Abstract

Combining models in appropriate ways to achieve high performance is commonly seen in machine learning fields today. Although a large amount of combinatorial models have been created, little attention is drawn to the commons in different models and their connections. A general modelling technique is thus worth studying to understand model combination deeply and shed light on creating new models. Prediction markets show a promise of becoming such a generic, flexible combinatorial model. By reviewing on several popular combinatorial models and prediction market models, this paper aims to show how the market models can generalise different combinatorial stuctures and how they implement these popular combinatorial models in specific conditions. Besides, we will see among different market models, Storkey’s Machine Learning Markets provide more fundamental, generic modelling mechanisms than the others, and it has a significant appeal for both theoretical study and application.

1 Introduction

The models that are built from combining some individual models are popular in machine learning. In a combinatorial model, the individual models are components and a structure is given to combine the components appropriately, such as taking the average of all individuals or outputing the majority of the results. The structure is thus called the combinatorial structure. The first impression of the combinatorial structures can be given by two popular models, ensemble learning[8] and graphical models[4].

Ensemble learning In machine learning, no individual model can always perform the best in all cases: one model is just suitable for a certain type of dataset or only part of the given dataset. However, we can always make good predictions by using multiple models instead of individual ones. This method is called ensemble learning. For example, for any dataset we compare and choose the algorithm that is the most suitable for this dataset, and then make predictions based on its result[16, 18]. By doing this our prediction will never be worse than the best algorithm on any dataset. Another example is the Netflix challenge[3]. The basic idea shown of ensemble learning is treating different algorithms as components and combining them in an appropriate way. An ensemble learning model can achieve high performance on the dataset without any prior knowledge of it, as long as it has an appropriate way of combining its components. In other words, with a good combinatorial structure an ensemble learning model can perform very well.

Graphical models In a graph the nodes or cliques can be treated as components and the edges represents how the components are combined, therefore it’s natural to think a graph as a combinatorial model. A component in a graphical model always gives a marginal belief on few random variables, which on many occasions differs from the components in ensemble learning models that always give their beliefs on all variables.

Generalising the combinatorial structures has a long-term appeal for both theoretical study and application. It can help understand combinatorial models deeply and inspire us to build new models with novel structures.
In the latest years, prediction markets show a promise for becoming such a generic combinatorial model\cite{1, 24}. Prediction markets are born to be the information aggregators and they naturally combine the agents’ beliefs through market mechanisms\cite{6}. The goods in the market are the contracts associated with certain outcomes of the future event\cite{23}. Each agent bets for the outcomes based on its own belief, by buying or selling some amount of the corresponding goods. When the market reaches equilibrium, the prices of goods are the aggregated beliefs for the outcomes associated with the goods.

A prediction market also has a combinatorial structure, and its agents are the components. Representing the structures in prediction markets have much flexibility, because there are quite many choices on the type of contracts, market mechanisms, and many ways of representing agent behaviours. Therefore, different views on the market structures will result in different prediction market models, such as Storkey’s \textit{Machine Learning Markets}, Lay’s \textit{Artificial Prediction Markets}, Chen’s \textit{Market Maker Prediction Markets}, etc.

This paper aims to show how the market mechanisms can generalise the modelling and implement these models in specific conditions. Besides, we will compare different market models. The criterion the paper uses to discuss modelling and compare different models is how fundamental and generic a model can be. Based on it, the paper thinks Storkey’s \textit{Machine Learning Markets} model provides a better modelling mechanism than others.

Our discussion is based on the key papers \cite{10, 20} and \cite{2, 14, 15}. The former two papers introduce Storkey’s \textit{Machine Learning Markets} and the latter three Lay’s \textit{Artificial Prediction Markets}. Both of them build a generic model successfully, but Machine Learning Markets model on more fundamental assumptions than the other. They also discuss the learning process briefly. Other papers such as \cite{6, 7} and \cite{9} introduce special agents, the market makers, and market scoring rules to represent the market mechanisms and to discuss the learning process. Although introducing market maker can help obtain some important results, it makes the prediction market less general than the former models, which do not introduce any special agents.

To show how generalisation is achieved, the paper will first review on few popular and typical models and the structures they hold before intruding prediction market models.

Chapter \cite{2} introduces the popular combinatorial models and makes a summary at the end. Chapter \cite{3} prepares for the following discussion by introducing some basic concepts in prediction markets. Chapter \cite{4} discusses modelling and compares three market models. Chapter \cite{5} talks about learning briefly. Finally, Chapter \cite{6} draws conclusion.

2 Models that have combinatorial structures

Although only few models will be mentioned here, their structures are quite typical, such as weighted average of the beliefs, product of the beliefs.

2.1 Boosting

Boosting is a class of algorithms\cite{19}. The idea is run the weak learner (who has many weak classifiers) on reweighed training data, then let learned classifiers vote. Boosting is an ensemble learning example. There are many implementations of boosting, some famous ones are \textit{AdaBoost}\cite{9} and \textit{Random Forrest}\cite{5}.

