Modeling Stock Order Flows and Learning Market-Making from Data

Adlar J. Kim, Christian R. Shelton, and Tomaso Poggio

Abstract

Stock markets employ specialized traders, market-makers, designed to provide liquidity and volume to the market by constantly supplying both supply and demand. In this paper, we demonstrate a novel method for modeling the market as a dynamic system and a reinforcement learning algorithm that learns profitable market-making strategies when run on this model.

The sequence of buys and sells for a particular stock, the order flow, we model as an Input-Output Hidden Markov Model fit to historical data. When combined with the dynamics of the order book, this creates a highly non-linear and difficult dynamic system. Our reinforcement learning algorithm, based on likelihood ratios, is run on this partially-observable environment. We demonstrate learning results for two separate real stocks.

This report describes research done at the Center for Biological & Computational Learning, which is affiliated with the McGovern Institute of Brain Research and with the Artificial Intelligence Laboratory, and which is in the Department of Brain & Cognitive Sciences at MIT.

This research was sponsored by grants from: Office of Naval Research (DARPA) under contract No. N00014-00-1-0907, National Science Foundation (ITR) under contract No. IIS-0085836, National Science Foundation (KDI) under contract No. DMS-9872936, and National Science Foundation under contract No. IIS-9800032.

Additional support was provided by: AT&T, Central Research Institute of Electric Power Industry, Center for e-Business (MIT), Daimler-Chrysler AG, Compaq/Digital Equipment Corporation, Eastman Kodak Company, Honda R&D Co., Ltd., ITRI, Komatsu Ltd., Merrill-Lynch, Mitsubishi Corporation, NEC Fund, Nippon Telegraph & Telephone, Oxygen, Siemens Corporate Research, Inc., Sumitomo Metal Industries, Toyota Motor Corporation, WatchVision Co., Ltd., and The Whitaker Foundation.

C.S. was supported by ONR contract N00014-00-1-0637 under the MURI program “Decision Making under Uncertainty” while at Stanford.

Introduction

Many economic markets, including most major stock exchanges, employ market-makers to aid in the transactions and provide a better quality market. Each commodity has one or more market-makers assigned to it. The market-maker's responsibility is to set prices and volumes for buying and selling. In particular, a market-maker will constantly quote a bid price and an ask price (the prices at which the market-maker is willing to buy or sell respectively) as well as associated sizes (the maximum volume to which the market-maker is committed at that price).

Market-makers supply an advantage to the market. By consolidating the trading in a few agents, the market becomes more efficient. Traders wishing to buy or sell do not need to find each other or wait for each other's arrival. Additionally, by quoting a single price which is guaranteed for all traders, market-makers can help to smooth price fluctuations due to spurious supplies or demands.

Market-makers benefit themselves from their position. They have an informational advantage over because they see more of the orders than an average trader. This is offset either by institutional regulations or by competition from other market-makers. Although they serve as a clearing house for orders, they make trades against their personal cash and inventory to cover their own quotes. Such a position carries risk.

Many major markets are now electronic. The NASDAQ is a distributed trading system completely run through networked computers. It uses competing market-makers to maintain a high quality market. However, the demands on human market-makers are high. A typical market-maker is responsible for 10 to 20 securities. At any given moment, it is only feasible for the human to be actively attentive to 2 to 3 of them. The market-maker is generally losing potential profit or volume on the other securities. Many other smaller markets (on-line and off-hour trading systems) have emerged as a result of the recent increases in computer networking. These systems are usually too small to employ human market-makers. Instead, orders are crossed against other orders that happen to be present at the time of the trade. This results in many unfilled orders and the loss of potential customers.

The goal of this paper is to explore the potential for automated electronic market-making. For the case of established systems like the NASDAQ, such a system could fill the role of an "autopilot" by taking care of stocks in an intelligent manner while being supervised by a human. This would allow a single human to more successfully manage a large set of securities. For the case of small on-line trading systems, such an automated system could replace the existing naive order crossing mechanism to provide a better market to its traders.

