Transaction Behavior Prediction in Financial Markets with DF-Lens

anonymous

Abstract

We propose an inverse reinforcement learning method, called DF-Lens, for behavior prediction in financial markets, which automatically and efficiently estimates the policy of a trader from his/her transaction records. The interaction between traders and markets is modeled by an incomplete Markov Decision Process (MDP$\backslash$R$\backslash\theta$). We predict the trader's behavior based on the rationality of his/her previous transactions. In other words, one's actions are the optimal solution of his/her Q value function on the observed state with one's preference, called sensitiveness, considered. The sensitiveness is learned by fitting DF-Lens on historical transaction records. Utilizing sensitiveness, we build a promoted greedy based policy to predict transaction behaviors. Our experiments conducted on a real-world stock dataset with 2,269,480 trading records, validate the effectiveness of our proposed approach.

Introduction

The rapid growth of financial markets, e.g. the stock market, has attracted a huge number of traders. A trader's transaction records (e.g. buying or selling) consist of a triple including time-stamp, financial subject identifier, and action, e.g. 2018/7/3, buying 100 units of stock A at $\$19.1$. By matching a trader's records with the price and volume movement derived from market data, we can estimate the market condition before he/she performed the action. Furthermore, we can discover the underlying reasoning process behind one trader's action by learning his/her sensitiveness. The trader's sensitiveness, or trading sensitiveness, has been widely discussed but has not been well-defined quantitatively. Just like the sensitiveness in other gaming scenarios (Agrawal et al. 2016), trading sensitiveness reflects a trader's tendencies and preference on state features (e.g. technical index).

This work attempts to predict transaction behavior for an individual trader by estimating one's policy. Consequently, the trades can be performed in a similar way to a real trader's action. Transaction behavior prediction in financial markets has the following benefits.

- For junior traders, transaction behavior prediction can minimize the market risk by identifying the inappropriate habits. For example, some junior traders may get anxious when facing big losses and take inappropriate actions. Trading behavior prediction can mimic senior traders' behavior and point out the profitable opportunities (make suggestions).
- For instruments, trading behavior prediction is a scalable and stable method for replacing human traders (Le 2009). By cloning a senior trader, the trained policy can earn a similar profit rate. Such a trading policy system can cover thousands of stocks at the same time, while a human senior trader can only handle four or five financial products at one time. In addition, it does not suffer from mood swings and is more stable than humans.
- For market governors, trading behavior prediction can help to simulate the market by mimicking all practical traders. Based on the simulation, governors can easily evaluate the possible influence of a new market rule or policy.

In this work, to predict transaction behavior, we propose an inverse reinforcement learning method, called Deep Feature Lens (DF-Lens), which estimates a trader's policy with his/her sensitiveness on market features from transaction records. Given a set of transaction records from a trader, our proposed method can recover and perform the trader's evaluation under a specific market condition. We introduce the concept of a trader's sensitiveness as a high-level state representing his/her preference to different market state features. For evaluation of the method, we conduct experiments with both supervised and unsupervised learning methods on a real-world dataset containing 2,269,480 trading records from 1165 traders in China. Our first experiment compares the action prediction accuracy of our proposed method with baseline methods. The results demonstrate the effectiveness of our DF-Lens. In the second experiment, we analyze the sensitiveness of the 1165 traders in the dataset, showing that the traders with similar sensitiveness have similar trading behaviors and strategies. Here we summarize our major contributions.

- We propose an inverse reinforcement learning method, called DF-Lens, to predict traders' transaction behaviors in financial markets. The policy learned from trading records, shows great similarity with real traders. It is scalable to any financial product.
- We derive traders' sensitiveness from trading records, as a

high-level personalized feature to represent their tendencies and preference to market conditions.

- Experiments conducted on real-world stock market data, demonstrate how the method can be used in financial applications in practice.

This paper is organized as follows: Related techniques are reviewed in Section 2. Section 3 presents some basic concepts used in this work. Section 4 introduces our method and its major components in detail. We conduct experiments in Section 5 to show the effectiveness of our approach compared with baseline approaches. At last, section 6 concludes this work.

Related Works

Existing methods of behavior prediction can be roughly divided into two categories. The first is learning the policy as a classification problem. The second uses inverse reinforcement learning (IRL) to learn the reward function and expert policy.

Policy Classifier based Prediction

Policy classifier based prediction models, estimate the conditional probability of action with respect to the state through classification (Hussein et al. 2017).

A variety of approaches have been used to do this. Chernova et al. (Chernova and Veloso 2007) employed Gaussian Mixture Model to demonstrate imitation vehicle navigation. An artificial neural network was used to imitate robots (Vogt et al. 2014). Ross et al. (Ross and Bagnell 2010) applied support vector machine to the imitation problem in online-gaming. (Ho and Ermon 2016) trained a generative adversarial model as a policy classifier. (Levine and Abbeel 2014) used a neural network structure to learn an imitation policy.

The major drawback of these approaches is the difficulty in generalization and explanation. Because they approximate the policy with a simple classifier, e.g. SVM, trained from demonstrations directly. Some approaches use a neural network to build a non-linear mapping from state to action, but these are even hard to train. Compared with these works, our proposed approach does not learn the policy directly. Instead, we learn traders' sensitiveness which is more stable and explainable.

IRL based Prediction

Inverse Reinforcement Learning estimates a reward function using the action trajectory modeled by a Markov Decision Process (MDP). A detailed introduction about MDP can be found in the next section. Here, we review the literature that has used IRL for action prediction. Apprenticeship Learning, introduced in (Abbeel and Ng 2004), evaluates the differences between observation of an apprentice and a master. Then it tries to iteratively modify the strategy of the apprentice and evaluates once more, until the apprentice's trajectory is similar to that of the master. In 2010, Lee (Lee and Popović 2010) designed a framework to leverage IRL on a small volume of data. Abbeel et al. (Abbeel, Coates, and Ng 2010) applied IRL to Autonomous Helicopter Aerobatics. The trained controller performed similarly to a human pilot. There are lots of studies using IRL based behavior prediction for inverse optimal control (Ziebart, Dey, and Bagnell 2012)(Ratliff et al. 2009). In fin-tech, (Yang et al. 2012) modeled a trader using IRL in the simulated E-Mini S&P 500 and identified a high-frequency strategy. In (Yang et al. 2014; 2015), Yang et al. compared Linear IRL and Gaussian IRL in strategy identification. The work demonstrates the feasibility of using IRL for strategy modeling. However, (Yang et al. 2012; 2014; 2015) are limited to strategy identification and cannot support trading action prediction.

The major limitation of IRL in addressing trading prediction comes from two aspects. First, most IRL methods, e.g. max-entropy IRL (Ziebart et al. 2008), assume a discrete state space, while the features in the stock market, e.g. price, are represented by continuous real numbers. Second, the transition probability $Pr(s_t|s_{t-1}, a_{t-1})$ required as a key element in MDP at IRL, is not accessible and hard to estimate in financial markets (Markov and Tamayo 2006). Given these limitations, we take the basic concepts of IRL and introduce a new feature expectation based approach to estimate the personalized reward and policy. We construct a trading agent according to the learned reward and policy.

Concepts and Problem Definition

In this section, we introduce some basic concepts in MDP and the formalization on the transaction behavior prediction problem.

