Gaussian Process Based Algorithmic Trading Strategy Identification

Steve Y. Yang, Qifeng Qiao, Peter A. Beling, William T. Scherer, Andrei A. Kirilenko, and Jeremy Cusimano

Steve Y. Yang, Qifeng Qiao, Peter A. Beling, and William T. Scherer are with the Deparment of Systems and Information Engineering, University of Virginia, Charlottesville, VA 22904 USA (email:yy6a@viginia.edu; qq2r@virginia.edu; pb3a@virginia.edu;wts@virginia.edu). Andrei A. Kirilenko and Jeremy Cusimano are with the Commodity Futues Trading Commission (akirilenko@cftc.gov;jcusimano@cftc.gov)

Abstract

The advent of electronic financial markets and associated technologies has dramatically improved the trading functions that are available to market participants in terms of speed, capacity and sophistication. Advanced data feed and audit trail information from market operators also make the full observation of market participants' behavior possible. The primary objective of this study is to model algorithmic trading behavior using Bayesian inference under the framework of inverse reinforcement learning (IRL). We model trader behavior as a Gaussian process in the reward space. With incomplete observations of different market participants, we aim to recover the optimal policies and the corresponding reward functions to explain trader behaviors under different circumstances. We show that algorithmic trading behavior can be accurately identified using the Gaussian Process Inverse Reinforcement Learning (GPIRL) algorithm developed by Qiao and Beling (2011), and that this algorithm is superior to the linear features maximization approach. Real market data experiments using the GPIRL model consistently give more than 95% trader identification accuracy using a classification method based on support vector machines (SVM). We also show that there is a clear connection between the existing summary statistic-based trader classification proposed by Kirilenko etc. (2011) and our behavior-based classification. To address the potential change in trading behavior over time, we propose a score-based classification approach to address variations of algorithmic trading behavior under different market conditions. We further argue that because our behavior-based identification is a better reflection of traders' actions and value propositions under different market conditions than the summary statistic-based method, it is therefore more informative and robust than the summary statistic-based approach, and is well suited for discovering new behavior patterns of market participants.

I. Introduction

Financial markets have changed dramatically in recent years. 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. These technologies have dramatically improved the speed, capacity, and sophistication of the trading functions that are available to market participants.

High-quality trading markets promote capital raising and capital allocation by establishing prices for securities and by enabling investors to enter and exit their positions in securities whenever they wish to do so. The one important feature of all types of algorithmic trading strategies is discovering the underlying persistent tradable phenomena and generating 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 forecast-ed by micro-structure theories, and several hour-long market movements that surround recurring events and deviations from statistical relationship ([3]). Algorithmic traders then design their trading algorithms and systems with the aim of generating signals that result in consistent positive outcomes under different market circumstances. These market circumstances can be described in high frequency terms. Different strategies may target different frequencies, and the profitability of a trading strategy is often measured by a certain return metric. The most commonly used measure is the Sharpe ratio, a risk-adjusted return metric first proposed by Sharpe ([4]).

In this study, we model the trading behavior of different market participants by the solution to the inverse Markov decision process (MDP). We try to describe how traders are able to take actions in a highly uncertain environment to reach return goals on different horizons. This task can be solved using dynamic programming (DP) and reinforcement learning (RL) based on MDP. The model accounts for traders' preferences and expectations of uncertain state variables. In a general MDP modeling setting, we describe these variables in two spaces: the state space and the action space. From the trading decision perspective, we can parameterize learning agents using reward functions that depend on state and action. We consider the market dynamics in view of the learning agents' subjective beliefs. The agents perform DP/RL through a sense, trial and learn cycle. First, the agents gain state information from sensory input. Based on the current state, knowledge and goals, the agents find and choose the best action. Upon receiving new feedback, the agents learn to update their knowledge with a goal of maximizing their cumulative expected reward. In the discrete-valued state and action problem space, DP and RL methods use similar techniques involving policy iteration and value iteration algorithms ([5], and [6]) to solve MDP problems. Formalisms for solving forward problems of RL are often divided into model-based and model-free approaches ([7], and [6]).

As framed by Abbeel et al. ([8]) under the Inverse Reinforcement Learning (IRL) framework, the entire field of reinforcement learning 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, so the following difficulties may arise: 1) We have no experience to tackle the problem; 2) We have experience but can not interpret the reward function explicitly; 3) The problem we solve may be interacting with the adversarial decision makers who make all their effort to keep the reward function secret. Rather than accessing the true reward function, it is easier to observe the behavior of some other agents (teacher/expert) to determine how to solve the problem. Hence, we have motivation to learn from observations. Technical approaches to learning from observations generally fall into two broad categories [9]. 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 other IRL is concerned with how to learn the reward function that characterizes the agent's objectives and preferences in MDP ([10]).

IRL was first introduced in machine learning research by Ng and Russel ([10]), under the assumption that the reward $r = \omega^T \phi(s)$, where ω is a coefficient vector and $\phi(s) : s \to [0, 1]^k$ is a function mapping state s to a k dimensional vector. IRL was formulated 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. Other algorithms have been developed or integrated into apprenticeship learning based on this linear approximation of the reward function. 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 ([8] and [11]).

Other algorithms have also been developed under the IRL framework. A game-theoretic approach to apprenticeship learning using IRL was developed in the context of a two-player zero-sum game in which the apprentice chooses a policy and the environment chooses a reward function ([12]). Another algorithm for IRL is policy matching, in which the loss function penalizing deviations from the expert's policy that is minimized by tuning the parameters of the reward functions ([13]). The maximum entropy IRL is proposed in the context of modeling real-world navigation and driving behaviors ([14]). The algorithms for apprenticeship learning using IRL do not actually aim to recover the reward function but instead are only concerned with the optimal policy. Ramachandran and Amir consider IRL from a Bayesian perspective without assuming the linear approximation of the reward function ([15]). Their model interprets the observations from the expert as the evidence that is used to obtain a posterior distribution over reward using Markov Chain Monte Carlo simulation. Recent theoretical works on IRL such as the framework of the linear-solvable MDP ([16]), bootstrap learning ([17]) and feature construction ([18]), have also improved the learning performance. IRL has also been successfully applied to many real-world problems, such as the automatic control of helicopter flight ([19]) and the motion control of an animation system in computer graphics ([20]).

We apply a Gaussian process-based IRL (GPIRL) model proposed by Qiao et al. ([1]) to learn the trading behavior of a particular financial market. In this GPIRL, a Gaussian prior is assigned on the reward function and the reward function is treated as a Gaussian process. This approach is similar to that of Ramachandran and Eyal ([15]), who view the state-action samples from experts as the evidence that will be used to update a prior value in the reward function, under a Bayesian framework. The solution ([15]) depends on non-convex optimization using Markov Chain Monte Carlo simulation. Moreover, the ill-posed nature of the inverse learning problem also presents difficulties. Multiple reward functions may yield the same optimal policy, and there may be multiple observations at a single state given the true reward function. The GPIRL model aims to address the ill-posed nature of this problem by applying Bayesian inference and preference graphs. One of the main novel features of this approach is that it not only represents a probabilistically coherent view but is also computationally tractable.

The dynamic nature of financial markets makes it possible to postulate a priori a relationship between the market variables we observe and those we wish to predict. The main contributions of this study can be summarized as follows:

1) We model traders' reward functions using a Gaussian process, which offers the advantage that is relatively insensitive to the number of observations and that it performs better than other algorithms when we only have partial market observations on the trading strategies under study.
2) We apply preference graphs to address the non-deterministic nature of the observed trading behaviors, reducing the uncertainty and computational burden caused by the ill-posed nature of the inverse learning problem. We build new likelihood functions for preference graphs and prove the effectiveness of these formulations in experiments.
3) We also perform behavioral clustering of the observed trading strategies, and we make connections between the existing summary statistic-based trader classification approach ([2]) and our behavior-based classification approach. We propose a quantitative behavioral approach to categorizing algorithmic trading strategies using weighted scores over time.

The remainder of this paper is organized as follows: First we discuss related work and some preliminary considerations in Section II. In Section III, we discuss IRL formulations and provide a Bayesian probabilistic model to infer the reward function using Gaussian processes. We apply the GPIRL algorithm to the most active financial market E-Mini S&P 500 Futures market as an experiment in Section IV. We show that the GPIRL algorithm can accurately capture algorithmic trading behavior based on observations of the high frequency data. We also compare our behavior-based classification results with the results from Kirilenko et al. ([2]), and show that our behavioral approach represents a consistent improvement. Finally in Section V we provide concluding remarks about the GPIRL and its applications.

II. Background and Related Work

Our solution to the inference from observations is based on the general Inverse Reinforcement Learning framework. In this section, we first introduce notations that will be used throughout this paper, and we then discuss some essential facts from the theory of Markov Decision Processes that are used here.

A. Inverse MDP Problem

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 a Markov Decision Process. 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 $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, r)$, 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 $r(s_n, a_m, s)$ 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.

In MDP, an 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 $(s^0, a^0, s^1, a^1, \cdots s^t, a^t, \cdots)$, [1] where s^{t+1} is connected to (s^t, a^t) by $\mathbf{P}_{a^t}(s^t, s^{t+1})$. The policy, which leads the agent to reach its goal, is called *the proper policy*.

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 $\{s^0, s^1, ...\}$ given the policy $\pi = \{\mu^0, \mu^1, \cdots\}$, 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 $V^\pi(s)$. Similarly, there is another function called *Q-functions* (*Q-factors*) 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.

In the infinite-horizon case, the stationary Markovian structure of the problem implies that the only variable that affects the agent's decision rule and the corresponding value function should be time invariant. We then have the essential theory of MDPs ([22]) as follows:

Theorem 1 (Bellman Equations): Given a stationary policy π, $\forall n \in \mathcal{N}, m \in \mathcal{M}$, $V^\pi(s_n)$ and $Q^\pi(s_n, a_m)$ satisfy

$$\begin{aligned} V^\pi(s_n) &= r(s_n, \pi(s_n)) + \gamma \sum_{n' \in \mathcal{N}} \mathbf{P}_{\pi(s_n)}(s_n, s_{n'}) V^\pi(s_{n'}), \\ Q^\pi(s_n, a_m) &= r(s_n, a_m) + \gamma \sum_{n' \in \mathcal{N}} \mathbf{P}_{a_m}(s_n, s_{n'}) V^\pi(s_{n'}). \end{aligned}$$

Theorem 2 (Bellman Optimality): π is optimal if and only if, $\forall n \in \mathcal{N}$, $\pi(s_n) \in \arg\max_{a \in \mathcal{A}} Q^\pi(s, a)$.

[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 $\mathcal{A}$).

Based on the above definitions of MDP, we further introduce the *Inverse Markov Decision Process (IMDP).*

Definition An IMDP model, denoted as $M_I = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{O})$, contains MDP variables such as, state set $\mathcal{S}$, action set $\mathcal{A}$, state transition probability set $\mathcal{P}$ and the discount factor γ. The variable $\mathcal{O}$ is a set of observations sampled from the decision-making process.