Previous Work

There has been much work on understand both the price formation process and market-making strategies. There are two main approaches from the economics literature. One focuses on the uncertainties in the order flow (the time series of orders placed) and the inventory of the market-maker (Garman 1976; Amihud & Mendelson 1980; Ho & Stoll 1981; O'Hara & Oldfield 1986). The other tries to explain the the price setting dynamics using information-based models (Glosten & Milgrom 1985). Most of these studies have developed conditions for optimally but provided no explicit price adjustment policies. For example, in Amihud & Mendelson (1980), bid and ask prices are shown to relate to inventory, but the exact dependence is unavailable. Some analyses do provide functional forms for market-making policies (O'Hara & Oldfield 1986), but the practical application of the results is limited due to stringent assumptions made in the models.

In previous work (Chan 2001; Chan & Shelton 2001; Shelton 2001a), we developed a simple information-based market model (similar to that of Glosten & Milgrom) and employed reinforcement learning to derive explicit market-making policies. The reinforcement learning algorithm knew nothing of the underlying assumed model and therefore its application was general. By using a flexible method based on experience, we hoped that we could apply the same algorithm to more complex models with differing dynamics without change.

Contribution

In this paper, we continue this research by building a more complex market model based on real market data. We employ the same reinforcement learning algorithm that we used in previous information-based models to derive a profitable market-making strategy. This strengthens our statement that by using a learning strategy we are assuring that the algorithm's success is not predicated on the exact details of our market model.

Modeling the Stock Order Flow

In order to argue how well the market-making model will perform in the real market, it is important to test under a realistic market environment. Previously, we tested under a simple environment where number of unrealistic assumptions were involved. Although the algorithm performed well in such market, the market assumption (*i.e.* the existence of a true price process and differentiation of the informed and uninformed traders) over-simplified the model. Additionally, the trading crowd did not react to any of the market-maker's actions nor their previous actions.

In this paper, we propose a new approach to empirically model the aggregated behavior of the trading crowd. Practically, this involves modeling the time sequence of orders placed on the market, called the order flow. Our new approach is more realistic than the previous model for two reasons. First, the new model does not assume any "true" pricing process. In the real market, the true price of a stock is generally unknown; the price emerges from the aggregated behaviors of the trading crowd. Each trader has a different preference toward the stock and their behavior can change based on their private or public information. Such information is disseminated or aggregated by the trading process. The trading process is the source of information to traders

(Easley, Kiefer, & O'Hara 1997). Second, the new model will react to the market-maker's quotes. The quoted bid and ask prices and the corresponding bid/ask spread (the difference between the two values) are relevant to trading crowd's decision making process. Therefore, in this paper we will generate orders based on probability distribution conditioned by the trading process and the market-maker's quotes.

TORQ Dataset

We fit our model to the transaction-level historical dataset - Trade, Order, and Quote (TORQ) database from New York Stock Exchange (NYSE). The TORQ dataset consists of trade, order, and quote information for 144 stocks traded at NYSE from November 1990 to January 1991. The trade and quote data show the transactions and changes of price and size of bid/ask in the market. Although TORQ contains 100% of trade and quote data, full order data are not available in the dataset. The order data covers only the orders that entered through the NYSE's electronic order routing system (SuperDot system). Other orders, which are usually larger, are assisted by floor brokers. In 1992, 75% of orders entered the market via SuperDot but only accounted for 28% of the executed share volume (Hassbrouk, Sofianos, & Sosebee 1993). Due to such limitation in our dataset, several assumptions are needed to be made in our order flow model. We will discuss this more in the later section.

Variables and Assumptions

Orders. The order flow model generates 4 types of orders: market[*] buy, market sell, limit[†] buy and limit sell. To define such orders, we introduce 4 variables:

- Arrival Time: The time difference between current order and preceding order.
- Size: The number of shares of an order.
- Side: Defines whether an order is buy or sell.
- Price: The price of an order. This only applies to limit orders.

We assume that the orders are stochastically generated conditioned on the market conditions. If the same market conditions are given as the training data, the generated orders will form a similar trading process. If different market conditions are given, the system will generate a sequence of orders that react to the new conditions.