Incomplete MDP

MDP is a tuple $(\mathcal{S}, \mathcal{A}, \gamma, R, \theta)$ consisting of state set $\mathcal{S}$, action set $\mathcal{A}$, discount factor γ, reward function R and transition function θ. In a financial market, one trader is the *agent* and the market is the *environment*. The environment provides the *state*, denoted by $s \in \mathcal{S}$. In a financial market, the state s contains the features such as the price and trading volume. Observing the state, the agent chooses to take an *action* $a \in \mathcal{A}$. Typical actions in a financial market include buying/selling an object and holding. It is worth noting that the action set could be different in different markets and varies from one country to another. For simplicity, we generally discuss the action a. Once the action a is given, the environment will provide a new state and a *reward* $R(s)$ to the agent. However, in financial markets, this reward R is not accessible. For this reward-function-missed MDP, namely MDP\R, IRL can learn the reward R from demonstrated transaction trajectories by minimizing the difference between the value V of learned policy π and expert policy π^E. In addition, for the transaction behavior prediction problem, transition function θ is missed (MDP\R\$\theta$) and even hard to estimate (Markov and Tamayo 2006). In other words, most existing IRL approaches cannot solve this problem directly.

State Features in Transactions

As mentioned above, the basic state features, denoted as s^b in financial markets are the price and volume. In particular, the price feature contains the opening, highest, lowest and closing price, abbr. OHLC. In addition to the basic features, the traders usually use some technical indexes to evaluate the price trends. Typical technical indexes, denoted by s^t,

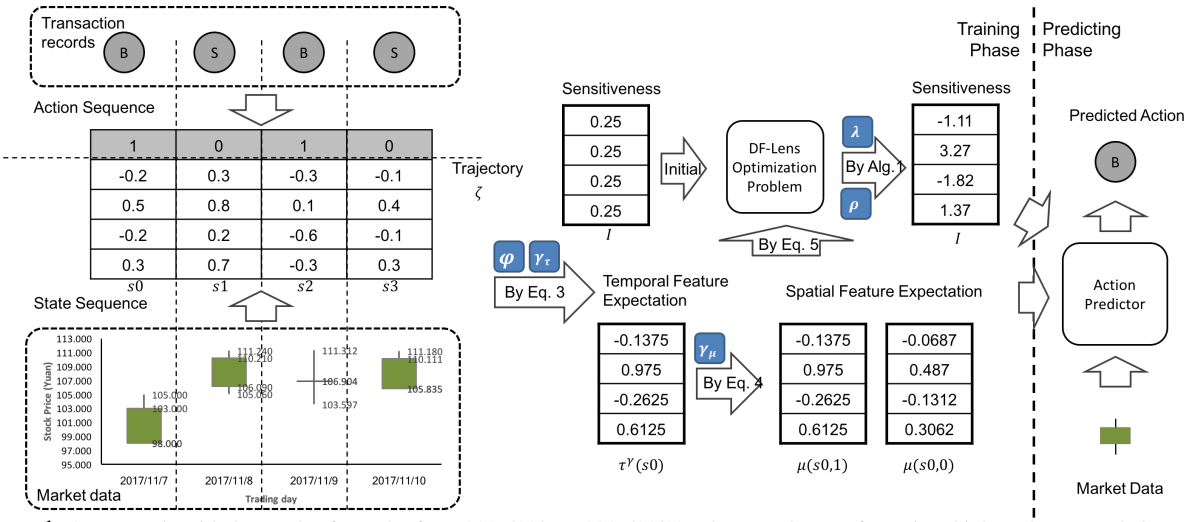

Figure 1: An example with 4 records of a trader from 2017/11/7 to 2017/11/10. The growth rate of opening, highest, lowest and close price (OHLC) are taken as features. The observed actions are buying (B), selling (S), buying (B) and selling (S). Extracting temporal and spatial feature expectation, we can solve the objective optimization and learn the sensitiveness. The prediction model is built on sensitiveness to forecast the transaction action by observing new market data.

include MA, MACD, KDJ, RSI, BOLL, W&R, ASI, BIAS and VR. Another kind of state features is the status of the traders, denoted by s^s. We consider three trader-related features: fixed profit, float profit, and position. Fixed profit is the whole earning rate from previous trading transactions. Float profit calculates the current float earning rate. Position feature is 1 if the trader holds a position and -1 otherwise.

Problem Definition

The problem we are tackling in this work is to predict a trader's action from his/her trading records. It is required to learn a policy that can closely approximate the trader's policy. A policy is a mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$, determining the action to take after observing a given state, whose stochastic form is $\Delta\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. Here is the formalization of this problem:

Definition 1 (Transaction Behavior Prediction Problem) *Given a set of trading records $\mathcal{D}_g = \{\boldsymbol{\zeta}_i\}_{i=1}^N$ from a trader $g \in \mathcal{G}$ where $\boldsymbol{\zeta}_i = \{(s_j, a_j)\}_{j=1}^T$ and incomplete MDP$\backslash R \backslash \theta = (\mathcal{S}, \mathcal{A}, \gamma)$, learn a (stochastic) policy π that maximizes the likelihood $\prod_{i=1}^{T*N} \Delta\pi(s_i, a_i)$.*

The major challenges of this problem come from the following two aspects.

First, policy classifier based prediction, behavior cloning, or statistical methods do not satisfy our requirements. It is not interpretable, extremely overfitting on observed states and not scalable to un-observed states. Consider a discrete state space with m-dimension features, each feature ranges from p values. There are p^m states in this space. If N trajectories whose length is T, are observed, the observed state cover rate is $N*T/p^m$. In the practical dataset, the cover rate is very small, for example, it is $1000 * 30/31^{20} = 4e-26$ in our dataset, and most states in prediction phase cannot be found in the training set.

Second, existing IRL approaches cannot perform well for the transaction behavior prediction problem. Because in financial markets, the trader related features are highly dependent on the trader's action. In particular, the status of a trader s^s at time t is affected by the trader action a_{t-1} and the previous state s_{t-1}. The other features, like the price and volume features s^b, and technical indexes s^t are mostly independent from the trader action a_{t-1}.

Deep Feature Lens

To overcome the challenges mentioned above, we propose a novel method, called DF-Lens. First, we study the behavior features and their expectation on time, and take them as the temporal features. Then we leverage observed actions to calculate the spatial feature expectation. After that, we introduce the behavior sensitiveness on actions as a key component in Q-Learning. The Learning objective function is studied. Finally, the model prediction is presented.

