DEEP REINFORCEMENT LEARNING FOR FINANCIAL TRADING USING PRICE TRAILING

Konstantinos Saitas Zarkias[1,2] *Nikolaos Passalis*[1,3] *Avraam Tsantekidis*[1] *Anastasios Tefas*[1]

[1]School of Informatics, Aristotle University of Thessaloniki, Greece
[2] School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden
[3]Faculty of Information Technology and Communication Sciences, Tampere University, Finland
kosz@kth.se , nikolaos.passalis@tuni.fi, {avraamt, tefas}@csd.auth.gr

ABSTRACT

Developing accurate financial analysis tools can be useful both for speculative trading, as well as for analyzing the behavior of markets and promptly responding to unstable conditions ensuring the smooth operation of the financial markets. This led to the development of various methods for analyzing and forecasting the behaviour of financial assets, ranging from traditional quantitative finance to more modern machine learning approaches. However, the volatile and unstable behavior of financial markets forbids the accurate prediction of future prices, reducing the performance of these approaches. In contrast, in this paper we propose a novel price trailing method that goes beyond traditional price forecasting by reformulating trading as a control problem, effectively overcoming the aforementioned limitations. The proposed method leads to developing robust agents that can withstand large amounts of noise, while still capturing the price trends and allowing for taking profitable decisions.

Index Terms— Deep Reinforcement Learning, Financial Markets, Price Forecasting, Trading

1. INTRODUCTION

Financial markets, where various securities and derivatives are traded, are at the heart of modern economies. For example, capital markets ensure that businesses will receive the necessary capital in order to grow and expand, while other markets, such as foreign exchange markets, enable the trading of various currencies, determining in this way the exchange rates and facilitating international trading. Apart from providing a practical tool for ensuring the smooth and efficient operation of businesses, financial markets also provide the opportunity for making profitable investments, given that we are able to forecast the future price trend of an asset. This has attracted the interest of numerous investors, who attempt to analyze and *predict the future behavior of various assets* in order to make the most appropriate investments. However, apart from predicting the price of assets for speculative purposes, analyzing the behavior of markets and promptly responding to unstable conditions is also extremely important to ensure the smooth operation of financial markets. For example, allowing speculative attacks or other forms of market manipulations to take place can have a tremendous negative effect on the actual economy.

The aforementioned reasons led to the development of various tools for analyzing and forecasting financial markets, ranging from traditional quantitative finance [1] to more modern machine learning approaches [2, 3]. Deep Learning (DL) methods were also recently used to exploit the vast amount of data collected from financial markets [4, 5, 6], further increasing the performance of these methods. However, most of these approaches attempt to predict the future direction of the price of an asset, instead of trying to directly learn a trading agent that will maximize the profit, which is usually the end goal of these methods. To this end, Reinforcement Learning (RL) methods were used to directly train a trading agent which will act in an *optimal way* in a given financial market [7, 8]. Combining deep learning with RL can further improve the trading performance, as it was recently demonstrated in [8]. Such agents can be used both for making speculative investments, as well as for better understanding the dynamics of markets, as described previously.

However, financial markets are intrinsically noisy and employing deep learning models, which exhibit an enormous learning capacity, in such environments often leads to overfitting, i.e., the models are prone to learning random price trails from the training data without being able to extract any useful knowledge for inferring the future behavior of an asset, as it was thoroughly demonstrated in the relevant literature [9]. In other cases, the models might be strongly influenced by the noise, especially if they are used for high frequency trading [9], limiting their ability to model the financial markets. It is worth noting we are not usually concerned with accurately predicting the exact price movement for each and every time step. Instead, we are usually interested in developing models that will capture the overall future trend of the market allowing for taking informed decisions.

with the deep Q-learning approach [10], is also proposed. It is experimentally demonstrated, using backtesting on real data collected from a foreign exchange market, that the proposed approach can indeed lead to more profitable positions, increasing the PnL (profit and loss) metric. At the same time, the proposed method also leads to developing agents that qualitatively exhibits the desired behavior, i.e., being robust to the noisy behavior of the price, while effectively capturing the trend of the market.

The rest of the paper is structured as follows. First, the related work is briefly introduced and compared to the proposed approach in Section 2. Then, the proposed method is introduced in Section 3, while the experimental evaluation is provided in Section 4. Finally, conclusions are drawn in Section 5.

2. RELATED WORK

Perhaps the most commonly used approach, when applying machine learning algorithms in financial problems, is to directly predict the future price (or price direction) of an asset [2, 3, 5, 9, 11, 12]. The main assumptions behind these approaches is that accurately predicting the future price of an asset allows for taking profitable trading decisions. Several models have been used to tackle this problem, ranging from Support Vector Machines [13] and simple feed-forward neural networks [14], to more powerful recurrent and convolutional neural networks [5, 9, 15]. However, the extremely volatile and unstable behavior of financial markets forbids the accurate prediction of future prices, leading to taking severely sub-optimal decisions, with simple strategies sometimes performing better than these models.

