


Deep Reinforcement Learning for Portfolio Management

Nov 22, 2017  Deep Learning (/~zhan527/tags/deep-learning), Reinforcement Learning (/~zhan527/tags/reinforcement-learning), Portfolio Management (/~zhan527/tags/portfolio-management), Convolutional Neural Networks (/~zhan527/tags/convolutional-neural-networks), Long Short-Term Memory (/~zhan527/tags/long-short-term-memory)

er.php?u=http%3a%2f%2fwww-scf.usc.edu%2f~zhan527%2fpost%2fcs599%2f)

?

Learning%20for%20Portfolio%20Management&url=http%3a%2f%2fwww-%2fcs599%2f)

Article?mini=true&url=http%3a%2f%2fwww-%2fcs599%2f&title=Deep%20Reinforcement%20Learning%20for%20Portfolio%20Management)

hare.php?url=http%3a%2f%2fwww-%2fcs599%2f&title=Deep%20Reinforcement%20Learning%20for%20Portfolio%20Management)

rcement%20Learning%20for%20Portfolio%20Management&body=http%3a%2f%2fwww-%2fcs599%2f)

Table of Contents

- ◦ Problem Formulation
 - Key Assumptions and Goal
 - MDP formulation
 - State and Action
 - State Transition
 - Reward
 - Datasets
- Methods
 - Data Preprocessing
 - Numerical Data
 - News Data
 - Predictor Network
 - Policy Network
 - Optimal Action and Imitation Learning
 - Deep Deterministic Policy Gradient (DDPG)
- Results
 - Imitation Learning
 - DDPG
 - Imitation Learning vs DDPG
 - Testing on Unseen Stocks
 - Related Work
 - Future Work

Problem Formulation

In this project, we would like to manage portfolio by distributing our investment into a number of stocks based on the market. We define our environment similar to this paper Jiang et al (<https://arxiv.org/abs/1706.10059>).

Concretely, we define N to be the number of stocks we would like to invest. Without losing generality, at initial timestamp, we assume our total investment volume is 1 dollar. We define close/open relative price vector as:

$$y_t = [1, \frac{v_{1,t,close}}{v_{1,t,open}}, \frac{v_{2,t,close}}{v_{2,t,open}}, \dots, \frac{v_{N,t,close}}{v_{N,t,open}}] \quad (1)$$

where $\frac{v_{i,t,close}}{v_{i,t,open}}$ is the relative price of stock i at timestamp t . Note $y[0]$ represents the relative price of cash, which is always 1. We define portfolio weight vector as:

$$w_t = [w_{0,t}, w_{1,t}, \dots, w_{N,t}] \quad (2)$$

where $w_{i,t}$ represents the fraction of investment on stock i at timestamp t and $\sum_{i=0}^N w_{i,t} = 1$. Note that $w_{0,t}$ represents the fraction of cash that we maintain. Then the profit after timestamp T is:

$$p_T = \prod_{t=1}^T y_t \cdot w_{t-1} \quad (3)$$

where $w_0 = [1, 0, \dots, 0]$. If we consider a trading cost factor μ , then the trading cost of each timestamp is:

$$\mu_t = \mu \sum | \frac{y_t \odot w_{t-1}}{y_t \cdot w_{t-1}} - w_t | \quad (4)$$

where \odot is element-wise product. Then equation 3 becomes:

$$p_T = \prod_{t=1}^T (1 - \mu_t) y_t \cdot w_{t-1} \quad (5)$$

Key Assumptions and Goal

To model real world market trades, we make several assumptions to simplify the problems:

- We can get any information about the stocks before timestamp t for stock i . e.g. The previous stock price, the news and tweets online.
- Our investment will not change how the market behaves.
- The way we calculate profit in equation 3 can be interpreted as: At timestamp t , we buy stocks according to the portfolio weight vector w_{t-1} computed by history data at **open** price and sell all the stocks at **close** price. This may not be true in practice because you will not always be able to **buy/sell** the stock at **open/close** price.

The **goal** of portfolio management is to maximum p_T by choosing portfolio weight vector w at each timestamp t based on history stock information.

