Predicting People's Bidding Behavior in Negotiation

Ya'akov Gal and Avi Pfeffer
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
{gal,avi}@eecs.harvard.edu

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]:

General Terms

Design, Experimentation

Keywords

Negotiation, decision-making, Opponent modeling

ABSTRACT

This paper presents a statistical learning approach to predicting people's bidding behavior in negotiation. Our study consists of multiple 2-player negotiation scenarios where bids of multi-valued goods can be accepted or rejected. The bidding task is formalized as a selection process in which a proposer player chooses a single bid to offer to a responder player from a set of candidate proposals. Each candidate is associated with features that affect whether or not it is the chosen bid. These features represent social factors that affect people's play. We present and compare several algorithms for predicting the chosen bid and for learning a model from data. Data collection and evaluation of these algorithms is performed on both human and synthetic data sets. Results on both data sets show that an algorithm that reasons about dependencies between the features of candidate proposals is significantly more successful than an algorithm which assumes that candidates are independent. In the synthetic data set, this algorithm achieved near optimal performance. We also study the problem of inferring the features of a proposal given the fact that it was the chosen bid. A baseline importance sampling algorithm is first presented, and then compared with several approximations that attain much better performance.

1. INTRODUCTION

Recent developments in automated decision-making are making it possible for agents to negotiate with each other in environments of increasing complexity, such as supply chains and market-based scheduling [16, 1]. At the same time, computer systems are becoming commonplace in our every day lives, and settings in which computers and people make decisions together are increasingly prevalent [4]. One area in which both of these trends interleave is electronic commerce, where computers acting as autonomous agents, or as proxies for individual people, are changing the way goods and services are traded on the Internet [21].

One of the main challenges for agents that negotiate in strategic environments is reasoning about the bidding strategies of others. For example, in the Supply Chain domain of the Trading Agent Competition [1] (TAC SCM), agents bid competing offers for supplying a series of multi-valued contracts, each satisfying a minimal configuration of computer hardware resources. The agent with the lowest offer is selected to supply each contract after procuring the necessary resources. Agents cannot observe each others' bids, resources or the winning bid configuration. For agents to out-bid their opponents without sacrificing future revenue, they must be able to predict their opponents' bids for current and future contracts.

Although much work has been done on predicting the bidding and pricing behavior of automatic agents in negotiation [16, 20, 19], modeling people's bidding behavior presents several new challenges. First, social factors such as altruism, self-interest and fairness affect people's negotiation strategies [3]. Second, people's behavior is diverse – the extent to which social factors affect behavior varies greatly among people [12]. Third, people's play does not generally adhere to traditional game-theoretic equilibria [13, 3]. Fourth, humans often make mistakes with regard to their reported utility function[2].

To explain these findings, researchers in behavioral economics have suggested that people's preferences in settings that include other decision-makers are affected by others' outcome as well as their own [15]. For example, in particular contexts, some buyers reject proposals that are highly beneficial to the sellers, even though they are beneficial to the buyers, because they see them as unfair. To be able to interact successfully with individuals, artificial agents must be able to learn how these social preferences affect the bidding behavior of different people.

This work presents a machine learning approach for meeting these challenges. Our study uses a series of take-it-or-leave-it negotiation scenarios in which one agent proposes a trade consisting of multiple goods to a responder agent who must accept or reject the trade. In previous work, we showed that a computer player that used a decision theoretic model to reason about the responder's social preferences outperformed other computer players that used standard game theoretic equilibria [10]. This work formalizes the task of bidding from the proposer's perspective without using a decision-theoretic model. It defines a *selection process* where a number of candidate proposals are available to the proposer and one of them is selected as the chosen bid. Previously we were only concerned with predicting whether or not the proposer's offer was accepted by the responder. In this work, we need to predict the outcome of a multi-way decision process where there are many possible offers for the proposer to choose from.

Each candidate is associated with features, that affect whether or not the candidate is chosen. In general, these features can represent anything about the decision-making that is salient to agents' models. In this work, these features are players' social preferences, defined in terms of functions over their payoffs.

In this work we address the following three tasks. First, we show how to predict the chosen bid given the features of all the candidate proposals. Second, we present and evaluate two algorithms for learning a selection process from data. One of the algorithms uses supervised learning, where an instance consists of the attributes of each proposal and the classification is whether or not the proposal was the chosen bid. In this approach, the probability of a particular candidate offer being selected does not depend on the quality of the attributes of other candidates. The other algorithm learns from each *pair* of candidate proposals, where one was the chosen bid and one was not. In this way, it learns from individual candidates, but also takes into account the quality of other candidates.

To capture the behavior of different types of people, both algorithms use a mixture model of belief networks, each associated with a separate utility function that is a weighted summation of its social preferences. In this way, each network predicts how likely a particular type of person is to bid any given proposal. The feature weights for each type are updated by using standard gradient descent. Because the type of a person is not observed, we estimate a probability distribution over the type space using a technique that is inspired by the Expectation Maximization algorithm [8].

