CSLT TECHNICAL REPORT-20160036 [Sunday 8th January, 2017]

Deep Q-trading

Yang Wang[1,3*], Dong Wang[1,2], Shiyue Zhang[1,5], Yang Feng[1,4], Shiyao Li[1,5] and Qiang Zhou[1,2]

[*]Correspondence: wangyang@cslt.riit.tsinghua.edu.cn
[1]Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China
[3]Department of Computer Science, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China
Full list of author information is available at the end of the article

Abstract

Algorithmic trading is a hot topic in machine learning. Compared to other methods, reinforcement learning (RL), particularly Q-learning, can learn decision rules directly with reasonable reward, and therefore is suitable for learning trading strategies. Recently, Q-learning based on deep neural models, also known as deep Q-learning, has been successfully applied to some challenging tasks like game playing and robot motion. In this paper, we propose to employ deep Q-learning to build an end-to-end deep Q-trading system which can automatically determine what position to hold at each trading time. Our experimental results show that the deep Q-trading system can outperform the buy-and-hold strategy as well as the strategy learned by recurrent reinforcement learning (RRL) that was known to be more effective than Q-learning.

Keywords: Quantitive analysis; Deep learning; Reinforcement learning; Finance

1 INTRODUCTION

Algorithmic trading for stocks is attractive for both researchers and market practitioners. Existing approaches for algorithmic trading can be categorized into knowledge-based methods and machine learning (ML) based methods. Knowledge-based methods design trading strategies based on either financial research or trading experience; ML-based methods, in contrast, learn trading strategies from the market data in history. An obvious advantage of the ML-based methods is that they can discover profitable patterns that are yet unknown to people. Refer to the review paper [1] for more details about algorithmic trading.

Among various ML methods, reinforcement learning (RL) is particularly interesting, especially the Q-learning approach. First, Q-learning does not model the market, instead of focusing on the benefit (Q value) associated with actions. This avoids the errors caused by any market model. Second, Q-learning is suitable to do online learning, which enables quick adaptation to new market status. Third, Q-learning pays attention to long-term benefit rather than instantaneous reward, which is congruent with the goal of stock trading, maximizing long-term profit. Currently, reinforcement learning has been applied in financial analysis and investment by a multitude of researchers. For instance, Moody et al. [2] proposed a recurrent reinforcement learning (RRL) algorithm to optimize security portfolios. Gao et al. [3] used the relative risk-adjusted profit (sharp ratio) as performance function to train the trading system based on Q-learning. Du et al. [4] compared the performance of Q-learning and RRL, and reported that RRL achieved better performance in stock trading. Lee et al. [5] proposed an approach that incorporates multiple Q-learning agents to perform pricing and selection for stocks.

Recently, deep learning [6] has achieved remarkable success in a wide range of research areas, including speech processing, image processing, and natural language processing [7]. An important advantage of deep learning is that it can learn high-level features from raw signals layer by layer. Deep learning has been combined with the Q-learning recently, leading to a powerful deep Q-learning method. In a netshell, deep Q-learning is a Q-learning with a deep model (e.g., deep neural network) to identify status. Deep Q-learning has shown great power in a multitude of tasks. For example, it has been utilized to learn to play Atari games and the GO game, which achieved very impressive performance [8, 9]. To the authors' best knowledge, this powerful approach has not been applied to stock trading tasks, although deep models are supposed to be useful to discover market status from the noisy raw data.

In this paper, we apply the deep Q-learning approach to algorithmic trading. Our goal is to build a deep Q-trading system that determines when to buy and sell, based on the current and historical market data. Our preliminary experiments on the Hong Kong and US stock markets demonstrate that the deep Q-trading system is highly effective.

The paper is organized as follows: Section 2 describes the deep Q-learning approach. Section 3 presents the implementation details of the deep Q-trading system. Section 4 reports the experimental results, and Section 5 concludes the paper and discusses the future work.

2 DEEP Q-Learning

This section first briefly introduces the Q-learning algorithm and its application to stock trading, and then extends to the deep Q-learning approach.

