2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Algorithmic Trading Behavior Identification using Reward Learning Method

Steve Y. Yang, Qifeng Qiao, Peter A. Beling, and William T. Scherer

Abstract— **Identifying and understanding the impact of algorithmic trading on financial markets has become a critical issue for market operators and regulators. Advanced data feed and audit trail information from market operators now make the full observation of market participants' actions possible. A key question is the extent to which it is possible to understand and characterize the behavior of individual participants from observations of trading actions. In this paper, we consider the basic problems of categorizing and recognizing traders (or, equivalently, trading algorithms) on the basis observed limit orders. Our approach, which is based on inverse reinforcement learning (IRL), is to model trading decisions as a Markov decision process and then use observations of an optimal decision policy to find the reward function. The approach strikes a balance between two desirable features in that it captures key empirical properties of order book dynamics and yet remains computationally tractable. Making use of a real-world data set from the E-Mini futures contract, we compare two principal IRL variants, linear IRL and Gaussian process IRL. Results suggest that IRL-based feature spaces support accurate classification and meaningful clustering.**

Index Terms— **Inverse Reinforcement Learning; Gaussian Process; High Frequency Trading; Algorithmic Trading; Behavioral Finance; Markov Decision Process; Support Vector Machine**

I. Introduction

FINANCIAL markets have changed dramatically over the past 10 years or so. These changes reflect the culmination of a decade-long trend from a market structure with primarily manual floor trading to a market structure dominated by automated computer trading. This rapid transformation has been driven by the evolution of technologies for generating, routing, and executing orders, which have dramatically improved the speed, capacity, and sophistication of the trading functions that are available to market participants.

High-quality trading markets promote capital formation and allocation by establishing prices for securities and by enabling investors to enter and exit their positions in securities wherever and whenever they wish to do so. The one important feature of all types of algorithmic trading strategies is to discover the underlying persistent tradable phenomena and generate trading opportunities. These trading opportunities include microsecond price movements that allow a trader to benefit from market-making trades, several minute-long strategies that trade on momentum forecasted by market microstructure theories, and several hour-long market movements that surround recurring events and deviations from statistical relationship ([1]). Algorithmic traders then design their trading algorithms and systems with the aim of generating signals that result in consistent positive outcomes under different market conditions.

In the past few years, there have been a number of studies of HFT and algorithmic trading more generally. Their primary objective is to understand the economic impact of these algorithmic trading practices to the market quality including liquidity, price discovery process, trading costs, etc. There have been a number of studies focused on algorithmic traders' behaviors. These studies examine the trading activities of different types of traders and try to distinguish their behavioral differences. Hendershott et al. ([2]) use exchange classifications that distinguish algorithmic traders from orders managed by humans. They find that algorithmic traders concentrate in smaller trade sizes, while large block trades of 5,000 shares or more are predominantly originated by human traders. Algorithmic traders consume liquidity when bid-ask spreads are relatively narrow, they supply liquidity when bid-ask spreads are relatively wide. This suggests that algorithmic traders provide a more consistent level of liquidity through time. Broggard ([3]) and Herdershoot et al. ([23]) work with Nasdaq data that flag whether trades involves HFT. Herdershott et al. ([23]) find that HFT accounts for about 42% of (double-counted) Nasdaq volume in large-cap stocks but only about 17% of volume in small-cap stocks. They estimate a state-space model that decomposes price changes into permanent and temporary components, and measures the contribution of HFT and non-HFT liquidity supply and liquidity demand to each of these price change components. They find that when HFTs initiate trades, they trade in the opposite direction to the transitory component of prices. Thus, HFTs contribute to price discovery and contribute to efficient stock prices. Brogaard ([3]) similarly finds that 68% of trades have an HFT on at least one side of the transaction, and he also finds that HFT participation rates are higher for stocks with high share prices, large market caps, narrow bid-ask spreads, or low stock-specific volatility. He estimates a vector autoregressive permanent price impact model and finds that HFT liquidity suppliers face less adverse selection than non-HFT liquidity suppliers, suggesting that they are somewhat judicious in supplying liquidity. Kirilenko et al. ([4]) use account-level tick-by-tick data on the E-Mini S&P 500 futures contract, and they classify traders into

Steve Y. Yang is with the Financial Engineering Program in School of Systems and Enterprises at Stevens Institute of Technology (email: steve.yang@stevens.edu).

Qifeng Qiao is with the Department of Systems and Information Engineering at University of Virginia (email: qq2r@virginia.edu).

Peter A. Beling is with the Department of Systems and Information Engineering at University of Virginia (email: pb3a@virginia.edu).

William T. Scherer is with the Department of Systems and Information Engineering at University of Virginia (email: wts@virginia.edu).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

various categories, including HFTs, opportunistic traders, fundamental traders and noise traders. Benos et al. ([5]) conduct a similar analysis using UK equity data. These different datasets provide considerable insight into overall HFT trading behavior.

Traders aim to optimize their decisions overtime and consequently maximize their reward under different market conditions. We can theoretically use reward functions to represent the value system that are encapsulated in the various different trading strategies. It is possible to derive new policies based on the reward functions learned and apply them in a new environment to govern a new autonomous process. This process is defined as reward learning under the framework of inverse reinforcement learning (IRL) ([24], [25], [6]). For example, a simple keep-or-cancel strategy for buying one unit, the trader has to decide when to place the order and when to cancel the order based on the market condition which may likely be characterized as a stochastic process. However the value proposition for the trader is to buy one unit of the security at the lowest price possible. This could be realized in a number of ways. It could be described as a reward function meaning when the system is in a particular state, the trader is always looking for a fixed reward. This notion of value proposition drives the trader to take corresponding actions according to the market conditions. This ultimately constitutes trader's policies or strategies. Therefore a strategy under certain value proposition can be consistently programmed in algorithms to achieve its goal of buy-one-unit in an optimal way. Consequently, strategies developed under certain value frameworks can be observed, learned and even reproduced in a different environment (such as a simulated financial market where impact of these strategies can be readily assessed). As documented in [8], [9], [10], manipulative or disruptive algorithmic strategies can be studied and monitored by market operators and regulators to prevent unfair trading practices. Furthermore, new emerging algorithmic trading practices can be assessed and new regulations and policies can be evaluated to maintain the overall health of the financial markets.

