Prediction Market Bots Homework

Released: March 23, 2018
Due: April 18, 2018 at 10:00 PM

Introduction

In this project, you will work with a partner to write your own prediction market trading bot! The market you will be participating in is similar to the prediction market 3 that Gabe ran in class. In this market, there is one true coin, and assuming \( n \) total agents, each one is assigned \( 1,2,\ldots,\lceil n/2 \rceil \) decoys (2 agents will have 1 decoy, 2 agents will have 2 decoys, and so on). There will be multiple rounds of trading (say, 100). Before each trading round begins, each agent will see a coin flip and a number of decoys. Then, they will trade contracts that are worth $0 if the true coin is tails, and $100 if the true coin is heads. Each round will end after a pre-specified amount of time, at which point all contracts traded during that round will be settled.

Note: Partners are **required** for this project. If you cannot find a partner, please use the Piazza “Search for Teammates” function to find a partner. If you are still unable to find a partner, email the TAs.

Written Questions

Please complete following questions, and hand in your solutions in the homework handin bins. The due date for these written questions is the same as the rest of the assignment. However, these questions will help you strategize about your bot’s, so we highly recommend starting early.

Updating Beliefs

Solve each of the following questions either analytically or using Monte Carlo. For each question, assume that you’re told heads.

1. Suppose you have one decoy. What is the probability that the true coin is heads?
2. Suppose you have two decoys. What is the probability that the true coin is heads?
3. Generalize the previous problem. That is, if you are given \( n \) decoys, what is the probability that the true coin is heads?
4. Suppose you have one decoy, and you know someone else has two decoys and was told heads. What is the probability that the true coin is heads?
5. Same as the previous question, but you know the other person was told tails. What is the probability that the true coin is heads?
6. (Optional; this question is harder than the others) Generalize the previous two questions. Suppose that you have \( n_i \) decoys, and you know that people with \( n_2, \ldots, n_k \) decoys were all told heads. Additionally, you know that people with \( m_1, \ldots, m_i \) decoys were all told tails. What is the probability that the true coin is heads?

**Market Strategy**

For each of the following scenarios, please write a short response detailing your approach to trading in each of these markets.

1. Imagine your bot is bidding against 5 copies of the following instructor bots. How would you design your bot to compete with these various instructor bots?
   
   a. Every few seconds, each instructor bot randomly chooses a price between 1 and 99, and randomly chooses to buy or sell at that price.
   
   b. Each instructor bot computes a fair value for the true coin based on its initial information. Then, it buys and sells at prices that are ±1 around this value. These instructor bots never update their fair value.
   
   c. Each instructor bot computes its own fair value, and again buys and sells at prices that are ±5 around this value. Each time it's involved in a trade, it then updates its fair value by 1 in the direction of the trade.
   
   d. Each instructor bot computes its own fair value, and again buys and sells at prices that are ±1 around this value. Each time it's involved in a trade, it then updates its fair value by 5 in the direction of the trade.

2. In reality, people are not pure profit-maximizers. Rather, they are risk averse. One way to model this behavior is through risk limits. In particular, a risk limit is a requirement that your bot not be overly exposed to the market. For example, suppose \( P \) is the net number of contracts your bot has bought/sold in the market; then, it obeys a risk limit of \( L \) if \( |P|<L \) at all times. Suppose all bots in the market are forced to obey a risk limit of 5. How would your trading strategy change?

3. Take a closer look at the instructor bot described in question 1 part d. The way the bot is currently implemented, it has a serious flaw. What is this flaw, and how would you take advantage of it? How would you fix it?

At first glance, it's easy to believe that this instructor bot as described is doing something reasonable. It computes an initial fair value, and tries to update its belief based on any trades it makes. However, it has a extremely exploitable flaw. How can you code your bot in a way that hedges against the possibility that you implement incorrect/exploitable logic? How would you do so while not restricting your bot's potential for making many favorable trades?
4. Now suppose your bot is competing in a market with two of each of the bots described in part 1 (excluding bot d). How would you adjust your bot’s strategy so that it performs well in this market? What kind of issues arise when a market’s participants are more diverse?

**Coding Portion**

**Assignment Specs**

Your job in this project is to create your own trading agent. We’ve provided a TemplateAgent file on the Homeworks page of the course website. You can either use this template, or write your own agent that extends AbsPredictionMarketAgent. There are three methods you must override in this abstract class:

- **onMarketStart()**: This method is called when the market is initialized, before trading begins. At this point, your bot is aware of its coin result, as well as the number of decoys it has. Use this method to initialize your initial beliefs, as well as any other variables you’ll need as the market develops.

- **onMarketRequest(CallMarketChannel channel)**: This method gives your bot a chance to submit a buy/sell in the market. This method is called in fixed time intervals (i.e., every 0.5 seconds), and is the only time your bot can buy/sell.

- **onTransaction(int quantity, double price)**: This method is called whenever a your bot is involved in a transaction. quantity is the number of contracts that were exchanged, and price is the price at which those contracts where bought/sold. price is net to you (i.e. positive if you sold, negative if you bought). If you bought and sold in a round, you will receive two calls to onTransaction; likewise if you bought/sold contracts at different prices.

- **getHighestBuy()**: This is a debug method that we may call to observe your bots bidding behavior. Please return the highest price at which your bot is willing to buy at the time the method is called. This method may be used to give your bot partial credit during grading, so make sure to implement it accurately. Remember that any changes to your local copy of the ledger/orderbook will persist into this method, so keep that in mind during implementation.