Market condition. The market conditions will be determined by the orders that are already generated and by the market-maker's actions. We will introduce 5 variables, which represent the market conditions at time t:

- Difference between bid and ask price (bid/ask spread): $P_{ask,t} - P_{bid,t}$.
- Difference between bid and ask size: $S_{ask,t} - S_{bid,t}$.
- Return of transaction price: $\frac{P_{trans,t}}{P_{trans,t-1}} - 1$
- Price moving average of 4 successive bids: $\frac{1}{4}\left[\frac{P_{bid,t}}{P_{bid,t-1}} + \frac{P_{bid,t-1}}{P_{bid,t-2}} + \frac{P_{bid,t-2}}{P_{bid,t-3}} + \frac{P_{bid,t-3}}{P_{bid,t-4}} - 4\right]$
- Price moving average of 4 successive asks: $\frac{1}{4}\left[\frac{P_{ask,t}}{P_{ask,t-1}} + \frac{P_{ask,t-1}}{P_{ask,t-2}} + \frac{P_{ask,t-2}}{P_{ask,t-3}} + \frac{P_{ask,t-3}}{P_{ask,t-4}} - 4\right]$

[*]An order to buy or sell a security at the current market price.

[†]An order to buy or sell a security at a specific price

The bid and ask prices and sizes are the best buy and sell quotes (and respective sizes) currently available on the market. We describe this in more detail in our discussion of generating order from the model below.

Assumptions To create a trainable order flow architecture, the following assumptions were made.

- The arrival times and sizes of orders are distributed in accordance with fixed gamma distributions.
- The side and price of orders are normally distributed given market conditions.
- The side and price of an order at time t and side and price of an order at time $t - 1$ are dependent given the market conditions.
- The arrival time and size of orders are independent from the side and price of orders.
- At every time step, the floor orders and electronic orders are similar in the sense that both can represent the current market state.

The last assumption was made to overcome the limitation of incomplete order data in TORQ dataset. It is safe to assume that both electronic and floor orders have similar characteristics in reacting to current market conditions.

Order Flow Model - Arrival Time and Size

For the arrival time and size, we fit the gamma distributions using the maximum likelihood estimators.

$$\alpha, \lambda = \text{gamma distribution parameters.}$$

The gamma probability density function:

$$f(x|\alpha, \lambda) = \frac{1}{\Gamma(\alpha)}\lambda^\alpha x^{\alpha-1} e^{-\lambda x}$$

By taking derivatives of the log-likelihood function with respect to each parameter, we can calculate the maximum likelihood estimators.

$$\begin{aligned}
\frac{\partial}{\partial\lambda} l(x|\alpha, \lambda) &= \frac{\alpha}{\lambda} - \overline{x} \\
&= 0 \\
\lambda &= \frac{\alpha}{\overline{x}} \\
\frac{\partial}{\partial\alpha} l(x|\alpha, \lambda) &= \log\frac{\alpha}{\overline{x}} + \overline{\log x} - \Psi(\alpha) \\
&= 0 \\
\text{where } \Psi(\alpha) &= \frac{\Gamma'(\alpha)}{\Gamma(\alpha)} .
\end{aligned}$$

The above equation cannot be solved in closed form since it is a nonlinear equation. Therefore, we used bisection method to find the root.

Order Flow Model - Buy/Sell and Price

In modeling the side and price of orders, we used Input/Output Hidden Markov Models (IOHMMs) (Rabiner 1989; Bengio 1999). IOHMMs are well suited model to generate orders sequentially over time. At each time step, the different order will be generated based on current market state (hidden) and the current market condition, which can partly be controlled by the market-maker.

Input/Output Hidden Markov Models (IOHMMs). Like regular Hidden Markov Models (HMMs), the IOHMMs are a probabilistic model with a certain fixed number of hidden states. Unlike HMMs, the output distribution and transition distribution are not only conditioned on the current state, but they are also conditioned on an observed input value. The IOHMM is formally defined by

- Number of states q.
- Initial state distribution $P_0(S_0)$.
- State transition probabilities $P_1(S_{t+1}|S_t, U_t)$.
- Output probabilities $P_o(O_t|S_t, U_t)$.

where additional U_t denotes an input variable at time t.

In our model, the output will be a two dimensional vector, which will represent side and price of the order. The input U_t will be a discrete variable, which will represent one of two different market conditions. In the training data, we used the k-means algorithm to cluster the market conditions into two sets. Then we labeled each order data with corresponding market condition. Finally, we used two hidden states in our model to represent market states.

