Technical report 1–14, 2015

Trading Strategy Management via Recurrent Neural Networks

Baoxiang Wang BXWANG@CSE.CUHK.EDU.HK
The Chinese University of Hong Kong

Abstract

Equity algorithmic traders aim to make a profit by executing the most profitable strategies. In this process, how to select strategies from a number of candidates is a key question. However, there have been very few studies on the strategy selection problem, and most of them were in an offline setting, partly because of the difficulty in modeling the delayed reward of selecting a strategy. Borrowing techniques from equity back-testing, we formulate strategy selection as a regret minimization problem. We then present *StrategySel*, an algorithm solving the problem via recurrent neural networks (RNNs) method. Experiments on stock trading transaction data demonstrate that *StrategySel* can assemble multiple trading strategies and outperform all of them, under a wide variety of settings.

Keywords: List of keywords

1. Introduction

As techniques in computer science and finance engineering continue to evolve, algorithmic trading, short as algo-trading, has been a more and more prevailing practice. Indeed, it contributed 70% of the volume in transactions on U.S. stock markets by the year of 2010, as reported by Securities and Exchange Commission U.S. Commission (2010), and an increase of algo-trading on Asian stock exchange markets is also witnessed in recent years Moosa and Bora Ramiah (2014). A large body of existing work on algo-trading focus on designing trading strategies, which decide what actions to take based on real-time market information Obizhaeva and Wang (2013). Very few strategies can always perform well, and thus an algo-trader often has a large number of base strategies to select from. However, for every trading strategy the trader owns, its entry/exit points and its money management rules need to be further defined. It would be beneficial if the most suitable strategy in the short future could be automatically selected, of which an example is shown in Fig. 1. The strategy selection approach is expected to attain optimal selection policy without human decision, in order to avoid subjectivity and reduce latency Hasbrouck and Saar (2013). However, few previous studies have been conducted on the strategy selection problem, and it remains an underexplored field in both academia and industry.

There have been some studies on strategy evaluation Harvey and Liu (2014); Lopez de Prado (2014), which leads to strategy selection policies. However the studies are in an offline framework, in which the select engines are developed on historical datasets. The systems developed in this manner are not able to adapt quickly according to real time market information, also they are prone to overfitting. This limits the applicability because to react in real time is a must in practical trading. In this work, we aim to address the strategy

Figure 1: **Illustrative output of a trading strategy selection system.** The two range indicators on top of the candlestick chart of Google Inc. Nasdaq, show the recommended trading strategies during from Sep. 2014 to May 2015. In this example, our model selects strategy *MACD Divergence* and *Mean Reversion* during the corresponding period in the figure, and selects not to execute any strategy during other periods.

selection problem in an online model. Many online learning methods and tools are available, such as multi-armed bandit Bubeck and Cesa-Bianchi (2012) and online optimization Hazan et al. (2006). However those methods usually rely on the assumption that the reward of an action be immediately observed, which is violated in strategy selection, as we by no means can evaluate the performance of a trading strategy right after its execution. Moreover, it is also expected to avoid involving empirical analyses of a strategy, especially when the strategy itself is complex, as those analyses could gradually become obsolete as the market itself evolves. All these facts make the strategy selection problem a challenging task.

To address the challenges mentioned above, we firstly propose a reward function of a strategy, which measures its profitability. The reward is computed based on the back-testing result on future market data. In this way, as the trading strategies are automatically applied to future market data, our approach involves no empirical analyses of strategies, and hence induces no subjectivity. While doing back-testing on future data yields huge delay between the entry point of a trading strategy and the time its full performance is returned, known as delayed rewards, we calibrate the reward as an alternative profitability measurement which is gradually observed as time goes by from the entry point. The regret of a strategy is then defined as the negative difference between its reward, and the highest reward of all base strategies, and the strategy selection problem is formulated into a regret minimization form.

