Agent-Based Stock Market Simulation

CX4230

Hans Bergren, Jeremy Stephens

Introduction

Our initial goal was to build such an accurate model of the stock market that we could make predictions on certain stocks with an accuracy ‘better than chance.’ After a review of the literature, we realized how difficult a problem we were attempting to solve. Our new focus is to find the best trading strategies through multiple simulation runs of agents operating in an experimental market.

Our potential customers are people who invest in the stock market, specifically people who wish to test their own trading strategies in a simulated environment. Ideally our simulation will be able to test multiple trading strategies to determine which strategies are profitable.

Literature Review

Our agent-based modeling approach is related to at least three distinct literatures: experimental markets, simulated markets, and forecasting techniques.

“Surveying stock market forecasting techniques - Part II: Soft computing methods” was a paper that looked at over 100 related scholarly articles working on stock market forecasting. They found the most popular approach was using some sort of neural network, with over 60 papers implementing a feed forward neural network. This approach makes sense when attempting to find patterns in a dataset as large as the stock market.

However, another paper “Agent-based Models of Financial Markets: A Comparison with Experimental Markets” took a different approach. Instead of trying to forecast the stock market, the goal of this paper was to study attributes of the market itself, such as: price efficiency, trading volume, rate of price convergence to the mean, and dynamics of wealth distribution among agents. This paper implemented a single stock double-auction market, where simulated agents buying and selling were responsible for the price changes in the market. They fully controlled the agents and the marketplace, giving them an advantage over using real-world historical data. This approach allowed the researchers to run a variety of experiments and study multiple attributes of the marketplace.

A third paper we looked at “Forecasting stock market prices: Lessons for forecasters *” was geared towards financial people instead of taking a computer science approach. This paper

analyzed various attributes commonly used to forecast prices such as high Earnings/Price ratio, a low market capitalization, and the January effect. Some rules-based trading techniques were also discussed, such as price reversals and divergence from the average.

We are attempting a different kind of agent-based stock market model. Instead of focusing on the marketplace itself, we will use our experimental market to test different trading strategies. Our model will also allow us to test out how different trading strategies perform on different types of stocks since we can vary the inputs at initialization, something that was not looked at in our third paper which focused on backtesting from 1926 to 1986, and did so from a financial not computer science perspective.

Conceptual Model

We will implement a simulated *double-auction market* where virtual trading agents of various types are able to buy and sell securities (i.e. stock shares). We will simplify the model by listing only one stock in the market, which is initialized with some initial price and total number of shares. We will also restrict the quantity traded to one share at a time and disallow margin trading (borrowing money to trade).

Agents will be allowed to trade during specified *trading intervals* with the interval length as a parameter to the model. At the beginning of each *trading interval*, the agents will be sorted in a uniformly random order, and each agent will place one *limit order* or one *market order*.

In terms of traders, we will be able to vary the distribution of wealth, allowing some agents to be “price movers” and others to be more like normal people who cannot significantly affect the price by themselves. We will also allow a range of trading strategies among agents, and be able to control the distribution of strategies among the agents. Trading agents can buy or sell within their budget constraints. All of the traders act in their own self-interest and seek to maximize profit.

<u>Software Architecture</u>

Parameters

- *D* = distribution of trading agent types
- *E* = initial distribution of wealth
- *S* = initial stock price and total number of shares

Initialization

- Create market containing one stock with initial price and number of shares
- (optional) Input historic data for the stock
- Create all agents according to *D*

Software Objects

- *Stock* - Represents a single stock in the market.
    - price
    - total shares
- *Market* - Double-auction market where agents can place bids and asks.
    - list of stocks
    - historic data for each stock
    - length of trading interval
    - volume of shares traded
    - order queue
- *Trader* - Represents a single actor in the financial market.
    - trading strategy
    - starting amount of money / shares

Stretch Goals

We have several stretch goals for this project given enough time.

- One goal would be to add multiple markets. If we had multiple markets with the same stock, we could simulate arbitrage trading, something that takes place in the real world with currency trading.
- Another goal would be to allow our markets to contain multiple stocks. This would allow agents to diversify their portfolios and experiment with risk strategies.
- A third goal would be to allow agents to do fundamental analysis of stocks (price to earnings ratio, market capitalization) and not just technical analysis (moving price average, standard deviation of price). This would probably be dependant on creating a multi-stock market in our model though, and will be more useful for real-world data than our randomly generated stocks.
- A fourth goal would be to introduce trading latency as a parameter. This would allow us to compare high-frequency traders to longer-term traders better, since latency makes a huge difference in high-frequency trading. Additionally we could vary the latency between agents and see whether it affected their performance when using the same trading strategies
- A fifth goal would be to add short selling stocks as an option for the trading agents. This would probably promote more anti-momentum trading and make our market model more realistic.
- A sixth goal would be to add margin trading as something trading agents could do. This would allow greater risk-taking behavior on the agent's part, and would also make our market model more realistic.
- A final stretch goal would be to add option trading as an action trading agents could take. Option trading is what several real-life traders spend their time doing since it allows traders to manage risk, and make money off of non-price variables like volatility.

Option trading is complicated, so if implemented we would likely simplify the available options.

Implementation Plan

Checkpoint C:

By now we should have a basic model of the stock market. Our goal for this checkpoint is to have a working single stock agent-based market. It is acceptable at this stage to have agents randomly buy and sell and not have an intelligent trading strategy yet. The market should have some 'historical data' that will be randomly generated.

Checkpoint D:

By this checkpoint we should have intelligent agents and stock market visualization. While at the last checkpoint we may have been using print statements to gauge the workings of our simulation, by now we hope to have a simple GUI that will communicate the movement of an individual stock, as well as the profit / loss of an agent and their trading strategy. More intelligent trading strategies should be implemented, possibly including price reversal traders (assuming a reversion to a moving price mean) and momentum traders. Additionally we should be able to specify what trading strategy an agent will use, and allow for a range of wealth distribution among agents.

Checkpoint E:

By this checkpoint we should be able to initialize the market with historical data for a single stock. The GUI should have an option to include a generated experimental market or import data for a specific stock from somewhere like Google Finance. All coding should be done by this checkpoint, and any 'stretch goals' not reached by now will most likely not be implemented.

Checkpoint F:

By this checkpoint several experiments should have been run on both a randomly generated market and a market based on real world data. We will backtest our simulation using real world data, and compare multiple trading strategies by the agents.