Learning Parameters by Prediction Markets and Kelly Rule for Graphical Models

Wei Sun
Center of Excellence in C4I
George Mason University
Fairfax, VA 22030

Robin Hanson
Department of Economics
George Mason University
Fairfax, VA 22030

Kathryn B. Laskey
Department of Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030

Charles Twardy
Center of Excellence in C4I
George Mason University
Fairfax, VA 22030

Abstract

We consider the case where a large number of human and machine agents collaborate to estimate a joint distribution on events. Some of these agents may be statistical learners processing large volumes of data, but typically any one agent will have access to only some of the data sources. Prediction markets have proven to be an accurate and robust mechanism for aggregating such estimates (Chen and Pennock, 2010), (Barbu and Lay, 2011). Agents in a prediction market trade on futures in events of interest. Their trades collectively determine a probability distribution. Crucially, limited trading resources force agents to prioritize adjustments to the market distribution. Optimally allocating these resources is a challenging problem. In the economic spirit of specialization, we expect prediction markets to do even better if agents can focus on beliefs, and hand off those beliefs to an optimal trading algorithm. Kelly (1956) solved the optimal investment problem for single-asset markets. In previous work, we developed efficient methods to update both the joint probability distribution and user's assets for the graphical model based prediction market (Sun et al., 2012). In this paper we create a Kelly rule automated trader for combinatorial prediction markets and evaluate its performance by numerical simulation.

1 INTRODUCTION

There was a time when "big" data meant too big to fit into memory. As memory capacity expanded, what was once big became small, and "big" grew ever bigger. Then came cloud computing and the explosion of "big" data to a planetary scale. Whatever the scale of "big," a learner faced with big data is by definition forced to subsample, use incremental methods, or otherwise specialize. Humans are no strangers to specialization: scientific knowledge alone has exceeded the capacity of any single head for at least four centuries. In this paper we consider a mechanism for fusing the efforts of many specialized agents attempting to learn a joint probability space which is assumed to be larger than any one of them can encompass. We seek a mechanism where agents contribute only where they have expertise, and where each question gets the input of multiple agents. As we discuss below, our mechanism is a kind of prediction market. However, in contrast to (Barbu and Lay, 2011), we wish our human and machine agents to concentrate on their beliefs, not on playing the market. Therefore we formulate and develop a helper agent which translates partial beliefs into near-optimal trades in a combinatorial market of arbitrary size. We do not here actually apply it to a big dataset.

It is well known that prediction accuracy increases as more human and/or machine forecasters contribute to a forecast (Solomonoff, 1978). While one approach is to ask for and average forecasts from many individuals, an approach that often works better is to combine forecasts through a prediction market (Chen and Pennock, 2010; Barbu and Lay, 2011). In a market-maker based prediction market, a consensus probability distribution is formed as individuals either edit probabilities directly or trade in securities that pay off contingent on an event of interest. Combinatorial prediction markets allow trading on any event that can be specified as a combination of a base set of events. However, explicitly representing the full joint distribution is infeasible for markets with more than a few base events. Tractable computation can be achieved by using a factored representation (Sun et al., 2012). Essentially, the probabilities being edited by users are the parameters of the underlying graphical model representing the joint distribution of events of interest. In

another words, prediction markets work as a crowdsourcing tool for learning model parameters from human or automated agents.

Prediction markets appear to achieve improved accuracy in part because individuals can focus on contributing to questions about which they have the most knowledge, and in part because individuals can learn by watching the trades of others. However, one important disadvantage of prediction markets is that users must figure out how to translate their beliefs into trades. That is, users must decide when to trade how much, and whether to make a new offer or to accept an existing offer. Prediction markets with market makers can simplify this task, allowing users to focus on accepting existing offers. Edit-based interfaces can further simplify the user task, by having users browse for existing estimates they think are mistaken, and then specify new values they think are less mistaken.

However, even with these simplifications, participants must think about resources as well as beliefs. For example, if users just make edits whenever they notice that market estimates differ from their beliefs, participants are likely to quickly run out of available resources, which will greatly limit their ability to make further edits. To avoid this problem, users must try to keep track of their best opportunities for trading gains, and avoid or undo trades in other areas in order to free up resources to support their best trades.

This problem is made worse in combinatorial prediction markets. By allowing users to trade on any beliefs in a large space of combinations of some base set of topics, combinatorial markets allow users to contribute much more information, and to better divide their labors. But the more possible trades there are, the harder it becomes for users to know which of the many possible trades to actually make.

Ideally, prediction market users could have a trading tool available to help them manage this process of translating beliefs into trades. Users would tell this tool about their beliefs, and the tool would decide when to trade how much. But how feasible is such a tool? One difficulty is that optimal trades depend in principle on expectations about future trading opportunities. A second difficulty is that users must not only tell the tool about their current beliefs, they must also tell the tool how to change such stated beliefs in response to changes in market prices. That is, the trading tool must learn from prices in some manner analogous to the way the user would have learned from such prices.

