Adaptive Quantitative Trading: An Imitative Deep Reinforcement Learning Approach

Yang Liu,[1] **Qi Liu,**[1]* **Hongke Zhao,**[2] **Zhen Pan,**[1] **Chuanren Liu**[3]
[1]Anhui Province Key Laboratory of Big Data Analysis and Application, School of Data Science & School of Computer Science and Technology, University of Science and Technology of China
[2]College of Management and Economics, Tianjin University
[3]Department of Business Analytics and Statistics, University of Tennessee
{liuyang0, pzhen}@mail.ustc.edu.cn, qiliuql@ustc.edu.cn, hongke@tju.edu.cn, cliu89@utk.edu

Abstract

In recent years, considerable efforts have been devoted to developing AI techniques for finance research and applications. For instance, AI techniques (e.g., machine learning) can help traders in quantitative trading (QT) by automating two tasks: market condition recognition and trading strategies execution. However, existing methods in QT face challenges such as representing noisy high-frequent financial data and finding the balance between exploration and exploitation of the trading agent with AI techniques. To address the challenges, we propose an adaptive trading model, namely iRDPG, to automatically develop QT strategies by an intelligent trading agent. Our model is enhanced by deep reinforcement learning (DRL) and imitation learning techniques. Specifically, considering the noisy financial data, we formulate the QT process as a Partially Observable Markov Decision Process (POMDP). Also, we introduce imitation learning to leverage classical trading strategies useful to balance between exploration and exploitation. For better simulation, we train our trading agent in the real financial market using minute-frequent data. Experimental results demonstrate that our model can extract robust market features and be adaptive in different markets.

Introduction

In financial security investment, quantitative trading (QT) is characterized by its high degree of automation and continuity. With the assistance of computer technology, quantitative traders aggregate information and place orders more and more efficiently. Instead of active judgments, quantitative traders can efficiently reduce irrational trading decisions by trading programs. To date, quantitative hedge funds have become the mainstream of security investment. To meet the great demand for financial innovation, leveraging machine learning in QT is becoming a central topic of Fintech.

In real financial market, unpredictable trading behaviors and economic events lead to the noisy and non-stationary financial data. Security prices reflect little part of market information, which means that we can never directly observe the actual market states. In the field of QT, technical analysis (Murphy 1999) is the most widely used method. Technical analysis aims at building technical indicators mainly using charts of *Opening-High-Low-Closing* prices (OHLC) and trading volumes. Dual Thrust strategy (Pruitt and Hill 2012) is a good example of technical analysis as shown in Figure 1. However, these predefined and handcraft technical indicators are frequently criticized for their poor generalization capacity (Deng et al. 2016). For instance, an identical technical strategy may perform differently in two similar financial markets. Thus, this brings a big challenge to represent robust features directly from financial data.

Figure 1: A diagram of Dual Thrust strategy comprised of OHLC price and volumes chart, where trading signals (down/up triangles) are formed once current price breaks through the BuyLine (green line) /SellLine (red line).

Machine learning approaches are helpful to improve the generalization ability. Previous studies concentrate on price trend prediction by approaches based on deep neural networks (Feng et al. 2019; Jin et al. 2019; Zhao et al. 2017a; Li et al. 2015; Zhao et al. 2017b). But price trend prediction is not the only thing in QT, we also need well-designed trading strategies. As an area of machine learning, reinforcement learning (RL) proposes a framework for sequential decision-making problems. The agent in RL learns a policy concerned with how to take actions in an environment to maximize cumulative reward (Sutton and Barto 2018). It seems suitable to build trading strategies by RL approaches. However, the

*Corresponding Author.

agent in RL faces the problem of balancing exploration (of uncharted territory) and exploitation (of current knowledge) (Kaelbling, Littman, and Moore 1996). In the real trading environment, constrained by the market friction factors (e.g., transaction fee, slippage, and market capacity), random exploration without goals may bring great losses. However, the agent can hardly learn an effective policy without adequate trials and errors, especially on trading tasks. Therefore, here comes another distinct challenge to find a balance between exploration and exploitation of the trading agent.

To address above challenges, we propose *imitative Recurrent Deterministic Policy Gradient* (iRDPG). To be specific, for the first challenge about representing the noisy high-frequent financial data, we model the whole QT process as a Partially Observable Markov Decision Process (POMDP) (Kaelbling, Littman, and Cassandra 1998). The POMDP aims at modeling the process where the underlying states cannot be observed. In this work, considering the inevitable noise in financial data, we deem that our trading agent cannot directly observe market states. Studies suggest that the POMDP can be solved by approaches with recurrent neural networks e.g., Recurrent Deterministic Policy Gradient (RDPG) (Heess et al. 2015) , an off-policy deep reinforcement learning (DRL) algorithm. For the second challenge concerned with balancing exploration and exploitation of the agent, we draw lessons from the technical analysis by imitation learning techniques (i.e., demonstration buffer and behavior cloning). Specifically, we set a demonstration buffer initialized by actions from Dual Thrust. With the help of it, we shorten the inefficient random exploration phases. To further keep the action continuity of the trading agent, we introduce the behavior cloning technique. By incorporating these imitation learning techniques to the POMDP framework, the trading agent can be enhanced by financial domain knowledge. Our proposed model, iRDPG, is tested on real financial data of futures. Compared with baseline trading strategies, iRDPG can learn profitable trading policies and has better generalization ability on different futures markets.