The set $\mathcal{O}$ can be viewed as a subset of the Cartesian product of $\hat{\mathcal{S}}$ and $\hat{\mathcal{A}}$, where $\hat{\mathcal{S}} \subset \mathcal{S}$ and $\hat{\mathcal{A}} \subset \mathcal{A}$. So $\forall s \in \hat{\mathcal{S}}$, and there is at least one action $a \in \hat{\mathcal{A}}$ providing $(s, a) \in \mathcal{O}$. We treat every $(s, a) \in \mathcal{O}$ as optimal in the expert's decision making process. The goal of IRL is to learn the reward function of the MDP model that generates $\mathcal{O}$.

III. Gaussian Process for the Generalized IRL Problem

We now turn to an IRL problem that addresses observations from a decision making process in which the reward function has been contaminated by Gaussian noise. In particular, we assume that the reward vector can be modeled as $r + \mathcal{N}(0, \sigma^2)$, where $\mathcal{N}(0, \sigma^2)$ is Gaussian noise. In the financial trading problem setting, we may observe certain trading behavior over a period of time, but we may not observe the complete polices behind a particular trading strategy. As discussed earlier, different trading strategies tend to look at different time horizons. 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 Belling ([1]) 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

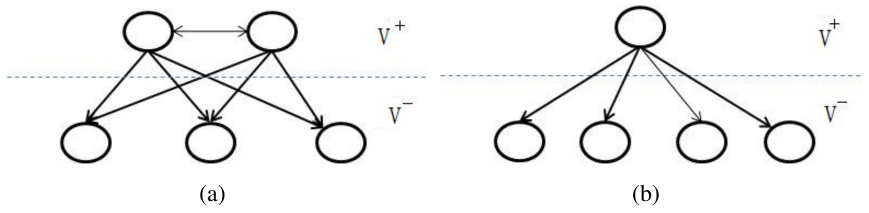


Fig. 1. **Examples Preference Graphs** (a) An example of observing two actions at a state. (b) An example of a unique observation at a state.

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.

A. 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 give Lemma (3) as a rule to build the preference graph, and then we show how to draw a preference graph at state s_n.

Lemma 3: At state s_n, if action $\hat{a}$ is observed, we have the following preference relations: $\hat{a} \succ_{s_n} \check{a}, \forall \check{a} \in \mathcal{A} \setminus \{\hat{a}\}$.

It is therefore straightforward to show the following according to Bellman optimality. 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 on 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 $\hat{a} \succeq_{s_n} \tilde{a}$.

At this point, we have a simple representation of the action preference graph that is constructed by a two-layer directed graph. We may have either multiple actions at s_n as in Figure (1) (a) or a unique action at s_n as in Figure (1) (b). In this two-layer directed graph, the top layer $\mathcal{V}_n^+$ is a set of nodes representing the observed actions and the bottom layer $\mathcal{V}_n^-$ contains the nodes denoting the other actions. The edge in the edge set $\mathcal{E}_n$ can be represented by a formulation of its beginning node u and ending node v. We write the k-th edge as $(u \rightarrow v)_k$ if $u \in \mathcal{V}_n^+, v \in \mathcal{V}_n^-$, or the l-th edge $(u \leftrightarrow v)_l$ if $u \in \mathcal{V}_n^-, v \in \mathcal{V}_n^-$. Recall the mapping between $\mathcal{V}_n$ and $\mathcal{A}$, the representation $u \rightarrow v$ indicates that action $\varphi(u)$ is preferred over $\varphi(v)$. Similarly, $u \leftrightarrow v$ means that action $\varphi(u)$ is equivalent to $\varphi(v)$.

In the context of financial trading decision process, we may observe multiple actions from one particular trader under certain market conditions. That is to say that the observation data $\mathcal{O}$ may represent multiple decision trajectories generated by non-deterministic policies. To address IRL problems in those cases, Qiao and Belling ([1]) propose to process $\mathcal{O}$ into the form of pairs of state and preference graphs similar to the representation shown in Figure (2), and then we apply Bayesian inference using the new formulation.

To apply Bayesian inference to the reward function, we need to show the equivalence of evidence for the inference of the reward function between decision trajectories and the independent pairs of state and preference graphs (See Proposition 4). Thus, we make the following proposition (see a proof in the Appendix Gaussian Processes).

Proposition 4: The observation dataset $\mathcal{O}_1$ is given as a set of decision trajectories. Assume independence among the observed decision trajectories. Observation of a policy at a state can be specified by an action preference graph. Let $\mathcal{O}_2$ be the set of independent pairs of state

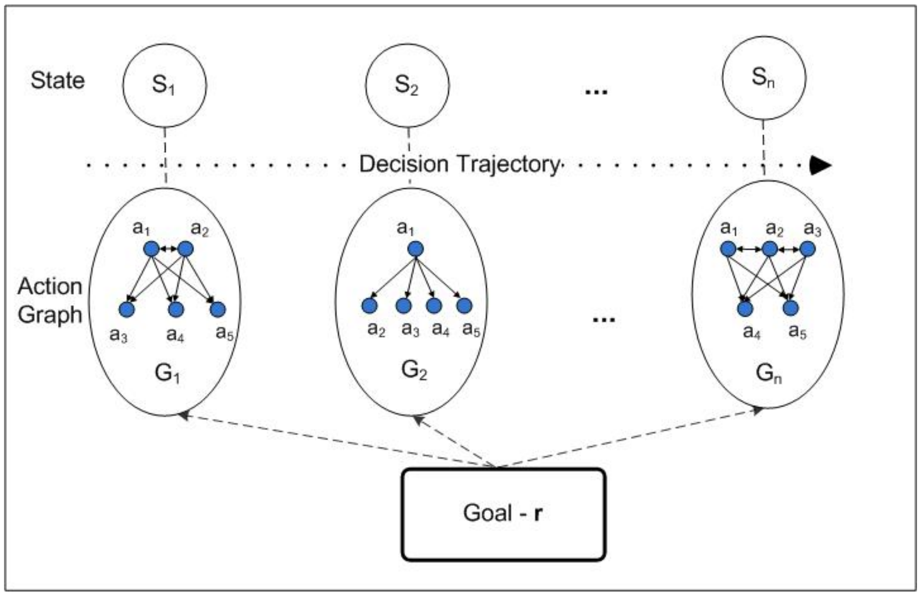


Fig. 2. Proposed observation structure for MDP.

and action preference graph, which is written as $\mathcal{O}_2 = \{(s_n, G_n)\}_{n=1}^N$. The inference of the reward function drawn from $\mathcal{O}_1$ and $\mathcal{O}_2$ is identical. There is a constant factor C that makes the likelihood function $p(\mathcal{O}_1|r) = Cp(\mathcal{O}_2|r)$.

Based on Proposition 4, we can represent $\mathcal{O}$ as shown in Figure (2). 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.

B. Bayesian Inference of a Gaussian Reward Process

We adopt a variation of the likelihood function proposed by Chu and Ghahramani in [23] to capture the strict and equivalent preference relations. The likelihood function for the reward without noise is as follows:

$$p_{\text{ideal}}(\hat{a} \succ_{s_n} \check{a} | \mathbf{r}_{\hat{a}}(s_n), \mathbf{r}_{\check{a}}(s_n)) = \begin{cases} 1 & \text{if } Q(s_n, \hat{a}, \mathbf{r}) > Q(s_n, \check{a}, \mathbf{r}) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

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

As we stated earlier, if we model the reward functions as being contaminated with Gaussian noise that has a mean of zero and an unknown variance σ^2, we can then define the likelihood function for both the k-th strict preference relation and the l-th equivalent preference relation. Finally, we can formulate the following proposition:

Proposition 5: The likelihood function, given the evidence of the observed data $\mathcal{O}$ in the form of pairs of state and action preference graphs, is calculated by

$$p(\mathcal{G}|\mathcal{S},\mathbf{r}) = \prod_{n=1}^{N} p(G_n|s_n,\mathbf{r}) = \prod_{n=1}^{N}\prod_{k=1}^{n_n} \Phi(f(\mathbf{r},s_n,k))e^{\sum_{n=1}^{N}\sum_{l=1}^{m_n} -\sigma^2 f^2(\mathbf{r},s_n,l)} \tag{3}$$

In conclusion, the probabilistic IRL model is controlled by the kernel parameters κ_{a_m} and σ_{a_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 $\boldsymbol{\theta} = (\kappa_{a_m}, \sigma_{a_m}, \sigma)$. 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.

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:

$$p(\mathbf{r}|\mathcal{S},\mathcal{G},\boldsymbol{\theta}) = \frac{p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta},\mathbf{r})p(\mathbf{r}|\mathcal{S},\boldsymbol{\theta})}{p(\mathcal{G}|\mathcal{S},\boldsymbol{\theta})}. \tag{4}$$

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.4 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 written as

$$U(\mathbf{r}) \triangleq \frac{1}{2}\sum_{m=1}^{M} \mathbf{r}_{a_m}^T \mathbf{K}_{a_m}^{-1}\mathbf{r}_{a_m} - \sum_{n=1}^{N}\sum_{k=1}^{n_n} \ln \Phi(\sum_{m=1}^{M} \rho_{a_m}^{nk}\mathbf{r}_{a_m})$$
$$+\sum_{n=1}^{N}\sum_{l=1}^{m_n}\frac{1}{2}(\sum_{m=1}^{M}\rho_{a_m}^{nl}\mathbf{r}_{a_m})^2 \tag{5}$$

where given $(\hat{a} \sim_{s_n} \hat{a}')_l$, let $\boldsymbol{\Delta}_l \triangleq \gamma(\mathbf{P}_{\hat{a}}(s_n,:) - \mathbf{P}_{\hat{a}'}(s_n,:))(\mathbf{I}_N - \gamma \mathbf{P}_{\pi(s_n)}(s_n,:))^{-1}$, then we have

$$\rho_{a_m}^{nl} = \mathbf{e}_n[\mathbf{1}(a_m = \hat{a}) - \mathbf{1}(a_m = \hat{a}')] + \boldsymbol{\Delta}_l \hat{\mathbf{I}}_{a_m}$$

where $\mathbf{e}_n$ is a $1 \times N$ vector whose entry $\mathbf{e}_n(n) = 1$, and $\mathbf{e}_n(j) = 0, \forall j \in \mathcal{N} \setminus \{n\}$. The notation $\mathbf{1}(.)$ is an indicator function. Similarly, $\rho_{a_m}^{nk}$ denotes the coefficient vector for the k-th strict preference relation $\hat{a} \succ_{s_n} \check{a}$.

