Algorithms Seminar 12.02.2014

Prediction Markets

Tawatchai Siripanya *Wolfgang Mulzer, Yannik Stein*

1 Introduction

1.1 What is a Prediction Market?

Prediction markets are defined as analytical markets that aim to create predictions for a future event; e.g., in 2008, *intrade.com* provided bets on whether avian flu will hit the US [1]. The outcome can be either true or false, and the participants or traders in prediction markets can be anonymous. The difference compared to a financial security market is that the consumption of security value has no affect on buyers (traders)—buyers (traders) only want to buy a security in order to make a profit. The outcome of a combinatorial space for betting on n horses in a horse race might be $n!$. A bet can be; e.g., "horse A beats horse B". In most cases, a bet becomes intractable and results in high computation costs [4]. This cost is referred to as "the auctioneer's matching problem" (Section 2.2).

The growth of prediction markets was inspired by the development of the Logarithmic Market Scoring Rule (LMSR) by Hanson [3] (which serves as a pricing mechanism called "automated market-maker", see Section 3). The automated market-maker is used to determine and post prices for all possible states. This allows the market organizer (or auctioneer) to promptly post prices for all states rather than to expect traders to post orders like in a continuous double auction.

In 2006, an additional mechanism called the share-ratio Dynamic Pari-mutuel Market-Maker (DPM) was built by Pennock et al. [4]. This mechanism also serves as an automated market-maker with controlled risk to the auctioneer. A prediction market that uses this mechanism is the *Yahoo! Tech Buzz Game*, whereas other online prediction markets; e.g., the *TheWSX* and *InklingMarkets* use LMSR [5].

Prediction Market Definition: Pennock et al. [4] state that "a prediction market is one mechanism designed to solve the information aggregation problem". The prediction market involves the aggregator (an individual) and the informants (a number of individuals). In this case, the aggregator creates a financial security and invites the informants to trade said security.

Definition 1. *Given Ω represents a set of all possible* ***states of the world****, $\omega \in \Omega$ represents the state of the world that happened exactly once at a certain point in time, an* **agent** *i may have partial information about the true state of Ω, however it is unnecessary to know its true state. We represent agent $i's$ information using a partition π_i of Ω. In other words, a collection $\{\pi_{i1}, \pi_{i2}, ..., \pi_{ik}\}$ is a subset of Ω; i.e., the different subsets are disjoint and their union yields Ω. The agent i knows in which subset of its partition the true state exists, however he does not know which member is the true state. Given n agents $1, 2, ..., n$, their combined information $\widehat{\pi}$ is the coarsest common refinement of the partition $\{\pi_1, \pi_2, ..., \pi_n\}$ [4].*

Figure 1 illustrates an example of eight states of Ω $(\omega_1, \omega_2, ..., \omega_8)$. The events X_1, X_2, and X_3 represent subsets of states (ω). The $1's$ partition π_1 is $\{\{\omega_1, \omega_2, \omega_3, \omega_4\}, \{\omega_5, \omega_6, \omega_7, \omega_8\}\}$. The

coarsest common refinement $\widehat{\pi} = \Omega$. We write $\omega_{X1\overline{X2}X3}$ to indicate that the states X_1, X_3 are true when the state X_2 is false.

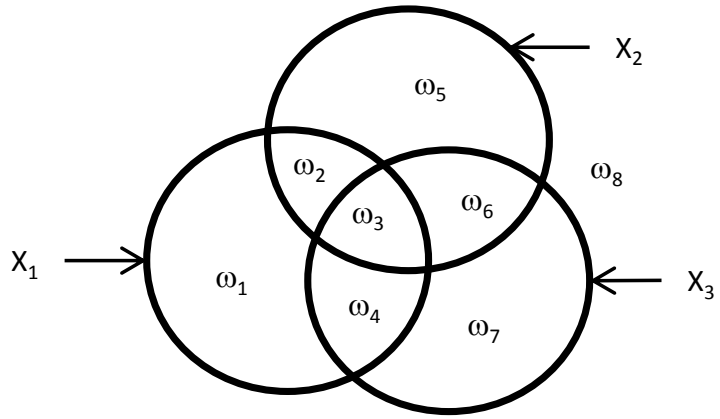

Figure 1: An overview of Partition model of knowledge. Source: [4]