Recurrent neural networks (RNNs), which is a recent advanced models in deep learning, is successful in sequential data processing. We propose *StrategySel*, an RNN architecture to solve the regret minimization and to predict the most profitable strategies. Our approach were tested on transaction data of Shanghai Stock Exchange (SSE) and Shenzhen Stock Exchange (SZSE), yielding convincing and stable results, where the profitability of our proposed model is constantly better than all of its underlying base strategies.

Our main contributions can be summarized as follows:

- We formulate the strategy selection problem into an online optimization problem
- We solve the problem using proposed RNN architecture, which provides a dynamic and automatic approach to strategy selection

2. Related Work

2.1. Machine Learning for Equity Data Processing

Machine learning methods, especially RNN, are popular in equity data processing, including forecasting prices and movements Dase and Pawar (2010), recognizing and classifying certain patterns Kamijo and Tanigawa (1990), and developing trading policies Nevmyvaka et al. (2006), etc. A large volume of publications have covered different settings in price forecasting and policy developing, including countries, markets, and sections etc. Most of the work focus on constructing input vectors via feature engineering, and building novel learning models. There are also cross-domain approaches taking texts from news and social networks as input, according to the studies on Twitter posts Bollen et al. (2011) and on news from Reuters and Bloomberg Ding et al. (2015), respectively.

Previous studies via online learning models are mostly conducted on low-level decision makings, where the actions (e.g. fulfill the existing limit orders) could affect the trading environment. A famous example is the *one-way trading* problem, where the agent takes actions to sell out all its shares before the given time horizon and maximizes the revenue received. However online settings, where a system must make decisions and adapt its policy at the same time, are also preferred by high-level applications such as day trading. This is mainly because of the dramatic exploration-exploitation tradeoff in real-life trading problems, especially when the system is managing a large amount of assets Dempster and Leemans (2006).

2.2. Trading Strategies

Trading strategies are key components of algo-trading systems, and thus have drawn many attentions from both industry and academia. The complexity and profitability of a trading strategy vary in a wide range, while highly profitable ones are always kept confidential in industry. To this end, when the development of complex strategies is not affordable, selecting the most suitable trading strategy at the moment is one way to compensate the lack of profitability. In studies related to trading strategies, the logical combinations of technical analysis indicators could be used as examples of base trading strategies Chan et al. (2014). Indeed, in this work we will use these basic strategies and show that a good selection among them can achieve a decent gain.

Many studies focus on interpreting the strategy and choosing the most suited strategy in the short future, but those approaches are out of the scope of this paper. In fact, on one hand, the interpretations of trading strategies are partially subjective, which is in contrast with the basic ideas of algorithmic trading; On the other hand, some proficient trading strategies, especially for those bought from other institutions, may appear black-boxed, which means no interpretation can be conducted. Though these studies are interesting, we will not be able to utilize them in this paper.

Some previous studies have been conducted to build trading strategies via machine learning. By the year of 1999, Allen et al. have developed a way to automatically learn the rules in a trading strategy using genetic algorithm Allen and Karjalainen (1999). With Q-learning, Liao et al. proposed to generate trade signals in an online learning perspective Liao and Chen (2001). Moody et al. in 2001 addressed the problem in a pure reinforcement learning way, where signals generated by the strategies were regarded as actions responding to the observations from the environment Moody and Saffell (2001). Those works are not in the setting as in this paper, while the trading strategies constructed by machine learning approaches could also serve as elements in the set of base strategies.

3. Method

We firstly introduce the mathematical formulation of a trading strategy and its reward, and formulate the strategy selection problem into a regret minimization problem. We then show that the delayed reward phenomenon is the main challenge of the optimization, and address it by applying time discount factor to the original regret. Finally we introduce our learning model based on Long Short-Term Memory (LSTM) units which minimizes the cumulative regret. The set of base trading strategies and the asset we are trading on are regarded as constant in this section.