Qiao and Beling ([1]) present a proof that Proposition (5) is a convex optimization problem (see Appendix Convex Optimization Problem). At the minimum of $U(\mathbf{r})$ we have

$$\frac{\partial U}{\partial \mathbf{r}_{a_m}} = 0 \Rightarrow \hat{\mathbf{r}}_{a_m} = K_{a_m}(\nabla \log P(\mathcal{G}|\mathcal{S}, \hat{\mathbf{r}}, \boldsymbol{\theta})) \tag{6}$$

where $\hat{\mathbf{r}} = (\hat{\mathbf{r}}_1, \cdots, \hat{\mathbf{r}}_{a_m}, \cdots, \hat{\mathbf{r}}_m)$. In Eq.6, we can use Newton's method to find the maximum of U with the iteration,

$$\mathbf{r}_{a_m}^{\text{new}} = \mathbf{r}_{a_m} - (\frac{\partial^2 U}{\partial \mathbf{r}_{a_m} \partial \mathbf{r}_{a_m}})^{-1} \frac{\partial U}{\partial \mathbf{r}_{a_m}}$$

IV. Experiment with the E-Mini S&P 500 Equity Index Futures Market

A. *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. For example, the S&P 500 Index futures contract is trading at \$1,400.00, then the value of one contract is \$70,000. The advantages to trading E-mini S&P 500 contracts include liquidity, greater affordability for individual investors and around-the-clock trading.

Trading takes place 24 hours a day with the exception of a short technical maintenance shutdown period from 4:30 p.m. to 5:00 p.m. The E-Mini S&P 500 expiration months are March, June, September, and December. On any given day, the contract with the nearest expiration date is called the front-month contract. The E-Mini S&P 500 is cash-settled against the value of the underlying index and the last trading day is the third Friday of the contract expiration month. The initial margin for speculators and hedgers are \$5,625 and \$4,500, respectively. Maintenance margins for both speculators and hedgers are \$4,500. There is no limit on how many contracts can be outstanding at any given time.

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.

Under the classification rule documented by Kirilenko et al. ([2]), we can group individual trading accounts into six categories based on their trading activities. These categories include: High Frequency Traders (high volume and low inventory), Intermediaries (low inventory), Fundamental Buyers (consistent intraday net buyers), Fundamental Sellers (consistent intraday net sellers), Opportunistic Traders (all other traders not classified) and Small Traders (low volume). For Fundamental Traders, we calculate their end of day net position, and if it is more than 15% of their total trading volume on that day, we categorize them either as Fundamental Buyers or Fundamental Sellers depending on their trading directions. We can also easily identify Small Traders as those accounts with a trading volume of 9 or less. We also apply the criteria ([2]) for Intermediaries, Opportunistic Traders and High Frequency Traders, and obtain consistent results for Intermediaries for the one-month data. There are two steps involved in this process. First, we ensure that the account's net holdings fluctuate within 1.5% of its end of day level, and second, we ensure the account's end of day net position is no more than 5% of its daily trading volume. Then if we define HFTs as a subset of Intermediaries (top 7% in daily trading volume), we find that there is a significant amount of overlap between HFTs and Opportunistic Traders. The problem is that the first criterion is not well defined, as the fluctuation of net holdings is vaguely defined. Net holdings could be measured in different ways. After talking to the authors of [2], we decided to use the standard deviation of an account's net position measured on the event clock as a measure of an account's holding fluctuation. With this definition, it turns out that a 1.5% fluctuation is too stringent for HFTs. This excessive stringency is evidenced by the fact that many high trading volume accounts are classified as Opportunistic Traders, while in reality their end of day positions are still very low compared with other Opportunistic Traders. Therefore, we decided to relax the first criterion requiring that the standard deviation of the account's net holdings throughout the day is less than its end of day holding level. We find that the newly adjusted criteria classify most high volume trading accounts as HFTs (without this adjustment, almost all the top trading accounts are classified as Opportunistic Traders). Table I summarizes the results after applying the new classification rule demonstrating that the modified classification criteria identified more HFTs. On average, there are 38 HTF accounts, 118 Intermediary accounts, 2,658 Opportunistic accounts, 906 Fundamental Buyer accounts, 775 Fundamental Seller accounts, and 5,127 Small Trader accounts. Over the 4-week period, only 36% of the 120 accounts that consistently classified as the same type of traders. If we rank these accounts by their daily trading volume, we find that only 40% of the top 10 accounts are consistently classified as the same trader types. The variation occurs among the HFTs, Intermediaries, and Opportunistic Traders. Next, we will show that our IRL based behavior identification approach is far superior to the statistic-based approach. We will use the top 10 trading accounts as examples to demonstrate improvement of behavior-based trading strategy identification achieved using the Gaussian Preference IRL model.

B. An MDP Model for Market Dynamics

This study uses a month of order book audit trial data from the E-Mini S&P 500 Futures contract market. 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.).

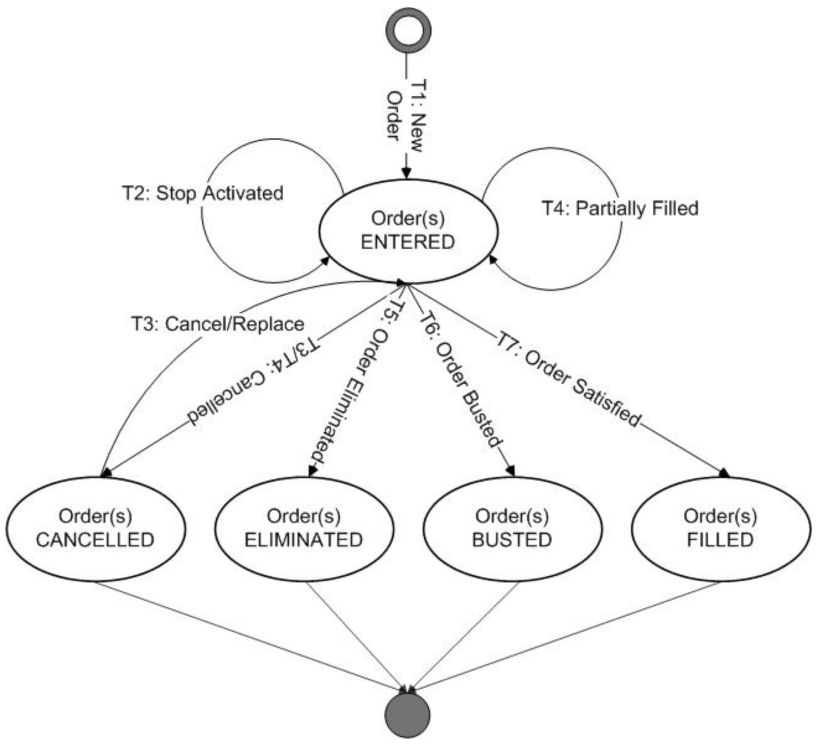
Figure 3 shows the entire life-cycle of an order initiated by a client. The order book audit trial data contains these messages, and the entire order history (i.e. order creation, order modifications,

TABLE I
THE E-MINI S&P 500 FUTURES MARKET DATA SUMMARY

Date	HFTs	Market Makers	Opportunistic Traders	Fundamental Buyers	Fundamental Sellers	Total Number of Accounts	Total V
10/04/2012	39	193	2,833	940	818	10,425	3,26
10/05/2012	38	162	2,598	1191	1055	11,495	3,87
10/06/2012	38	167	2,401	895	712	9,065	2,85
10/07/2012	39	196	2,726	919	747	9,841	3,42
10/08/2012	32	162	2,511	847	812	9,210	3,09
10/11/2012	21	118	1,428	636	573	6,230	1,76
10/12/2012	38	186	2,687	896	745	9,771	3,23
10/13/2012	38	187	2,582	1020	840	10,297	3,69
10/14/2012	30	198	3,001	1070	795	10,591	4,05
10/15/2012	46	210	3,109	890	773	9,918	4,43
10/18/2012	37	173	2,126	869	724	8,735	2,45
10/19/2012	52	216	3,651	1030	974	11,600	5,27
10/20/2012	39	176	2,949	951	877	10,745	3,95
10/21/2012	43	240	3,370	952	771	10,980	4,23
10/22/2012	32	143	1,837	676	629	7,370	2,02
10/25/2012	38	181	2,533	888	684	9,228	3,07
10/26/2012	37	175	2,726	816	709	9,568	3,00
10/27/2012	45	186	2,973	919	820	10,472	3,85
10/28/2012	39	185	2,873	914	705	9,777	3,48
10/29/2012	37	160	2,247	794	744	8,369	3,01

fills, cancellation, etc.) can be retrieved and analyzed. The first step in our study is to reconstruct the limit order book using the audit trail messages. The order book will then contain bid/ask prices, market depth, liquidity, etc. During this process, we process billions of messages for each trading date, and build price queues using the price and time priority rule.

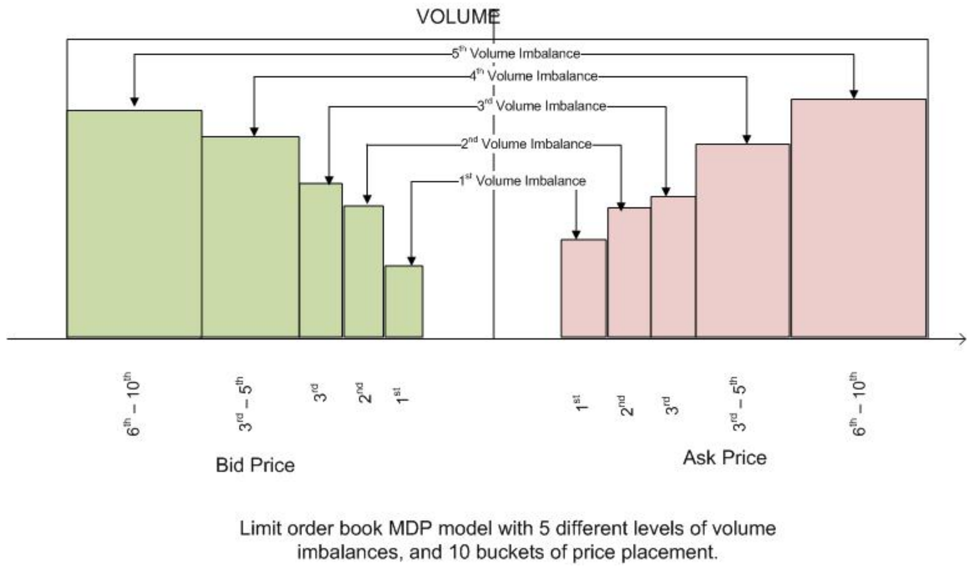
Once we have the order book at any given event tick, we take the market depth at five different levels as our base variables and then discretize these variables and generate our MDP model state space. This study extends the MDP model documented by ([24]) to obtain five variables, i.e., order volume imbalance 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



(a) Order Lifecycle on CME Globex

Fig. 3. **CME Globex Order Lifecycle.** T1: Trader submits a new order; T2: The state of an order is changed, if a stop is activated; T3: A trader may choose to cancel an order, and the state of an order can be modified multiple times; T4: When an order is partially filled, the quantity remaining decreases; T5: Order elimination is similar to order cancellation except it is initiated by the trading engine; T7: An order may be filled completely; T6: Trades can be busted after the fact by the exchanges.