Using RL to directly tackle the task at hand, i.e., making the most appropriate trades, instead of predicting the price at each time-step, can overcome, to some extent, these limitations [7, 8, 16, 17]. For example, in [8], the complex behavior of price data originating from foreign exchange markets was modeled using DL. Then, a profitable strategy was learned using RL, where the financial positions that would result in maximizing the total reward were chosen. Note that the reward function was appropriately designed to ensure that maximizing the expected reward would lead to an agent capable of performing actions (trades) that also maximize the PnL. Even though these approaches aim to directly optimize the obtained profit, they are still vulnerable to overfitting and prone to noise.

In this work, we overcome the aforementioned limitations by proposing a method that is intrinsically regularized and able work under large amounts of noise. Therefore, instead of blindly following the noisy fluctuations of the original time-series, we propose using a margin-based approach that is capable of absorbing the (possibly) chaotic movements of financial time-series. To this end, the problem of maximizing the profit is re-formulated into a control problem, where the agent maintains an internal estimation of the current price and decides to follow the current price trend only if there are strong signals that dictate to do so. It is worth noting that the proposed approach share some similarities with max-margin approaches [18, 19]. However, in contrast with these approaches, that are merely used for classification tasks, the proposed method introduces a formulation adapted to the needs of financial time-series forecasting using RL (instead of employing traditional classification algorithms that are used to predict a predefined attribute). To the best of our knowledge, this is the first work that employs a novel control-based formulation for modeling the investors behavior and optimizing the behavior of trading agents using RL.

3. PROPOSED METHOD

Let p_t be the current price of an asset at time t. The goal of the proposed agent is to control its position, denoted by s_t, in order to follow the trend of the price. This position can be also interpreted as the internal *estimation* of the agent regarding the state of the market. Initially the agent starts at the same position as the price, i.e., $s_0 = p_0$. After each time-step the agent must decide whether or not its position regarding the current price must be updated. Three possible actions are available:

1. *stay* at the previous level, i.e., $s_{t+1} = s_t$,
2. *follow an upward trend*, i.e., $s_{t+1} = s_t + s_t \cdot \alpha$, and
3. *follow a downward trend*, i.e., $s_{t+1} = s_t - s_t \cdot \alpha$,

where α is the factor that control the change of the internal estimation of the agent. Note that the action of following an upward trend can be directly translated into a *long* position, since the agent estimates that an upward trend will follow. Similarly, a downward movement can be similarly translated into a *short* position. If the agent decides to do nothing, then the same position as before is maintained. Therefore, the reward the agent receives must be inversely proportional to the distance between its current position and the current value of the time-series, i.e., the current *control error* e_t:

$$e_t = |s_t - p_t|. \tag{1}$$

The closer the agent is to the current price, the higher the reward that will be obtained. Finally, note that the agent never exits the market, i.e., the stay action corresponds to maintaining the latest open position (either long or short), while simply avoiding updating its internal price trend. An example of an agent following a time-series in shown in Figure 1. The red line corresponds to the agent's trail, while the blue lines depict the price and the upper and lower margins.

Reward Shaping: Designing an appropriate reward function, that is able to accurately express the goal of the proposed agent, while rewarding the agent at each time-step, is critical for the smooth convergence and successful optimization using RL [20]. First, we set a margin m around the price that dictates when and how much the agent should be rewarded. If the agent is within the desired margin it receives a positive reward proportional to the distance from the closest margin. However, if the agent is outside this margin, a negative reward is obtained. This allows for learning agents that will absorb small price fluctuations without updating their estimation regarding the price trend, since maintaining a position close to the current price will still yield a large position reward. Therefore, the position-based reward is defined as:

$$r_t^{(p)} = \begin{cases} \frac{s_t - p_t + m}{m}, & \text{if } s_t \leq p_t \\ \frac{p_t - s_t + m}{m}, & \text{otherwise} \end{cases}. \tag{2}$$

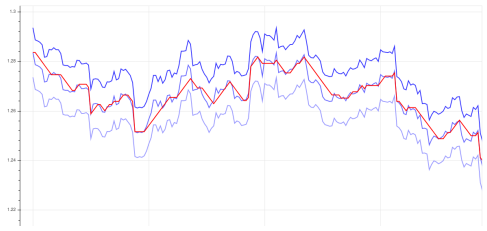

Fig. 1. The agent's trail (shown in red color) while following the time-series within a small margin ($m = 0.01$). Note that the agent quickly responds to most of the price fluctuations. The agent position was also frequently reset, due to the smaller margin that allowed only for smaller deviations around the current price. Figure best viewed in color.

