Algorithmic Currency Trading based on Reinforcement Learning Combining Action Shaping and Advantage Function Shaping

Hongyong Sun, Nan Sang, Jia Wu, Chen Wang
School of Information and Software Engineering
University of Electronic Science and Technology of China
Chengdu, China
sunhongyong0@gmail.com; sn@uestc.edu.cn; jiawu@uestc.edu.cn; lv00533@163.com

***Abstract*—This paper investigates high frequency currency trading with neural networks trained via Reinforcement Learning. A neural network-based agent is proposed to learn the temporal pattern in data and automatically trades according to the current market condition and the historical data. We propose two techniques: action shaping and advantage function shaping, to improve the total profit. The action shaping is used to avoid the agent outputting illegal actions since we assume that the agent trades fixed position sizes in a single security. The advantage function shaping is proposed to increase the probability of actions that lead to more profit. The proposed system has been back-tested on the currency market. The results demonstrate that our method performs well in most conditions.**

***Index Terms*—Neural networks for finance, Reinforcement learning, Deep learning**

I. Introduction

Algorithmic trading is essentially a sequential decision-making problem. It involves two key steps: summarizing market conditions and executing optimal actions to maximize profits according to the current market conditions. Reinforcement learning (RL) is learning that map states to actionsso as to maximize the cumulative reward for agent's interaction with environment. Hence, RL has a great potential to be used to realize the algorithmic trading.

RL have achieved a great success in a number of task areas, including robots navigation [1], atari game playing [2], and helicopter control [3]. It even outperforms human experts in some tasks [4], [5]. However, compared with conventional RL tasks, the algorithmic trading is much more difficult due to the following challenges.

The first challenge comes from the temporal correlation of trading action execution. The action space of algorithmic trading is {buy, hold, sell}. If the agent is not in the neural position, it can take any action; otherwise, if the agent is in the long or short position, it can only sell or buy to clean trading position, or keep in the neutral position. The legality of actions depends on the state of the trading position.

The second challenge arises from the assignment of the reward or the advantage. A profitable position contains a series of actions, including buy, hold and sell. If we assignee the reward equally to the actions, i.e., all actions contribute equally to the ultimate profit, the probability of the action hold will increase. Since the number of a hold action is more than the buy and sell actions. Thus, the algorithmic trader will prefer to choose hold and keep the position unchanged for a long time, and miss the opportunity to make profits.

In order to solve the above issues, we have proposed a reinforcement learning method combining action shaping and advantage function shaping to realize an algorithmic trader. The trader consists of fully-connected neutral network, which can learn the hidden pattern from the input data. The trader is trained by reinforcement learning algorithm to learn how to make profit as much as possible. In order to prevent the trader from choosing illegal actions, we discipline the actions depending to the state of a position. In addition, we shape the advantage function to improve the probability of flipping trading position, which can make more profit.

The remaining parts of this paper are organized as follows. Section II generally review some works about RL. Section III introduces the detail of the proposed algorithm. Section IV is the experiment part to test our method. Section V concludes this paper and gives some future remarks.

II. Related Work

Reinforcement learning (RL) is an effective framework for solving decision-making problem through trial-and-error interactions with a dynamic environment. RL algorithms include Mode-Free RL and Mode-Based RL. For the problem of the algorithmic trading, a ground-truth model of the environment is not available to the agent. Hence, model-free RL is used here. There are two main approaches to train agents with model-free RL: Policy Optimization and Q-Learning [6].

While the Q-Learning methods perform well for a number of problems, it is not a good way to solve the trading problem, as indicated in [7] and [8]. This is because the trading environment is too complex to estimate the value of the state. While Q-Learning methods are plausible for the offline scheduler problems [9], they are not ideal for dynamic online trading [7], [10]. Accordingly, rather than learning the value functions, a pioneering work [7] suggests learning the actions directly that falls into the policy optimization framework.

2375-0197/19/$31.00 ©2019 IEEE
DOI 10.1109/ICTAI.2019.00212

Deep reinforcement learning (DRL) [11] integrates the advantages of the perception of deep learning (DL) and the decision making of reinforcement learning, and gains the output control directly based on raw inputs by the end-to-end learning process.