Temporal Feature Expectation

Given the state s at any position in the trajectory $\boldsymbol{\zeta}$, we consider the expectation of features on future observations after $\tilde{s}$. Formally speaking, given a trajectory $\boldsymbol{\zeta} = \{(s_i, a_i)\}_{i=1}^T$ and state $\tilde{s} \in \{s_i\}_i$, the *temporal feature expectation* is defined by

$$\begin{aligned} \tau^E(\tilde{s}|\boldsymbol{\zeta}) &\triangleq \mathbb{E}[\phi(s_t)|s_0 = \tilde{s}] \\ &= \frac{1}{T} \sum_{t=0, s_0=\tilde{s}}^{T} Pr(s_t|s_{t-1}, ..., s_0)\phi(s_t; \theta_\phi) \end{aligned} \tag{1}$$

where $Pr(s_t|s_{t-1}, ..., s_0)$ is the transition probability of the state and ϕ is an activation function parameterized by θ_ϕ. In MDP, the next state only depends on the current state, i.e. $Pr(s_{i+1}|s_i) = Pr(s_{i+1}|s_i, s_{i-1}, ...)$. In observation of ζ, $Pr(s_t|s_{t-1}, ..., s_0) = 1$. Therefore we can ignore this term. For simplicity, we ignore the normalization term $1/T$ and just set the feature expectation $\tau^E(\tilde{s}|\boldsymbol{\zeta}) = \sum_{t=0, s_0=\tilde{s}}^{T} \phi(s_t|\boldsymbol{\zeta})$. Note that $\tau^E(\tilde{s})$ is an ideal value while in practice the future states after $\tilde{s}$ are uncertain. Therefore,

we multiply the exponential decay, parameterized by γ_τ on feature expectation

$$\tau^\gamma(\tilde{s}|\boldsymbol{\zeta}) \triangleq \sum_{t=0,s_0=\tilde{s}}^{T} \gamma_\tau^t \phi(s_t;\theta_\phi). \tag{2}$$

The feature expectation is a temporal mixture of state features, and its dimensionality is the same as that of $\phi(s)$. The selection of γ_τ is tightly related to the record length T. For longer records, the smaller γ_τ could take more forward observations into consideration. If $\gamma_\tau = 0$, only the current state is leveraged without the future states estimated. If $\gamma_\tau = 1$, no decay is involved, namely $\tau^\gamma = \tau^E$.

Example 1 *Consider the example shown in Fig.1 with 4 states with OHLC features. We set $\gamma_\tau = 0.5$ and $\phi(x) = x$. So for the state s_0, the expectation of the first dimension of the features, can be calculated as $-0.2+0.5\times0.3+(0.5)^2\times(-0.3)+(0.5)^3\times(-0.1) = -0.1375 > -0.2$. It indicates that this feature is growing over time.*

Spatial Feature Expectation

We have just discussed feature forward temporal expectation, without considering transaction actions. Here we will discuss the influence of an arbitrary action a. Given a trajectory $\boldsymbol{\zeta}$, the states and actions are already fixed. We refer to these actions as the optimal action or observed action $\tilde{a}$ on observed state $\tilde{s}$. When applying the observed action $\tilde{a}$ for the state $\tilde{s}$, we can simply use feature expectation $\tilde{s}$. When applying any other action a, we can give it a discount γ_μ as a penalty since different actions may lead to unknown states. We refer to this as *spatial feature expectation*:

$$\mu(\tilde{s}, a|\tilde{a}, \boldsymbol{\zeta}) \triangleq \gamma_\mu^{(\mathbf{1}(a\neq\tilde{a}))}\tau^\gamma(\tilde{s}) \tag{3}$$

where $\mathbf{1}(q)$ is the indicator function returning 1 when the input condition q is true and 0 otherwise. Spatial feature expectation μ is a function on both state and action by scaling the expectation of un-observed actions ($a \neq \tilde{a}$) with parameter γ_μ. Given the observed action $\tilde{a}$, $\mu(\tilde{s}, a|\tilde{a}, \boldsymbol{\zeta})$ denotes feature expectation with the action a considered.

Example 2 *In Fig.1, we can get $\mu(s_0, a = 0) = 0.5 * \tau(s_0) = (-0.0687, 0.487, -0.1312, 0.3062)^T$ and $\mu(\tau(s_0), a = 1) = \tau(s_0)$.*

Sensitiveness Q-Learning

Senior traders have the ability to sense the rhythm and pace of the market, and can capture the nuances of the market (Intuition 2013). The ability shows the sensitiveness on market, which helps the trader decide which features mentioned in Sec. 3.2 to pay attention to, so as to evaluate the market state. The trader has a weight or preference for each feature. Intuitively, the weights show how features contribute to his/her personal evaluation, either positively or negatively. Moreover, if one feature's weight is higher, its (temporal/spatial) expectation will be more important for the evaluation. For example, a trader that only cares about the closing price, thinks that the price will rise on the next day if the stock rose today. Then, for this trader, the weight of the closing price is positive, while the others are 0. We refer to these weights as *sensitiveness* $\mathcal{I}$. Formally speaking, a vector $\mathcal{I}_g \in \mathbb{R}^F$ with the same dimensionality as the features ϕ, denotes the sensitiveness of the trader $g \in \mathcal{G}$ where $\mathcal{G}$ is the set of all traders. In practice, the sensitiveness varies from trader to trader, which is associated with the trader's individual trading style and policy. For example, a trader who prefers to buy after days of rising prices may have a positive high weight on the closing price. However, a trader who would like to buy after a stock has fallen for days may have a negative weight on closing price, since the more the stock falls, the more positive his assessment of the stock.

In order to estimate the market state based on the sensitiveness, we introduce the value functions: *sensitiveness Q function*, $Q_{\mathcal{I}_g}(\tilde{s}, a|\tilde{a}, \boldsymbol{\zeta})$, computing the evaluation result of a trader g taking the action a on the market state $\tilde{s}$ when an observed action $\tilde{a}$ and records $\boldsymbol{\zeta}$ are given. It is formally described as (4).

$$\begin{aligned} Q_{\mathcal{I}_g}(\tilde{s}, a|\tilde{a}, \boldsymbol{\zeta}) &\triangleq \mathcal{I}_g^T \mu(\tilde{s}, a|\tilde{a}, \boldsymbol{\zeta}) \\ &= \mathcal{I}_g^T \gamma_\mu^{(\mathbf{1}(a\neq\tilde{a}))}\tau^\gamma(\tilde{s}) \\ &= \mathcal{I}_g^T \gamma_\mu^{(\mathbf{1}(a\neq\tilde{a}))} \sum_{t=0,s_0=\tilde{s}}^{n} \gamma_\tau^t \phi(s_t;\theta_\phi) \\ &= \sum_{t=0,s_0=\tilde{s}}^{T} \gamma_\tau^t (\gamma_\mu^{(\mathbf{1}(a\neq\tilde{a}))}\mathcal{I}_g^T)\phi(s_t;\theta_\phi) \end{aligned} \tag{4}$$

where the last form is the widely used Q function in Inverse Reinforcement Learning, by taking $\gamma_\mu^{(\mathbf{1}(a\neq\tilde{a}))}\mathcal{I}_g^T$ as the reward function parameter. The Q function employs the spatial and temporal feature expectation, which can only be accessed in the model training procedure. In predicting procedure, only the current state s can be used for evaluation. Therefore we introduce a degraded Q function using only the current state, called *degraded Q function* $Q^*_{\mathcal{I}_g}$, as shown in (5). This function is the degraded form of 4 by setting temporal feature expectation as $\tau^\gamma(\tilde{s}) = \phi(\tilde{s})$. Intuitively speaking, Q^* is the approximation of Q with just current state observed. It maps the state into a scalar, denoting the evaluation of the current state from the perspective of trader g.

$$Q^*_{\mathcal{I}_g}(s) \triangleq \mathcal{I}_g^T \phi(s) \tag{5}$$

Degraded Q function is in a similar form with the reward function in IRL. The former denotes the subjective evaluation of an agent, e.g. a trader, g on the current state, while the latter is objective depending on the environment or simulation system. Even in IRL, where reward function is estimated by trajectory, the reward is still shared and fixed, (but hidden) for all agents. For our degraded Q function, it is personalized and tightly reflects the individual preference, i.e. sensitiveness $\mathcal{I}_g$, on the state.

