A Decentralised Peer-Prediction Market

CS907 Dissertation Project - Interim Report

Thomas Archbold
1602581

Department of Computer Science
University of Warwick

July 16, 2020

1 Introduction

Prediction markets are exchange-traded markets which allow users to trade on the outcomes of future events as opposed to traditional financial instruments.

2 Background

3 Current Progress

The project is at a point where most of the core functionalities behind a prediction market are implemented. Users are able to create markets by specifying a wager, such as "Arsenal will beat Tottenham", and a deadline by which the outcome will be known. All users are then able to trade in these markets in an attempt to make a profit; buying shares pushes the price up, while selling them does the opposite. Profits can be made in the markets in the traditional way, by buying low and selling high (before the deadline has passed), or by holding onto the positions up to the deadline and receiving a payout based on the outcome of the wager.

A key aspect behind any prediction market is that of outcome determination. In traditional prediction markets this is done by a central authority who will decide whether or not a wager had a positive outcome. Typically the system then pays out \$1 for each share held if the outcome was positive, and \$0 otherwise. The market we implement is *decentralised* in the sense that we do not rely on a central figure to verify market outcomes; instead, we rely on user reports to determine whether or not the events occur. In our system these users, known as arbiters, are able to offer a report on what they have observed to be the outcome of the market (their *signal*), a "yes" or a "no", and they receive a reward for doing so, to incentivise participation and more accurate outcome determination.

The outcome of a market and thus the payout per share held is set to the fraction of arbiters who reported a positive outcome of the event. This means that, in an ideal setting with no signal noise,

the event the wager describes indeed took place once their deadlines have passed. Since this system is a decentralised market, we rely on the reports of certain users, known as arbiters, to determine a market's outcome. Currently, users can offer to report what they believe the outcome to be, either a "yes" or a "no", and they will possibly receive a reward for doing so, based on the closing price of the market and how the other arbiters reported. Finally, users are then paid out for any shares they hold in the market, and similarly anyone with a short position must buy back their shares. The details of this will be covered in Section 3.2.

3.1 Tools

The project is implemented in Common Lisp and the code has been developed and tested within the Steel Bank Common Lisp (SBCL) compiler and runtime system. Writing the web application itself has required the use of several libraries available from the library manager Quicklisp, specifically:

- Hunchentoot
- CL-WHO
- Mito
- SXQL
- Parenscript
- Smackjack

Hunchentoot provides the environment on which we host the server. Most importantly for our uses it provides automatic session handling, allowing for multiple users to be logged in at once, and easy access of GET and POST parameters, meaning users may interact with the website and send data to the server via HTML forms.

Hunchentoot does not provide any functionality to actually specify the webpages; for this we use CL-WHO. This library simply converts Lisp statements into valid HTML. A benefit of using a fully-fledged programming language to generate the markup, as opposed to simply specifying the HTML explicitly, is that we are able to mix code that defines the structure of the data and code that processes the data itself into one clean interface. This is particularly apparent when making use of Lisp's powerful macro system.

Mito and SXQL provide functionality to connect to and interact with a Relational Database Management System (RDBMS), and in particular we use MySQL for persistent storage. For the most part this provides a quick and intuitive means of interacting with the database, though as we will detail in Section 7, this library leads to some deficiencies in code efficiency and elegance.

Parenscript is used to incorporate Javascript into the site, with the main goal of improving the user experience. Currently it is only used to ensure that all necessary fields during market creation, trading, and market resolution are non-empty to avoid errors. This avoids sending erroneous data to the server, only to be redirected to the page. An important aspect behind any exchange market is responsiveness: all information displayed to the user must be current, to ensure users are interacting with the correct and up-to-date state of the system. As such we plan on making more use of Parenscript along with the AJAX library for Lisp, Smackjack, to allow relevant information to be fetched and displayed to the user asynchronously.

3.2 Design

We implement the mechanism for peer prediction proposed by Freeman, Pennock, and Lahaie [1]. In this section we will introduce the associated terminology and give an overview for the mechanism as a whole.