Related work

In general, the related work of our research could be classified into the following two categories.

Technical Analysis. Technical analysis (Murphy 1999) is the most widely used method of interest in QT. Technicians believe that part of market information is reflected in price and volume data (Malkiel and Fama 1970). They tend to build technical indicators to generate trading signals. Technical indicators are mathematical formulas for modeling some aspects of the security price trends. Two primary types of indicators are those based on moving averages and the oscillators. Those based on moving averages tend to identify price trends through data smooth, while strategies based on the oscillators (e.g., Dual Thrust) are employed for identifying momentum(Kim and Shin 2007). However, technical indicators cannot adapt to different market patterns.

Reinforcement Learning for Quantitative Trading. Recent years have witnessed the successful marriage of machine learning and security investment. Reinforcement learning (RL) is an area of machine learning and specializes in the sequential decision-making process. Although innovating QT with RL is still under-explored, many trails have been made. Neuneier (1996) made the first attempt to solve trading problems using Q-learning, a typical value-based RL algorithm. The value-based approaches learn the optimal policy through state-action value functions. However, the value-based approaches are not good at large scale problems. Sutton et al. (2000) found that policy-based RL enables a simpler problem representation than that in value-based algorithms. Moody and Saffell(1999) introduced a policy-based approach, namely recurrent reinforcement learning (RRL).

Nevertheless, classical reinforcement learning approaches have difficulties in the choice of market features. Deep learning approaches are well-suited to deal with large input states. The combination of RL and DL, called deep reinforcement learning (DRL) performs well in problems with high dimensional data. DRL has achieved great strides in complex tasks, such as video games (Mnih et al. 2015). Meanwhile, DRL also has the potential for QT. For instance, Jiang, Xu, and Liang (2017) uses the model-free Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015) to dynamically optimize cryptocurrency portfolios. The fuzzy learning and deep neural networks (DNNs) are extended to improve financial signal representation (Deng et al. 2016). However, these model-free RL algorithms are sampling inefficient for the large state space problem like QT. Yu et al. (2019) proposed a model-based RL framework for daily frequency portfolio trading. Instead of daily-frequency, minute-frequent data are commonly used in QT (Pruitt and Hill 2012). On minute timescale, human traders cannot synthesize information as fast as algorithms.

Different from previous work, in this paper, we introduce continuous-time policy-based RL enhanced by Recurrent Neural Networks (RNNs). Our adaptive DRL model is designed for representing minute-frequent financial data and adapting to different financial markets.

Problem Definition

In this section, we first clarify mathematical symbols then formally introduce the quantitative trading problem in detail.

Preliminaries

At time step t, we denote the OHLC price vector as $\mathbf{p}_t = \left[p_t^o, p_t^h, p_t^l, p_t^c\right]$. The price sequence of a financial security is written as $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_t, \cdots]$. We simplify the historical price sequence at a time period as $\mathbf{P}_{t-n:t}$, where n is a window length. We denote t technical indicator vector at time t as $\mathbf{q}_t = \bigcup_j q_t^j$,where q_t^j is related to historical price sequence with : $q_t^j = f\left(\mathbf{P}_{t-n:t}; \theta^j\right)$, and θ^j is the parameter of technical strategy j. We write the technical indicator sequence as $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_t, \cdots]$. Similarly, at time step t, we denote the account profit as r_t. And the account profit sequence is written as $\mathbf{R} = [r_1, \cdots, r_t, \cdots]$.

Partially Observable MDP

In this subsection, we introduce domain-specific characteristics in QT and further explain the reason why it is suitable to model the whole QT process as a partially observable Markov Decision Process (POMDP).

In the financial market, security prices are formed by orders from bulls (investors with optimistic market outlooks) and bears (investors with pessimistic market outlooks). At a high level, prices are influenced by macroeconomic and microeconomic activities. The unpredictable events and trading behaviors lead to the noisy financial market. Thus We cannot directly observe the actual market states. For instance, no one knows exactly whether a piece of good news leads the price up or whether orders can be executed at expected prices. The only data we can use are historical prices and volumes. In other words, the price and volume is a part of the underlying market state. The technical indicators in technical analysis can be treated as the observations of the prospective price trends. In general, QT is exactly a sequential decision-making problem about what and when to trade.

The POMDP is a realistic generalization of a Markov Decision Process (MDP) for model the QT problem. In general, an MDP is a 5-tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$. Specifically, $\mathcal{S}$ is a finite set of states. $\mathcal{A}$ is a finite set of action set. $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a state transition function, which consists of a set of conditional transition probabilities between states. $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the reward function, where $\mathcal{R}$ is a continuous set of possible rewards. R indicates the immediate reward from taking an action in a state. And $\gamma \in [0, 1)$ is the discount factor. For the deterministic policy, the goal of an agent is to learn a policy $\mu : \mathcal{S} \to \mathcal{A}$, which maximizes the expected discounted reward $J = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t\right]$. The action-value function $Q^{\mu} = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | \mu\right]$ is introduced to estimate the performance of policy μ. When it comes to the POMDP, $\mathcal{O}$ and Z are incorporated, where $\mathcal{O}$ is a set of observation and $Z : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$ is the observation transition function. At each time period, the agent takes an action $a_t \in \mathcal{A}$ in a particular environment state $s_t \in \mathcal{S}$, which leads to the transition to state s_{t+1} with probability $T(s_{t+1}|s_t, a_t)$. Meanwhile, the agent receives an observation $o_{t+1} \in \mathcal{O}$ on the state s_{t+1}, with probability $Z\left(o_{t+1}|s_{t+1}, a_t\right)$. In particular, it is helpful for the decision maker to take into account the observable history up to time t. In other word, a history of observations can be used as a pseudo-state (François-Lavet et al. 2018).