III. Algorithmic Currency Trading Based on Reinforcement Learning

A. Problem Definition

In this paper, the algorithmic traders are assumed to take only long, neutral, or short positions of *constant* magnitude. The methods described here can be generalized to trade varying quantities of a security, allocate assets continuously or manage multiple asset portfolios.

To verify our work, we will back-test our trading policy, the following two assumptions are imposed for the model learning and the back-test experiments:

1) Zero slippage: the liquidity of all market assets is high enough that, each trade can be carried out immediately at the closing price when a order is placed.
2) Zero market impact: the capital invested by the trading agent is so insignificant that is has no influence on the market.

In a real-world trading environment, if the trading volume in a market is high enough, these two assumptions are near to reality.

For an automatic trading agent, these investment decisions are made periodically. The real-time trading action (policy) is defined as:

$$a_t \in \{buy, hold, sell\} = \{1, 0, -1\} \tag{1}$$

Similarly, the trading position of the agent is:

$$F_t \in \{long, neutral, short\} = \{1, 0, -1\} \tag{2}$$

F_t corresponds to a position (the state) of the agent, which is established or maintained at the end of each time t, and is reassessed at the end of $t+1$. A trade is thus possible at the end of each time period, although nonzero transaction cost will discourage excessive trading.

Suppose that $p_1, p_2, \ldots, p_t, \ldots$ denote the closing price sequences released from the exchange center. Then, the change of closing price at time point t is easily determined by $z_t = p_{t+1} - p_t$. The reward r_t made by the trading agent at time t is calculated by:

$$r_t = a_t z_t - c|F_{t+1} - F_t| \tag{3}$$

In Equation (3), the first term is the profit/loss made from the fluctuations of the prices and the second term is the transaction cost. If $F_{t+1} = F_t$, there is no transaction cost since the position during t and $t+1$ is held. Otherwise, the transaction cost is imposed according to the difference in shares held.

The goal of the agent is to maximize the ultimate profit. The optimization function is defined as the accumulated profits in the period of $1, \ldots, T$:

$$\max J(\theta) = \max_{\theta} \sum_{t=1}^{T} r_t \tag{4}$$

A policy π is a rule used by the agent to decide what actions to take.

$$a_t \sim \pi_\theta(.|s_t) \tag{5}$$

In Equation (5), the state s_t fed to the agent at time t includes F_t and X_t, where X_t denotes the feature vector of the current market condition at time t and it consists of the raw information of the market, such as the open, high, low, close of the price and the volume, and the selected technique indicators (see Table II).

The structure of the agent is presented in Fig. 1. The agent is constructed by a fully-connected neural networks. It is noted that the trading position F_t is not only fed to the input layer, but also fed to other hidden layers. The reason of this treatment is that on the one hand, the trading position is critical to the decision of the agent; on the other hand, the dimension and the range of values of X_t is larger than F_t, and the information of F_t can be easily neglected. Therefore, to strengthen the role of F_t, we feed it to other hidden layers.

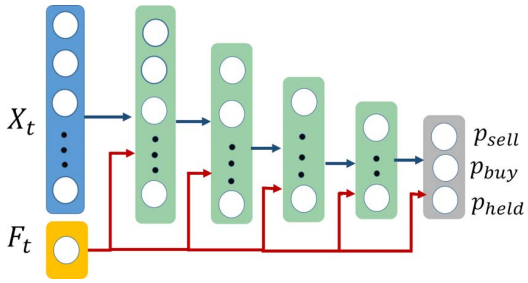


Fig. 1: Structure of the agent: s_t includes the the current market state X_t and the trading position F_t

B. Action shaping

As we have explained in the previous part, the decision of the action depends on the trading position. In order to avoid outputting illegal actions, we introduce a technique of action shaping to revise the design of the action, according to the position state of the agent. The shaped action is defined as:

$$a_t = \begin{cases} \{1, 0, -1\} & \text{if } F_t = 0 \\ \{0, -1\} & \text{if } F_t = 1 \\ \{1, 0\} & \text{if } F_t = -1 \end{cases} \tag{6}$$

