In Proceedings of International Conference on Neural Information Processing, ICONIP 2000, Taejon, Korea
pp 832 - 837, 2000.

An Algorithm for Trading and Portfolio Management Using Q-learning and Sharpe Ratio Maximization

Xiu Gao

Department of Computer Science
and Engineering
The Chinese University of HongKong
Shatin, HongKong
xgao@cse.cuhk.edu.hk

Laiwan Chan

Department of Computer Science
and Engineering
The Chinese University of HongKong
Shatin, HongKong
lwchan@cse.cuhk.edu.hk
http://www.cse.cuhk.edu.hk/~lwchan

Abstract

A trading and portfolio management system called QSR is proposed. It uses Q-learning and Sharpe ratio maximization algorithm. We use absolute profit and relative risk-adjusted profit as performance function to train the system respectively, and employ a committee of two networks to do the testing. The new proposed algorithm makes use of the advantages of both parts and can be used in a more general case. We demonstrate with experimental results that the proposed approach generates appreciable profits from trading in the foreign exchange markets.

1 Introduction

Nowadays billions of dollars are daily pushed through the international capital markets while traders want to shift their investments to more promising assets. It will be very useful if given some information a machine can assist an investor by correctly indicating the trading actions. This is the goal of trading and portfolio management.

Trading and portfolio management is the investment of liquid capital to various trading opportunities like stocks, futures, foreign exchanges and others. In recent years, the application of artificial intelligence techniques for trading and portfolio management has experienced significant growth. Many trading systems have been proposed based on different methodologies and investment strategies.

There are mainly two kinds of approaches to optimizing trading systems. They are trading based on forecasts [1, 2, 3] and trading based on labelled data [4, 5] . The former one consists of two modules: a prediction module followed by a trading module. First, it predicts the price value at some time in the future from historical data. Some prediction criterions such as the minimization of the mean square error (MSE) are used. Then, a trading module is employed to produce a trading signal based on the prediction and some investment strategy. Since this type of trading system is optimized to a prediction criterion that is poorly correlated with the ultimate measure of performance of the system, it usually leads to sub-optimal performance. The latter one is training a trading system on labelled data. It also contains two parts. First, a labelling procedure produces a sequence of desired target trades used for training the system according to some measurement strategy. Then, the trading module is trained on the labelled data using a supervised learning approach. The ultimate performance of the system depends on how good the labelling algorithm is, and how well the trading module can learn to trade using the input variables and labelled trades. Since the ultimate measure of performance is not used to optimize the trading system parameters directly, performance of such a system is thus likely to be sub-optimal.

One alternative to the above approach is to optimize a trading system by using reinforcement learning. The ultimate measure of performance can be used directly to optimize the trading system. Neuneier [6] used Q-learning to train an asset allocation system to maximize profit. Transaction costs are included in his strategy. But in his analysis, it is assumed that the investor has no risk aversion, therefore, the goal is to maximize profit (return). When in an environment with high-level risk, the system trained by maximizing profit can not work well. Actually, the problem of portfolio optimization can be viewed as the finding of a desirable combination of risks and returns [7].

In this paper, we propose a hybrid of Q-learning and Sharpe ratio maximization algorithm for trading and portfolio management (QSR). We use absolute profit and relative risk-adjusted profit (Sharpe ratio) as performance function to train the system respectively, and employ a committee of two networks to do the testing. The new proposed algorithm makes use of the advantages of both parts. Here, we demonstrate its utility for trading in the foreign exchange markets. Experimental results show that the new algorithm generates appreciable profits.

2 Reinforcement Learning and Q-learning

Reinforcement learning deals with sequential decision making tasks in which an agent needs to perform a sequence of actions to reach some goal states. Compared with supervised learning which requires of input/target pairs, reinforcement learning learns behavior through trial-and-error interactions with a dynamic environment. The interaction takes the form of the agent sensing the environment, and based on this sensory input choosing an action to perform in the environment. The action changes the environment in some manner and this change is communicated to the agent through a scalar reinforcement (reward) signal, and the signal measures how good the action performed on the current state is.

A commonly-used criterion to maximize the future reward is the cumulative discounted reward,

$$\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}), \tag{1}$$

where $0 \leq \gamma < 1$ is a discount factor that favors reinforcement received sooner relative to that received later.

