Learning an Agent's Utility Function by Observing Behavior

Urszula Chajewska URSZULA@CS.STANFORD.EDU
Daphne Koller KOLLER@CS.STANFORD.EDU
Dirk Ormoneit ORMONEIT@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305-9010

Abstract

This paper considers the task of predicting the future decisions of an agent A based on his past decisions. We assume that A is rational — he uses the principle of *maximum expected utility*. We also assume that the probability distribution P he assigns to random events is known, so that we need only infer his utility function $\mathbf{u}$ to model his decision process. We consider the task of using A's previous decisions to learn about $\mathbf{u}$. In particular, A's past decisions can be viewed as constraints on $\mathbf{u}$. If we have a prior probability distribution $p(\mathbf{u})$ over $\mathbf{u}$ (e.g., learned from a set of utility functions in the population), we can then *condition* on these constraints to obtain a posterior distribution $q(\mathbf{u})$. We present an efficient Markov Chain Monte Carlo scheme to generate samples from $q(\mathbf{u})$, which can be used to estimate not only a single "expected" course of action for A, but a distribution over possible courses of action. We show that this capability is particularly useful in a two-player setting where a second learning agent is trying to optimize her own payoff, which also depends on A's actions and utilities.

1. Introduction

Consider the problem of trying to predict the future actions of an agent A. There are many settings in which this capability is useful. In a cooperative setting, we might want to help the agent make good decisions. In a more competitive setting, we might want to predict the agent's actions so as to better optimize our own payoffs.

Our approach is based on the assumption that A is a rational decision maker. According to decision theory, rational decision making amounts to the maximization of the expected utility (von Neumann & Morgenstern, 1947). This paradigm requires two types of information: the *probabilities* and the *utilities* of all possible outcomes of the decision problem. Probabilities can often be estimated using known techniques from machine learning and statistics. Thus, given a history of our interactions with A, we can often assume that we know the probabilistic model that A uses.

The acquisition of the utility function is typically a much more formidable task. Even in a cooperative setting where A is willing to answer questions about his preferences, the task of utility elicitation is cognitively difficult and error prone; there are many elicitation techniques that often produce very different results when applied to the same person (Fromberg & Kane, 1989). Furthermore, for many real-life decision problems, the outcome space is very large, and there is a limit to the number of questions a given user is willing to answer. Finally, in non-cooperative situations, where any information may constitute a strategic advantage, we cannot rely on utility elicitation at all.

An alternative to active solicitation is to try and estimate A's utility function by passively observing his behavior over time. This approach was proposed by Ng and Russell (2000) in the context of *inverse reinforcement learning* in *Markov Decision Processes (MDPs)*. As they show, the agent's decisions can be viewed as a set of linear constraints on the space of possible utility (reward) functions. The set of utility functions consistent with the constraints is infinite. Ng and Russell propose a set of heuristics that attempt to select one specific utility function from this set.

Our approach differs from the work of Ng and Russell, and from traditional approaches to decision making, in a very important way: To account for the uncertainty regarding the agent's utility function, we consider the utility to be a random quantity that is governed by a prior probability distribution. This assumption is often quite reasonable in practice: For example, we might collect a database of (partially specified) utility functions for a given type of decision problem and apply machine learning algorithms to estimate the empirical distribution of utility functions in the population (Chajewska & Koller, 2000). When a new agent A is encountered, this estimate serves as a prior distribution $p(\mathbf{u})$ over A's utility function. As we observe A's behavior, we use the constraints implied by his choices to condition this distribution, obtaining a posterior distribution $q(\mathbf{u})$.

This formalism provides a coherent framework for interacting with the agent A. In a cooperative setting, it can help a decision support system to find a (nearly) optimal strategy for agent A without the need for full utility elicitation (Chajewska et al., 2000). In this paper, we extend this idea to a more general, not necessarily cooperative setting. In our framework, an informed agent B is interacting with an oblivious agent A, each of which is trying to maximize their own utility. The oblivious agent A is unaware of the informed agent's efforts to adapt her strategy in response to his own behavior; rather, he believes that B's decisions are selected at random, using some prespecified strategy. Such situations are in fact quite common in practice, for example in electronic commerce or in automated recommendation systems, where an agent interacts with a human who may be unaware of the agent's learning efforts.

2. Decision models and simple games

A *sequential decision problem* for a single agent consists of several decisions, taken in sequence, often with some information revealed between one decision and the next. There are many possible models for this type of situation, including *influence diagrams*, *Markov decision processes*, and *decision trees*. For finite horizon situations, decision trees are a very general representation from a semantic perspective. Any influence diagram can be expanded into a decision tree (Pearl, 1988), and any MDP can be unrolled into a tree whose depth depends on the horizon. Of course, these transformations lose some of the structure of the representation, but they are equivalent in terms of the semantics of the decision task. Hence, we define decision problems in terms of decision trees.

