Dynamic Asset Allocation Exploiting Predictors in Reinforcement Learning Framework

Jangmin O[1], Jae Won Lee[2], Jongwoo Lee[3], and Byoung-Tak Zhang[1]

[1] School of Computer Science and Engineering, Seoul National University, San 56-1, Shillim-dong, Kwanak-gu, Seoul, Korea 151-742
{jmoh,btzhang}@bi.snu.ac.kr

[2] School of Computer Science and Engineering, Sungshin Women's University, Dongsun-dong, Sungbuk-gu, Seoul, Korea 136-742
jwlee@cs.sungshin.ac.kr

[3] Department of Multimedia Science, Sookmyung Women's University, 53-12 Chongpa-dong 2-ga, Yongsan-gu, Seoul, Korea 140-742
jwlee44@naturesol.co.kr

Abstract. Given the pattern-based multi-predictors of the stock price, we study a method of dynamic asset allocation to maximize the trading performance. To optimize the proportion of asset to be allocated to each recommendations of the predictors, we design an asset allocator called meta policy in the Q-learning framework. We utilize both the information of each predictor's recommendations and the ratio of the stock fund over the asset to efficiently describe the state space. The experimental results on Korean stock market show that the trading system with the proposed asset allocator outperforms other systems with fixed asset allocation methods. This means that reinforcement learning can bring synergy effects to the decision making problem through exploiting supervised-learned predictors.

1 Introduction

During the last a few decades, several algorithms have been applied to the stock market problems [4]. But attempts on modeling or predicting the stock market have not been successful in *consistently* beating the market. This is the famous EMH (Efficient Market Hypothesis) saying that the future prices are unpredictable since all the information available is already reflected on the history of past prices [7]. However, if we step back from *consistently*, then we can find several empirical results saying that the market might be somewhat predictable [1]. Especially, the newest algorithms in artificial intelligence, equipped with powerful representational and modeling power, have been applied to the problems of the stock market, such as price prediction, risk management and portfolio optimization.

Many works have been applied to the price prediction. Supervised learning such as neural networks, decision trees, and SVMs (Support Vector Machines) are intrinsically well suited to the problem [5, 11]. The risk management and portfolio optimization have been intensively studied in reinforcement learning [6, 8–10].

In this paper, we confine our main interest on trading the individual stocks in the market. The works with the stock price prediction based on supervised learning [5, 11] lack in considering the risk management and portfolio optimization.

The mechanism of the time delayed learning with the cumulative rewards in reinforcement learning makes it natural to consider the risk management and portfolio optimization. But the researches [9] put simple assumptions on the market to make the problem manageable in the reinforcement learning framework. Also the portfolios of the researches [8] are simple because they focus on switching just between two price series. The works [6, 10] treat trading individual stocks in reinforcement learning but lack in asset allocation.

In order to handle both stock price prediction and trading with efficient asset allocation, we dive the problem into two separate parts. For the price prediction, we adopt neural networks approach. But rather than an all-purpose predictor, we build a multi-predictors approach based on several meaningful patterns. For the asset allocation, we want the asset to be dynamically distributed on each predictor's recommendation. To achieve this aim we design a Q-learning framework of the dynamic asset allocator called *meta policy* which makes a decision on the asset allocation exploiting the information of recommendations of the multi-predictors and the stock fund ratio over the total asset.

The resulting trading performance is compared with the performances of other fixed asset allocation techniques through a simulation on Korean stock market.

2 Summary of Predictors

We have constructed the pattern based predictors. Rather than only one predictor to cover the whole stock price patterns, we focus only on several typical patterns. Table 1 represents the price patterns our predictors treat[1]. The patterns cover roughly 80% of stock price series in Korean stock market. The predictors, *engine_bear*, *engine_bull*, *engine_GC* and *engine_TU* are developed from these datasets using artificial neural networks.

Table 1. Patterns of predictors used in this paper

Pattern	Description
D_{bull}	Moving average lines are arranged in order.
D_{bear}	Moving average lines are arranged in reversed order.
D_{GC}	Golden cross between moving average lines occurs.
D_{TU}	Sign of gradient of moving average line is changed.

[1] For more detailed description about the meaning of Table 1, see [3].

After each prediction engine was trained, its performance might be evaluated using several criteria. Many evaluation criteria are summarized in [2] but we have adopted a simulation-based evaluation using, $\mathcal{LP}$, called *local policy*.