3.1. Formulation of Strategy Selection Problem

In equity trading environments such as a stock exchange market, a trading strategy can be characterized by a function transforming cumulative historical information to an action. We discretize the time into intervals, and assume that the information observed from the market during each time interval can be encoded by a d-dimensional vector. The set $\mathcal{S}$ of all valid trading strategies is

$$\mathcal{S} = \{S | S : (\mathbb{R}^d)^* \to \mathcal{A}\},$$

where $\mathcal{A}$ is the set of all feasible actions and $(\mathbb{R}^d)^*$ is the set of arbitrary-length sequences of d-dimensional vectors. Note a trading strategy S specifies an action for all time intervals. Let $\mathcal{I} \subseteq \mathcal{S}$ be the set of given base trading strategies. Note that the strategies are black-boxed, meaning that we have no access to the definition of function $S(\cdot)$ for $S \in \mathcal{I}$, neither can we get any property of it; all we can do is to execute strategy S and to get an output $S(\cdot)$.

For example, in a stock exchange market, the duration of a time interval is the minimum time the trader takes to react to the market and make decisions, which could be one second, one millisecond, or otherwise. The set $\mathcal{A}$ of all feasible actions consists of putting a limit order, putting a market order, canceling an existing limit order, and holding whatever the agent owns without further moves, etc. In some practical scenarios, the base strategies are indeed black-boxed, e.g. when they are bought from other institutions without source code delivery. Strategy selection approaches in real-life are expected to handle this case for most of the time.

We utilize back-testing technique on our base trading strategies. The back-test system takes a base strategy S as input and simulates it on the asset, which we are trading on, starting with unit net worth. Let $v_t(S)$ to be the net worth of the agent's portfolio at the end

of the t-th time interval, given that the agent follows strategy S for all its actions. Note that portfolio is the grouping of whatever with monetary value the agent's possesses, including currency, shares, and options, etc. Also define $v_k(S)|_t$ and $v_k(S)|_{\leq t}$, to be the value of the portfolio at the end of the k-th time interval, given that the agent follows strategy S only at time t and only at or before time t, respectively, and performs no action otherwise. As the computing of $v_k(S)$, $v_k(S)|_t$ and $v_k(S)|_{\leq t}$ is based on the market information up to time k, they can only be computed at the beginning of the $(k+1)$-th time interval.

In most applications, the reward of the agent's actions is its expected capital gain. Motivated by this fact, we define the reward $r(S)$ of executing strategy S throughout time 1 to horizon T as the relative growth of the value of the agent's portfolio (recall $v_0(S) = 1$)

$$r(S) = v_T(S) - v_0(S).$$

To model an agent adapting its policy by following different trading strategies over time, we firstly consider the fact that a sequence of trading strategies compose a new trading strategy. In fact, if the agent chooses strategy $S_t \in \mathcal{I}$ and follows its output at time t, the agent also transforms cumulative historical information to an action. Denote the composed strategy as $\hat{S} \in \mathcal{S}$, where $\hat{S}(x) = S_t(x)$ for $x \in (\mathbb{R}^d)^t$, $1 \leq t \leq T$, and $\hat{S}(x)$ returns arbitrary value for $x \in (\mathbb{R}^d)^t$, $t > T$. The agent is trying to maximize the reward of its composed trading strategy, i.e.

$$\max_{\{S_t : 1 \leq t \leq T\} \subseteq \mathcal{I}} r(\hat{S}). \tag{1}$$

The maximization in Formula (1) is hard to solve in general, but we can approximate it using cumulative rewards. To achieve this, we define the reward of strategy S at time t, as

$$r_t(S) = v_T(S)|_t - v_{t-1}(S)|_t.$$

A desirable property of $r_t(S)$ is its additivity, i.e. the cumulative reward over time $\sum_{1 \leq t \leq T} r_t(S) = r(S)$. In fact,

$$\begin{aligned}
\sum_{1 \leq t \leq T} r_t(S) &= \sum_{1 \leq t \leq T} (v_T(S)|_t - v_{t-1}(S)|_t) \\
&= \sum_{1 \leq t \leq T} \sum_{t \leq k \leq T} (v_k(S)|_t - v_{k-1}(S)|_t) \\
&= \sum_{1 \leq k \leq T} \sum_{1 \leq t \leq k} (v_k(S)|_t - v_{k-1}(S)|_t) \\
&= \sum_{1 \leq k \leq T} (v_k(S)|_{\leq t} - v_{k-1}(S)|_{\leq t}) \\
&= \sum_{1 \leq k \leq T} v_k(S) - v_{k-1}(S) = r(S).
\end{aligned}$$