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 4 (b)). Then we discretize the values of the five variables into three levels defined as high (above $\mu + 1.96\sigma$), neutral ($\mu \pm 1.96$), and low (below $\mu - 1.96\sigma$). As argued by ([24]), 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 a previous work ([24]), 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 ([2], [25] and [26]). HFTs and Market Makers tend to turn over their inventory



(a)

Fig. 4. **Order Book MDP Model:** This graph shows the state variables used in the MDP model.

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 30 actions. The price markers used to define the price ranges are illustrated in Figure (4).

Once the state and action space are defined, we can create an action preference graph based on the statistics of actions under different states. Here we use two examples to demonstrate how the action preference graphs have been constructed based on the MDP model and observed actions. Table II shows two example states with multiple observed actions. We then sort the frequency in

TABLE II
ACTION PREFERENCE GRAPH EXAMPLES

State	Action	Frequency Observed	State	Action	Frequency Observed
14	1	0.23	158	1	0.30
14	2	0.14	158	3	0.07
14	7	0.06	158	7	0.11
14	11	0.26	158	11	0.30
14	12	0.09	158	17	0.07
14	16	0.17	158	18	0.07
14	26	0.06	158	20	0.07

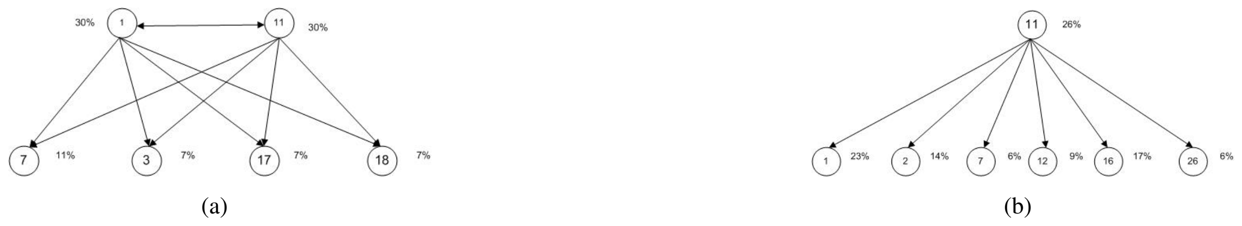


Fig. 5. **Action Preference Graph Examples:** (a). This graph shows an example action preference graph at state 158; (b). This graph shows an example action preference graph at state 14.

descending order and construct a two-layer graph: the top layer has the most frequently observed actions and the bottom layer holds all the other actions. Based on this preference observation, we can construct two preference graphs as shown in Figure (5). The state transition matrix can be constructed for the entire market for the observation period. In our MDP model, we have a 243x243 matrix for every single action.

C. Trader Behavior Identification

Yang et al. ([24]) 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.

This paper tries to address two important issues during the modeling process to solve a realistic market strategy learning problem using real market data. The first issue is that in reality, we often do not have a complete set of observations of a trader's policies. As the market presents itself as a random process in terms of both prices and volume, it is unlikely that we will be able to capture all possible states during our observation window. In contrast, the study performed by Yang et al. ([24]) assumes complete observation of a trader's decision policies for the simulated trading strategies. In other words, the policies simulated by a distribution can be completely captured when the simulation is run long enough. The convergence of these simulated policies and the testing results are consistent with their assumptions. However, when we use real market data to learn about trading strategies, it is necessary to address the incomplete observation problem. The second issue is to the assumption of deterministic policy vs. non-deterministic policy. Yang et al. ([24]) make a deterministic policy assumption. Under the linear feature optimization framework, non-deterministic policies can be represented by a single maximum deterministic policy (see proof in the Appendix Deterministic Policy vs. Randomized Policy). In this study, we relax the deterministic policy assumption and allow non-deterministic policies under a Gaussian process framework. As we argue earlier, Gaussian process learning allows us to infer policies even when we have a very limited set of observations. At the same time, we incorporate Gaussian preference learning into our inference process. This approach helps us to incorporate less frequently observed policies into our reward learning process. Together, the proposed GPIRL approach results in a model that relies less on observations and makes fewer assumptions on the polices we are to learn.

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

This section uses the 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 the SVM classification method is its flexibility that allows us to explore feature separation in different high dimensional spaces using kernel functions. We aim to compare the performance of the two behavior learning algorithms LNIRL and GPIRL, and to show that GPIRL perform better in real world trading strategy identification.

We constructed 80 sample trajectories for each of the top 10 trading accounts. While there are 121 trading accounts consistently traded over the 4-week period, this study focuses on the top

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

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[
[1,]	0.0000	0.5437	0.5187	0.4812	0.6375	0.4812	0.5312	0.5750	0.77
[2,]	0.5437	0.0000	0.5250	0.5125	0.7437	0.5562	0.4937	0.4250	0.76
[3,]	0.5187	0.5250	0.0000	0.4687	0.6875	0.5250	0.5187	0.5250	0.73
[4,]	0.4812	0.5125	0.4687	0.0000	0.6937	0.5000	0.4937	0.5062	0.65
[5,]	0.6375	0.7437	0.6875	0.6937	0.0000	0.6625	0.7375	0.6875	0.77
[6,]	0.4812	0.5562	0.5250	0.5000	0.6625	0.0000	0.5500	0.5500	0.65
[7,]	0.5312	0.4937	0.5187	0.4937	0.7375	0.5500	0.0000	0.4937	0.80
[8,]	0.5750	0.4250	0.5250	0.5062	0.6875	0.5500	0.4937	0.0000	0.64
[9,]	0.7750	0.7625	0.7312	0.6562	0.7750	0.6500	0.8000	0.6437	0.00
[10,]	0.5937	0.6812	0.6250	0.6625	0.5437	0.6375	0.6125	0.6562	0.74

Notes: The columns and rows of this table represent anonymous traders.

10 trading accounts. We apply both the LNIRL ([10] and [24]), and GPIRL ([1]) 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 III, and GPIRL results in Table IV. 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 nondeterministic policies through Gaussian preference learning.

E. Trading Strategy Clustering and Comparison with the Summary Statistic-Based Approach

In the previous section, we discovered that using reward functions we can reliably identify a particular trading strategy over a period of time with a relatively high accuracy. In this section, we want to study the similarity of reward characterization among the different trading strategies. This problem can be characterized as an unstructured learning problem - clustering. We have the characterization of rewards over the state space and action space, and we aim to group trading strategies based on their similarity over the Cartesian product of the state and action spaces. We also attempt to establish connections between these trading strategy classification definitions

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

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[
[1,]	0.0000	1.0000	0.9875	0.9750	0.9500	0.9750	0.9625	1.0000	0.97
[2,]	1.0000	0.0000	0.9750	0.9375	0.9875	0.9750	0.9625	0.9625	0.98
[3,]	0.9875	0.9750	0.0000	0.9750	0.9625	0.9875	1.0000	0.9750	0.97
[4,]	0.9750	0.9375	0.9750	0.0000	0.9750	0.9500	0.9375	0.9875	0.98
[5,]	0.9500	0.9875	0.9625	0.9750	0.0000	1.0000	1.0000	0.9625	0.98
[6,]	0.9750	0.9750	0.9875	0.9500	1.0000	0.0000	0.9625	0.8750	0.91
[7,]	0.9625	0.9625	1.0000	0.9375	1.0000	0.9625	0.0000	0.8625	0.96
[8,]	1.0000	0.9625	0.9750	0.9875	0.9625	0.8750	0.8625	0.0000	0.80
[9,]	0.9750	0.9875	0.9750	0.9875	0.9875	0.9125	0.9625	0.8000	0.00
[10,]	1.0000	1.0000	0.9875	0.9750	1.0000	0.9750	0.9875	1.0000	0.96

Notes: The columns and rows of this table represent anonymous traders.

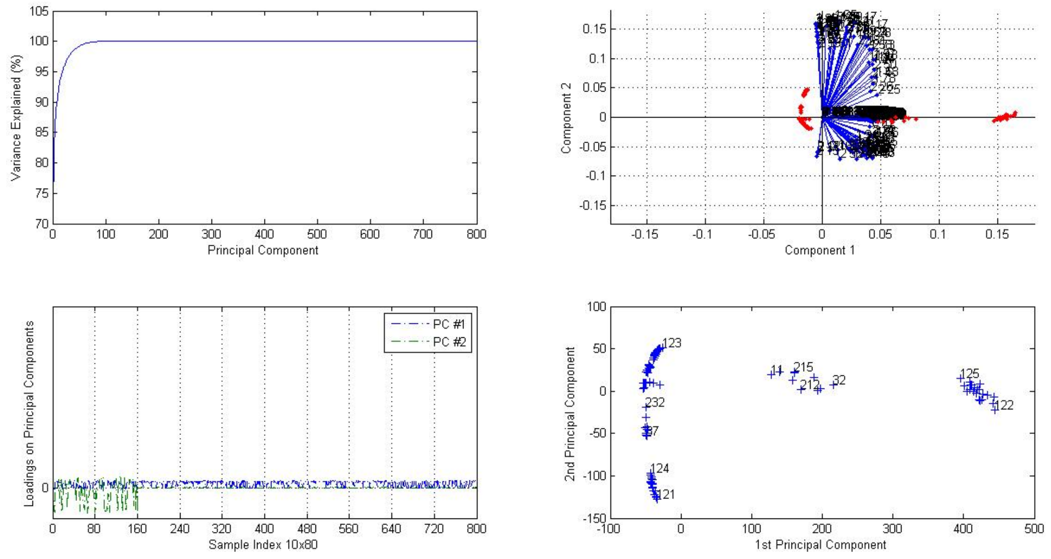
established by Kirilenko et al. ([2]) and our behavior-based trading strategy clustering.

The first problem we have to address is the dimensionality of the feature space. We essentially have a reward structure over a large set of feature sets. This feature set is a product of the state space and the action space in our computational model. Fortunately, under the LNIRL algorithm, we reduce the feature space to only the state space because in this linear feature expectation optimization problem we only consider reward at a particular state. Under the deterministic policy assumption, we assume that the value function converges at a particular state. In other words, the reward function is not a function of actions. In this case, we have 243 features that must be considered during the clustering. However, under the GPIRL framework we do not assume deterministic policy, and we treat reward as a function of both states and actions. Therefore we have 243x30 features for the latter approach. We also observe that the reward matrix is relatively sparse where there are zero values at many states. To consider computational tractability and efficiency, we first examine the data structure through Principal Component Analysis.

