Prediction Markets: Connections with Proper Scoring Rules
Game Theory Mini-project Report (May 2012)

Anirban Laha
anirbanl@csa.iisc.ernet.in

Palash Dey
palash@csa.iisc.ernet.in

Sambaran Bandyopadhyay
sambaran@csa.iisc.ernet.in

Suvam Mukherjee
suvam@csa.iisc.ernet.in

Rohith D Vallam (Mentor)
rohithdv@csa.iisc.ernet.in

Dinesh Garg (Mentor)
dgarg77@yahoo.com

Ratul Ray (Mentor)
ratulray2006@yahoo.com

Prof. Y. Narahari (Course Instructor)
hari@csa.iisc.ernet.in

Abstract
Prediction Markets are markets created for aggregating and eliciting information from diverse and frequently self-interested sources and making prediction about future events. We investigate a particular class of scoring rule called pseudo spherical scoring rule and study various issues regarding its implementation. We use the Market Maker Equivalence theorem proposed in Chen and Pennock([1]) to reduce the spherical scoring rule to its corresponding cost function and analyze different aspects of the process. We use an open source framework called Zocalo to experiment a prediction market. We also make the Zocalo framework more flexible to incorporate other cost functions easily. We develop a real-world prediction market predicting the outcome of Indian Premier League (IPL) 2012 cricket tournament and study the market trading activities of the students of IISc over a two week period. We use the real-data obtained from the test bed prediction market to study various theoretical and experimental properties of proper scoring rules like logarithmic, quadratic and spherical scoring rules.

1 INTRODUCTION

1.1 What is Prediction Market?
Prediction markets are an exciting research area of economics, game theory and mechanism design. The roots of Prediction markets originate at the well known Hayek Hypothesis [2].

A prediction market is a market created for the purpose of obtaining a belief probability distribution, based on the information of multiple agents. Prediction markets deal with knowledge integration and price discovery. It is basically concerned with eliciting and aggregation of information from diverse and frequently self-interested sources. It is a market designed primarily for price discovery. For example, the market operator may be happy to pay for the information it seeks, instead of enforcing neutral or positive revenue. They are also used to make prediction about future events. A common approach for the prediction of a particular event (say, whether Team X in IPL 2012 will win today’s match) is to create a security that will pay out some predetermined amount (say Re.1) if the event happens, and agents trade this security until a stable price emerges. The price can then be interpreted as the consensus probability that the event will happen. We will use a Market Scoring Rule (MSR) to determine the amount payed by a trader to change the current probability to a different value. A good prediction market ensures that agents are rewarded for contributing useful and accurate information. In a market based on securities, this corresponds to the assumption that an agent will buy if the price is below her current belief probability or belief probability (which takes what happened in the market so far into account), and will sell if the price is above her current belief probability.

1.2 Example of a Prediction Market
This is an example of a simple prediction market to illustrate the whole scenario. Let us say we want to predict the outcome about who will win today’s IPL match between Team X vs. Team Y. Particularly we create an event called “Team X will win today’s match”. A prediction market can trade a share for this event which pays Re.1 if the event comes true. Let W represent the event that Team X wins today. Now, the expected payoff of a share of ‘Team X’s security’ is

\[ p = Pr(W) \times 1 + [1 - Pr(W)] \times 0 \]

where p is the price of Team X’s security. For instance, if current price of the security is Re. 0.6, it means that market traders believe that, with probability 0.6, Team X will beat Team Y in the match. This probability will become consensus among the market participants. If some market traders possess crucial information that leads them to believe that Team X only has half chances to win, they will sell their security holdings at the current price, which in turn drives down the price.

1.3 Motivation and advantages of Prediction Market
There are many advantages of prediction markets (PM) over other approaches of information aggregation.
Statistical methods for forecasting mainly depends on historical data only, whereas PM can incorporate real-time information.

PM is less expensive in compared to 'Expert Opinions' method, and also PM does not suffer from the problem of avoiding issues like conflicting opinions among experts over a particular prediction.

It is well structured than simple polls.

### 1.4 Some Popular Prediction Markets

Some well known prediction markets are summarized below:

- Iowa Electronic Market, run by the University of Iowa. It has run markets based on many US presidential elections.

- Web based prediction markets like Hollywood Stock Exchange where users can use virtual currency to speculate on movie-related questions like opening weekend performance, revenue earned and Oscar winners.

- More recently, private organizations have employed prediction markets as a business forecasting tool. For example, Goldman Sachs and Deutsche Bank have launched markets on the likely outcome of future readings of economic statistics like retail sales, industrial production etc.

### 1.5 Some Problems regarding Prediction Market

Following are some problems of prediction market:

- It often suffer from the problem of thin market. When a market fails to attract sufficient traders, it may not succeed in the purpose of price discovery.

- To trade in a conventional continuous double auction (CDA) setting, traders must co-ordinate on the assets they will trade. For example, in the Team X-Team Y match, a person places an order to buy shares of Team X. For the trade to be successfully completed, there must be an corresponding sell order from another trader to sell Team X shares. In combinatorial markets, this problem is further exacerbated by dividing the traders’ attention among an exponential number of outcomes, thus making the likelihood of agreeable bilateral trade even more remote.

An automated market maker can address this issue of thin markets. It can improve the liquidity of the market by continually announcing prices offering both to buy and sell shares of all outcomes. The prices are adjusted according to trader demand.

### 2 Relevant Work

A recent research direction in prediction markets is in the area of design of automated market makers. Chen and Pennock discuss a utility function based framework for bounded-loss market makers in [1]. In this paper, they define important concepts like scoring rules, cost functions, utility based functions and their relationships. An important result in the paper is the Market Maker Equivalence Theorem. Conitzer [3] discusses the basic relationship between mechanism Design framework and prediction markets. Mathematical analysis of different types of scoring rules and their various properties are studied by Gneiting and Raftery [4].