In this study, we model the trading behavior of different market participants in terms of a Markov decision process (MDP). In this model, states are defined in terms of the quantity of orders in each of a coarse set of bins imposed upon the limit order book. Actions are limited to initiating limit or market orders or canceling existing orders. IRL is used to learn the reward function for the MDP on the basis of observations of trader actions.

IRL was first introduced in machine learning literature by Ng et al. ([24]) in formulating it as an optimization problem to maximize the sum of differences between the quality of the optimal action and the quality of the next-best action. It is founded on the presupposition that the reward function, rather than policy, is the most succinct, robust, and transferable definition of the task. However, the reward function is often difficult to know in advance for some real-world tasks, which gives motivation to the idea of learning from observations. Technical approaches to learning from observations generally fall into two broad categories ([7]). The first category, called imitation learning, attempts to use supervised learning to predict actions directly from observations of features of the environments, which is unstable and vulnerable to highly uncertain environment. The second category is concerned with how to learn the reward function that characterizes the agent's objectives and preferences in MDP ([24]). The principal idea of apprenticeship learning using IRL is to search mixed solutions in a space of learned policies with the goal that the cumulative feature expectation is near that of the expert ([25] and [15]).

The remainder of this paper is organized as follows: First, we discuss the framework of which we use to model market dynamics and the traders' decisions. In section III, we extend the MDP and introduce IRL formulation. We discuss the original linear IRL formulation and provide a Bayesian probabilistic model to infer the reward function using Gaussian processes IRL (GPIRL). We apply the GPIRL algorithm to the E-Mini S&P 500 Futures market as experiments in section IV-B. We show that the GPIRL approach can accurately capture algorithmic trading behavior based on observations of the high frequency data. Finally in Section V we provide concluding remarks about the GPIRL and its applications.

II. Markov Decision Process Model of Market Dynamics

In this section, we develop a Markov decision process (MDP) model of trader behavior. This model will then serve as the basis for the inverse reinforcement learning process described in section III.

A. MDP Background and Notation

The primary aim of our trading behavior-based learning approach is to uncover decision makers' policies and reward functions through the observations of an expert whose decision process is modeled as an MDP. In this paper, we restrict our attention to a finite countable MDP for easy exposition, but our approach can be extended to continuous problems if desired. A discounted finite MDP is defined as a tuple $\mathbb{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, r)$, where

- $\mathcal{S} = \{s_n\}_{n=1}^{N}$ is a set of N states. Let $\mathcal{N} = \{1, 2, \cdots, N\}$.
- $\mathcal{A} = \{a_m\}_{m=1}^{M}$ is a set of M actions. Let $\mathcal{M} = \{1, 2, \cdots, M\}$.
- $\mathcal{P} = \{\mathbf{P}_{a_m}\}_{m=1}^{M}$ is a set of state transition probabilities (here $\mathbf{P}_{a_m}$ is a $N \times N$ matrix where each row, denoted as $\mathbf{P}_{a_m}(s_n, :)$, contains the transition probabilities upon taking action a_m in state s_n. The entry $\mathbf{P}_{a_m}(s_n, s_{n'})$ is the probability of moving to state $s_{n'}, n' \in \mathcal{N}$ in the next stage.).
- $\gamma \in [0, 1]$ is a discount factor.
- r denotes the reward function, mapping from $\mathcal{S} \times \mathcal{A}$ to $\Re$ with the property that

$$r(s_n, a_m) \triangleq \sum_{n' \in \mathcal{N}} \mathbf{P}_{a_m}(s_n, s_{n'}) r(s_n, a_m, s_{n'})$$

where $r(s_n, a_m, s_{n'})$ denotes the function giving the reward of moving to the next state $s_{n'}$ after taking action a_m in current state s_n. The reward function $r(s_n, a_m)$ may be further reduced to $r(s_n)$, if we neglect the influence of the action. We use $\mathbf{r}$ for reward vector through out this paper. If the reward only depends on state, we have $\mathbf{r} = (\mathbf{r}(s_1), \ldots, \mathbf{r}(s_N))$. If we let $\mathbf{r}$ be the vector of the reward depending on both state and action, we have

$$\begin{aligned}\mathbf{r} &= (\underbrace{\mathbf{r}_1(s_1), ..., \mathbf{r}_1(s_N)}, \ldots, \underbrace{\mathbf{r}_M(s_1), \ldots, \mathbf{r}_M(s_N)}) \\ &= (\qquad\quad \mathbf{r}_1, \qquad \cdots, \qquad\quad \mathbf{r}_M).\end{aligned}$$

B. Optimal Markov Decision Model

In an MDP, the agent selects an action at each sequential stage, and we define a *policy* (*behavior*) as the way that the actions are selected by a decision maker/agent. Hence this process can be described as a mapping between state and action, i.e., a random state-action sequence $\left(s^0, a^0, s^1, a^1, \cdots s^t, a^t, \cdots\right)$, [1] where s^{t+1} is connected to (s^t, a^t) by $\mathbf{P}_{a^t}(s^t, s^{t+1})$.