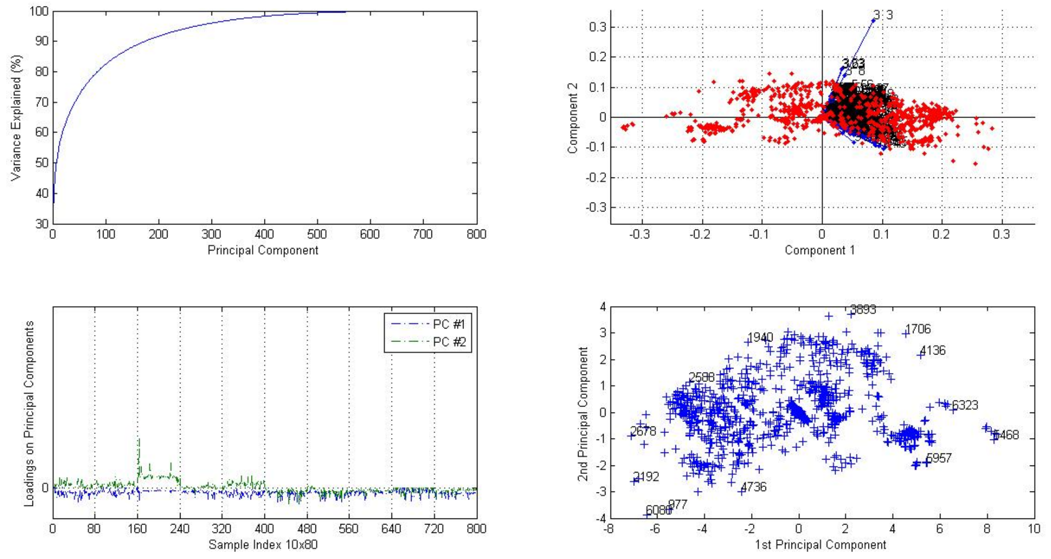
In the LNIRL case, the first two Principal Components (PCs) explain 79.78% of the data variation, and from the upper left plot in Figure (6) (a) we see that the first 200 PCs provide nearly 100% explanatory power. In the GPIRL case, the first two PCs only explain 38.98% of the data variation. Looking at the upper left plot in Figure (6) (b), we see that more PCs are needed to have better represent the data. To balance the accuracy and computational efficiency, we choose the first 200 PCs for the LNIRL and the first 400 PCs for the GPIRL case. This reduction leads to significant gain in computational efficiency and sacrifices less than 2% data variation (lower left figure in both Figure (6) (a) and (b)). From the upper right plots in both the LNIRL and the GPIRL spaces, we see that the first two PCs give a good representation along the first PC and that in the LNIRL case, the feature vector representation is evenly distributed between the first two PCs. The LNIRL space includes distinctly separated observations. On the other hand, the GPIRL space contains concentrations of observations, but unclear boundaries. In both cases, we would expect the PC dimension reduction technique to achieve relatively good representation of the data variation.

Now we apply unsupervised learning method to group the trading behavior observed on a selected group of trading accounts over the observation period. We select 10 trading accounts with the highest average daily trading volume over a period of 4 weeks (20 days) in our first experiment. We define an observation instance as a continuous period covering two hours over which we take all the activities of a particular trader including placing new orders, modifying and canceling existing orders, and placing market orders. For each trader, we collect four observation instances on each trading date: two observation instances during the morning trading and two observation instances during the afternoon trading. The two observation periods in the morning and in the afternoon have an hour overlap time, but the observations in the morning and the afternoon do not overlap. This observation distribution is selected based on the general theory of intraday U-shaped patterns in volume - namely, that trading is heavy at the beginning and the end of the trading day and relatively light in the middle of of the day ([31], [32], [33], and [34]). We also examined traders' actions throughout the entire trading day. We found that the two-hour observation time is a good cut-off, and with the overlapping instances in both the morning and the afternoon we expect to capture the U-shaped pattern of the market.

We then perform hierarchical clustering and generate a heat map and dendrogram of the observations in both the LNIRL reward space and the GPIRL reward space. The simplest form of matrix clustering clusters the rows and columns of a dataset using Euclidean distance metric and average linkage. For both Figure (7) (a) and (b), the left dendrogram shows the clustering of the observations (rows), and the top dendrogram shows the clustering of the PCs (columns). It is evident that there is a clear division of the observations (rows) in both cases. Upon closer

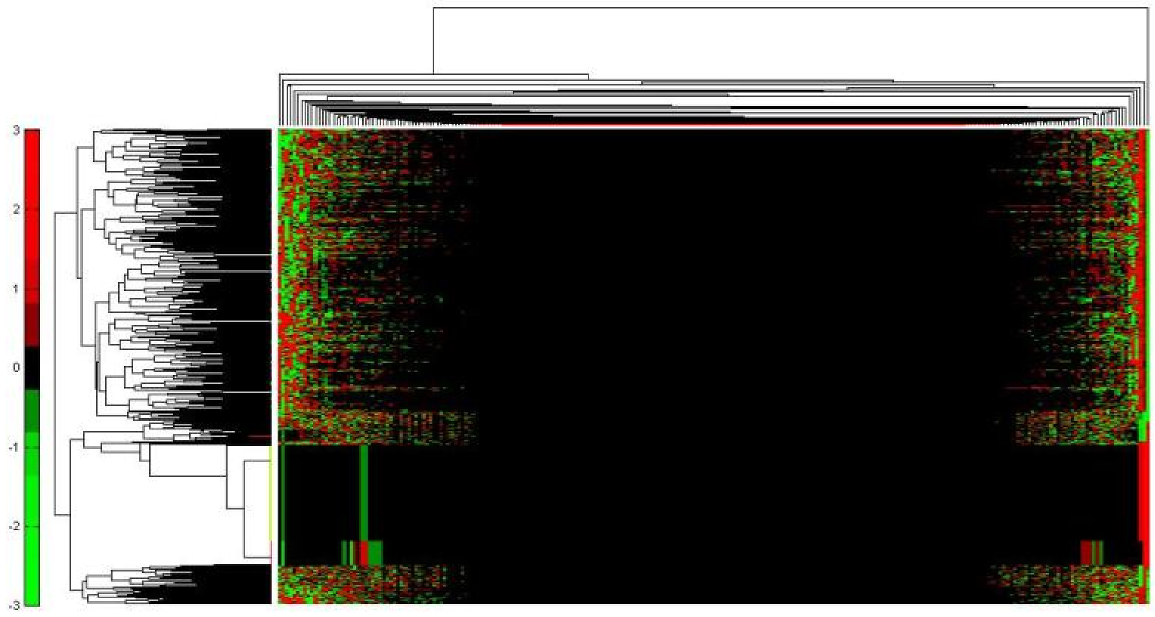


(a) The upper left figure shows cumulative percentage of the data variance explained by the PCs. The lower left figure plots the loadings of all the observations onto the first two PCs; The upper right figure shows the projection of the observation and feature vector onto the first two PCs. The lower right shows the projection of the observation onto the first two PCs with boundary point markers.

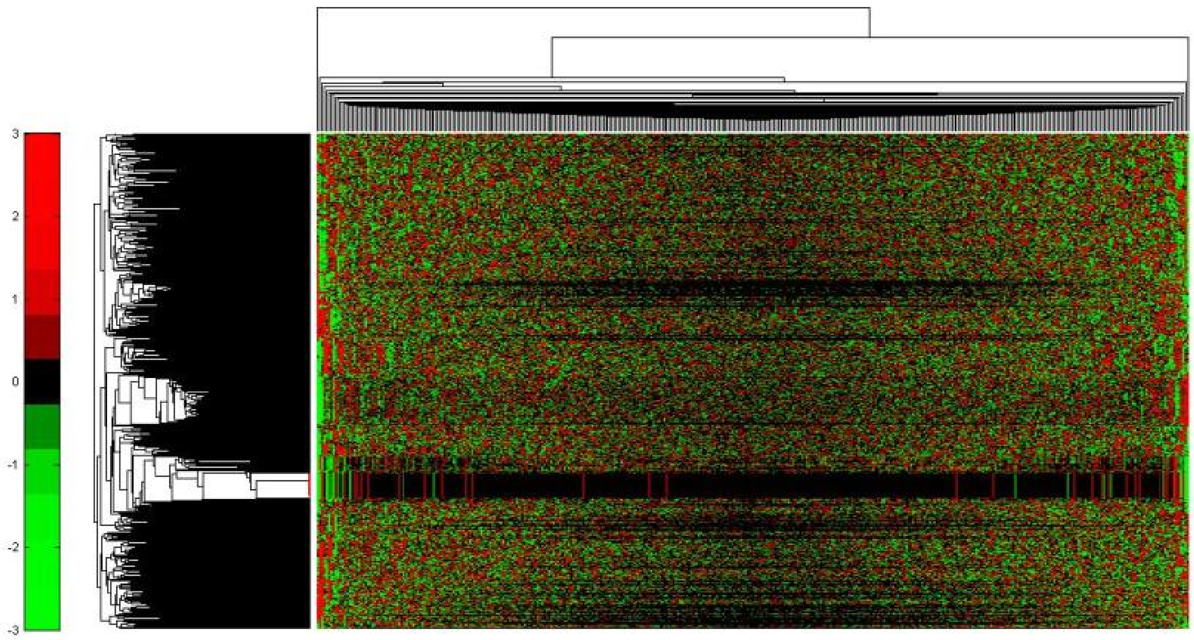


(b) The upper left figure shows the cumulative percentage of the data variance explained by the PCs. The lower left figure plots the loadings of all the observations on to the first two PCs. The upper right figure shows the projection of the observation and feature vectors onto the first two PCs. The lower right figure shows the projection of the observation onto the first two PCs with boundary point markers.

Fig. 6. **Principal Component Representation of the Reward Data:** (a). Data representation under the first two PCs in the LNIRL reward space; (b). Data representation under the first two Principal Components in the GPIRL reward space.



(a)



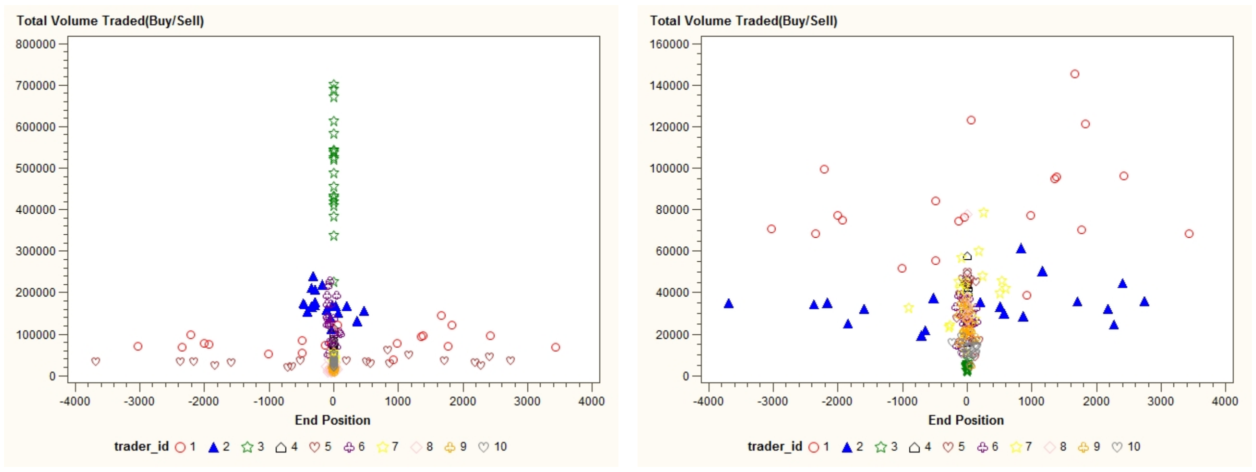
(b)

Fig. 7. **Hierarchical Clustering of Data Matrix:** (a). Heat map of 800 observations of the LNIRL Rewards in the first 200 PCs; (b). Heat map of 800 observations in GPIRL Rewards in the first 400 PCs.

examination, the left dendrogram contains two clusters: the top cluster and the bottom cluster with a black dividing strip in the middle of the second small cluster. We then zoomed into the small cluster and look for the sources of these observations[2]. In the LNIRL reward space, we find that the small cluster consists mostly of observations from trader 1 (observations numbered from 1 to 80) and trader 2 (observations numbered from 81 to 160). Observations from trader 9 (observations numbered from 641 to 720) form the black division between these two groups. In the GPIRL reward space, we find that the small cluster consists of three traders: trader 1 (observations numbered from 1 to 80), trader 2 (observations numbered from 81 to 160), and trader 5 (observations numbered from 321 to 400) with the observations from the rest of the traders lying on the other side of the divide. Moreover, we find that the observations from trader 9 (observations numbered from 641 to 720) form the black division between these two groups. These observations show that the majority of the top 10 traders form one group with 2 or 3 traders behaving a little differently. Furthermore, we observe that the clustering has less than perfect purity. In other words, individual observations from the top cluster occasionally lie in the small cluster at the bottom indicating that behavior changes overtime. The interpretation of this observation is that the HFTs may behave like Opportunistic Traders for a short period of time. We also occasionally observe Opportunistic Traders behaving like HFTs. In this case, observations cross the divide into the top cluster.