\textbf{AdaBoost} The model of AdaBoost is simply weighted average,

\[ f(x) = \sum_i \alpha_i h_i(x) \]  (1)

Where \( \alpha_i \) is the weight for the weak classifier \( h_i \). Sometimes, \( h_i \) is also called basis, hypothesis or “feature”. These names just reflect the different angles of views on AdaBoost. For this algorithm, learning seems more important than modelling. If it wants to achieve good performances, it should have good choices on the weights. Here PAC (Probably Approximately Correct) learning theory\cite{21} supports AdaBoost to choose weights appropriately. Despite the complicated learning process, the structure of AdaBoost is quite simple.

\textbf{Random Forrest} The main difference between Random Forrest and other boosting algorithms is it introduces stochastic properties to the model. Because of this Random Forrest is not even treated as a boosting method. We won’t dwell on the terminology issue. It’s more important to see the connections between them. The structure of Random Forrest is even simpler, where the weights are all units,

\[ f(x) = \sum_i h_i(x) \]  (2)

However, although its structure seems less flexible without weights, Random Forrest also achieves good performances. The reason is this algorithm puts more efforts on choosing bases. In fact, each basis \( h_i(x) \) is a tree grown on the training data and controlled by a random vector \( \Theta_i \), \( h_i(x) = h_i(x, \Theta_i) \). Similarly, Random Forrest has a simple structure but a complex learning process.

2.2 Mixture model

Suppose there is a set of experts. Each expert performs well only on part of the whole data domain. If a model can give the outputs based on the most suitable experts for the data, the model can always achieve high performances. This kind of model is called mixture model. One example of mixture models is mixture of experts\cite{13}.

\textbf{mixture of experts} To construct the mixture structure, this model assigns different weights to each expert according to the data points. If the experts are good to explain the data point, their weights will be larger than others. Therefore the structure is,

\[ f(x) = \sum_i w_i(x) h_i(x) \]  (3)


Here the weights can be treated as the posterior knowledge given the observations. One special case in mixture of experts is mixture of Gaussians, where each expert is a Gaussian distribution and the corresponding weight \( w_i(x) \) is its responsibility for the observed data. It’s worth noting that one graphical model, the hidden space model[3], can represent this structure.

2.3 Product of experts

This model has a combinatorial structure in a product form[2]. The product is always associated with graphs. The components are thus represented by nodes or cliques. The components give their own beliefs. When they are combined by different types of edges (directed or undirected), the final output is always in the form of the product of these beliefs. One example is product of HMMs[13].

**product of HMMs** The graphical model for product of HMMs is a mixture. It contains both directed edges and undirected edges (Figure 1). The probabilities of variables in this case is complex. However, in graphs the joint probability distribution has a general product form. Denote \( x \) as the vector which contains several random variables. We have,

\[
P(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C)
\]

Here \( \mathcal{C} \) is the set of cliques in the graph, \( C \in \mathcal{C} \) is a certain clique and \( x_C \) is the variables contained in the clique. The final output of the model are always derived from the joint distribution, by either doing marginalization or introducing evidence. For example, using (4) we can write the probability of observed variable \( V \) in this model[13], which we will omit here.

2.4 Summary

So far this paper has introduced several models and the combinatorial structures they represent. There are some common features in these structures. It’s helpful to summarise these features before we discuss the generalisation of the structures.

In combinatorial structures, a component is an agent which has its own belief on the data. Different components (agents) act differently, by mapping the data to different probability distributions. To show this idea more clearly, we consider a sample space \( \Omega \) and its \( \sigma \)-field \( \mathcal{F} \), and suppose the data are drawn from this space \((\Omega, \mathcal{F})\). What the agents actually do is they give their own probability measures on \((\Omega, \mathcal{F})\). Thus agent \( i \) has its own probability space \((\Omega, \mathcal{F}_i, P_i)\).

Combining these agents is actually a process of finding an appropriate map \( \mathcal{F} : \{P_i\} \mapsto P \), where \( P \) is still a probability measure on \((\Omega, \mathcal{F})\). We’ve already seen some examples of \( \mathcal{F} \). In boosting it is the weighted average. Now because of the much flexible definition, \( \mathcal{F} \) should not be restricted to few forms such as weighted average or product, although only these structures have been seen in this paper so far.

Sometimes it’s necessary to consider the situation that some agents may have their beliefs defined on the subspace of \((\Omega, \mathcal{F}_i)\). For example, in product models, the beliefs of agents are only defined on the cliques; they are also called local beliefs or marginal beliefs. There are two ways of interpreting these beliefs. The first one is introducing the subspace \((\Omega_i, \mathcal{F}_i)\). Now the probability space of agent \( i \) is \((\Omega_i, \mathcal{F}_i, P_i)\), and thus \( P_i(\Omega_i) = 1 \). The second one is “slicing” the probability measure using Dirac \( \delta \), so that the probability of an event can be positive only if it is in \((\Omega_i, \mathcal{F}_i)\). However, compared with the first way, the subspace is not explicitly defined.

