tries to capture a world in the form of a grid by dividing it into states, actions, models, and rewards. The solution to an MDP is called a policy and the objective is to find the optimal policy for that MDP task. A RL task composed of a set of states, actions, and rewards that follows the Markov property would be considered an MDP [11].

A *finite* MDP is one in which the sets of states, actions and rewards all have finite number of elements. For a finite MDP the transition probabilities give the probability of transitioning from one state to another by taking some action. Let s be the initial state, a be the action and s' be the subsequent state then the transition probabilities is given by.

$$\mathbb{P}^a_{ss'} \doteq P(s_{t+1} = s' | s_t = s, a_t = a) \qquad \text{(Eqn 2.1)}$$

And the expected reward of taking an action a at time t depends only of the current state s and the subsequent state s'.

$$\mathbb{R}^a_{ss'} = \mathbb{E}\big[r_{t+1} | s_{t+1} = s', s_t = s, a_t = a\big] \qquad \text{(Eqn 2.2)}$$

The MDP framework is abstract and flexible and can be applied to different problems in different ways. Much of the current theory of RL is limited to the finite MDPs [9]

Elements of Reinforcement Learning Problem

Beyond the agent, the environment and states there are other important elements of RL: a *policy*, *reward* signal, the *value function*, and the *action-value* function [9].

The policy defines the behaviour of the agent at any given time. It is the mapping of state, action pairs to the probabilities of selecting that action at each time step t. The aim of a RL algorithm is to specify how the agent should evaluate and update the policy given the experiences it has seen [9]. The policy is usually represented by the symbol π

The reward signal defines the aim of the RL problem. The use of reward signal to define the idea of goal is one of the most distinctive features of RL. It might appear limiting at first to define the goal of the agent in terms of the reward signal, but in practice it has proved to be very flexible. At each time step the environment sends a single number to the RL agent called the reward. The reward signal defines the good and bad events for the agent [9]. The reward signal should be designed such that it accurately reflects the aim of the overall objective function rather than various sub-goals. An example of designing a simple reward function can be taken from an agent who is learning to play Chess. The reward can be +1 for a win, -1 for a loss and 0 for a tie. If we also assign reward signal for each of the opponents pieces captured, the agent will learn to focus on purely on capturing the opponents pieces rather than winning the aiming to win the game and therefore acting against its intended behaviour.

As mentioned earlier the goal of the agent is to maximize the expected cumulative rewards. The expected cumulative rewards are called the *expected returns*, where the return is defined as some function of the reward sequence. The simplest definition of return is the sum of the reward from a given time period to a terminal time period.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \qquad \text{(Eqn 2.3)}$$

In Eqn 2.3 T is the final time step. Eqn 2.3 as defined guarantees that the agent will reach a terminal state, implying that T $< \infty$. These tasks are called *episodic* tasks as they start in valid starting state, transitions through a finite of states and then reach a terminal state. However in many cases the agent-environment interaction can continue indefinitely i.e. T = ∞. These type of tasks are called *continuing* tasks. Eqn 2.3 cannot be used for continuing tasks as the sum can diverge to infinity and will be of little use to the agent. To counter this problem we use *discounting*, which places more emphasis on immediate reward than it does to future rewards. We choose a parameter $\gamma \in [0,1]$ called the discounting parameter and the return is defined as.

$$G_t = R_t + \gamma^1 R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad \text{(Eqn 2.4)}$$

The value of the discounting factor determines the present value of future rewards. If the value of γ is zero then the agent is myopic i.e. it is concerned with maximizing only the current rewards. If the value of γ the agent becomes more far-sighted and values immediate and future rewards equally. If $\gamma < 1$ then the infinite sum in Eqn 2.4 has as finite value.