The second to last equation holds by the assumption that the actions have no impact on the trading environment, which is true under a wide variety of real-life settings. With the additivity, we have

$$r(\hat{S}) = \sum_{1 \leq t \leq T} r_t(\hat{S}) = \sum_{1 \leq t \leq T} r_t(S_t),$$

given that $\hat{S}$ is composed of $(S_t)_{1\leq t\leq T}$. Formula (1) could then be deduced into a regret minimization problem

$$\min_{\{S_t:1\leq t\leq T\}\subseteq\mathcal{I}} \sum_{1\leq t\leq T} (\max_{S'\in\mathcal{I}} r_t(S') - r_t(S_t)), \tag{2}$$

for which we can resort to online learning methods.

3.2. Addressing Delayed Rewards

Different from typical applications in online learning, such as movie recommendation, network routing, etc, where the rewards are immediately observed after the actions, the term $\max_{S'\in\mathcal{I}} r_t(S') - r_t(S_t)$ can only be fully observed at time $T+1$ i.e. at the end of the whole trading period. However in equity trading, the agent is required to perform policy adaption quickly according to the dynamics of the environment. Consider that

$$\begin{aligned} r_t(S) &= v_T(S)|_t - v_{t-1}(S)|_t \\ &= \sum_{k=t}^{T} (v_k(S)|_t - v_{k-1}(S)|_t), \end{aligned} \tag{3}$$

the agent is able to utilize partially observed reward $\sum_{k=t}^{k'-1}(v_k(S)|_t - v_{k-1}(S)|_t)$ by time $k' \in [t+1, T]$.

In order to learn under delayed and partially-observed rewards, we apply discount over time to the reward function. With time discount factor $\gamma \in (0, 1]$, the reward in Eq. (2) is replaced by its time discounted version

$$r_t^\gamma(S) = \sum_{t\leq k\leq T} \gamma^{k-t}(v_k(S)|_t - v_{k-1}(S)|_t),$$

yielding a new regret

$$\min_{\{S_t:1\leq t\leq T\}\subseteq\mathcal{I}} \sum_{1\leq t\leq T} (\max_{S'\in\mathcal{I}} r_t^\gamma(S') - r_t^\gamma(S_t)). \tag{4}$$

With the portion $\sum_{k=t}^{k'-1} \gamma^{k-t}(v_k(S)|_t - v_{k-1}(S)|_t)/r_t^\gamma(S)$ of reward observed by time k' dramatically increased, the model is released from its suffering of delayed rewards.

Apart from the fact the discount factor γ helps training, it actually improves the result of the model as well. In fact, the value $\gamma^{k-t}(v_k(S)|_t - v_{k-1}(S)|_t)$ is significant only for those k that are closed to t. When minimizing the time discounted regret, the difference between $v_k(S)|_t$ and $v_{k+1}(S)|_t$ becomes negligible for $k >> t$. In another word, the arbitrages are incented to happen soon after the strategies are executed. Taking stock markets as an example, the agent receives a high reward if its going long is followed by a bullish immediately, and a relatively low reward if the bullish happens a while after the action. The agent thus tends to hold its portfolio until the chance of arbitrage is right in front of it, of which the behavior is usually desired by traders.