We also define rational agents as those that behave according to the optimal decision rule where each action selected at any stage maximizes the value function. The *value function* for a policy π evaluated at any state s^0 is given as $V^\pi(s^0) = E[\sum_{t=0}^{\infty} \gamma^t r(s^t, a^t)|\pi]$. This expectation is over the distribution of the state sequence $\left\{s^0, s^1, ...\right\}$ given the policy $\pi = \left\{\mu^0, \mu^1, \cdots\right\}$, where $a^t = \mu^t(s^t)$, $\mu^t(s^t) \in U(s^t)$ and $U(s^t) \subset \mathcal{A}$. The objective at state s is to choose a policy that maximizes the value of $V^\pi(s)$. The optimal policy is then $V^*(s^0) = \sup_\pi E[\sum_{t=0}^{\infty} \gamma^t r(s^t, a^t)|\pi]$. Similarly, there is another function called the *Q-function* (or *Q-factor*) that judges how well an action is performed in a given state. The notation $Q^\pi(s, a)$ represents the expected return from state s when action a is taken and thereafter policy π is followed.

III. Inverse Reinforcement Learning

Given an MDP $\mathbb{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, r)$, let us define the inverse Markov decision process (IMDP) $\mathbb{M}_{\mathbb{I}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{O})$. The process $\mathbb{M}_{\mathbb{I}}$ includes the states, actions, and dynamics of $\mathbb{M}$, but lacks a specification of the reward function, r. By way of compensation, $\mathbb{M}_{\mathbb{I}}$ includes a set of observations $\mathcal{O}$ that consists of state-action pairs generated through the observation of a decision maker. We can define the *inverse reinforcement learning* (IRL) problem associated with $\mathbb{M}_{\mathbb{I}} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{O})$ to be that of finding a reward function such that the observations $\mathcal{O}$ could have come from an optimal policy for $\mathbb{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, r)$. The IRL problem is, in general, highly under-specified, which has led researchers to consider various models for restricting the set of reward functions under consideration. Ng et al. ([24]), in a seminal consideration of IMDPs and associated IRL problems, observed that, by the optimality equations, the only reward vectors consistent with an optimal policy π are those that satisfy the set of inequalities

$$(\mathbf{P}_\pi - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{r} \geq \mathbf{0}, \forall a \in \mathcal{A}, \tag{1}$$

where $\mathbf{P}_\pi$ is the transition probability matrix relating to observed policy π and $\mathbf{P}_a$ denotes the transition probability matrix for other actions. Note that the trivial solution $\mathbf{r} = \mathbf{0}$ satisfies the constraints (1), which highlights the under-specified nature of the problem and the need for reward selection mechanisms. In this study, we learn the reward function with IRL and then directly use the rewards as features for classifying and clustering traders or trading algorithms.

A. Linear IRL

Ng et al. ([24]) advance the idea choosing the reward function to maximize the difference between the optimal and suboptimal policies, which can be done using a linear programming formulation.

Most of the existing IRL algorithms make some assumption about the form of the reward function. Prominent examples include the model ([24]), which we term linear IRL (LIRL) because of its linear nature. In LIRL, the reward function is written linearly in terms of basis functions, and effort is made to maximize the quantity

$$\sum_{s \in \mathcal{S}} [Q^\pi(s, a') - \max_{a \in \mathcal{A} \setminus a'} Q^\pi(s, a)], \forall a \in \mathcal{A}. \tag{2}$$

The optimization problem in [24] is equivalent to the following optimization program:

$$\begin{aligned} \max_{\mathbf{r}} \textstyle\sum_{s \in \mathcal{S}} \beta(s) \quad & -\lambda \textstyle\sum_{s \in \mathcal{S}} |\mathbf{r}(s)| \\ \text{s.t.} & \\ (\mathbf{P}_\pi - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{r} \quad & \geq \beta(s), \quad \forall a \in \mathcal{A}, \forall s \in \mathcal{S} \\ \beta(s) \quad & \geq \mathbf{0}, \quad \forall s \in \mathcal{S}, \end{aligned}$$

where λ is a regularization parameter included to encourage sparse solution vectors. Yang et al. ([8]) used this approach to find a feature space that can be used to classify and cluster simulated trading agents.

B. Bayesian IRL Framework

Ramachanjan et al. ([6]) originally proposed a Bayesian Framework for IRL. The posterior over reward is written as

$$p(r|\mathcal{O}) = p(\mathcal{O}|r)p(r) \propto \prod_{(s,a) \in \mathcal{O}} p(a|s, r).$$

Then, the IRL problem is written as $\max_r \log p(\mathcal{O}|r) + \log p(r)$. For many problems, however, the computation of $p(r|\mathcal{O})$ may be complicated and some algorithms use Markov chain Monte Carlo (MCMC) to sample the posterior probability. Below we adopt a different approach that uses the idea of selecting reward on the basis of *maximum a posteriori* (MAP) estimate computed using convex optimization.

[1] Superscripts represent time indices. For example s^t and a^t, with the upper-index $t \in \{1, 2, \cdots\}$, denote state and action at the t-th horizon stage, while s_n (or a_m) represents the n-th state (or m-th action) in $\mathcal{S}$ (or $\mathcal{A}$).

C. Gaussian Process IRL

Given the complex state space associated with trading, one may observe a trading strategy at length with out learning the entirety of the policy. Trading strategies vary in the time horizons over which they are defined. Therefore, the observation period becomes critical to the learning process. Furthermore, two types of errors may be introduced into our observations: The first type of error may be introduced during our modeling process. Resolution of these discrete models will introduce errors into our observations. The second potential source of error is the strategy execution process. Execution errors will occur due to the uncertainty of market movements and will eventually appear in our observations, confounding our efforts to determine the true policy. Overall, there are two types of challenges in this learning problem: the uncertainty about reward functions given the observation of decision behavior and the ambiguity involved in observing multiple actions at a single state.

Qiao and Beling ([29]) argue for two different modeling techniques in learning reward functions. To lessen the ambiguity of observing multiple actions at a state, they argue that Bayesian inference should be the basis for understanding the agent's preferences over the action space. This argument is reasonable because the goal of IRL is to learn the reward subjectively perceived by the decision maker from whom we have collected the observation data. The intuition is that decision makers will select some actions at a given state because they prefer these actions to others. These preferences are among the countable actions that can be used to represent multiple observations at one state.