- **getLowestSell()**: This is a debug method that we may call to observe your bots bidding behavior. Please return the lowest price at which your bot is willing to sell at the time the method is called. This method may be used to give your bot partial credit during grading, so make sure to implement it accurately. Remember that any changes to your local copy of the ledger/orderbook will persist into this method, so keep that in mind during implementation.
In addition to these required methods, we have provided helper methods your bot can use:

- **buy(double price, int quantity, CallMarketChannel channel):** This method submits a buy for a specified quantity of contracts at a specified price. This must be called within onMarketRequest.

- **sell(double price, int quantity, CallMarketChannel channel):** This method submits a sell for a specified quantity of contracts at a specified price. This must be called within onMarketRequest.

- **cancel(double price, boolean buy, CallMarketChannel channel):** This method cancels open orders your bot currently has. Pass in true to cancel buy orders and false otherwise. Cancels all orders above price for buys and below price for sells.

- **getCoin():** This method returns true if your coin is heads, and false if tails.

- **getNumDecoys():** This method returns the number of decoys your bot was assigned.

- **getLedger():** This method returns a ledger of completed transactions. The ledger itself is a List<Transaction> object, and each Transaction has a PRICE and QUANTITY field. Note that the ID in each transaction will be null, unless your bot was involved in the transaction. The ledger is automatically kept up to date for your bot.

- **getOrderBook():** This method returns an OrderBook object, which stores the current outstanding buys and sells. The relevant methods in OrderBook are getBuys() and getSells(), which each return a PriorityQueue of Buy/SellOrder objects, sorted by price. See [here](#) for how to use PriorityQueues

**Testing**

In order to test your bot, you will have to use the Trading Platform. To do this in Eclipse, first import the following jars from `/course/cs1951k/pub/trading_platform`:

1. commons-math3-3.6.1.jar

2. kryonet-2.21-all.jar

3. The most recent trading platform JAR file

   a. **IMPORTANT:** Periodically check to make sure that you are using the most updated JAR file for the trading platform, as this will likely change over the course of this project due to bug fixes and the like.

Once you have your build path all set up, you can test your bot by doing the following:
1. Make a main method in your agent class similar to the main method in the bot template code found on the website. In this main method, you should instantiate an instance of your bot, along with whatever other bots you want to test against.

   a. You should be able to instantiate RandomAgent(...), FixedAgent(...), or UpdateAgent(...), which correspond to the first three tiers we will evaluate you against. These three agents are described in detail in the rubric section of this document.

2. Then make a class for a Server – this should look exactly like the Server template on the website. The class PredictionMarketServer(seconds, numSims, init_delay, lag) takes in a few parameters:

   a. seconds - the number of seconds the trading round will be open for. We will be testing you with 30 second rounds.

   b. numSims - the number of trading rounds to be simulated. We will run 100 simulations when grading, although you should probably run less for your own tests.

   c. init_delay - the initial delay period (in seconds) where bots connect to server.

   d. lag - the time between each tick within each trading session in milliseconds (specifically, each tick ends up being 2*lag ms).

3. To run a test simulation, run your server class, then run your agent class (which should create all the bots you want to create) within the specified delay time.

Bugs
For this project, you will be using the Trading Platform that you have encountered in lab throughout the semester. As I’m sure you are aware, there are still bugs throughout the platform. If you encounter any bugs, please send us an email with detailed documentation of the bug - for each new bug, we will be giving 1 bonus point to whoever reports the bug.

Evaluation / Rubric
Please hand in the following items: a paper copy of the written questions in the homework handin bins, as well as code and a write-up for your bot through the handin script. These will be graded according to the following rubric.
Written Questions (30%)

We will be grading the written questions essentially as a normal homework assignment, although some questions are likely more open-ended than what you would find on a normal homework.

Bot Performance (60%)

We will be benchmarking bot performance depending on how your bot performs against certain tiers, where we run your bot against 5 other bots of a certain type:

1. Tier 1: Random Bot  
a. This bot will just send in random orders between 1 and 99 with no risk limit.
2. Tier 2: Non-Updating Bot  
a. This bot will calculate a fair value based on its private information and will always submit a buy and sell ε away from its fair value. Additionally, it has a risk limit of 5.
3. Tier 3: Updating Bot  
a. This bot will behave the same way as the Tier 2 bot, but it will also update its information after each trade it makes (it will increase fair value after a sell and decrease fair value after a buy).
4. Tier 4: Mystery Bot  
a. ???

You will have access to bots from the first 3 tiers via the trading platform jar, but the 4th tier will not be released. We will be running your bots for 100 30-second rounds for each tier (subject to change).

You will be graded as follows:

- Each tier is worth 15 points:
  - 15 points for beating a tier, where beating is defined as making positive profit
  - 5 points for tying a tier, defined as having a net profit a small amount under 0
    - This is with the exception of the Tier 4 bot, where tying will still give 15 points
  - 0 points for losing a tier, defined as having net profit significantly below 0
- If your bot ties or loses any tier, we will give back partial credit based on a qualitative assessment of the strategies and ideas in your bot.
  - This means it’s important to have a clear and descriptive write-up and readable code, as well as accurate implementation of the bidding limit debug methods in the bot template.
README (10%)

Your README should cover all of the following:

- Strategy
  - A high-level description of the strategies your bot employs
  - The motivation for why you chose to use each of these strategies
  - A quick description of how these strategies are implemented code-wise
  - Any ideas you thought of implementing but couldn’t due to time limitations or something else
- Known bugs or quirks in your bot encountered through testing

Extra Credit

We will have at least 2 forms of extra credit for this project:

- Groups that find new bugs in the platform will be rewarded 1 point for each new bug found.
- The next lab will serve as office hours for this project. One hour into each lab, we will run all the bots in each lab against each other, and the winner will receive 5 extra points on this project. To try to make this more fair, we would like lab numbers to be roughly even, so we will release a first-come first-serve signup for the labs (after Spring Break).