3.3. RNN for Strategy Selection

We now present *StrategySel*, the learning model to solve strategy selection. Considering the probabilistic nature of the problem, the learning model is defined to make soft predictions, i.e. the model selects a strategy S_t from $\mathcal{I}$ at probability $\mathbb{P}(S|t)$ at time t. The loss function, which measures the expected cumulative regret, is

$$J = \sum_{1 \le t \le T} (\max_{S' \in \mathcal{I}} r_t^{\gamma}(S') - \mathbb{E}_{S \sim \mathbb{P}(S|t)}[r_t^{\gamma}(S)]).$$

The regret observed at time t, is

$$j_t = \sum_{1 \le k \le t} \delta_{t,k}, \tag{5}$$

where

$$\begin{aligned} \delta_{t,k} = \gamma^{t-k} \{ & \max_{S' \in \mathcal{I}} v_t(S')|_k - \max_{S'' \in \mathcal{I}} v_{t-1}(S'')|_k \\ & - \mathbb{E}_{S \sim \mathbb{P}(S|k)}[v_t(S)|_k - v_{t-1}(S)|_k] \} \end{aligned}$$

We leverage the prevailing and general-purpose sequence modeling, *Long Short Term Memory* (LSTM) Hochreiter and Schmidhuber (1997), together with the idea of *forget gate* Gers et al. (2000), to predict the strategy with the least regret. An LSTM unit could be formalized as a function which updates its output h_t and its internal memory c_t using the current input x_t according to the following rules.

$$\begin{aligned} h_t &= o_t \circ \tanh(c_t) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo} x_t + W_{ho} h_{t-1} + W_{co} \circ c_{t-1} + b_o) \\ f_t &= \sigma(W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\ i_t &= \sigma(W_{xi} x_t + W_{hi} h_{t-1} + W_{ci} \circ c_{t-1} + b_i). \end{aligned}$$

Here $\circ$ is Hadamard product, σ is a sigmoid function, $(x_t)_{t \ge 1}$ denotes the input sequence of the LSTM unit, and o_t, f_t, and i_t are the output of *output gate*, *forget gate*, and *input gate*, respectively. LSTM has a great advantage of being able to learn both long-term and short-term dependencies, which is needed for processing equity market data.

To facilitate the model with a deeper understanding of the dynamics of the trading environment, we use two concatenated LSTM units, where the output h_t of lower one serves as the input x_t of the upper one, as illustrated in Fig. 2. The input x_t of the lower LSTM unit is z_t, which carries the market information at time t. The historical information before time t is kept in the memory cells of both the LSTM units. The output value h_t of the upper LSTM unit is used to calculate the probability distribution over base strategies. For $S \in \mathcal{I}$, we have $\log(\mathbb{P}(S|t)) = W_S^T h_t - \log C$, where $C = \sum_{S \in \mathcal{I}} \exp(W_S^T h_t)$ is the normalizing constant. One dedicated vector W_S is learned for each strategy, which means the size of the fully connected (FC) layer equals to $\mathcal{I}$'s cardinality.

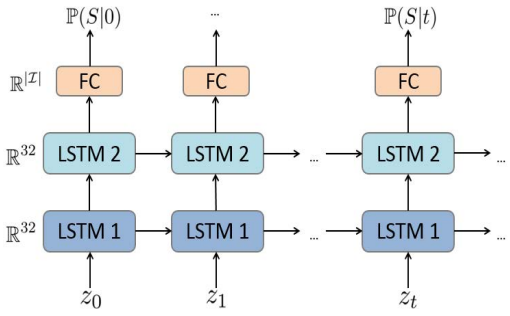

Figure 2: Proposed RNN Architecture

We apply stochastic gradient descent and standard backpropagation on $(z_t)_{1\leq t\leq T}$, i.e.

$$W := W - \alpha_t \sum_{1\leq k\leq t} \frac{\partial \delta_{t,k}}{\partial W}|_{x_t=z_k}.$$

W in the above equation represents any parameter to be learned in our model. For delayed rewards, we retrieve their corresponding input $z_k, z_{k-1}, \ldots$ to substitute $x_t, x_{t-1}, \ldots$ of the lower LSTM unit during gradient calculation, while we keep W to be the current value. This is because the variation of W is limited by the learning rate, and can be ignored in Eq. (5). The learning rate α_t of our model is decreased linearly during the first 30 time intervals, and is kept constant afterwards, to maintain the model's ability to quickly adapt its policy. The network is trained over a set of assets simultaneously, where the model parameters are shared over them and the loss are averaged.