2.1 Q-learning

Reinforcement learning is a general framework to deal with sequential decision tasks. At each time step t, RL observes the status s_t of the environment, takes an action a_t, and receives some reward r_t from the environment. With sufficient pairs of (action,reward), RL can learn an optimal decision policy π^* that maximizes the long-term accumulated reward $\sum_t r_t$.

Formally, Q-learning learns a value function $Q(s, a)$ that represents the expected accumulated reward when taking an action a when the environment status is s. This is formulated as follows:

$$Q_\pi(s, a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi], \tag{1}$$

where π is the decision policy, and R_t is the accumulated reward given by:

$$R_t = \sum_t \gamma^{t'-t} r_{t'}. \tag{2}$$

where $\gamma < 1.0$ is a discount factor to weight future reward. The goal of the learning is to find the best policy π^* that maximizes the expected return. The corresponding optimal Q-function is given by:

$$Q^*(s,a) = max_\pi Q_\pi(s,a). \tag{3}$$

The Q-function holds a nice property formulated as the Bellman equation:

$$Q^*(s,a) = r_t + \gamma \max_{a'} Q^*(s',a') \tag{4}$$

where s' is the new status after taking action a under status s. The main idea behind the Q-learning is that we can iteratively approximate the Q-function at each (s,a) using the Bellman equation, and this iterative update will converge to the optimal Q-function $Q^*(s,a)$.

2.2 Q-learning with deep neural networks

In the case of continuous state s, a neural network is often used to approximate the value $Q(s,a)$. This network is often referred as a Q-network [10]. The Q-network can be trained by minimizing the Q prediction error, i.e., the difference between the left-hand and right-hand side of Eq. (4). The loss function is formulated as follows:

$$L(\theta_i) = \mathbb{E}_{(s,a)\sim\rho(\cdot)}[(y - Q(s,a;\theta_i))^2], \tag{5}$$

where i denotes the training iteration, θ denotes the parameters of the Q-network. The training examples are in the form of (s,a,r,s'), and $\rho(s,a)$ denotes the distribution of the training examples. Additionally, y is the prediction of $Q(s,a)$ given by the Bellman equation:

$$y = r + \gamma \max_{a'} Q^*(s',a';\theta_{i-1})|s,a. \tag{6}$$

Note that y_t is computed with θ_{i-1}, the weight of the previous iteration. This loss function can be minimized by the stochastic gradient descend (SGD) algorithm. The gradient with respect to θ is given by:

$$\begin{aligned}\nabla_{\theta_i} L(\theta_i) = \mathbb{E}_{(s,a)\sim\rho(\cdot)}[(r + \gamma \max_{a'} Q^*(s',a';\theta_{i-1})\\ - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)]\end{aligned} \tag{7}$$

where $\nabla_{\theta_i} Q(s,a;\theta_i)$ can be easily computed by the back-propagate (BP) algorithm.

If the Q-network involves multiple layers, we obtain the deep Q-learning architecture. It has been well-known that deep learning is capable of learning hierarchical patterns, and the patterns learned by the high-level layers tend to be abstract and invariant against unexpected disturb. This abstraction and invariance is essentially important for stock trading. In stock markets, there is vast raw and noisy information (e.g., price, volume, correlation and auto-correlation among securities, special events of enterprises, macro economic indicators,...). This information forms the status of the market, but it is fairly challenging to identify the market status from these noisy information. With deep Q-learning, abstract and robust market patterns can be extracted from the vast, noisy and heterogenous information. These patterns form a much better representation for the market status s than the raw input features. As a summary, deep Q-learning involves a deep learning component that learns the market status s, and a Q-learning component that learns the value function $Q(s,a)$, although these two components are integrated as an entire deep Q-network in the real implementation.