C. Advantage shaping

In RL, the advantage function is usually used to evaluate the performance of an action:

$$\nabla_\theta(J(\theta)) = \nabla_\theta \sum_{t=1}^{T} \hat{A}_t \log \pi(a_t|s_t;\theta) \tag{7}$$

In Equation (7), $\hat{A}_t$ represents an estimate of the Advantage Function by state-action pairs (s_t, a_t), which is usually constructed as follows:

$$\hat{A}_t^{\gamma} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots - V(s_t) \tag{8}$$

In Equation (8), $\gamma \in [0, 1]$ represents the factor of discount.

The uncertainty of state-action pairs is usually measured by the number of visits to empirical samples. When a state is visited less times, its novelty is higher. Therefore, more exploration rewards need to be allocated to the state.

In the algorithmic trading, since the number of flipping trading positions is generally less than that of the neutral position, the probability of establishing long or short position will decrease as the policy optimization proceeds, which will lead the agent keep the neural position for a long time, thus loses the opportunity to gain profits.

In order to solve the above issue, we have proposed a technique to shape the advantage function to encourage the decision of establishing long or short positions. In this paper, we propose four advantage functions, which are constructed as follows:

$$\hat{A}_t^1 = \begin{cases} \sum_{i=t_e}^{t_c} r_i & \text{if } t = t_e \text{ or } t_c \\ r_t & \text{if } t \in (t_e, t_c) \end{cases} \tag{9}$$

Where t_e and t_c denotes, respectively, the time of establishing and closing a position. $\sum_{i=t_e}^{t_c} r_i$ is the total profit of a position. It is noted that the advantage function value is large at the time t_e and t_c. In this way, the probability of establishing a position will be improved. In order to better understand the new advantage function, we draw a picture to illustrate the advantage at each time step (Figure 2). For example, a position is established at $t = 1$ and closed at $t = 5$. The total profit made by this position is $\sum_{i=1}^{5} r_i$. To increase the probability of establishing a position, the advantage function values at $t = 1$ and $t = 5$ are set to the total profit.

$$\hat{A}_t^2 = \begin{cases} \sum_{i=t_e}^{t_c} r_i & \text{if } t = t_e \text{ or } t_c \\ \mu \sum_{i=t_e}^{t_c} r_i & \text{if } t \in (t_e, t_c) \end{cases} \tag{10}$$

Where μ is a discount factor, $\mu \in (0, 1)$. The advantage value is same for each action between t_e and t_c.

$$\hat{A}_t^3 = \sum_{i=t_e}^{t_c} r_i \tag{11}$$

The fourth advantage function is designed as follows:

$$\hat{A}_t^4 = \sum_{i=t}^{t_c} r_i \tag{12}$$

The advantage function value is a cumulative rewards from t to the end of a position.

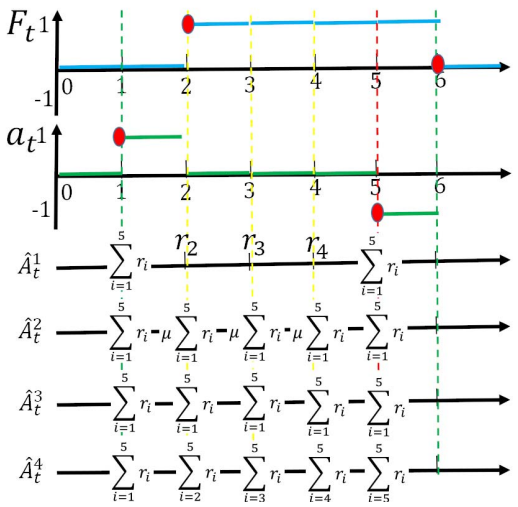


Fig. 2: Relationship between a_t, F_t and the way to obtain$\hat{A}_t$

IV. EXPERIMENTAL VERIFICATIONS

A. Experimental Setup

In this section, we test our trading model on the real-world financial data. We use the currency future data to test. Here, we select the EUR/USD contract.