Next we propose a continuous measure of clustering using the hierarchical clustering method. We use the summary statistic-based trader classification method proposed by Kirilenko et al. ([2]) to create reference labels. For this market data, we do not have true labels on those trading accounts. We aim to improve the labeling methods documented by Kirilenko et al. The motivation for creating a continuous measure of clustering is to address the potential changes in trading behavior over time. As we mentioned earlier, we applied the summary statistic-based classification rule on the 200 observations over the 4-week period and found we can only consistently label the traders as a single type 40% of the time. We now define a weighted scoring system to evaluate both the rule-based classifier and the behavior-based classifier. Among the 6 types of traders defined in the data section, we only concerned with labeling HFTs,

[2]Note: In both Figure (6) (a) and (b), we group observations from the same trader together in our data matrix. We have 10 traders and each has 80 observations. From the lower left graph in both (a) and (b), observations are ordered sequentially by trader IDs. For example, observations 1 through 80 come from trader 1, and observations 81 through 160 come from trader 2. This continues along the X-axis up to observations 721 through 800 from trader 10.

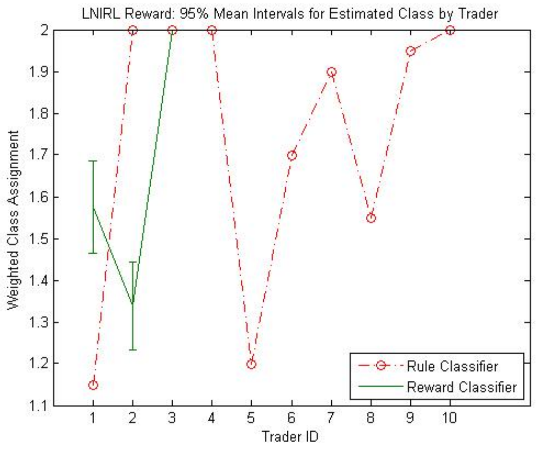


(a) Top 10 Traders (b) Randomly Selected 10 Traders

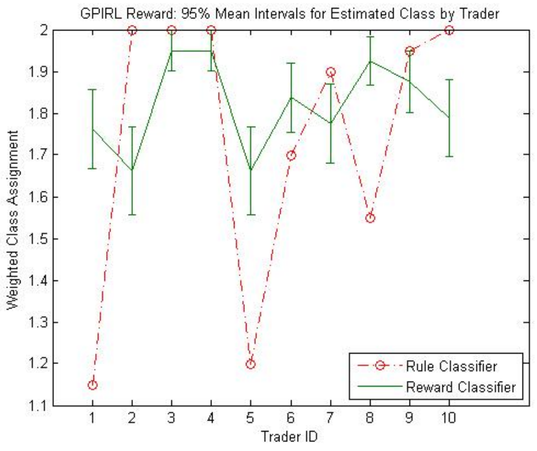
Fig. 8. **Trader's Daily Trading Volume vs. Daily End Position during a 20 Day Period.** (a) Traders 1, 2 and 5 have varying end positions. (b) Traders 1, 2, and 7 have varying end positions.

Intermediaries, and Opportunistic Traders. The other three types of traders, e.g., Fundamental Buyers, Fundamental Sellers, and Small Traders, can be reliably identified by their daily volume and their end of day positions. Here, we assign score 2 if a trader is classified as a HFT; we assign score 1 if a trader is classified as an Opportunistic Trader; and we assign 0 if a trader is classified as an Intermediary. Labels for clustering are assigned using the majority voting rule based on the summary statistic classification rule. We then combine the scores using a weight defined as the frequency with which a particular score is assigned to a particular trader. Here, we want to compare the summary statistic-based trader type classification with the behavior-based trader type classification in an effort to find connections between these two methods.

The visual representations in Figure (8) (a) show that trader 1 and trader 5 have a wide range of end of day positions, but their daily trading volumes remains at relatively the same levels. These traders will likely be classified sometimes as HFTs and sometimes as Opportunistic Traders. While trader 2 exhibits a smaller range of end of day positions than trader 1 and trader 5, the general pattern is very similar to that of traders 1 and 5, and we should classify trader 2 as an Opportunistic Trader. Based on this manual examination, traders 1, 2 and 5 should be classified as Opportunistic Traders and the rest should be classified as HFTs. Now, we compare the results of the summary statistic-based classification rule with those of the behavior-based classification.



(a) Hierarchical clustering in the LNIRL reward space

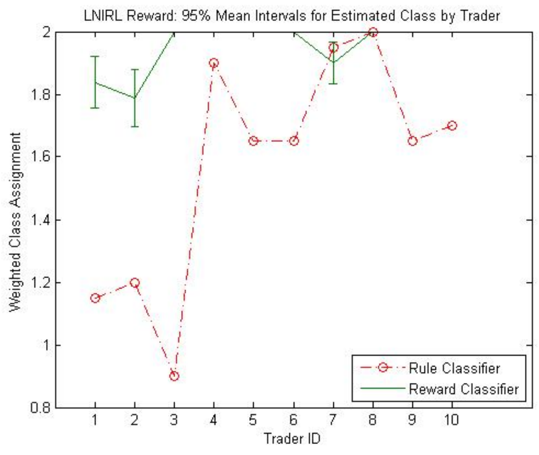


(b) Hierarchical clustering in the GPIRL reward space

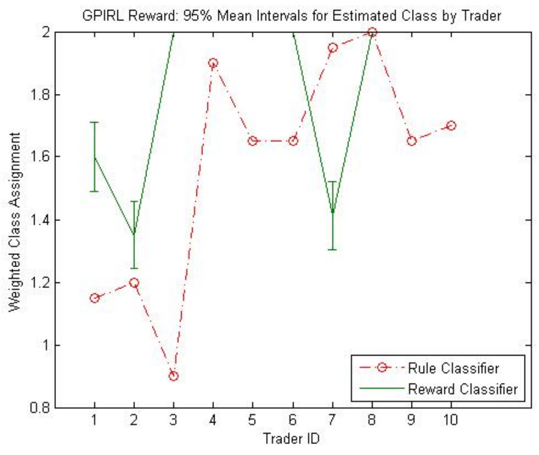
Fig. 9. **Trader Type Classification Compared with the Summary Statistic Based Rule Classification for the Top 10 Traders.**

Figure (9) (a) shows that two groups of traders exist in the LNIRL reward space. Eight out of ten are identified as HFTs and only trader 1 and trader 2 are classified as Opportunistic Traders. This result is consistent with our observation from the dendrogram in Figure (7) (a). When we compare this result with the GPIRL reward space, we can locate all three traders (1, 2, and 5) that we identified through the manual process. This result is also consistent with our observation from the heat map in Figure (7) (b). The statistic-based classification method misclassified trader 2 because the cut-off in the statistic-based approach is based on a simple ratio between the trading volume and the end position. We can see that trader 2 has a relatively small spread of end position. However, the behavior-based approach can identify this pattern and is able to cluster this trader with other traders with similar patterns.

We run another experiment using 10 randomly selected traders out of the traders with the top 30 trading volumes. We know this selection will only result in three types of traders, i.e., HFTs, Intermediaries and Opportunistic Traders. We feed these 800 observations to both LNIRL and GPIRL algorithms to obtain reward representations of their trading behaviors. Based on visual examination (see Figure (8) (b)), we see that trader 1, trader 2 and trader 7 are Opportunistic Traders and the rest are HFTs. We apply the same techniques as before and we use the same cut-off scores (1.85 in the LNIRL reward space, and 1.75 in the GPIRL reward space). As a result,



(a) Hierarchical clustering in LNIRL reward space



(b) Hierarchical clustering in GPIRL reward space

Fig. 10. Trader Type Classification Compared with the Summary Statistic-Based Rule Classification for the 10 Randomly Selected Traders.

we can accurately identify the two classes of traders using the same cut-off score we used for the top 10 case (see Figure (10)). The classification in the LNIRL reward space gives the same result as that in the GPIRL reward space, while the statistic-based classification method misclassified trader 3 as an Opportunistic Trader. Based on the daily end position, daily total trading volume and inventory variance, trader 3 should be classified as a HFT. Again, this misclassification is due to the aggregate cut-off ratio. However, the behavior-based approach can identify this pattern and is able to cluster trader 3 with other traders with similar behavioral patterns. Overall, we argue that the GPIRL reward space score-based classification rule provides an advantage over the summary statistic-based approach in that it is based on similarity in behavior and it can be clearly interpreted. Because the GPIRL rule a better reflects traders' choices of actions under different market conditions than the summary statistics, it is well suited for the discovery of new behavioral patterns of the market participants. We also conclude that the GPIRL reward space is more informative and is a superior measure of trading behavior in terms of the LNIRL reward space.

V. Conclusion

We assume incomplete observation of algorithmic trading strategies and model traders' reward functions as a Gaussian process. We also incorporate traders' action preferences under different market conditions through preference graph learning. The aim of this study is to quantify trader behavior-based on IRL reward learning under a Bayesian inference framework. We apply both a linear approach (a linear combination of known features) ([8]) and GPIRL ([1]) to a real market dataset (The E-Mini S&P 500 Futures), and we conclude that GPIRL is superior to the LNIRL methods, with a 6% greater rate of identification accuracy. Furthermore, we establish a connection between the summary statistic-based classification ([2]) and our behavior-based classification. We propose a score-based method to classify trader types, and because of the transferable property of the reward structure the cut-off score for classifying a group of traders can be applied to different market conditions.