These algorithms are evaluated on real data that was obtained by observing people negotiate with each other in laboratory settings as well as synthetic data that was designed by the researchers by sampling from pre-generated selection models. We expected the second algorithm to be better suited for learning selection processes because it learns from candidate proposals that are *not* chosen as well the chosen bid. This gives us vital information to use in the learning process. For example, if there are many candidate proposals with feature values that are similar to those of the chosen bid, their likelihood should be higher than that of a proposal whose feature values are *less* similar to the chosen bid. The results confirm our hypothesis, in that they show that the algorithm that learns from pairs of candidates performs much better for both data sets. In particular, the second algorithm achieved near optimal performance on the synthetic data set.

The third task is to infer the features of a proposal given the fact that it was the chosen bid. For example, suppose that the contracts in the TAC SCM domain specified minimal hardware configurations. Given a winning bid and a prior distribution over the possible configuration values, it might be possible for agents to infer that the RAM size feature of the winning bid was significantly larger than the minimal required RAM size. They could then tailor their future bids to include larger RAM components.

As a baseline, we present an importance sampling algorithm that weights the features of a candidate proposal by the probability that the object is the winner. We then discuss several alternative approximation algorithms that are much quicker than the baseline algorithm. As a result, many more samples may be generated in the same amount of time, leading to great increases in accuracy over the baseline.

While this paper focuses on the application of selection processes to negotiation, many domains in which a chosen element is selected out of a set of candidates can be modeled as selection processes. For example, the winner of a racing match can be modeled as the chosen candidate out of a set of participants. The features in this example might be skill, age and years of training. Another example might be choosing a spokesperson for a large company where the skills of the spokesperson are the features. We can thus infer that the chosen individual had good communication skills. Lastly, selection processes can also be of use to behavioral economists who wish to describe and predict how behavior differs across cultures and social contexts. The mixture models we present in this work can be used to capture how societies differ in their negotiation behavior.

2. RELATED WORK

Several works provided models for predicting the probability of acceptance of proposals in negotiation. Most of these studied automatic agents. Pardoe and Stone [19] used data obtained from past trading agent competitions to learn the probability that a customer will accept a given bid price. They estimated a separate probability distribution for each range within the possible bid prices and then converted these probabilities to a density function that was biased towards assigning a higher probability of acceptance to lower bids. Lawrence [14] used a naive Bayes classifier to learn the winning probability of seller bids given features relating to buyer characteristics (type of company, past profitability, etc...) The optimal seller price maximized its expected profit given the winning probabilities. All of these works learned a two-way decision process – whether to accept or reject a bid based on past history. We consider a different problem of choosing between multiple candidates where the likelihood of different candidates depends on the relationship between their features.

Das *et al.* [7] showed that computer agents playing stochastic strategies that are based on observed market data could outperform humans in a continuous double auction. This work compared between the performance of automatic agents with the collective performance of people and did not consider social factors. Our criteria of performance is not whether a computer agent can beat humans on average, but the predictive accuracy of a computer model that uses social preferences to predict *individual* bids of people. Also we use a different domain, in which multi-valued goods can be traded by players.

Because we represent social preferences in agents' utility functions, Our work is related to recent approaches for learning agents' utility functions. Ng and Russell [18] defined the inverse reinforcement learning problem as to infer an agent's utility function by observing its behavior. The main technique was to view the agent's decisions as defining a set of linear constraints on the utility functions. This approach relied explicitly on the assumption that the agent is fully rational, and therefore any decision constrains its utility function. Chajewska *et al.* [6], like Ng and Russell, learned a utility function by observing past decisions, but they also allowed a prior probability distribution over the space of utility functions. They also made the assumption that the agent is rational.

In contrast, we do not assume that a decision maker is rational, for two reasons. First, as stated in Section 1, people may choose actions that are contrary to their perceived utility functions [2]. We wish to model the behavior of real agents, so we need to allow for the possibility of mistakes. Second, choosing an action is only one kind of selection process, and we wish are approach to apply to selection processes in general, including those not resulting from a decision-making problem.

Lastly, Chajewska and Koller [5] used a probabilistic approach to learn the structure of utility functions from a database of partially elicited utility functions of individuals. This is a different problem from the one we study, as we are not given any explicit information about the utility function, we are only told which action is selected.

3. SELECTION PROCESSES

We begin by presenting notation. There is a set of N data instances $d^{(1)}, \ldots, d^{(N)}$. Each instance $d^{(k)}$ represents one selection process, consisting of a set $m^{(k)}$ of candidate proposals $c_1^{(k)}, \ldots, c_{m^{(k)}}^{(k)}$ and a chosen bid that that is denoted as $c_*^{(k)}$. We usually drop the superscript (k) where unambiguous and write $c_1, \ldots, c_m$ for the set of candidates. Each candidate proposal c_j has a set of n features $\mathbf{x}_j = x_{j:1}, \ldots, x_{j:n}$, where each feature $x_{j:i}$ is a real number. Following our convention, we use $\mathbf{x}_*$ to denote the features of the chosen bid. In general, i will be used to index features, j will be used to index candidates within an instance, and k will be used to index instances.

Our task is to develop a model for selecting the chosen bid c_* from the set of candidate proposals in each instance. For any candidate, we let $S(c_j)$ denote the decision made for candidate proposal c_j according to the model. This decision can be to select or not to select the candidate as the chosen bid. Given the features of all the candidates, we assume that one of several types is used for making each decision. A type represents a particular way of making a selection, and there may be more than one way of deciding a particular selection process. For example, two types of agents in the TAC SCM domain can make different bids for supplying the same hardware configuration because they weigh the configuration features in different way.