Observation. Considering the particularity of financial market, we divide the observation set $o_t \in \mathcal{O}$ into two parts: the account observation set $o_t^a \in \mathcal{O}^a$ and the market observation set $o_t^m \in \mathcal{O}^m$, where o_t^a denotes the cumulative account profit $\sum_{k=1}^{t} r_k \in \mathbf{R}$, and o_t^m is related with the price $\mathbf{p}_t \in \mathbf{P}$ and technical indicator $\mathbf{q}_t \in \mathbf{Q}$. For most of RL tasks, states are directly transformed by actions. Different from the general situations, trading actions from an individual investor have little impact on the entire market. In other words, trading actions are irrelevant to market observation transition function, which means $Z = Z\left(o_{t+1}^m | s_{t+1}\right)$. While the personal account observation set is totally dependent on actions. However, if we sum up these two parts, the transition function of the whole observation set $\mathcal{O}$ can fit the POMDP framework: $Z = Z\left(o_{t+1}|s_{t+1}, a_t\right)$. That is why we regard the entire observation set as two different parts.

The general price trends can be treated as a crucial part of the actual market state. Drawing on the experiences of technical analysis, we select the technical indicators (BuyLine, SellLine) from the Dual Thrust strategy as observations of price trends. As thus, th whole observation set can be denote by $\mathcal{O} = \{\mathbf{P}, \mathbf{Q}, \mathbf{R}\}$.

Action. To compare different trading strategies, we stipulate that the agent makes trades with the minimum security amount. The trading action here is defined as a continuous probability vector $\mathbf{a}_t = [\mathrm{P}_{\mathrm{long}}, \mathrm{P}_{\mathrm{short}}]$. The agent executes the action with the maximum probability. At time t, actions can be written as $a_t \in \{\text{long}^1, \text{short}^2\} = \{1, -1\}$. To a certain extent, this setting can ease the challenge of position[3] management. Also, the influence from market capacity[4] can be alleviated. Especially, considering the trading action continuity, we regard the actions from a certain policy just as trading signals, which means the actual executed actions depend on the positions. In practice, rules are described below:

- A new position will be taken (long or short the target security at a certain price) following the signal (except 'close' signal) if it is empty.
- The original position will not be changed until receiving a different signal. At the same time, the position will be closed (place a contrary direction order). Then, a new position will be taken according to the new signal.

Reward. To narrow the gap between the simulation and reality, we simulate trading with practical constraints. To be specific, we take into account the crucial parts of market friction factors, i.e., trading transaction fee δ and slippage ζ [5]. With these pratical market constraints, at time t, the account profit r_t is calculated as:

$$r_t = \left(p_t^c - p_{t-1}^c - 2\zeta\right) a_{t-1} - \delta \left| a_t - a_{t-1} \right| p_t^c. \quad (1)$$

However, previous work suggets that the account profit r_t maybe not an effective reward function for QT problems. (Moody and Saffell 1999). Using the reward function in RRL algorithm (Moody and Saffell 1999) for reference, we select the *differential Sharpe ratio* (D) as our reward function. Here, the *Sharpe ratio* (Sr) (Sharpe 1966) is an evaluation for risk-adjusted return. The Sharpe ratio (Sr) indicates the ratio of the excess return (cumulative return minus risk-free return) over one unit of total risk. Without the loss of mathematical generality, at time t, Sr_t is defined as:

$$Sr_t = \frac{\mathbb{E}\left[\mathbf{R}_{t-n:t}\right]}{\sigma\left[\mathbf{R}_{t-n:t}\right]}. \quad (2)$$

[1] Buy the security now for resale later.

[2] Sell the security now to buy later.

[3] The ratio of investment capital to total capital.

[4] Orders may not be traded at the expected price.

[5] A constant added/subtracted into the quote price for simulation where a long/short order is usually traded at a higher/lower price.

