Combining technical trading rules: a recurrent reinforcement learning approach

Yufeng Tao

Advisor: Dr. Philip L.H. Yu

Department of Statistics and Actuarial Science, The University of Hong Kong

February, 2020

Abstract

This study proposes a Rule-based Recurrent Reinforcement Learning (RbRRL) strategy for financial trading to bridge the power of data-driven machine intelligence and the insight from human interpretable industry practice. Technical trading rules from different classes are combined with regularized weighting approximated by counterfactual returns and neuron activation. Although the conventional recurrent reinforcement learning outperforms RbRRL in terms of cumulative capital gain, the latter shows lower volatility as well as higher interpretability.

1 Introduction

Technical indicators have been used in the capital market to identify trading opportunities for more than a century. Nevertheless, a single trading rule may not be sufficient to predict the stock price trend accurately. Although some sophisticated trading strategies combining various classes of trading rules have been proposed in the literature, they often pick only one rule for each class, which may lose valuable information from other rules in the same class.

Dr. Philip Yu and his team[1] proposed a Performance-Based Reward strategy (PRS) that combines the two most popular classes of technical trading rules: moving average (MA) and trading range break-out (TRB). For both MA and TRB, PRS considers various combinations of the rule parameters to produce a universe of 140 component trading rules in all. Each component rule is assigned a starting weight, and a reward/penalty mechanism based on its recent profit is proposed to update the weighting over time.

However, the above reward/penalty mechanism closely resembles the rationale of reinforcement learning: through trial and error to maximize cumulative rewards. Recurrent reinforcement learning (RRL) was first introduced by Moody, Wu, Liao, and Saffell [2] where a trading system was developed focusing on a single asset. Most recently, Almahdi and Yang [3] introduced a more generalized recurrent reinforcement learning approach on portfolio management.

This study intends to reframe the PRS by RRL, which we later model as Rule-based Recurrent Reinforcement Learning (RbRRL), in the hope of capturing both the power of data-driven trading automation and the insight from human-designed industry practice.

2 Methodology

2.1 Component trading rules

Same as the PRS, we consider two types of the simplest and most popular technical trading rules, MA and TRB, with various parameter combinations.

Specifically, a moving average is the mean of stock prices over a moving window of n days as follows:

$$\bar{p} = \frac{1}{n} \sum_{i=t-n+1}^{t} p_i$$

where t is the current trading day and p_i is the close stock price on day i. It is recalculated and updated each trading day. In MA rules, there are two averages (long-period and short-period averages) over two moving windows of m days and n days, respectively, where $m > n$. Consider a trading day t, a MA rule initiates buy (sell) signal if the short-period moving average is above (below) the long-period moving average. Below are the parameter values to be used in this study:

m (number of days in a long-period moving average) = 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 125, 150, 200, 250 (14 values);

n (number of days in a short-period moving average) = 1, 2, 5, 10, 15, 20, 25, 30, 40, 50, 75, 100, 125, 150, 200 (15 values).

Because n should be less than m, the total number of MA rules generated is 119.

The second technical trading rule is trading range break-out (TRB). It calculates the highest close price H and the lowest close price L over a fixed n days interval as follows:

$$H = \max(p_{t-1}, p_{t-2}, \ldots, p_{t-n})$$

$$L = \min(p_{t-1}, p_{t-2}, \ldots, p_{t-n})$$

The highest and lowest price form a running channel (trading range) for each day's stock price and the trading signals are invoked by the stock price's breakout from the channel. Suppose the close price of trading day t is p_t, a buy signal is generated when $p_t > H$ and a sell signal is generated when $p_t < L$. Below are the parameter values:

n (number of days for a trading range) = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 75, 80, 90, 100, 125, 150, 175, 200, 250 (21 values). Because there is only one parameter, the total number of TRB rules generated is 21.

Thus, in total we have number of rules $(NR) = 140$.