Suppose the data is drawn from the space \((\Omega, \mathcal{F}_i, P_i)\), where \( P_T \) is the true probability distribution of the data, the learning process aims to give \( P \) which is as close to \( P_T \) as possible. In combinatorial structure, \( P \) is constructed by the beliefs of agents \( \{P_i\} \). Therefore we can think \( \{P_i\} \) as bases and they form a hypothesis space. That’s why in AdaBoost a component is also called basis or hypothesis. Besides, it helps understand modelling and learning in combinatorial structures.

3 Basic Concepts in Prediction Markets

To fill the gap between the economics and machine learning, the paper will first introduce some basic concepts in Prediction Markets.

3.1 Definitions

**Definition 1.** A market is a mechanism for the exchange of goods. The market itself is neutral with respect to the goods or the trades. The market itself cannot acquire or owe goods. Unless explicitly stated otherwise, perfect liquidity is assumed and there is no transaction fee. All participants in the market use the same currency for the purposes of trade.

**Definition 2.** The market equilibrium is a price point for which all agents acting in the market are satisfied with their trades and do not want to trade any further.

**Definition 3.** Prediction market (also predictive market, information market) is a speculative market created for the purpose of making predictions. Current prices in market equilibrium can be interpreted as predictions of the probability of an event or the expected value of the parameter. One good is used as currency and each of the remaining
goods is a bet on a particular outcome of a future occurrence. A bet pays off if and only if the particular outcome associated with this bet actually occurs.

Types of Prediction Market We can design different types of bets (contracts) to realise different types of predictions in the prediction markets (Table 1). For the purpose of predicting based on probabilities, we would like to choose the winner-take-all type.

Definition 4. Machine learning market (or artificial prediction market) is a special type of prediction market. Market participants are classifiers, which are also called agents. The market uses the winner-take-all contract. In the market the agents bet for the outcomes of the future events. They buy and sell bets based on their own beliefs. Prices in market equilibrium will estimate probabilities over the outcomes.

Definition 5. Utility is a measure of satisfaction, referring to the total satisfaction received by a consumer from consuming a good or service. A utility function is defined on the current wealth, and maps the satisfaction to a set of ordinal numbers, for which the common choice is \( \mathbb{R} \). Any function is appropriate as long as it keeps the same ordinals.

Definition 6. A buying function (also betting function) represents how much an agent is willing to pay for a good (or how many contracts it would like to buy) depending on its price.

3.2 Notations

For the consistency this paper follows the notations in [20] and [10].

- An outcome is an event defined on the sample space with a proper \( \sigma \)-field \((\Omega, \mathcal{F})\). This space is mapped to a random vector \( \mathbf{x} \) and each outcome is denoted by \( \mathbf{y} \). In prediction markets each outcome is a good. Goods are enumerated by \( k = 1, 2, \ldots, N_G \). In our discussion, \( \{ y_k \} \) are mutually exclusive. For the sake of simplicity, \( k \) can also denote the good \( \{ y_k \} \). If there is only one random variable, the vector \( \mathbf{x} \) will reduce to variable, and \( G = N_G \) is the number of its outcomes. If there are \( N \) variables in \( \mathbf{x} \), the number of outcomes will increase exponentially on \( N \).

- The price of good \( k \) is denoted by \( c_k \). \( \mathbf{c} = (c_1, c_2, \ldots, c_{N_G})^T \) denotes the price vector. \( \sum_k c_k = 1 \) maths its probability meaning.

- The agents are enumerated by \( i = 1, 2, \ldots, N_A \). The wealth of agent \( i \) is denoted by \( W_i \). The beliefs of agents on \( \mathbf{x} \) are denoted by \( P_i(\mathbf{x}) \). So the belief of agent \( i \) on good \( k \) is \( P_i(k) \). \( \sum_k P_i(k) = 1 \).

- Stockholding of agent \( i \) in good \( k \) is denoted by \( s_{ik} \). Negative stockholding indicates the agent sells the good, so \( s_{ik} \) can be a negative value. \( \mathbf{s}_i = (s_{i1}, s_{i2}, \ldots, s_{iN_G})^T \) denotes the vector of stockholding of agent \( i \).

- The utility of agent \( i \) is denoted by \( U_i \). The buying function of agent \( i \) is denoted by \( s_{ik}(W_i, \mathbf{c}) \).

- If \( \mathbf{x} \) contains multiple variables, they are enumerated by \( j = 1, 2, \ldots, J \). As we discussed in Chapter 2, we can introduce subspaces or cliques to make the beliefs defined only on part of the variables in \( \mathbf{x} \). The subspace of agent \( i \) is denoted by \( S_i \), and the variables in \( S \) is denoted by \( x^S, x^S = \{ x_j | x_j \in S_i \} \). The agent’s belief is denoted by \( P_i(y^S_i) \). \( \{ y^S_i \} \) are the outcomes of \( x^S \).