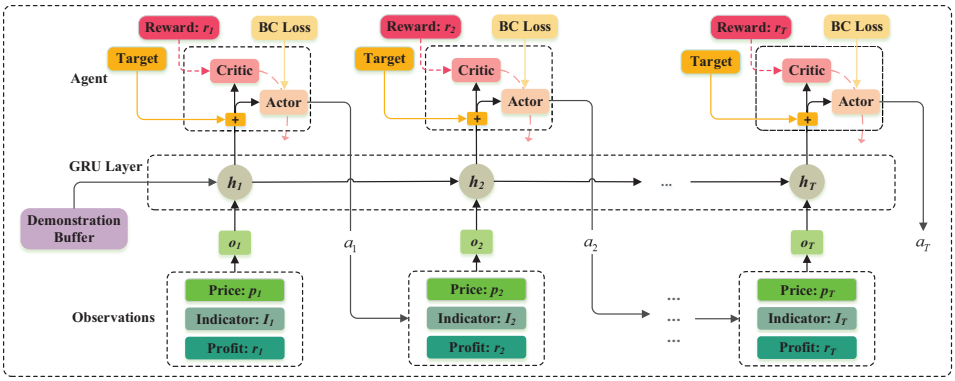


Figure 2: The overview of iRDPG model

After expanded to first order in the adaptation rate η, Sr_t can be denoted as:

$$Sr_t \approx Sr_{t-1} + \eta \frac{\mathrm{d}Sr_t}{\mathrm{d}\eta}|_{\eta=0} + O\left(\eta^2\right). \tag{3}$$

Since only the first order term in Equation (3) depends on the account profit r_t, at time t, we define the differential Sharpe ratio d_t as:

$$d_t := \frac{\mathrm{d}Sr_t}{\mathrm{d}\eta} = \frac{\beta_{t-1}\Delta\alpha_t - \frac{1}{2}\alpha_{t-1}\Delta\beta_t}{(\beta_{t-1} - \alpha_{t-1}^2)^{\frac{3}{2}}}, \tag{4}$$

where α_t and β_t are exponential moving estimates of the first and second moments of r_t. They can be written as:

$$\begin{aligned} \alpha_t &= \alpha_{t-1} + \eta\Delta\alpha_t = \alpha_{t-1} + \eta(r_t - \alpha_{t-1}), \\ \beta_t &= \beta_{t-1} + \eta\Delta\beta_t = \beta_{t-1} + \eta(r_t^2 - \beta_{t-1}), \end{aligned} \tag{5}$$

where we treat α_{t-1}, β_{t-1} as numerical constants. Actually, η in the update Equation (5) control the magnitude of the influence of account profit r_t on the Sharpe ratio Sr_t. Therefore, d_t actually represents the influence of r_t on Sr_t. To sum up, the differential Sharpe ratio has several attractive properties involving facilitating recursive updating, enabling efficient on-line optimization, weighting recent returns more and providing interpretability (Moody and Saffell 2001). Note that d_t indicates the value of reward function R at time t, rather than account profit r_t.

Imitative RDPG

In this section, we introduce our model, iRDPG, which is designed to solve the POMDP framework for the QT problem. We introduce Recurrent Deterministic Policy Gradient and imitation learning orderly. In additon, we present the overview of our iRDPG framework in Figure 2.

Recurrent Deterministic Policy Gradient

Deterministic Policy Gradient (DPG) (Silver et al. 2014) is a class of off-policy RL algorithm designed for continuous control. At a high level, high-frequent QT is actually of interest in continuous control. Note that QT cares much about trading continuity because shifting trading actions frequently is costly. Thus, QT problems can be better addressed by the algorithm of DPG class, such as Recurrent Deterministic Policy Gradient (RDPG) (Heess et al. 2015), a recurrent extension of DPG.

In our POMDP framework for QT, the agent receives an observation o_t from the market and personal account at each time period. The observation consists of the market price, technical indicators and the account profit. Though the underlying market state cannot be observed directly, our trading agent may benefit from considering the history $\mathbf{H}$. The observation-action history $\mathbf{H}$ can be described as $h_t = (o_1, a_1, \cdots, o_{t-1}, a_{t-1}, o_t)$. RDPG makes use of the recurrent neural networks (RNNs) to effectively synthesize historical information. Meanwhile, our agent in RDPG learns to preserve beneficial information for the trading decision process. Specifically, RDPG is an actor-critic approach, which bridges the gap between policy gradient methods and value approximation methods for RL. In an actor-critic framework, the agent learns an action-value function (critic) by minimizing the TD error like Equation (8). Simultaneously, the agent learns a deterministic policy μ (actor) by directly maximizing the estimated action-value function Q^{μ}. Requiring the access to the history $\mathbf{H}$, RDPG maintains an actor function $\mu(h)$ with parameters θ, a critic function $Q(h, a)$ with parameters ω as well as a replay buffer as a set of episodes $(o_1, a_1, r_1, \cdots, o_T, a_T, r_T)$.

In this work, we introduce the Gate Recurrent Unit (GRU) (Chung et al. 2015) to the QT domain. We treat the previous observation-action history h_{t-1} as the hidden state returned by the RNN at time step $t-1$. In that case, history h_t can be written as:

$$h_t = GRU\left(h_{t-1}, a_{t-1}, o_t\right). \tag{6}$$

At each training step, the training rollouts are collected with extra noise from a stochastic process $\mathcal{N}$. The agent action can be written as: $a_t = \mu^{\theta}(h_t) + \epsilon, \epsilon \sim \mathcal{N}$. After the agent performs a trading action (place orders), it will receive the reward d_t and next observation o_{t+1} returned from the market and personal account. Upon the maximum time length T is arrived, the whole episode $(o_1, a_1, d_1, \cdots, o_T, a_T, d_T)$ is stored to the prioritized replay buffer $\mathcal{D}$. Subsequently, a minibatch comprised of N complete episode is sampled from $\mathcal{D}$ for model updating.