Learning Objective Function

To learn the sensitiveness $\mathcal{I}_g$ from the records $\boldsymbol{\zeta}$ for each g in $\mathcal{G}$, we introduce an optimization problem to minimize the loss between predicted action and observed the action in training set. For the predicted action, we leverage the greedy policy in Q learning: $\arg\max_{a\in\mathcal{A}} Q_{\mathcal{I}_g}(\tilde{s}, a|\tilde{a}, \boldsymbol{\zeta})$. However, this policy causes over-fitting, especially for small amount of records. Therefore, we add a margin in the objective function parameterized by ρ. This promoted greedy policy is formalized by (6),

$$\arg\max_{a\in\mathcal{A}} Q_{\mathcal{I}_g}(\tilde{s}, a|\tilde{a}, \boldsymbol{\zeta}) + \rho\mathbf{1}(a \neq \tilde{a}). \tag{6}$$

Algorithm 1 Learning Sensitiveness by Gradient Descent

Input: Trajectory dataset $\boldsymbol{\zeta}$ of all traders $\mathcal{G}$, MDP\R\$\theta$ = $(\mathcal{S}, \mathcal{A}, \gamma)$ and learning rate α.
Output: Sensitiveness $\boldsymbol{\mathcal{I}}$.

for g in $\mathcal{G}$ **do**
  Initialize $\mathcal{I}_g$ randomly
  Derive $\boldsymbol{\zeta}_g$ from $\boldsymbol{\zeta}$
  while not converge **do**
    Initialize $\nabla \mathcal{I}_g \leftarrow 0$
    for s_i, a_i in $\boldsymbol{\zeta}_g$ **do**
      $\hat{a}_i \leftarrow \arg\max_{a \in \mathcal{A}} Q_{\mathcal{I}_g}(s_i, a|a_i) + \rho \mathbf{1}(a \neq a_i)$
      $\nabla \mathcal{I}_g \leftarrow \nabla \mathcal{I}_g + \mu(s_i, \hat{a}_i) - \mu(s_i, a_i)$
    end for
    $\mathcal{I}_g \leftarrow \mathcal{I}_g + \alpha * \nabla \mathcal{I}_g / n$
  end while
end for
return $\boldsymbol{\mathcal{I}}$

Given the demonstrated trajectories from trading records, we need to train the model and learn the sensitiveness. Since the greedy policy is associated with the Q function, a good evaluation of the Q function can significantly improve performance. As shown in (7), given the observed $(\tilde{s}, \tilde{a})$, the difference between trained evaluation on greedy policy action and observed action, is required to be small enough, called $\varepsilon - optimal$.

$$\max_{a \in \mathbb{A}} (Q_{\mathcal{I}}(\tilde{s}, a; \tilde{a}) + \rho \mathbf{1}(a \neq \tilde{a})) - Q_{\mathcal{I}}(\tilde{s}, \tilde{a}) \leq \varepsilon \tag{7}$$

For all pairs $\boldsymbol{\zeta} = \{(s_i, a_i)\}_{i=1}^{T}$, the trading sensitiveness is initialized randomly and tuned by the following optimization problem

$$\begin{aligned} &\min_{\mathcal{I}, \varepsilon} \frac{1}{2} \|\mathcal{I}\|^2 + \lambda \mathbb{E}[\boldsymbol{\varepsilon}] \\ &s.t. \forall i, Q_{\mathcal{I}}(s_i, a_i) + \varepsilon_i \geq \max_{a \in \mathbb{A}} Q_{\mathcal{I}}(s_i, a; a_i) + \rho \mathbf{1}(a \neq a_i). \end{aligned} \tag{8}$$

By reducing the constrains, we minimize the following objective function directly

$$\begin{aligned} J(\mathcal{I}) = &\frac{1}{N_t} \sum_{i=1 \ldots N_t} \mathbb{E}[\max_{a \in \mathbb{A}} Q_{\mathcal{I}}(s_i, a; a_i) + \rho \mathbf{1}(a \neq a_i) \\ &- Q_{\mathcal{I}}(s_i, a_i)] + \frac{\lambda}{2} \|\mathcal{I}\|^2 \end{aligned} \tag{9}$$

where N_t is the capacity of training set, λ is the Lagrangian multiplier.

The key idea here is employing gradient descent on sensitiveness in two phases. In the beginning, the sensitiveness is initialized randomly as $\mathcal{I}^0$, whose superscript denotes the iteration indicator in gradient descent. Then the gradient is updated at each iteration step t as follows.

- For each pair (s_i, a_i) in all trajectories, estimate the action $\hat{a_i}$ by solving (6):
- Given $\hat{a_i}$, calculate the gradient $\nabla \mathcal{I}^t$ by the difference of the spatial feature expectation between estimated action and observed action.

$$\nabla \mathcal{I}^t = \mu(s_i, \hat{a}) - \mu(s_i, a_i) \tag{10}$$

The pseudo-code of this algorithm is illustrated in Alg. 1. This algorithm takes transaction records from all traders $\mathcal{G}$, and learns their sensitiveness $\mathcal{I}$. The learning rate α determines the moving distance of sensitiveness with respect to the gradient. Like most learning algorithms, the self-adaptive gradient methods can be used to promote efficiency, e.g. AdaDelta (Zeiler 2012).

Table 1: Accuracy (average $\pm$ standard deviation) of approaches on four subsets: high winning-rate, low winning-rate, high frequency and low frequency group. The highest accuracy is marked by bold style for each subset.

Group	High-W	Low-W	High-F	Low-F	ALL
Win_Rate	68.26	52.25	60.40	57.75	59.40
Op_Rate	20.11	18.53	23.85	11.67	19.24
SCIRL	43.37±8.96	46.39±6.45	45.06±6.47	45.02±9.63	45.05±7.82
PolicyNet	70.25±6.92	69.06±7.89	67.85±6.35	72.44±8.32	69.59±7.50
SVM	71.49±6.88	70.61±7.36	69.62±6.03	73.27±8.23	71.71±7.17
RF	73.67±6.59	72.90±7.76	71.53±6.18	76.06±8.01	73.24±7.27
GDBT	72.84±6.30	72.30±7.05	70.84±5.96	75.32±6.98	72.54±6.73
LSTM	70.91±8.17	69.88±8.42	68.41±6.99	73.51±9.31	70.34±8.33
DNN	68.97±7.29	67.92±7.81	66.73±6.34	71.12±8.64	68.39±7.60
DF-Lens	**75.30**±6.60	**74.79**±7.76	**73.18**±5.99	**78.03**±8.13	**75.02**±7.27

Example 3 *In Fig. 1, the learned trading sensitiveness is* $\mathcal{I} = (-1.11, 3.27, -1.82, 1.37)^T$*, in which the second value is the highest. It suggests that the second feature, namely the highest price, is the most important factor for this trader.*

Model Prediction

Recall that our goal is to build an agent that trades like human traders. From the perspective of reinforcement learning, we should train the policy $\pi = Pr(\cdot|s)$. The policy is hard to learn because it is deeply hidden behind the observation s and a. Here, we join the degraded reward $Q^*_{\mathcal{I}}(s)$ with state s, to build the policy $Pr(a|[s, Q^*_{\mathcal{I}}(s)])$ via any off-the-shelf classification approaches, e.g. decision tree. The solution is highly scalable and easy to deploy on any existing price prediction system e.g. NN-based predictor (White 1988), or policy learning framework (Silver et al. 2016).

