Algorithm Trading using Q-Learning and Recurrent Reinforcement Learning

Xin Du
duxin@stanford.edu

Jinjian Zhai
jameszjj@stanford.edu

Koupin Lv
koupinlv@stanford.edu

Abstract

The reinforcement learning methods are applied to optimize the portfolios with asset allocation between risky and riskless instruments in this paper. We use classic reinforcement algorithm, Q-learning, to evaluate the performance in terms of cumulative profits by maximizing different forms of value functions: interval profit, sharp ratio, and derivative sharp ratio.

Moreover, direct reinforcement algorithm (policy search) is also introduced to adjust the trading system by seeking the optimal allocation parameters using stochastic gradient ascent. We find that this direct reinforcement learning framework enables a simpler problem representation than that in value function based search algorithm, and thus avoiding Bellman's curse of dimensionality and offering great advantages in efficiency.

Key words: Value Function, Policy Gradient, Q-Learning, Recurrent Reinforcement Learning, Utility, Sharp Ratio, Derivative Sharp Ratio, Portfolio

1. Introduction

In the real world, trading activities is to optimize rational investors' relevant measure of interest, such as cumulative profit, economic utility, or rate of return. In this paper, we study the performance of Q-learning algorithm with different value functions subject to optimize: internal profit, sharp ratio, and derivative sharp ratio. Empirical results indicate that derivative sharp ratio outperform the other two alternatives by accumulating higher profit in the value iteration.

Due to the property of financial decision problems, especially when transaction costs are included, the trading system must be recurrent and immediately assessment of short-term performance becomes essential for the gradually investment allocations [1, 2]. Direct reinforcement learning approach is able to provide an immediate feedback to optimize the strategy. In this report, we apply an adaptive algorithm called Recurrent Reinforcement Learning (RRL) to achieve superior performance of collecting higher cumulative profit compare to the case of using Q-Learning[2~5].

Investment performance is path dependent if we use the cumulative profit as the criterion [6~8]. Optimal trading strategy and asset rebalancing decision require the information of current position of portfolio and market status [9, 10]. Besides, market imperfections like transaction costs and taxes will make the high frequency trading overwhelmingly expensive [11].

Our algorithm trading results indicate that RRL has more stable performance compared to the Q-learning when exposed to noisy datasets. Q-learning algorithm is more sensitive to the value function selection (perhaps) due to the recursive property of dynamic optimization, while RRL algorithm is more flexible in choosing objective function and saving computational time [6, 8, 10].

2. Portfolio and Trading System Setup and Performance Criterion

2.1. Structure of Portfolio

The most important cornerstone in 1960's in the field of finance theory is the establishment of Capital Asset Pricing Model (CAPM) [12, 13]. A very important conclusion of CAPM is the holding strategy of risky/riskless assets for the investors: assuming risk free asset (like cash or T-Bill) is uncorrelated with other assets; there is only one optimal portfolio that can achieve lowest level of risk for any level of return, which is market portfolio (Market portfolio is the weighted sum of all the risky assets within the financial market, and it is totally diversified for the risk) [14].

Our investment account is built up following the CAPM theory, which is the combination of a riskless asset (cash) and a risky market portfolio. The relative weights of these two assets will be rebalanced by trading within each time step. Figure 1 gives the intuitive way of presenting our transaction account, which is the combination of a riskless asset and a risky asset from the efficient frontier.

Our traders are assumed to take only long, short positions, $F_t \in \{1, -1\}$, of a certain magnitude. Long positions are initiated by purchasing some amount of assets, while short positions correspond to selling accordingly[15].