### 3 Basic Definitions and Concepts

#### 3.1 Scoring Rule

Let us consider a random variable \( X \). It takes value from the set of outcomes \( \Omega \) where \( \Omega = \{1,2,...,m\} \). Let us consider \( P \) as the set of all probability distributions over \( \Omega \). So, \( P = \{ p \in \mathbb{R}^m : 0 < p_i < 1, \sum_{i=1}^m p_i = 1 \} \)

For a trader, suppose his Reported Probability distribution : \( \hat{p} \) and his True Probability distribution : \( p \)

A scoring rule is a function \( s : P \times \Omega \rightarrow \mathbb{R}^m \). For each report \( \hat{p} \in P \) and each outcome \( i \in \Omega \), it specifies a payment \( s(\hat{p}, i) \). Random variable is realized.

Suppose the current probability estimate in MSR is \( \hat{p} \). A trader who has probability estimate \( \hat{p}' \) changes the market probabilities from \( \hat{p} \) to \( p' \) and gets profit \( s_i(\hat{p}) - s_i(\hat{p}) \)

With a simple scoring rule, a person reports a probability for each outcome, and gets paid depending on that report and the actual event. Market scoring rules are scoring rules where anyone can change the current report, and be paid according to their new report, as long as he agrees to pay the last person reporting according to that person’s report. A proper scoring rule is a scoring rule that motivates truthful reporting.

Examples of some common scoring rules are Quadratic scoring rule, Logarithmic scoring rule etc [1].

#### 3.2 Pseudo Spherical Scoring Rule

The scoring function of weighted pseudo-spherical scoring rules class is

\[
s_i(\vec{r}) = a_i + \frac{b}{\beta - 1} \left[ \frac{1}{\sum_{j=1}^N \left( \frac{r_j}{\pi_j} \right)^{\beta/2}} \right]^{-\frac{\beta-1}{\beta}} - 1
\]

where \( b > 0 \) and the reported probability estimate \( \vec{r} \) is weighted by a baseline probability estimate \( \vec{\pi} \). When \( \beta = 2 \) and \( \vec{\pi} \) is uniform, we get the spherical scoring rule. Hence a spherical scoring rule is as follows:

\[
s_i(\vec{r}) = a_i + b \left[ \frac{r_i \sqrt{N}}{(\sum_{i=1}^n r_i^2)^{1/2}} - 1 \right]
\]
The class pseudo-spherical scoring rule satisfies many desirable properties of scoring rule including the followings:

- **Strictly proper scoring rule**: This ensures that it is always the best response for the traders to report their true probability estimates for the events.
- **Bounded loss**: The market maker’s worst case loss is bounded by some constant.

### 3.3 Utility Function

Suppose $X$ represents money. Then a utility function $u : X \rightarrow \mathbb{R}$ measures a market participant’s satisfaction with his/her current wealth. For a market maker, it is always desirable to make his utility a constant value. Chen and Pennock ([1]) propose a new class of market makers who has a utility function and set prices equal to their risk neutral probabilities. In a securities market setting, risk-neutral probabilities are the price levels that the agent is indifferent between buying or selling an infinitely small number of shares. It can be shown that a market maker can design a utility function based on risk neutral probabilities such that his maximum loss is bounded. One important class of utility functions which has good properties like bounded loss is hyperbolic absolute risk aversion (HARA) utility function. A utility function in HARA class is as follows:

$$ u(m) = \frac{1}{1-\gamma} \left[ \gamma \left( M + \frac{\alpha}{\gamma} m \right)^{1-\gamma} - 1 \right] , $$

where $\gamma \in \mathbb{R} \cup \{-\infty, \infty\}, M + \frac{\alpha}{\gamma} m \geq 0$

A utility-based market maker using a HARA utility function, defined as above, is guaranteed to have bounded loss.

### 3.4 Cost Function

Cost function captures the total amount of money spent by traders till now. Suppose $\vec{q}$ be the current quantity vector. Then $C : \vec{q} \rightarrow \mathbb{R}$ records the total amount of money traders have spent as a function of the total number of shares held of each security.

Cost functions depend on shares. So, for the implementation purposes, cost function seems to be more natural than the scoring rule. The market maker contains a total of $N$ securities, each paying $1$ per share if its corresponding outcome happens. $\vec{q}$ is the vector of all quantities of shares held by the traders. Then the market maker works as follow: (1) The market maker utilizes a cost function $C(\vec{q})$ that records total amount of money traders have spent as a function of the total numbers of shares held by each security. The market maker initiates the market with a quantity vector $\vec{q}_0$. (2) A trader who buys or sells any security or any bundles of securities in the market changes the total number of outstanding securities, i.e. $\overrightarrow{q_i}$ from $\vec{q}_m$ to $\vec{q}_n$. The market maker charges the trader $C(\vec{q}_m) - C(\vec{q}_n)$ dollars of transactions. Negative quantities encode sell orders and similarly negative payment represents the amount earned by the trader for that sale. (3) At any time of the market, the going price of security $i$, $p_i(\vec{q})$, equals $\partial C/\partial q_i$. The price is the cost per share for purchasing an infinitesimal quantity of security $i$. The full cost for purchasing any finite quantity is the integral of price evaluated from $\vec{q}_m$ to $\vec{q}_n$, which equals to $C(\vec{q}_n) - C(\vec{q}_m)$. (4) Once the true outcome becomes known, the market maker pays $1$ per share to traders holding the winning security.