Definition 1. *Given a space Ω = (B_THRES, A_THRES, H_DURATION), a local policy $\mathcal{LP}$ is a set of optimized parameters for each predictor.*

$$\mathcal{LP}(predictor) = (b\_thres, a\_thres, h\_duration).$$

Table 2 shows the meaning of each parameter. We regard the prediction values of a predictor larger than *b_thres* as bid signals. *a_thres* is the threshold for ask signals and *h_duration* is the maximal duration of holding a stock. Using an $\mathcal{LP}$, the trading based on each predictor is simulated as follows. The stocks of which predicted values are larger than *b_thres* are retrieved and each purchased stock is sold when its predicted value becomes lower than *a_thres* or its number of holding days expires. For each predictor, its $\mathcal{LP}$ is greedily optimized over the parameter space. For comparison, two metrics, PPT and accuracy, are used. PPT means average profit rate of the trades and accuracy is defined as;

$$accuracy = \frac{\#\text{of successful trades}}{\#\text{of recommendations}}, \tag{1}$$

where successful trade means the trade achieving positive profit after subtracting the transaction cost.

Table 3 summarizes the results. The *engine_normal* corresponds to all-purpose predictor. The accuracy of each predictor is improved 12% compared with *engine_normal*. The predictors are remarkably superior to *engine_normal*. Although our multi-predictors approach can not cover whole patterns of stock prices, this result is very promising.

3 Need for Meta-policy

Here we analyze the tendencies of our predictors. Figure 1 is the bar graph of the successful or failed recommendations of *engine_bear* with its $\mathcal{LP}$. This figure shows that the profitable recommendations are not equally distributed over the trading period. They are concentrated on some specific days. It means that profits induced from the trades of a few days dominate others.

Table 2. Parameters of the local policy of an predictor

Name	Meaning
b_thres	threshold of bid signal
a_thres	threshold of ask signal
h_duration	period of holding a stock

Table 3. Performances of the predictors

Engine	Accuracy(%)	PPT(%)
engine_normal	57.10	0.79
engine_bear	69.42	1.41
engine_bull	73.37	1.99
engine_GC	71.10	1.72
engine_TU	70.97	1.63

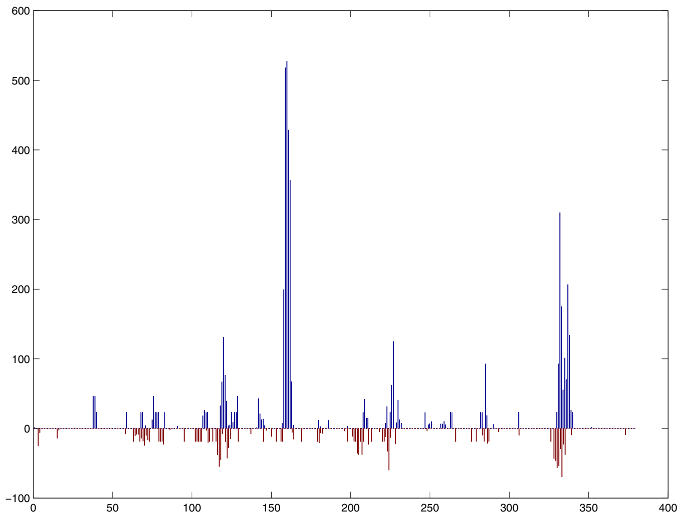
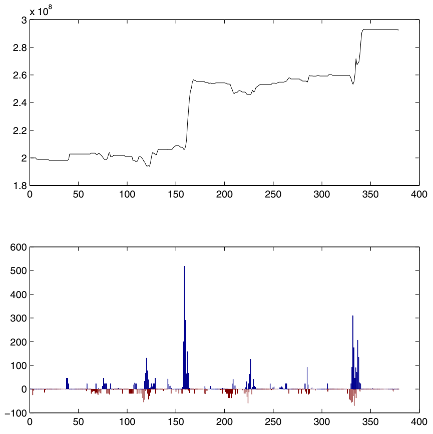


Fig. 1. The tendency of recommendations of *engine_bear*: x-axis represents trading days. y-axis represents the total profits induced by recommendations of a day. Up-headed bar means the trades of the day result in positive profit, and vice versa