MDP formulation

State and Action

We define state s_t as o_t , where o_t is the obseration of timestamp t . As the time goes by, the impact of history data decreases. Thus, we only consider the history price and news in a fixed window length W . Hence,

$$o_t = [\vec{v}_{1,t}, \vec{v}_{2,t}, \dots, \vec{v}_{N,t}]$$

where $\vec{v}_{i,t} = \begin{bmatrix} v_{i,t-W} \\ v_{i,t-W+1} \\ \vdots \\ v_{i,t-1} \end{bmatrix}$ and N is the number of stocks. The action a_t is just portfolio

weight vector w_t . Note that this is a continuous state and action space problem. We try to directly solve it in continuous space instead of using discretization in previous work Du et al (<http://cs229.stanford.edu/proj2009/LvDuZhai.pdf>) and Jin et al (<http://cs229.stanford.edu/proj2016/report/JinElSaawy-PortfolioManagementusingReinforcementLearning-report.pdf>). Essentially, we want to train a policy network $\pi_\theta(a_t|o_t)$.

State Transition

The underlining state evolution is determined by the market, which we don't have any control. What we can get is the observation state, which is the price. Since we will collect history price of various stocks, o_t is given by the dataset instead of o_{t-1} .

Reward

Instead of having reward 0 at each timestamp and p_T at the end, we take logarithm of equation 5:

$$\log p_T = \log \prod_{t=1}^T \mu_t y_t \cdot w_{t-1} = \sum_{t=1}^T \log(\mu_t y_t \cdot w_{t-1})$$

Thus, we have $\log(\mu_t y_t \cdot w_{t-1})$ reward each timestamp, which avoids the sparsity of reward problem.

Datasets

Stocks used: We choose 16 target stocks from NASDAQ100 that we feel are representative of different sectors in the index fund. They include "AAPL", "ATVI", "CMCSA", "COST", "CSX", "DISH", "EA", "EBAY", "FB", "GOOGL", "HAS", "ILMN", "INTC", "MAR", "REGN" and "SBUX".

Price Data: We collected history price of the stocks from 2012-08-13 to 2017-08-11. The price on each day contains open, high, low and close. We use 2012-08-13 to 2015-08-12 as training data and 2015-08-13 to 2017-08-11 as testing data.

News Data: We gather all tweets referencing the stocks from 2016-03-28 to 2016-06-15.

Additional Testing Data: As another form of backtesting, we randomly select 16 stocks from NASDAQ100 the network has never seen and test whether the network can generalize well.

Methods

We mainly consider model-free approach in our project. First, we train a predictor given a fixed history window length W of price and news. With the predicted price, we can train a policy network $\pi_\theta(a_t|s_t)$ to maximize our portfolio value at the end of the trading period. Note that in practice, they are trained end-to-end instead of separately.

Data Preprocessing

Numerical Data

Instead of using the raw open, high, low and close price, we normalize the history price as $(\frac{close}{open} - 1) \times scale$. There are two main advantages:

1. The history price are all in the same scale regardless of their actual price.
2. Since the final portfolio is determined by the close/open ratio of each timestamp, it serves as a better feature compared with raw price.

The `scale` factor is heuristically set to 100 in our experiments. For missing data, we pad all open, high, low, close as the close price of previous day.

News Data

The steps to preprocess the news data are:

1. First, we filter each tweets by the most frequent 2000 words.
2. We pad each word with end of sentence mark to make them equally long.
3. We use a sequence of end of sentence mark to fill the missing days.

Predictor Network

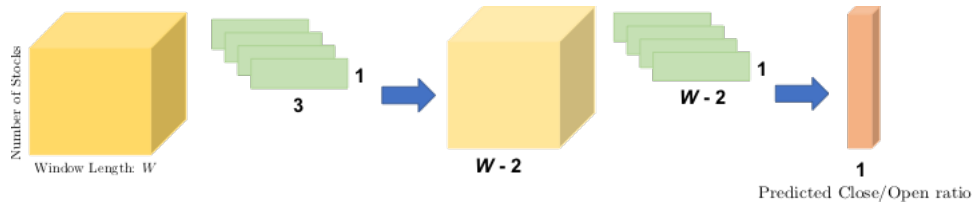


FIGURE 1: CNN predictor based on numerical history

The rationale behind the CNN predictor is adapted from Jiang et al (<https://arxiv.org/abs/1706.10059>). For each timeseries of each stock, we use a 1×3 kernel to gather information in each window, then combine all the information to produce a single vector for each stock. Note that these convolutional windows doesn't cover information across any two stocks.