- The training points are enumerated by \( t = 1, 2, \ldots, T \). The \( t \)-th point is denoted by \( D_t \).

4 Modelling with Prediction Markets

The basic idea of modelling is that markets interpret the agents’ behaviours in an appropriate way, and describing how they interact with each other. Then the connection between market prices and beliefs reveal the relationship between markets and other model structures.

4.1 General market mechanism

The prices of goods are determined by the equilibrium status of the market. Different agents interact with each other in the way that they buy (sell) their preferred amount of goods to (from) others. Their behaviours are interpreted by the buying functions. When the market reaches the equilibrium, the supply matches the demand, thus no one would like to trade any more goods.

\[
\sum_{i=1}^{N_A} s_i(W_i, \mathbf{c}) = 0 \quad (5)
\]

Note that \( s_i(W_i, \mathbf{c}) \), so there are \( N_G \) equations for \( N_G \) goods. Substitute all buying functions into (6) and we can solve the prices for \( N_G \) goods. The prices are the aggregated probability distribution of the future events.

In many situations calculating the market equilibrium using (5) is difficult. For the simplicity of numerical calculation, (10) gives a score function which is called market equilibrium function.

\[
E'(\mathbf{c}) = \sum_{k} \left( \sum_{i} s_{ik}(W_i, \mathbf{c}) \right)^2 \quad (6)
\]

Here \( E'(\mathbf{c}) \geq 0 \), and the equality holds if and only if (5) holds. Therefore minimising \( E'(\mathbf{c}) \) will give the market equilibrium.

Therefore, market equilibrium is the general market mechanism that aggregates the individual beliefs in the market. \( s_i(W_i, \mathbf{c}) \) has not been specified here. Implement the buying function in different ways will give different models.

4.2 Model buying function – Artificial Prediction Markets

One way to implement the buying function is we model the buying function directly in [2] [13] [15]. These papers call
buying function “betting function”, and they think we can choose its form arbitrarily. These papers define the buying function in a factorial form,

\[ s_{ik}(W, \epsilon) = W_i \phi_i(k, \epsilon) \]  

(7)

Where \( \phi_i(k, \epsilon) \) means the proportion of wealth the agent would like to use. These papers give three types of the proportion functions.

- Constant proportion functions
  \[ \phi_i(k, \epsilon) = P_i(k) \]  
  (8)

- Linear proportion functions
  \[ \phi_i(k, \epsilon) = (1 - c_k) P_i(k) \]  
  (9)

- Aggressive proportion functions
  \[ \phi_i(k, \epsilon) = \begin{cases} 1 & \text{if } c_k \leq P_i(k) - \epsilon \\ 0 & \text{if } c_k \geq P_i(k) \end{cases} \]  
  (10)

The advantage of modelling the buying functions directly is, we can suppose some functions with simple forms and can obtain the results analytically. For example, use constant proportion function and \[ \epsilon \], we have

\[ c_k = \frac{\sum_i W_i P_i(k)}{\sum_i W_i}, \quad k = 1, 2, \ldots, N_G \]  
(11)

This is exactly the weighted average of all beliefs, and it interprets the structure of AdaBoost in \[ \epsilon \]. Especially, if the weights are all the same, it interprets Random Forest in \[ \epsilon \].

### 4.3 Model utility – Machine Learning Markets

The drawback of this formulation is, however, the buying function doesn’t always have a factorial form. Besides, from the economics point of view, the buying function is not the foundation to interpret the agent’s behaviour\[ \epsilon \]. Instead, utility function is the one. According to \[ \epsilon \], buying function is derived from the agent’s rational behaviour, that the agent always wants to maximise its utility subject to its budget constraint. Therefore the utility function is defined on the wealth. In prediction markets, the utility has uncertainty. The expected utility function (which is called Von Neumann-Morgenstern utility) can be written as\[ \epsilon \],

\[ E[U_i] = \sum_{k=1}^{NG} P_i(k) U_i(W_i - s^T \epsilon + s_{ik}) \]  
(12)

\[ i = 1, 2, \ldots, N_A \]

This formulation introduces a free degree: agents can change their stockholdings by making risk free trades, namely changing the holding from \( s_i \) to \( s_i + \alpha \epsilon \), and these trades don’t affect their utilities (these utilities keep the same ordinals). We can introduce the gauge or standardization constraint to eliminate this free degree. One choice of the gauge is,

\[ s^T \epsilon = 0 \]  
(13)

Then (12) is rewritten as,

\[ E[U_i] = \sum_{k=1}^{NG} P_i(k) U_i(W_i + s_{ik}) \]  
(14)

\[ s.t. \quad s^T \epsilon = 0, \quad i = 1, 2, \ldots, N_A \]

It’s worth noting that, (12) doesn’t guarantee the invariances under translation, because

\[ U_i(W_i - (s_i + \alpha \epsilon)^T \epsilon + s_{ik}) = U_i(W_i - \alpha - s^T \epsilon \epsilon + s_{ik}) \]  
(15)

The invariances can hold only when \( U_i(x) = U_i(x + t) \), which may not always be met. In \[ \epsilon \] and \[ \epsilon \] the author constructs a cost function that can always hold the translational invariances.