One approach for representing this diversity in the model is to define a separate type for every instance. This captures the fact that each instance is a separate negotiation process that is independent from the other instances. In this case the number of types would have to equal the size of the training set, making it impossible to learn. Instead, we say that agents fall into a finite number of types. There is a prior probability distribution over types, denoted by $P(\tau)$. Once a type is selected according to $P(\tau)$ it is used to select the chosen bid. According to our model, the same type is used for making the decision for all candidate proposal in an instance. This is because it is the same agent that picks the chosen bid and at the same time rejects all the other proposals in that instance.

Associated with each type is a set of weights. The weight w_i^τ is the weight associated with feature i according to type τ. Given the candidate features, and the type, we compute a score for each candidate offer, which is a weighted sum of its feature values. Formally, we write u^τ for the scoring function associated with each type τ, and define

$$u_j^\tau = u^\tau(\mathbf{x}_j) = \sum_{i=1}^{n} w_i^\tau x_{j:i}$$

Now, consider first a simple selection process in which there are only two candidate proposals. For a fully rational decision maker, the chosen bid will be the proposal associated with the higher utility. To capture the fact that people sometimes make mistakes with respect to their utility function, we can allow noise to exist in the selection process, by making the probability of the chosen bid depend on the sigmoid function. That is

$$P(S(c) = \text{chosen}|\mathbf{x}, \tau) = \frac{1}{1 + e^{-u^\tau(\mathbf{x})}} \qquad (1)$$

This function makes sense, because when the score is large and positive, the probability that a particular proposals is the chosen bid tends towards 1; when it is large and negative this probability tends towards 0. At 0, the proposer is completely indifferent between the two possibilities and has a probability of $\frac{1}{2}$ of choosing either. The closer the score is to 0, the more likely the proposer will make a mistake and choose the proposal with lower expected utility.

However, we need to generalize this to a selection process which involves more than two candidate proposals. To this end, we turn the scores of proposals into probabilities by making the probability that candidate c_j is selected to be the chosen offer be proportional to $e^{u_j^\tau}$. That is

$$P(S(c_j) = \text{chosen}|\mathbf{x}_1, \ldots, \mathbf{x}_m, \tau) = \frac{e^{u^\tau(\mathbf{x}_j)}}{\sum_{j=1}^{m} e^{u^\tau(\mathbf{x}_j)}} \qquad (2)$$

This soft-max function is helpful in two ways. First, it is a generalization of the sigmoid model for the simpler model. Indeed, if the decision comes down to two candidates, we have

$$\begin{aligned} P(S(c_1) = \text{chosen}|S(c_1)\text{or}S(c_2) = \text{chosen}, \mathbf{x}_1, \mathbf{x}_2, \tau) &= \frac{e^{u_1^\tau}}{e^{u_1^\tau} + e^{u_2^\tau}} \\ &= \frac{1}{1 + e^{-(u_1^\tau - u_2^\tau)}} \end{aligned} \qquad (3)$$

which is just the sigmoid function using the difference between scores of c_1 and c_2.

Second, by using the soft-max function, we make the probability of each candidate offer to depend not only on its own features, but also on the features of the other candidates which were not chosen.

4. LEARNING TO CHOOSE BIDS

The learning task is to simultaneously learn the prior distribution over type $P(\tau)$, as well as the feature weights $\mathbf{w}^\tau$ associated each type. We are given the features of each candidate, and a classification representing whether the candidate proposal was the chosen bid. We define mixture model over types, where each type is associated with a single-layer sigmoid belief network [17], but the type that generates each data instance is not observed. We present two algorithms for providing a mapping from the candidate features to a classification.

The first algorithm, called LEARN-SUPERVISED is shown in Figure 1. It uses supervised learning where each candidate offer is viewed as a separate training instance. The sigmoid function of Equation 1 is used to compute the predicted output $P(S(c_j))$, which is the likelihood that a particular candidate proposal is selected as the chosen bid. To learn the feature weights of the utility function associated with each type we use gradient descent. We make the true output 1 if the candidate was selected as the chosen bid and 0 if candidate was not selected. The error function to minimize for each candidate c_j is the absolute difference between the true output and the predicted output $P(S(c_j))$.

For each type, we compute the posterior probability $P(\tau \mid S(c_j))$ of generating the candidate c_j using Bayes rule. This probability is then used to adjust the learning rate for the gradient descent step. Intuitively, if a type is more likely to have been used for rejecting or selecting a candidate, that candidate should have more influence in learning the feature weights of that type. Similarly, if a type is unlikely to have been used for a candidate, the candidate should have little impact on the features weights. Therefore the learning rate is made proportional to the probability that the type was used to make this decision. Once the learning rate has been determined, a standard gradient descent update is performed.