EM for IOHMM Training IOHMMs using EM algorithm is somewhat similar to HMMs, but it incorporates input variables. The forward-backward probabilities are

$$\begin{aligned}
\alpha_t(i) &= P(O_0, ..., O_t, S_t = i|U_t) \\
&= [\Sigma_j \alpha_{t-1}(j) P_1(S_t = i|S_{t-1} = j, U_{t-1})] \cdot \\
&\quad P_o(O_t|S_t = i, U_t) \\
\beta_t(i) &= P(O_{t+1}, ..., O_T|S_t = i, U_t) \\
&= \Sigma_j P_1(S_{t+1} = j|S_t = i, U_t) \cdot \\
&\quad P_o(O_{t+1}|S_{t+1} = j, U_{t+1})\beta_{t+1}(j) \\
\gamma_t(i) &= P(S_t = i|O_0, ..., O_T, U_0, ..., U_T) \\
&= \frac{\alpha_t(i)\beta_t(i)}{\Sigma_j \alpha_t(j)\beta_t(j)} \\
\xi_t(i,j) &= P(S_t = i, S_{t+1} = j|O_0, ..., O_t, U_0, ..., U_T) \\
&= [\alpha_t(i) P_o(O_{t+1}|S_{t+1} = j, U_{t+1}) \\
&\qquad P_1(S_{t+1} = j|S_t = i, U_t)\beta_{t+1}(i)] \\
&\qquad\quad /(\Sigma_j \alpha_t(j)\beta_t(j)) \ .
\end{aligned}$$

Using above probabilities, we can run EM.

- E-step: Compute the posterior probabilities $\gamma_t(i)$ and $\xi_t(i,j)$ for all states i and time t.
- M-step: Update the transition probabilities using $\xi_t(i,j)$ for each input value. Solve the maximum likelihood estimation problem. For each input value and state, we maximize:

$$\Sigma_{t=0}^{T} \gamma_t(i) \log P(O_t|S_t = i, U_t)$$

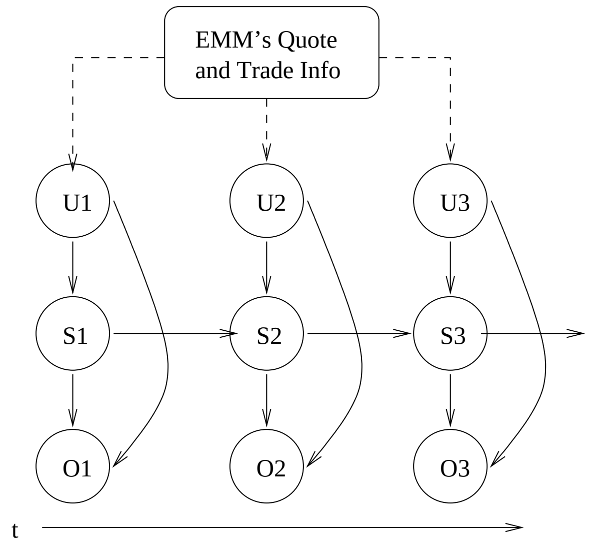

Figure 1: Side and Price Generation

Generating Orders from Order Flow Model

For each new order, the arrival time and size are sampled from the estimated gamma distributions. For the side and price of the order, more work has to be done. First, the current input values are calculated by taking current values of the market condition (a five dimensional vector) and measuring the Euclidean distances to the means of the clusters of market condition vectors in training data. The nearest cluster is used as the input for the time step of the IOHMM. The output (order) of IOHMM is sampled from the output distribution Gaussian conditioned on the input and the state. If the price is less than the current best selling price (for buy orders) or greater than the current best buying price (for ask orders), the order is assumed to be a market order. Otherwise, it is treated as a limit order. Finally, the next state is generated from the transition probabilities given the current state and the input. Figure 1 illustrates the side and price generation process.

Order Book We keep an order book of all previous limit orders that have not been completely filled. If the order can be satisfied by a corresponding order on the opposite side of the book, the two orders are crossed and executed against each other to the extend allowed by their volumes. The new order is crossed against as many orders in the book as possible until it no longer matches or the entire volume of the order has been filled. Orders on the book of the same price are matched in the order in which they were received. If the new order still has remaining volume unfilled, it is placed on the book (with the potentially reduced volume).