The implication of this research is that reward/utility-based trading behavior identification can be applied to real market data to accurately identify specific trading strategies. As documented by Abbeel et al. ([8]) and confirmed by many other researchers, the reward function is the most succinct, robust, and transferable definition of a control task. Therefore, the behavior learned under the reward space has much broader applicability than observed policies. Furthermore, these learned reward functions will allow us to replicate a particular trading behavior in a different environment to understand their impact on the market price movement and market quality in general.

We also want to note some future research on improving the identification accuracy and discuss applications of this behavioral characterization:

- During our preference learning inference phase, we only considered a simple two layer preference graph. However, traders' preferences can be further distinguished with multi-layer graphs or other preference learning techniques.
- Our study focused on the sampled algorithmic traders on a market. Future studies can extend these results to a large scale experiment to include market participants (specifically Opportunistic traders), and study their behavioral similarity through clustering. We can then associate the group behavior with market quality measures.
- Under the GPIRL framework, we are able to recover a detailed reward structure. These reward functions can be used to generate new policies under a simulated market condition to understand the full behavior of certain trading strategies. This framework provides a particularly interesting way for market regulators to see how the various trading strategies will interact during stressed market conditions.

APPENDIX

[Deterministic Policy versus Randomized Policy] Let $\pi = (d_1, d_2, ...) \in D^{MR}$. The expected total discounted reward of this policy is defined by

$$\begin{aligned}
v_\gamma^\pi = & \sum_{t=1}^{\infty} \gamma^{t-1} P_\pi^{t-1} r_{d_t} \\
= & \, r_{d_1} + \gamma P_{d_1} r_{d_2} + \gamma^2 P_{d_1} P_{d_2} r_{d_3} + \\
= & \, r_{d_1} + \gamma P_{d_1} (r_{d_2} + \gamma P_{d_2} r_{d_2} + \gamma^2 P_{d_2} P_{d_3} r_{d_4} + ...) \\
v_\gamma^\pi = & \, r_{d_1} + v_\gamma^{\pi'}, where, \pi' = (d_2, d_3, ...) \in \Pi^{MR} \qquad (7)
\end{aligned}$$

or it can be expressed in component notation as:

$$v_\gamma^\pi(s) = r_{d_1}(s) + \sum_{j \in S} \gamma P_{d_1}(j|s) v_\gamma^{\pi'}(j) \qquad (8)$$

The interpretation of this equation is that the discounted reward corresponding to policy p equals the discounted reward in a one-period problem in which the decision maker uses decision rule d_1 in the first period and receives the expected total discounted reward of policy π as a terminal reward. However, when p is stationary so that $\pi^{'} = \pi$, they simplify further. Let $d^\infty \equiv (d, d, \ldots)$ denote the stationary policy which uses policy $d^\infty \equiv (d, d, \ldots) \in D^{MR}$ at each decision epoch. For this policy equation (8) becomes

$$v_\gamma^{d^\infty}(s) = r_{d_1}(s) + \sum_{j \in S} \gamma P_{d_1}(j|s) v_\gamma^{d^\infty}(j) \qquad (9)$$

and equation 9 becomes

$$v_\gamma^{d^\infty} = r_{d_1} + \gamma p_d v_\gamma^{d^\infty}, where, d^\infty = (d_2, d_3, ...) \in \Pi^{MR} \qquad (10)$$

Thus, $v_\gamma^{d^\infty}$ satisfies the system equations

$$v = r_d + \gamma P_d v, or v = (I - \gamma \mathbf{P_d})^{\mathbf{-1}} \mathbf{r_d} \qquad (11)$$

The matrix $v = (I - \gamma \mathbf{P_d})^{-\mathbf{1}} \mathbf{r_d}$ plays a crucial role in the theory of discounted Markov decision problems. The optimality equation or **Bellman** equations can be written as:

$$v^*(s) = \sup_{a \in A_s} r_d + \gamma \mathbf{P_d} \mathbf{v} \tag{12}$$

Note that when the supremum on the right-hand side of 13 is attained for all $v \in V$, we define L an operator on V by:

$$Lv \equiv \sup_{d \in D_{MD}} r_d + \gamma \mathbf{P_d} \mathbf{v} \tag{13}$$

Lemma 6: Let w be a real-valued function on an arbitrary discrete set W and let $q(.)$ be a probability distribution on W. Then

$$\sup_{u \in W} w(u) \geq \sum_{u \in W} q(u) w(u) \tag{14}$$

Proof: Let $w* = \sup_{u \in W} w(u)$. Then

$$w* = \sup_{u \in W} q(u) w* \geq \sum_{u \in W} q(u) w(u) \tag{15}$$

Note that the lemma remain valid with W a Borel subset of a measurable space, $w(u)$ an integrable function on W, and the summation replaced by Integration.

■

Proposition 7: For all $v \in V$ and $0 \leq \gamma \leq 1$,

$$\sup_{d \in D_{MD}} r_d + \gamma \mathbf{P_d} \mathbf{v} = \sup_{d \in D_{MR}} r_d + \gamma \mathbf{P_d} \mathbf{v} \tag{16}$$

Proof: Since $D^{MR} \supset D^{MD}$, the right-hand side of 16 must be at least as great as the left-hand side. To establish the reverse inequality, choose $v \in V, \delta \in D^{MR}$ and apply 6 at each $s \in S$ with $W = A_s, q(\textbf{.}) = q_\delta(\textbf{.})$, and

$$w(\textbf{.}) = r(s,\textbf{.}) + \sum_{j \in S} \gamma \mathbf{P}_d(j|s,\textbf{.})v(j)$$

to show that

$$\sup_{d \in A_s} r(s,\textbf{.}) + \sum_{j \in S} \gamma \mathbf{P}_d(j|s,\textbf{.})v(j) \geq \sum_{a \in A_s} q_\delta(a)[\sum_{j \in S} \gamma \mathbf{P}_d(j|s,\textbf{.})v(j)]$$

Therefore, for any $\delta \in D^{MR}$,

$$\sup_{d \in D_{MD}} r_d + \gamma \mathbf{P_d v} \geq r_d + \gamma \mathbf{P_d v}$$

■

From which it follows that the left-hand side of 16 is at least as great as the right-hand side. Hence we show that we obtain the same value of the supremum in a deterministic policy as if we were to allow randomized policy.

Appendix

[Gaussian Processes] *Proof:* Gaussian processes, as mathematical models of random phenomena that are Gaussian distribution, have attracted more and more attention in machine learning field, and they are very useful in applications. Since our work use Gaussian processes, we will give a brief introduction based on the work in [35], [23], [36], and [37].

A. Gaussian Processes Regression and Classification

Given a dataset $\{\mathbf{X}, \mathbf{y}\}$, where $\mathbf{X}$ is a matrix that is composed of N input example vectors $\mathbf{x}_c, c \in \mathcal{N}$ and $\mathbf{y}$ is a vector of corresponding targets value y_c (real value for regression or categorical value for classification). Gaussian process model treats the latent functions as random processes with Gaussian prior, which is different from the parametric form in classical statistical models. Denote the latent function by $u(\mathbf{x}_c)$, which is assumed to be a random process. Then the first and second order statistics of $u(\mathbf{x}_c)$ are its mean function $m(\mathbf{x}_c)$ and covariance function $k(\mathbf{x}_c, \mathbf{x}_d), \forall c, d \in \mathcal{N}$.

Both estimation of mean function and variance function depend on the finite dimensional distribution. Since Gaussian process is a process whose finite dimensional distributions are Gaussian, a Gaussian process is determined by its mean and variance functions. For every set of realizations of random variables of a Gaussian process, the symmetric matrix $\mathbf{K}$, whose entries are calculated by the covariance function $k(\mathbf{x}_c, \mathbf{x}_d)$, is positive semi-definite[3]. It is sufficient that for $\mathbf{K}$ is positive semi-definite, there exists a Gaussian random field with this covariance matrix and zero-mean function $m(\mathbf{x}_c) = 0$ (Kolmogorov's theorem [38]).We denote such random field as $u(\mathbf{x}_c) \sim N(0, \mathbf{K})$.

The simplest Gaussian process model is $y_c = u(\mathbf{x}_c) + \epsilon$, where ϵ is an independent Gaussian noise, written as $\epsilon \sim N(0, \sigma_\epsilon^2)$. Within a Bayesian framework, the inference of $u(\mathbf{x})$ at the test location $\mathbf{x}$ is described by maximization of the posterior probability

$$p(u(\mathbf{x})|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, u(\mathbf{x}))p(u(\mathbf{x}))}{p(\mathbf{y}|\mathbf{X})}.$$

The joint distribution of the observed target values and the function values follows a Gaussian distribution. Therefor the posterior conditional distribution is written as

$$u(\mathbf{x})|\mathbf{X}, \mathbf{y} \sim N(\mathbf{K}(\sigma_\epsilon^2\mathbf{I} + \mathbf{K})^{-1}\mathbf{y}, \sigma_\epsilon^2(\sigma_\epsilon^2\mathbf{I} + \mathbf{K})^{-1}\mathbf{K}).$$

The generative model based on Gaussian processes paves an efficient road in which we are able to perform inference with finite observations in the large space. And it is also worth mentioning that it keeps tractable computation while offering a guarantee of performance.

B. Gaussian Process Preference Learning

In machine learning field, a learning scenario, called learning label preference, studies how to find the latent function that predicts preference relations among a finite set of labels, for an instance from the instance space. This scenario is a generalization of some standard settings, such as classification and label ranking [39]. Considering the latent function values as Gaussian process, Chu and Ghahramani observed that Bayesian framework is an efficient and competitive method for learning label preferences [23]. They proposed a novel likelihood function to capture

[3]Positive semi-definite implies that $\mathbf{x}^T\mathbf{K}\mathbf{x} \geq 0$ for all $\mathbf{x}$.

the preference relations using *preference graph*, a directed graph encoding the label preferences of each sample [40, 41].

Let $u(\mathbf{x}, y)$ denote the latent function depending on both label y and instance $\mathbf{x}$, and $\mathcal{G}$ denote the observed preference graphs. The Bayesian inference is written as

$$\hat{u}(\mathbf{x}, y) \triangleq \arg\max_{u(\mathbf{x},y)} p(u(\mathbf{x}, y)|\mathcal{G}) \propto \arg\max_{u(\mathbf{x},y)} p(\mathcal{G}|u(\mathbf{x}, y))p(u(\mathbf{x}, y)) \tag{17}$$

where $p(\mathcal{G}|u(\mathbf{x}, y))$ is the likelihood function derived from preference relations. Given a new instance $\mathbf{x}^*$, the labels y^* can be predicted by ranking the values of latent function, $y^* = \arg\max_y \hat{u}(\mathbf{x}^*, y), y \in \mathcal{Y}$, where $\mathcal{Y}$ is a finite set of labels.

