CS221 Project Proposal
Deep Reinforcement Learning in Portfolio Management

Ruohan Zhan rhzhan@stanford.edu
Tianchang He th7@stanford.edu
Yunpo Li yunpoli@stanford.edu

1. Background

In this project, we consider the problem of Portfolio Management where a limited resource must be allocated among a set of assets to maximize the expected return. It is an extremely complex problem and requires insights in many fields. The advance in deep learning in recent years spawned many attempts to automate decision making in finance [1] [2] [3]. [3] achieved promising result with models using LSTM to train a trading agent. After the training the agent learned to earn long term profit and avert from risky investments. Others like [1] generated 4-fold return during a period of 50 days in managing portfolio in cryptocurrency market using an Ensemble of Identical Independent Evaluators (EIIE) architecture. However others failed to replicate its results [4].

We wish to approach this problem using deep reinforcement learning and possibly integrate outside information such as news sentiment to produce an allocation strategy to maximize its return.

2. Problem Statement

Consider a stock market with L different stocks, we limit the portfolio management action only happens at discrete time steps $\{0, T, 2T, 3T, 4T, \dots\}$. At each time step kT, the total wealth is M^k, and we assume the agent invests all M_k plus the return in this time period on the next time step. The stock price is

$$S^k = (S_1^k, \dots, S_L^k). \tag{1}$$

The portfolio is

$$P^k = (P_0^k, P_1^k, \dots, P_L^k) \tag{2}$$

where $\sum_{i=0}^{L} P_i^k = 1$, P_0^k is the percentage of reserved capital, $P_j^k \geq 0, j = 1, \dots, L$ is the percentage of capital allocated into different stocks.

Profit At time step kT, we use P^k portfolio, then our profit is:

$$r^k = \sum_{i=1}^{L} \frac{M^k P_i^k}{S_i^k}(S_i^{k+1} - S_i^k). \tag{3}$$

Transaction Cost For simplicity, we only consider the transaction cost consists of two parts: the cost proportional to the change of stock assets, and the cost due to the change of portfolio and proportional to the whole wealth.

$$\begin{aligned} C^k = &\sum_{i=1}^{L} c_i \left| \frac{M^k P_i^k}{S_i^k} - \frac{M^{k-1} P_i^{k-1}}{S_i^{k-1}} \right| S_i^k \\ &+ c_0 \mathbf{1}_{[P^k \neq P^{k-1}]} M^k, \end{aligned} \tag{4}$$

where c_i is the cost ratio of change in stock i, and c_0 is the cost ratio of change in portfolio given the whole wealth.

Return Between kT to $(k+1)T$, the return from our portfolio is:

$$\begin{aligned} R^k &= \frac{r^k - C^k}{M^k} \\ &= \sum_{i=1}^{L} \frac{P_i^k}{S_i^k}(S_i^{k+1} - S_i^k) - \sum_{i=1}^{L} c_i \left| \frac{P_i^k}{S_i^k} - \frac{M^{k-1} P_i^{k-1}}{M_k S_i^{k-1}} \right| S_i^k \\ &\quad - c_0 \mathbf{1}_{[P^k \neq P^{k-1}]} \end{aligned} \tag{5}$$

States (Input) The states is the historical stock prices plus the previous portfolio(if transaction costs included).

$$\text{State}^k = (S^{k-N}, S^{k-N+1}, \dots, S^{k-1}, S^k, P^{k-1}) \tag{6}$$

N can be a fixed finite number, which means we use the windowed prices in states. We can also let $N \to +\infty$, then recurrent neural network could be applied to this problem.

Actions (Output) Actions is defined as the portfolio P^k at each time step.

Policy For the deep reinforcement learning methodology, policy is our learned deep neural networks. Given the states, we use this network to extract features and acquire corresponding action. Based on the return for this time period, we optimize our networks to increase the return.

3. Data

Our data is chosen to be the stock prices over a certain period of time from Standard & Poor's 500 (S&P 500).

Data	Year	Source
Stock Prices	2014-2017	Yahoo Finance

4. Methods

4.1. Baselines

Here we take the famous Eigen-portfolio as our baseline. For time period $[kT, (k+1)T]$, denote expected profit return for different stocks are $E[r^k] = (E[r_1^k], \ldots, E[r_L^k])$. The eigen-portfolio idea is as follows: given a portfolio P^k, we hope to maximize $E[P^k \cdot r^k]$. Denote $cov(r^k)$ is the covariance of Return for L stocks, then the normalized eigenvector with largest eigenvalue of $cov(r^k)$ is called eigen-portfolio, which has been empirically verified to have the largest correlation with the market.