In the following, we first introduce the preference theory for the IMDP model, and then we formalize the idea of modeling the reward function as a Gaussian process under the Bayesian inference framework.

1) Action Preference Learning: In this section, we first define the action preference relationship and the action preference graph. At state $s_n, \forall \hat{a}, \check{a} \in \mathcal{A}$, we define the *action preference relation* as:

1) Action $\hat{a}$ is weakly preferred to $\check{a}$, denoted as $\hat{a} \succeq_{s_n} \check{a}$, if $Q(s_n, \hat{a}) \geq Q(s_n, \check{a})$;
2) Action $\hat{a}$ is strictly preferred to $\check{a}$, denoted as $\hat{a} \succ_{s_n} \check{a}$, if $Q(s_n, \hat{a}) > Q(s_n, \check{a})$;
3) Action $\hat{a}$ is equivalent to $\check{a}$, denoted as $\hat{a} \sim_{s_n} \check{a}$, if and only if $\hat{a} \succeq_{s_n} \check{a}$ and $\check{a} \succeq_{s_n} \hat{a}$.

An *action preference graph* is a simple directed graph showing preference relations among the countable actions at a given state. At state s_n, the action preference graph $G_n = (\mathcal{V}_n, \mathcal{E}_n)$ comprises a set $\mathcal{V}_n$ of nodes together with a set $\mathcal{E}_n$ of edges. For the nodes and edges in graph G_n, let us define

1) Each node represents an action in $\mathcal{A}$. Define a one-to-one mapping $\varphi : \mathcal{V}_n \to \mathcal{A}$.
2) Each edge indicates a preference relation.

Furthermore, we make the following assumption as a rule to build the preference graph, and then we show how to draw a preference graph at state s_n:

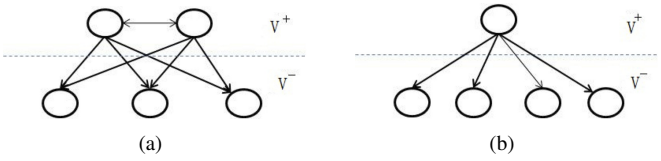


Fig. 1. **Examples Preference Graphs:** (a) An example of observing two actions at a state. (b) An example of a unique observation at a state.

At state s_n, if action $\hat{a}$ is observed, we have the following preference relations: $\hat{a} \succeq_{s_n} \check{a}, \forall \check{a} \in \mathcal{A} \setminus \{\hat{a}\}$.

The variable $\hat{a}$ is observed if and only if $\hat{a} \in \arg\max_{a \in \mathcal{A}} Q(s_n, a)$. Therefore, we have

$$Q(s_n, \hat{a}) > Q(s_n, \check{a}), \ \forall \check{a} \in \mathcal{A} \setminus \{\hat{a}\}$$

According to the definition of preference relations, it follows that if $Q(s_n, \hat{a}) > Q(s_n, \check{a})$, we have $\hat{a} \succ_{s_n} \check{a}$. Hence, we can show that the preference relationship has the following properties:

1) If $\hat{a}, \check{a} \in \mathcal{A}$, then at state s_n either $\hat{a} \succeq_{s_n} \check{a}$ or $\check{a} \succeq_{s_n} \hat{a}$.
2) If $\hat{a} \succeq_{s_n} \check{a}$ or $\check{a} \succeq_{s_n} \tilde{a}$, then $\hat{a} \succeq_{s_n} \tilde{a}$.

According to Qiao and Beling ([29]), we can represent $\mathcal{O}$ as shown in Figure (1). At state s_n, its action preference graph is constructed by a two-layer directed graph: a set of nodes $\mathcal{V}_n^+$ in the top layer and a set of nodes $\mathcal{V}_n^-$ in the bottom layer. Under the non-deterministic policy assumption, we adopt a reward structure depending on both state and action.

2) Gaussian Reward Process: Recall that the reward depends on both state and action, and consider r_m, the reward related to action a_m, as a Gaussian process. We denote by $k_m(s_i, s_j)$ the function generating the value of entry (i, j) for covariance matrix $\mathbf{K}_m$, which leads to $\mathbf{r}_m \sim N(0, \mathbf{K}_m)$. Then the joint prior probability of the reward is a product of multivariate Gaussian, namely $p(\mathbf{r}|\mathcal{S}) = \prod_{m=1}^{M} p(\mathbf{r}_m|\mathcal{S})$ and $\mathbf{r} \sim N(0, \mathbf{K})$. Note that $\mathbf{r}$ is completely specified by the positive definite covariance matrix $\mathbf{K}$, which is block diagonal in the covariance matrices $\{\mathbf{K}_1, \mathbf{K}_2 ..., \mathbf{K}_M\}$ based on the assumption that the reward latent processes are uncorrelated. In practice, we use a squared exponential kernel function, written as:

$$k_m(s_i, s_j) = e^{\frac{1}{2}(s_i - s_j)\mathbf{T}_m(s_i - s_j)} + \sigma_m^2 \delta(s_i, s_j),$$

where $\mathbf{T}_m = \kappa_m \mathbf{I}$ and $\mathbf{I}$ is an identity matrix. The function $\delta(s_i, s_j) = 1$, when $s_i = s_j$; otherwise $\delta(s_i, s_j) = 0$. Under this definition the covariance is almost unity between variables whose inputs are very close in the Euclidean space, and decreases as their distance increases.

Then, the GPIRL algorithm estimates the reward function by iteratively conducting the following two main steps:

(a) Get estimation of $\mathbf{r}_{MAP}$ by maximizing the posterior $p(\mathbf{r}|\mathcal{O})$, which is equal to minimize $-\log p(\mathcal{O}|\mathbf{r}) - \log p(\mathbf{r}|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the vector of hyper-parameters including κ_m and σ_m that control the Gaussian process.

(b) Optimize the hyper-parameters by using gradient decent method to maximize $\log p(\mathcal{O}|\boldsymbol{\theta}, \mathbf{r}_{MAP})$, which is the Laplace approximation of $p(\boldsymbol{\theta}|\mathcal{O})$.