The *value function* represents how good the state is for the agent to be in. It is equal to the expected total reward for an agent starting from the state s. The value function is specific to the policy which the agent selects the action to perform. If the agent selects action from the policy π in state s then the value function is given by.

$$v_\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| s_t = s\right] \qquad \text{(Eqn 2.5)}$$

Among all the possible value functions there exists a an *optimal value function* that has higher value than all the other value functions for all the states.

$$v_{\pi}^{*}(s) = max_{\pi} v_{\pi}(s) \qquad \forall s \in \mathbb{S} \tag{Eqn 2.6}$$

In Eqn 2.6 $\mathbb{S}$ is the set of all possible states. The policy that corresponds to the optimal value function is called the optimal policy.

$$\pi^{*} = argmax_{\pi} v_{\pi}(s) \qquad \forall s \in \mathbb{S} \tag{Eqn 2.7}$$

Solving a RL tasks roughly means finding a policy that achieves a lot of reward over the long run i.e. the optimal policy.

Similar to the value function, the state action function defines the state action function or the *Q function*. The Q function gives the expected return the agent will get starting from state s, taking action a and thereafter following the policy π.

$$q_{\pi}(s,a) = \mathbb{E}_{\pi}\big[G_t | S_t = s, A_t = a\big] = \mathbb{E}_{\pi}\big[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\big] \tag{Eqn 2.8}$$

The value function v_{π} and the action-value function q_{π} are both estimated from experience. The fundamental problem that is used throughout RL to estimate the value and action-value function is that they satisfy recursive relationship [9]. For any policy π and any state s the following condition hold between the value of the state s and the value of the successor states.

$$\begin{aligned} v_\pi &\doteq \mathbb{E}_\pi\big[G_t|S_t=s\big] \\ &= \mathbb{E}_\pi\big[R_t+\gamma G_{t+1}|S_t=s\big] \\ &= \sum_a \pi(a|s)\sum_{s'}\sum_r p(s',r|s,a)\big[r+\gamma\mathbb{E}_\pi[G_{t+1}|S_{t+1}=s']\big] \\ &= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\big[r+\gamma v_\pi(s')\big] \end{aligned} \tag{Eqn 2.9}$$

Eqn 2.9 is the *Bellman equation* for v_π and it expresses the relationship between the value of a state and the value of its successor state. Bellman equation for the basis of a number of ways that can be used to compute v_π

Solving Reinforcement Learning Problem

The dynamics of the RL problem are not known and there are a large number of possible states therefore alternatives approaches are used to solve the reinforcement learning problem. Most methods use the Generalize Policy Iteration(GPI) to solve RL problems.

GPI iteratively performs policy evaluation and policy improvement until sufficient convergence is observed. Policy Evaluation evaluates the value function for all the states with respect to the current policy. Policy improvement then uses these estimates to create a new policy that is greedy with respect to these estimates. The iteration are stopped when thee are no changes observed in the value function or the policy between iterations. It has been shown that the converged policy in GIP is actually the optimal policy.

Different methods can be used for policy evaluation and improvement/ We will briefly discuss a few of them

- **Dynamic Programming(DP)** refer to a set of algorithms that can be used to compute the optimal policies given a perfect model of the environment as a MDP. DP algorithms recursively update the value function for each state by starting at that state and use the Bellman equation to update the values of all the states based on the expectation of their reward and the estimate of the values of the subsequent states. The process of using the estimates of values of the subsequent states to estimate the value of the current state is known as *bootstrapping*. DP algorithms are of limited utility in RL both because of there assumptions of a perfect environment and because of their great computational cost [9].

- **Monte Carlo Methods** compute the optimal policy solely based on experience and do not require a full description of the environment, as long as we can produce sample runs from the environment. The value function is computed by averaging the return from each state over a number of sample runs also known as traces, until a convergence is reached. As we require each trace run to eventually terminate, Monte Carlo methods are only valid for episodic tasks. The policy is made greedy with respect to the value function computed and the algorithms repeats until it converges to the optimal policy.

- **Temporal Difference(TD) Learning** combine the best of DP and Monte Carlo methods. Like in DP they bootstrap by using the estimates to update other estimates. They also sample like Monte Carlo methods and do not require a full model of the environment to as they compute their estimates based on experienced runs. TD methods use the difference of value estimate for the states experienced at different points in time. TD methods do not wait until the end of an episode before updating their value estimates and thus can be used for non-episodic tasks.