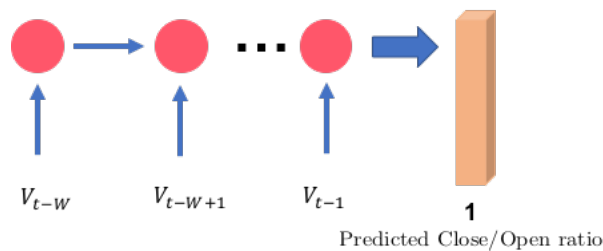


FIGURE 2: LSTM predictor based on numerical history

The LSTM predictor based on numerical history is very straight forward. Note that all the stocks share the same LSTM.

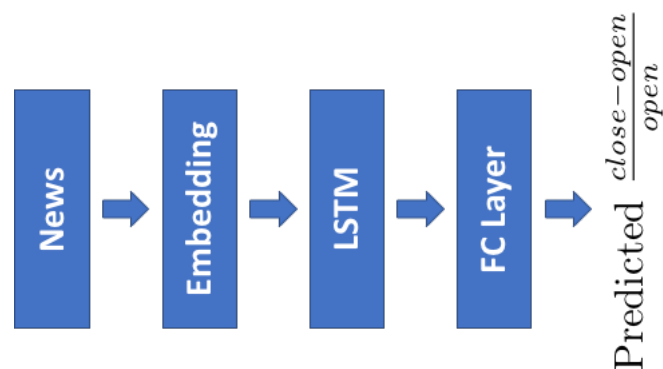


FIGURE 3: LSTM predictor based on history of news

For news based approach, we try to predict the $\frac{close-open}{open}$ ratio. We use pretrained 50d-GloVe to initialize the embedding layer followed by a LSTM and a fully-connected layer.

Policy Network

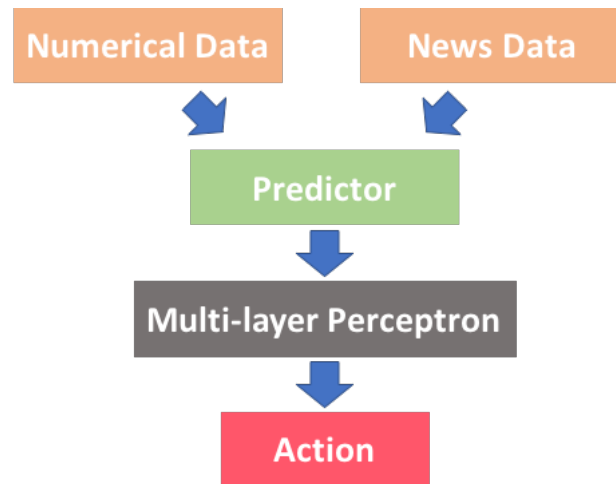


FIGURE 4: Policy Network

All the policy network follows the same topology with different predictors.

Optimal Action and Imitation Learning

Suppose we know the stock price of tomorrow, we greedily choose the stock with the highest close/open ratio (taking into account trading cost of changing stocks), buying as much as possible on the open and selling all at the close and it will yield the maximum portfolio value.

Given this fact, we can collect ground truth labels for each timestamp by choosing the optimal action. (e.g. $[0, 0, 1, \dots, 0, 0]$, the stock with highest close/open to be 1 and all the others to be 0) Then, we can train a policy network by imitating optimal action conditioned on the current observation of history prices o_t .

Deep Deterministic Policy Gradient (DDPG)

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ .
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
 Initialize a random process \mathcal{N} for action exploration
 Receive initial observation state s_1
 for t = 1, T **do**
 Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
 Execute action a_t and observe reward r_t and observe new state s_{t+1}
 Store transition (s_t, a_t, r_t, s_{t+1}) in R
 Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
 Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
 Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
 Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

Update the target networks:

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

end for
end for

FIGURE 5:

Deep Deterministic Policy Gradient Algorithm [Lillicrap et al](<https://arxiv.org/abs/1509.02971>)

We try to directly learn a policy network $\pi_\theta(a_t|s_t)$ by continuous action space reinforcement learning algorithm Lillicrap et al (<https://arxiv.org/abs/1509.02971>). We show the algorithm above. The configurations are:

- Actor Network: The same as policy network.
- Critic Network: A linear combination of actor network structure of state (observation) and action.
- Exploration noise: Ornstein-Uhlenbeck with zero mean, 0.3 sigma and 0.15 theta.
- For fairness, we train models with different settings in 500 episodes. Some of the models are not fully converged at that time though.