3) Likelihood Function and MAP Optimization: GPIRL adopts the following likelihood functions to capture the strict preference and equivalent preference respectively.

$$p((\hat{a} \succ_{s_n} \check{a})_k|\mathbf{r}) = \Phi(\frac{Q(s_n, \hat{a}) - Q(s_n, \check{a})}{\sqrt{2}\sigma}) \tag{3}$$

$$p((\hat{a} \sim_{s_n} \hat{a}')_l|\mathbf{r}) \propto e^{-\frac{1}{2}(Q(s_n,\hat{a}) - Q(s_n,\hat{a}'))^2} \tag{4}$$

In Eq. 3, the function $\Phi(x) = \int_{-\infty}^{x} N(v|0,1)dv$, where $N(v|0,1)$ denotes a standard Gaussian variable.

The probabilistic IRL model is controlled by the kernel parameters κ_m and σ_m which compute the covariance matrix of reward realizations, and by σ which tunes the noise level in the likelihood function. We put these parameters into the hyper-parameter vector $\boldsymbol{\theta} = (\kappa_m, \sigma_m, \sigma)$. More often than not, we do not have prior knowledge about the hyper-parameters. And then we can apply maximum a posterior estimate to evaluate the hyper-parameters.

We use $\mathcal{G}$ to denote the action preference graph. Essentially, we now have a hierarchical model. At the lowest level, we have reward function values encoded as a parameter vector $\mathbf{r}$. At the top level, we have hyper-parameters in $\boldsymbol{\theta}$ controlling the distribution of the parameters. Inference takes place one level at a time. At the bottom level, the posterior over function values is given by Bayes' rule:

$$p(\mathbf{r}|\mathcal{S}, \mathcal{G}, \boldsymbol{\theta}) = \frac{p(\mathcal{G}|\mathcal{S}, \boldsymbol{\theta}, \mathbf{r})p(\mathbf{r}|\mathcal{S}, \boldsymbol{\theta})}{p(\mathcal{G}|\mathcal{S}, \boldsymbol{\theta})}. \tag{5}$$

The posterior combines the prior information with the data, reflecting the updated belief about $\mathbf{r}$ after observing the decision behavior. We can calculate the denominator in Eq.5 by integrating $p(\mathcal{G}|\mathcal{S}, \boldsymbol{\theta}, \mathbf{r})$ over the function space with respect to $\mathbf{r}$, which requires a high computational capacity. Fortunately, we are able to maximize the non-normalized posterior density of $\mathbf{r}$ without calculating the normalizing denominator, as the denominator $p(\mathcal{G}|\mathcal{S}, \boldsymbol{\theta})$ is independent of the values of $\mathbf{r}$. In practice, we obtain the maximum posterior by minimizing the negative log posterior, which is shown to be convex in [29].

IV. E-Mini Market Data Description

The E-Mini S&P 500 is a stock market index of futures contracts traded on the Chicago Mercantile Exchange's (CME) Globex electronic trading platform. The notional value of one contract is $50 times the value of the S&P 500 stock index. The tick size for the E-Mini S&P 500 is 0.25 index points or $12.50. The CME Globex matching algorithm for the E-Mini S&P 500 offers strict price and time priority. Specifically, limit orders that offer more favorable terms of trade (sell at lower prices and buy at higher prices) are executed prior to pre-existing orders. Orders that arrived earlier are matched against the orders from the other side of the book before other orders at the same price. This market operates under complete price transparency. This straight forward matching algorithm allows us to reconstruct the order book using audit trail messages archived by the exchanges and allows us to replay the market dynamics at any given moment.

In this paper, empirical work is based on a month of E-Mini order book audit trail data. The audit trail data includes all the order book events timestamped at a millisecond time resolution, and contains the following data fields: date, time (the time when the client submits the order to the exchange), conf_time (the time when the order is confirmed by the matching engine), customer account, tag 50 (trader identification number), buy or sell flag, price, quantity, order ID, order type (market or limit), and func_code (message type, e.g. order, modification, cancellation, trade, etc.).

A. Constructing an MDP Model from Order Book Data

The order book audit trail data contains the entire order history i.e. order creation, order modifications, fills, cancellation, etc. To construct an MDP model of trader behavior, we first reconstruct the limit order book using the audit trail messages. The order book then contains bid/ask prices, market depth, liquidity, etc. During this process on the E-Mini data, we processed billions of messages for each trading date, and built price queues using the price and time priority rule.

Once we have the order book at any given event tick, we take the market depth at five different levels as our base variables and then discretize these variables to generate an MDP model state space. This study extends the MDP model documented by Yang et al. ([8]) to obtain five variables, i.e., order volume imbalance between the best bid and the best ask prices, order volume imbalance between the 2nd best bid and the 2nd best ask prices, order volume imbalance between the 3rd best bid and the 3rd best ask prices, the order book imbalance at the 5th best bid and the 5th ask prices, and the inventory level/holding position (see Figure 2 (b)). Then we discretize the values of the five variables into three levels defined as high (above $\mu + 1.96\sigma$), neutral ($\mu \pm 1.96\sigma$), and low (below $\mu - 1.96\sigma$). Based on our observation that the first 3 best bid and ask prices change the most, we select the first 3 level order book imbalance variables in modeling the limit order book dynamics. As argued by Yang et al. ([8]), these volume-related variables reflect the market dynamics on which the traders/algorithms depend to place their orders at different prices.

As the volume imbalance at the best bid/ask prices is the most sensitive indicator of the trading behavior of HFTs, Intermediaries and some of the Opportunistic traders, we also hypothesize that the volume imbalance at other prices close to the book prices will be useful in inferring trader behavior. As demonstrated in previous work ([8]), the private variable of a trader's inventory level provides critical information about trader's behavior. Traders in high frequency environments strive to control their inventory levels as a

critical measure of controlling the risk of their position ([4] and [22]). HFTs and Market Makers tend to turn over their inventory level five or more times a day and to hold very small or even zero inventory positions at the end of the trading session. These observations provide strong support for the introduction of a position variable to characterize trader behavior in our model. Therefore, together with the volume imbalance variables, we propose a computational model with $3^5 = 243$ states.