For each episode i, RDPG minimizes the following loss L w.r.t ω to update the critic:

$$y_t^i = d_t^i + \gamma Q^{\omega'}(h_{t+1}^i, \mu^\theta(h_{t+1}^i)), \tag{7}$$

$$L = \mathbb{E}\left[(y_t^i - Q^\omega(h_t^i, a_t^i))^2\right]. \tag{8}$$

The actor is updated using the sampled policy gradient $\nabla_\theta J$ with backpropagation through time (BPTT):

$$\nabla_\theta J = \mathbb{E}\left[\nabla_a Q^\omega(h,a)|_{h=h_t^i, a=\mu^\theta(h_t^i)} \nabla_\theta \mu^\theta(h)|_{h=h_t^i}\right], \tag{9}$$

where θ and θ' as well as ω and ω' are two copy parameters of the value function Q and policy μ respectively. θ and ω are the parameters updated during training; θ' and ω' track them with some delay. In other words, θ' and ω' are the target values for updating.

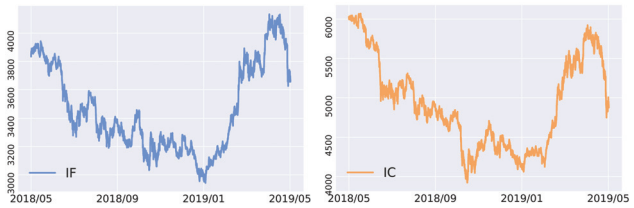
Imitative Learning

The dynamic financial market data leads to an exponentially growing value space for exploration. The agent with the model-free RL algorithm can hardly learn a profitable policy in QT. In addition, considering trading continuity and market friction factors, random exploration without goals may be inefficient. However, the model-free RDPG can be leveraged with training goals. As an off-policy algorithm, RDPG can suit the auxiliary data. In particular, we introduce Demonstration Buffer and Behavior Cloning to guide our RDPG agent, where these two modules respectively represent the passive and active imitation learning algorithms.

Demonstration Buffer. Initially, we set a prioritized replay buffer $\mathcal{D}$. And $\mathcal{D}$ is filled with demonstration episode $(o_1, a_1, r_1, \cdots, o_T, a_T, r_T)$ from the Dual Thrust strategy in advance. Drawing on the lessons from DQfD (Hester et al. 2018) and DDPGfD (Večerík et al. 2017), we pre-train the agent using demonstrations before the actual interaction. With the help of technical analysis pre-training, the agent can learn a fundamental trading strategy at the start. During the training process, each minibatch consisting of demonstration and agent episode is sampled by prioritized experience replay (PER) (Schaul et al. 2015). PER encourages to sample more valuable episodes more frequently. And the probability of the episode $P(i)$ is proportional to its priority, namely: $P(i) = \frac{p_t^\phi}{\sum_i p_i^\phi}$, where p_i is the priority of the episode i and ϕ is a constant. In practice, we modify the episode priority definition in (Večerík et al. 2017). In this work, p_i is defined as:

$$\mathbb{E}\left[|y_t^i - Q^\omega(h_t^i, a_t^i)| + \lambda_0 |\nabla_a Q^\omega(h_t^i, a_t^i)|\right] + \epsilon_D, \tag{10}$$

where the first term represents the loss L^i in Equation (8) of the episode i; the second term indicates the absolute value of actor gradient in Equation (9); ϵ_D is a positive constant for demonstration episode to increase the probability of getting sampled; and λ_0 weighs the contributions to the actor gradient. Considering the change in the sample distribution, updates to the network are weighted with importance sampling weights, $w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\psi$ where ψ is a constant. In this way, the prioritized demonstration buffer controls the ratio of data between the demonstration and agent episode. More importantly, it enables the efficient propagation of reward.



(a) IF (b) IC

Figure 3: Closing price sequences of IF and IC stock-index futures in our test set.

Behavior Cloning. To set a goal for each trading action, we introduce intra-day greedy actions as the expert actions $\bar{a}$. In hindsight, we can create a prophetic trading expert who always takes a long position at the lowest price and takes a short position at the highest price. For each training step, we use *Behavior cloning* technique (Ross and Bagnell 2010) to measure the gap between agent actions and the actions from the prophetic expert. In addition, we record behavior cloning losses (BC Loss) only when the critic $Q(h,a)$ indicates that the expert actions perform better than the actor actions:

$$L' = -\mathbb{E}\left[\left\|\mu^\theta(h_t^i) - \bar{a}_t^i\right\|^2 \mathbf{1}_{Q(h_t^i, \bar{a}_t^i) > Q(h_t^i, \mu^\theta(h_t^i))}\right]. \tag{11}$$

This modification is called *Q-Filter* in (Nair et al. 2018). In general, the behavior cloning loss L' is an auxiliary loss for updating. As thus, a modified policy gradient $\nabla_\theta \bar{J}$ is applied to the actor:

$$\nabla_\theta \bar{J} = \lambda_1 \nabla_\theta J + \lambda_2 \nabla_\theta L', \tag{12}$$

where $\nabla_\theta J$ is the policy gradient in Equation (9); $\lambda 1$ and $\lambda 2$ control the weights between the losses. With the help of expert actions, we set goals for each training step. The expert actions shorten the inefficient exploration phases.