2 Combinatorial Prediction Markets

This section aims to report the complexity of operating combinatorial markets. Suppose ε is some finite set of base event and these events are linearly independent—the value (*true* or *false*) cannot be determined with certainty; e.g., the proposition "the price of an apple is greater than 1 Euro" might be true or false in the future. Then, the size of the state space Ω is $2^{|\varepsilon|}$. We use the symbols X_1, X_2, X_3, ... to represent the individual Boolean event in ε.

2.1 Settings

Securities: We use ϕ and ψ to represent arbitrary boolean formulas– the securities are based on the Boolean formulas (ϕ and ψ) over the set of a proposition (e.g., "the price of apple is greater than 1 Euro"). We write $S_{\phi|\psi}$ to indicate that the owner of a security S is paid \$1 if both ϕ and ψ are true, and paid \$0 if ψ is true but ϕ is false. The security is cancelled (the owner gets all money back) iif ψ is false.

Orders: We use the letter "o" to represent an *order*. An order is defined in the form e.g., $o :=$ "q units of $S_{\phi|\psi}$ at price p per unit". We call a *buy order* if $q > 0$, and a *sell order* if $q < 0$. Furthermore, orders can be *indivisible* (must be accepted or rejected in full) or *divisible* (can be accepted partially) [1]. The divisible order is referred as an agent that accepts the quantity αq, $\forall q \in (0; 1]$, whereas the indivisible order is referenced to as the agent will only accept exactly q units or none. Additionally, every order o can be translated into a pay off vector γ. The payoff γ^ω (in state ω) is $q * 1_{\omega\in\psi}(1_{\omega\in\phi} - P)$, where $1_{\omega\in E}$ is the indicator function equaling 1 iff $\omega \in E$ and zero otherwise. We write $O = \{o_i\}$ to represent the set of all orders and $P = \{\gamma_i\}$ to represent the

set of corresponding payoff vector. We (traders) place orders that specify the security, number of shares to buy/sell, and the maximum/minimum price at which we are willing to buy/sell.

2.2 The auctioneer's matching problem

Prediction markets provide traders with a feature to trade without risk. The auctioneer's matching problem lies in the determination of which orders to accept without any risk. Chen et al. [1] formulates that the auctioneer's matching problem as an integer programming problem for indivisible orders and a linear programming problem for divisible orders. The auctioneer's matching problem is NP-hard [1, 4] for both orders. We can solve the auctioneer's matching problem for divisible orders in polynomial time using *ellipsoid algorithm* [1]. Fortnow et al. [2] show that we can compute the matching problem for divisible orders in polynomial time when there are $O(log\ m)$ events, but is co-NP-complete for $O(m)$ events, where m is the length of the description of all orders. On the other hand, the matching problem for the indivisible orders is non trivial: it is NP-complete for $O(log\ m)$ events and $\sum_2^p$-complete for $O(m)$ events.

3 Automated Market Makers

This section describes the mechanisms used to limit the exposure of the market maker as well as to encourage informed traders to trade. It provides *automated market maker* algorithms: a market scoring rules market maker and a dynamic-pari-mutuel market maker. This in contrast to a traditional market that uses double auction or order books mechanisms, waiting for buyers/sellers to arrive to trade in the market. Market makers on the other hand are agents that are ready to trade without hesitation and provide high liquidity when the number of traders is low (market is unpopular). The automated market makers are used to determine how the contracts are priced using a "cost potential function C". Pennock et al. [4] suggests three desirable properties that automated market makers should have as follows:

- It should run a predictable or bounded loss
- Informed traders should have an incentive to trade whenever their information would change the price
- After any trade, computing the new prices should be a tractable problem

The prediction market generates forecasts for a binary event with an outcome space $\Omega = \{Yes, No\}$. Many real-world prediction markets focus on such binary events; e.g., "whether the price of an apple will be less than 1 Euro in 2015", "whether people will stop smoking in 2050", and "whether cars will be sold more than bicycles in 2020".