Next, we need to define the action space. In general, there are three types of actions: placing a new order, canceling an existing order, or placing a market order. We divide the limit order book into 10 buckets at any given point of time by the following price markers: the best bid price, the 2nd best bid price, the 3rd best bid price, between the 4th and 5th bid prices, below the 5th best bid price, the best ask price, the 2nd best ask price, the 3rd best ask price, between the 4th and 5th ask prices, and above the 5th best ask price. Then, at any given point of time, a trader can take 22 actions. The price markers used to define the price ranges are illustrated in Figure (2).

B. Experiment with the E-Mini S&P 500 Futures Market

In this section, we conduct an experiment using the MDP model defined earlier to identify algorithmic trading strategies. We consider the six trader classes defined by Kirilenko et al. ([4]), namely High Frequency Traders, Market Makers, Opportunistic Traders, Fundamental Buyers, Fundamental Sellers and Small Traders. As we argue earlier, the focus of our study will be more on HFTs and Market Makers due to the large daily volume and their potential impact to the financial markets. In Kirilenko et al. ([4])'s paper, there are only about from 16 to 20 HFTs on the S&P500 Emini market. Although this is a small population, their impact to the market has drawn increased attention from policy makers, regulators and academia. That is why we focus our attention on this small population. Among the roughly 10,000 trading accounts for the S&P500 Emini market, we narrow down to about 120 accounts based their high daily trading volume. In the experiment, we select the top 10 trading accounts by their volume and end-of-the-day positions, which guarantees our subjects are HTFs, the high impact population of algorithmic trading strategies.

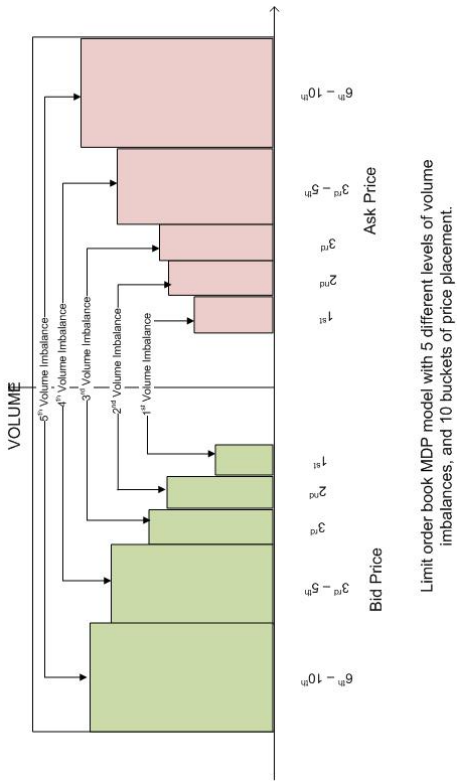


Fig. 2. **Order Book MDP Model:** This graph shows the state variables used in the MDP model.

C. Trader Behavior Identification

Yang et al. ([8]) examine different trading behaviors using a linear IRL (LNIRL) algorithm with the simulated E-Mini S&P 500 market data. That MDP model contains three variables: the volume imbalance at the bid/ask prices, the volume imbalance at the 3rd best bid/ask prices, and the position level. Although this MDP model is relatively simple, it is evident from the experimental results that the IRL reward space is effective in identifying trading strategies with a relatively high accuracy rate.

D. Multi-class SVM Trader Classifier using GPIRL vs. LNIRL

In this section, we use support vector machine (SVM) classification method to identify traders based on reward functions that we recover from the observations of the trader's behaviors. We select a group of traders whose behaviors are consistently observed during the period we study. The primary reason for choosing SVM classification method is its flexibility that we can explore feature separation in different high dimensional spaces using kernel functions.

We select 10 trading accounts with the highest average daily trading volume over a period of 4 weeks (20 days) in our experiment. We define an observation instance as a continuous period covering two hours where we take all the activities from a particular trader including placing new orders, modifying and canceling existing orders, and placing market orders. For each trader, we take four observation instances on each trading day: two observation instances in the morning trading and two observation instances in the afternoon trading. The two observation periods in the morning and in the afternoon have one hour overlapping time, but the observations in the morning and the afternoon do not overlap. We do so based on the general theory of intraday U-shaped patterns in volume - namely, the heavy trading in the beginning and the end of the trading day and the relatively light trading in the middle of of the day. This has been documented in a number of studies ([16], [17], [18], and [19]).

We assume stationary policies for the MDP decision process. We perform an empirical test to verify this assumption.

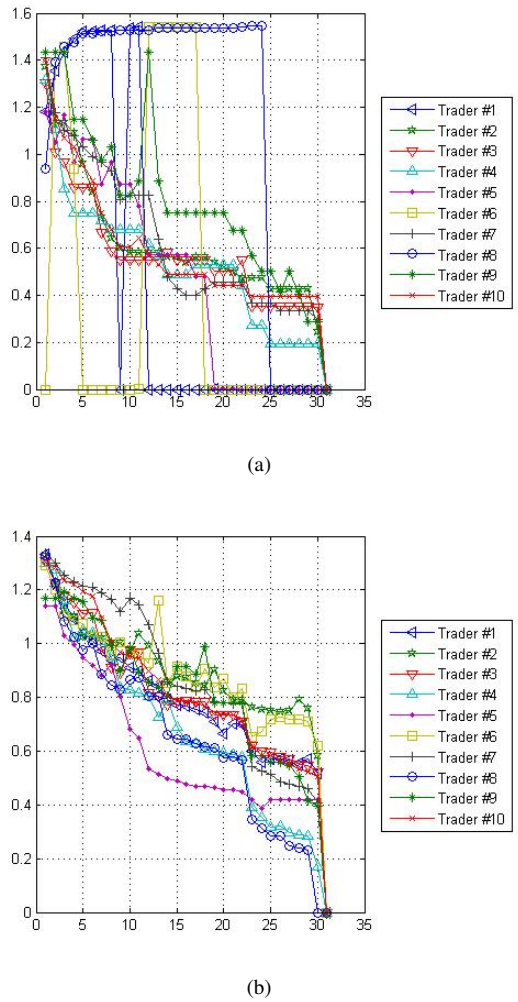
In this test, we take all the top 10 trading accounts and increase our observation window from 5 minutes to 5 hours. We observe from Figure (3) a) that at roughly 2 hour mark, which is our 12th sample, almost all the trading reward functions start to flatten out. It is therefore safe for us to assume our two-hour observation window will capture reward variation, and that the two-hour observation window is a good cut-off. With overlapping instances both in the morning and the afternoon we expect to capture U-shaped pattern for the market.