Results

Imitation Learning

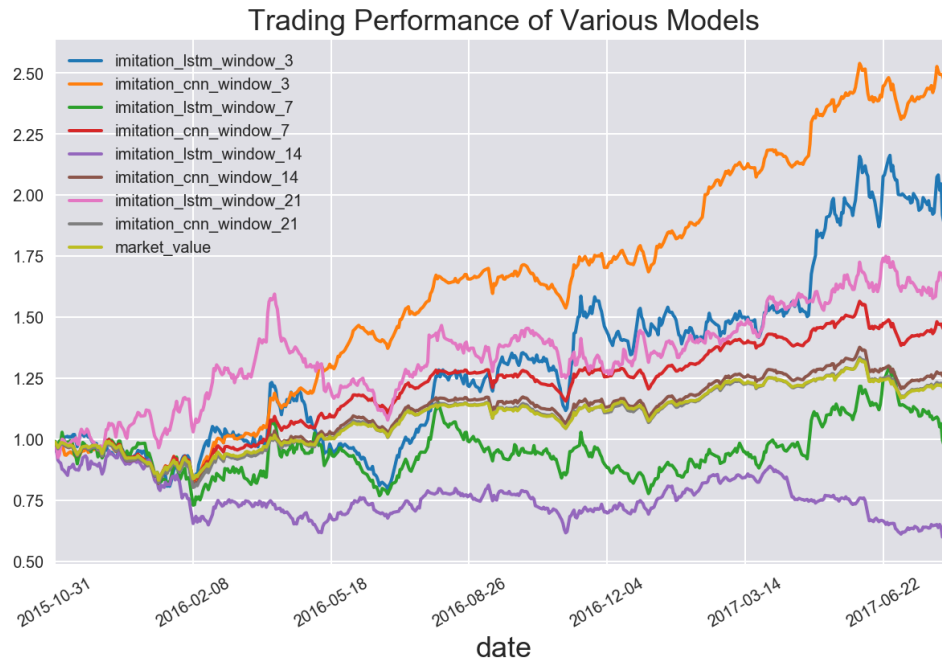


FIGURE 6: Results of trading on testing data using policy trained by imitation learning

The market value is obtained by equally distributing your investment to all the stocks. It turns out that smaller windows work better. Larger window means larger models and it tends to overfit very quickly since the training data is just around 1000.

DDPG

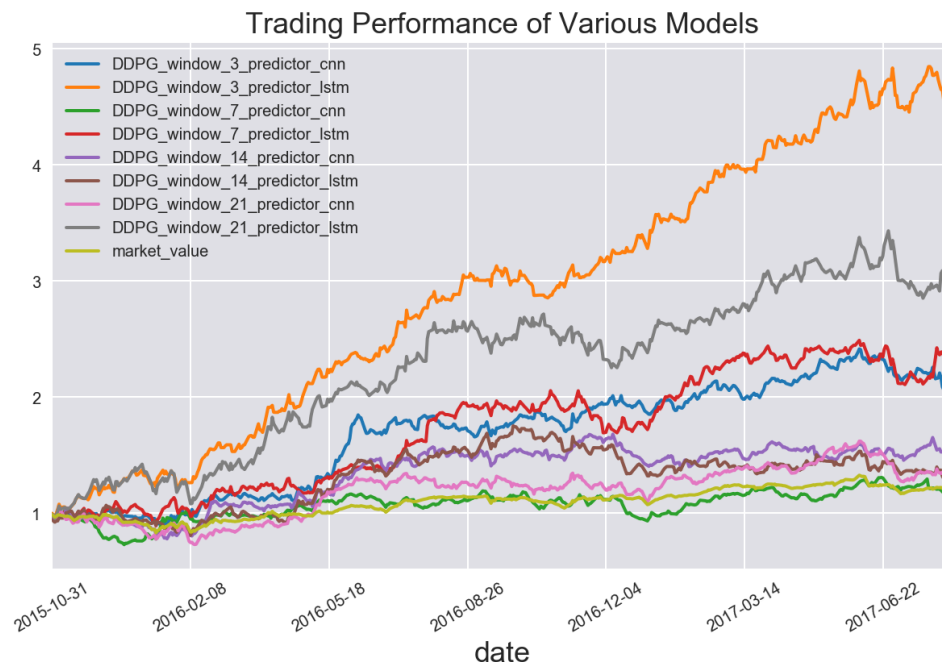


FIGURE 7: Results of trading on testing data using policy trained by DDPG

We find that LSTM predictor based models have better performance than the CNN predictor based models. It is not surprising that smaller windows perform best because of the temporal relationship among price in a short window.