We use Eq. ((**??**)) as the loss function. In training, if the model reaches the end of a sequence and we are going to make prediction on a new sequence, the values of memory units on both LSTM layers are set to zero, which means the model has no tell about a sequence with no history.

4. Experiments

We firstly evaluated the reward of the proposed models on transaction data of Shanghai Stock Exchange (SSE) and Shenzhen Stock Exchange (SZSE). The evaluation was conducted under a daily trading periodicity, where each time interval spans one day. The result was then compared with the rewards of base strategies, an approach based on strategy evaluation, and the perfect play. The performance verifies the effectiveness of our proposed *StrategySel* model. We also leveraged the probabilistic nature of our model to show its ability to handle the risk-averse case, which is a common scenario in trading systems. Our proposed approach was then tested under a minutely periodicity. Armed with a set of proficient base trading strategies, our trading system showed consistent and convincing performance. Finally we investigated on how to find the trading frequency with the most market inefficiency, straightforwardly using *StrategySel*. The results of the experiments lead to the following findings:

- The proposed model conducts lower regret and hence better profitability comparing to systems with single trading strategy and approaches based on strategy evaluation.

Model	**Average Reward** $\mathbb{E}[r_t^\gamma(S_t)]$
StrategySel	0.2373
Evaluation Approach	0.0915
Sleeping	0
Moving Momentum	-0.0004
RSI Threshold	-0.0064
MACD Divergence	-0.0045
CCI Correlation	-0.0100
Mean Reversion	-0.0100
Perfect Play	0.6320

Table 1: The average rewards of proposed model and existing models. The table compares the regrets of proposed models, 6 trading systems with single strategy, an empirical approach via back test and the oracle. Average reward in the table represents $\frac{1}{T}\sum_{1\leq t\leq T} r_t^\gamma(S_t)$, averaged over all symbols

- The proposed model is capable to handle the market with different markets, risk tolerances, and frequencies etc.
- It provides a new idea to determine the frequency at which the trades should take place.

4.1. Test on Daily Periodicity

We firstly tested our model using the daily transaction data (known as swing trading) recorded on Shanghai Stock Exchange (SSE) and Shenzhen Stock Exchange (SZSE), with 1455 symbols spanning the year of 2014. The transaction cost charged during the buy and sell executions, was estimated using the sum of average fees and taxes, which was 0.17% of the transaction value. We built 6 strategies into $\mathcal{I}$ using technical analysis indicators introduced in Chan et al's book Chan et al. (2014). The models go through the dataset $(z_t)_{1\leq t\leq T}$ with time horizon $T = 262$, Where the market information $z_t \in \mathbb{R}^5$ is encoded by a 5-dimensional vector, derived from the day's return, adjusted closing price and opening price, volume, and volatility.

During testing, the time discount factor γ was set to 0.90 throughout the experiments, which halves the rewards induced by transactions 7 days after the strategy's execution. The model was evaluated in an online perspective, where the model makes decision at each round, and those decisions directly contribute to the regret defined in Eq. (4). The average rewards are shown in Table 1.

As shown in the table, the profitability of any single strategy was compromised by the transaction cost. In contrast, decent rewards were observed in our method. Also, *StrategySel* outperformed the evaluation approach based on back-test, which selects the best performed strategy on latest historical data. The rewards of *StrategySel* were within the same scale