In general, the support vector machine classifier can handle both separable and non-separable cases. Furthermore through choosing a proper kernel function, the original feature space can be coerced into a transformed feature space to handle nonlinear boundary problems. Here we build an optimal SVM model for classification using a training dataset. In this tuning process the 10-fold cross-validation method will be used to pick the best penalty parameter gamma and cost parameter. We tried cost values in the ranges of [0.25-0.5], [0.5-1], [1-2], and [2-4] and gamma in the range of [0.0001, 0.001], [0.001, 0.01], [0.01, 0.1] and [0.1, 1]. We found that when gamma = 0.001 and cost=0.06, we have the lowest error rate of 10% for the training dataset. The total number support vectors used is 132.

We constructed 80 sample trajectories for each of the top 10 trading accounts. While there are 120 trading accounts consistently traded over the 4-week period, this study focuses on the top 10 trading accounts. We apply both the LNIRL ([24] and [8]), and GPIRL ([29]) to these 800 samples. And then we apply the SVM algorithm to the 10 traders using pair-wise classification. For each pair, we first train a SVM classifier (with Gaussian kernel) with 60 randomly selected samples, and test the classification on the remaining 20 samples. We repeat the sampling 100 times and then take the average classification accuracy. We list both LNIRL classification results in Table I, and GPIRL results in Table II. In these two tables, rows and columns represent anonymous trader IDs. On average, LNIRL gives a classification accuracy of 0.6039, while GPIRL achieves a classification accuracy of 0.9650. This result confirms our earlier assumption that GPIRL performs better when we have incomplete observations, and incorporate non-deterministic policies through Gaussian preference learning.

V. Conclusions

The primary focus of this paper is to use Inverse Reinforcement Learning to capture the key characteristics of the HFT strategies. From the results using both linear programming and Gaussian process methods for solving an IRL problem with a E-Mini S&P 500 futures market dataset, we attain a high identification accuracy ranging between 95% and 99% for the targeted trading strategy class using GPIRL. From our experiments on real market data, we find we can identify individual trading strategies with certain accuracy. It means that certain trading strategies that have unique characteristics can be identified with a satisfactory



(a)

(b)

Fig. 3. **Reward Trajectory Convergence:** (a). LIRL (b). GPIRL

accuracy. We also argue that reward space is better suited for identification of trading strategies than policy space.

We investigate and address the issues of modeling algorithmic trading strategies using IRL models such as, addressing non-deterministic nature of the observed policies in learning, constructing efficient MDP models to capture order book dynamics, achieving better identification accuracy in reward space, etc. The practical implication of this research is that we demonstrate that the market operators and regulators can use this behavior based learning approach to perform trader behavior based profiling, and consequently monitor the emergence of new HFTs and study their impact to the market.

References

[1] A. Irene. A Practical Guide to Algorithmic Strategies and Trading Systems - High Frequency Trading. *John Wiley & Sons, Inc,* 2010, 339.

[2] Hendershott, T., and Riordan, R. Algorithmic trading and the market for liquidity. *Journal of Financial and Quantitative Analysis,* forthcoming 2013.

[3] Brogaard, J. High frequency trading and market quality *SSRN Working Paper* 2012.

[4] Kirilenko, A., Kyle, A.S., Samadi, M., and Tuzun, T. The Flash Crash: The Impact of High Frequency Trading on an Electronic Market, *SSR Working Paper Series,* 2011.

[5] Benos, E., and Sagade, S. High-frequency trading behavior and its impact on market quality: evidence from the UK equity market *Bank of England Working Paper No. 469* 2012.

[6] Ramachandran, D., and Amir, E. Bayesian Inverse Reinforcement Learning. *In Proc. IJCAI,* 2007, 2586-2591.

[7] Ratliff, N., Ziebart, B., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A.K., and Srinivasa, S. Inverse Optimal Heuristic Control for Imitation Learning. *Proc. AISTATS,* 2009, 424–431.

[8] Yang, S.Y., Paddrik, M.E., Hayes, R.J., Todd, A., Kirilenko, A.A., Beling, P., and Scherer, W. Behavior Based Learning in Identifying High Frequency Trading Strategies. *Proceedings of IEEE Computational Intelligence in Financial Engineering and Economics, 2012,* 2012.

[9] Hayes, R., Paddrik, M. E., Todd, A., Yang, S. Y., Scherer, W., and Beling, P Agent Based Model of the E-MINI Future Market: Applied to Policy Decisions. *Proceedings of 2012 Winter Simulation Conference, Berlin, Germany* 2012.

[10] Paddrik, M. E., Hayes, R., Todd, A., Yang, S. Y., Scherer, W., and Beling, P. An Agent Based Model of the E-Mini S&P 500 and the Flash Crash. *Proceedings of IEEE Computational Intelligence in Financial Engineering and Economics, 2012,* 2012.

[11] Bertsekas, D.P. Neuro-Dynamic Programming. *Athena Scientific,* 2007.