We are interested in setting up a prediction market for the outcome of the binary event (random variable) $X \in \{0, 1\}$. The terms "market" and "security" are used interchangeably to refer to the entity comprising a wager (e.g. "Arsenal will beat Tottenham") and a deadline that represents the event X. The mechanism is divided into two main stages in which users may interact with the system: the market stage, where users may buy and sell shares in various securities whose deadlines have not yet passed; and the arbitration stage, where a subset of the users report on the outcome of the security and the payout price per share is computed. Since we rely heavily on user participation for the mechanism to run correctly, it is important that users act in the desired manner. Using this mechanism, whose details we will expand upon in the coming sections, allows us to achieve certain theoretical guarantees on user behaviour, and most importantly it incentivises truthful reporting in the arbitration stage.

3.3 Market Stage

The market stage is concerned with creating the markets for all manner of events that users wish to bet on, and then trading shares in these markets. The share price $p \in [0, 1]$ is calculated using a *market scoring rule (MSR)*, which is a means of assigning a probability that the outcome of the event is positive. If we denote by q the total number of shares in circulation for a given market (which increases when shares are bought and decreases when shares are sold), then an agent wishing to buy a quantity of $q' - q$ shares (i.e. taking the new total number of shares to q') will pay $C(q') - C(q)$, for our choice of convex, differentiable, monotonically increasing cost function C. We tune the behaviour of the cost function through the use of a *liquidity parameter* b such that $C_b(q) := b \cdot C(q/b)$, which controls the price responsiveness of C. A lower value of b means that the share price changes more quickly for fewer shares bought, and vice versa.

As in any market buying shares pushes the price up, in our case towards \$1, while selling pushes it towards \$0. In order to raise the money to pay stakeholders when the event's outcome is realised, and reward arbiters for reporting on the outcome, we use trading fees. For a "buy" transaction that pushes the share price to p, a user is charged fp for some fixed parameter f, while a "sell" transaction would incur a total payment of $f(1 - p)$. Transactions are only charged a fee when the user is increasing their risk: if a user is simply liquidating their position (selling owned shares or buying back shares sold short), then no fee is charged.

Users may trade in the market as long as they have enough funds to make the transaction and the deadline for the event has not passed. After the deadline, supplied by the user when creating the market, has expired the users' positions are final and we move on to determining the outcome of the event in the arbitration stage.

3.3.1 Arbitration Stage

The arbitration stage is concerned with determining the perceived outcome of event X by taking reports from a subset of the users in the system, known as the arbiters. Each arbiter i receives a (private) signal

$x_i \in \{0,1\}$ that tells them the result of the event – this is analogous to reading the news, watching the match, or even hearing about it from a friend. They then submit a report $\hat{x}_i \in \{0,1\}$ to the system that tells us what they believe the outcome to be. Note that since the signal they receive is private, we have no way to know whether the user is reporting what they truly believe, or they are trying to manipulate the system for their own gain. Arbiters are then paired randomly, and paired arbiters i and j are paid a reward of $u(\hat{x}_i, \hat{x}_j)$ according to the "1/prior with midpoint" mechanism, which we will detail below.

We can now determine the outcome of the market. In a traditional prediction market, users are paid \$1 for each share owned (or they must buy shares back at \$1 if they have shorted the security) if the event has a positive outcome, and \$0 otherwise. Hence if an event is almost certain to occur, more users will buy shares than sell shares, as they expect they will be paid out for each share they own when the outcome is realised, pushing the share price towards \$1. Similarly, if an event is believed to be unlikely, more users will sell shares than buy them and the price will approach \$0. This prevents arbitrarily large profits being made for risk-free bets.

In our implementation, the outcome of the event X is the random variable $\hat{X} \in [0,1]$ and is set to the proportion of arbiters that submit a report of $\hat{x}_i = 1$.

3.3.2 1/prior mechanism

The 1/prior payment mechanism was conceived by Jurca and Faltings [2, 3] as a means of rewarding arbiters for participation in opinion polls, in particular giving rewards for output agreement, and Witkowski [4] generalises this mechanism to pay out different amounts depending on the signals reported by paired arbiters. We apply the modification proposed by Freeman et al. [1] to ensure that arbiters are incentivised to act truthfully when reporting on market outcomes.