We also use Bayesian inference and build off several of the ideas in [10] and related work, but our method differs from label preference learning for classification and label ranking. Our input data depends on states and actions in the context of an MDP. Moreover, we are learning the reward that indirectly determines how actions are chosen during the sequential evolution of an MDP, while preference learning studies the latent functions preserving preferences. On the grounds of Bellman optimality for MDPs, the decision maker chooses optimal actions with the maximum value of Q-function at a given state. The preference relation will be determined by Q-functions, the expected long-term reward, while the random variable we concern is the immediate reward function, the intrinsic function determining the decision maker's behavior. Next, we give the details of our method. ■

APPENDIX

[Convex Optimization Problem] *Proof:* The second order derivative with respect to $\mathbf{r}_{a_m}$ is

$$\begin{aligned}\frac{\partial^2 U}{\partial \mathbf{r}_{a_m} \partial \mathbf{r}_{a_m}^T} &= \mathbf{K}_{a_m}^{-1} + \frac{\partial^2 \sum_{i=1}^{n} \sum_{k=1}^{n_i} - \ln \Phi(\sum_{j=1}^{m} \rho_{a_m}^{ik} \hat{\mathbf{r}}_{a_m})}{\partial \mathbf{r}_{a_m} \partial \mathbf{r}_{a_m}^T} \\ &\quad + \sum_{i=1}^{n} \sum_{l=1}^{m_i} (\rho_{a_m}^{il})^T \rho_{a_m}^{il} \end{aligned} \tag{18}$$

It is obvious that $\mathbf{K}_{a_m}^{-1}$ and $(\rho_{a_m}^{il})^T \rho_{a_m}^{il}$ are positive definite matrix. If the second part in Eq.18 is also positive definite, the minimization of Eq.5 is a convex problem. Let $\mathbf{W}$ denote the $n \times n$ matrix for the second part and W_{cd} be the entry at c-th row and d-th column, which is calculated in the following,

$$W_{cd} = \frac{\partial^2 \sum_{i=1}^{n} \sum_{k=1}^{n_i} - \ln \Phi(\sum_{j=1}^{m} \rho_{a_m}^{ik} \hat{\mathbf{r}}_{a_m})}{\partial \mathbf{r}_{a_m}(s_c) \partial \mathbf{r}_{a_m}(s_d)} \tag{19}$$

and let $z_i^k \triangleq \sum_{j=1}^{m} \rho_{a_m}^{ik} \hat{\mathbf{r}}_{a_m}$. We have

$$\frac{-\partial ln \Phi(z_i^k)}{\partial \mathbf{r}_{a_m}(s_c)} = -\frac{\rho_{a_m}^{ik}(x_c) N(z_i^k|0,1)}{\sqrt{2}\sigma \Phi(z_i^k)}$$
$$\frac{-\partial^2 \ln \Phi(z_i^k)}{\partial \mathbf{r}_{a_m}(s_c) \partial \mathbf{r}_{a_m}(s_d)} =$$
$$\frac{\rho_{a_m}^{ik}(s_c) \rho_{a_m}^{ij}(s_d) N(\sum_{j=1}^{m} \rho_{a_m}^{ik} \hat{\mathbf{r}}_{a_m}|0,1)}{2\sigma^2 \Phi(z_i^k)} \left[z_i^k + \frac{N(z_i^k|0,1)}{\Phi(z_i^k)} \right]$$

where let $\omega_{ik} = \frac{N(z_i^k|0,1)}{2\sigma^2 \Phi(z_i^k)} \left[z_i^k + \frac{N(z_i^k|0,1)}{\Phi(z_i^k)} \right]$, we have

$$W_{cd} = \begin{cases} \sum_{i=1}^{n} \sum_{k=1}^{n_i} \left[\rho_{a_m}^{ik}(s_c) \right]^2 \omega_{ik} \geq 0 \text{ if c=d} \\ \sum_{i=1}^{n} \sum_{k=1}^{n_i} \rho_{a_m}^{ik}(s_c) \rho_{a_m}^{ik}(s_d) = W_{dc} \text{ otherwise} \end{cases} \tag{20}$$

Let $y = [y_1, y_2, \cdots, y_n]$ denotes a $n \times 1$ vector. Then

$$\begin{aligned} y^T W y = &\sum_{i=1}^{n} \sum_{k=1}^{n_i} y_1 \sum_{b=1}^{n} \rho_{a_m}^{ik}(s_b) \rho_{a_m}^{ik}(s_1) y_b \\ &+ \sum_{i=1}^{n} \sum_{k=1}^{n_i} y_2 \sum_{b=1}^{n} \rho_{a_m}^{ik}(s_b) \rho_{a_m}^{ik}(s_2) y_b \\ &+ \cdots + \sum_{i=1}^{n} \sum_{k=1}^{n_i} y_n \sum_{b=1}^{n} \rho_{a_m}^{ik}(s_b) \rho_{a_m}^{ik}(s_n) y_b \\ &= \sum_{i=1}^{n} \sum_{k=1}^{n_i} [\sum_{b=1}^{n} y_b^2 [\rho_{a_m}^{ik}(s_b)]^2 \\ &+ 2 \sum_{b=1}^{n} \sum_{b' \neq b} y_b y_{b'} \rho_{a_m}^{ik}(s_b) \rho_{a_m}^{ik}(s_{b'})] \\ &= \sum_{i=1}^{n} \sum_{k=1}^{n_i} \left(\sum_{b=1}^{n} y_b \rho_{a_m}^{ik}(s_b) \right)^2 \geq 0 \end{aligned} \tag{21}$$

we prove the matrix W is semi-positive definite. So the Hessian matrix of Eq.5 is positive semi-definite on the interior of a convex set. Hence, minimizing Eq.5 is a convex programming problem. ■

References

[1] Q. Qiao and P. Beling. Inverse reinforcement learning with gaussian process. *Proceedings of 2011 American Control Conference*, 2011.

[2] A. Kirilenko, A. S. Kyle, M. Samadi, and T. Tuzun. The flash crash: The impact of high frequency trading on an electronic market. (1686004), 2011.

[3] I. Aldridge. *A Practical Guide to Algorithmic Strategies and Trading Systems - High Frequency Trading*. John Wiley & Sons, Inc, 2010.

[4] W.F. Sharpe. Mutual fund performance. *Journal of Business*, 39:119–138, 1966.

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

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

[7] Daw Nathaniel D., Niv Yael, and Dayan Peter. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8:1704–1711, 2005.

[8] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. pages 1–, 2004.

[9] Nathan Ratliff Brian Ziebart Kevin Peterson J. A. Bagnell Martial Hebert Anind K. Dey and Siddhartha Srinivasa. Inverse optimal heuristic control for imitation learning. In *Proc. AISTATS*, pages 424–431, 2009.

[10] A. Y. Ng and S. Russel. Algorithms for inverse reinforcement learning. *In Proc. ICML*, pages 663–670, 2000.

[11] Umar Syed Michael Bowling and Robert E. Schapire. Apprenticeship learning using linear programming. In *Proc. 25th international Conf. on Machine learning*, pages 1032–1039. ACM, 2008.

[12] Umar Syed Robert E. Schapire. A game-theoretic approach to apprenticeship learning. In *In Advances in Neural Information Processing Systems*, pages 1449–1456. MIT Press, 2008.

[13] Gergely Neu and Csaba Szepesvari. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proc. Uncertainty in Artificial Intelligence*, 2007.

[14] 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*, page 14331438, 2008.

[15] Ramachandran Deepak and Amir Eyal. Bayesian inverse reinforcement learning. In *Proc. 20th International Joint Conf. on Artificial Intelligence*, 2007.

[16] Krishnamurthy Dvijotham and Emanuel Todorov. Inverse optimal control with linearly-solvable mdps. In *Proc. 27th International Conf. on Machine learning*. ACM, 2010.

[17] Abdeslam Boularias and Brahim Chaib-draa. Bootstrapping apprenticeship learning. In *Advances in Neural Information Processing Systems 24*. MIT press, 2010.

[18] Sergey Levine Zoran Popovic and Vladlen Koltun. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems 24*. MIT press, 2010.

[19] Pieter Abbeel Adam Coates and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, (1-31), 2010.

[20] S.J. Lee and Popovi Zoran. Learning behavior styles with inverse reinforcement learning. In *SIGGRAPH '10: ACM SIGGRAPH 2010 papers*, pages 1–7, New York, NY, USA, 2010. ACM.

[21] A.A.Kirilenko A.S.Kyle M.Samadi and T.Tuzun. The flash crash: The impact of high frequency trading on an electronic market. *SSRN Working Paper*, 2010.

[22] Bellman R. *Dynamic programming*. Princeton University Press, 1957.

[23] Chu Wei and Ghahramani Zoubin. Preference learning with gaussian processes. In *Proc. 22th Iinternational Conf. on Machine learning*, pages 137–144. ACM, 2005.

[24] S.Y. Yang, M.E. Paddrik, R.J. Hayes, T. Andrew, A.A. Kirilenko, P. Beling, and W. Scherer. Behavior based learning in identifying high frequency trading strategies. *Proceedings of IEEE Computational Intelligence in Financial Engineering and Economics, 2012*, 2012.

[25] D. Easley, M.M. Lopez de Prado, and M. O'Hara. The microstructure of the "flash crash". *Cornell University Working Paper, 2010*, 2010.

[26] J.A. Brogaard. High frequency trading and its impact on market quality. *Ph.D. Dissertation, Kellogg School of Management, Northwestern University*, 2010.

[27] V. Vapnik. *The Nature of Statistical Learning Theory*. 2nd Edition, Spinger, New York, 1999, 1999.

[28] C.J.C. Burges. A tutorial on support vector machine for pattern recognition. *Data Mining*

Knowledge Discovery, 2:121, 1998.

[29] T. Joachims. *Making large scale SVM learning practical*. MIT Press, Cambridge, MA, 1998.

[30] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[31] P. Ekman. Intraday patterns in the s&p 500 index futures market. *The Journal of Futures Markets*, 12:365–381, 1992.

[32] A.R. Admati and P. Pfleiderer. A theory of intraday patterns: Volume and price variability. *The Review of Financial Studies*, 1:3–40, 1988.

[33] Y.T. Lee, R.C Fox, and Y. Liu. Explaining intraday pattern of trading volume from the order flow data. *Journal of Business Finance and Accounting*, 28:306–686, 2001.

[34] T. Chordia, R. Roll, and A. Subrahmanyam. Market liquidity and trading activity. *Journal of Finance*, 56:501–530, 2001.

[35] Carl Edward Rasmussen and Christopher K.I.Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[36] Matthias Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14:2004, 2004.

[37] Takeyuki Hida and Masuyuki Hitsuda. *Gaussian Processes*. American Mathematical Society, 1993.

[38] A.N. Kolmogorov. *Foundations of the Theory of Probability*. AMS Chelsea, 2nd edition, 1956.

[39] J. Furnkranz and E. Hullermeier. Preference learning. In *Kunstliche Intelligenz*, 2005.

[40] Fabio Aiolli and Alessandro Sperduti. Learning preferences for multiclass problems. In *Advances in Neural Information Processing Systems 17*, pages 17–24. MIT Press, 2004.

[41] Ofer Dekel Christopher D. Manning and Yoram Singer. Log-linear models for label ranking. In *21st International Conference on Machine Learning*, 2004.