3.1 Market scoring rules market maker

The prediction market operates using a logarithmic market scoring rule (LMSR) to determine the price in the market. The payments are determined by a cost function $C(\overrightarrow{q})$. The function represent how much money has been collected within the market. Suppose the market contains $|\Omega|$ mutually exclusive and exhaustive contracts (securities or shares). Let q_i be the current number of contracts(securities or shares) for outcome i that have been purchased. The current cost of

purchasing x shares of the outcome i is $C(q_1, ..., q_i + x \ ..., q_{|\Omega|}) - C(q_1, ..., q_i, ..., q_{|\Omega|})$ dollars; i.e., $C(\overrightarrow{q}_{new}) - C(\overrightarrow{q}_{old})$.

The corresponding cost function $C(\overrightarrow{q})$ of the LMSR is determined by

$$C(\overrightarrow{q}) = b \ ln \left(\sum_j e^{q_j/b} \right)$$

and the price function is $\partial C/\partial q_j = e^{q_j/b} / \sum_k e^{q_k/b}$, where b is the free parameter used to control for a market maker's loss risk and the liquidity of the market. The LMSR pays a fixed \$1 per share to winning shareholders.

3.2 Dynamic-pari-mutuel market maker

Unlike a Market scoring rules market maker, a dynamic-pari-mutuel market maker (DPM) pays an equal portion of the total amount wagered to winning shareholders [4]. In a pari-mutuel market traders place wagers on which of two or more mutually exclusive and exhaustive outcomes will occur at some time in the future. After the true outcome becomes known, all the money of traders with the incorrect outcome is redistributed to the traders with the correct outcome, in direct proportion to the amount they wagered. The market maker uses the *share-ratio cost function* as the mechanism that sets the price and the securities. It is defined by:

$$C(\overrightarrow{q}) = k \sqrt{\sum_j q_j^2}$$

, where k is a free parameter and the corresponding price function is $p_j = kq_j / \sqrt{\sum_k q_k^2}$.

4 Distributed Computation through Markets

This section aims to present a simple market model and analyse its computational properties; e.g., what can it compute and how fast it can run. We first introduce a Boolean market model to compute with a single security. Then, we give a short introduction on how bid and price can be adjusted. Finally, we describe how many number of rounds it will take for market convergence.

4.1 Boolean Market Model:

Boolean network (BN) models have been used for modelling networks in which the node activity is described by one of two states, 1 or 0. The edges of the network affect the rules that determine the state transitions of the nodes. BN modelling allows the exploration of the dynamics of relevant nodes and the prediction of their future states, as well as exploring the overall dynamics of the network. This is especially useful for large networks like prediction markets where analysing the global behaviour of the system and tracking the individual nodes is computationally intensive. The BN approach has already been used to model a variety of real or artificial networks including, artificial life, scale-free networks, genetic regulatory networks, and strongly disordered systems that are common in physics, biology and neural networks.

Suppose we have n number of traders, the state space is $\Omega = \{0, 1\}^n$, each trader has a single bit x_i (also called input bit) of private information, and each agent i has a partition

$\pi_i = \{\{x \in \Omega | x_i = 0\}, \{x \in \Omega | x_i = 1\}\}$, where x denotes the vector $(x_1, ..., x_n)$. We can use the Boolean function $f : \{0,1\}^n \to \{0,1\}$ to gain the value of combined information x. We setup a security F that will pay \$1 if $f(x) = 1$ and \$0 otherwise. If the market is truly efficient, we expect its equilibrium trading price to equal $f(x)$.