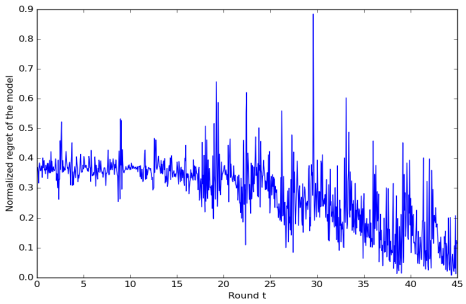

Figure 3: Training curve tracking the regret. Note that the cost happens at time t is normalized by the best possible reward $\max_{S' \in \mathcal{I}} r_t^\gamma(S')$

of the perfect play's, where the perfect play is the oracle selecting optimal strategy at any time. Over a wide variety of settings throughout our experiments, *StrategySel* consistently outperforms all of its underlying base trading strategies, due to its more complex structure and its architecture designed for sequence modeling, as well as its online and dynamic settings. As

$$\max_{S' \in \mathcal{I}} \sum_{1 \le t \le T} (r_t^\gamma(S') - r_t^\gamma(S_t)) < 0$$

always hold true throughout our experiments, our approach is concluded to be effective and useful in a trading system.

The training curve is illustrated in Fig. 3. The normalized cost in Eq (5), $1 - \mathbb{E}_{S \sim \mathbb{P}(S|t)}[r_t^\gamma(S)] / \max_{S' \in \mathcal{I}} r_t^\gamma(S')$, after the first 20 rounds where the strategies were inactive, decreased to 0.2 or less. Considering the time horizon was 262 in the experiment, the speed of convergence is quite acceptable.

4.2. The Risk-Return Tradeoff

A trading system is usually required to make risk-averse moves, and one common way to achieve this is by reducing the number of selections made during the trading period. To cater this, we appended strategy *sleeping*, which constantly generates no action, to $\mathcal{I}$. The model selects *sleeping*, until another strategy S is predicted with probability $\mathbb{P}(S)$ more than the predefined threshold. While the original model outputted a strategy other than *sleeping* 85% of the time, our method with the probability threshold set as 0.9 turned about 47% of the those into *sleeping*. The tradeoff between the profitability and risk is shown in Table 2.

4.3. Test on Minutely Periodicity

StrategySel was eventually tested on a day trading setting, where the duration of each time interval is set to one minute. The learning environment was different from previous settings in many ways. Firstly, the movement of price in a few time intervals was likely to be weak, hence the trading system was expected to make more moves. Secondly, because of the

Threshold	#Non-sleeping	Average Reward
N/A	85%	0.24
0.4	83%	0.24
0.6	78%	0.24
0.8	66%	0.22
0.9	45%	0.19

Table 2: The results of risk-averse play

Model	Average Reward
StrategySel	0.0101
Base Strategies	0.0049∼0.0075
Evaluation Approach	0.0065
Perfect Play	0.0217

Table 3: The rewards tested on minutely periodicity, armed with proficient base strategies

restrictions on Chinese stock exchange markets, such as a share can be traded only once a day, the market dynamics were more complex. Also the base strategies in $\mathcal{I}$ were advanced and confidential ($\mathcal{I}$'s cardinality is still known). Our model was challenged to assemble those advanced strategies and achieve an even better performance, of which the settings were closer that of a practical trading system.

We used data from transaction records in 2014. The time horizon was set to $T = 240$, which is the number of minutes the markets were open for in a day. Also at the end of a day, the memory units was reset to its initial value. $z_t \in \mathbb{R}^{10}$ was constructed using the trend, variance, volume, and order book information within each minute.

The rewards are averaged over all symbols and all days in 2014, which are shown in Table 3. Our model outperformed any single strategy in $\mathcal{I}$, and the approach via strategy evaluation as well. Even when the environment changes a lot from the previous experiment settings, the model yielded convincing result, which demonstrates the applicability of our model to various conditions.

Comparing to Table 1, an interesting finding can be observed in Table 3: as the frequencies under which the agent makes decisions increases, the performance of our model gets closer to the oracle's. This is partially because that the market trending contributes less, when trading under relative high frequency, to the market inefficiencies, and the fact that our model does not emphasis on the importance of the market trending. As a result, our proposed approach may benefit from involving trend analysis Ding et al. (2015). We leave this interesting topic for future studies.