### 3.5 Relation Between Cost Functions and Market Scoring Rules

Suppose the current probability estimate in MSR is $\vec{p}$. A trader who has probability estimate $\vec{p}'$ changes the market probabilities from $\vec{p}$ to $\vec{p}'$ and gets profit $s_i(\vec{p}') - s_i(\vec{p})$. Now suppose, in a market maker mechanism that offers $N$ mutually exclusive and exhaustive securities, the current quantity vector is $\vec{q}$ and price vector is $\vec{p}$. A trader with probability estimate $\vec{p}'$ is myopically incented to buy or sell securities until the market price becomes $\vec{p}'$. His trading behaviour changes the quantity vector to $\vec{q}'$. His profit when outcome $i$ happens is $(q'_i - q_i) - (C(\vec{q}') - C(\vec{q}))$. Trader spent the cost of ($(C(\vec{q}') - C(\vec{q})$)) to buy his securities and he gets the amount $(q'_i - q_i)$ from the market maker if the $i$-th outcome is true. So this is the profit of the trader. If the two mechanisms are equivalent the trader should obtained the same profit no matter which outcome of the random variable is realized. Thus, without loss of generality, the following equation system establishes the equivalence between a MSR market maker and a cost function formulation

$$ s_i(\vec{p}) = q_i - C(q_i) $$

$$ \sum_i p_i = 1 \tag{1} $$

$$ p_i = \frac{\partial C}{\partial q_i} $$

Actually we get the first equation from the profit equivalence. As, $s_i(\vec{p}') - s_i(\vec{p}) = (q'_i - q_i) - ((C(\vec{q}') - C(\vec{q}))$ or by rearranging the terms we can write, $s_i(\vec{p}') - s_i(\vec{p}) = (q'_i - C(q_i)) - (q_i - C(q_i))$. Now second equation is obvious as the probability values always sum upto 1. We already have described the third equation in the previous section.

### 4 Deriving the Cost Function Corresponding to the Spherical Scoring Rule

#### 4.1 Spherical Scoring Rule Market Maker

Recall that spherical scoring rule is the subclass of pseudo-spherical scoring rule corresponding to $\beta = 2$ and $\pi_i = \frac{1}{N}, \forall i = 1, \ldots, N$. Hence spherical scoring rule is given
by,

\[ s_i(\vec{r}) = a_i + b \frac{r_i \sqrt{N}}{(\sum_{i=1}^{n} r_i^2)^{1/2} - 1} \]

### 4.1.1 Bounded Loss of the Spherical Scoring Rule Market Maker

Let \( L \) be the market maker’s maximum possible loss. Recall that \( L \) is given by

\[ L = s_j(e_j) - s_j(r_0), \]

where \( r_0 \) is uniformly distributed, i.e. \( r_i = \frac{1}{N} \). Simplifying, we get

\[ L = b[\sqrt{N} - 1] \]

### 4.2 Utility Based Market Makers

Recall that an utility function of the class of hyperbolic absolute risk aversion (HARA) utility functions is given by:

\[ u(m) = \frac{1}{1 - \gamma} \left[ \gamma \left( M + \frac{\alpha}{\gamma} m \right)^{1-\gamma} - 1 \right], \]

where \( \gamma \in \mathbb{R} \cup \{-\infty, \infty\}, M + \frac{\alpha}{\gamma} m \geq 0 \)

A recent result in this area is the Market Maker Equivalence Theorem which discusses the equivalence relationship between a utility based market maker and an MSR market maker. In particular, the theorem establishes the equivalence for a class of utility functions, the HARA utility class and the class of proper scoring rules, the weighted pseudo spherical scoring rules. The theorem is stated below.

**Theorem 1** (Market Maker Equivalence Theorem [Chen and Pennock [1]]) A utility-based market maker who has a subjective probability estimate \( \vec{p} \) and a HARA utility function with \( \gamma \neq 0 \) is equivalent to a market scoring rule market maker who utilizes a \( \vec{p} \)-weighted pseudospherical scoring rule with \( \beta = 1 - \frac{1}{\gamma} \).

Another result showing the translation from a utility-based market maker to a cost-function formulation is given below.

**Theorem 2** (Chen and Pennock [1]) A utility-based market maker has a cost function that is defined by

\[ \sum_j \pi_j u(C - q_j) = K \]

where \( K \) is a constant.

The following figure shows the relationships among Market Scoring Rule (MSR), utility based markets and cost function.

**Lemma 1** The set of equations relating MSR and cost function are linearly dependent for spherical scoring rule.

**Proof:** Equations 1 gives direct relationship between MSR and cost function. From first equation in Equations 1, we get following for spherical scoring rule,

\[ \frac{r_i \sqrt{N}}{(\sum_{i=1}^{n} r_i^2)^{1/2} - 1} = q_i - C - a_i - b \]

\[ \Rightarrow \sum_i (C - q_i + a_i + b)^2 = \sum_i b^2 \frac{r_i^2 N}{\sum_j r_j^2} = b^2 N \]

Now applying partial derivative w.r.t. \( q_i \) and using third equation of Equations 1,

\[ \Rightarrow p_i \sum (C - q_i + a_i + b) = C - q_i + a_i - b \]

\[ \Rightarrow \sum p_i \sum (C - q_i + a_i - b) = \sum_i (C - q_i + a_i - b) \]

\[ \Rightarrow \sum p_i = 1 \]

Hence from first and third equations, we are able to derive second equation. This shows that the above set of equations are linearly dependent with each other.

Using this lemma the following observation is immediate.

**Observation 1** The set of equations relating MSR and cost function (i.e., Equations 1) can not be used to get cost function corresponding to pseudo-spherical scoring rule. Hence to derive cost function for pseudo-spherical scoring rule, we have to first find utility function and then cost function both by applying Market maker equivalence theorem.

### 4.3 Deriving Cost function for spherical scoring rule

To derive cost function for the spherical scoring rule, we have the following equalities from Market Maker Equivalence theorem (Refer to the proof in Appendix of [1]) where \( b, a_i, \beta \) are parameters for the pseudospherical scoring rule (Refer Section 4.1)

\[ b = \frac{(1 - \beta (K + 1))^{\frac{\alpha+1}{\alpha}}}{\alpha} \]

\[ a_i = M - (1 - \beta (K + 1))^{\frac{\alpha+1}{\alpha}} \]

\[ \beta = 1 - \frac{1}{\gamma} \]
where \( K \) is market maker’s constant expected utility.