Maximising the utility function gives the buying function. For agent \( i \), the buying function is,

\[ s_i(W_i, \epsilon) = \arg \max_{s_i} E[U_i] \quad s.t. \quad s^T \epsilon = 0 \]  
(16)

Taking derivatives w.r.t each \( s_{ik} \) to get the maximum. Use Lagrange multiplier to include the gauge and we have,

\[ P_i(k) U_i(W_i + s_{ik}) - \lambda_i c_k = 0 \]  
(17)

Solve the \( s_{ik} = s_{ik}(W_i, \epsilon, \lambda_i) \) from the above and combine it with the gauge \[ \epsilon \], we can finally solve the buying function \( s_{ik}(W_i, \epsilon) \). In \[ \epsilon \] the author gives three types of utility functions and derives the corresponding buying functions.
• Logarithmic

\[ U_{\log}(x) = \begin{cases} \log(x) & \text{if } x > 0 \\ -\infty & \text{otherwise} \end{cases} \] (18)

\[ s_{ik}(W_i, c) = \frac{W_i(P_i(k) - c_k)}{c_k} \]

This buying function has a linear form.

- Exponential

\[ U_{\exp}(x) = -\exp(-x) \] (19)

\[ s_{ik}(W_i, c) = \log P_i(k) - \log c_k \]

Because in exponential utility function we have \(-e^{-W-x} = -e^{-W}e^{-x}\), so when taking derivatives the wealth term in the utility function is eliminated. Therefore the buying function do not depend on the wealth.

- Isoelastic (\(\eta > 0\))

\[ U_{iso}(x) = \frac{x^{1-\eta} - 1}{1 - \eta} \] (20)

\[ s_{ik}(W_i, c) = W_i \left[ \frac{\left( \frac{P_i(k)}{c_k} \right)^{1/\eta}}{\sum_j c_j \left( \frac{P_j(j)}{c_j} \right)^{1/\eta}} - 1 \right] \]

Note that when \(\eta \to 1\), it becomes the logarithmic case.

4.4 Implementing combinatorial structures

In former section, we have seen how prediction markets generally work to aggregate information and interpret it in prices of goods. Aggregation is determined by market equilibrium, which is based agents’ behaviours described by their utilities. There is no restriction on choosing agents’ utilities, so all agents can either have the same utility function, or their unique ones. The market whose agents share the same utilities is called homogeneous market, otherwise it’s called inhomogeneous market. An inhomogeneous market is more general since it can become a homogeneous one by assigning the same utility to all agents. However, homogeneous market can implement these popular combinatorial structures we mentioned before and can even bring completely new structures, let alone the inhomogeneous market that may give much more outcomes. The prediction market models, Storkey’s Machine Learning Markets[10, 20] and Lay’s Artificial Prediction Markets[2, 14, 15], both use homogeneous markets to form the combinatorial structures.

Homogeneous market with logarithmic utilities

Using (18) and market equilibrium condition (5), we have,

\[ c_k \propto \prod_{i=1}^{N_A} P_i(k)^{1/N_A} \] (22)

It interprets the structure of Product of HMMs in [4]. Here every clique \(C\) on which the beliefs are defined is a markov chain (Figure 1).

Homogeneous market with iselastic utilities

Using (20) and market equilibrium condition (5), we have,

\[ c_k \propto \sum_{i=1}^{N_A} W_i \left[ \frac{P_i(k)}{c_k} \right]^{-1/\eta} \sum_{j=1}^{N_J} c_j \left[ P_j(j) \right]^{-1/\eta} \] (23)

This is not a closed form because the right side contains prices. Besides, it’s a novel combinatorial structure, and all the models we ever discussed don’t have such kind of structure. Despite the lack of model examples, we can infer some properties of this structure according to (23).

- Similar to logarithmic case, the agents that have large weights (or that are more wealthy) \(W_i\) contribute more to the market pricing. They would like to make the market price close to their beliefs. In economics, these agents are described to act like the “price makers”[6, 7, 11].

- The personal belief is not so important as in logarithmic case. In stead, it is the relative belief \(P_i(k)/c_k\) that really affects the prices. If an agent is wealthy but his belief has a large deviation from others, the market will still treat it as an “outlier” and reduce its contribution.

4.5 Agents with beliefs on subspaces

We have discussed the agents that always have their beliefs on the whole random vector \(x\). This situation is true if \(x\) is actually a random variable, which means \(x\) contains only one entry. However, if there are multiple entries in \(x\), it will be more general to see the agents have beliefs only on part of \(x\). Introducing the agents with marginal beliefs make a model more general since the agents are not required to have their knowledge on all random variables now. As we mentioned in Chapter 2, there are two ways to define the belief on the subspace.