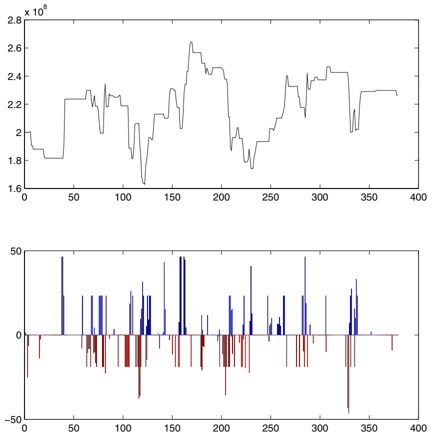


Fig. 2. Funding log and traded recommendations when the purchase money per recommendation is 0.4 million Won

Fig. 3. Funding log and traded recommendations when the purchase money per recommendation is 40 million Won

To explore this, we simulate the tradings with two different purchase money per recommendation (PMR), 0.4 and 40 million Won, given initial asset 200 million Won[2]. Figure 2 is for the former and Fig. 3 is for the latter. The upper

[2] Won is a monetary unit of Korea.

part of each figure shows the history of the asset change and the lower part is the bar graph.

With a small PMR we can trade most of the recommended stocks even when a number of recommendations are made on a specific day, such as around the 160-th or 340-th trading day. These many profitable trades increase the asset volume. But in this case, too small portion of the asset is invested to the stocks in usual days with small number of recommendations. In those days, though the trades of the recommended stocks induce high profits, the total asset is little influenced by those trades.

Whereas with a large PMR, the asset is heavily affected by individual trades. The history of the asset in Fig. 3 says the asset is fluctuated by trades. Moreover, with a large PMR we can trade just a few stocks recommended at the peak days. So the bar graph of Fig. 3 looks less profitable.

Therefore, it is desirable to adapt the amount of PMR in order to achieve more effective trading performance. Furthermore since we have multiple predictors, we need a more complicated asset allocation policy which can effectively adjust PMR for each predictor. For this, we define a *meta policy*:

Definition 2. *Let the number of recommendations of multi-predictors be $\mathcal{N} = (N_{bear}, N_{bull}, N_{GC}, N_{TU})$ and the ratio of stock fund over the total asset be $\mathcal{SF}$. A meta policy $\mathcal{MP}$ is defined as a function over $\mathcal{N}$ and $\mathcal{SF}$,*

$$\mathcal{MP}(\mathcal{N}, \mathcal{SF}) := (P_{bear}, P_{bull}, P_{GC}, P_{TU}),$$

where P_{engine} is the proportion of PMR for engine.

Figure 4 summarizes the trading process with a meta policy. At t-th day of T trading period, E predictors retrieve their recommendations and $\mathcal{MP}$ determines the purchase money for each predictor of that day according to $N_1, \ldots, N_E$ and $\mathcal{SF}$. Each candidate of a predictor is traded according to the $\mathcal{LP}$ of the predictor.

4 Asset Allocator Using Reinforcement Learning

Reinforcement learning is a computational approach to understand and automate goal directed learning and decision making [12]. One of the most important

```
for $t = 1$ to $T$
    for $e = 1$ to $E$
        $S_e$ = retrieve($D_e$, $b\_thres$)
        $N_e$ = numberof($S_e$)
    end for
    $(P_1, \cdots, P_E) = \mathcal{MP}(N_1, \cdots, N_E, \mathcal{SF})$
    for $e = 1$ to $E$
        local_trade($P_e$, $S_e$, $\mathcal{LP}(P_e)$)
    end for
end for
```

Fig. 4. Trading process with a meta policy

keys to achieve reasonable performance in machine learning is the representation of the input space. Specially reinforcement learning is an art of the design of state and reward [10]. Here, we describe how we formulate a meta policy in the Q-learning framework.

The state s_t is a state vector at time t that the environment gives to the agent. We define s_t as;

$$s_t = (N_1^{bits}(t), N_2^{bits}(t), N_3^{bits}(t), N_4^{bits}(t), F^{bits}(t)).$$

The state is divided into two parts one about the predictors and the other part about the stock fund ratio. Each $N_i^{bits}(t)$ is the bit vector which represents the number of recommendations of predictor i at time t. Since the number of recommendations is unbounded integer valued, we use an orthogonal vector representation with its length restricted to 10 as in Table 4.