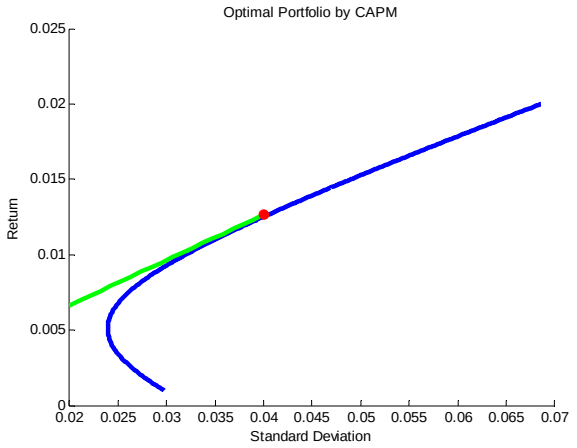


Figure 1. Construction of Trading Account: Combination of a Riskless Asset and a Market Portfolio

Within the market portfolio, there are m risky assets, and their prices at time t are denoted as: $z_t^i, i = 1, 2, ..., m$. The position for asset i, F_t^i, is established at the end of time step t, and will be rebalanced at the end of period t+1. Market imperfections like transaction costs and taxes are denoted as δ to incorporate into the decision function iteratively.

In a word, our decision function at time step t is defined as $F_t = G(\theta_t; F_{t-1}, I_t)$, in which, θ_t is the learned weights between risky and riskless assets, and I_t is the information filtration including security price series z and other market information y over the time up to t: $I_t = (z_t, z_{t-1}, ...; y_t, y_{t-1}, ...)$.

The F_t mentioned above denotes the asset allocation between risky and riskless assets in the portfolio at time t. Moreover, from time t-1 to time t, the risky asset, which is the market portfolio, also changes relative weights among all the securities because of their price/volume fluctuations. We also define the market return of asset i during time period t is: $r_t^i = (z_t^i / z_{t-1}^i - 1)$. At any time step, the weights of risky assets sum up to one:

$$\sum_{i=1}^{m} F^i = 1 \tag{1}$$

2.2. Profit and Wealth for the Portfolios

A natural selection of performance criterion is the cumulative profits of the investment accounts. In discrete time series, Additive Profits are applied for the accounts whose risky asset is in the form of standard contracts, such as future contracts of USD/EUR. While in our case, since we use market portfolio as the risky asset, with each security fluctuates all the time, we use Multiplicative Profits to calculate the wealth:

$$\begin{aligned} W_T &= W_0 \prod_{t=1}^{T} (1 + R_t) \\ &= W_0 \prod_{t=1}^{T} (1 + (1 - F_{t-1}) r_t^f + F_{t-1} r_t)(1 - \delta |F_t - F_{t-1}|) \end{aligned} \tag{2}$$

In which W_T is the accumulative wealth in our account up to time T, and W_0 is the initial wealth, r_t^f is the risk free rate, such as interest rate for cash or T-bills. When the interest rate r_t^f is ignored ($r_t^f = 0$), a simplified expression of cumulative wealth is obtained:

$$\begin{aligned} W_T &= W_0 \prod_{t=1}^{T} (1 + R_t) \\ &= W_0 \prod_{t=1}^{T} (1 + F_{t-1} r_t)(1 - \delta |F_t - F_{t-1}|) \\ &= W_0 \prod_{t=1}^{T} (\sum_{i=1}^{m} F_{t-1}^i (\frac{z_t^i}{z_{t-1}^i}))(1 - \delta \sum_{i=1}^{m} \left| F_t^i - \widetilde{F}_t^i \right|) \end{aligned} \tag{3}$$

, where

$$\widetilde{F}_t^i = \frac{F_{t-1}^i (z_t^i / z_{t-1}^i)}{\sum_{j=1}^{m} F_{t-1}^j (z_t^j / z_{t-1}^j)} \tag{4}$$

, is the effective portfolio weight of asset i before adjusting.

2.3. Performance Criterion

Traditionally, we use utility function to express people's welfare in the position of certain wealth[13]:

$$\begin{aligned} U_\gamma(W_t) &= W_t^\gamma / \gamma \quad for\ \gamma \neq 0, \\ U_0(W_t) &= \log W_t \quad for\ \gamma = 0 \end{aligned} \tag{5}$$

When $\gamma = 1$, the investors only care about the absolute wealth or profit, and be viewed as risk neutral; $\gamma > 1$ indicates that people prefer to take more risk, and described as risk-seeking; $\gamma < 1$ corresponds to the case of risk averse, which means investors are more sensitive to loss than gains. By the assumption of CAPM, the rational investors are falling into this category. We take $\gamma = 0.5$ in our simulation.