The paper [20] introduces the subspace explicitly and defines the belief on it. Suppose \(x\) contains \(J\) variables, \(x = (x_1, x_2, \ldots, x_J)^T\), and agent \(i\) has its belief only on those variables in its subspace \(S_i\). Then the belief is written as \(P_i(y^{S_i})\). \(\{y^{S_i}\}\) are the outcomes of the subspace variables \(x^{S_i}\). Because \(P_i(y^{S_i})\) is a marginal probability distribution, [20] calls the agents who have their beliefs on subspaces the marginal agents.

The expected utility function for marginal agents is writ-
ten as,
\[
\mathbb{E}[U_i] = \sum_{y^S_i} P_i(y^S_i) \times U_i(W_i - \sum_{y^S_i} s_i(y^S_i)c(y^S_i) + s_i(y^S_i))
\]
(24)

Where \(c(y^S_i) = \sum_{y^S_i} c(y^S_i)\). If \(S_i\) is exactly the whole space, then the above equation is back to (12). There is no simple representation for the prices with marginal agents, but we know the market will give prices based on these marginal beliefs.

4.6 Prediction markets and the map \(F: \{P_i\} \mapsto P\)

Flexible choices of utilities (buying functions) and beliefs make market models general. Again we can understand market models using the map. What the combinatorial structure provides is a map that makes the agents’ beliefs \(\{P_i\}\) map to the aggregated belief \(P\). The prediction market can implement a large number of combinatorial structures by choosing different utilities or buying functions. In former sections we have seen how the market results in those popular structures, such as homogeneous market with logarithmic/exponential utilities, and the structure has not ever been used in any combinatorial models, such as homogeneous market with isoelastic utilities. The flexibility the prediction market has when it implements the map \(F: \{P_i\} \mapsto P\) shows that the prediction market is likely to be a generic combinatorial model.

One interesting question is how many maps on earth can be interpreted by the market. So far no work has been done, but solving this question will help refine this theory and thus it’s worth further study.

4.7 Another market mechanism – Market Maker Prediction Markets

Besides Storkey’s Machine Learning Markets and Lay’s Artificial Prediction Markets, another work in this field is done by Chen et al. [6, 7]. They introduce a market maker to help analyse good pricing and the bounded loss of learning. In [7], the authors prove the equivalence of three different market maker mechanisms: market scoring rule (MSR) market maker, utility-based market maker and cost function based market maker. It’s worth noting that, the utility-based market maker is pretty similar to the machine learning markets, where the market maker’s behaviour is based on its expected utility function. However, instead of maximising the utility function, the agents in the markets aim to maximise their expected wealth subject to the invariance of the market maker’s utility. Suppose there is only two agents, one is a trader (denoted by \(t\)) and another is the market maker (denoted by \(m\)). Then the market mechanism is written as,

\[
\max_{m} \sum_{k} P_t(k)(-m_k) \quad \text{s.t.} \quad \sum_{k} P_m(k)U_m(m_k) = C
\]
(25)

Where \(m_k\) is the money the market maker spend on the good \(k\). The money the trader spend is \(-m_k\) for the assumption that the total wealth of two agents is 0. \(C\) is a constant.

Introducing a special agent has its own drawback: it makes the market less general because in many cases there should be no any special agents in a market. Compared with Market Maker Prediction Market, the Machine Learning Markets model doesn’t introduce any special agents and thus is better to be a generic model.

5 Learning Process

The problems of learning discussed here not only refer to the training, but also refer to the evaluation. Given one model, people are most curious about how good it can actually perform. For prediction markets both problems have not been completely solved yet. However, current results do show prediction market models have good performances, at least from the Bayesian view.

5.1 Training market models

In [10] the author discusses two ways of training the homogeneous market with logarithmic utilities, whose structure is represented by [21]. Because agents’ beliefs don’t change, only their wealths \(\{W_i\}\) keep updating during the training. Therefore the training is also called wealth updating in this paper. The two ways of wealth updating are online update and batch update. Online update means \(\{W_i\}\) update after every training sample. Batch update means that each agents wealth is divided into equal pieces, one piece for each training point, and the updated wealth is the sum of all the updated pieces. Before training, we need choose an appropriate initial wealths. For example, we can choose uniform wealths \(\{1/N_A\}\). Denote them as \(\{W^0_i\}\).

**online update**

\[
W^{t+1}_i = W^t_i - s^T_i c + s_{ik^t} = W^t_i + s_{ik^t}
\]
\[
= W^t_i + \frac{W^t_i(P^t_i(k^t) - c_{k^t})}{c_{k^t}}
\]
\[
= \frac{W^t_i P^t_i(k^t)}{c_{k^t}}
\]
(26)

Where \(k^t\) means the true outcome of training point \(D^t\), and because of the standardisation constraint, \(s^T_i c = 0\), we have \(W^t_i - s^T_i c + s_{ik^t} = W^t_i + s_{ik^t}\). Train the market with \(T\) data points, we will get the final wealth for each agent \(W_i = W^T_i\).