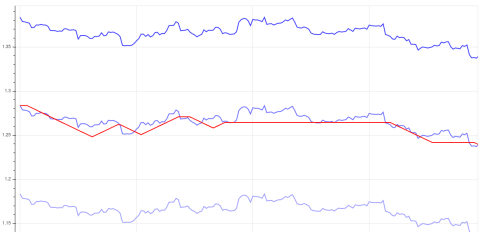

Fig. 2. The agent's trail (shown in red color) while following the time-series within a large margin ($m = 0.1$). Note that the agent absorbs most of the price fluctuations, while following the general trend of the time-series. Figure best viewed in color.

The agent receives a (positive) reward proportional to the control error as long as it maintains a small distance (at most m) from the current price. The effect of using different margins is demonstrated in Figures 1 and 2. Note that agent is capable of absorbing most of the price fluctuations when a larger margin is employed.

However, apart from following the price trend, the agent must pay a small commission/penalty each time that it is decided to update its estimation with a trend of the opposite direction. This commission further discourages the agent from closely following insignificant price fluctuations and corresponds to the commission that is usually paid in an exchange, along with the price slippage due to the delayed execution of orders. This commission is paid only if the agent switches from an upward trend to a downward trend and vice versa. Note that performing the stay action does not alter the action that will be used to calculate the penalty, since the agent always maintains an open position (either short or long). Therefore, let δ_t denote a binary variable that denotes whether the agent changed the direction of the price trend estimation:

$$\delta_t = \begin{cases} 1, & \text{if the agent changes its price trend estimation} \\ & \text{from upwards to downward (or vice versa)} \\ 0, & \text{otherwise} \end{cases} . \tag{3}$$

Then, the final reward is calculated as:

$$r_t = \begin{cases} r_t^{(p)} - |r_t^{(p)}| \cdot c, & \text{if } \delta_t = 1 \\ r_t^{(p)}, & \text{otherwise} \end{cases} , \tag{4}$$

where c is the commission, which proportionally reduces the obtained reward.

Deep Reinforcement Learning for price trailing: Several approaches have been proposed for learning agents that will behave optimally in order to maximize a given reward function. In this work, the Double Deep Q-learning approach was used [10]. A deep neural network was used to estimate the Q-values of the three possible actions. The neural network receives an input window composed of the l previous observation vectors. Each observation vector $\mathbf{x}_t$ is defined as:

$$\mathbf{x}_t = [p_t - s_t + m, s_t - p_t + m, \mathbf{a}_{t-1}]^T \in \mathbb{R}^5, \tag{5}$$

where $\mathbf{a}_{t-1} \in \{[1,0,0],[0,1,0],[0,0,1]\}$ is a one-hot vector that corresponds to the action performed at the previous step, while the first two values of the observation vector express the distance between the current estimation and the upper and lower margin of the agent. The distances from the two margins were appropriately normalized using z-score scaling.

The employed neural network consists of 3 hidden layers of 64 neurons each and one output layer with 3 neurons that correspond to the three Q-values. The PReLU activation function was used for the hidden layers [21], while no activation function was used for the output layer (in order to appropriately estimate the Q-values). The Adam algorithm was employed for learning the parameters of the network [22], while the learning rate was set to 0.001. The size of the experience replay pool was set to 100,000 and batches of 64 samples were sampled from the replay pool during the optimization. The optimization ran for 1,000 episodes, each one consisting of 500 steps. The agent was randomly initialized at a different point of the time-series for each episode. A linear exploration policy was em-

ployed during the optimization. Finally, the size of the input window was set to $l = 100$, the penalty cost was set to $c = 0.3$, the margin was set to $m = 0.01$, while the step for updating the estimation of the agent was set to $\alpha = 0.001$. These hyper-parameters were used for all the experiments conducted in this paper and they are expected to work well for a wide range of different financial time-series. To ensure the smooth behavior of the agent during the training process, the agent was automatically reset inside the working margin if it did not manage to stay within the predefined margin around the time-series.

4. EXPERIMENTAL EVALUATION

The proposed method was evaluated using data from the Euro - Dollar exchange rates collected between 2007 and 2015. The data were re-sampled with each value corresponding to the close price after a period of 4 hours. The first 12,000 exchange rates were used for training the agent, while the following 2,000 exchange rates were used for evaluating the behavior of the agent using data that were not seen during the training (back-testing).