To make this problem manageable, we introduce four simplifications. First, we set aside the problem of how users can easily and efficiently specify their current beliefs, and assume that a user has somehow specified a full joint probability distribution over some set of variables. Second, we set aside the problem of guessing future trading opportunities, by assuming that future opportunities will be independent of current opportunities. Third, we assume that a user can only accept trade offers made by a continuous market maker, and cannot trade directly with other users. Fourth, we set aside the problem of how a tool can learn from market prices, by assuming that it would be sufficient for the tool to optimize a simple utility function depending on the user's assets and market prices.

Given these assumptions, the prediction market trading tool design problem reduces to deciding what prediction market trades to make any given moment, given some user-specified joint probability distribution over a set of variables for which there is a continuous combinatorial market maker. It turns out that this problem has largely been solved in the field of evolutionary finance.

That is, if the question is how, given a set of beliefs, to invest among a set of available assets to minimize one's chances of going broke, and maximize one's chance of eventually dominating other investors, the answer has long been known. The answer is the "Kelly rule" (Kelly, 1956), which invests in each category of assets in proportion to its expected distant future fraction of wealth, independent of the current price of that category. When they compete with other trading rules, it has been shown that Kelly rule traders eventually come to dominate (Lensberg and Schenk-Hoppé, 2007).

In this paper we report on an implementation of such a Kelly rule based trading tool in the context of a combinatorial prediction market with a continuous market maker. Section 2 reviews the basics of such a combinatorial prediction market. Section 3 reviews the basics of a Kelly rule and then describes how to apply the Kelly rule in a combinatorial prediction market to achieve automated trading. Section 4 reports on simulation tests of an implementation of the Kelly autotrader. Last, in Section 6 we summarize our work and point to potentially promising future research opportunities.

2 COMBINATORIAL PREDICTION MARKETS

In a prediction market, forecasters collaboratively form a probability distribution by trading on assets that pay off contingent on the occurrence of relevant events. In a logarithmic market scoring rule based (LMSR-based) prediction market, a market maker of-

fers to buy and sell assets on any relevant events, varying its price exponentially with the quantity of assets it sells. Tiny trades are fair bets at the consensus probabilities (Hanson, 2003). Larger trades change the consensus probabilities; we call such trades "edits." Suppose $\{z_i, i = 1, \cdots, n\}$ is a set of n possible outcomes for a relevant event, and prior to making a trade, the user's assets contingent on occurrence of z_i are a_i. Suppose the user makes an edit that changes the current consensus probability from p_i to x_i. In a LMSR-based market, as a result of the trade, the user's assets contingent on occurrence of z_i will now be $a_i + b \log_2(\frac{x_i}{p_i})$.

A combinatorial prediction market increases the expressivity of a traditional prediction market by allowing trades on Boolean combinations of a base set of events (e.g., "A and B") or contingent events defined on the base events (e.g., "A given B"). While it is in general NP-hard to maintain correct LMSR prices across an exponentially large outcome space (Chen et al., 2008), limiting the consensus distribution to a factored representation of the joint distribution provides a tractable way to achieve the expressiveness of a combinatorial market. Sun, et al. (2012) showed how adapt the junction tree algorithm to jointly manage each user's assets along with a market consensus probability distribution for a combinatorial prediction market. Our probability and asset management approach has been implemented in a combinatorial prediction market for forecasting geopolitical events (Berea et al., 2013).

3 KELLY RULE AUTO-TRADER

Unlike financial or commodities markets, where financial gain or loss is the primary purpose, the aim of a prediction market is to form consensus forecasts from a group of users for events of interest. A successful prediction market depends on participants who are knowledgeable about and interested in the events in the market. However, lack of experience with or interest in the management of assets can be a significant barrier to participation for some forecasters. Finding a way to engage such content-knowledgeable but market-challenged forecasters could significantly improve the performance of a prediction market. Thus, there is a need for a tool to translate beliefs of forecasters into market trades that efficiently allocate assets according to the forecaster's beliefs.

Fortunately, just such a tool is available from the finance literature. A firmly established result is that we should expect financial markets in the long run to be dominated by investment funds which follow a "Kelly Rule" of investing. Such a strategy invests in each category of assets in proportion to its expected future financial value (Evstigneev et al., 2006; Lensberg and Schenk-Hoppé, 2007; Amir et al., 2001). That is, a Kelly Rule fund that expects real estate to be 20% of distant future wealth should invest 20% of its holdings in real estate. In our context this is equivalent to maximizing an expected log asset holdings, as shown below.