A decision tree is a rooted tree containing two types of interior nodes: agent choice points and nature choice points. Let $S(n)$ denote the set of successors of node n. At agent nodes, the agent selects a move, i.e., a node in $S(n)$. Nature nodes n are associated with a probability distribution $\pi_N(n)$ over $S(n)$. The leaves of the tree ℓ are annotated with *utility values* $U(\ell)$ for the agent. A deterministic strategy $\mathbf{s}$ for the agent is a mapping that assigns to each of his nodes n a particular successor in $S(n)$. Each strategy has an *expected utility* $\mathrm{EU}(\mathbf{s})$, which is a weighted average of the utility values at the leaves. The weights correspond to the probability of reaching each leaf given $\mathbf{s}$. The optimal strategy $\mathbf{s}^*$ is the one that maximizes the agent's expected utility; it can (in principle) be computed using a process of backward induction up the tree (also known as the expectimax algorithm).

Recall that our goal is to reach conclusions about the agent's utility function U based on observing his actions. This problem is of limited interest if the agent's utility function is specific to one particular decision task. Thus, we often assume that the utility function is derived from more general components, that reflect the agent's general preferences regarding various aspects of the situation (Keeney & Raiffa, 1976). For example, consider a problem where we are trying to learn the utility function of a customer of an online bookseller. Over time, the customer might be offered different books at various prices, which he may like to varying degrees. The customer's overall utility might be a combination of *subutility components*, corresponding to such aspects as the price he paid or his enjoyment of the book. More formally, we assume that the agent's utility function U is *linearly additive*: There exists some set of subutilities $\mathbf{u} = (u_1, \ldots, u_m) \in [0,1]^m$ such that for any leaf ℓ in a decision tree, $U(\ell) = \sum_{j=1}^{m} \alpha_{\ell,j} u_j$. We assume that the decomposition of U into subutilities is specified as part of the decision problem, so that the coefficients $\alpha_{\ell,j}$ are known. An agent can encounter multiple decision problems, but his utility function is always composed of the same set of subutilities. An advantage of this formulation is that we can acquire information about the agent's subutilities in one decision problem, and then apply it in another. We assume that the subutility functions are *normalized*, i.e., restricted to $[0,1]$ range. Since utility functions are determined only up to positive linear transformation, the normalization assumption is necessary for the purpose of comparison between utilities of different individuals.

The discussion above assumed that we have a single active agent A, and a passive observer whose only goal is to learn A's utility function. However, it is rarely possible or necessary to learn A's exact utility function. The observer typically needs to deal with A in the context of a particular interaction, and learning A's utility function is only useful inasmuch as it helps predict A's actions in that context. To model this type of interaction, we generalize our framework to handle a particular type of two-agent "game", where the two agents have very different "levels of awareness". Our game contains one *informed* or *strategic* player, B, who is aware of the nature of the strategic interaction, and one *oblivious* player, A, who does not perceive the interaction as a game — he attributes the strategic player's actions to random moves by nature. We note that the asymmetry is an important simplification which is necessary for making the problem tractable to a simple learning approach; if both players attempt to

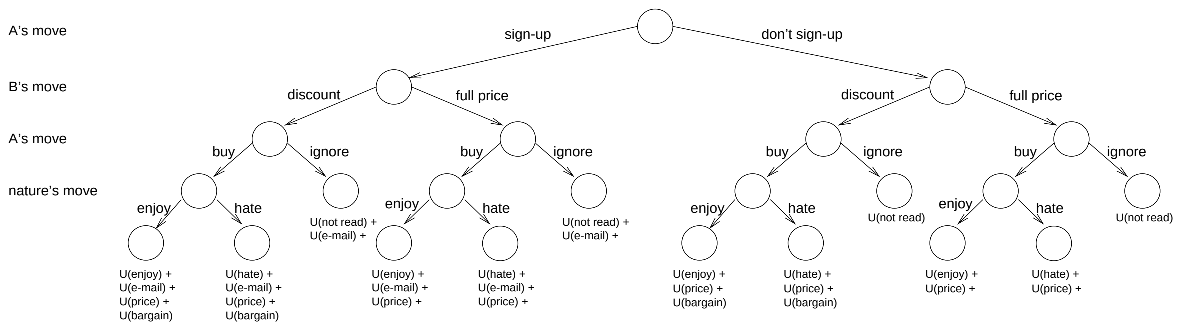


Figure 1. Bookseller example

learn, then their anticipation of their mutual learning strategies might lead to infinite cycles (Selten, 1991; Fudenberg & Levine, 1998).