3 Deep Q-trading system

In this section, we use the deep Q-learning algorithm to build an end-to-end algorithmic trading system. We consider a simple trading task that operates on a single security, and at each trading day t, only one operation (action) is allowed. The action a_t has three options: long (1), neutral (0) or short (-1), and a reward r_t is obtained. Our task is to learn a deep Q-function $Q(s,a)$ that maximizes the long-term accumulated profit $\sum_{t'} \gamma^{t'-t} r_{t'}$. No transaction cost is considered in this study. This trading approach based on the deep Q-learning algorithm is referred to as deep Q-trading.

In principle, the learning can be conducted by minimizing the loss function in Eq. (5) with SGD or any other gradient-based method, with the gradient given by Eq. (7). In practice, however, several issues need to be addressed before the learning can be successful. Firstly, since only one action is taken at each trading day, the total training data is quite limited; secondly, the Q-learning is intrinsically unstable [8]; thirdly, an online learning is required so that the Q-function can be quickly adapted to new status of the market. We describe some implementation details of our deep Q-trading system in what follows.

3.1 Online learning

A particular reason why many supervised learning approaches failed in stock market prediction and algorithmic trading is that the market is very volatile. This volatility is not only caused by short-term noise, e.g., unexpected events, but also by the intrinsic evolution of the market fundamentals, such as economic growth, technology development, population structure, etc. This means any models trained with historical data, no matter how accurate they are on the training set, will ultimately fail in new market conditions. An online learning approach that can quickly adapt the model to new market conditions is therefore essentially important for a practical trading system.

Fortunately, it is easy to implement an online version of the deep Q-trading system. We first train an initial deep Q-network using a small historical data, and then re-run the deep Q-learning from the beginning. At each trading day, a new training example (s_t,a_t,r_t,s_{t+1}) is added to a buffer that consists of recent trading history. The examples in the buffer are then used as a mini-batch to update the Q-network following Eq. (7).

3.2 Update scheme

It is well-known that Q-learning is unstable, particularly with deep neural networks. One reason is that a small change on the Q-network may change the Q-value significantly, hence change the decision policy as well. This is because that both the present Q value $Q(s,a)$ and estimated Q value $y = r + \gamma \max_{a'} Q^*(s',a')$ are computed by a single Q-network.

We design two approaches to mitigate this problem. First, notice that only three actions are allowed to be chosen in the trading task, and after each trading day, the reward associated with each action is actually known. This means that we don't need a random exploration to sample an action as in many reinforcement learning tasks [8]; instead we can emulate all the three actions to update the Q-network. This not only enriches the training data, but makes the training more stable.

Another method is to use a separate target network $\tilde{Q}$ to compute the estimated Q value y, as suggested by Mnih [8]. The gradient in Eq. (7) is then changed to:

$$\begin{aligned}\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}[(r + \gamma \max_{a'} \tilde{Q}(s',a';\tilde{\theta}) \\ - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)].\end{aligned} \tag{8}$$

The target network $\tilde{Q}$ is updated by interpolating with the latest Q, as follows:

$$\tilde{\theta} \leftarrow \tau\theta + (1-\tau)\tilde{\theta} \tag{9}$$

where τ is the interpolation factor. Note that the decision of the trading is made based on the target network $\tilde{Q}$, rather then the present network Q.

3.3 Reward function

Reward function is another key ingredient of the deep Q-trading system. Most of previous researches on ML-based methods often use instantaneous reward such as daily profit, either in an additive form $r_t = z_t - z_{t-1}$ or in a productive form $r_t = z_t/z_{t-1} - 1$, where z_t is the price of the target equity at the t-th day. This instantaneous reward, however, is not suitable for deep Q-learning, as the instantaneous reward is quite noisy so it can not provide reliable supervision for the model training. Furthermore this reward is not consistent with our goal of maximizing long-term profit[1]. We experimented with several long-term reward functions and found that the accumulated wealth over n days in the past was a good candidate for deep Q-trading. Formally, the reward is given by

$$r = (1 + sgn(a_t)\frac{z_t - z_{t-1}}{z_{t-1}})\frac{z_{t-1}}{z_{t-n}}, \tag{10}$$

where $a_t \in \{-1, 0, 1\}$ and $sgn(\cdot)$ is the sign function.