3.1 OPTIMAL ASSET ALLOCATION

The Kelly rule (Kelly, 1956) determines the best proportion of a user's assets to invest in order to achieve the maximum asset growth rate. We briefly review how the Kelly rule works in a simple binary lottery, where a loss means losing one's investment and a win means gaining the amount bet times the payoff odds. Suppose an investor starts with assets y_0; the return is z for a unit bet; and each investment is a fixed percentage c of the user's current assets. After each lottery, the user's assets are multiplied by $(1 + zc)$ in case of a win and $(1 - c)$ in case of a loss. This yields an expected exponential growth rate of

$$\begin{aligned} g(c) &= E\left[\log(\frac{y_n}{y_0})^{\frac{1}{n}}\right] \\ &= E\left[\frac{w}{n}\log(1+zc) + \frac{l}{n}\log(1-c)\right] \end{aligned} \tag{1}$$

where n is the number of trades the user has made; w is the number of times the user has won; and l is the number of times the user has lost. As n approaches infinity, Eq. 1 becomes

$$\begin{aligned} &\lim_{n\to+\infty} E\left[\log(\frac{y_n}{y_0})^{\frac{1}{n}}\right] \\ &\quad = p\log(1+zc) + (1-p)\log(1-c) \end{aligned} \tag{2}$$

Now, suppose there are n possible outcomes $\{z_i, i = 1, \cdots, n\}$; the user's probability for outcome z_i is h_i; and the user's current assets if z_i occurs are a_i. The current market distribution is $\{p_i, i = 1, \cdots, n\}$, and the user is contemplating a set of edits that will change the distribution to $x_i, i = 1, \cdots, n$. Given these edits, the user's assets if outcome z_i occurs will be $\log(a_i + b\log_2(\frac{x_i}{p_i}))$. The maximum asset growth rate is obtained by solving the following optimization problem:

$$\max_x \sum_{i=1}^{N}\left[h_i \times \log(a_i + b\log_2(\frac{x_i}{p_i}))\right] \tag{3}$$

subject to

$$\sum_{i=1}^{N}(x_i) = 1$$

$$0 < x_i < 1, \forall i$$

and

$$a_i + b \log_2(\frac{x_i}{p_i}) >= 0.$$

where i is the global joint state.

3.2 APPROXIMATELY OPTIMAL ALLOCATION

It is straightforward to define the optimization shown in Equation (3) for a joint probability space. However, as noted above, representing the full joint space is in general intractable. We therefore consider the problem in which the user's edits are further constrained to *structure-preserving* edits, which respect the underlying factored representation, ensuring computational tractability (if probabilistic inference itself is tractable).

The objective function in Equation (3) is the expected updated asset w.r.t. the user's beliefs $\{h_i\}$ over the entire joint space. It is desirable to decompose the optimization according to the cliques in the junction tree of the graphical model. However, this is non-trivial because of the logarithm. If edits are structure preserving, assets decompose additively (Sun et al., 2012) as:

$$a_i = \sum_{c \in \mathcal{C}} a_c - \sum_{s \in \mathcal{S}} a_s, \tag{4}$$

where $\mathcal{C}$ is the set of cliques and $\mathcal{S}$ is the set of separators in the junction tree. Therefore (Cowell et al., 1999), computation of expected assets can be decomposed via a local propagation algorithm. However, the logarithmic transformation $\log(a_i + b \log_2(\frac{x_i}{p_i}))$ is not additively separable, and no local propagation algorithm exists for computing the expected utility.

We therefore seek a local approximate propagation algorithm. It is often reasonable to assume that a user has beliefs on only a few of the variables in the market. If all the edited variables are confined to a single clique c_j, we know

$$\frac{x_i}{p_i} = \frac{x^i_{c_j}}{p^i_{c_j}}.$$

One approximation approach is to initialize the user's asset tables $\{a_c, c \in \mathcal{C}\}$ and $\{a_s, s \in \mathcal{S}\}$ with all zeros, and keep a separate cash account containing the user's assets prior to any trades. Each single trade is confined to a single clique. At the time of the trade, we move the cash amount into the asset table into the clique the user is editing. We then solve the optimization problem 3 for the single clique of interest. The user invests the optimal amount. We then find the minimum post-trade assets for the clique of interest. Because of the logarithmic utility, this amount will be greater than zero. We then subtract this positive amount from the clique asset table and add it back to the cash account. We then move to another clique and make the optimal edit there in the same way.

We call this process *local cash-asset management*. This process iteratively moves all cash into a clique, finds the optimal edit confined to that clique, and then moves the maximal amount back to cash while ensuring that all entries in the clique asset table are non-negative. This process proceeds through all the cliques, and increases the expected utility at each step. Local cash-asset management always leaves all separators with zero assets, thus removing the need to manage separators. Furthermore, the global asset computation is simplified to be the sum of all clique assets.

The separately maintained cash amount is guaranteed to be less than or equal to the global minimum assets as computed by the algorithm of (Sun et al., 2012). Thus, this approach is a conservative strategy, improving the user's expected utility but is not guaranteed to reach the maximum expected utility.

4 NUMERICAL SIMULATION

We implemented the Kelly Auto-Trader in MATLAB with an open-source nonlinear optimization solver called IPOPT (Wächter and Biegler, 2006).

4.1 EXPERIMENT DESIGN

To test market performance with the Kelly auto-trader, we simulated a market over a 37-node network whose structure matches *ALARM* (Beinlich et al., 1989). The ALARM network (see Figure 1) is often used as a benchmark for graphical model algorithms, and is substantial enough to provide a reasonable test of our approach. The approach itself is limited only by the efficiency of the nonlinear solver.