For Spherical scoring rule we have \( \beta = 2 \). For simplicity of derivation let us assume \( a_i = 0 \), \( \forall i = 1, 2, \ldots, N \). Hence we get the following.

\[
M = \sqrt{-2K} - 1 \quad \alpha = \sqrt{-2K} - 1 \quad \gamma = -1
\]

[23x368])

\[\therefore u(M) = \frac{(2K + 1) \left( 1 - \frac{\alpha}{\beta} \right)^2}{2} \tag{2}\]

Now we will derive cost function from the utility function derived above. From Theorem 2, we know

\[
\sum_{j=1}^{N} \pi_j u(C - q_j) = K \tag{3}
\]

From Equations 2 and 3 we get following,

\[
\sum_{j=1}^{N} \left[ (2K + 1) \left( 1 + \frac{q_j}{b} - \frac{C}{b} \right)^2 - 1 \right] = 2KN
\]

\[
\Rightarrow (2K + 1) \sum_{j=1}^{N} \left[ 1 + \frac{q_j^2}{b^2} + \frac{C^2}{b^2} - \frac{2q_j}{b} - 2 \frac{C}{b} \left( \frac{q_j}{b} + 1 \right) \right] = (2K + 1)N
\]

\[
\Rightarrow NC^2 - 2C \frac{\sum_{j=1}^{N} q_j}{b^2} + N + 2b \sum_{j=1}^{N} q_j + \sum_{j=1}^{N} q_j^2 = 0
\]

\[
\Rightarrow NC^2 - 2C \left( \sum_{j=1}^{N} q_j + bN \right) + 2b \sum_{j=1}^{N} q_j + \sum_{j=1}^{N} q_j^2 = 0
\]

\[
\Rightarrow \sum_{j=1}^{N} q_j + bN = \sqrt{\left( \sum_{j=1}^{N} q_j \right)^2 + b^2N^2 - N \sum_{j=1}^{N} q_j^2} \tag{4}
\]

Equation 5 gives us explicit formula for computing cost function. But Equation 4 can be arbitrarily complex and in general it may not be possible to come up with such an explicit formulation. Below we describe a general approach for finding cost of some \( \tilde{q} \) when we do not have an explicit formula. The approach is based upon numerical techniques for solving equations. The method we used is Bisection method ([5]). Below we provide a short overview of Bisection method and then explain how we used that.

The bisection method in mathematics is a root-finding method which repeatedly bisects an interval and then selects a sub-interval in which a root must lie for further processing. The method is applicable when we wish to solve the equation \( f(x) = 0 \) for the real variable \( x \), where \( f \) is a continuous function defined on an interval \([a, b]\) and \( f(a) \) and \( f(b) \) have opposite signs. At each step the method divides the interval in two by computing the midpoint \( c = (a+b)/2 \) of the interval and the value of the function \( f(c) \) at that point. Unless \( c \) is itself a root (which is very unlikely, but possible) there are now two possibilities: either \( f(a) \) and \( f(c) \) have opposite signs and bracket a root, or \( f(c) \) and \( f(b) \) have opposite signs and bracket a root. The method selects the sub-interval that is a bracket as a new interval to be used in the next step. In this way the interval that contains a zero of \( f \) is reduced in width by 50% at each step. The process is continued until the interval is sufficiently small.

We may use Bisection method to solve Equation 5. To solve this we first have to find two quantities \( q_i \) and \( q' \), such that the value of the function at those points are of opposite sign. We take the current quantity \( q_i \) since apriori we do not know whether the function if increasing or decreasing, we shall try both possibilities to find those two points. For that we simultaneously doubling \( q_i \) and halving \( q_i \). The method works since for pseudo spherical scoring rules the function is either increasing or decreasing. Once we get the required two points, we can apply Bisection method.

Now we know,

\[
p_i = \frac{\partial C}{\partial q_i} \tag{6}
\]

From Equation 5 and Equation 6 we get,

\[
p_i = \frac{1}{N} \pm \frac{\sum_{j=1}^{N} q_j - Nq_i}{N\sqrt{\left( \sum_{j=1}^{N} q_j \right)^2 + b^2N^2 - N\sum_{j=1}^{N} q_j^2}}
\]

\[
\Rightarrow N^2 \left( p_i - \frac{1}{N} \right) = \frac{\left( \sum_{j=1}^{N} q_j - Nq_i \right)^2}{\left( \sum_{j=1}^{N} q_j \right)^2 + b^2N^2 - N\sum_{j=1}^{N} q_j^2}
\]

\[
\Rightarrow N^2p_i^2 - 2Np_i + 1 = \frac{[S_{i1} - (N-1)q_i]^2}{(S_{i1} + q_i)^2 + b^2N^2 - NS_{i2} - Nq_i}
\]

where

\[
S_{i1} = \sum_{j \neq i} q_j, S_{i2} = \sum_{j \neq i} q_j^2
\]

\[
\Rightarrow N^2p_i^2 - 2Np_i + 1 = \frac{S_{i1}^2 - 2S_{i1}(N-1)q_i + (N-1)^2q_i^2}{S_{i1}^2 + b^2N^2 - NS_{i2} + 2S_{i1}(N-1)q_i}
\]

\[
\Rightarrow \frac{((N-1)\left( N^2p_i^2 - 2Np_i + 1 \right) + (N-1)^2)q_i^2}{2 \left[ S_{i1}(N-1) + S_{i1} \left( N^2p_i^2 - 2Np_i + 1 \right) \right] q_i}
\]

\[
+ \frac{S_{i1}^2 - (S_{i1}^2 + b^2N^2 - NS_{i2}) \left( N^2p_i^2 - 2Np_i + 1 \right)}{0} = 0 \Rightarrow e^2q_i^2 - 2efq_i + g = 0
\]

where

\[
e = (N-1)\left( N^2p_i^2 - 2Np_i + 1 \right) + (N-1)^2,
\]

\[
f = S_{i1}(N-1) + S_{i1} \left( N^2p_i^2 - 2Np_i + 1 \right),
\]

\[
g = S_{i1}^2 - (S_{i1}^2 + b^2N^2 - NS_{i2}) \left( N^2p_i^2 - 2Np_i + 1 \right)
\]

\[
\Rightarrow q_i = \frac{f \pm \sqrt{f^2 - ge}}{e}
\]
Lemma 2 For $\gamma > 0$ the domain of utility functions in HARA utility class is bounded below and not bounded above.