2.2 Combined signal generation

2.2.1 Counterfactual rule returns

Note that for each individual rule, we hereby specify an ideal construct that measures the capital gain if we executed the trading order following exactly the signals on each trading day compared to the day before. To express the returns as percentages, we adopt the multiplicative profit representation from [2]. Thus on day t, a component rule i has return $r_{t,i}$ where

$$1 + r_{t,i} = [1 + (1 - |S_{t-1,i}|)R_t^f + S_{t-1,i}R_t] \times [1 - \delta|S_{t,i} - S_{t-1,i}|] \tag{1}$$

Consider on trading day t, each component rule i indicates a signal $S_{t,i} \in \{long, neutral, short\} = \{1, 0, -1\}$, which is a discretized categorical variable according to some threshold parameters. With P_t being the asset price at time t, $R_t = (P_t/P_{t-1}) - 1$ is the asset price return. The risk-free rate of interest R_t^f is calculated similarly. δ denotes the transaction cost rate. The rationale behind is to additionally hold risk-free assets (e.g. U.S. Treasury bills) when the combined signal indicates *neutral*. In contrast, conventional RRL adopts overall return as $\mu \cdot (F_{t-1} \cdot r_t - \delta|F_t - F_{t-1}|)$, where μ is the maximum possible number of shares per transaction, r_t is the simple difference of prices $p_t - p_{t-1}$ and F_t is the trader function. Equation 1 is therefore not implemented in this study to ensure consistency of comparing RbRRL with RRL.

2.2.2 Signal weighting approximation

The trader function F_t in RRL resembles a neuron activated with the hyperbolic tangent function yielding outputs between -1 and 1. The value of F_t determines the current action.

$$F_t = tanh(\theta^T x_t)$$

where parameter $\theta \in \mathbb{R}^{m+2}$, input vector $x_t = [1, r_t, \ldots, r_{t-m}, F_{t-1}]$, m is the number of time series inputs to the trader.

In connection to integrating multiple indicator signals, we assign each component rule i a weight $\omega_{t,i}$ at time t. A system signal is then generated as the weighted average of individual signals. Inspired by the nonlinear framework of approximating trading action (policy) in [2], we contextualize the core idea into the current setting and propose the following approximation for individual signal weights.

$$\omega_{t,i} = softmax[\langle \mathbf{v}, \mathbf{x}_{t,i} \rangle + b + u\omega_{t-1,i}] \tag{2}$$

In the bracket of 2, $\langle \cdot, \cdot \rangle$ is the inner product, $\mathbf{x}_{t,i}$ defines the feature vector of the current market condition at time t for rule i, and $(\mathbf{v}, b)$ are the feature regression coefficients. We adopt the recent m rule returns as the feature vector, so as to extract information of each rule by quantifying the counterfactual capital gains in the case that individual rules were used for trading

$$\mathbf{x}_{t,i} = [r_{t-m,i}, \ldots, r_{t-1,i}] \in \mathbb{R}^m$$

In addition to the features, another term $u\omega_{t-1,i}$, where $\omega_{t-1,i}$ stands for the rule weighting at time $t-1$, is also added into the regression to take the latest weighting into consideration. This term is supposed to discourage the agent from frequently changing the trading positions and, hence, to avoid heavy transaction costs. With the linear transformation in the brackets, $softmax(\cdot)$ further ensures that weights to each trading rule satisfy the constraint $\sum_i^{NR} \omega_{t,i} = 1$, where NR is the total number of rules. The optimization of the RbRRL learns in part the parameter set $\Theta = \{\mathbf{v}, b, u\}$ that maximizes the global reward function.

2.2.3 Regularization

As aforementioned, PRS generates the system signal S_t simply by $S_t = \sum \omega_{t,i} S_{t,i}$. Nevertheless, this may result in the case that many under-performing rules dominate the few truly useful rules due to adding many yet small weights. To address the possible noise, we only select the best rule (that is assigned the highest weighting) each time within each rule class and combine them across different classes. For instance, in the predefined set of rule classes $C = \{\text{MA, TRB}\}$, we have

$$S_t = \begin{cases} 1 & \text{if } \alpha \sum_{k \in C} \omega_t^k S_t^k + (1-\alpha) S_{t-1} > l \\ -1 & \text{if } \alpha \sum_{k \in C} \omega_t^k S_t^k + (1-\alpha) S_{t-1} < s \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where ω_t^k and S_t^k represents the weight and signal of the best rule within class k at time t, respectively. In particular, ω_t^k has to be re-balanced through $softmax$. A new decay parameter $\alpha \in [0, 1]$ is introduced to incorporate the previous trading decision for the similar cost-reducing purpose. Two threshold parameters $l \in [0, 1)$ and $s \in (-1, 0]$ are utilized to discretize the signal and filter out possible noises.