The agent's job is to find a policy π , which delivers for every state an admissible action. One can compute the value function, denoted V^π of a given policy, by

$$V^\pi(s) = E_\pi\{r_t | s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}, \tag{2}$$

The optimal value function V^* is the unique solution of the well-known Bellman equation [8]. Given the optimal value function, we can specify the optimal policy as

$$\pi^*(s) = \arg\max_a \left(r(s,a) + \gamma \sum_{s' \in S} P_{ss'}(a) V^*(s') \right) \tag{3}$$

Q-learning [9] is a widely used RL algorithm. The key to Q learning is to replace the value function $V(s)$ with an action-value function, $Q(s,a)$. The quantity $Q(s,a)$ gives the expected cumulative discounted reward of performing action a in state s and then pursuing the current policy thereafter.

We can write down the Q version of the Bellman equation as follows:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} P_{ss'}(a) \max_{a'} Q^*(s', a') \tag{4}$$

which suggests the successive approximation recursion:

$$Q^{(k+1)}(s,a) = r(s,a) + \gamma \sum_{s'} P_{ss'}(a) \max_{a'} Q^{(k)}(s', a') \tag{5}$$

Neuneier [6] used Q-learning to train an asset allocation system to maximize profit. To do that, he defined the immediate reward of a state/action pair as the absolute return gained when perform the selected action.

3 Trading Strategy Based on the Sharpe ratio

A trader ideally prefers very high profit (return) with very low risk (variability of return). In other words, it is the finding of a desirable combination of risks and returns. One way of representing the tradeoff between profit and risk is to use risk-adjusted performance. One widely-used measure of risk-adjusted return is the Sharpe ratio [10].

The Sharpe ratio (SR) takes the mean of return (profit) and divides it by the standard deviation of the return. The standard deviation of return quantifies the risk.

$$SR = \frac{average\ \ return}{standard\ deviation\ of\ \ return}. \tag{6}$$

We can train trading system by maximizing the Sharpe ratio. Here, the Sharpe ratio takes the form of performance function. Define x_t to be the price of the asset on day t, and r_t to be the relative returns on day t.

$$r_t = ln(x_t) - ln(x_{t-1}) \approx \frac{x_t}{x_{t-1}} - 1. \tag{7}$$

Additionally, define the asset returns (that includes the portfolio weight α_t) as

$$R_t = \alpha_t * r_t. \tag{8}$$

where the portfolio weight α_t takes on continuous value between $[0, 1]$.

The average daily return is given by

$$\overline{R} = \frac{1}{N} \sum_{t=1}^{N} R_t \tag{9}$$

The standard deviation of the return is given by

$$\sigma = \left(\frac{1}{N-1} \sum_{t=1}^{N} (R_t - \overline{R})^2 \right)^{\frac{1}{2}}. \tag{10}$$

Then

$$SR = \frac{\overline{R}}{\sigma}. \tag{11}$$

Choey and Weigend [11] used a supervised learning method to train trading system by optimizing the Sharpe ratio. For simplicity of computation, in this paper we use a similar but variant method. The Sharpe ratio can be viewed as function of the asset prices and portfolio weights:

$$SR = F(x_0, x_1, \ldots, x_N, \alpha_1, \ldots, \alpha_N). \quad (12)$$

We can get desired target portfolio weights by maximizing the Sharpe ratio, and using a supervised learning algorithm to train the trading system on the input/ desired target data.

The desired target can be learned in batch mode by repeatedly computing the value of SR on forward passes through the data and adjusting the desired target by using gradient ascent (with learning rate ρ)

$$\Delta\alpha_t = \rho \frac{\partial SR}{\partial \alpha_t}, \qquad t = 1, \ldots, N. \quad (13)$$