Experiments

We back-test our model on the minute-frequent futures data with practical constraints. Specifically, we collect minute-frequent data of IF and IC financial futures. Both of them are representative stock-index futures in China. The IF data are based on the index calculated on account of the prices of the top 300 stocks from both Shanghai and Shenzhen exchange centers. The IC data are based on another similar index, which focuses on the stocks with mid and small capitalization. The minute-frequent closing prices series of IF and IC futures are shown in Figure 3.

Experimental Setup

In our experiment, we use minute-bar OHLC prices of futures. One minute bar reflects fluctuations within 1 minute. For RL, it is difficult to keep action continuity on such high-frequency data. But in the real financial market, minute-frequent data are quite common. We collect minute frequent

futures data from JoinQuant.com [6], a famous Chinese quantitative platform. The Data in the training set spans from Jan 1st, 2016 to May 8th, 2018 while the data in the testing set spans from May 9th, 2018 to May 8th, 2019. For better simulation, we take into accounts practical constraints, including the transaction fee $\delta = 2.3 \times 10^{-5}$ and the constant slippage $\zeta = 0.2$. Furthermore, we assume that each order can be traded in the opening time of each minute bar and the reward is calculated in the closing time. Additionally, exclusive features about the futures including margin, contrast settlement are considered. The training epoch will be broken off once positions are lost by 50% or lacking margin. We initialize our account with $ 500,000 in cash at the beginning of the test. The most widely used criteria of the interest in QT are used to evaluate the policy performance:

- **Total return rate** Tr:= $(P_{end} - P_{start})/P_{start}$ (P is the total value of the position and cash).
- **Sharpe ratio** (Sharpe 1966) Sr:= $\mathbb{E}[\mathbf{r}]/\sigma[\mathbf{r}]$ considers benefits and risks synthetically and reflects the excess return over unit systematic risk.
- **Volatility** Vol:= $\sigma[\mathbf{r}]$ ($\mathbf{r}$ denotes the historical sequence of return rate.) measures the uncertainty of return rate and reflects the risk level of strategies.
- **Maxium Drawdown** (Magdon-Ismail and Atiya 2004) Mdd:= $\max(P_i - P_j)/(P_i), j > i$ measures the largest decline in history and shows the worst possible scenario.

In general, Tr is the most intuitive criterion to evaluate the performance of the trading strategy, while Vol and Mdd measure the uncertainty of return. Sr measures the risk-adjusted return which considers the return and risk synthetically.

Imitation Learning Detail

As the discussions in the Demonstration Buffer module, we select the Dual Thrust strategy as the demonstration trading policy. In the field of technical analysis, Dual Thrust is one of the representative strategies based on oscillators. Specifically, the Dual Thrust strategy determines a reasonable price oscillation interval $Range$ using the sequences of *high* price $\mathbf{P}^h$, *low* price $\mathbf{P}^l$ and *closing* price $\mathbf{P}^c$ among the previous n periods. At each trading day, the upper track $BuyLine$ and the lower track $SellLine$ are decided by adding/subtracting a certain proportion of $Range$ at *day opening* price. These elements in the Dual Thrust can be calculated as:

$$
\begin{aligned}
Range &= \max\left[HH - LC, HC - LL\right], \\
BuyLine &= Open + K_1 \times Range, \\
SellLine &= Open - K_2 \times Range,
\end{aligned}
$$

where $Open$ denotes the opening price of the day; K_1, K_2 are both constants respectively controlling the difficulty of market prices to break $BuyLine$ and $SellLine$; $HH = \max[\mathbf{P}^h_{t-n:t}]$ denotes the highest *high* price of previous n time periods. Similarly, $LC = \min[\mathbf{P}^c_{t-n:t}]; HC = \max[\mathbf{P}^c_{t-n:t}]; LL = \min[\mathbf{P}^l_{t-n:t}]$. In practice, when the current price breaks up a certain percentage of Range up/down, technical traders believe that a trend of up/down is formed. Simultaneously, the long/short positions are taken. At time step t, the demonstration action can be written as:

$$
\breve{a}_t = \begin{cases} 1, & \text{if } p^o_t > BuyLine \\ -1, & \text{if } p^o_t < SellLine \end{cases}
$$

In the Behavior Cloning module, intra-day greedy actions from the hindsight are introduced as expert actions. In hindsight, to take a long position at the lowest price and a short position at the highest price is always a relatively optimal greedy strategy. Such prophetic policy can act as expert actions for the agent during the training procedure. At each time step t the expert action is determined by:

$$
\bar{a}_t = \begin{cases} 1, & \text{if } t = \arg\min \ \mathbf{P}^o_{t-n_d:t} \\ -1, & \text{if } t = \arg\max \ \mathbf{P}^o_{t-n_d:t} \end{cases}
$$

where n_d denotes the length of one trading day; $\mathbf{P}^o_{t-n_d:t}$ denotes the sequence opening price of the day.