Besides cumulative wealth, risk adjusted return are more widely used in the financial industry as the inputs for the value function. As suggested in the modern portfolio theory, Sharp Ratio is most widely accepted measure of risk adjusted return. Denote the time series returns as R_t, we have the Sharp Ratio definition:

$$S_T = \frac{Mean(R_t)}{Std.Deviation(R_t)} = \frac{\frac{1}{T}\sum_{t=1}^{T} R_t}{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(R_t - \frac{1}{T}\sum_{t=1}^{T} R_t)^2}} \quad (6)$$

Notice that in both cases we talk about above, the wealth and Sharp Ratio can be expressed as the function of R_t, so the performance criterion we talked so far can be expressed as:

$$U(R_T, ..., R_t, ...R_1; W_0) \quad (7)$$

3. Learning Theories

Reinforcement learning adjusts the parameters of a system to maximize the expected payoff or reward that is generated due to the actions of the system[16,17,18]. This process is accomplished through trial and error exploration of the environment and action strategies. In contrast to the supervised learning with decision data provided, the reinforcement algorithm receives a signal from the environment that current actions are good or not[19,20 21]. More specifically, our recurrent reinforcement learning can be illustrated in Figure 2.

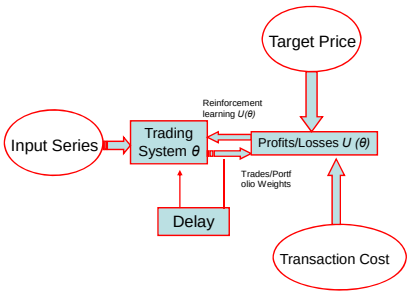


Figure 2. Algorithm Trading System using RRL

Reinforcement learning algorithms can be classified as either "policy search" or "value search"[22,23,24]. In the past 2 decades, value search methods such as Temporal Difference Learning (TD-Learning) or Q-learning are dominant topics in the field[19,25,26]. They truly work well in many application cases with convergence theorems existing at certain conditions [19].

However, the value function approach has many limitations. The most important shortcoming of value iteration is "Bellman's curse of dimensionality" due to the property of discrete states and action spaces[18,19]. Also, when Q-learning is extended to functional approximations, there are cases that the Markov Decision Processes do not converge. Q-learning also suffers from instability for optimal policy selection even when tiny noise exists in the datasets [27~30].

In our simulation experiments, we find that the "policy search" algorithm is more stable and efficient in stochastic optimal control. A compelling advantage of RRL is to produce real valued actions (portfolio weights) naturally without resorting to the discretization method in the Q-learning. Our experiment results also indicate that the RRL are more robust than value based search in noisy datasets, and thus more adaptable to the true market conditions.

3.1. Value Function and Q-Learning

The value function $V^{\pi}(x)$, is an estimate of future discounted reward that will be received from initial state x. Our goal is to improve policy π in each iteration step to maximize the value function which satisfies Bellman's equation[17,19,24]:

$$V^{\pi}(x) = \sum_a \pi(x,a) \sum_y p_{xy}(a)\{D(x,y,a) + rV^{\pi}(y)\} \quad (8)$$

Where $\pi(x,a)$ is the probability of taking action a in state x, $p_{xy}(a)$ is the transitional probability from state x to state y under action a. $D(x,y,a)$ is the intermediate reward, and r is the discount factor weighting relative importance of future rewards to current rewards.

The value iteration method is defined as:

$$V_{t+1}^{\pi}(x) = \max_a \sum_y p_{xy}(a)\{D(x,y,a) + rV_t^{\pi}(y)\} \quad (9)$$