Experiments

Our data is collected from an open stock competition, called Candlestick-Trainer [1], held by Zhejiang Hexin Inc., whose stock transaction software [2] is one of the most widely used in China. The dataset contains 2,269,480 trading transactions collected from 1165 traders. The first experiment result illustrates the performance of our prediction. In the second experiment, we analyze the sensitiveness when using unsupervised and statistical approaches.

Comparsion with Baselines

Experiment Setup. For the first experiment, we want to study the effectiveness of DF-Lens in transaction behavior prediction. To observe the promotion of our approach on various traders, we select four typical groups of traders. The traders in the first group have a high winning rate ($> 60\%$), called High-W, and those from the second group have a low winning rate ($\leq 60\%$), called Low-W. They represent the senior and junior traders in the market, respectively. The third and fourth group have different operation rates (split by 12%), called High-F and Low-F, resp. Traders from the

[1] http://igame.10jqka.com.cn/kline/index
[2] http://www.10jqka.com.cn/

Table 2: Accuracy comparison on Different Approaches

Group	High-W			Low-W			High-F			Low-F			ALL
ID	T001	T002	AVG	T003	T004	AVG	T005	T006	AVG	T007	T008	AVG	
#Records	2,975	238		4,213	654		1,329	201		1,236	544		
Win_Rate	97.88	64.49	68.26	49.36	43.61	52.25	54.42	61.88	60.4	45.92	56.48	57.75	59.40
Op_Rate	12.6	17.11	20.11	12.36	12.49	18.53	43.92	22.5	23.85	7.93	11.67	11.67	19.24
PolicyNet	81.11	63.89	70.25±6.92	72.39	65.01	69.06±7.89	89.97	60.33	67.85±6.35	77.40	67.07	72.44±8.32	69.59±7.50
DF-Lens(PolicyNet)	**90.49**	**65.19**	**72.09**±6.88	**81.95**	**70.64**	**71.23**±8.06	**90.6**	**63.17**	**69.69**±6.04	**78.87**	**73.33**	**74.77**±8.68	**71.61**±7.57
SVM	83.49	65.69	71.49±6.88	74.82	66.40	70.61±7.36	86.23	64.15	69.62±6.03	80.45	67.17	73.27±8.23	71.71±7.17
DF-Lens(SVM)	**86.41**	**66.57**	**72.27**±6.78	**79.31**	**66.75**	**71.34**±7.37	**88.51**	**64.81**	**70.29**±6.04	**80.63**	**70.28**	**74.16**±8.06	**71.76**±7.13
RF	84.14	66.44	73.67±6.59	76.38	73.76	72.90±7.76	92.97	67.38	71.53±6.18	81.96	69.98	76.06±8.01	73.24±7.27
DF-Lens(RF)	**91.81**	**70.97**	**75.30**±6.60	**83.65**	**74.91**	**74.79**±7.76	**93.12**	**69.67**	**73.18**±5.99	**82.16**	**77.56**	**78.03**±8.13	**75.02**±7.27
GDBT	**84.72**	65.32	72.84±6.30	76.59	71.76	72.30±7.05	92.2	64.70	70.84±5.96	80.21	67.89	75.32±6.98	72.54±6.73
DF-Lens(GBDT)	84.55	**69.72**	**73.86**±6.18	**81.79**	**71.83**	**73.42**±7.07	**92.29**	**66.01**	**71.76**±5.74	**84.57**	**76.34**	**76.66**±7.02	**73.62**±6.69
LSTM	89.05	62.45	70.91±8.17	**84.58**	67.04	69.88±8.42	92.06	54.21	68.41±6.99	81.51	63.62	73.51±9.31	70.34±8.33
DF-Lens(LSTM)	**89.86**	**62.50**	**71.06**±8.70	84.19	**68.14**	**70.37**±8.47	**92.35**	**57.98**	**68.68**±7.43	**83.03**	**66.34**	**73.95**±9.31	**70.68**±8.58
DNN	81.7	61.71	68.97±7.29	71.42	65.08	67.92±7.81	89.01	59.29	66.73±6.34	76.64	62.91	71.12±8.64	68.39±7.60
DF-Lens(DNN)	**89.13**	**63.15**	**71.06**±7.08	**79.35**	**66.26**	**70.24**±7.83	**89.34**	**62.68**	**68.66**±5.84	**76.98**	**70.71**	**73.80**±8.77	**70.61**±7.52

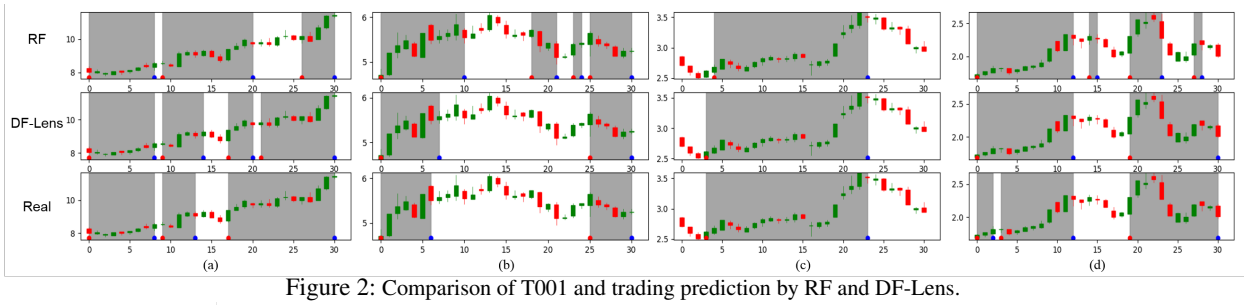

Figure 2: Comparison of T001 and trading prediction by RF and DF-Lens.

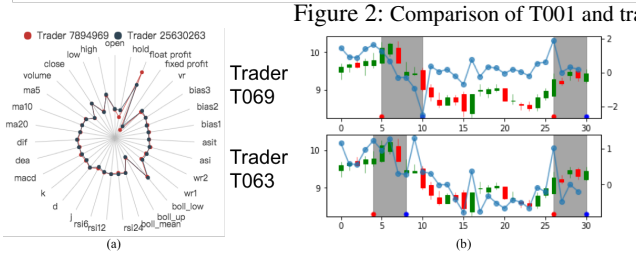

Figure 3: Comparison on traders from the same cluster

High-F are called high-frequency traders, who pursue short-term profit.