[1]Q-learning indeed targets for a long-term reward. Specifically, it relies on the Bellman equation to accumulate a series of discounted instantaneous reward to a long-term reward. However, the factorization assumption is fair strong and may not hold in practical usage.

4 EXPERIMENTS

The proposed deep Q-trading system is evaluated on two stock markets: Hong Kong and US, and choose HSI and S&P500 as the trading target. we first describe the data and configurations of the experiment, and then present the performance results.

4.1 Data and settings

The database used in the experiment involves 15 years of daily data of HSI and S&P500 downloaded from Yahoo finance, ranging from 01/01/2001 through 12/31/2015. This database is divided into a training data set (01/01/2001 - 12/31/2004) and a test set (01/01/2005 - 12/31/2015). Only the daily closing price is used in this study, though other features can be easily incorporated into the model and are under investigation.

We compare the deep Q-trading system with two baselines: the buy-and-hold (BH) strategy and the recurrent reinforcement learning (RRL) system [2, 11]. For deep Q-trading, the training dataset is used to initialize the deep Q-network, and then the system runs in an online fashion where trading decision making and model adaptation are conducted simultaneously. For the Q-learning part, the discount factor in Eq. (7) is set to $\gamma = 0.85$, and the reward window n in Eq. (10) is set to 100. For the deep-learning part, the architecture of the deep Q-network involves four layers in total (two are hidden), with the number of units set to $200, 100, 50$ and 3 respectively. The input units (features) are composed by the delta price $z_t - z_{t-1}$ of 200 days in the past, and the output units correspond to the three actions in trading. The learning rate for Q-network is $1e - 4$, and the training stops after 10 iterations. The interpolation factor τ in Eq. (9) is set to 0.0003, and the minibatch involves the training examples of the past 64 days. Note that the training examples involve all the 3 actions and their rewards at each trading day, so the minibatch size is actually 192.

As for the RRL system, the code provided by Yupu Song and Kartik Shetty is used to build the system[2]. Note that the original setting of the RRL system is based on monthly prices. For a fair comparison, we migrate the algorithm to daily prices. The input features are delta prices of the past 120 days, and the learning rate is set to 0.005. The maximum training epoch is set to 100. These parameters have been chosen to optimize and balance the performance of the RRL system on both of the two data sets (HSI and S&P 500). The RRL system is trained offline with the training data and then applied to the test data without any adaptation.

4.2 Experimental results

The results with the three trading approaches (BH, RRL, and deep Q-trading) on the HSI test set are presented in Fig. 1, and the results on the S&P500 test set are presented on Fig. 2. In each figure, the top plot reports the performance in terms of accumulated wealth (starting from the initial wealth 1.0), and the bottom plot presents the positions held by the deep Q-trading system. It can be seen that on both of the two test sets, the deep Q-trading system accumulates clearly more wealth than the other two systems. More detailed numerical comparison is shown

[2]https://github.com/MagicEthan/CS534_AI_Proj

	HSI			S&P500		
	BH	DQT	RRL	BH	DQT	RRL
Accumulated Return(%)	154	350	174	169	214	141
Sharp Ratio	0.28	0.59	0.89	0.34	0.45	1.23
Maximum Drawdown(%)	65	42	55	57	31	43

Table 1 Comparison of trading performance

in Table 1, where we report three widely used measures for stock trading: accumulated return, sharp ratio and maximum drawdown. It can be seen that our trading system outperforms the other two methods in terms of total return and maximum drawdown, but performs worse than the the RRL system in terms of sharp ratio. This is not surprising because the RRL system uses sharp ratio as its reward signal.

Comparing deep Q-trading and RRL, it seems that the former performs more consistently than the later: deep Q-trading performs well on both the two markets, while RRL performs poorly on S&P500. This advantage of deep Q-trading can be attributed to two advantages. One of them is the ability to detect the status of the market from the raw and noisy data. And the another is the online nature that adapts itself to new market status quickly. More interesting observations can be found in the two position plots. From the positions held by the Q-trading system, it seems that it has learned how to take different actions in different market situations. For example, in a downside market, it learns selling is more profitable than holding, and thus tends to hold a short position. This sensitivity against market status can be largely attributed to the power of the deep Q-network in discovering the status of the market from the vast and noisy historical price signals.