This iteration will converge to the optimal value function defined as:

$$V^*(x) = \max_{\pi} V^{\pi}(x) \quad (10)$$

, which satisfies the Bellman's optimality equation:

$$V^*(x) = \max_a \sum_y p_{xy}(a)\{D(x,y,a) + rV^*(y)\} \quad (11)$$

, and corresponding optimal action can be inversely determined by optimal value function:

$$a^* = \arg\max_a \sum_y p_{xy}(a)\{D(x,y,a) + rV^{\pi}(y)\} \quad (12)$$

The Q-Learning version of Bellman's optimality equation is:

$$Q^*(x,a) = \sum_y p_{xy}(a)\{D(x,y,a) + r \max_b Q^*(y,b)\} \quad (13)$$

, and the update rule for training a function approximator is based on the gradient of the error:

$$\frac{1}{2}(D(x,y,a) + r \max_b Q(y,b) - Q(x,a))^2 \quad (14)$$

, which will lead to the optimal choice of action:

$$a^* = \arg\max_a \{Q(x,a)\} \tag{15}$$

In Q-learning, we need to discretize the states, and at each time step, we have the price state either be rise or fall, i.e {rise, fall}, and we have the corresponding action set be {buy, sell}. Due to the fixed combination at any time t with relative prices of securities and risk aversion assumption in the formalization of the problem, we have the transitional probability matrix as the following:

$$\begin{pmatrix} rise/buy & rise/sell \\ fall/buy & fall/sell \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \tag{16}$$

, where,

$$\begin{cases} \mathbf{A} = \mathbf{D} = \dfrac{\sum_b F_{t-1}{}^b (z_t{}^b / z_{t-1}{}^b - 1) \cdot 1\{z_t{}^b / z_{t-1}{}^b \geq 1\}}{\sum_a F_{t-1}{}^a \left| z_t{}^a / z_{t-1}{}^a - 1 \right|} \\ \mathbf{B} = \mathbf{C} = \dfrac{\sum_b F_{t-1}{}^b (z_t{}^b / z_{t-1}{}^b - 1) \cdot 1\{z_t{}^b / z_{t-1}{}^b < 1\}}{\sum_a F_{t-1}{}^a \left| z_t{}^a / z_{t-1}{}^a - 1 \right|} \end{cases} \tag{17}$$

, in which $1\{z_t{}^b / z_{t-1}{}^b \geq 1\}$ is the indicator function to be either 1 or 0 when the condition within the bracket is true or false.

3.2. Recurrent Reinforcement Learning

As we described above, the RRL algorithm will make the trading system $F_t(\theta)$ be real valued, and we can find the optimal parameter θ to maximize criterion U_T after T sequence of time step[2,15]:

$$\frac{dU_T(\theta)}{d\theta} = \sum_{t=1}^{T} \frac{dU_T}{dR_t} \{ \frac{dR_t}{dF_t} \frac{dF_t}{d\theta} + \frac{dR_t}{dF_{t-1}} \frac{dF_{t-1}}{d\theta} \} \tag{18}$$

The parameters can be optimized in batch gradient ascent by repeatedly computing value of U_T :

$$\Delta\theta = \rho \frac{dU_T(\theta)}{d\theta} \tag{19}$$

Notice that $\frac{dF_t}{d\theta}$ is total derivative that is path dependent, and we use the approach of Back-Propagation Through Time(BPTT) to approximate the total derivative in two time steps:

$$\frac{dF_t}{d\theta} = \frac{\partial F_t}{\partial \theta} + \frac{\partial F_t}{\partial F_{t-1}} \frac{dF_{t-1}}{d\theta} \tag{20}$$

So the simple online stochastic optimization can be obtained by taking derivatives on the most recent realized returns:

$$\frac{dU_t(\theta)}{d\theta} = \frac{dU_t}{dR_t} \{ \frac{dR_t}{dF_t} \frac{dF_t}{d\theta_t} + \frac{dR_t}{dF_{t-1}} \frac{dF_{t-1}}{d\theta_t} \} \tag{21}$$

, and parameters are updated online using:

$$\Delta\theta_t = \rho \frac{dU_t(\theta_t)}{d\theta_t} \tag{22}$$

4. Results and Discussions

4.1. Risky Asset and Trading Signal

Simply saying, trading signal is buying or selling decisions of investors, and it is reversely determined by maximizing the expectations of value functions in each time step. In our trading model, the buy action is assigned with value 1 and selling action is assigned with value -1, and more concretely, buying means enhance the relative weight of risky asset (increase θ) while selling indicates the vice versa. The first panel of Figure 3 is an example of trading signaling by reversely solving maximization problems based on the price information of risky assets. The second panel is the relative weights of the risky and riskless asset after the time series trading signal.