To update the portfolio weights with gradient ascent, we now need to calculate the gradient by taking the partial derivative of the SR with respect to the weights, α_t

$$\frac{\partial SR}{\partial \alpha_t} = \frac{1}{\sigma}\frac{\partial \overline{R}}{\partial \alpha_t} - \frac{\overline{R}}{\sigma^2}\frac{\partial \sigma}{\partial \alpha_t}. \quad (14)$$

$$\frac{\partial \overline{R}}{\partial \alpha_t} = \frac{1}{N}\frac{\partial R_t}{\partial \alpha_t} = \frac{1}{N} r_t \quad (15)$$

$$\frac{\partial \sigma}{\partial \alpha_t} = \frac{1}{2\sigma}\frac{\partial \sigma^2}{\partial \alpha_t} \quad (16)$$

$$\frac{\partial \sigma^2}{\partial \alpha_t} = \frac{1}{N-1}\frac{\partial \sum_{t=1}^{N}(R_t - \overline{R})^2}{\partial \alpha_t} \quad (17)$$

$$= \frac{2}{N-1}\left(R_t r_t - \frac{N-1}{N}\overline{R} r_t\right). \quad (18)$$

Substituting equation (15), (16), (17) and (18) into equation (14), we have

$$\frac{\partial SR}{\partial \alpha_t} = \frac{r_t}{\sigma N}\left(1 - \frac{N}{\sigma^2 (N-1)}\overline{R} R_t + \frac{\overline{R}^2}{\sigma^2}\right). \quad (19)$$

4 QSR — A Hybrid of Q-learning and Sharpe Ratio Maximization Algorithm

From the above analysis, we know that both Q-learning method and Sharpe ratio maximization technique can be used to deal with trading and portfolio management problem. It is clear that using risk-adjusted return which is measured by Sharpe ratio is more suitable than using profit to measure the performance of a trading system. But the Sharpe ratio can only be gained after a complete sequence of trades, it is a future reward, and in that kind of method, if we take into account transaction costs, it is not avoidable to use recurrent structures. While in Q-learning, we use temporal difference method to assign immediate reward. It can be obtained after an individual action is performed. And when we use Q-learning, it is convenient to take into account transaction costs. The above analysis motivates us to combine these two kinds of methods. That is to say,we use absolute profit and relative risk-adjusted profit (Sharpe ratio) as performance function to train the system by the two methods respectively, then we employ a committee of two networks to do the testing. The QSR algorithm can make use of the advantages of both parts and can be used in a more general case.

In the Q-learning stage, we use the definition of state and immediate reward function proposed by Neuneier [6]. In this problem, the state vector, s_t, is the triple of the exchange rate, x_t, the wealth of the portfolio, c_t, and a binary variable b_t, which represents the fact that currently the investment is in DM or USD. There are two available actions (decisions), investing in DM ($a_t = 0$) or USD ($a_t = 1$). The transaction costs are $\xi_t = 0.5\% * c_t$. Transactions only apply, if the currency is changed from DM to USD. The immediate reward r_t is computed as in table 1.

	a_t =DM	a_t =USD
b_t =DM	$r_t = 0$	$r_t = (x_{t+1}/x_t)(c_t - \xi_t) - c_t$
b_t =USD	$r_t = 0$	$r_t = (x_{t+1}/x_t - 1)c_t$

Table 1: The immediate reward function.

The Q function is represented by feed-forward neural network and learned using a combination of temporal difference method and the error back-propagation algorithm. Temporal difference method computes the error between temporally successive predictions and the back-propagation algorithm minimizes the error by modifying the weights of the network.

Then the procedure of learning the Q function can be sketched as follows:

1. Observe the current state s, for each action a_i, compute $Q(s, a_i)$.

2. Use some action-selection criterion to select an action a.

3. Perform action a on state s, observe the resulting state s' and calculate immediate reward r.

4. Compute $Q' = r + \gamma \max_{a'} Q(s', a')$.

5. Adjust network by back-propagating error ΔQ

$$\Delta Q_i = \begin{cases} Q' - Q(s, a_i) & \text{if } a_i = a; \\ 0 & \text{otherwise.} \end{cases}$$

6. Go to 1.

The initial Q values, $Q_0(s, a_i)$ for all states and actions are assumed given. The network described above has multiple outputs (one for each action). In practice, we use multiple networks (one for each action); each with a single output. The former implementation should be less desirable, because whenever the single network is modified with respect to an action, no matter whether it is desired or not, the network is also modified with respect to the other actions as a result of shared hidden units between actions.

After we have finished training the networks, the agent's policy is, given a state s, to choose the action a for which $Q(s, a)$ is maximal.