Proof: Recall that the domain of HARA utility functions are given by,

$$M + \frac{\alpha}{\gamma} m \geq 0 \Rightarrow M \geq -\frac{\alpha}{\gamma} m \Rightarrow m \geq -\frac{\gamma}{\alpha} M \quad (\because \gamma > 0)$$

Hence $m$ is bounded below and not bounded above.

Lemma 3 For $\gamma < 0$ the domain of utility functions in HARA utility class is bounded above and not bounded below.

Proof: The domain of HARA utility functions are given by,

$$M + \frac{\alpha}{\gamma} m \geq 0 \Rightarrow M \geq -\frac{\alpha}{\gamma} m \Rightarrow m \leq -\frac{\gamma}{\alpha} M \quad (\because \gamma < 0)$$

Hence $m$ is bounded above and not bounded below.

Observation 2 Spherical scoring rule is not implementable in Zocalo framework.

Proof: We have seen in Equation 2 that the utility function corresponding to Spherical scoring rule has $\gamma = -1$. From Lemma 2, we have $m$ bounded above. For spherical scoring rule, we got $\alpha = \frac{\sqrt{N}}{b}$ in Equation 2. From lemma 2 we know that $m \leq -\frac{\alpha}{\sqrt{N}} m$. Putting the values of $\alpha$ and $\gamma$ for spherical scoring rule, we get the following bound for $m$.

$$m \leq b$$

From practical point of view, $m$ can be arbitrarily large. Hence we need to make $b$ very large. But recall that,

$$p_i = \frac{1}{N} \pm \frac{\sum_{j=1}^{N} q_j - Nq_i}{N\sqrt{\left(\sum_{j=1}^{N} q_j\right)^2 + b^2N^2 - N \sum_{j=1}^{N} q_j^2}}$$

Now $b$ is very large implies that $p_i \approx \frac{1}{N}$. Hence with high $b$, probabilities will remain almost constant which is uniform which is very bad from implementation point of view.

We can also see the problem in the corresponding cost function. The discriminant of the quadratic Equation 4 is

$$D = \left(\sum_{j=1}^{N} q_j\right)^2 + b^2N^2 - N \sum_{j=1}^{N} q_j^2$$

where $b$ and $N$ are constants and $q_i$ is the quantity of $i^{th}$ security. Notice that $q_i$ can be any arbitrary positive real number. We will show that there exists $\bar{q}$ for which $D$ will be negative. Let

$$q_i > \frac{bN}{\sqrt{N-1}} \quad \text{and} \quad q_i = 0, \forall i = 2, \ldots, N$$

For the above $\bar{q}$, $D$ is negative. Hence there exists $\bar{q}$ for which cost is not real and thus it can not be implemented in Zocalo framework.

5 Implementation of a Prediction Market - The Zocalo Framework

5.1 Overview of Zocalo

Zocalo is a toolkit for building prediction markets, markets in securities that pay out depending on outcomes of future events. They provide estimates of the likelihood of specific outcomes that are more reliable than other sources of predictions.

The core of the Zocalo software consists of a variety of Markets, which allow Users to trade Claims (which have Positions representing particular outcomes). Coupons represent holdings of particular Positions, which can be bought with Funds, which represent money. Currency represents the common elements of Funds and Coupons.

There are three types of Markets: Binary Markets are used for prediction markets with two possible outcomes (usually YES and NO), MultiMarkets are for prediction markets with two or more mutually exclusive outcomes, and UnaryMarkets allow for conventional trading of a single asset. These markets can be managed with a MarketMaker that trades with Users directly, or a Book that tracks Users’ standing Orders that other Users are allowed to accept. Binary Markets can be set up with both Book and MarketMaker enabled simultaneously.

There are two primary user interfaces for interacting with the markets. The primary one is ordinary prediction markets, managed by MarketOwner and AllMarkets. A variety of web pages allows users to navigate between different markets of various types, look at the user’s trading history, create new markets, etc. The other major interface supports trading in economic lab experiments. An experimenter configures an experiment, which consists of a number of timed rounds in which Users are assigned initial positions in an artificial good, and allowed to trade with other Users.

- **Languages involved** - The Experiment version relies heavily on AJAX to drive the user interfaces and ensure users see others’ trades and changes in their holdings immediately. HTML pages are generated using JSP, though most of the actual content is generated directly in Java. The Zocalo server uses Jetty to act as a web server. The main() entry points are provided by AllMarkets and ExperimentServer, which use helper classes to set up the Servlets and AJAX connections. AJAX is available in the Prediction Market version, though it isn’t used extensively yet. The project uses Log4J, both to log trading activity and as a transport mechanism for events that trigger AJAX activity. Extensive JUnit tests exercise nearly all of the functionality of Zocalo, though there is no javadoc coverage of these classes. Hibernate is also heavily used and serves as an interface to the in-memory database used by Zocalo.

- **Creation of Zocalo** - Zocalo was built by Chris Hibbert while working on Prediction markets at MIT. The development of Zocalo started in early 2004. Ever since initial development, the number of features in the frame-
work has grown rapidly over the years and the present number of lines of code stands at 58,311. The latest version focuses on integrating book orders with an automated market maker.

5.2 How to Download Zocalo

The latest version Zocalo is available for download at http://sourceforge.net/projects/zocalo/files/zocalo-2011.2/
This link contains both the src package as well as the production-ready package for both Windows and Linux.