For each user, we compare the prediction accuracy on three kinds of baseline methods:

- off-the-shelf policy classifiers, including Supported Vector Machine (SVM) (Hearst et al. 1998), Random Forest (RF) (Liaw, Wiener, and others 2002), Gradient Boosting Decision Tree (GBDT) (Friedman 2002);
- deep learning approaches, including Long Short Term Memory(LSTM) (Hochreiter and Schmidhuber 1997), Deep Neural Network(DNN) (Hinton et al. 2012);
- inverse reinforcement learning approaches, SCIRL (Klein et al. 2012), policy net (PolicyNet) (Silver et al. 2016).

In this experiment, SVM, RF, GBDT are implemented by scikit-learn.LSTM and DNN are implemented by keras.The number of trees of RF is set as 500, the SVM uses an RBF kernel, and the DNN has six hidden layers. For SCIRL, we randomly select 10,000 records as training data for the LSPI (least squares policy iteration) to obtain the policy. DF-Lens is implemented by Tensorflow 1.2 by Python 2.7.12.. All programs are executed on the same computer with 32G RAM, 2.8GHz CPU, and NVidia GeForce TitanX GPU. The training set and test set are randomly divided in the ratio of $7:3$ and we ensure that all approaches are compared on the same test set. For each experiment, we repeat 5 times to get the average accuracy and standard deviation.

For DF-Lens, in our experiment, we adopt a linear score function as the activation function ϕ, the decay $\gamma_\mu = 0.9$, $\gamma_\tau = 0.9$, $\lambda = 1$ and $\rho = 1$. The default policy classifier in DF-Lens is Random Forest. Given the test collection of N_s state-action pairs, each state s_i with expert action a_i and predicted action $\hat{a}_i$, the performance is measured by accuracy formalized as follows:

$$Accuracy = \frac{\sum_{i=1}^{N_s} \mathbf{1}(\hat{a}_i = a_i)}{N_s}.$$

Overall Comparison Result. At first, we report the behavior prediction accuracy of all methods on four groups and entire dataset in Tab. 1. For each group, we report the average winning rate and operation frequency.

From this result, we make two interesting observations. *First*, DF-Lens outperforms SCIRL, PolicyNet, SVM, RF, GDBT, LSTM and DNN on all 4 groups and entire dataset with accuracy gain on average by 7.9% and up to 33%. Concretely, using DF-Lens, the accuracy gain for High-W is 7.9%, for Low-W is 7.7%, for High-F is 7.5% and for Low-F is 8.5%. the accuracy of SCIRL is much lower than other approaches. For these traders, the accuracy is below 50%. It is because the randomness of the stock market increases the difficulty of LSPI in estimating the state transition probability. *Second*, DF-Lens, and GBDT have the smallest standard derivation compared with most baselines on most datasets. Concretely, for High-W, the standard derivation of GBDT is 6.30 and DF-Lens is 6.60. For High-F, GBDT is 5.96 and DF-Lens is 5.99. For the rest methods, LSTM has the largest

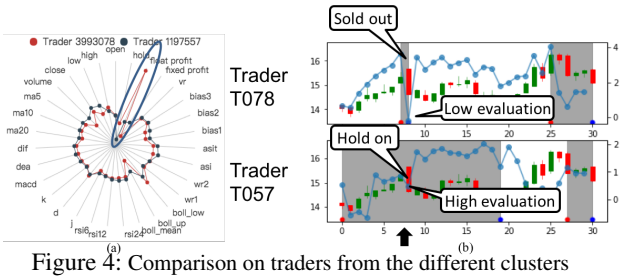

Figure 4: Comparison on traders from the different clusters

standard derivation. It may be caused by the under-fitting due to the small data amount for training.

Performance Comparison Result. DF-Lens is scalable to any off-the-shelf classifier. To further study the performance of DF-Lens, we conduct an experiment where we compare baselines with the specific DF-Lens version with the corresponding baseline approached involved. For example, we compare the accuracy of SVM and DF-Lens(SVM) which uses SVM as the policy classifier. To show the result, for each group, we analyze the entire set of traders and two individual traders who are the most representative ones. For example, for the first group, one trader with more records and another with fewer records are selected. Tab. 2 shows the characteristics of the traders including the number of records, win_rate and op_rate, and reports the accuracy of comparison results. We make two interesting observations. For PolicyNet, SVM, RF, GBDT, LSTM, and DNN, the DF-Lens versions generally have a higher accuracy both on entire sets of each group and individual cases. For each group, the average accuracy of the entire set obtained by the model with sensitiveness is higher for all the methods. Our approach performs better in the high winning rate group and low operating frequency group. For individual cases, out of 48 cases (6 classifiers $\times$ 8 users), sensitiveness improves or has the same accuracy in 46 cases. For example, PolicyNet is 81.11% and DF-Lens(PolicyNet) is 90.49% for trader T001. Random forest achieved the best performance across all users. Because RF is an ensemble learning approach, it can cover more complex market conditions with its inner weak classifiers.

We also compare the real actions of trader T001 and the predicted actions by RF, DF-Lens(RF) on the same market condition. In Fig. 2, (a) (b) (c) and (d) are four different market conditions from our dataset. The background figures are candle-stick charts. The red and blue dots at the x-axis represent the predicted buying and selling actions, respectively. For better illustration, we use the gray color to represent the period when the trader holds this stock. We can see that the action trajectory of DF-Lens(RF) is much more similar than the real one. In (a), DF-Lens(RF) only misses a buying and selling point around the 20-th day, while RF performs quite different from the real trader. This is similar in (b) (c) and (d), where DF-Lens(RF) takes almost the same actions as the real trader.

Clustering Experiment

In order to study the relation between sensitiveness and traders, we learn sensitiveness of all traders in our dataset and cluster the sensitiveness by employing k-Means with cluster number =10.

For better understanding, we randomly select two traders (T069 and No. T063) from the same cluster and compare their actions, as illustrated in Fig. 3. The trading records of these two traders on the same stock on the same trading day, namely same market data sequence, are shown as a candlestick chart in Fig. 3 (b). The buying and selling actions are marked at the x-axis with red and blue dots, respectively. It can be seen that the two traders take almost the same actions. Their sensitivenesses, shown in (a), are quite similar, showing the two traders with similar sensitiveness take similar actions. Also, we calculate the values of their personalized reward functions at each state (the blue lines). It can be seen that they have almost the same reward values for each state.

Then, we consider two traders (T078 and T057) from two different clusters. As shown in Fig. 4, these two traders' sensitiveness are different, especially at the "float-profit" and "BOLL_up" features. Similarly, their trading records on the same market sequence data are shown in Fig. 4 (b). We can see that the buying and selling actions are quite different.

Concretely, in Fig. 4 (a), Trader T078 has a higher value on "float-profit" than the trader T057. It means the former trader is more sensitive to float-profit which is the real-time profit when holding a stock. This difference will result in a significant variance when the two traders face the same market state. For example, see the 8-th trading day in Fig. 4 (b), marked by the black arrow. Before that day, these two traders both held this stock, and faced the same state on market, but their evaluation, degraded Q function, have different values. For trader T078 who is more sensitive when holding a stock, performs a conservative evaluation which is lower than that of trader T057. And their behaviors on the 8-th day, on which one sold out stock and another kept holding, verify our explanation. Through this case, we show that our behavior prediction model is interpretable, which can help to explain the behavior from their evaluation on the state.