In order to estimate covariance matrix $cov(r^k)$, we use the weighted average based on the historical data. The weight for data from $[(k-p)T, (k-p+1)T]$ is $\exp(-(p\mu))/C$, where μ is a model parameter and C is the denominator for normalization.

$$\begin{aligned} \hat{E}(r^k) &= \sum_{p=1}^{P} \frac{\exp(-(p\mu))}{C} r^{k-p} \\ \hat{cov}(r^k) &= \sum_{p=1}^{P} \frac{\exp(-(p\mu))}{C} \times \\ &\quad (r^{k-p} - \hat{E}((r^{k-p}))(r^{k-p} - \hat{E}((r^{k-p}))^T \end{aligned} \tag{7}$$

After getting eigen-porfolio, we use equation (5) to calculate return with transaction costs.

4.2. Oracle

The oracle of this problem is set to be as the agent would know the true stock prices at the next time step. Thus, he/she can simply maximize the return of current time step in equation (5) by varying portfolio P. The resulted values are the local minimums for each time step, but let's say they are good for now to give us an idea of how well the agent can possibly perform.

4.3. Deep Reinforcement Learning

Deep reinforcement learning can be applied in several different ways. One way is to take a the data (e.g. prices, allocations) in a time window as input to a network, which outputs a distribution. Another major trend is to take into account the fact that the data comes in the form of a time series, thus the application of different Recurrent Neural Network (RNN) cells such as Gated Recurrent Unit (GRU) and Long-Short Term Memory (LSTM) is natural and is applied in [1] and [3]. We wish to try different architectures of the network and compare them to find which one gives best result.

5. Evaluation

We use the reward, i.e. total percentage return to evaluate the performance of our agent on test set, which consists of stock prices of other companies. We wish to compare the average return between our agent and the baseline to evaluate our method. We also attempt to use other statistical evaluations such as Sharpe Ratio, which is defined as the ratio of the expected return and the total risk, to measure the risk of a certain portfolio.

6. Preliminary Results

As a starting point, we chose to use data of 10 stocks over 3 years (companies are: AAPL, ADSK, EBAY, FB, GOOGL, ITNC, ITNU, NFLX, ORCL, SYMC) and calculated the return with Baseline and Oracle. As it's clear in figure 1, the gap between oracle and baseline is consistent with time. The covariance matrix estimated in baseline comes from noisy historical data, thus has limited accuracy in representing current market. Our goal is to use deep reinforcement learning to reduce the gap from oracle, and integrate enough information to characterize market trend.

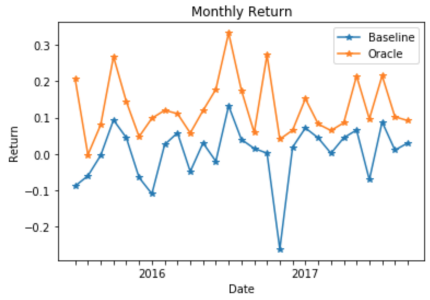

Figure 1. Monthly Return

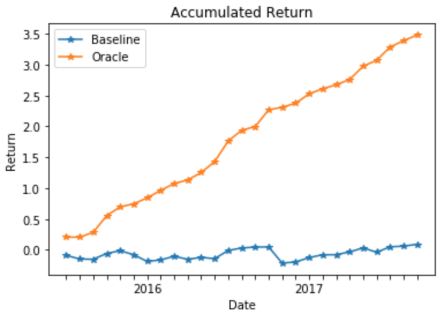

Figure 2. Accumulated Return

7. Challenges

The portfolio management problem is notoriously difficult to approach due to its complicated nature. Data in finance is known for suffering from large amount of noise, which is why simpler models often outperforms the deep learning models. Also the market might be affected by unpredictable events. Hence extracting features solely from stock prices is extremely challenging.

References

[1] A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem, Jiang et al, arXiv:1706.10059v2

[2] Independent Factor Reinforcement Learning for Portfolio Management, Li et al, The Chinese University of Hong Kong

[3] Agent Inspired Trading Using Recurrent Reinforcement Learning and LSTM Neural Networks, David W. Lu, arXiv:1707.07338

[4] https://github.com/wassname/rl-portfolio-management