The second algorithm views each data instance as consisting of *all* the possible candidate proposals, together with the identity of the selected candidate. The algorithm, called LEARN-PAIRS, is presented in Figure 2. It compares the features of the selected candidate c_* with those of every other candidate c_j, and treats the choice

```
Repeat until convergence of parameters:
  For each instance $d^{(k)}$:
    For each candidate $c_j \in d^{(k)}$:
      For each type $\tau$:
        Let $u_j^\tau = \sum_{i=1}^n w_i^\tau x_{j:i}$
        If $S(c_j) =$ chosen, $P(S(c_j) \mid \tau, \mathbf{x}_j) = \frac{1}{1+e^{-u_j^\tau}}$
        Else $P(S(c_j) \mid \tau, \mathbf{x}_j) = 1 - \frac{1}{1+e^{-u_j^\tau}}$
      For each type $\tau$:
        $P(\tau \mid S(c_j), \mathbf{x}_j) = \frac{P(S(c_j)|\tau,\mathbf{x}_j)P(\tau)}{\sum_\tau P(S(c_j)|\tau,\mathbf{x}_j)P(\tau)}$
        Let learning rate $\alpha_\tau = \alpha * P(\tau \mid S(c_j), \mathbf{x}_j)$
        $p = P(S(c_j) = \text{chosen} \mid \tau, \mathbf{x}_j) = \frac{1}{1+e^{-u_j^\tau}}$
        For each feature $w_i^\tau$:
          If $S(c_j) =$ chosen
            $w_i^\tau \leftarrow w_i^\tau + \alpha_\tau * x_{j:i} * (1-p)$
          Else
            $w_i^\tau \leftarrow w_i^\tau + \alpha_\tau * x_{j:i} * (-p)$
  Reestimate $P(\tau) = \frac{\sum_{k=1}^N \sum_{j=1}^{m^{(k)}} P(\tau|S(c_j^{(k)}),\mathbf{x}_j)}{\sum_{k=1}^N m^{(k)}}$
```

Figure 1: Algorithm LEARN-SUPERVISED

```
Repeat until convergence of parameters:
  For each instance $d^{(k)}$:
    For each type $\tau$:
      For each candidate $c_j \in d^{(k)}$, let $u_j^\tau = \sum_{i=1}^n w_i^\tau x_{j:i}$
      $P(c_*\text{chosen} \mid \tau, \mathbf{x}_1, \ldots, \mathbf{x}_m) = e^{u_*^\tau} / \sum_{j=1}^m e^{u_j^\tau}$
    For each type $\tau$:
      $P(\tau \mid S(c_*) = \text{chosen}, \mathbf{x}_1, \ldots, \mathbf{x}_m) =$
          $\frac{P(S(c_*)=\text{chosen}|\tau,\mathbf{x}_1,\ldots,\mathbf{x}_m)P(\tau)}{\sum_\tau P(S(c_*)=\text{chosen}|\tau,\mathbf{x}_1,\ldots,\mathbf{x}_m)P(\tau)}$
      Let learning rate $\alpha_\tau = \alpha * P(\tau \mid S(c_*) = \text{chosen}, \mathbf{x}_1, \ldots, \mathbf{x}_m)$
      For each $c_j \in d^{(k)}$ where $c_j \neq c_*$:
        For each feature $i$, let $d_i = x_{*:i} - x_{j:i}$
        Let $u_{diff} = \sum_i w_i^\tau d_i = u_*^\tau - u_j^\tau$
        Let $p = P(S(c_*) = \text{chosen} \mid S(c_*)\text{or } S(c_j) = \text{chosen}, \mathbf{x}_*, \mathbf{x}_j)$
            $= 1/(1 + e^{-u_{diff}})$
        For each feature $w_i^\tau$:
          $w_i^\tau = w_i^\tau + \alpha_\tau * d_i * (1-p)$
  Reestimate
    $P(\tau) = \frac{1}{N} \sum_{k=1}^N P(\tau \mid S(c_*^{(k)}) = \text{chosen}, \mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_{m^{(k)}}^{(k)})$
```

Figure 2: Algorithm LEARN-PAIRS

between them as a two-way decision. The utility to use in making the decision is the difference between the individual utilities u_*^τ and u_j^τ. Because the score is a linear function of the features, this difference is equal to the result of applying the scoring function u^τ to the differences between the features. Therefore, the predicted output is simply the probability of c_* being the chosen candidate, given that c_j is not chosen, which can be computed as

$$P(S(c_*) = \text{chosen} \mid c_* \text{ or } S(c_j) = \text{chosen}, \mathbf{x}_*, \mathbf{x}_j) = \frac{1}{1 + e^{-(u_*^\tau - u_j^\tau)}}$$

The error function to minimize is always 1 minus this quantity, which is just applying Equation 3 to two decisions. The learning rate for the gradient descent step is computed in a similar fashion to algorithm LEARN-PAIRS.

Both algorithms need to compute the distribution over the type space. Because the actual type that generates each instance is unknown, we compute a new distribution over the type space by aggregating the likelihood of each and every data instance, and then normalizing. This process is similar in fashion to the Expectation Maximization algorithm. After all candidates for all instances have been processed, new values of $P(\tau)$ are estimated by the Bayes rule using the previously computed $P(\tau|S(c_j), \mathbf{x}_j)$.