2.3 Optimization

The set of parameters to be learned is $\{\Theta, \alpha, l, s\}$, where $\Theta = \{\mathbf{v}, b, u\}$

The Sharpe ratio is our objective function, which the trader will attempt to maximize. It essentially identifies investment strategies with less volatile profit.

$$S_T = \frac{Average(R_t)}{Standard\ deviation(R_t)} \qquad \text{for } t \in \{1, 2, \ldots, T\}$$

Particle Swarm Optimization was initially tried out because it does not assume differentiability as is required by classic optimization methods such as Newton's method. Automatic differentiation (and thus numerically approximating gradients) seems appealing but the Python package *Autograd* currently does not support certain functions like array assignment. Therefore, a more straightforward random optimization with clipped boundaries is adopted.

3 Results

Figure 1 and 2 demonstrate the two technical indicators of stock Amazon (NASDAQ: AMZN) over 4000 trading days. We can see that SMA (5-day vs 150-day) signals much less frequently compared to TRB (20-day), representing different decision making rationales which we aim to combine.

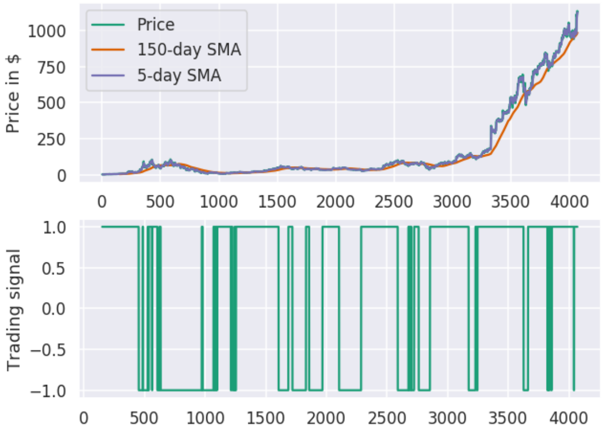

Figure 1: Simple Moving Average

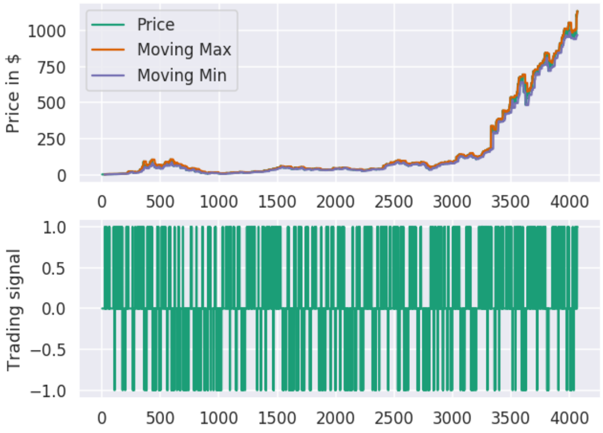

Figure 2: Trading Range Break-out

Comparing the automated trading decisions and returns side by side shows that overall RRL still outperforms RbRRL and the buy-and-hold strategy. The discretized action policy by RbRRL in 6 could miss out on plausible trading opportunities of different lot sizes. The reasonable performance in the training set considering transaction costs turns out to be overfitting as suggested by the poor testing set outcome 8. However, figure 3 and 4 display that despite its higher overall gain, RRL brings greater volatility on a daily basis.

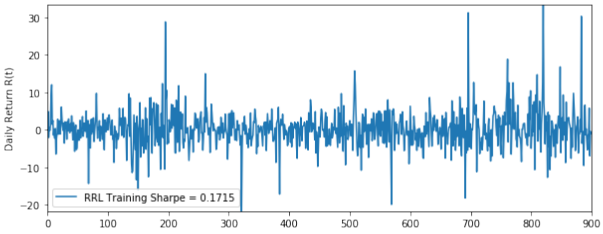

Figure 3: RRL Daily Return

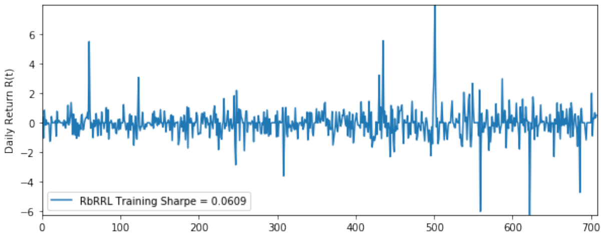

Figure 4: RbRRL Daily Return

4 Discussion and future work

It is not a full-fledged implementation because the model was only run on a single stock AMZN over a fixed period and parameters together with optimization could be better tuned. Financial markets are not necessarily stationary and thus the selected training period may not correctly represent the value of both models.