**batch update**

\[
W_i = \frac{\sum_{t=1}^{T} \left[ \frac{W^0_i}{T} + \frac{W^0_i (P^t_i(k^t) - c_{k^t})}{c_{k^t}} \right]}{T}
\]
\[
= \frac{W^0_i}{T} \sum_{t=1}^{T} \frac{P^t_i(k^t)}{c_{k^t}}
\]
(27)

5.2 Bayesian view on training

Now we think the agent’s personal belief in another way. Suppose agents are represented by a random variables \(z\), which has \(N_A\) outcomes \(i = 1, 2, \ldots, N_A\). The joint
portability distribution over the random vector \((x^T, z)^T\) is denoted by \(P'(x, z)\). Then each agent’s belief \(P_i(k)\) is the conditional probability \(P'(x = k | z = i)\), namely we have \(P_i(k) = P'(x = k | z = i)\). For simplicity we write \(P'(i) = P'(x = k | z = i)\). In fact, we treat agents as the outcomes of a latent variable \(z\). If we marginalise it on \(z\), we will obtain \(P(k)\).

The market prices form another probability distribution on \(x, c_k = P(k)\). Note that \(P'\) and \(P\) are two different probability measure on the sample space. \(P'\) is directly obtained by the sum rule, while \(P\) represents an arbitrary aggregation process. Only in one special case we have \(P' = P\): the aggregation process is the (weighted) sum over all agents, which is the same as marginalization. The homogeneous market with logarithmic utilities just provide such a process. When \(P' = P\) we have,

\[
P(k) = P'(k) = \sum_i P'(i, k)
= \sum_i P'(i)P'(k|i) = \sum_i P'(i)P_i(k)
\]

(28)

Where \(P'_i\) can be treated as the weight of agent \(i\), \(W_i = P'_i\) (suppose \(W_i\) has been normalised). This is exactly the Bayesian learning in [4]. Now we consider the learning process and introduce training data.

\[
P(k|D) = \sum_i P'(i|D)P_i(k|D)
\]

(29)

We see that \(W'_i = P'(i|D)\). So \(W_i\) is the responsibility of agent \(i\) for data \(D\), or the posterior probability of agent \(i\) given data. In Bayesian learning, as the number of data points increases, the posterior distribution \(P'(z|D)\) becomes sharp. After enough training points, the agent whose belief is the most close to the true probability distribution will gain the largest weight, while all the others have nearly zero weights. Therefore, in the homogeneous market with logarithmic utilities, finally the prices will be made by the best agent.

In fact we can understand this Bayesian process as a Bayesian model averaging[18]. It is often used to select the best algorithm for a given dataset, among a number of algorithms, but it can select only one algorithm[17]. Strict deduction shows that, with infinite number of training points \(T \to \infty\), if the hypothesis space contains the true probability distribution, Bayesian model averaging can select it out; if the hypothesis space doesn’t contain the true one, then it will select the distribution that is the most close to true one.

Therefore, with enough points, the homogeneous market with logarithmic utilities will perform the same good as the best agent in the market. They have the same loss, which means how well the market can perform will depend on how good the best agent is.

5.3 Further discussion

Bayesian learning seems have solved the learning problem elegantly, but in fact it’s too ideal. The so-called “with enough points” is a condition that can never be met, because we need \(T \to \infty\) training points. It will be much more significant to answer how good performance the market can achieve with finite or few training points. In fact, this question is where some learning theories originate, such as Vapnik-Chervonenkis theory.

For some models, such as boosting, this question has been well solved. So if the market mechanism results in boosting algorithms, we can know the performance of the market. However, we cannot know its performance if the market gives other combinatorial structures. What we want to know is whether it’s possible to have a general idea on the performance of market, regardless of the combinatorial structure it represents.

Some work has done by Chen et al. and their discussion is based on introducing the market maker. In [6] they show that “any cost function based prediction market with bounded loss can be interpreted as a no-regret learning algorithm”. Besides, in [7] they have shown the equivalence between cost function based prediction market and utility based prediction market under some conditions. As we mentioned before, it’s possible to connect the market maker model and Machine Learning Markets model. For example, we can specify an agent in the machine learning markets as a market maker. If it’s true, the results of the market maker model can be used on Machine Learning Markets, making this model better and more general.

6 Conclusion

Prediction markets have shown a huge potential of becoming a generic combinatorial models. In this review, we see how prediction markets provide flexible mechanisms to implement different combinatorial structures. By assigning specific utilities or buying functions to each agent and using market equilibrium condition, Artificial Prediction Markets and Machine Learning Markets can give not only the structures of some well-known models (logarithmic utilities implement averaging structure, and exponential utilities implement product structure), but also ones that have never been used (isoelastic utilities). More generally, prediction markets provide many optional choices of the map \(F : \{P_i\} \to P\) where each \(F\) stands for a combinatorial structure. Therefore the prediction market models are general.