[12] Daw, N.D., Niv, Y., and Dayan, P. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience,* 2005, **8**, 1704–1711.

[13] Dvijotham, K., and Todorov, E. Inverse Optimal Control with Linearly-solvable MDPs. *Proc. 27th International Conf. on Machine learning,* 2010 (ACM).

[14] R.E. A game-theoretic approach to apprenticeship learning. *In Advances in Neural Information Processing Systems,* 2008, 1449–1456.

[15] Syed, U., Bowling, M., and Schapire, R.E. Apprenticeship learning using linear programming. *Proc. 25th international Conf. on Machine learning,* 2008, 1032–1039.

[16] P. Ekman Intraday patterns in the S&P 500 index futures market. *The Journal of Futures Markets* v. 12 pp. 365-381, 1992.

[17] A.R. Admati and P. Pfleiderer A Theory of Intraday Patterns: Volume and Price Variability. *The Review of Financial Studies* 1 pp.3-40, 1988.

[18] Y.T. Lee and R.C Fox and Y. Liu Explaining Intraday Pattern of Trading Volume from the Order Flow Data. *Journal of Business Finance and Accounting* 28 pp. 306-686, 2001.

[19] T. Chordia and R. Roll and A. Subrahmanyam Market Liquidity and Trading Activity. *Journal of Finance* 56 pp. 501-530, 2001.

[20] J. Hasbrouchk, D. J. Seppi. Common factors in prices, order flows and liquidity. *Journal of Financial Economics*, 59, pp. 383-411, 2001.

[21] T. Hendershott et al.. Algorithmic Trading and Information. *NET Institute Working Paper*, No. 09-08, 2008.

[22] J. Brogaard. High Frequency Trading and its Impact on Market Quality. *Ph.D. thesis*, Northwestern University, 2010.

[23] T. Hendershott, C. M. Jones, and A. J. Menkveld. Does Algorithmic Trading Improve Liquidity?. *Journal of Finance*, V. 66 pp.1-33, 2011.

[24] A. Y. Ng and S. Russel. Algorithms for inverse reinforcement learning. In *Proc. ICML*, pp. 663-670, 2000.

[25] P. Abbeel, and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML '04 Proceedings of the twenty-first international conference on Machine learning*, 2004.

[26] Dimitri P. Bertsekas. Dynamic Programming and Optimal Control. *Athena Scientific*, (1995).

[27] R. S. Sutton, and A. G. Barto. Reinforcement Learning: An Introduction. *The MIT Press*, Cambridge, Massachusetts, 1998.

[28] B. D. Ziebart, A. Mass, J. A. Bagnell, and A. K. Dey. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the Twenty-Third AAAI on Artifical Intelligence*, 2008.

[29] Q. Qiao, and P. Beling. Inverse Reinforcement Learning with Gaussian Process. *Proceedings of 2011 American Control Conference*, San Francisco, CA, 2011.

TABLE I
PAIR-WISE TRADER CLASSIFICATION USING SVM BINARY CLASSIFICATION USING LNIRL

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	0.0000	0.5437	0.5187	0.4812	0.6375	0.4812	0.5312	0.5750	0.7750	0.5937
[2,]	0.5437	0.0000	0.5250	0.5125	0.7437	0.5562	0.4937	0.4250	0.7625	0.6812
[3,]	0.5187	0.5250	0.0000	0.4687	0.6875	0.5250	0.5187	0.5250	0.7312	0.6250
[4,]	0.4812	0.5125	0.4687	0.0000	0.6937	0.5000	0.4937	0.5062	0.6562	0.6625
[5,]	0.6375	0.7437	0.6875	0.6937	0.0000	0.6625	0.7375	0.6875	0.7750	0.5437
[6,]	0.4812	0.5562	0.5250	0.5000	0.6625	0.0000	0.5500	0.5500	0.6500	0.6375
[7,]	0.5312	0.4937	0.5187	0.4937	0.7375	0.5500	0.0000	0.4937	0.8000	0.6125
[8,]	0.5750	0.4250	0.5250	0.5062	0.6875	0.5500	0.4937	0.0000	0.6437	0.6562
[9,]	0.7750	0.7625	0.7312	0.6562	0.7750	0.6500	0.8000	0.6437	0.0000	0.7437
[10,]	0.5937	0.6812	0.6250	0.6625	0.5437	0.6375	0.6125	0.6562	0.7437	0.0000

TABLE II
PAIR-WISE TRADER CLASSIFICATION USING SVM BINARY CLASSIFICATION USING GPIRL

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	0.0000	1.0000	0.9875	0.9750	0.9500	0.9750	0.9625	1.0000	0.9750	1.0000
[2,]	1.0000	0.0000	0.9750	0.9375	0.9875	0.9750	0.9625	0.9625	0.9875	1.0000
[3,]	0.9875	0.9750	0.0000	0.9750	0.9625	0.9875	1.0000	0.9750	0.9750	0.9875
[4,]	0.9750	0.9375	0.9750	0.0000	0.9750	0.9500	0.9375	0.9875	0.9875	0.9750
[5,]	0.9500	0.9875	0.9625	0.9750	0.0000	1.0000	1.0000	0.9625	0.9875	1.0000
[6,]	0.9750	0.9750	0.9875	0.9500	1.0000	0.0000	0.9625	0.8750	0.9125	0.9750
[7,]	0.9625	0.9625	1.0000	0.9375	1.0000	0.9625	0.0000	0.8625	0.9625	0.9875
[8,]	1.0000	0.9625	0.9750	0.9875	0.9625	0.8750	0.8625	0.0000	0.8000	1.0000
[9,]	0.9750	0.9875	0.9750	0.9875	0.9875	0.9125	0.9625	0.8000	0.0000	0.9625
[10,]	1.0000	1.0000	0.9875	0.9750	1.0000	0.9750	0.9875	1.0000	0.9625	0.0000