Bid Format and Price Formulation: In a market, we have an *agent i*, a *bid* b_i and a *quantity* q_i; i.e, i has a security of number of q_i and b_i amount of money available in the market. The clearing price is $p = \sum_i b_i / \sum_i q_i$ at each round of trade. At the end of the round, agent i holds a quantity $q_i^{'}$ that is proportional to the money that has been bid ($q_i^{'} = b_i/p$).

Example 1. *Suppose a market m with two agents with private bits* x_1, x_2*. Each of four possible values of x is* 1/4*. The security F is based on the* OR *function* $f(x) = x_1 \vee x_2$*. Assume that agent 1 observed* $x_1 = 0$*, then,* $P((x_1, x_2) = (0,0)) = P((x_1, x_2) = (0,1)) = 1/2$*. For this reason, the agent 1's initial expectation of the value of F is 0,5, therefore we would bid* $b_1 = 0.5$ *in the first round. Suppose* $x_2 = 1$*, then the posterior belief yields* $P((x_1, x_2) = (0,1)) = P((x_1, x_2) = (1,1)) = 1/2$*.*

4.2 Convergence Time:

Let p^{∞} represents the clearing price at equilibrium and F denote the security. $f(x)$ is a Boolean function of the combined information x.

Definition 2. *A function* $f : \{0,1\}^n \to \{0,1\}$ *is a "weighted threshold function" iff there are real constant* $\omega_0, \omega_1, \omega_2, ..., \omega_n$*; i.e.,*

$$f(x) = 1 \ iff \ \omega_0 + \sum_{i=1}^{n} \omega_i x_i \geq 1$$

Pennock et al. [4] introduces Theorems 3,4, and 5 to analyse the number of rounds taken in the market to be converged (without prove).

Theorem 3. *If f is a weighted threshold function, then for any prior probability distribution P, the equilibrium price of F is equal to* $f(x)$*.*

Theorem 4. *Let f be a weighted threshold function with n inputs, and let P be an arbitrary probability distribution. Then, after at most n rounds of trading, the price reaches its equilibrium values* $p^{\infty} = f(x)$*.*

Theorem 5. *There is a function* C_n *with* $2n$ *inputs and a prior distribution* P_n *such that, in the worst case, the market takes n rounds to reveal the value of* $C_n(.)$*.*

5 Summary

We have introduced a model of prediction markets with aggregate uncertainty, by characterizing the uncertainty of market participant's private information. We described prediction market mechanisms which provide traders with the possibility to trade without risk. In most cases, the computation of combinatorial space in prediction markets is intractable and causes high computation cost, which introduces the so called auctioneer's matching problem (NP-hard). Prediction markets use the automated market to determine how the contracts (securities or shares) are priced using a "cost potential function C". Three desirable properties that automated market makers should have are the following: (1) it should run a predictable or bounded loss, (2) informed traders should have an incentive to trade whenever their information would change the price, and (3) after any trade, computing the new prices should be a tractable problem. We described two mechanisms of adjusting prices in the market using cost functions: a logarithmic market scoring rule (LMSR) and a dynamic pari-mutuel market-maker (DPM). The first one pays a fixed amount per share, whereas the second one pays an equal portion of the total amount wagered to winning shareholders. Finally, we described how prediction market can be computed with a single security using a Boolean market model. At n rounds of trading, the price converges at equilibrium $p^{\infty} = f(x)$.

References

[1] Yiling Chen, Lance Fortnow, Evdokia Nikolova, and David M. Pennock. Combinatorial betting. *SIGecom Exch.*, 7(1):61–64, December 2007.

[2] Lance Fortnow, Joe Kilian, David M. Pennock, and Michael P. Wellman. Betting boolean-style: A framework for trading in securities based on logical formulas. *Decis. Support Syst.*, 39(1):87–104, March 2005.

[3] Robin Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, January 2003.

[4] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[5] Mark Peters, Anthony Man-Cho So, and Yinyu Ye. Pari-mutuel markets: Mechanisms and performance. In *Proceedings of the 3rd International Conference on Internet and Network Economics*, WINE'07, pages 82–95, Berlin, Heidelberg, 2007. Springer-Verlag.