Table 4. Bit vector of a predictor's recommendations

# of recommendations	bit vector
0	0000000000
1	0000000001
...	...
8	0100000000
9∼	1000000000

Table 5. Bit vector of the stock fund ratio

ratio of the stock fund	bit vector
$[0, 5)$	00000000000000000001
$[5, 10)$	00000000000000000010
...	...
$[90, 95)$	01000000000000000000
$[95, 100)$	10000000000000000000

$F^{bits}(t)$ is a bit vector representing the ratio of the stock fund[3] over the asset. Since the ratio can range between 0% to 100%, we discretize it into 20 intervals and express it with 20 bit orthogonal vector as in Table 5.

The action a_t at s_t is a PMR of each predictor over the asset. Although it is natural to use real valued ratio, it is hard to consider the real valued action under Q-learning . We have to properly discretize to reduce the cardinality of the candidate actions. If we restrict the ratio to take one of four values, $(0.5, 1.0, 3.0, 5.0)$, the action can be written as

$$a_t = (P_1^{bits}(t), P_2^{bits}(t), P_3^{bits}(t), P_4^{bits}(t)),$$

where $P_i^{bits}(t)$ is four bit vector to pick up one of the four values. After all, the number of possible actions is 256.

If we use a table representation as the conventional Q-learning does, we might need $10^4 \times 20 \times 4^4 = 51.2$ million entries. The table size is pretty big, but not unmanageable in the present computing power. Here we do not adopt any function approximators for Q-value but use only the table representation.

[3] Stock fund is the estimated money of stocks the investor is holding. Ready fund if the difference between total asset and stock fund.

```
Initialize $Q(s,a)$ as zero.
Repeat (for specified number of episodes):
    Make an episode by sampling $T_1$ and $T_n$.
    Environment initializes $asset_{T_1}$ and $s_{T_1}$.
    for $t = T_1$ to $T_{n-1}$
        $a_t \longleftarrow Action(s_t)$
        $\mathrm{asset}_{t+1} \longleftarrow Trade(\mathrm{asset}_t, a_t)$
        if $t < T_{n-1}$ then
            $r_t \longleftarrow 0$
        else
            $r_t \longleftarrow$ profit ratio
        end if
        Environment produces $s_{t+1}$
        $\delta_t \longleftarrow r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)$
        $Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha \times \delta_t$
    end for
```

Fig. 5. Q-learning algorithm for meta policy

Figure 5 describes our Q-learning framework with a meta policy. To produce episodes as many as over the training period, both the start and the end day of an episode are uniformly sampled over the total training period. While an episode is lasting, the agent chooses an action from the function *Action*, trades stocks with chosen action, is given reward, and updates the table of Q-value. The function $Trade(asset_t, a_t)$ simulates one-day trading process according to a_t with initial asset $asset_t$. The output of $Trade$ is the initial asset of the next day.

As the reward function, we choose the profit ratio on the trading period which is defined as

$$\text{profit ratio} = 100 \times \frac{\mathrm{asset}_{T_n} - \mathrm{asset}_{T_1}}{\mathrm{asset}_{T_1}}. \tag{2}$$

The reward of each action is zero during the inside of the trading period and is given only when the final day of the episode.

5 Experiments

In this section, we experimented the performance of the trading system with meta policy optimized by Q-learning, named `MPG`. Two other trading systems, named `trader1` and `trader2` as shown in table 6, were used for the performance comparison. Each trading system shares the predictors, which were constructed on the stock price data of Korean Stock Exchange market[4] from January 1998 to March 2000, but trades on its own asset allocation policy.

[4] Korean Composite Stock Price Index (KOSPI) is widely used as an abbreviated word for Korean Stock Exchange Market.

Table 6. Policies of the Trading Systems

name	description
`trader1`	Fixed policy, initially partitioned asset
`trader2`	Fixed policy, united asset
`MPG`	Adaptive policy, united asset

5.1 Asset Allocation Policy and Performance of Each System

In case of `trader1`, as if we had a sub-trading systems per predictor, an initial asset was equally divided into parts, totally the number of predictors. The stocks recommended by a predictor were traded with the exclusive asset for that predictor. The exclusive asset can not be utilized to trade the recommended stocks from other predictors. The asset allocation policy of `trader1` consists of four constants,

$$\mu_{bear}, \mu_{bull}, \mu_{GC}, \mu_{TU},$$

where μ_e is the proportion of the asset as purchase money of predictor e . For each predictor the constants were calculated so that it could achieve maximal profit on the period between April 2000 to December 2001.