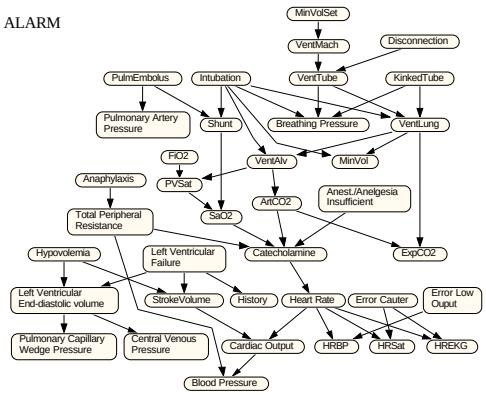

Figure 1: *ALARM*

ALARM has 27 cliques in its junction tree, among which the biggest clique has 5 variables and table size of 108.

To ensure consistency of beliefs across cliques, we use the following procedure to generate a simulated user's beliefs:

1. Choose n cliques to have beliefs, usually about 1/3 of the total number of cliques;
2. Proceeding sequentially for each of the chosen cliques, generate its random belief by a random walk, simulating an efficient market. Update beliefs on the clique and propagate to other cliques using the junction tree algorithm. This procedure gives n junction tree propagations in total.
3. After all propagations, take the potentials on the chosen cliques as the user's final beliefs.

We simulate market participation using three types of traders.

Type 1: EVmaxer invests all of her cash in the market to achieve the maximum asset gain. Formally, EVmaxer solves the following optimization to find the best edit x with her beliefs h,

$$\max_x \sum_{i=1}^{N} \left[h_i \times b \log_2(\frac{x_i}{p_i}) \right] \tag{5}$$

subject to

$$\sum_{i=1}^{N} (x_i) = 1$$

$$0 < x_i < 1, \forall i$$

and

$$a_i + b \log_2(\frac{x_i}{p_i}) >= 0.$$

where i is the global joint state. EVmaxer can suffer catastrophic losses.

Type 2: Kelly-trader is risk-averse and theoretically has the best growth rate in long run. Kelly-trader uses Equation 3 to find the best edit at each trade opportunity.

Type 3: Noiser trades randomly. In any market, we expect a number of noisy traders who have less knowledge than other traders. We simulated Noiser's behavior by moving the current market distribution by random walk of white noise with 15% deviation.

At each trade step, we determine the trader type by taking a random draw from a distribution on the proportion of trader types. Assuming both EVmaxer and Kelly-trader know the pre-generated beliefs, we allow them to determine their optimal edits according to their respective objective functions. They make their edits and the market distribution is updated. Edits continue in this way until interrupted by a question is resolved – that is, its value becomes known to all participants, and all trades depending on the resolved question are paid off by the market maker. The inter-arrival time for question resolutions is modeled by an exponential distribution with mean μ_t. When resolving, we track the min-asset and max-asset for EVmaxer and Kelly-traders to measure their performance. Further, after resolving a question, we add a new question back to the model at the same place where the resolved question was located in the network. Finally, after a certain number of trades, we resolve all questions one by one based on their probabilities. The final assets for different types of users are then calculated and compared.

There are some free parameters in the simulation setting, such as the user's initial assets, the market scaling parameter b, the proportions of different types of traders, etc. The following are the parameter values for a typical simulation run, with explanation of how to choose appropriate values:

- Initial assets $S = 20$ – the small starting assets of 20 will show how EVmaxer goes broke because of her aggressive trading, and how the Kelly-trader is able to grow her assets from a small starting point;
- Market scaling parameter $b = 1000$ – the bigger b, the less influence each trader has; a large b mimics a thick market with many traders;
- Number of trades 1000 – to model the concept of long run effect;
- Mean time between resolutions 30 trades – actual resolutions are drawn from an exponential inter-

arrival distribution with this mean; 30 trades provides sufficient opportunities for well-informed traders to move the market to the correct direction before resolving questions.

- Market participants are composed of 20%, 20%, 60% of EVmaxer, Kelly-trader, and Noiser respectively. Basically, we expect better accuracy of the market probability estimation when there are more Kelly traders, and/or more frequent editing by Kelly traders.

4.2 RESULTS AND ANALYSIS

For a typical run of 1000 trades, Figure 2 presents the marginal probabilities of the resolving states (totally 35 resolutions in this run). At each resolution point, a question was randomly chosen from the market and resolved based on its marginal distribution at the moment. Those resolving probabilities are the "ground truth" values generated by the random walk. Figures 3 and 4 show, in the same simulation run, the performance for EVmaxer and Kelly-trader in terms of their min-asset and max-asset at each resolution point.