5.3 Environments
- **Zocalo Production Environment** - This is the execution environment containing all the binaries, scripts and config files ready for execution. This production environment can be deployed on any machine and we are ready to run the prediction market.
- **Zocalo Source Environment** - This is the development environment containing the source code. We need different environments for production and source so that changes can take place in the code even after our prediction market has gone live. In that case we need to change only the source environment and the most stable working and tested version can be incrementally updated in the production environment.

5.4 Folder Structure
The folders of interest in the production environment:
- **etc** - Contains the hibernate properties file and the zocalo configuration file.
- **jars** - Contains the jar files compiled from the source code.
- **logs** - Log files generated while the service is running.
- **webpages** - Contains the JSP, javascript, html, AJAX and CSS files which are components of the UI.

The folders of interest in the source environment:
- **src** - Contains the whole source code of the whole Zocalo framework arranged in package structure.
- **web** - Contains the files for web development.

5.5 Essential commands
Since the Zocalo build tools are supported by Apache Ant program, for building we can use the following command (Please note that this command needs to be run from the /build directory of the source environment):
```
ant build jsp-pm
```
Commands for starting the Zocalo service in Prod Env:
```
./startDB
./zocalo.sh
```
Command for shutting down Zocalo service:
```
Ctrl + C
```

5.6 Build and deployment process
Please don’t forget to shutdown zocalo before a new deployment.
After every build in the source environment, we need to copy zocalo.jar, hibernate.jar and jsp.jar to the /jars directory in the prod environment.
Restart Zocalo using the above commands.
For any configuration changes, we need to change the zocalo.conf file in the /etc directory of the prod environment.

5.7 Database and Hibernate
Zocalo uses in-memory DB to store all the data corresponding to the prediction market and uses Hibernate as an interface between the code and the database.
Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.
Hibernate’s primary feature is mapping from Java classes to database tables (and from Java data types to SQL data types). Hibernate also provides data query and retrieval facilities. Hibernate generates the SQL calls and attempts to relieve the developer from manual result set handling and object conversion and keep the application portable to all supported SQL databases with little performance overhead.

5.8 Components of Hibernate
Hibernate essentially consists of 3 main types of entities:
- **Model file** - This is the java class which is to be mapped to a table in the relational DB.
- **HBM.xml file** - This file contains the mapping from every java object instance (corresponding to the model class defined above) to the corresponding column in the table.
- **hibernate.properties** - This file contains configurations corresponding to hibernate in general like the underlying DB, the connection pool, max retries in case of failure, access interval etc.

Example: In our case our model class is MultiMarketMaker.java. Let us consider and instance variable 'stocks' that we had introduced. We had to include the above mapping in the MarketMaker.hbm.xml:

5.9 Design of Zocalo
Point to Note according to figure 2: Here only the 'Logarithmic.java' varies according to the scoring rules. Rest of the flow remains the same. Our implemented changes in
Let us consider the case when during a trade the stock vector $\vec{q}$ changes to $\vec{q}'$ and in the process cost function changes from $C(\vec{q})$ to $C(\vec{q}')$.

- **baseC** - baseC corresponds to $C(\vec{q}') - C(\vec{q})$, which denotes the change in the cost function corresponding to the change in the stock vector.
- **totalC** - totalC corresponds to $\vec{q}' - \vec{q}$ which denotes the change in stock vector.
- **incrC** - incrC is the quantity which denotes the difference between totalC and baseC.

**Explanation:** - The user spends baseC in funds, and receives totalC in coupons (the Marketmaker adds incrC in funds to the user's baseC to buy totalC sets from the bank, and keeps the unwanted coupons in its account.) The next simplest case is selling a position when the user already has sufficient coupons. The user provides totalC coupons and receives baseC in funds, which the MARKETMAKER gets by combining the user's coupons with its own and turning them in to the bank for cash, of which it keeps incrC. If the user wants to sell and doesn't have coupons, it costs incrC, which earns totalC in complementary assets. In this case, the MARKETMAKER contributes baseC in funds. So the ratios apply differently in buying and selling and sometimes refer to the user's contribution and sometimes the MARKETMAKER's.

**Example:** - In the multi-outome case, the arithmetic works out that the user can specify any subset of the outcomes, and everything works out. If a claim has 5 possible outcomes (a, b, c, d, and e) and the user wants to buy a and c in preference to b, d, and e, then you add the probabilities for a and c to find p (the others add up to [1-p]) and follow the same procedure. The user pays baseC for baseC sets (which means baseC of each outcome), and then trades the baseC coupons of b, d, and e for incrC coupons of a and incrC of c. The user is out baseC in funds, and now has totalC of a and totalC of c. The market maker has spent incrC in funds and has totalC of b, d, and e. The bank has collected incrC of c. The user is out baseC in funds and issued totalC complete sets. The code currently only supports specifying one outcome to purchase, but that's mostly because the UI issues for specifying other combinations are complicated.

### 5.10 Mapping Scoring Rules in Zocalo

Let us consider the case when during a trade the stock vector $\vec{q}$ changes to $\vec{q}'$ and in the process cost function changes from $C(\vec{q})$ to $C(\vec{q}')$.

- **baseC** - baseC corresponds to $C(\vec{q}') - C(\vec{q})$, which denotes the change in the cost function corresponding to the change in the stock vector.
- **totalC** - totalC corresponds to $\vec{q}' - \vec{q}$ which denotes the change in stock vector.
- **incrC** - incrC is the quantity which denotes the difference between totalC and baseC.

**Explanation:** - The user spends baseC in funds, and receives totalC in coupons (the Marketmaker adds incrC in funds to the user's baseC to buy totalC sets from the bank, and keeps the unwanted coupons in its account.) The next simplest case is selling a position when the user already has sufficient coupons. The user provides totalC coupons and receives baseC in funds, which the MARKETMAKER gets by combining the user's coupons with its own and turning them in to the bank for cash, of which it keeps incrC. If the user wants to sell and doesn't have coupons, it costs incrC, which earns totalC in complementary assets. In this case, the MARKETMAKER contributes baseC in funds. So the ratios apply differently in buying and selling and sometimes refer to the user's contribution and sometimes the MARKETMAKER's.