In `trader2`, the initial asset was not divided but was managed as united form. Whenever there are any recommendations from predictors, `trader2` allocates $1/k_0$ times of asset to purchase, where k_0 was tuned on the same period as `trader1`.

5.2 Training MPG

`MPG` was constructed by Q-learning as described in Sect. 4. It was trained on the period from April 2000 to December 2001, the same period as the tuning period of `trader1` and `trader2`. To optimize MPG, we specified the parameters for Q-learning as follows. γ was set to 0.9, α was set to 0.02 and ϵ was set to 0.2. High uncertainty of the stock market makes the overfitting on the training period inevitable. Therefore, for the training set of MPG, we excluded the last six months of the training set as the validation set.

Figure 6 shows the training tendency of MPG. Whenever 1,000 episodes were experienced, we plotted the trading performances both on the entire training period and the validation period. Horizontal axis is each measuring point and vertical axis is the trading performance, the profit ratio. Solid line is the profit ratio on the training period and dotted line is on the validation period. On the validation period, after the best performance reaches about 210% with 500,000 episodes experienced, the performance is declined, which means the more updating Q-values leads overfitting on the training set. So we stop further training and produce the final MPG at that step.

5.3 Trading Performances

Figure 7 shows the comparison of three trading systems on the test period from January 2002 to May 2003. Horizontal axis is the trading day and vertical axis

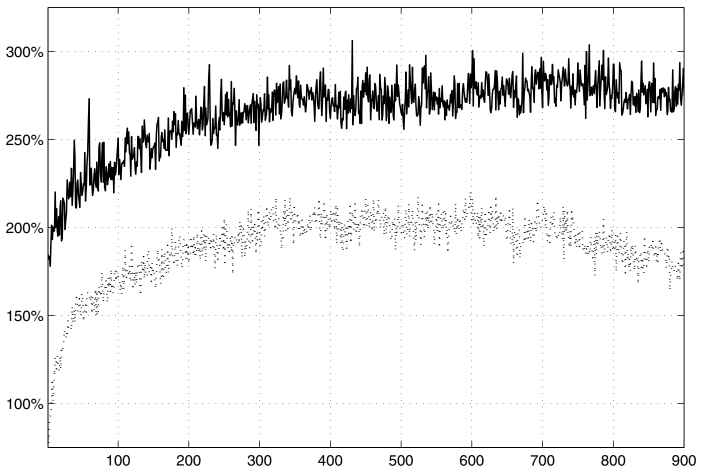


Fig. 6. Training tendency of MPG

Table 7. Profit induced by each trading system

trading systems	profit ratios	relative profit ratios
trader1	76.92%	102.49%
trader2	115.74%	146.92%
MPG	257.76%	309.46%

is the total asset. Each trading system started with its trading money 25 million Won. The KOSPI values were scaled up so that it looks as if it were 25 million Won at January 2, 2002. Solid line is the history of asset of `MPG`, dashed-dotted is of `trader1`, dashed line is of `trader2`, and dotted line is of composite index.

Table 7 summarizes the performance induced by each trading system. Profit ratio is calculated as Eq. (2) and the relative profit ratio is the relative profit compared with composite index at the end of trading period. Clearly, `MPG` outperforms other two trading systems. After 17 months' trading, its asset rose by nearly 65 million Won, or 257.76%, to about 90 million Won. `trader1` shows the worst among three trading systems. There may be some situations where one predictor meets too many recommendations to manage only using its ready fund, while there are much of ready funds for other predictors.

`trader2` also shows limited performance. Since it tries to buy stocks whenever any predictor recommends, it might avoid the ill conditions of `trader1`. Although it outperforms `trader1` over most of periods, `trader2` does not consider the relationship between predictors or the ratio of stock fund. During the hard time of August and September 2002, `trader1` and `trader2` suffer the decline of asset, but `MPG` endures this hard period. It shows somewhat avoiding the risky declination and getting more chance of level up of the total asset using adaptive asset allocation.