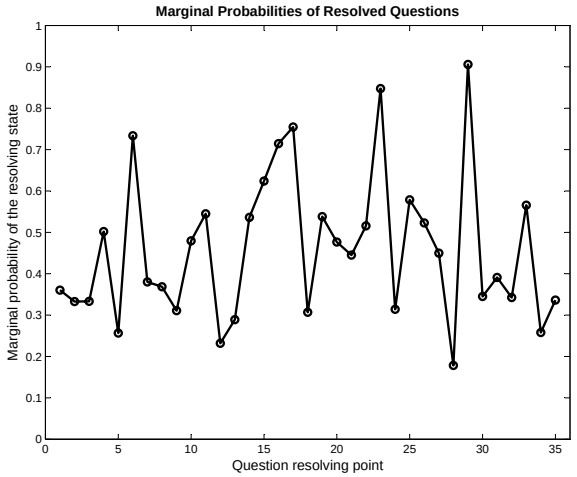

Figure 2: Marginal Probabilities of Resolving State

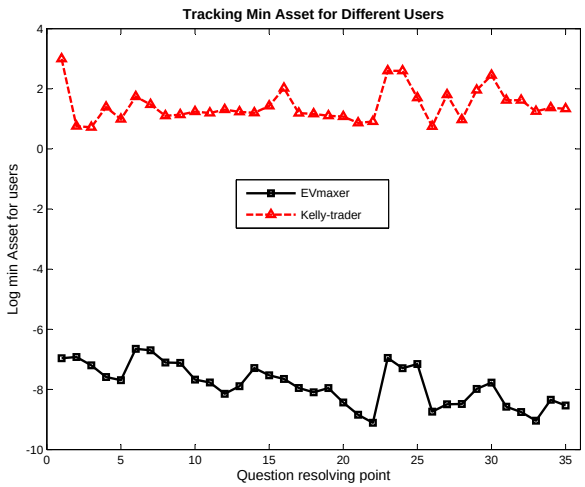

Figure 3: Min-asset Comparison between EVmaxer and Kelly-trader

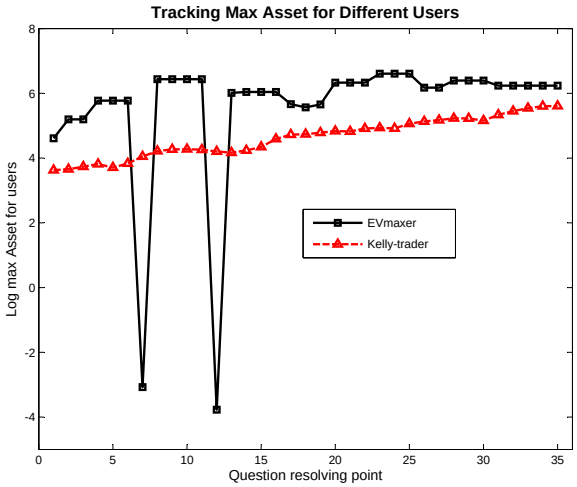

Figure 4: Max-asset Comparison between EVmaxer and Kelly-trader

As expected, Figure 3 shows that the Kelly trader always reserves some assets, while EVmaxer makes very aggressive bets. Notice in Figure 4 that EVmaxer went broke twice at the 7^{th} and 12^{th} question resolution points (we re-initialize EVmaxers' assets at bankruptcy to let them continue to trade). Basically, EVmaxer has a lot bigger volatility while Kelly-trader grows assets consistently. In the simulation, we re-initialize the EVmaxer with the starting asset. But in practice, just one strike will make the EVmaxer deeply hurt. At the end of this run, we sampled the market distribution for 1000 times. Using each sample as the final resolving states for all questions, we compared the final payoff asset for both EVmaxer and

Kelly-trader. Histograms of the final assets are shown in Figure 5, and 6. As you may notice, Kelly-trader has very consistent distribution with mean of 58, and standard deviateion of about 31. But EVmaxer's final asset is distributed very sparsely. Most of time (almost dominant), EVmaxer will have final asset close to zero, although its possible maximum value can be more than 1000.