**Example:** - In the multi-outome case, the arithmetic works out that the user can specify any subset of the outcomes, and everything works out. If a claim has 5 possible outcomes (a, b, c, d, and e) and the user wants to buy a and c in preference to b, d, and e, then you add the probabilities for a and c to find p (the others add up to [1-p]) and follow the same procedure. The user pays baseC for baseC sets (which means baseC of each outcome), and then trades the baseC coupons of b, d, and e for incrC coupons of a and incrC of c. The user is out baseC in funds, and now has totalC of a and totalC of c. The market maker has spent incrC in funds and has totalC of b, d, and e. The bank has collected incrC of c. The user is out baseC in funds and issued totalC complete sets. The code currently only supports specifying one outcome to purchase, but that’s mostly because the UI issues for specifying other combinations are complicated.

### 5.10.1 New Source Files Added to Zocalo for Generalization of Scoring Rule Implementation

We have added an abstract class ScoringRule.java which is the parent for any child class which implements a different class of scoring rule. This abstract class acts like a template which the static child class implementations need to follow strictly. We have implemented child classes Logarithmic.java, Quadratic.java and Spherical.java for the corresponding scoring rules and they implement the following six fundamental methods as specified in the parent class. The methods actually contain the formulas for all calculations corresponding to the scoring rules.

- **baseC**, **totalC**, **incrC**, **newPFromBaseC**, **newPFromTotalC**, and **newPFromIncrC** are the fundamental methods in every scoring rule class.

Also we need to note that there is a different initialization factor(beta) for each and every scoring rule that is actually stored in the DB during the creation of the market. This beta is passed into each of the above methods by the caller. Introduction of different classes for different scoring rule enabled us to make the code more generic such that we can make our prediction market work with any scoring rule with minimal changes in code. By minimal we mean we just need to change the name of the scoring rule while calling, say from Logarithmic.baseC() to Quadratic.baseC().
We designed the market as follows:

We used the Zocalo Prediction Market framework for experimenting Prediction League ( IPL). Users can log-in and trade.

6.1 Putting Prediction Markets into Action- the Indian Institute of Science Prediction League

We used the Zocalo Prediction Market framework for experimentation. We designed the market as follows:

- The DLF Indian Premier League is a cricket series, where teams play 40-over games. The cricket teams are from various parts of India. In the 2012 edition of the League, on which our market is based, 9 teams are participating. The teams are Mumbai Indians, Delhi Daredevils, Royal Challengers Bangalore, Kolkata Knight Riders, Chennai Super Kings, Rajasthan Royals, Deccan Chargers, Pune Warriors India and Kings XI Punjab.

- Our market is called the Indian Institute of Science Prediction League (IPL). Users can log-in and trade.

- The objective of each user would be to predict which team would be at the top of the points table before April 14, 2012. The user having the highest number of shares of the winning team wins.

During the two weeks when the market was floated, we had 114 users and around 2800 trades. This rich dataset allowed us to make a variety of observations on Prediction Markets in general, and on scoring rules in particular.

6.2 IISc Prediction League Results vs DLF IPL 2012 Points Table: April 14, 2012

We first compare some of the outcomes predicted by our Market, with the actual points table on April 14, 2012. The results are shown in the graphs in Figure 7.

The blue lines indicate the share price fluctuation of the two teams. The green line indicates the actual position of the team in the DLF-IPL 2012 points table. Initially, the share prices of all the teams was 11 paisa (as we have 9 teams, and we follow a uniform probability distribution). For Mumbai Indians (MI), users were initially pessimistic of them performing well. This is why their share prices remained low for the first couple of days. However, once their share prices started rising, users became more optimistic, and the share prices of MI rose ever since. In fact, just before April 14, 2012, Mumbai Indians share price was 63 paisa. The bias, however, was temporary. Owing to a string of losses, RCB dropped in the points table, and this drop was foretold by the declining share prices of the team. A similar interpretation can be made for the other teams.

6.3 Comparing Scoring Rules

Whenever a user clicks Trade in the Zocalo Prediction Market, a sequence of entries is updated in the Zocalo database. We convert these entries into a probability vector. For example, suppose the current market price is 11 paisa for each team. This corresponds to a market probability estimate of (0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11). Now a trader enters the market and changes the price of MI from 11 paisa to 20 paisa. This will trigger a change in price for all the other teams, and result in a new probability vector of say (0.2, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1). This corresponds to the new market price vector. We track the probability vectors as different traders trade, giving us a sequence of probability vectors. We collected about 2400 such probability vectors from all the trades that happened till April 14, 2012.

Now, we use this information to compare the Logarithmic, Quadratic and Spherical scoring rules with each other. We study the ranges of 3 scoring rules in particular namely the

<table>
<thead>
<tr>
<th>baseC</th>
<th>incrC</th>
<th>totalC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b \log \left( \frac{1-p}{1-p'} \right)$</td>
<td>$b \log \left( \frac{p}{p'} \right)$</td>
<td>$b \log \left( \frac{1-p}{1-p'} \right)$</td>
</tr>
<tr>
<td>$\frac{bN}{N-1} (p_i^2 - p_i'^2)$</td>
<td>$\frac{b(N-1)}{N} (p_i - p_i') (p_i + p_i' - 2)$</td>
<td>$\frac{2bN}{N-1} (p_i' - p_i)$</td>
</tr>
</tbody>
</table>

Figure 4: baseC, totalC, incrC for a few scoring rules
logarithmic, quadratic and the spherical (positive sign) scoring rules. We present the various comparisons in Figure 8. To refresh the definitions of these scoring rules, we present it below.

\[
\text{Logarithmic: } s_i(\vec{r}) = (b \times \log_e(\vec{r})) \\
\text{Quadratic: } s_i(\vec{r}) = \left( 2 \times br_i - \left( b \sum_j r_j^2 \right) \right) \\
\text{Spherical (positive sign): } s_i(\vec{r}) = \left( a - b + \frac{r_i \times \sqrt{N}}{\sqrt{\sum_j r_j^2}} \right) \\
\text{Spherical (negative sign): } s_i(\vec{r}) = \left( a - b - \frac{r_i \times \sqrt{N}}{\sqrt{\sum_j r_j^2}} \right)
\]

6.3.1 Explanation of Figure 8

In Figure 8 (a), we plot the combined plot of 3 scoring rules corresponding to the probability range [0, 1]. Please note that the points in these plots have been obtained by applying scoring rules to real data probability vectors that were inferred from the Zocalo Prediction Market.