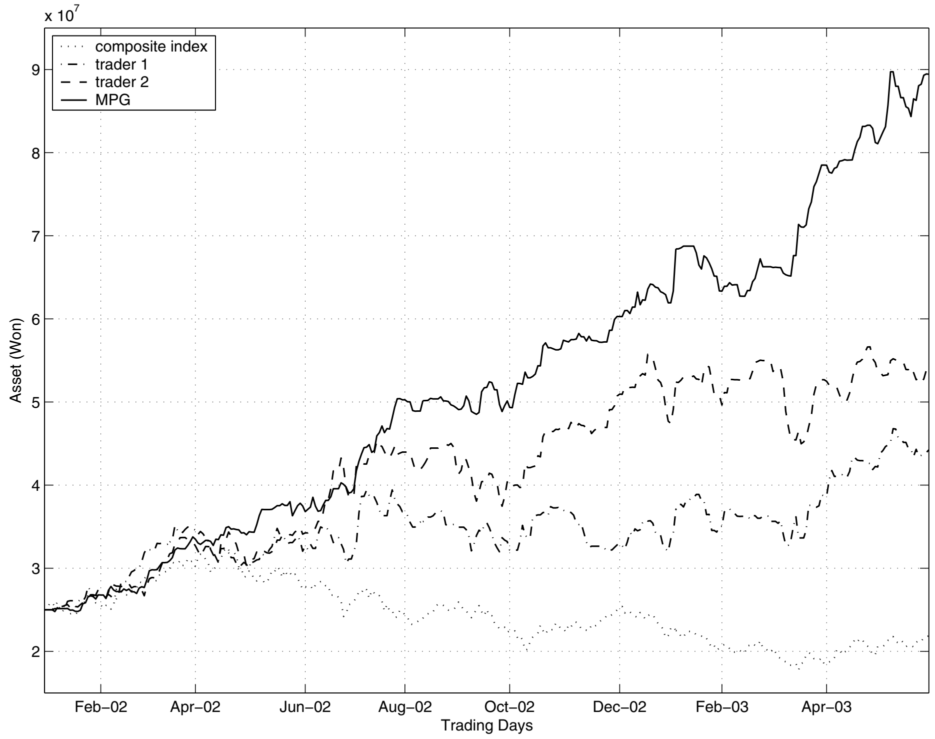


Fig. 7. A comparison of the performances of the trading systems

6 Discussion

Figure 8 shows a synthetic analysis of MPG including the proportion of stock fund and the information of recommendations. Upper part of the figure is about asset history, middle part is about the proportion of stock fund to total asset, and lower part is about the sums of positive or negative profits of each day under the assumption that every recommendation might be purchased. After April 2002, KOSPI is in bear market. From late June 2002 to mid July 2003, there is a sharp rally in KOSPI. *engine_bear* meets its explosive recommending day at June 27, 2002. The asset allocation rule of that day is $(5.0, 0.5, 0.5, 1.0)$, where 5.0 is for *engine_bear*. Stock fund is 5% at June 27, 2002, so we can purchase most of the recommendations. As the result, stock fund of June 28, 2002 increases to 98%.

While KOSPI is in its declining bear at late December 2002, the number of recommendations leading to loss increases. At December 20, 2002 *engine_bear* and *engine_bull* recommend 4 and 6 stocks respectively. But their allocation ratios are 0.5 and 0.5. During a few consecutive dangerous days, the small allocation ratios for corresponding predictors prevent the stock fund from being dangerously increased. Although some amount of loss is inevitable, it is more stable than other traders. This situation is more vivid in early March 2003 when `trader2` suffers severe loss of the asset while MPG can minimize the fatality.