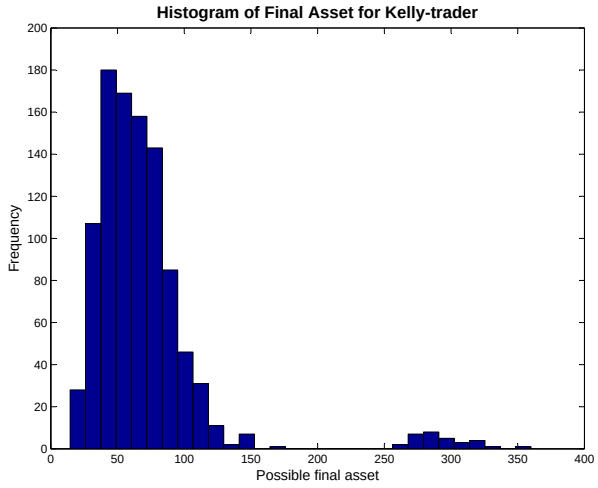

Figure 5: Histogram of the final assets for Kelly-trader

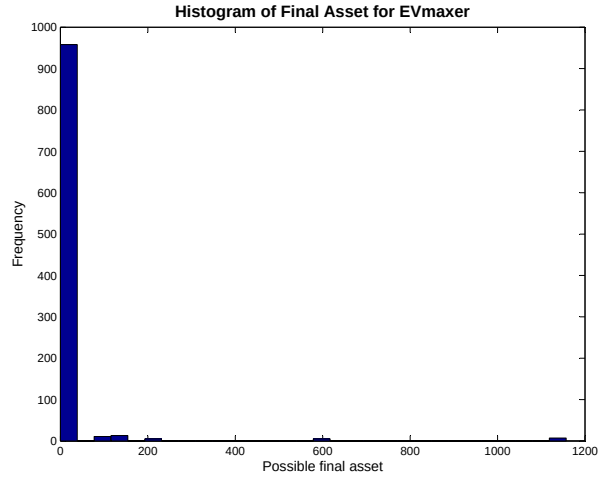

Figure 6: Histogram of the final assets for EVmaxer

5 TAYLOR APPROXIMATION

In this section, we present an alternate approach in which we approximate the utility function by a second-order Taylor series, yielding an approximation to the expected utility in terms of first and second moments of the utility random variable. We then apply methods for local computation of moments of real-valued functions defined on graphical models (Nilsson, 2001; Cowell et al., 1999) to obtain an approximation to the expected utility.

5.1 NOTATION AND DEFINITIONS

Let $\{Z_v : v \in V\}$ be a set of finitely many discrete random variables; let Ω_v denote the set of possible values of Z_v, and let z_v denote a typical element of Ω_v. For $C \subset V$, we write Z_C for $\{Z_v : v \in C\}$, Ω_C for the Cartesian product $\times_{c \in C} \Omega_c$ and z_c for a typical element of Ω_C.

A junction tree $\mathcal{T}$ on V is an undirected graph in which the nodes are labeled with subsets $C \subset V$ called *cliques*; each arc is labeled with the intersection $S = C \cap D$, called the *separator*, of the cliques at the ends of the arc; every $v \in V$ is in at least one clique; there is exactly one path between any two cliques, i.e., $\mathcal{T}$ is a tree; and $C \cap D$ is contained in very clique along the path from C to D. We let $\mathcal{C}$ denote the set of cliques and $\mathcal{S}$ denote the set of separators.

We say a real-valued function f on Ω_V factorizes on the junction tree $\mathcal{T}$ if there exist non-negative real-valued functions $\{h_C : C \in \mathcal{C}\}$ on Ω_C and and $\{h_D : D \in \mathcal{D}\}$ on Ω_D such that for all $v \in V$:

$$f(z_V) = \frac{\prod_{C \in \mathcal{C}} h_C(z_C)}{\prod_{D \in \mathcal{D}} h_D(z_D)} \tag{6}$$

where z_C and z_D denote the components of z_V corresponding to C and D, respectively, and $h_D(z_D) = 0$ only if there is at least one clique D with $D \cap C \neq \emptyset$ and $h_C(z_C) = 0$. In case $h_D(z_D) = 0$ and $h_C(z_C) = 0$ for $D \cap C \neq \emptyset$, we take $h_C(z_C)/h_D(z_D)$ to be 0.

We say a function f on Ω_V decomposes additively on the junction tree $\mathcal{T}$ if there exist non-negative real-valued functions $\{h_C : C \in \mathcal{C}\}$ on Ω_C and and $\{h_D : D \in \mathcal{D}\}$ on Ω_D such that for all $v \in V$:

$$f(z_V) = \sum_{C \in \mathcal{C}} h_C(z_C) - \sum_{D \in \mathcal{D}} h_D(z_D) \tag{7}$$

where z_C and z_D denote the components of z_V corresponding to C and D, respectively. The functions $h_C(x_c)$ and $h_D(x_D)$ in (6) and (7) are called *potentials*; the set $\{h_B : B \in \mathcal{C} \cup \mathcal{D}\}$ is called a (multiplicative or additive, respectively) *potential representation* of f.

5.2 ASSETS AND PROBABILITIES

We assume the user's probability distribution g factorizes according to the junction tree $\mathcal{T}$. We assume trades are constrained to be structure preserving with respect to $\mathcal{T}$; hence, the before-trade market distribution p and the market distribution x after a structure preserving trade also factorize according to $\mathcal{T}$. Consequently, as proven in (Sun et al., 2012), the user's current assets $a(z_V)$ decompose additively on $\mathcal{T}$. Further, if the user makes a structure-preserving trade to change p to x, the user's new assets

$$y(z_V) = a(z_V) + b \log \frac{x(z_V)}{p(z_V)} \tag{8}$$

decompose additively on $\mathcal{T}$.

The expected assets $\mu_Y = \sum_{z_V} g(z_V) y(z_V)$ can be computed efficiently by junction tree propagation (Nilsson, 2001).