The action-selection mechanism used in the algorithm is a widely-used method, proposed by Watkins [9], suggesting to choose an action, a, via the Boltzman distribution:

$$Pr\{a = a_j\} = \frac{e^{Q(s,a_j)/T}}{\sum_j e^{Q(s,a_j)/T}} \tag{20}$$

where the Boltzman temperature, T, is initialized to a relatively high value, resulting in a uniform distribution for prospective actions. As computation proceeds, temperature is gradually lowered, in effect raising the probability of selecting actions with higher Q values.

In the Sharpe ratio maximization stage, we use the method demonstrated in the last section to train the trading system.

After finishing the training process according to the two kind of algorithms, we get a set of neural networks which represent the Q function. With it, the Q values of testing data can be obtained, then action (decision) a_t^Q can be made. We also get a trading system which can generate portfolio weights α_t for the testing data set. To produce the final action signal which combines the result of the two algorithms, we use the following method

$$\begin{cases} a_t = 1 & \text{if } (\beta_1 a_t^Q + \beta_2 \alpha_t) > 0.5; \\ a_t = 0 & \text{if } (\beta_1 a_t^Q + \beta_2 \alpha_t) \leq 0.5. \end{cases}$$

where β_1, β_2 are parameters that satisfy

$$\beta_i \geq 0, \quad i = 1, 2 \quad \text{and} \quad \beta_1 + \beta_2 = 1.$$

5 Experimental Results

We can demonstrate the usefulness of the new proposed algorithm by simulating trading in the foreign exchange markets. For simplicity, here we consider a single foreign exchange rate series of US

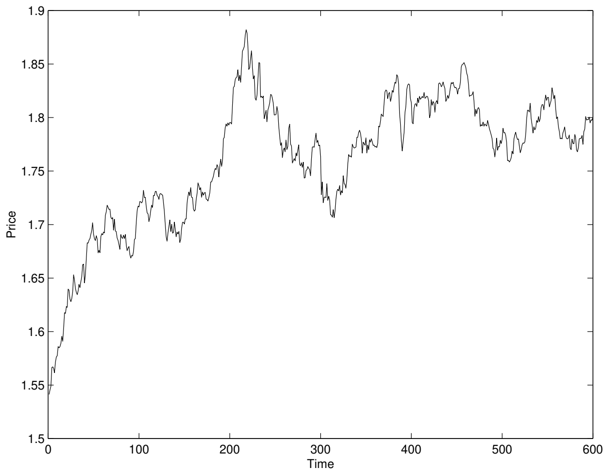

Figure 1: The USD-DM rate series. The first 500 data are used as training dataset and the remaining 100 points are used as testing dataset.

Dollar(USD) versus German Deutschmark (DM). The followings are our assumptions. DM is the basis currency. The profit gained is expressed in the basis currency. The investment is small and does not influence the whole market by the trading, and the investor always invests the whole amount of the asset. A total of 600 daily data are used, from January 1, 1997 through August 23, 1998, as shown in Figure 1. The first 500 data points are used as the training dataset, while the remaining 100 points are used in the testing stage. We assume that the transaction costs rate is 0.5% and transactions only apply, if the currency is changed form DM to USD.

To measure the performance of the proposed algorithm — QSR, we implement the following methods to simulate trading on the data mentioned above.

(A) Trading based on forecasts. Using the past four day's price to predict the price in the future, then employing the prediction and an investment strategy. (If the predicted price is 3% higher than the last day's, then invest in USD; if the predicted price is 3% lower than the last day's, invest in DM; otherwise, hold on the last investment.)

(B) Trading based on labelled data. Using optimizing Sharpe ratio to get labelled data, and using supervised learning to do trading.

(C) Q-leaning method as described by Neuneier [6].

(D) The proposed QSR algorithm.