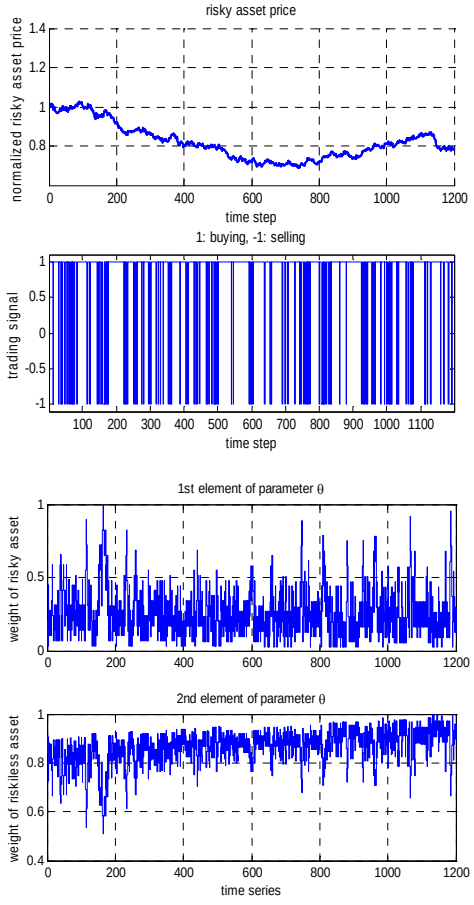
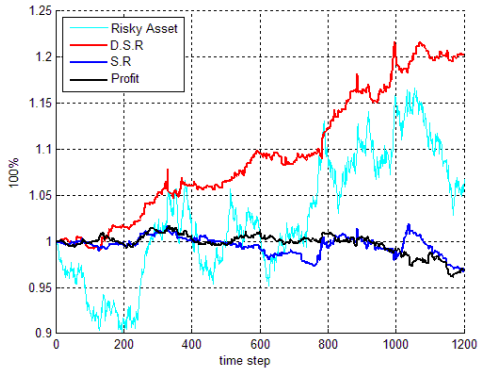
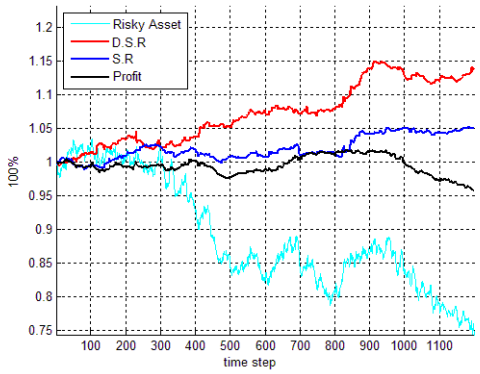


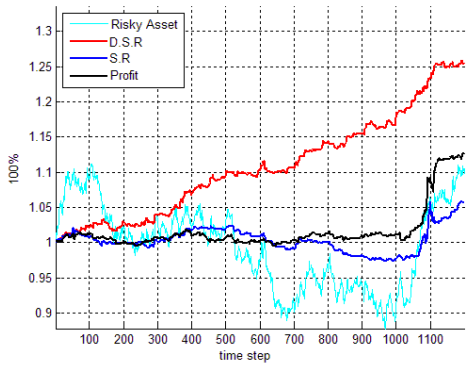
Figure 3. Simulation of Risky Asset and Trading Signal

4.2. Q-learning with Different Value Functions

We conduct a group of simulation using different value functions. The results of cumulative profits by Q-Learning are presented in the Figure 4. Each graph stands for a cumulative profit gains using interval profit, sharp ratio, and derivative sharp ratio as the value functions under one time series simulation of risky asset. We can see the cumulative profit varies considerably when different functions are applied. This is consistent with the instable property of Q-learning. Empirical results indicate that the change of value functions or the noisy of dataset may greatly change the policy selection and thus affect the final performance.