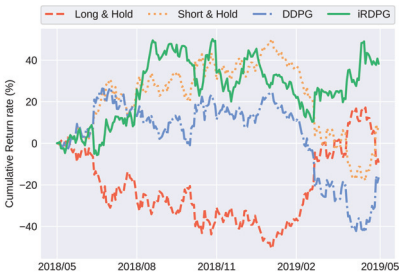


Figure 4: Cumulative return rates of comparison methods.

Baseline Methods

We compare our proposed iRDPG with several baselines.

- **Long & Hold** indicates that we take a long position at the beginning and hold the position till the end of the test period and it is just the return of the futures itself.
- **Short & Hold** indicates that we take a short position at the beginning and hold the position.
- **DDPG** (Lillicrap et al. 2015) is an off-policy model-free actor-critic reinforcement learning algorithm and it usually performs well in continuous control tasks.

Experimental Results

In this section, we conduct experiments to compare our iRDPG with baseline methods. And we conduct ablation experiments to show the function of each module. To compare the generalization abilities of iRDPG and Dual Thrust, we respectively test them in two different futures markets.

Data Representation. Using minute-frequent IF futures data, we conduct experiments with our proposed iRDPG and baselines methods. Their performance on financial criteria are shown in Table 1. Also, we summarize the cumulative return rate for each trading day of tested methods in Figure 4. In Table 1, we can further observe that iRDPG achieves the best performance on almost all criteria, yet DDPG gains

[6] https://www.joinquant.com/

Table 1: Performance of comparison methods on IF.

Methods	Tr (%)	Sr	Vol	Mdd (%)
Long & Hold	-9.30	-0.207	0.656	51.84
Short & Hold	8.53	0.108	0.444	45.30
DDPG	-16.36	-0.378	0.550	54.88
iRDPG	**38.26**	**0.842**	**0.422**	**26.57**

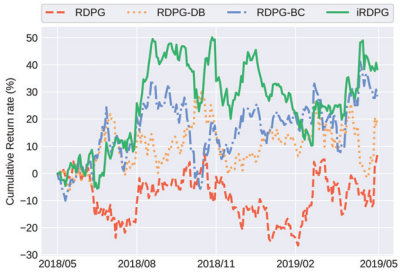


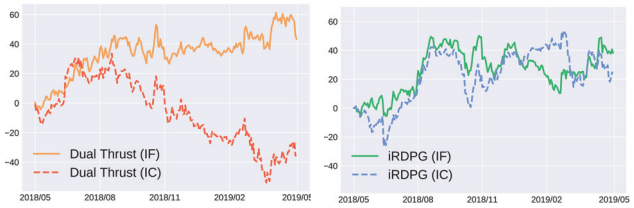
Figure 5: Cumulative return rates of ablation methods.

Table 2: Ablation experiments on IF.

Methods	Tr (%)	Sr	Vol	Mdd (%)
RDPG	6.96	0.060	0.530	32.66
RDPG-DB	19.99	0.377	0.440	**24.68**
RDPG-BC	28.34	0.579	0.436	29.81
iRDPG	**38.26**	**0.842**	**0.422**	26.57

the worst performance. This indicates that the feedforward neural networks in DDPG is weak in representing such high-frequent data. The GRU networks in iRDPG take into account observation and action history recurrently. The experimental results have provided its ability to filter the market noise and extract robust market features. Compared with straightforward strategies (i.e., Long & Hold and Short & Hold), iRDPG gains substantial improvement (Tr: 38.26%, Sr: 0.842). Such beyond market performance reveals that our model is able to gain a high excess return rate, which is always the goal of quantitative traders. This further proves that our trading agent can benefit from the historical information.

Ablation Experiments. We conduct several ablation experiments on IF futures to show how each part of iRDPG affects the final performance. We present ablation experimental results in Table 2 and summarize daily cumulative return rates in Figure 5. As shown in Figure 5, it is clear that the cumulative yield curve of iRDPG move above others for almost test period. In Table 2, there are three variations of iRDPG, each of which takes out one opponent of the full iRDPG. In particular, RDPG refers to iRDPG with only GRU neural networks. RDPG-DB refers to RDPG with the demonstration buffer module while RDPG-BC refers to RDPG with the behavior cloning module. As shown in Table 2, RDPG has a poor performance on all metrics, which shows the agent has difficulty with keeping action consistency and learning a profitable strategy. Compared with RDPG, we can see that RDPG-DB gains an increase in profitability measures (i.e., Tr and Sr). This indicates that the prioritized demonstration buffer boosts the profitability indeed, which may result from the improvement of sampling efficiency. And we notice that RDPG-BC also performs well on profitability measures. This suggests that intra-day expert actions can help the agent reduce costly random exploration. However, if we focus on the Vol value of RDPG-DB (0.440) and RDPG-BC (0.436), we can find that they are both risk-sensitive. Note that iRDPG outperforms the above methods on almost all criteria. In other words, our proposed model can generate good returns as well as resist risks, which shows the effectiveness of imitation learning.



(a) Dual Thrust (b) iRDPG

Figure 6: Cumulative return rates of of Dual Thrust and iRDPG on IF and IC.

Table 3: Performance of comparison methods on IF and IC.