Shown in Figure 2 to Figure 5 are the results using the above methods. Figure 2 shows the results using method (A)— trading based on forecasts. Profits gained using the trading signals indi-

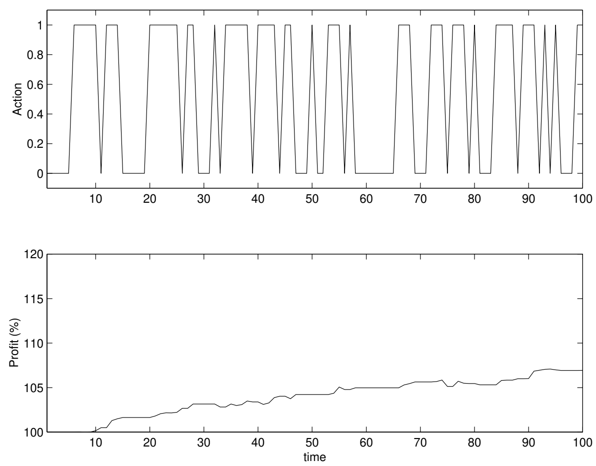

Figure 2: The results by method (A). Upper graph: the trading signal on the testing data. Lower graph: the profit gained (%).

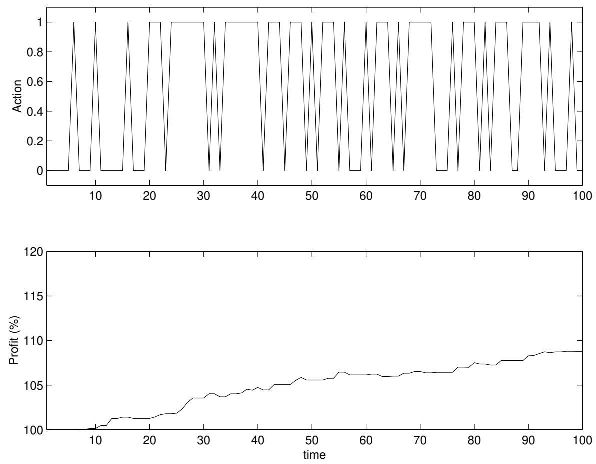

Figure 3: The results by method (B). Upper graph: the trading signal on the testing data. Lower graph: the profit gained (%).

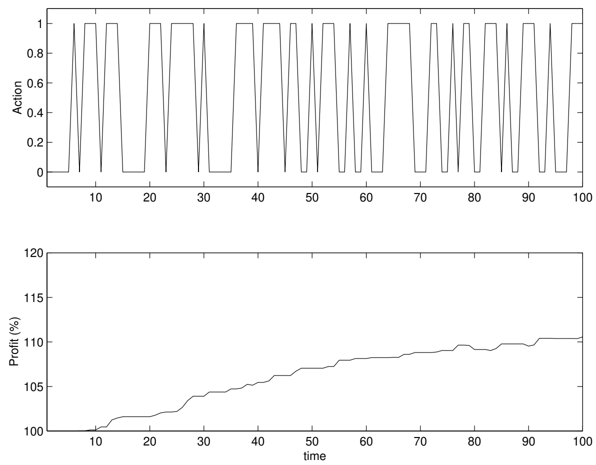

Figure 4: The results by method (C). Upper graph: the trading signal on the testing data. Lower graph: the profit gained (%).

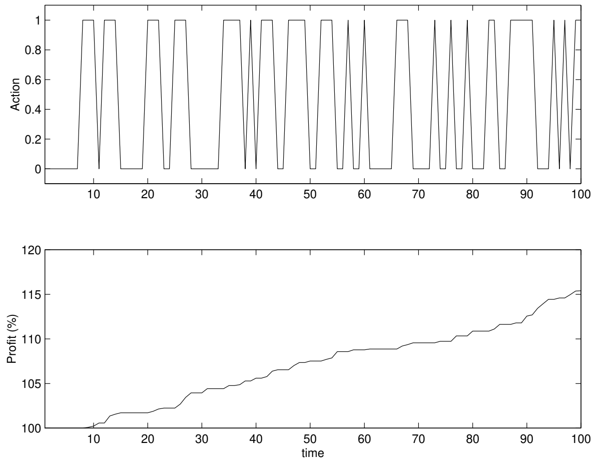

Figure 5: The results by method (D). Upper graph: the trading signal on the testing data. Lower graph: the profit gained (%).