Conclusion

In this paper, we propose a novel approach called DF-Lens for transaction behavior prediction, which extracts one's sensitiveness reflecting his/her sensitivity to state features in the financial markets. DF-Lens involves the MDP framework from IRL by regarding the human trader as an agent and fits the transaction records to estimate the personalized sensitiveness. The sensitiveness, introduced for the first time in this work, reflects one's preference on state features, and makes contribution to evaluate the entire market trend in near future. To achieve this, we leverage the temporal and spatial feature expectation in DF-Lens to deliver one's observed actions with arbitrary actions. With sensitiveness learned from trading records, we discuss the way to join it with off-the-shelf policy classifiers in DF-Lens to build a complete behavior predictor. The experiment on a real-world trading record dataset, demonstrates the effectiveness of DF-Lens for solving the transaction behavior prediction. The results of unsupervised learning show that one's sensitiveness can not only influence the prediction but also be interpretable. The future will concentrate on learning the changes of traders' sensitiveness over time.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First ACM ICML International Conference on Machine Learning*, 1–8. ACM.

Abbeel, P.; Coates, A.; and Ng, A. Y. 2010. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research* 29(13):1608–1639.

Agrawal, P.; Nair, A. V.; Abbeel, P.; Malik, J.; and Levine, S. 2016. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, 5074–5082.

Chernova, S., and Veloso, M. 2007. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 1–8. ACM.

Friedman, J. H. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38(4):367–378.

Hearst, M. A.; Dumais, S. T.; Osuna, E.; Platt, J.; and Scholkopf, B. 1998. Support vector machines. *IEEE Intelligent Systems and Their Applications* 13(4):18–28.

Hinton, G.; Deng, L.; Yu, D.; Dahl, G. E.; Mohamed, A.-r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T. N.; et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29(6):82–97.

Ho, J., and Ermon, S. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 4565–4573.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Hussein, A.; Gaber, M. M.; Elyan, E.; and Jayne, C. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)* 50(2):21.

Intuition, T. 2013. Trader intuition: How to know when you've got it and when to trust it. *URL: http://eminimind.com/trader-intuition-trust/*.

Klein, E.; Geist, M.; Piot, B.; and Pietquin, O. 2012. Inverse reinforcement learning through structured classification. In *Advances in Neural Information Processing Systems*, 1007–1015.

Le, D. 2009. Automated trading system. *University of Nottingham*.

Lee, S. J., and Popović, Z. 2010. Learning behavior styles with inverse reinforcement learning. In *ACM Transactions on Graphics (TOG)*, volume 29, 122. ACM.

Levine, S., and Abbeel, P. 2014. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, 1071–1079.

Liaw, A.; Wiener, M.; et al. 2002. Classification and regression by randomforest. *R news* 2(3):18–22.

Markov, S., and Tamayo, A. 2006. Predictability in financial analyst forecast errors: Learning or irrationality? *Journal of Accounting Research* 44(4):725–761.

Ratliff, N.; Ziebart, B.; Peterson, K.; Bagnell, J. A.; Hebert, M.; Dey, A. K.; and Srinivasa, S. 2009. Inverse optimal heuristic control for imitation learning. In *Artificial Intelligence and Statistics*, 424–431.

Ross, S., and Bagnell, D. 2010. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 661–668.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484–489.

Vogt, D.; Amor, H. B.; Berger, E.; and Jung, B. 2014. Learning two-person interaction models for responsive synthetic humanoids. *Journal of Virtual Reality and Broadcasting* 11(1).

White, H. 1988. Economic prediction using neural networks: The case of ibm daily stock returns. *Department of Economics, University of California*.

Yang, S.; Paddrik, M.; Hayes, R.; Todd, A.; Kirilenko, A.; Beling, P.; and Scherer, W. 2012. Behavior based learning in identifying high frequency trading strategies. In *Proceedings of 2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics*, 1–8. IEEE.

Yang, S. Y.; Qiao, Q.; Beling, P. A.; and Scherer, W. T. 2014. Algorithmic trading behavior identification using reward learning method. In *IJCNN*, 3807–3414. IEEE.

Yang, S. Y.; Qiao, Q.; Beling, P. A.; Scherer, W. T.; and Kirilenko, A. A. 2015. Gaussian process-based algorithmic trading strategy identification. *Quantitative Finance* 15(10):1683–1703.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial intelligence (AAAI)*, volume 8, 1433–1438. Chicago, IL, USA.

Ziebart, B.; Dey, A.; and Bagnell, J. A. 2012. Probabilistic pointing target prediction via inverse optimal control. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, 1–10. ACM.

Appendix: Candlestick-Trainer

Our data is collected from an open stock competition, called Candlestick-Trainer, held by Zhejiang Hexin Inc., whose stock transaction software is one of the most widely used in China.

Candlestick-Trainer

Candlestick-Trainer is an open competition. The minimum unit of competition is a **game**. A **participant** (i.e. trader) is ranked according to the average **yield** (rate of return) and **winning rate** across all games played. At beginning of a game, a sequence of market data from an anonymous stock, containing 90 days of trading data, is selected randomly by the system. The market data is real but the stock name and date information are invisible to the participant. Trading data for the preceding 60 days is illustrated in a Candlestick Chart. As shown in Fig. 5, a candlestick denotes the highest, lowest, opening and closing price daily. In China, the red candlestick means rising while green means falling. Besides price, the volume and some technical indexes, e.g. MACD, will also be illustrated. By observing these charts, the participant plays this investing game by **turns**. In each turn, the latest data is illustrated by the charts and the participant can take an **action**: **buying**, **waiting**, **selling** or **holding**. These actions can be divided into two groups: ⟨buying, waiting⟩ and ⟨holding, selling⟩. This division is based on whether the participant takes a **position**, namely has bought this stock. If no position is taken, the participant can choose to buy or just wait. If the participant chooses to buy this stock, it deals at the closing price of that day. Each purchase transaction is fixed for each deal in the amount of 1 million RMB. In other words, the system ignores the participant's balance and provides 1 million RMB for the participant for each transaction. If choosing to wait, the participant skips this turn. If having a position, the participant can choose to sell or hold. The former action sells all of that stock and the latter one skips this round. After an action is taken, the next trading day's data is added to the chart and next turn begins. The game ends after 30 turns (for 30 days). This game process simulates the real stock market. Note that in China, there are two special rules about the stock market:

- You cannot sell a stock the same day it is bought. This rule is called T+1, which means the selling transaction should wait at least one day.

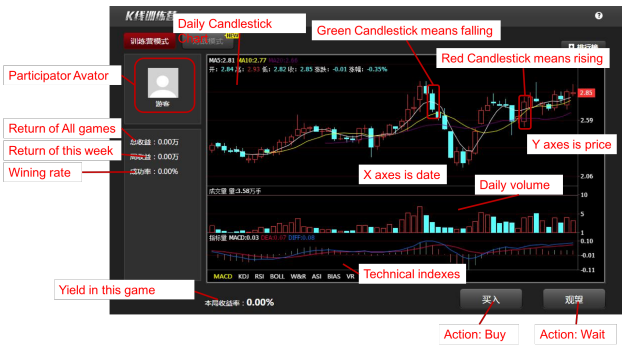

Figure 5: The screen snapshot of Candlestick-Trainer.