Methods	Data	Tr(%)	Sr	Vol	Mdd(%)
Dual Thrust	IF	43.40	1.110	0.369	17.24
	IC	-35.59	-0.577	0.700	65.51
iRDPG	IF	38.26	0.842	0.422	26.57
	IC	24.73	0.413	0.521	29.35

Generalization Ability. We further present the generalization ability of iRDPG in different markets (i.e., IF and IC). In Table 3 we show the performance of Dual Thrust and iRDPG in these two markets. Meanwhile, their daily cumulative return rates in IF and IC are summarized are plotted in Figure 6. In Table 3 and Figure 6, if we focus on the comparison of Dual Thrust strategy on IF and IC, we will find that the performance of the identical strategy is utterly different on two futures markets. However, it is clear that in Figure 3, the IF has a similar price trend with IC. This further proves that strategies based on static technical indicators have poor generalization abilities. For better comparison, we train our agent in the IF training set and test it in the IF and IC test set respectively. Though the Dual Thrust strategy little better than iRDPG in IF, we can see that iRDPG outdistances the Dual Thrust in IC. Note that iRDPG has never been trained by IC. Such performance proves that our proposed model is adaptive for different markets, and our agent can extract robust features and learn dynamic trading strategies.

Conclusion

In our work, we proposed iRDPG, an adaptive imitative model for QT problem. We designed a POMDP framework for the representation of noisy minute-frequent financial

data. In addtion, we implemented imitation learning techniques to balance the exploration and exploitation of the trading agent. Our model was tested on the real stock-index futures data with practical constraints. The profitability and ability to resist risks of iRDPG were verified. Furthermore, comparison experiments provided its generalization ability for different financial markets. Overall, our iRDPG suggests that the trading agent in real financial market can benefit from experiences of classical trading strategies.

Acknowledgements

This research was supported by grants from the National Natural Science Foundation of China (Grants No. 61922073, 61672483, U1605251, 71790594). Qi Liu acknowledges the support of the Young Elite Scientist Sponsorship Program of CAST and the Youth Innovation Promotion Association of CAS (No. 2014299).

References

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2015. Gated feedback recurrent neural networks. In *ICML*.

Deng, Y.; Bao, F.; Kong, Y.; Ren, Z.; and Dai, Q. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems* 28(3):653–664.

Feng, F.; He, X.; Wang, X.; Luo, C.; Liu, Y.; and Chua, T.-S. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)* 37(2):27.

François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M. G.; Pineau, J.; et al. 2018. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning* 11(3-4):219–354.

Heess, N.; Hunt, J. J.; Lillicrap, T. P.; and Silver, D. 2015. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*.

Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Osband, I.; et al. 2018. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jiang, Z.; Xu, D.; and Liang, J. 2017. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*.

Jin, B.; Zhao, H.; Chen, E.; Liu, Q.; and Ge, Y. 2019. Estimating the days to success of campaigns in crowdfunding: A deep survival perspective. In *Thirty-Third AAAI Conference on Artificial Intelligence*.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101(1-2):99–134.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4:237–285.

Kim, H.-j., and Shin, K.-s. 2007. A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets. *Applied Soft Computing* 7(2):569–576.

Li, Q.; Jiang, L.; Li, P.; and Chen, H. 2015. Tensor-based learning for predicting stock movements. In *Twenty-Ninth AAAI*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *ICLR*.

Magdon-Ismail, M., and Atiya, A. F. 2004. Maximum drawdown. *Risk Magazine* 17(10):99–102.

Malkiel, B. G., and Fama, E. F. 1970. Efficient capital markets: A review of theory and empirical work. *The journal of Finance* 25(2):383–417.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Moody, J. E., and Saffell, M. 1999. Reinforcement learning for trading. In *NeurIPS*, 917–923.

Moody, J., and Saffell, M. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* 12(4):875–889.

Murphy, J. J. 1999. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

Nair, A.; McGrew, B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *ICRA*.

Neuneier, R. 1996. Optimal asset allocation using adaptive dynamic programming. In *NeurIPS*, 952–958.

Pruitt, G., and Hill, J. R. 2012. *Building Winning Trading Systems with Tradestation,+ Website*, volume 542. John Wiley & Sons.

Ross, S., and Bagnell, D. 2010. Efficient reductions for imitation learning. In *AISTATS*, 661–668.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Sharpe, W. F. 1966. Mutual fund performance. *The Journal of business* 39(1):119–138.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *ICML*.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, 1057–1063.

Večerík, M.; Hester, T.; Scholz, J.; Wang, F.; Pietquin, O.; Piot, B.; Heess, N.; Rothörl, T.; Lampe, T.; and Riedmiller, M. 2017. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*.

Yu, P.; Lee, J. S.; Kulyatin, I.; Shi, Z.; and Dasgupta, S. 2019. Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*.

Zhao, H.; Liu, Q.; Zhu, H.; Ge, Y.; Chen, E.; Zhu, Y.; and Du, J. 2017a. A sequential approach to market state modeling and analysis in online p2p lending. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48(1):21–33.

Zhao, H.; Zhang, H.; Ge, Y.; Liu, Q.; Chen, E.; Li, H.; and Wu, L. 2017b. Tracking the dynamics in crowdfunding. In *SIGKDD*, 625–634. ACM.