The financial data are downloaded from OANDA API [12]. In our experiment, the minute-level prices are used, implying that there is a 1-min interval between the price p_t and p_{t+1}. The historical data of three different periods of EUR/USD currency futures are shown in Figure 3 (a)-(c).

From Fig. 3(a)-(c), we can see that these three different periods exhibit quite different market patterns. Fig. 3(a) presents very large upward movements. Fig. 3(b) has both very large upward and downward movements. Fig. 3(c) shows a downward trend. For practical usage, some other issues related to trading should also be considered. For example, the transaction cost charged by the brokerage company is also provided. The transaction cost of EUR/USD currency future is set to 0.0002.

The raw price and indicators of the last 16 min and current positions state is directly used as the input of the trading system $s_t \in R^{659}$. The input layer is sequentially passed through four deep transformation layers with 512, 256, 128, and 64 hidden nodes per layer. The feature representation $f_t \in R^{67}$ of the final deep layer is connected with the output layer for trading policy making.

B. Details on Model Training and Back-test

In this part, we introduce briefly the training details and the back-test details. The agent is trained via online updating. The system is exploited to trade the time points from 1 to 6000. Then, the sliding window of the training data is moved 6000 ticks forward covering a new training set from 6001 to 12000. With the sliding window of the training set moving ahead, we back-test trading polity.

TABLE I: Combinations of models combining action shaping and advantage functions

Models	Action shaping	Advantage Function shaping
Action shaping + Advantage function shaping 1 **(AS-AFS1)**	✓	Equation (9)
Action shaping + Advantage function shaping 2 **(AS-AFS2)**	✓	Equation (10)
Advantage function shaping 3 (**AFS3**)	✗	Equation (11)
Action shaping + Advantage function shaping 3 (**AS + AFS3**)	✓	Equation (11)
Advantage function shaping 4 (**AFS4**)	✗	Equation (12)
Action shaping + Advantage function shaping 4 (**AS-AFS4**)	✓	Equation (12)

C. Experimental Evaluations

We evaluate different models on the practical data. These models are combinations of different techniques, such as with or without the action shaping, four different advantage functions. The detailed information about the combinations of models are summarized in Table I.

The performance of different models are measured by four criteria:

- Total profits (TP)
- Sharp ratio (SR)
- Total number of trading positions (TT)
- Total number of profitable trading positions (TPT)

All the models are trained with policy gradient in Equation (7). The performances of all methods are summarized in Table III. The details of the performance, in terms of the profit and loss (P&L), during training and the back-test, are presented in Figure 3. The quantitative evaluations are summarized in Table III, where the testing performances are also reported in the forms of total profit (TP) and sharp ratio (SR). From the experimental results, three observations can be found as follows.

The first observation is that all methods have gained more profits in different market trends, the most profitable one is in the upward trend market, the second in the downward trend market, and the least profitable in the upward and downward movements market. Since the agent can buy or sell, it can make profit in different market patterns.

Second, the action shaping technique indeed helps to gain more profits, even in different market trends.

Third, the advantage function shaping technique can help the agent make more profit. The model that using advantage function with Equation (9) achieves the best performance, and using the advantage function with Equation (10) achieves the second best performance. The model with the advantage shapingEquation (9) obtains more reliable performance since Equation (9) puts more advantage at the actions of buy and sell, that encourages the agent to trade. This reliability can be observed from Table III and the shape of P&L curve in Figure 3.

In conclusion, the proposed model can make reliable profits in different markets by using action shaping and advantage function shaping.

V. CONCLUSION

This paper proposed a reinforcement learning method combining the action shaping and the advantage function shaping to implement the algorithmic currency trading. In this framework, a RL agent can automatically sense the dynamics of the market and alleviate the difficulty of manually designing indicators from massive data. Moreover, we propose two techniques: action shaping and advantage function shaping to improve the total profit. The action shaping tries discipline the output of the action based on the trading position. The advantage function shaping can improve the probability of flipping trading position, which can make more profit. We have tested our model in real data. The experiment results show the effectiveness of the model on EUR/USD currency data. In the future, we will extend our model to test more currencies and in more complex market to verify the generality of it, whether the performance varies widely for different currency markets.