After the event's outcome has been realised, each arbiter will have received a private signal $x_i \in \{0,1\}$ which they must report to the system in order to determine the market's outcome. Arbiters are paired randomly and paid according to the 1/prior payment rule. If agents i and j with reports $\hat{x}_i, \hat{x}_j$ respectively are paired, then they are paid:

$$u(\hat{x}_i, \hat{x}_j) = \begin{cases} k\mu & \text{if } \hat{x}_i = \hat{x}_j = 0 \\ k(1-\mu) & \text{if } \hat{x}_i = \hat{x}_j = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

in which k is a parameter and μ is the (common) prior belief that $X = 1$, as a reward for participating in the mechanism for crowdsourced outcome determination.

During trading of a market for event X, users will have bought and sold various quantities of shares in the security. By the time the market closes, if more shares have been bought than sold then the final price of the security will be closer to \$1, and if more shares have been sold short than bought then the price will be closer to \$0. The closing price hence gives us a way to measure measure how likely the userbase feels that the event will have a positive outcome: in other words, we have a proxy for $Pr[X = 1]$ that is equal to the closing price of the market. Given that the closing price of a security represents how strongly users feel about the expected outcome of the event, we can use this as the value of μ.

Freeman et al. show how to modify Equation 1 to ensure that arbiters act truthfully – that is, arbiters report the signal they received. This yields the "1/prior with midpoint" payment mechanism. For common prior μ, the probability that a user receives a positive signal, let μ_1 (respectively μ_0) be the probability that, given that agent i receives a positive (respectively negative) signal, another randomly chosen agent receives a positive signal. The modified mechanism is then simply the same as Equation 1, with μ replaced by $(\mu_1 + \mu_0)/2$.

3.4 Implementation

In its current state we have so far implemented the correct means of calculating most relevant figures quoted to the user, such as pricing information, rewards for reporting an outcome, and trading fees. However, many of these depend on parameters that we are yet to tune, meaning we do not yet achieve the theoretical guarantees of the mechanism. The main parameters we are yet to tune are the trading fee f and the parameter k, used in the 1/prior mechanism.

3.4.1 Overview

The system is implemented several largely independent files, which each tackle a specific area of functionality:

- `arbitration.lisp`
- `database.lisp`
- `market.lisp`
- `msr.lisp`
- `server.lisp`

In `arbitration.lisp` we implement the 1/prior-with-midpoint mechanism and handle the calculation of the probabilities μ_1 and μ_0. We perform the latter computation with insights from Witkowski [4], meaning we can derive μ_1 and μ_0 using only μ and each users' estimations of $Pr[x_i = 1|X = 1]$ and $Pr[x_i = 1|X = 0]$.

In `database.lisp` we define all the tables we use as well as the interface with which we interact with it.

In `market.lisp` and `msr.lisp` we provide the interface for all functionality related to creating and trading in user-created markets. The former implements the calculation of trading fees, differentiating between transactions that increase risk and liquidation transactions, which incur no extra payment. The latter implements the market scoring rule that we consult to calculate how transactions change a security's share price. In particular, we implement the Logarithmic Market Scoring Rule, which we detail in 3.4.4.

Finally, `server.lisp` incorporates all the separate functionality into one file and is responsible for setting up and handling requests from the Hunchentoot server. The remaining files, `ajax.js` and `javascript.lisp`, implement very little so far, though will be expanded upon in the near future.

3.4.2 Server

3.4.3 Database

3.4.4 Logarithmic Market Scoring Rule

3.4.5 Arbitration

4 Ethics

5 Project Management

6 Next Steps

7 Evaluation

References

[1] R. Freeman, S. Lahaie, and D. M. Pennock, *Crowdsourced Outcome Determination in Prediction Markets*, arXiv:1612.04885 [cs], (2016). arXiv: 1612.04885.

[2] R. Jurca and B. Faltings, *Incentives for expressing opinions in online polls*, in Proceedings of the 9th ACM conference on Electronic commerce - EC '08, Chicago, Il, USA, 2008, ACM Press, pp. 119–128.

[3] ——, *Incentives for Answering Hypothetical Questions*, in Workshop on Social Computing and User Generated Content - EC '11, 2011, p. 9.

[4] J. Witkowski, *Robust Peer Prediction Mechanisms*, PhD thesis, 2014.