The ask price is the smallest sell price in the book. The bid price is the largest buy price in the book. The corresponding volumes are the sum of the volumes in the book with these prices. The ask price is always greater than the bid price (or the two orders would have been executed against each other). The book can be thought of as two lists (on for each side: the buy side and the ask side). The order book can be viewed as two lists (one for each side: the buy side and the sell side) that are sorted by price in opposite directions. The top (best) order on each list is the first order against which incoming transactions are matched. The market-maker's position (bids and asks) are also entered in the book along with the other orders. The market-maker affects the model by changing the market situation with its bids and asks (which affect the book) and they corresponding transactions (which remove orders from the book).

Experimental Results

We compared the market model's reaction when market maker's quotes are significantly different. The Figure 2 shows the traded price over time when market maker was forced to quote bid and ask prices with spread of \$1 and \$10 respectively. We trained the market model based on the data from the stock symbol IBM for November 1, 1990.

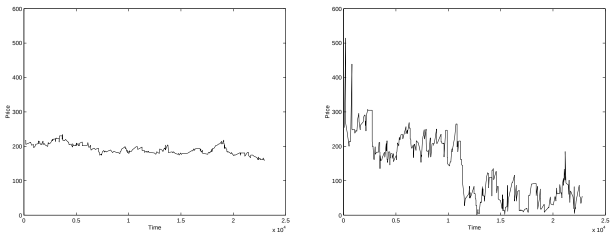

Figure 2: The price changes over time when the market maker was forced to quote with bid/ask spread of \$1 and \$10 respectively.

The result shows that when the market maker is forced to quote prices with larger spread, the market's volatility is significantly increased. The average bid/ask spread of above cases are \$0.0906 and \$7.7768. However, since the same market model were used for both cases, the volume weighted average prices were similar (\$227.51 and \$237.52).

Reinforcement Learning

The goal of reinforcement learning is to find a strategy for acting in an unknown environment that will maximize the long-term return of the agent. Because the dynamics of the environment are unknown, experience is used to guide the search for better strategies.

The goals of a market-makers can be many-fold. In particular, they are interested in maximizing their profit, minimizing their risk, maximizing the volume traded, and minimizing the resources (cash, inventory) necessary. In (Shelton 2001a) we discuss methods for balancing various goals in the context of reinforcement learning in detail. For this paper, we will assume the only goal of the market-maker is to maximize profits.

Reinforcement Learning for Markets

The profits of a market-making system can only be realized once the inventory has been liquidated. The reinforcement learning algorithm needs a return (profit in this case) at the end of each episode by which to judge it ability. Therefore, at the end of an episode, we force the market-maker to zero out its inventory by quoting bids or asks to sell or buy as much stock as needed to reset the inventory. This may take a number of model time steps. During this time, the reinforcement learning strategy has no control over the agent. The return is for the episode is set to be the resulting profit after the inventory has been liquidated.

During the market simulation, the market-maker may quote a price that is worse than the current best price in the book (on either the buy or sell side). In this case, the best bid or ask from the book are used as input to the market model. The market-maker's bids and asks sit like any other orders on the book. This can be viewed as either the market-maker is competing to buy and sell like any other agent (possibly against other market-makers) or that the market-maker can quote the better price and volume and make the trade without resorting to personal inventory and does not need to change its internal position do to so.

Importance Sampling Algorithm

In (Shelton 2001b) we developed a reinforcement learning algorithm for partially-observable environments based on an episodic learning framework. It uses importance sampling estimates to calculate the expected return for untried strategies. Although there is not space in this paper to fully detail the algorithm, we will describe its basics and refer the reader to our previous paper for more details.

We assume that after T episodes, the market-maker has tried strategies π_1 through π_T and observed as a consequence histories[‡] h_1 through h_T and returns R_1 through R_T. The estimated return for a new strategy π is

$$\hat{R}(\pi) = \frac{\sum_i R_i \frac{p(h_i|\pi)}{\sum_j p(h_i|\pi_j)}}{\sum_i \frac{p(h_i|\pi)}{\sum_j p(h_i|\pi_j)}}$$

where $p(h_i|\pi)$ is the probability of history h_i happening when the agent is executing strategy π. $p(h_i|\pi)$ is not computable because it depends on the hidden dynamics of the environment. However, the ratio of $p(h_i|\pi)$ to $p(h_i|\pi')$ for any other strategy π' is, because the histories in question are the same and therefore the probability of the realized world dynamics are equal and they cancel out of the ratio.