REFERENCES

[1] Beom, Hee Rak, and Hyung Suck Cho. "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," IEEE transactions on Systems, Man, and Cybernetics 25.3 (1995): 464-477.

[2] Mnih, Volodymyr, et al."Human-level control through deep reinforcement learning," Nature 518.7540 (2015): 529.

[3] Kim, H. Jin, et al. "Autonomous helicopter flight via reinforcement learning," Advances in neural information processing systems. 2004.

[4] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[5] Song, Yong-Duan, Qi Song, and Wen-Chuan Cai. "Fault-tolerant adaptive control of high-speed trains under traction/braking failures: A virtual parameter-based approach," IEEE Transactions on Intelligent Transportation Systems 15.2 (2013): 737-748.

[6] G.Tesauro.TD-Gammon. "a self-teaching backgammon program, achieves master-level play," Neural Comput., vol. 6, no.2, pp.215-219, 1994.

[7] Moody, John, and Matthew Saffell. "Learning to trade via direct reinforcement," IEEE transactions on neural Networks 12.4 (2001): 875-889.

[8] Dempster, Michael AH, and Vasco Leemans. "An automated FX trading system using adaptive reinforcement learning," Expert Systems with Applications 30.3 (2006): 543-552.

[9] Tesauro, Gerald. "TD-Gammon, a self-teaching backgammon program, achieves master-level play," Neural computation 6.2 (1994): 215-219.

[10] Deng, Yue, et al. "Sparse coding-inspired optimal trading system for HFT industry," IEEE Transactions on Industrial Informatics 11.2 (2015): 467-475.

[11] Mnih V, Kavukcuoglu K, Silver D, et al,"Human-level control through deep reinforcement learning," Nature, vol.518, no.7540, pp.529-533, 2015.