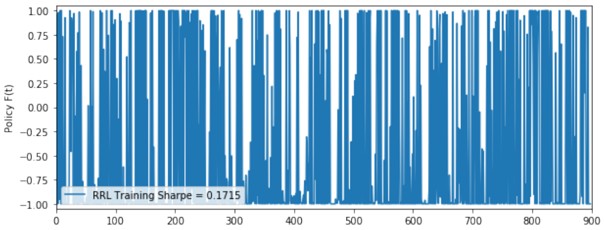

Figure 5: RRL Trading Policy

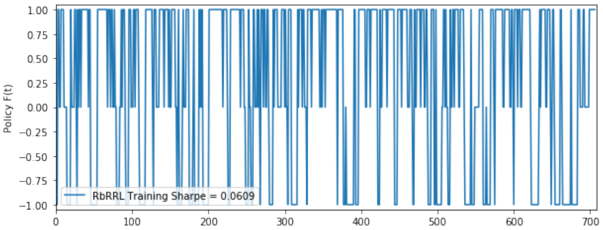

Figure 6: RbRRL Trading Policy

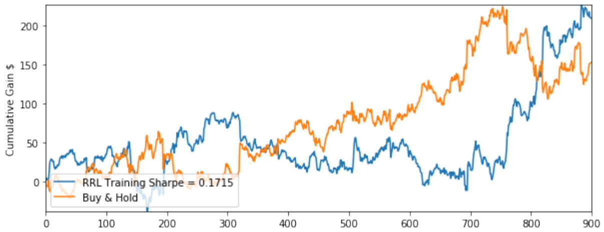

Figure 7: RRL Cumulative Gain

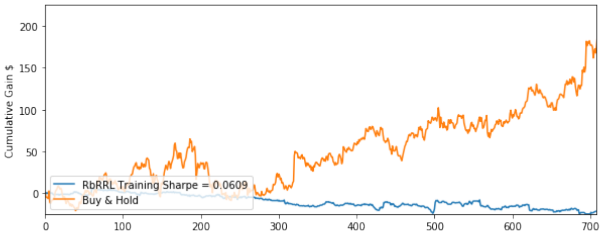

Figure 8: RbRRL Cumulative Gain

If the RbRRL is to be extended to a portfolio, we could first try the following trading logic: each stock is traded independently with equal initial portfolio weights and no mutual funding is allowed as that in [1]. Furthermore, stock selection (e.g. by low correlation), risk diversification and trading frequency control could be considered.

Since RRL is in fact a one-layer neural network [4], deep learning techniques specifically recurrent neural network could be worth exploring. Nonetheless, one caveat is that efforts to increase the model interpretability such as this study trying to bridge machine intelligence and human insight should not be underestimated in practice.

Acknowledgement

Special thanks to Dr. Philip L.H. Yu who provided insightful ideas and patient guidance.

GitHub Link to Code

`https://github.com/AndyYFTao/RbRRL-strategy-for-financial-trading/blob/master/RbRRL.ipynb`

References

[1] Wang, F. , Philip, L. , Cheung, D. W. *Combining technical trading rules using particle swarm optimization.* Expert Systems with applications, 41 (6), 3016–3026, 2014.

[2] Moody, J. , Wu, L. , Liao, Y. , Saffell, M. *Performance functions and reinforcement learning for trading systems and portfolios.* Science, 17 (February 1997), 441–470, 1998.

[3] S. Almahdi, S. Y. Yang. *A constrained portfolio trading system using particle swarm algorithm and recurrent reinforcement learning.* Expert Systems With Applications, 130:145–156, 2019.

[4] Y. Deng, F. Bao, Y. Kong, Z. Ren and Q. Dai. *Deep Direct Reinforcement Learning for Financial Signal Representation and Trading.* IEEE Transactions on Neural Networks and Learning Systems vol. 28, no. 3, pp. 653-664, March 2017.