This estimator is an weighted importance sampling estimate (Rubinstein 1981; Hesterberg 1995). The samples are reweighted according to the ratio of their likelihood under the target strategy to their likelihood under the sampling distribution. Shelton(2001b; 2001a) and Peshkin & Mukherjee(2001) detail more of the theoretical properties of this estimator in the context of reinforcement learning.

Based on this estimator, we build a greedy search algorithm that selects a new strategy by maximizing the estimate $\hat{R}(\pi)$ with respect to the strategy π. This new strategy is

[‡]A history denotes the sequence of observations that the agent made and the actions the agent took in response.

then tried resulting in a new history and return. The strategy, history, and return are all added to the dataset and the process repeats.

Strategy Space

In order to employ the algorithm from above, we must select a parameterized class of strategies. We selected eight input values on which the market-maker can base its decisions.

- The difference between the market-maker's ask price and the best ask price (zero if the market-maker's ask price is the best price).
- The spread between the best bid and ask prices.
- The difference between the market-maker's bid price and the best bid price (zero if the market-maker's bid price is the best price).
- The market-maker's ask size.
- The market-maker's bid size.
- The total volume of sell orders of price less than or equal to the market-maker's ask price.
- the total volume of buy orders of price greater than or equal to the market-maker's bid price.
- The market-maker's inventory.

These values have some overlap with the values used as inputs to the market model, but there is additional information that affects the market that is not available to the market-maker including trend data and the hidden state of the IOHMM.

We give the market-maker the ability on each time step to increase by a unit, decrease by a unit, or keep constant its bid price, ask price, bid size, and ask size. It is forced to maintain both a bid and ask price (with positive size) that are separated by at least $1/16^{th}$ (one unit). This makes a total of $81 = 3^4$ actions. We divide these sets of actions into four groups of three subactions and specify a neural-network for each with eight inputs (from above), five hidden units, and three output units (one for each action). The middle layer is of sigmoid units and the output layer is of softmax functions (to produce a probability distribution over the outputs) and are combined to form a complete action from the set of 81.

Thus a strategy consists of four networks. To calculate an action, the same input is fed into each network. The output of each represents the probability distribution of increasing, decreasing, or keeping constant the corresponding value. The subactions are sampled independently from these output distributions.

Market-Making Experiments

We trained two different market models. One (denoted GE) is based on the data from the stock symbol GE for November 1, 1990. The second (denoted IBM) is based on the data from the stock symbol IBM for the same day. Each training day we broke into six one-hour increments. The market simulation and the order book had no knowledge of the artificial division and kept running as normal across the division boundaries. The market-maker was forced to liquidate its inventory at the end of each segment by interacting with the market using a strict strategy of buying or selling. Thus each day contained six episodes. After a day, the market-maker was allowed to change strategies greedily as above.

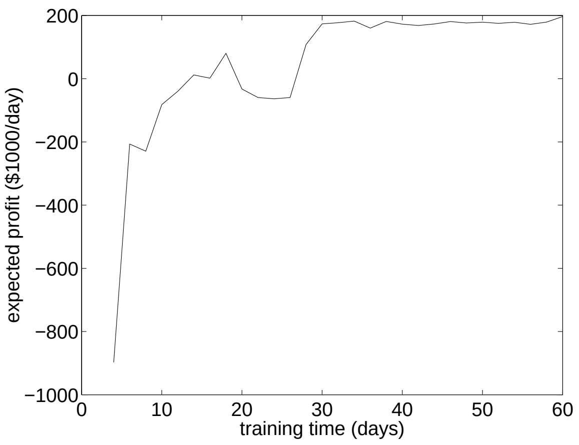

Figure 3: The expected profit for the strategies produced as a function of the number of days of training for the stock GE.