Deep Reinforcement Learning

CHAPTER 3

DEALING WITH ERRORS

CHAPTER 4

LOREM IPSUM

CHAPTER 5

HANDLING CITATIONS

CHAPTER 6

CONCLUSIONS

That’s all folks!

Bibliography

[1] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[2] Xiu Gao and Laiwan Chan. An algorithm for trading and portfolio management using q-learning and sharpe ratio maximization. In *Proceedings of the international conference on neural information processing*, pages 832–837, 2000.

[3] James Cumming, Dr Dalal Alrajeh, and Luke Dickens. *An investigation into the use of reinforcement learning techniques within the algorithmic trading domain*. PhD thesis, Masters thesis, Imperial College London, United Kiongdoms, 2015. URL http , 2015.

[4] Seyed Taghi Akhavan Niaki and Saeid Hoseinzade. Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, 9(1):1, 2013.

[5] Fabio D Freitas, Alberto F De Souza, and Ailson R de Almeida. Prediction-based portfolio optimization model using neural networks. *Neurocomputing*, 72(10-12):2155–2170, 2009.

[6] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

[7] Stoyan V Stoyanov, Svetlozar T Rachev, and Frank J Fabozzi. Optimal financial portfolios. *Applied Mathematical Finance*, 14(5):401–436, 2007.

[8] Frank K Reilly and Keith C Brown. *Investment analysis and portfolio management*. Cengage Learning, 2011.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[10] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[11] Martijn van Otterlo and Marco Wiering. Reinforcement learning and markov decision processes. In *Reinforcement Learning*, pages 3–42. Springer, 2012.

APPENDIX A

APPENDIX: HOW TO ADD AN APPENDIX

This is Appendix A.

You can have additional appendices too, (*e.g.*, `apdxb.tex`, `apdxc.tex`, *etc.*). These files need to be included in `thesis.tex`.

If you don't need any appendices, delete the appendix related lines from `thesis.tex`.

Equations

An example mathematical formulae is show in Eqn 1.1.

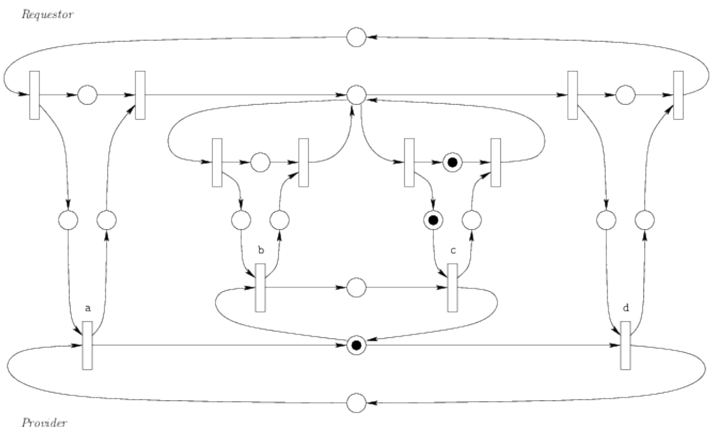

Figure FA1.1: Image of a deadlocked Petri net at 40% scaling.

Table TA1.1: Fall Semester Enrollment

	Undergraduate			Graduate		
	F/T	P/T	Total	F/T	P/T	Total
2004	13,191	2,223	15,414	1,308	879	2,187
2005	13,184	2,143	15,327	1,375	920	2,295
2006	12,809	2,224	15,033	1,373	899	2,272
2007	12,634	2,155	14,789	1,403	899	2,302
2008	12,269	2,208	14,477	1,410	1,005	2,415
2009	12,382	2,323	14,705	1,567	1,106	2,673

$$\sum_{i=0}^{n} i^2 \qquad \text{(Eqn 1.1)}$$

APPENDIX B

APPENDIX: HOW TO ADD ANOTHER ONE

This is Appendix B.