4.1 Experimental Results

We evaluated the algorithms using data originating from the Colored Trails (CT) framework [11], in which two players must exchange resources in order to achieve their goals. Each game consists of a one-shot deal, in which one of the players, deemed the allocator, must make an offer to the other player, deemed the deliberator, which can in turn accept or reject the offer. The deliberator is not allowed to counter the allocator's offer with another proposal. The score that each player receives if no offer is made is identical to the score each player receives if the offer is rejected by the deliberator. We refer to this event as the *no negotiation alternative*. The score that each player receives if the offer is accepted by the deliberator is referred to as the *proposed outcome* score. The outcome of the game is determined by an offer-response pair, and each player's score in the game depends solely on his or her own performance. Games can differ in the resource players hold and the dependency relationships that hold between the players.

Each instance in our data set consisted of a CT game, in which the candidates were the set of possible proposals for the allocator in the game, and the chosen bid. There were about 30 candidates in each instance. Let NN_A and NN_D be the no-negotiation alternative scores for the allocator and deliberator, and PO_A and PO_D, the proposed outcome scores The properties of each proposal represents four social preferences that were computed as a function of players' scores: the individual benefit to the allocator if the proposal is accepted $PO_D - NN_D$; the joint benefit for both players $(PO_D + PO_A) - (NN_D + NN_A)$; the degree to which the outcome benefits the allocator more than the deliberator $PO_D - PO_A$; and the advantageousness of the trade to the allocator $(PO_D - NN_D) - (PO_A - NN_A)$.

We evaluated the algorithms using real data, generated by people playing CT games in a lab setting, and synthetic data, where each instance was generated by sampling a CT game, generating the set of candidate proposals for the game and choosing a bid according to a predetermined model. There were 169 instances in the real data set, whereas the synthetic data set included 1,000 instances. In both cases, we trained and tested the algorithms separately, using ten-fold cross validation. We evaluated the algorithms on a held-out data set using two metrics. First, "Likelihood" measures the average probability specified by the learned model that the chosen bid wins (the average is the geometric mean over all instances in the test set). Second, "Accuracy" measures the frequency with which the predicted bid, i.e. the candidate proposal with highest probability of winning, was actually the chosen candidate in the test set. With 30 candidates to choose from, we would expect both numbers to be quite small, but the relative sizes of the numbers indicate which algorithm performs better.

Table 1 compares the performance of LEARN-PAIRS to LEARN-SUPERVISED on real data, when using one, two and three types to learn a selection process. It also includes a naive predictor as a baseline that assumes a uniform probability distribution over all candidates for each instance. The relative sizes of the numbers indicate which algorithm performs better.

As shown in the table, while both algorithms performed better than the baseline, the performance of LEARN-PAIRS was consistently better than that of LEARN-SUPERVISED for both evaluation metrics. In addition, both algorithms improved in one metric as the number of types increased.

	Algorithm	Likelihood	Accuracy
	UNIFORM	0.0318	0.0375
One type	LEARN-SUPERVISED	0.0345	0.0562
	LEARN-PAIRS	0.0483	0.0812
Two types	LEARN-SUPERVISED	0.0344	0.0625
	LEARN-PAIRS	0.0591	0.0812
Three types	LEARN-SUPERVISED	0.0341	0.0688
	LEARN-PAIRS	0.0609	0.0812

Table 1: Performance comparison on real data

Table 2 evaluates performance of both algorithms on synthetic data. Here, we ran in an experiment in which a model of 3 types was used to generate the data and 3 types were used by the learning algorithms. We also report the performance of the actual model used to generate the data, which constitutes a gold standard. We ran experiments with many different numbers of types, both for the actual model and the learning algorithm, with comparable results. Results show that the likelihood of LEARN-PAIRS is much better than that of LEARN-SUPERVISED, and is close to the performance represented by the gold standard. Interestingly, both algorithms have the same accuracy as the gold standard. This suggests that simply predicting the chosen bid is an easier task than determining the probability of selecting each candidate as the chosen bid.

One possible explanation about this difference in performance of the algorithms in the CT domain is that LEARN-SUPERVISED ignores the dependencies between candidate proposals. It assumes that the probability a candidate is accepted depends only on its own features and not on the features of all the other candidates. In reality, if the other candidate proposals are exceptionally good, this candidate should be less likely to be the chosen bid, and conversely if they are bad. By ignoring this fact, LEARN-SUPERVISED is missing crucial information relevant to the selection of the chosen bid. Furthermore, the probabilities of acceptance for all candidates in an instance will in general not sum to 1, and a further drawback of LEARN-SUPERVISED is that a different type may be used for the decision for each candidate. In contrast, LEARN-SUPERVISED models the dependencies between candidates, and assumes the same type makes the decision for each instance.

Algorithm	Likelihood	Accuracy
UNIFORM	0.0320	0.0374
LEARN-SUPERVISED	0.0346	0.2212
LEARN-PAIRS	0.0603	0.2212
ACTUAL MODEL	0.0620	0.2212

Table 2: Performance comparison on synthetic data

5. INFERRING THE PROPERTIES OF A CHOSEN BID

In this section we consider the problem of inferring the properties of a proposal from the fact that it was the chosen bid. We assume now that the model is known, and that we have a prior probability distribution over the properties of any candidate proposal. For now, we do not assume that different candidates have the same prior. If a given proposal happens to be the chosen bid, it means that those property values that lead to a higher score are more likely. The task is to obtain the posterior distribution over the properties of a target proposal, given that it is the chosen bid.