We assume the user has a logarithmic utility function

$$u(z_V) = \log y(z_V) = \log \left(a(z_V) + b \log \frac{x(z_V)}{p(z_V)} \right). \tag{9}$$

We seek to maximize

$$EU = \sum_{z_V} q(z_V) \log \left(a(z_V) + b \log \frac{x(z_V)}{p(z_V)} \right) \tag{10}$$

The utility (9) does not decompose additively, and exact computation of the expected utility is intractable. However, we can approximate the utility by the first-order Taylor expansion of $\log y(z_V)$ around μ_Y as:

$$u(z_V) \approx \log y(z_V) + \frac{(y(z_V) - \mu_Y)}{\mu_Y} - \frac{(y(z_V) - \mu_Y)^2}{2\mu_Y^2}. \tag{11}$$

We therefore wish to optimize

$$\begin{aligned} EU &\approx \sum_{z_V} \log q(z_V) \left(y(z_V) - \frac{(y(z_V) - \mu_Y)^2}{2\mu_Y^2} \right) \\ &= \log \mu_Y - \frac{\sigma_Y^2}{2\mu_Y^2}, \end{aligned} \tag{12}$$

where σ_Y^2 is the variance of Y. The objective function (12) can be calculated from the first and second moments of Y. Nilsson (2001) showed how to modify the standard junction tree propagation algorithm to compute first and second moments of additively decomposable functions efficiently. These results can be applied to efficient calculation of the approximate expected utility (12), as described below.

5.3 PROPAGATING SECOND MOMENTS

The standard junction tree algorithm (Jensen, 2001; Dawid, 1992; Lauritzen and Spiegelhalter, 1988) operates on a potential representation $\{h_B : B \in \mathcal{C} \cup \mathcal{D}\}$ for a probability distribution g that factorizes on a junction tree $\mathcal{T}$. A clique C is said to be eligible to receive from a neighboring clique D if either C is D's only neighbor or D has already received a message from all its neighbors other than C. Any schedule is allowable that sends messages along arcs only when the recipient is eligible to receive from the sender, and that terminates when messages have been sent in both directions along all arcs in the junction tree.

Passing a message from D to C has the following effect:

$$h'_S(z_S) = \sum_{D \setminus S} h_D(z_D), and \tag{13}$$

$$h'_C(z_C) = h_C(z_C) \left(\frac{h'_S(z_S)}{h_S(z_S)} \right) \tag{14}$$

That is, the new separator potential is obtained by marginalizing the sender's potential over the variables not contained in the separator, and the new recipient clique potential is obtained by multiplying the old recipient potential by the ratio of new to old separator potentials.

It is clear that message passing preserves the factorization constraint (6). Furthermore, when the algorithm terminates, the new clique and separator potentials are the marginal distributions $g_B(z_B) = \sum_{V \setminus B} g_V(z_V), B \in C \cup S$.

Now, suppose in addition to the multiplicative potential representation for g, we have an additive potential representation $\{t_B : B \in \mathcal{C} \cup \mathcal{S}\}$ for an additively decomposable function y on Ω_V. We modify the message-passing algorithm to pass a bivariate message along each arc. Now, in addition to the effects on the multiplicative potential for the probability distribution g, a message from D to C results in the following change to the additive potential for y:

$$t'_S(z_S) = \frac{\sum_{D \setminus S} h_D(z_D) t_D(z_D)}{\sum_{D \setminus S} h_D(z_D)}, and \tag{15}$$

$$t'_C(z_C) = t_C(z_C) + t'_S(z_S) - t_S(z_S) \tag{16}$$

After the algorithm terminates with messages sent in both directions along all arcs, the final clique and separator multiplicative potentials contain the associated marginal distributions; and the clique and separator additive potentials contain the conditional expectation of Y given the clique/separator state $\mu_{Y|B}(z_B) = \sum_{V \setminus B} g_V(z_V) y_V(z_V) / \sum_{V \setminus B} g_V(z_V), B \in C \cup S$.

After propagation, finding the first moment of Y is straightforward: we simply marginalize the additive potential on any clique or separator:

$$\mu_Y = \sum_B \mu_{Y|B}(z_B) \tag{17}$$

Recall that after propagation, the multiplicative potentials are the marginal probabilities $g_B(z_B)$ and the multiplicative potentials are the conditional expectations $\mu_{Y|B}(z_B)$ for $B \in C \cup S$. Nilsson (2001) proved that the second moment of the additively decomposable function Y can be calculated from the post-propagation additive and multiplicative potentials as follows:

$$\begin{aligned} E[Y^2] = & \sum_{C \in \mathcal{C}} \sum_{z_C} g_C(z_C) \mu_{Y|C}(z_C)^2 \\ & - \sum_{S \in \mathcal{S}} \sum_{z_S} g_D(z_S) \mu_{Y|S}(z_S)^2. \end{aligned} \tag{18}$$

To see why (18) is correct, note that by definition, $E[Y^2] = \sum_{z_u} g(z_u) y(z_u)^2$. Then:

$$\begin{aligned} E[Y^2] &= \sum_{z_u} g(z_u) y(z_u)^2 \\ &= \sum_{z_u} g(z_u) y(z_u) \left(\sum_{C \in \mathcal{C}} \mu_{Y|C}(z_C) - \sum_{S \in \mathcal{S}} \mu_{Y|S}(z_S) \mu \right) \\ &= \sum_{C \in \mathcal{C}} \sum_{z_u} \{ g_u(z_u) y(z_u) \mu_{Y|C}(z_C) \} \\ &\quad - \sum_{D \in \mathcal{D}} \sum_{z_u} \{ g_u(z_u) y(z_u) \mu_{Y|S}(z_S) \} \\ &= \sum_{C \in \mathcal{C}} \{ \sum_{z_C} \mu_{Y|C}(z_C) \sum_{z_{U \setminus C}} g_u(z_u) y(z_u) \} \\ &\quad - \sum_{S \in \mathcal{S}} \{ \sum_{z_S} \mu_{Y|S}(z_S) \sum_{z_{U \setminus S}} \{ g_u(z_u) y(z_u) \} \\ &= \sum_{C \in \mathcal{C}} \sum_{z_C} g_C(z_C) \mu_{Y|C}(z_C)^2 \\ &\quad - \sum_{S \in \mathcal{S}} \sum_{z_S} g_S(z_S) \mu_{Y|S}(z_S)^2. \end{aligned}$$