Table 3: Record Data Structure

Item	Explanation	Example
code	stock code	002689
stockName	stock name	Yuanda Zhineng
startDate	start date	2012/11/21
endDate	end date	2013/04/09
userId	participator id	2937193
itemData	daily market data	{63.84, 63.90, 63.84, 63.89, 23134;...}
actionData	action data	{1;...;0}

- You cannot sell a stock before you buy it. In most financial markets, there is a Futures market, where you can sell a product and buy it in the future. However, this is not allowed in China's stock market, nor in Candlestick-Trainer.

The entire game procedure can be summarized as follows:

Step 1 System provides 60 days of trading data.

Step 2 The newest day data is appended to the existing data. The participant chooses to buy or wait. If buy, spends 1 million RMB to buy this stock at the latest closing price and goes to step 3 else goes to step 2.

Step 3 The newest trading day's data is appended after the existing data in the chart. The participant chooses to sell or hold. If sell, sell all of the stock at the latest closing price and goes to step 2 else goes to step 3.

Step 4 Repeating above step 2 or step 3, 30 times. If stopping at step 3, the participant is forced to sell all stock.

For each game, the total **yield** is calculated. If the yield is positive, the system marks this game as **win**, else marks it as **lose**. The **winning rate** is calculated as the ratio of the number of games won with respect to all games played. For example, a participator played 3 games, got 2 wins and 1 lose. Then the winning rate is 66.66%. We collected the playing records from 1165 traders, which contained 2,269,380 games. The structure of the data and an example game record can be found in Tab. 3. Note that the item data is the sequence of the stock's price and dealing volume over 90 trading days. For example, the first day's data $63.84, 63.90, 63.84, 63.89, 23134$ means the opening, highest, lowest and close price and the dealing volume after the first day. The action data is also a sequence whose item denotes the action for that day. For this item, 1 denotes buying or holding and 0 denotes waiting or selling.

Technical Index as Feature

As discussed before, traders generally rely on historical prices and various technical indexes including MA, MACD, KDJ, RSI, BOLL, W&R, ASI, BIAS and VR, to make decisions. So, we select all 28 market technical indexes that appeared in the game, including OHLC (opening, highest, lowest, closing price), and volume as basic features to represent the status of a trading day. We add yield as an extra feature to denote the participant's status. Intuitively, the chosen action is not only affected by the market status but also by the status of the participant. For example, some traders tend to choose negative actions when their yield is too low. All technical indexes can be found in Tab. 4.

In the stock market, the price of different stocks varies a lot and the indexes can be quite different as well. To normalize these features, we turn OHLC data into a relative value compared to the closing price of the last day, namely

$$OHLC_t^* = \frac{OHLC_t - Close_{t-1}}{Close_{t-1}} \tag{11}$$

where the subscript t denotes the t-th day of data. For volumes and technical indexes, we use Z-score normalization to normalize them, as shown in Equation (12),

$$f_i^* = \frac{f_i - \mathbb{E}(f_i)}{Std(f_i)} \tag{12}$$

where f_i and f_i^* denotes the i-th feature before and after normalization, respectively. With all features normalized, we define the representation of **state** s, $s \in \mathbb{S}$. State set $\mathbb{S} \subseteq \mathbb{R}^F$ where F is the number of features, and in Candlestick-Trainer, $F = 29$.

Trading Behavior

In Candlestick-Trainer, a trader can only take the action "Buy(B)", "Wait(W)", "Sell(S)", "Hold(H)". The action set is thus $\{B, W, S, H\}$. For convenience, we can reduce it to two categories, **Rising action** and **Falling action**. Rising action includes "Buy" and "Hold", which means the trader thinks the stock price will rise. Falling action includes "Wait" and "Sell", which means the trader thinks it will fall. The action set can be defined as $\mathbb{A} = \{0, 1\}$, where "0" represents a rising action and "1" represents a falling action. For the data of a game, we define it as a **trajectory** $\boldsymbol{\zeta}$, which is a set of state-action pairs $(s_t, a_t)_{t\in\{1...T\}}$, $s_t \in \mathbb{S}$, $a_t \in \mathbb{A}$, where t denotes the t-th trading day, and T is 30 in Candlestick-Trainer.

Table 4: Technical indexes

Technical Indicator	Description	Formula
Moving Average (MA)	Moving average (MA) is the unweighted mean of the previous n data	$MA = \frac{\sum_{n-1}^{i=0} C_{t-i}}{n}$
MACD	MACD is supposed to reveal changes in the strength, direction, momentum, and duration of a trend in a stock's price	$DIF = EMA(C,12) - EMA(C,26)$ $DEA = EMA(DIF,9)$ $MACD = 2 * (DIF - DEA)$
Stochastic Oscillator (KDJ)	Stochastic oscillator is a momentum indicator that uses support and resistance levels. This method attempts to predict price turning points by comparing the closing price of a security to its price range.	$RSV = \frac{C_t - LLV(L,9))}{HHV(H,9) - LLV(L,9)} * 100$ $K_t = \frac{2}{3} * K_{t-1} + \frac{1}{3} * RSV$ $D_t = \frac{2}{3} * D_{t-1} + \frac{1}{3} * K_t$ $J_t = 3 * K_t + 2 * D_t$
RSI	The relative strength index (RSI) is intended to chart the current and historical strength or weakness of a stock or market based on the closing prices of a recent trading period.	$RSI = \frac{SMA(Max(C_t - C_{t-1}, 0), n, 1)}{SMA(ABS(C_t - C_{t-1}, 0), n, 1)} * 100$
BOLL	Bollinger Bands and the related indicators %b and bandwidth can be used to measure the "highness" or "lowness" of the price relative to previous trades	$MID = MA(C,n)$ $UPPER = MID + 2 * STD(C,n)$ $LOWER = MID - 2 * STD(C,n)$
W&R	Williams %R showing the current closing price in relation to the high and low of the past N days (for a given N)	$WR = \frac{HHV(H,n) - C - t}{HHV(H,n) - LLV(L,n)} * 100$
ASI	Accumulation swing index (ASI) is used to gauge a security's long-term trend by comparing bars which contain its opening, closing, high and low prices throughout a specific period of time	$AA = ABS(H_t - C_{t-1})$; $BB = ABS(L_t - C_{t-1})$ $CC = ABS(H_t - L_{t-1})$; $DD = ABS(C_t - O_{t-1})$ $R = \begin{cases} AA + BB/2 + DD/4 & if(AA > BB \ \& \ AA > CC) \\ BB + AA/2 + DD/4 & if(BB > CC \ \& \ BB > AA) \\ CC + DD/4 & otherwise \end{cases}$ $X = C_t - O_{t-1} + \frac{C_t - O_t}{2}$ $SI = 16 * \frac{X}{R} * Max(AA, BB)$ $ASI = SUM(SI, 26)$ $ASIT = MA(ASI, 10)$
BIAS	BIAS indicator is used to show the deviation of the stock price and its moving average over a certain period of time	$BIAS = \frac{C_t - MA(C,n)}{MA(C,n)} * 100$
VR	VR(Volatility Volume Ratio) can reflect the heat of the stock price from the perspective of the volume	$VR = \frac{SUM(if(C_t > C_{t-1}), VOL, n)}{SUM(if(C_t \leq C_{t-1}), VOL, n)} * 100$