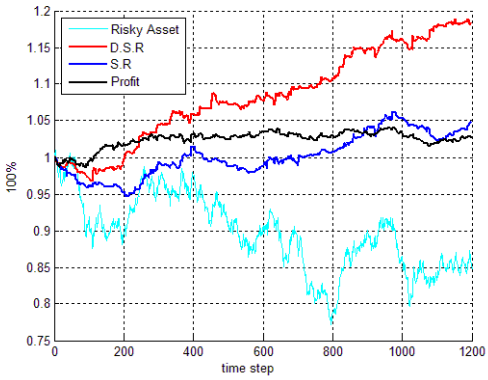


Figure 4. Trading Performance using Q-Learning by maximizing Derivative Sharp Ratio, Sharp Ratio, and Internal Profit

These simulations are carried out using 1200 time intervals, with underlying risky asset to be artificial normalized market portfolio price produced by geometric brown motion. The simulation results indicate that the derivative sharp ratio outperform in accumulating account wealth compared to the sharp ratio and internal profit. Economically speaking, the derivative sharp ratio is analogous to the marginal utility in terms of willingness to bear how much risk for one unit increment of sharp ratio. This value function incorporates not only the variance of the dataset but also the risk aversion of the investor, and thus more reasonable in probabilistic modeling.

When Q-learning algorithm applied to the noisy datasets, properly choosing value function plays key role in the stability of performance. As we discussed, in the cases of using sharp ratio and interval profit, the algorithm trading is not profitable and varies considerably when exposed to the different value scenarios of underlying asset. Another point worth noticing here is that the transaction cost may accrue.

4.3. Recurrent Reinforcement Learning Using Different Optimization Functions

Stochastic gradient ascent methods are widely applied in the backward optimization problems such as reinforcement learning. We optimize the value criterion of general function U_T by regulating the relative weight parameter θ as we discussed in 2.1.

Direction reinforcement learning uses policy iteration to achieve the optimal action without recursively solving a set of equations, and thus computational cheaper and more flexible. RRL algorithm has more stable performance than Q-learning when they are exposed to the fluctuating datasets. When sharp ratio and derivative sharp ratio are used for the optimization search, they both have a stable and profitable performance under different underlying price scenarios.