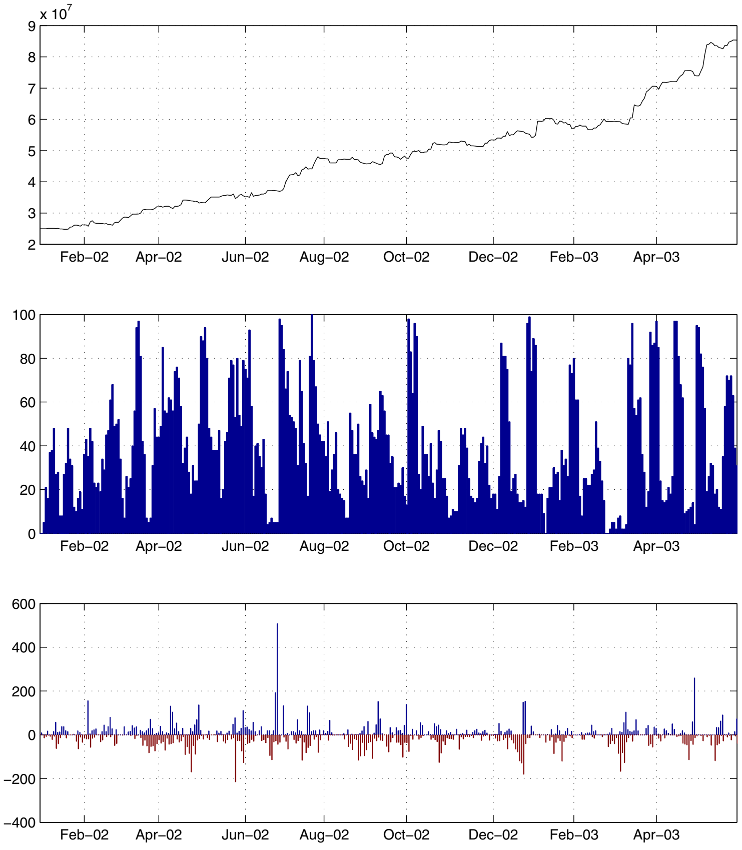


Fig. 8. Synthetic Analysis of MPG

However `MPG` is not always winning. For example, during late January 2003, the stock fund increases with wrong recommendations purchased by somewhat large amount money. Since the four predictors are not perfect, it is not possible to induce constantly winning trades. But, the higher level consideration of several predictors and stock fund ratio in `MPG` makes trading more profitable and unrisky.

7 Conclusion

In this paper, the multi-predictor approach is further utilized by reinforcement learning to adaptively allocate the asset over the recommendations from each predictor. We make an adaptive asset allocator, meta policy, incorporating the information of recommendations of predictors and the stock fund ratios as the part of the state representation of Q-learning.

Using meta policy induces more profits than the trading based on other fixed asset allocation methods in the simulation on Korean stock market over specified test period. This results show that using reinforcement learning to utilize supervised learning can makes the complex problems manageable such as the stock trading as well as synergetic effect of both algorithms in decision making.

Acknowledgement

This research was supported in part by the Ministry of Education and Human Resources Development under the BK21-IT Program. The RIACT at Seoul National University provides research facilities for this study. The Nature Solutions Corp. provides the stock data and simulation tools.

References

1. E. F. Fama and K. R. French, Dividend Yields and Expected Stock Returns, *Journal of Financial Economics*, 22, pp. 3-26, 1988.
2. T. Hellström, *A Random Walk through the Stock Market*, ph.D. Thesis, Department of Computing Science, Umeå University, 1998.
3. P. J. Kaufman, *The New Commodity Trading Systems and Methods*, Wiley, NewYork, 1987.
4. S. M. Kendall and K. Ord, *Time Series*, Oxford, New York, 1997.
5. S. D. Kim, J. W. Lee, J. Lee, and J.-S. Chae, A Two-Phase Stock Trading System Using Distributional Differences, *In Proceedings of International Conference on Database and Expert Systems Applications*, pp. 143-152, 2002.
6. J. W. Lee and J. O, A Multi-agent Q-learning Framework for Optimizing Stock Trading Systems, *In Proceedings of International Conference on Database and Expert Systems Applications*, pp. 153-162, 2002.
7. B. G. Malkiel, *A Random Walk Down Wall Street*, Norton, New York, 1996.
8. J. Moody and M. Saffell, Learning to Trade via Direct Reinforcement, *IEEE Transactions on Neural Networks*, 12(4), pp. 875-889, 2001.
9. R. Neuneier, Risk Sensitive Reinforcement Learning, *Advances in Neural Information Processing Systems*, pp. 1031-1037, MIT Press, Cambridge, 1999.
10. J. O, J. W. Lee, and B.-T. Zhang, Stock Trading System Using Reinforcement Learning with Cooperative Agents, *In Proceedings of International Conference on Machine Learning*, pp. 451-458, Morgan Kaufmann, 2002.
11. E. W. Saad, D. V. Prokhorov, D. C. Wunsch II, Comparative Study of Stock Trend Prediction Using Time Delay, Recurrent and Probabilistic Neural Networks, *IEEE Transactions on Neural Networks*, 9(6), pp. 1456-1470, 1998.
12. R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction.* MIT Press, Cambridge, 1998.