For a more accurate approximation, we can add additional terms to the Taylor expansion and apply Cowell's method (1999, Sec. 6.4.7) for local propagation of higher moments. In general, we need to propagate $n + 1$ potentials to compute the first n moments of the distribution of Y.

6 SUMMARY

By any definition, big data requires learners to consider only a portion of the data at a time. The problem is then to fuse the beliefs of these specialized agents. We consider a market to be an effective mechanism for fusing the beliefs of an arbitrary mixture of human experts and machine learners, by allowing each agent to concentrate on those areas where they can do the most good (and therefore earn the most points). However, agents that are good at learning are not necessarily good at trading. Our contribution is a general formulation of an agent which will translate a set of beliefs into optimal or near-optimal trades on a combinatorial market that has been represented in factored form such as a Bayesian network.

It is known from theory and empirical results in evolutionary finance that an informed trader seeking to maximize her wealth should allocate her assets according to the Kelly (1956) rule. This rule is very general, and applies even to combinatorial markets. However, it would be intractable to solve on an arbitrary joint state. We have derived two efficient ways to calculate Kelly investments given beliefs specified in a factored joint distribution – such as a Bayesian network. We tested the more conservative rule and found that it has the desired properties: an exponential wealth increase which never goes broke, slightly underperforming an EV-maximizer in the short run, but outperforming it in the long run because the EV-maximizer will go broke.

Our results mean that we can let experts focus on their beliefs in any effective manner, and let an automated agent convert those beliefs into trades. This ability has frequently been requested by participants in the IARPA ACE geopolitical forecasting tournament, and we expect to offer the feature this autumn in our new Science & Technology market.

As a reminder, we assume we have a current set of beliefs from our expert. In practice, we will have to specify a rate at which expressed beliefs converge to the market, in the absence of future updates from the expert, so as not forever to chain our participants with the ghosts of expired beliefs. In addition, we plan to extend our results for systems using approximate inference to update the probability distribution.

Acknowledgments

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20062. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

Amir, R., Evstigneev, I. V., Hens, T., and Schenk-Hopp, K. R. (2001). Market selection and survival of investment strategies. Working Paper 91, University of Zurich. Institute for Empirical Research in Economics.

Barbu, A. and Lay, N. (2011). An introduction to artificial prediction markets for classification. *arXiv:1102.1465*.

Beinlich, I., Suermondt, G., Chavez, R., and Cooper, G. (1989). The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceeding of 2nd European Conference on AI and Medicine*.

Berea, A., Twardy, C., Laskey, K., Hanson, R., and Maxwell, D. (2013). Daggre: An overview of combinatorial prediction markets. In *Proceedings of the Seventh International Conference on Scalable Uncertainty Management (SUM)*.

Chen, Y., Fortnow, L., Lambert, N., Pennock, D. M., and Wortman, J. (2008). Complexity of combinatorial market makers. In *Proceedings of the 9th ACM Conference on Electronic Commerce (EC)*, pages 190–199.

Chen, Y. and Pennock, D. M. (2010). Designing markets for prediction. *AI Magazine*, 31(4):42–52.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36.

Evstigneev, I. V., Hens, T., and Schenk-Hopp, K. R. (2006). Evolutionary stable stock markets. *Economic Theory*, 27(2):449–468.

Hanson, R. (2003). Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119.

Jensen, F. V. (2001). *Bayesian Networks and Decision Graphs*. Springer-Verlag, Berlin, 2001 edition.

Kelly, J. J. (1956). A new interpretation of information rate. *Bell System Technical Journal*, 35:917–926.

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their applications to expert systems. In *Proceedings of the Royal Statistical Society, Series B.*, volume 50, pages 157–224.

Lensberg, T. and Schenk-Hoppé, K. R. (2007). On the evolution of investment strategies and the kelly rule — a darwinian approach. *Review of Finance*, 11:25–50.

Nilsson, D. (2001). The computation of moments of decomposable functions in probabilistic expert systems. In *booktitle = "Proceedings of the 3rd International Symposium on Adaptive Systems"*, pages 116–121.

Solomonoff, R. J. (1978). Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transactions on Information Theory*, IT-24:422–432.

Sun, W., Hanson, R., Laskey, K., and Twardy, C. (2012). Probability and asset updating using bayesian networks for combinatorial prediction markets. In *Proceedings of the 28th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Wächter, A. and Biegler, L. T. (2006). On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.