Imitation Learning vs DDPG

Generally, models trained by DDPG is better than models trained by imitation learning. The difficulty of training good policy networks by imitation learning lies in the extreme tendency to overfit while DDPG doesn't suffer from this problem because each episode during training is sampled from the whole training period with different starting date and lengths.

Testing on Unseen Stocks

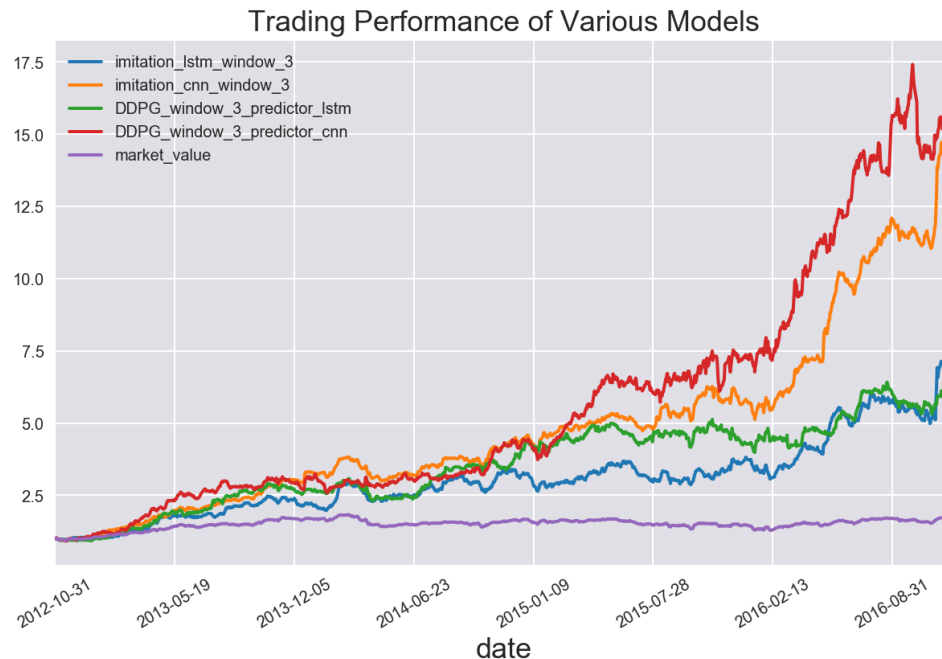


FIGURE 8:

Results of trading on 16 unseen stocks using best two policies trained by imitation learning and DDPG

To see how the network generalizes, we choose another 16 unseen stocks including 'FOX', 'FISV', 'EXPE', 'FAST', 'ESRX', 'DLTR', 'CTSH', 'CSCO', 'QCOM', 'PCLN', 'CELG', 'AMGN', 'WFM', 'WDC', 'NVDA' and 'STX'. The result shows that these models generalize pretty well and CNN-based predictor outperforms LSTM-based predictor in both cases.

Related Work

In Du et al (<http://cs229.stanford.edu/proj2009/LvDuZhai.pdf>) and Jin et al (<http://cs229.stanford.edu/proj2016/report/JinElSaawy-PortfolioManagementusingReinforcementLearning-report.pdf>), the author proposes to use DQN to trade in 2 stocks market with discretized state and action space. Their settings are far simpler than ours. In Jiang et al (<https://arxiv.org/abs/1706.10059>), the author proposes to use Deterministic Policy Gradient (DPG) to trade in bitcoin market. It's very hard to compare with them because of different dataset. But DDPG is strictly better than DPG in terms of performance and convergence time.

Future Work

- Combine news based predictor with numerical base predictor and train on a larger dataset with finer timestamp end-to-end with policy network.
- Model ensemble: combine the result of models trained by DDPG and imitation learning.
- Online learning: automatic scraper news and data each day, update the network and

make the action.

- Instead of optimizing portfolio value, we consider risk aware portfolio by optimizing the sharpe ratio define as $\frac{\text{Mean}(r_t)}{\text{Variance}(r_t)}$, where r_t is the rate of return in the period. It turns out to be a very difficult problem since it is not a standard MDP anymore.

© 2017 Chi Zhang · Powered by the Academic theme (<https://github.com/gcushen/hugo-academic>) for Hugo (<http://gohugo.io>).