As a baseline, we propose a simple importance sampling method to achieve this task. Let the target candidate be c_1. We begin by sampling from the prior distribution for each candidate including the target, to obtain features $\mathbf{x}_1$ for the target and $\mathbf{x}_j$ for each of the other candidates c_j. We also sample the type τ used to make the decision from the probability distribution $P(\tau)$. We compute the score for each candidate $u_j^\tau = \sum_i w_i^\tau x_{j:i}$. We then compute the probability $p = \frac{e^{u_1^\tau}}{\sum_{j=1}^m e^{u_j^\tau}}$ that the target candidate was chosen. We then weight the sample consisting of the features $\mathbf{x}_1$ by the probability p. We thus obtain a set of weighted samples that is an approximation to the posterior distribution over features of the target candidate. This set of samples can be used to obtain expectations of the features we are interested in.

This method, which we call IMPORTANCE-SAMPLE, requires sampling all the candidate proposals. We can do better by making the observation that the probability that a candidate is the chosen bid depend most heavily on its own score. Once the type has been fixed, and the score of the target candidate has been determined, we can attempt to estimate the probability that the target is the chosen bid without sampling the other candidates. This approach has the advantage that because only one candidate proposals needs to be sampled, many more samples can be taken for the same running time. On the other hand, because this approach uses an estimate of the probability that the target candidate is the winner, rather than sampling, it is biased. The degree of the bias depends on the quality of the estimates.

There are several ways to estimate the probability that the target proposal is the chosen bid, given its features. We wish to estimate

$$P(c_1 \text{ wins}|\mathbf{x}_1, \tau) = E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\frac{e^{u^\tau(\mathbf{x}_1)}}{e^{u^\tau(\mathbf{x}_1)} + \sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}}] \quad (4)$$

where $E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[f]$ denotes the expectation of f with respect to $\mathbf{x}_2, ..., \mathbf{x}_m$. This probability is hard to estimate directly without sampling $\mathbf{x}_2, \ldots, \mathbf{x}_m$ many times for each $\mathbf{x}_1$, which is precisely what we wish to avoid. However, we can hope to approximate it, either with a method that does not sample $\mathbf{x}_2, \ldots, \mathbf{x}_m$ at all, or with a method that only samples $\mathbf{x}_2, \ldots, \mathbf{x}_m$ a fixed number of times, and amortizes that sampling over all samples of $\mathbf{x}_1$. Our first approximation comes from the consideration that in the case where c_1 is unlikely to be the winner, $e^{u^\tau(\mathbf{x}_1)}$ will usually be small compared to $\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}$. Therefore the ratio $\frac{P(c_1^{(1)} \text{ wins}|\mathbf{x}_1^{(1)}, \tau)}{P(c_1^{(2)} \text{ wins}|\mathbf{x}_1^{(2)}, \tau)}$ of the probability the target candidate wins for two different samples $\mathbf{x}_1^{(1)}$ and $\mathbf{x}_1^{(2)}$will usually be approximately equal to

$$\frac{E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\frac{e^{u^\tau(\mathbf{x}_1^{(1)})}}{\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}}]}{E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\frac{e^{u^\tau(\mathbf{x}_1^{(2)})}}{\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}}]} = \frac{e^{u(\mathbf{x}_1^{(1)})}}{e^{u(\mathbf{x}_1^{(2)})}} \quad (5)$$

Since the only thing that is required to get good weighted samples is that the ratios between the weights be correct, we can simply use $e^{u^\tau(\mathbf{x}_1^{(k)})}$ as the weight of the kth sample. The scheme that uses these weights is called SAMPLE-SIMPLE.

However, this is a very crude approximation. In particular, the approximation is inaccurate in the cases where $e^{u^\tau(\mathbf{x}_1)}$ is large. In such cases, the weights will be biased upwards, because we are ignoring the $e^{u^\tau(\mathbf{x}_1)}$ term in the denominator of Equation 4 which is no longer negligible. A better approximation can be obtained by not ignoring the $e^{u^\tau(\mathbf{x}_1)}$ term in the denominator. Instead, we approximate Equation 4 by

$$\frac{e^{u^\tau(\mathbf{x}_1)}}{e^{u^\tau(\mathbf{x}_1)} + E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}]} \quad (6)$$

This scheme, called SAMPLE-EXPECT, uses a preprocessing step in which a number of samples of $\mathbf{x}_2, ...\mathbf{x}_m$ are taken to estimate $E_{\mathbf{x}_2,\ldots,\mathbf{x}_m}[\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}]$ for each type τ. The estimate for the selected type is then substituted in Formula 6 to obtain the approximate weight for sample $\mathbf{x}_1$.