This paper has first compared two prediction market models, Artificial Prediction Markets, which models the buying functions, and Machine Learning Markets, which models the utilities. Both of them perform well in generalising combinatorial models. However, because utility is more conceptually fundamental than buying function and it can give buying function naturally, Machine Learning Markets model is more general and more elegant, than Artificial Prediction Market. Besides, the paper also discusses another market model, called Market Maker Prediction Markets. This model introduce a special agent, the market maker, to help analyse the market mechanism and pricing (namely learning) process. Admittedly some significant results are obtained. However, introducing the special agent makes it less generic than the former two models, which is a big problem. Therefore, based on the criterion that a better model should be more fundamental
and more general, the paper holds the conclusion that Machine Learning Markets is the better one. A possible work might be trying to find the connections between Machine Learning Markets and Market Maker Prediction Markets, in order to use these significant results to refine Machine Learning Markets model.

A brief discussion on the learning process from Bayesian view shows that market models (Machine Learning Markets) do have high performances and thus are attractive to practitioners.

Apart from the modelling and learning theories, Machine Learning Markets model shows a promise in practical implementations. In the market all agents have the same priority, and their behaviours are independent with each other, only based on their own utilities. Parallelisation is therefore able to be used in the model, which shows a long-term appeal when people need to deal with more and more data.

In sum, prediction market models are becoming the generic combinatorial models. Among different market models, Machine Learning Markets has a particular promise of becoming the best market model, for it builds the model more general and elegantly, and gives attractive applications in the future.

References

[1] K.J. Arrow, Robert Forsythe, Michael Gorham, Robert Hahn, Robin Hanson, J.O. Ledyard, Saul Levmore, Robert Litan, Paul Milgrom, F.D. Nelson, and Others. The Promise of Prediction Markets. Science, 320(5878):877, 2008.

[2] Adrian Barbu and Nathan Lay. An Introduction to Artificial Prediction Markets for Classification. Defense, pages 1–28.

[3] R.M. Bell and Y. Koren. Lessons from the Netflix prize challenge. ACM SIGKDD Explorations Newsletter, 9(2):75–79, 2007.

[4] Christopher M Bishop. Pattern Recognition and Machine Learning, volume 4 of Information science and statistics. Springer, 2006.

[5] Leo Breiman. Random Forests. Machine Learning, 45(1):5–32, 2001.

[6] Y. Chen and J.W. Vaughan. A new understanding of prediction markets via no-regret learning. In Proceedings of the 11th ACM conference on Electronic commerce, pages 189–198. ACM, 2010.

[7] Yiling Chen and D.M. Pennock. A utility framework for bounded-loss market makers. In Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence, pages 49–56. Citeseer, 2007.

[8] T. Dietterich. Ensemble methods in machine learning. Multiple classifier systems, pages 1–15, 2000.

[9] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting *. Analysis, 139:119–139, 1997.

[10] Krzysztof Jerzy Geras. Prediction Markets for Machine Learning. PhD thesis, 2011.

[11] Robin Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. The Journal of Prediction Markets, 1(1):3–15, 2007.

[12] Geoffrey E Hinton. Training Products of Experts by Minimizing Contrastive Divergence. Neural Computation, 1800:1771–1800, 2002.

[13] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive Mixtures of Local Experts. Technology, 87:79–87, 1991.

[14] Nathan Lay. Supervised Aggregation of Classifiers using Artificial Prediction Markets Input. PhD thesis, 2009.

[15] Nathan Lay and Adrian Barbu. Supervised Aggregation of Classifiers using Artificial Prediction Markets Input. In Johannes Fürnkranz and Thorsten Joachims, editors, Online, pages 591–598. Omnipress, 2010.

[16] Nick Littlestone and Manfred K Warmuth. The Weighted Majority Algorithm. Information Sciences, 1992.

[17] T.P. Minka. Bayesian model averaging is not model combination. Available electronically at http://www.stat.cmu.edu/minka/papers/bma.html, pages 1–2, 2000.

[18] J.M. Montgomery and Brendan Nyhan. Bayesian Model Averaging: Theoretical developments and practical applications. Political Analysis, 18(2):245, 2010.

[19] R.E. Schapire. The boosting approach to machine learning: An overview. LECTURE NOTES IN STATISTICS-NEW YORK-SPRINGER VERLAG-, pages 149–172, 2003.

[20] Amos Storkey. Machine Learning Markets. Artificial Intelligence, 15, 2011.

[21] LG Valiant. A theory of the learnable. Communications of the ACM, 27(11):1134–1142, 1984.

[22] Hal R Varian. Intermediate Microeconomics: A Modern Approach. WW Norton, 2010.

[23] J. Wolfers and E. Zitzewitz. Interpreting prediction market prices as probabilities. Technical report, National Bureau of Economic Research, 2006.

[24] Justin Wolfers and Eric Zitzewitz. Prediction Markets. Journal of Economics Perspectives, 18(2):107–126, 2004.