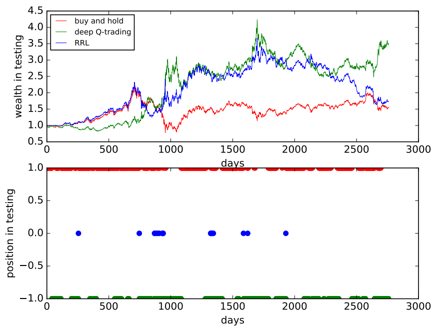

Figure 1 The performance of various trading strategies on HSI. The top plot shows the performance on the test set, evaluated in terms of accumulated wealth. The bottom plot shows the position held by the deep Q-trading system.

5 CONCLUSION

In this paper we propose a novel deep Q-trading method that applies the deep Q-learning approach to algorithmic trading. Compared to existing methods, deep Q-trading is able to detect market status from raw and noisy data, and pays attention to long-term returns. Our experiments on HSI and S&P demonstrated that the deep

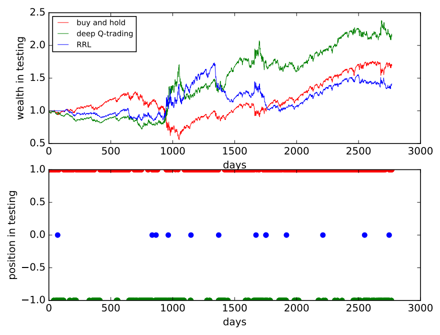

Figure 2 The performance of various trading strategies on S&P500. The notations are the same as in Fig. 1.

Q-trading system consistently outperforms the BH baseline, as well as the RRL system. Despite these interesting results, our study is still in a preliminary stage. In future work, we will investigate the contributions of other features, particularly the ones from financial research.

Acknowledgment

This work was supported by the National Science Foundation of China (NS-FC) Project No. 61371136, and the MESTDC PhD Foundation Project No. 20130002120011.

Author details
^{1}Center for Speech and Language Technology, Research Institute of Information Technology, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China. ^{2}Center for Speech and Language Technologies, Division of Technical Innovation and Development, Tsinghua National Laboratory for Information Science and Technology, ROOM 1-303, BLDG FIT, 100084 Beijing, China. ^{3}Department of Computer Science, Tsinghua University, ROOM 1-303, BLDG FIT, 100084 Beijing, China. ^{4}Huilan LTD , Beijing, China. ^{5}Beijing University of Posts and Telecommunications, Beijing, China.

References

1. Philip Treleaven, Michal Galas, and Vidhi Lalchand, “Algorithmic trading review,” *Communications of the ACM*, vol. 56, no. 11, pp. 76–85, 2013.
2. John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell, “Performance functions and reinforcement learning for trading systems and portfolios,” *Journal of Forecasting*, vol. 17, no. 56, pp. 441–470, 1998.
3. 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*. Citeseer, 2000, pp. 832–837.
4. Xin Du, Jinjian Zhai, and Koupin Lv, “Algorithm trading using q-learning and recurrent reinforcement learning,” *positions*, vol. 1, pp. 1.
5. Jae Won Lee, Jonghun Park, O Jangmin, Jongwoo Lee, and Euyseok Hong, “A multiagent approach to q-learning for daily stock trading,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 6, pp. 864–877, 2007.
6. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
7. Li Deng and Dong Yu, “Deep learning: Methods and applications,” *Foundations and Trends in Signal Processing*, vol. 7, no. 3-4, pp. 197–387, 2013.
8. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
9. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
10. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
11. John Moody and Lizhong Wu, “Optimization of trading systems and portfolios,” in *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*. IEEE, 1997, pp. 300–307.