Of course, simply moving the expectation inwards is likely to produce an inaccurate approximation. In particular, the expectation of Equation 4 is likely to be dominated by cases where $\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}$ is small, not by the expected value of $\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}$. This suggests a different approximation scheme, where we use small values of $\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}$. In this scheme, called SAMPLE-MIN, we take a certain number of samples of $\mathbf{x}_2, ..., \mathbf{x}_m$ in a preprocessing step. From those samples we take the minimum value of $\sum_{j=2}^m e^{u^\tau(\mathbf{x}_j)}$ for each type. We use that value for the selected type instead of the expectation in Equation 6 to compute the weight for sample $\mathbf{x}_1$. An interesting question in this scheme is how many samples of $\mathbf{x}_2, ...\mathbf{x}_m$ to take in the preprocessing step. Even if we had infinite time, taking more samples would not necessarily be a good thing, because the minimum may become too small, yielding a less accurate approximation.

Thus far, we have assumed there is a particular target proposal whose features we want to infer. But another kind of inference problem is to infer the features of a generic proposal, given the fact that it was chosen. For example, the runners in a race might include three runners whom we know by name and who have a reputation, as well as fifteen other anonymous, generic runners. Suppose now that we want to infer the posterior probability distribution over the properties of an anonymous candidate. One approach would be simply to choose one of the anonymous candidates to be our target candidate, and proceed using one of the methods described earlier. But this would be unnecessarily wasteful — it would ignore the probability that one of the other anonymous candidates is chosen as the winner. In the extreme case, all the candidates have the same prior distribution over features. In that case the probability that one of the anonymous candidates is selected is 1, and every sample should have a weight of 1.

A better approach, assuming there is a reasonable number of anonymous candidates, is to use a rejection sampling method, which we call SAMPLE-GENERIC . We sample a type from $P(\tau)$, and features for each of the candidates, as in the baseline IMPORTANCE-SAMPLE method. We then select a winning candidate using the model. If the winner is an anonymous candidate with the prior we are interested in, we keep its features as a sample. If not, we reject the sample.

It is worth examining the advantage of this approach more closely. The advantage over IMPORTANCE-SAMPLE is indeed what we have stated, that each sample can be expected to have a much higher weight. The advantage of this approach over the estimated schemes SAMPLE-SIMPLE, SAMPLE-EXPECT and SAMPLE-MIN is not that the samples taken have higher weight, since with those schemes more samples could be taken, enough so that their total weight is equal to that of SAMPLE-GENERIC. Rather, it comes from the fact that SAMPLE-GENERIC does not use a biased approximation of the probability that the target candidate wins.

5.1 Experimental Results

We first tested the algorithms for inferring the properties of a winning candidate using synthetic data, where there is only one feature x, one type, and the scoring function is simply $u = x$. The prior over the feature is a Gaussian.

		Probability that c_1 wins		
		0.001	0.01	0.1
Algorithm	Samples	Error	Error	Error
IMPORTANCE-SAMPLE	100	0.1770	0.1689	0.1454
SAMPLE-SIMPLE	1000	0.0576	0.0610	0.2412
SAMPLE-EXPECT	900	0.0828	0.0586	0.0554
SAMPLE-MIN	900	0.0680	0.0577	0.0806

Figure 3: Relative error rates for the different algorithms on three experiments, varying the probability that c_1 wins

We considered three cases, varying the probability that the target candidate c_1 will be the winning candidate from 0.001 to 0.1. Figure 3 shows the results for an experiment in which the number of samples was chosen to ensure that each algorithm had the same running time. We report the relative error of the estimated expected value of the feature x of c_1 compared to an estimate of ground truth obtained by taking 1,000,000 samples and keeping the ones where c_1 wins. The results are averaged over 10,000 runs. Also shown is the number of samples used by each algorithm.

We see that in the first case, where there is a very low probability of c_1 winning, algorithm SAMPLE-SIMPLE is a very good approximation and has the best performance. On the other hand, in the third case where the probability is quite high, SAMPLE-SIMPLE performs very poorly indeed. These results are as one would expect, because the approximation of Equation 5 assumes that the score of the target candidate is very small relative to the scores of the other candidates. Meanwhile, algorithms SAMPLE-EXPECT and SAMPLE-MIN, which do not make this assumption, significantly outperform IMPORTANCE-SAMPLE in all three cases. SAMPLE-MIN is slightly better for low probabilities of c_1 winning and SAMPLE-EXPECT is better for high probabilities.

Algorithm	Signed Error	Absolute Error
IMPORTANCE-SAMPLE	-0.0069	0.0232
SAMPLE-SIMPLE	0.0242	0.0277
SAMPLE-EXPECT	0.0257	0.0259
SAMPLE-MIN	0.0239	0.0239
SAMPLE-GENERIC	-0.0062	0.0212

Figure 4: Signed and absolute errors for the five algorithms on the Colored Trails model

Finally, we compare the algorithms, and also SAMPLE-GENERIC, on a richer model learned from the Colored Trails data. Each instance now consisted of the candidate proposals for a Colored Trails game. Because all proposals have the same prior, we can use SAMPLE-GENERIC here. Note that now the candidates are correlated with each other — if a game has one high-scoring proposal it is likely to have more. This fact will hurt the algorithms that only sample c_1, because they will not discover the correlations between candidates. The results are shown in Figure 4. The figure shows both the average signed errors, and the average absolute errors, of the estimate of the individual utility feature of the winning proposal, taken over 100 runs. SAMPLE-GENERIC is the best performer. Examining the signed errors, we see that they are very high and positive for all three algorithms that only sample c_1. These three algorithms are consistently overestimating, and are suffering from an upward bias which causes them to underperform the other two algorithms even though they use many more samples.