For each strategy tried, we calculated its expected profit by separately (unknown to the learning algorithm) averaging the profits from 10000 full day simulations where we only enforced liquidation at the end of the day. Figure 3 demonstrates a typical learning run for the GE model. The algorithm was run for 60 days and after each day the value of the strategy was calculated and plotted. As we can see, it takes roughly 40 days (or 2 months) of training to converge to a stable profitable strategy. The IBM model is more volitile. However, our algorithm still performs well as demonstrated in figure 4.

Discussion

We believe these results to be encouraging in two respects. First, the market-model represents the true order flow well using the understood statistical model of the IOHMM. This should allow in the future a better understanding of market-making by analysis of the model. Second, the same learning framework which we used for our previous papers also successfully finds profitable solutions in this framework and adapts well to the new circumstances. This implies the learning algorithm is general and has a good chance of working in a real market.

Although there have been previous market models and associated market-making strategies, we feel this is the most flexible model and market-maker. The dynamics of the combination of the market model and the order book are difficult to analyze in closed form. However, by employing learning both at the stage of order-flow modeling and at the stage of finding market-making strategies, we did not need to find a closed-form solution to our problem and we insured that the resulting system is flexible and adaptive.

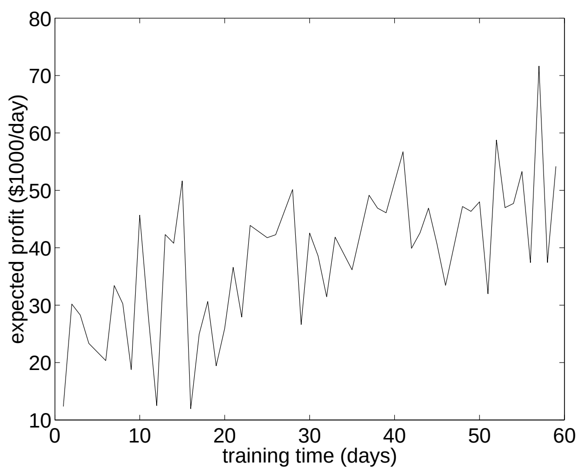

Figure 4: The expected profi t for the strategies produced as a function of the number of days of training for the stock IBM.

References

Amihud, Y., and Mendelson, H. 1980. Dealership market: Market-making with inventory. *Journal of Financial Economics* 8:31–53.

Bengio, Y. 1999. Markovian models for sequential data. *Neural Computing Surveys* 2:129–162.

Chan, N. T., and Shelton, C. 2001. An electronic market-maker. Technical Report AI-MEMO 2001-005, MIT, AI Lab. appeared in the Seventh International Conference of the Society for Computational Economics.

Chan, N. 2001. *Artifical Markets and Intelligent Agents*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Easley, D.; Kiefer, N. M.; and O'Hara, M. 1997. The information content of the trading process. *Journal of Emprical Finance* 4:159–186.

Garman, M. 1976. Market microstructure. *Journal of Financial Economics* 3:257–275.

Glosten, L. R., and Milgrom, P. R. 1985. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics* 14:71–100.

Hassbrouk, J.; Sofi anos, G.; and Sosebee, D. 1993. New York Stock Exchange: Systems and trading procedures. Technical report, NYSE Working Paper.

Hesterberg, T. 1995. Weighted average importance sampling and defensive mixture distributions. *Technometrics* 37(2):185–194.

Ho, T., and Stoll, H. R. 1981. Optimal dealer pricing under transactions and return uncertainty. *Journal of Financial Economics* (9):37–73.

O'Hara, M., and Oldfi eld, G. 1986. The microeconomics of market making. *Journal of Financial and Quantitative Analysis* 21:361–376.

Peshkin, L., and Mukherjee, S. 2001. Bounds on sample size for policy evaluation in markov environments. In *Fourteenth Annual Conference on Computational Learning Theory*.

Rabiner, L. R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77(2):257–286.

Rubinstein, R. Y. 1981. *Simulation and the Monte Carlo Method*. John Wiley & Sons.

Shelton, C. R. 2001a. *Importance Sampling for Reinforcement Learning with Multiple Objectives*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Shelton, C. R. 2001b. Policy improvement for POMDPs using normalized importance sampling. In *Proceedings of the Seventeenth International Conference on Uncertainty in Artificial Intelligence*, 496–503.