- We find that the Logarithmic scoring rule gives us scores which are only negative. This is due to the property of logarithm function as probabilities are between 0 and 1 and the fact that \( b > 0 \).
- However, with the quadratic scoring rule, we get positive (for high probabilities) and negative (for low probabilities) scores. For the spherical scoring rule, there are two versions which are applicable as shown in Equation 8 and Equation 9. We see from Figure 8 (a) that the negative version is not a good choice for scoring as it gives lower values for higher probabilities. This means that a trader who assigns probability 1 to the winning outcome will get lower score that a trader who assigns a lower probability for the winning outcome which is clearly undesirable.

- The variations for the scores of logarithmic scoring rules can be seen in more detail at very low probabilities in Figure 8 (b) where the graph is obtained by plotting in logarithmic scale. Through this figure we can examine the exact values of logarithmic scores at low probabilities in the range of \( 10^{-12} \) to \( 10^{-2} \) which is approximated to 0 in Figure 8 (a).

In Figure 8 (c), we plot the scores obtained by applying the logarithmic, quadratic and spherical (positive sign) scoring rules to the sequence of trade probability vectors that was obtained from the Zocalo Prediction Market as explained before. We see that Figure 8 (c) is a double plot, i.e., we have 2 y axes and one x axis. The left side y axis corresponds to the plot of quadratic and logarithmic scores while the right side y axis corresponds to the plot of spherical and logarithmic scores.

We also compare the scoring rules in a pairwise manner i.e., quadratic versus spherical, logarithmic versus spherical, logarithmic versus quadratic, and present the results in Figure 8 (d), Figure 8 (e) and Figure 8 (f) respectively. Through these plots, we observe a pattern in the behaviour of the scoring rules though it is left for future work to determine more information about the relation between these rules mathematically.

6.4 Some theoretical observations of Logarithmic Scoring Rule

We now examine a property of logarithmic scoring rule that is undesirable from a practical viewpoint. Recall the definition of logarithmic scoring rule from Equation 7. Usually, in practice, the choice of the parameter \( b \) has to be made before the market is started. This is a serious drawback as a constant value of \( b \) is the reason for the liquidity insensitivity property of the logarithmic scoring rule. We elaborate on this issue below.

Liquidity is the term which is used to indicate the amount of trade in a market. A high liquid market is an indication
Liquidity Insensitivity

A market is said to be liquidity insensitive when there is hardly any trade happening in the market, i.e., the wealth in the market is almost constant over a large range of the number of outstanding shares. In other words, the change in price when the number of outstanding shares of Outcome 1(2) and the market is large. Now, if the trader buys a share of the same outcome as he/she did at the start of the market and if the price jump in the outcome price is 0.1, the market maker is said to be liquidity insensitive.

We now show through plots (Figure 7) the property of Liquidity Insensitivity that is prevalent with the logarithmic scoring rule. We consider a hypothetical two-outcome scenario. For the plot, we keep the number of shares of Outcome 2 fixed, and vary the number of shares of Outcome 1. We then plot the corresponding price of a share of Outcome 1. Doing so gives us a graph like Figure 9 (a) - Figure 9 (c) for b = 10, 1000, 100 respectively.

We observe that the rate of change of the price remains almost constant over a large range of the number of shares of Outcome 1. In other words, the change in price when the number of shares of Outcome 1 is, say, 500 is the same as the change in price when the number of shares is 1000. This property of Liquidity Insensitivity, is an undesirable one—as the number of shares increases, we wish that the price be less sensitive to changes.

The problem primarily lies with the constant factor b. One solution is to make the factor a function of the total number of outstanding shares. If we do so, we obtain a graph Figure 9 (d). To generate the plot above, we made b a function of the total number of outstanding shares i.e., \( b = \alpha \times (q_1 + q_2) \) (Othman et.al. [6]) where \( q_1(q_2) \) corresponds to outstanding shares of outcome 1(2) and \( \alpha = 0.1 \). The result is that the rate of change of price decreases as the number of shares of Outcome 1 increases which essentially makes the market maker sensitive to liquidity.

7 Conclusions and Future Work

In this mini-project, we studied the theoretical and practical capabilities of prediction markets and examined its connections with proper scoring rules in detail. We began with de-
riving the cost functions for the spherical scoring rule which is a member of the class of pseudospherical scoring rules. We showed, through detailed arguments, that it is impractical to implement the spherical scoring rules through the cost function framework as suggested by Chen and Pennock([1]). We then addressed some practical issues in implementing prediction markets by harnessing the open-source framework, Zocalo which uses the J2EE techniques to implement a real-world prediction market based on the logarithmic scoring rule. We floated a prediction market for public trading to predict the outcome of the ongoing Indian Premier League 2012 cricket tournament. The prediction market attracted participation by the student community of IISc and we analyzed the trading activities in this market for a 2 week period. Using the real-world data gathered from Zocalo, we studied properties of three different proper scoring rules namely the logarithmic, quadratic and spherical scoring rules leading to many interesting observations related to these scoring rules.

As part of our future work, it will be interesting to study the theoretical properties of proper scoring rules in more detail and examine many other potential applications of these rules namely demand prediction in smart grids[7], etc. On the implementation side, it may be interesting to extend the test bed prediction market, Zocalo for handling larger loads of traffic and thus, make it a highly scalable web application.

References


Figure 9: Liquidity Insensitivity of Logarithmic Scoring Rule