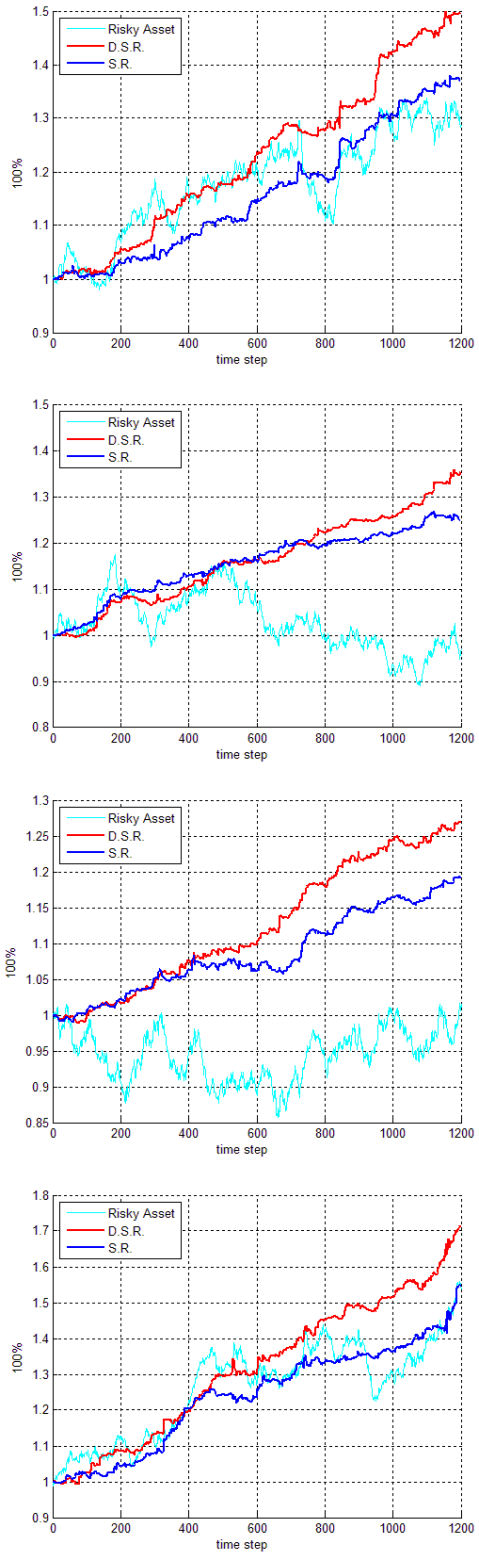


Figure 5. Investment Performance of Recurrent Reinforcement Learning using Stochastic Batch Gradient Ascent.

4.4. Recurrent Reinforcement Learning Using Different Length of Backward Steps

Formula (18) gives the stochastic batch gradient ascent algorithm. In practice, we do not include all the derivative terms up to time T, instead we just take several previous temporal steps for computational savings. We studied the effects of this omission on the invest performance by taking different number of backward time steps in the optimization.

We use normalized S&P 500 index as the market portfolio and compared the performance of algorithm trading using different amount of backward time steps:

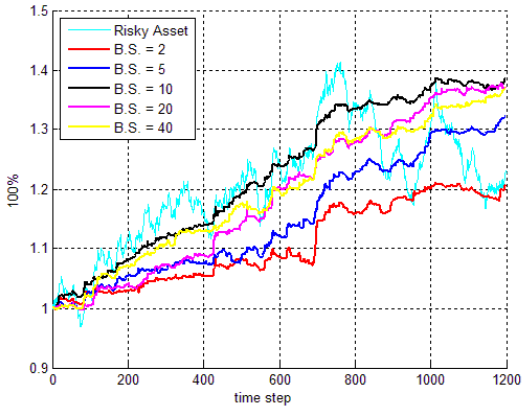


Figure 6. RRL Algorithm Trading Performance using Different Amount of Temporal Information

We use derivative sharp ratio as the objective function subject to optimize at each time step, and choose backward time steps to be 2 hours, 5 hours, 10 hours, 20 hours, and 40 hours respectively. The RRL algorithm has steady increased on the account wealth in all the learning cases and as more information incorporated, cumulative profit grows with higher rates. When backward temporal step is over 10, the investment action does not improve much if we include more previous temporal data.

5. Conclusion

We have proposed and tested a trading system via reinforcement learning method. We use value iteration and policy iteration respectively to test the algorithm trading performance using Q-learning and Recurrent Reinforcement Learning respectively. In the implementing of Q-learning, we use interval profit, sharp ratio, derivative sharp ratio as the value functions subject to optimize at each time step and tested their performance. Our empirical study indicates that value function selection is very important to achieve stable learning algorithm in the value iteration especially applied to the noisy dataset. Value function with information on the noisy (variance) and dynamic properties (marginal utility) has better

performance in the learning algorithm than that with stationary and fixed value functions.

In general, RRL with policy iteration outperform than Q-Learning in the sense of stability and computational convenience. RRL algorithm provides more flexibility in choosing objective functions to optimize using stochastic batch gradient ascent. More inclusion of backward information helps the learning process, but increment of performance tends to converge as more previous temporal steps are incorporated.

References

[1] Moody, J., Saffell, M., Liao, Y. & Wu, L. (1998), Reinforcement learning for trading systems and portfolios: Immediate vs future rewards,

[2] V.S. Borkar and S.P. Meyn. The O.D.E method for convergence of stochstic approximation and reinforcement learning. Siam J. Control, 38 (2):447–69, 2000.

[3] Michael Kearns and Satinder P. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms.

[4] F. Beleznay, T. Grobler, and C. Szepesvari. Comparing value-function estimation algorithms in undiscounted problems. Technical Report TR-99-02, Mindmaker Ltd, 1999. 24

[5] F. Sehnke, C. Osendorfer, T. Rückstiess, A. Graves, J. Peters, and J. Schmidhuber. Policy gradients with parameter-based exploration for control. In J. Koutnik V. Kurkova, R. Neruda, editors, Proceedings of the International Conference on Artificial Neural Networks ICANN-2008

[6] J. N. Tsitsiklis and B. Van Roy, "Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives," IEEE Transactions on Automatic Control, vol. 44, no. 10, pp. 1840-1851, October 1999.