The proposed method was also compared to a competitive RL method for learning the optimal way to perform trading, as proposed in [8]. To ensure a fair comparison between the evaluated methods, the reward function proposed in [8] was employed for learning a deep RL agent with the same Deep Q-learning algorithm, network structure, and input, as in the proposed approach. Furthermore, it was established that the method proposed in [8] works better by interpreting the neutral position as a *market exit* position that secures the obtained profit instead of continuously staying in the market. This approach is called "*Trading RL*" in the conducted experiments, since it directly optimizes an agent for performing trading, instead of following the trail of the time-series, as the proposed method does. All the hyper-parameters of the "Trading RL" method were appropriately tuned for the used foreign exchange time-series before conducting the experimental evaluation.

The final Profit and Loss (PnL) obtained using both the "Trading RL" approach and the proposed method are shown in Table 1. Note that regardless the used commission the proposed method outperforms the "Trading RL" approach. The PnL during the trading is also depicted in Figure 3. The proposed method leads to much more profitable positions, even though the agent is not directly optimized for performing trading, demonstrating the ability of the proposed agent to withstand the intrinsic noise of financial time-series.

Table 1. Backtesting: Comparing the final PnL obtained using the proposed method and the "Trading RL" method. The commission refers to the cost payed to the exchange for each unit of the corresponding currency that is traded.

Commission	Trading RL	Proposed Method
0	0.02	**0.27**
$2 \cdot 10^{-6}$	0.01	**0.26**
$2 \cdot 10^{-5}$	0.01	**0.25**

Fig. 3. Comparing the PnL during the backtesting. The proposed agent (light green) leads to significantly better trading position than the "Trading RL" agent (dark green). Figure best viewed in color.

5. CONCLUSIONS

A novel price trailing approach, that goes beyond traditional price forecasting and RL techniques for trading, was proposed in this paper. The proposed method employs a novel price trailing formulation, where the RL agent is trained to trail the price of an asset, instead of directly predicting the future price or deciding whether or not a trade must be performed. This also allows for effectively overcoming the limitations of existing methods. The proposed method can be readily adapted to different applications by tuning the margin used during the training, effectively controlling the amount of noise that the agent can withstand. That is, using a larger margin allows for developing stable agents that are more robust to price fluctuations, while using a smaller margin allows for training agents that more closely follow the

price. The proposed method was evaluated using a real dataset that contains foreign exchange rates and it was demonstrated that it can indeed perform better than a competitive deep RL method.

6. REFERENCES

[1] Paul Wilmott, *Paul Wilmott on quantitative finance*, John Wiley & Sons, 2013.

[2] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha, "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques," *Expert Systems with Applications*, vol. 42, no. 1, pp. 259–268, 2015.

[3] Rodolfo C Cavalcante, Rodrigo C Brasileiro, Victor LF Souza, Jarley P Nobrega, and Adriano LI Oliveira, "Computational intelligence and financial markets: A survey and future directions," *Expert Systems with Applications*, vol. 55, pp. 194–211, 2016.

[4] Matthew Dixon, Diego Klabjan, and Jin Hoon Bang, "Classification-based financial markets prediction using deep neural networks," *Algorithmic Finance*, pp. 1–11, 2016.

[5] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Proceedings of the IEEE Conference on Business Informatics (CBI)*, 2017, pp. 7–12.

[6] Wei Bao, Jun Yue, and Yulei Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLOS ONE*, vol. 12, no. 7, pp. 1–24, 07 2017.

[7] 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. 5-6, pp. 441–470, 1998.

[8] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2017.

[9] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2017, pp. 2511–2515.

[10] Hado Van Hasselt, Arthur Guez, and David Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 2, p. 5.

[11] Adamantios Ntakaris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Mid-price prediction based on machine learning methods with technical and quantitative indicators," *SSRN*, 2018.

[12] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Temporal bag-of-features learning for predicting mid price movements using high frequency limit order book data," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.

[13] Kyoung-jae Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.

[14] Dong-xiao Niu, Hui-feng Shi, and Desheng Dash Wu, "Short-term load forecasting using bayesian neural networks learned by hybrid monte carlo algorithm," *Applied Soft Computing*, vol. 12, no. 6, pp. 1822–1827, 2012.

[15] Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[16] John Moody and Matthew Saffell, "Learning to trade via direct reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.

[17] John E Moody and Matthew Saffell, "Reinforcement learning for trading," in *Proceedings of the Advances in Neural Information Processing Systems*, 1999, pp. 917–923.

[18] Subhransu Maji and Alexander C Berg, "Max-margin additive classifiers for detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2009, pp. 40–47.

[19] Xinggang Wang, Baoyuan Wang, Xiang Bai, Wenyu Liu, and Zhuowen Tu, "Max-margin multiple-instance dictionary learning," in *Proceedings of the International Conference on Machine Learning*, 2013, pp. 846–854.

[20] Andrew Y Ng, Daishi Harada, and Stuart Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the International Conference on Machine Learning*, 1999, pp. 278–287.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015.