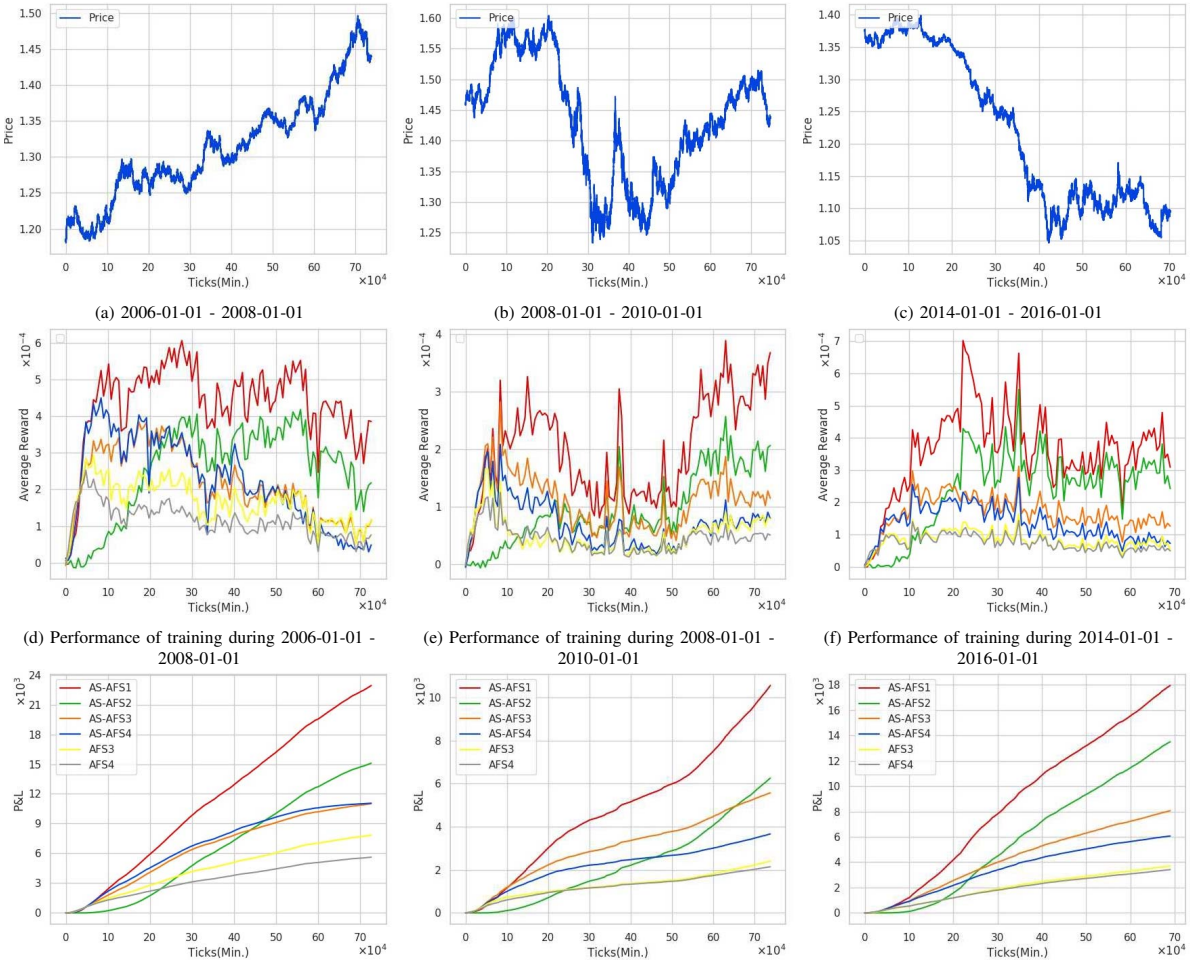
[12] https://api-fxpractice.oanda.com

TABLE II: Financial technical indicators

Bollinger Bands	Exponential Moving Average	Double Exponential Moving Average	Moving Average Convergence/Divergence
Momentum	Directional Movement Index	Kaufman Adaptive Moving Average	Hilbert Transform - Dominant Cycle Phase
Balance Of Power	Money Flow Index	Hilbert Transform - Phasor Components	Hilbert Transform - Instantaneous Trendline
Stochastic	Relative Strength Index	Average Directional Movement Index	Hilbert Transform - Trend vs Cycle Mode
Stochastic Fast	Chaikin A/D Oscillator	Stochastic Relative Strength Index	Chaikin A/D (Accumulation/Distribution) Line
True Range	Average True Range	Normalized Average True Range	Hilbert Transform - Dominant Cycle Period
Moving average	On Balance Volume	Hilbert Transform - SineWave	

TABLE III: Performances of different models on different market patterns

	2006-01-01 - 2008-01-01				**2008-01-01 - 2010-01-01**				**2014-01-01 - 2014-01-01**			
	TP	SR	TT	TPT	TP	SR	TT	TPT	TP	SR	TT	TPT
AS-AFS1	22921.94	1.20	980900	878150	10548.76	1.03	214194	175090	17941.08	0.96	57815	53977
AS-AFS2	15094.77	0.89	159734	125555	6253.60	0.54	68292	42500	13503.68	0.57	87726	63736
AS-AFS3	11010.49	1.10	108607	107043	5568.62	0.98	20680	20406	8059.18	0.62	5748	5735
AS-AFS4	11061.28	1.08	145332	141234	3660.66	0.98	12500	12142	6066.13	0.66	10939	10905
AFS3	7840.44	1.02	64504	63566	2397.93	1.00	2179	2144	3700.15	0.64	3052	3043
AFS4	5617.79	0.95	34904	34445	2136.64	1.09	1403	1315	3410.20	1.19	347	342



(a) 2006-01-01 - 2008-01-01 (b) 2008-01-01 - 2010-01-01 (c) 2014-01-01 - 2016-01-01

(d) Performance of training during 2006-01-01 - 2008-01-01 (e) Performance of training during 2008-01-01 - 2010-01-01 (f) Performance of training during 2014-01-01 - 2016-01-01

(g) Back-test during 2006-01-01 - 2008-01-01 (h) Back-test during 2008-01-01 - 2010-01-01 (i) Back-test during 2014-01-01 - 2016-01-01

Fig. 3: Training and Back-testing models on there data (top). The R&L curves of different trading systems with total profits are presented.