6. CONCLUSION AND FUTURE WORK

We presented a model used to predict selection processes, and two learning algorithms for learning the parameters of a selection process from data. We showed that a model which reasons about dependencies between candidates is superior to one that does not. We also presented several methods for inferring the properties of a candidate from the fact that it won, and showed that algorithms that only sample the target candidate perform much better than an algorithm that samples all candidates.

In other domains, it is possible for zero or more than one candidates to be chosen but the choice of one candidate does have a bearing on whether other candidates are chosen. For example, consider hiring a candidate to fill a faculty position. If the search turns up more than one outstanding candidate, more than one offer will be made. If the committee feels that there are no good candidates, no offer will be made. But in general the candidates are in competition with each other, and if an offer is made to one candidate that reduces the probability that one will be made to another candidate. In future, we intend to model these types of processes.

Selection processes can naturally be accommodated in probabilistic models. There is a close relationship between selection processes and probabilistic relational models (PRMs) [9]. In a PRM, an object X may be related to another object Y via a complex attribute. We may have uncertainty as to which other object Y the object X is related to. This is known as reference uncertainty. In a PRM, this uncertainty has been traditionally modeled by specifying an explicit probability distribution over the possible objects Y. The distribution over possible objects traditionally does not depend on the attributes of the possible objects. However, it is natural to make the choice of possible object Y a selection process, which depends on the attributes of the possible objects. For example, we may have a *Job* object. This object will have a complex attribute *Candidate*, which is a multi-valued attribute. The *Job* will be related to a number of *Person* objects via the *Candidate* attribute. In addition, the *Job* object will have an *Occupant* attribute, which is single-valued. We have reference uncertainty with regards to the value of the *Occupant* attribute. Modeling *Occupant* as a selection process, will make the candidates the values of the *Candidate* attribute, and the winner depends on the features of the candidates.

7. ACKNOWLEDGMENTS

We thank Barbara Grosz, Adam Juda and Loizos Michael for helpful comments and discussions. This work was supported in part by NSF grant CNS-0453923. Development and dissemination of the Colored Trails formalism is supported in part by DARPA.

8. REFERENCES

[1] R. Arunachalam, J. Eriksson, N. Finne, S. Janson, and N. Sadeh. The TAC supply chain management game. Technical report, Swedish Institute of Computer Science, 2003.

[2] M. Bazerman. *Judgment in Managerial Decision Making*. Wiley, 2001.

[3] C. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, 2003.

[4] J. Cavano. Computers as partners: A technology forecast for decision-making in the 21st century. In *Proceedings of the Twenty-Third Annual International Computer Software and Applications Conference*, 2001.

[5] U. Chajewska and D. Koller. Utilities as random variables. In *Proc. 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

[6] U. Chajewska, D. Koller, and D. Ormoneit. Learning an agent's utility function by observing behavior. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

[7] R. Das, J. E. Hanson, J. O. Kephart, and G. Tesauro. Agent-human interactions in the continuous double auction. In *IJCAI*, pages 1169–1187, 2001.

[8] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1), 1977.

[9] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[10] Y. Gal, A. Pfeffer, F. Marzo, and B. Grosz. Learning social preferences in games. In *Proc. 19th National Conference on Artificial Intelligence (AAAI)*, 2004.

[11] B. Grosz, S. Kraus, S. Talman, and B. Stossel. The influence of social dependencies on decision-making. Initial investigations with a new game. In *Proc. 3rd International Joint Conference on Multi-agent systems (AAMAS)*, 2004.

[12] J. Henrich, R. Boyd, S. Bowles, H. Gintis, E. Fehr, and C. Camerer, editors. *Foundations of Human Sociality: Economic Experiments and Ethnographic Evidence in Fifteen Small-Scale Societies*. Oxford University Press, 2004.

[13] J. Kagel and A. Roth, editors. *The hanbook of experimental economics*. Princeton University Press, 1995.

[14] R. Lawrence. A machine-learning approach to optimal bid pricing. In *Proceedings of the Eighth INFORMS Computing Society Conference on Optimization and Computation in the Network Era, Arizona, 2003. 6*, 2003.

[15] G. Loewenstein, M. Bazerman, and L. Thompson. Social utility and decision making in interpersonal contexts. *Journal of Personality and Social psychology*, (57)(3):426–441, 1989.

[16] J. MacKie-Mason, A. Osepayshivili, D. Reeves, and M. Wellman. Price prediction strategies for market-based scheduling. In *Proc. of 18th International Conference on Automated Planning and Scheduling*, June 2004.

[17] R. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113, 1992.

[18] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 2000.

[19] D. Pardoe and P. Stone. Bidding for customer orders in TAC SCM. In P. Faratin and J. Rodriguez-Aguilar, editors, *Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems (AMEC 2004)*, volume 3435 of *Lecture Notes in Artificial Intelligence*, pages 143–157, Berlin, 2005. Springer Verlag.

[20] R. Schapire, P. Stone, D. McAllester, M. Littman, and J. Csirik. Modeling auction price uncertainty using boosting-based conditional density estimation, 2002.

[21] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In K. P. Sycara and M. Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 301–308, New York, 1998. ACM Press.