Strategy	Neg Regret	#Selected
MA_2	-0.03	15%
MA_3	-0.02	15%
MA_5	-0.03	9%
MA_7	-0.03	4%
MA_9	-0.03	5%
StrategySel	-0.01	N/A

Table 4: Performances on different frequencies

4.4. Solving Frequency Selection

We investigated on solving frequency selection problem. The frequency selection problem is to find the duration of the time interval (between consecutive z_{t-1} and z_t) so as to maximize the market inefficiency. Though it's hard to give an quantitative evaluation to market inefficiency directly, we address the problem using our model, straightforwardly. Let $\mathcal{I}$ be a set of Moving Average strategies, denoted by MA_d, which simply compare the average price over recent d days and over recent $2d$ days. Considering that MA_d utilizes the market inefficiency at the frequency of d time intervals, the performance of MA_d is highly correlated with the potential profit of trading on every d time intervals. When our strategy selection model selects a strategy from $\mathcal{I}$, the strategy's underlying frequency, indicated by d, would be favored by a trading system. As shown in Table 4, though the performances of MA strategies are similar, they are focusing on different potentials. In this way, assembling multiple MA strategies, *StrategySel* yielded a preferred MA strategy, and thus a preferred frequency, at each time. As directly concluded from the table, in general, higher frequencies or smaller d are favored in swing trading. We hope these findings could bring some new ideas into the classical problem, and address it in a new perspective.

5. Conclusion

We present *StrategySel*, which is an algorithmic and online solution for the trading strategy selection problem. In the approach, we have addressed both the difficulty of formulating the problem and handling with heavily delayed rewards. *StrategySel* yields consisting and convincing performance, and outperforms all of its base strategies as well as the approach based on strategy evaluation. It's also noticeable that our framework can be applied to a wide range of problems such as profitable frequency discovering, which worth the attentions of both academia and industry in the future.

Acknowledgments

To be assigned

References

Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of financial Economics*, 51(2):245–271, 1999.

Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Machine Learning*, 5(1):1–122, 2012.

Raymond Hon Fu Chan, Spike Tsz Ho Lee, and Wing-Keung Wong. Technical analysis and financial asset forecasting: From simple tools to advanced techniques. *World Scientific Books*, 2014.

Securities & Exchange Commission. Concept release on equity market structure. *Federal Register*, 75(13):3594–3614, 2010.

RK Dase and DD Pawar. Application of artificial neural network for stock market predictions: A review of literature. *International Journal of Machine Intelligence*, 2(2):14–17, 2010.

Michael AH Dempster and Vasco Leemans. An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552, 2006.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *AAAI*, pages 2327–2333, 2015.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

Campbell R Harvey and Yan Liu. Backtesting. *Available at SSRN 2345489*, 2014.

Joel Hasbrouck and Gideon Saar. Low-latency trading. *Journal of Financial Markets*, 16 (4):646–679, 2013.

Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *Learning Theory*, pages 499–513. Springer, 2006.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Ken-ichi Kamijo and Tetsuji Tanigawa. Stock price pattern recognition-a recurrent neural network approach. In *Neural Networks, 1990 IJCNN International Joint Conference*, pages 215–221. IEEE, 1990.

Pen-Yang Liao and Jiah-Shing Chen. Dynamic trading strategy learning model using learning classifier systems. In *Evolutionary Computation*, volume 2, pages 783–789. IEEE, 2001.

Marcos Lopez de Prado. Optimal trading rules without backtesting. *Available at SSRN 2502613*, 2014.

John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *Neural Networks, IEEE Transactions on*, 12(4):875–889, 2001.

Imad Moosa and V Bora Ramiah. The regulation of high-frequency trading: An asian perspective. *Handbook of Asian Finance: REITs, Trading, and Fund Performance*, 2: 153, 2014.

Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680. ACM, 2006.

Anna A Obizhaeva and Jiang Wang. Optimal trading strategy and supply/demand dynamics. *Journal of Financial Markets*, 16(1):1–32, 2013.