[7] F. J. Gomez, J. Togelius, J. Schmidhuber. Measuring and Optimizing Behavioral Complexity for Evolutionary Reinforcement Learning . Proceedings of the 19th International Conference on Artificial Neural Networks (ICANN-09), Cyprus, 2009.

[8] J. N. Tsitsiklis and B. Van Roy, "Regression methods for pricing complex American-style options," IEEE Transactions on Neural Networks, vol. 12, no. 4, July 2001,

[9] D. Nawrocki, "Optimal algorithms and lower partial moment: Ex post results," Applied Economics, vol. 23, pp. 465-470, 1991.

[10] Martin L. Puterman. Markov Decision Processes—Discrete Stochastic Dynamic Programming. John Wiley and Sons, Inc., New York, NY, 1994.

[11] Xu, X., Hu, D., & Lu, X. (2007). Kernel-Based Least Squares Policy Iteration for Reinforcement Learning. IEEE Transactions on Neural Networks (pp. 973–992).

[12] Kang, J., Choey, M. & Weigend, A. (1997), Nonlinear trading models and asset allocation based on Sharpe ratio, 'Decision Technology for Financial Engineering',World Scientific, London.

[13] Darrell Duffie, Dynamic Asset Pricing Theory, Princeton University Press, 2 edition, 1996.

[14] L. Brock, J. Lakonishok, and B. LeBaron. Simple technical trading rules and the stochastic properties of stock returns. Journal of Finance, 47:1731–1764, 1992.

[15] Moody, J. & Wu, L. (1996), Optimization of trading systems and portfolios, Neural Networks in the Capital Markets Conference Record, Caltech, Pasadena.

[16] D. Wierstra, A. Foerster, J. Peters, J. Schmidhuber. Recurrent Policy Gradients. Journal of Algorithms, 2009.

[17] Neuneier, R. (1996), Optimal asset allocation using adaptive dynamic programming, 'Advances in Neural Information Processing Systems 8', MIT Press.

[18] John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell, "Performance functions and reinforcement learning for trading systems and portfolios," Journal of Forecasting, vol. 17, pp. 441-470, 1998.

[19] R. Sutton and A. Barto. Reinforcement Learning. MIT Press., Cambridge, MA., 1998.

[20] C. J. Watkins and P. Dayan, "Technical note: Q-Learning," Machine Learning, vol. 8, no. 3, pp. 279-292, 1992.

[21] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore, "Reinforcement learning: A survey," Journal of Artificial Intelligence Research, vol. 4, 1996.

[22] Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., & Littman, M. (2008). An Analysis of Linear Models, Linear Value-Function Approximation, and Feature Selection for Reinforcement Learning. International Conference of Machine Learning (pp. 752–759).

[23] Robert H. Crites and Andrew G. Barto, "Improving elevator performance using reinforcement learning," in Advances in Neural Information Processing Systems,

[24] Peter Marbach and John N. Tsitsiklis, "Simulation-based optimization of Markov reward processes," in IEEE Conference on Decision and Control, 1998.

[25] D. Wierstra, A. Foerster, J. Peters, J. Schmidhuber. Recurrent Policy Gradients. Journal of Algorithms, 2009.

[26] Leemon Baird and Andrew Moore, "Gradient descent for general reinforcement learning," in Advances in Neural Information Processing Systems, Eds. 1999, MIT Press.

[27] A- W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less real time," Machine Learning, vol. 13, pp. 103-130, 1993.

[28] Mahadevan, S., & Maggioni, M. (2006). Proto-value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes (Technical Report). University of Massachusetts.

[29] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour, "Policy gradient methods for reinforcement learning with function approximation," in Advances in Neural Information Processing Systems, Eds. 2000, vol. 12, pp. 1057-1063, MIT Press.

[30] John Moody and Matthew Saffell, "Reinforcement learning for trading," in Advances in Neural Information Processing Systems, Eds. 1999, vol. 11, pp. 917-923, MIT Press.