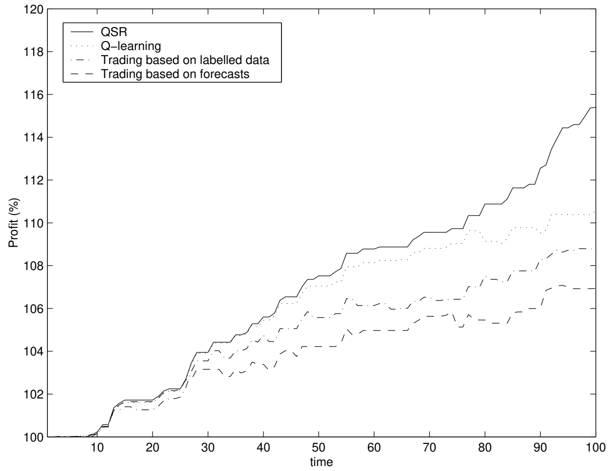

Figure 6: A plot of the results by all of the four methods.

cated in Figure 2(a) are shown in Figure 2(b). Similarly, shown in Figure 3, Figure 4 and Figure 5 are the results using trading based on labelled data, Q-learning method and the proposed QSR algorithm, respectively.

From these results, we find that the performance of our new proposed QSR algorithm outperforms the other three methods, and among them, Q-learning method outperforms trading based on forecasts and trading based on labelled data algorithms. The profits of method (A)-(D) are plotted together in Figure 6 for comparison.

6 Conclusion and Future Work

As an alternative to two kinds of conventional approaches to optimizing trading systems— trading based on forecasts and trading based on labelled data, the proposed algorithm combines Q-learning and Sharpe ratio maximization method, using absolute profit and relative risk-adjusted profit as performance functions for training respectively. Then the committee of the two networks is employed to do the testing. Based on the trading example using a foreign exchange rate, the profits obtained from the proposed system appear very promising, and thus, the techniques presented deserve further exploration.

The proposed method can be generalized to trading and portfolio management problem with multiple foreign exchanges easily by redefining the corresponding definition and expression according to multiple assets condition. However there is a limitation of the proposed approach, that is it can only deal with discrete action space, and in real world problems, many of them require continuous action space. To generalize the proposed algorithm to continuous action space is our future work.

Acknowlegement

The authors would like to thank The Research Grants Council, HK, for support of this project. We also thank Yingqian Zhang, SiuMing Cha and Xiaohui Yu for their helpful discussions.

References

[1] C. Granger, and P. Newbold. Forecasting Economic Time Series. New York: Academic Press, 1986.

[2] D. Montgomery, L. Johnson, and J. Gardiner. Forecasting and Time Series Analysis. New York: McGraw-Hill, 1990.

[3] R. R. Trippi, and E. Turban. Neural Networks in Finance and Investing. Chicago: Probos, 1993.

[4] L. Xu, and Y. M. Cheung. Adaptive Supervised Learning Decision Networks for Trading and Portfolio Management. *Journal of Computational Intelligence in Finance*, 11-16, 1997.

[5] Y. Bengio. Training a Neural Network with a Financial Criterion Rather than a Prediction Criterion. In Y. Abu-Mostafa, A. N. Refenes and A. Weigend (eds), *Decision Technology for Financial Engineering*, 36-48, London: World Scientific, 1997.

[6] R. Neuneier. Optimal asset allocation using adaptive dynamic programming. In K. Touretzky, M. Mozer and M. Hasselmo (eds),*Advances in Neural Information Processing Systems* 8, 953-958, Cambridge, MA: MIT Press, 1996.

[7] B. Van Roy. Temporal-Difference Learning and Applications in Finance. *Computational Finance (Proceedings of the Sixth International Conference on Computational Finance)* Edited by Y. S. AbuMostafa, B. LeBaron, A. W. Lo, and A. S. Weigend. Cambridge, MA: MIT Press, 1999.

[8] R. E. Bellman. Dynamic Programming. Princeton University Press, Princeton, 1957.

[9] C. J. Watkins. Learning from Delayed Rewards. Ph.D. thesis, Cambridge University, 1989.

[10] W. F. Sharpe. Mutual Fund Performance. *Journal of Business*, 119-138, 1966.

[11] M. Choey, and A. S. Weigend. Nonlinear Trading Models Through Sharpe Ratio Maximization. In Y. Abu-Mostafa, A. N. Refenes and A. Weigend (eds), *Decision Technology for Financial Engineering*, 3-22, London: World Scientific, 1997.