More formally, the game tree now contains decision nodes for both players. However, the oblivious player associates with each of the strategic player's nodes n a probability distribution $\pi_S(n)$ which is known to both players. In this setting, the oblivious player selects his strategy by maximizing his expected utility, as in the single-agent case. He treats all nodes except his own as belonging to nature, where π_N and π_S specify the distribution at these nodes. The strategic player, on the other hand, also maximizes his expected utility, but is aware of the fact that his utility value depends on the simple player's actions, and perhaps even on the other player's actual utilities. However, he does not know the simple player's utility. Therefore, he cannot predict the simple player's actions with certainty, nor necessarily even the expected utility of his own actions.

This asymmetric type of interaction, although fairly restricted, arises in a variety of situations. As our running example, let us return to the interaction between a bookseller and her online customers, illustrated in Fig. 1. Here, the bookseller B is considering whether to offer a frequent customer A a discount on a newly published title. B would prefer to sell the book at the full price, but selling at a discount is better than not making a sale at all. B can furthermore alert her frequent customers by e-mail and notify them of the book's availability. However, this is only possible if the customer A has signed up for this e-mail service. As discussed above, A's utility is a linear function of subutilities: the price he pays, whether or not he signed up for e-mail notification, the extent to which he enjoys the book, and the satisfaction from obtaining a bargain. We assume that both players can predict from A's previous purchases the chances that he will enjoy a particular title. A is the oblivious player, and considers the discount offer (B's move) to be an act of nature (or a result of a predefined strategy, possibly determined by other factors not known to A and not affected by his actions). For example, A may assume that the bookseller will offer a discount on any specific book to some (randomly selected) fraction of his frequent customers or that the discount will be offered to all frequent customers on a fraction of titles. Clearly, B's utility depends on the actions taken by A, which in turn depend on A's unknown utility function.

3. From behavior to utility constraints

As we discussed above, the goal of the informed agent is to learn about the utility function of the oblivious agent A. In other words, we assume that A's subutilities $\mathbf{u} = (u_1, \ldots, u_m)$ are unknown. If we assume that A is an expected utility maximizer, we can derive constraints on A's utilities by observing his actions. In this section, we show that A's decisions can be used to derive linear constraints on $\mathbf{u}$. Our result is similar to that of Ng and Russell (2000), but differs in two important ways. First, our linear constraints apply not only to an MDP, but to any general decision problem represented as a tree. Second, we provide an alternative solution to the problem of deriving constraints in cases where we do not observe A's full strategy.

3.1 Fully observed strategies

What can we conclude about $\mathbf{u}$ by passively observing a utility maximizing player whose utility function is defined by $\mathbf{u}$? The answer to this question is straightforward in principle if the optimal strategy $\mathbf{s}^*$ can be observed entirely: In this case, given the optimality of $\mathbf{s}^*$, we obtain the set of optimality conditions:

$$\forall_{\mathbf{s} \in \mathbf{S}} \mathrm{EU}(\mathbf{s}^*) \geq \mathrm{EU}(\mathbf{s}) \tag{1}$$

where $\mathbf{S}$ is the space of pure strategies for A. Note that Eq. (1) translates into a set of linear constraints on the utility vector $\mathbf{u}$ due to the linearity of the expectation operator. Hence Eq. (1) defines a polytope $\mathcal{U}^* \subseteq [0, 1]^m$ which contains all of the possible subutility values $\mathbf{u}$ consistent with the observed behavior.

The problem with using Eq. (1) in an algorithmic context is that the number of strategies $\mathbf{s}$ and hence the number of constraints implied by Eq. (1) may be extremely large in practice. However, we can use the structure of the decision tree to obtain a set of equivalent constraints which is exponentially smaller. The basic idea is quite simple: we simulate the backward induction in the decision tree, deriving the constraints along the way. We define a set of variables V_n representing A's expected utility at each node n, assuming he acts optimally from then on. We then define constraints in terms of these V_n.

We start with an empty constraint set $\mathcal{C}$. Working backwards from the leaves, consider a node n. If n is a leaf ℓ, we are given the linear expression $V_\ell[\mathbf{u}] = \sum_{j=1}^{m} \alpha_{\ell,j} u_j$ involving the subutilities $u_1, \ldots, u_m$. If n is an internal node, and A perceives n to be a chance node (whether truly a chance node or a decision node of the informed player), we are given fixed probabilities $p_{n'}$ for each successor $n' \in S(n)$. We can then define V_n in terms of the $V_{n'}$: $V_n[\mathbf{u}] = \sum_{n' \in S(n)} p_{n'} V_{n'}[\mathbf{u}]$. If n is a decision node for A, the observed strategy $\mathbf{s}^*$ tells us which child $\sigma(n)$ of n is chosen by the agent. We define $V_n[\mathbf{u}] = V_{\sigma(n)}[\mathbf{u}]$, and then add to $\mathcal{C}$ a set of constraints that implies consistency with the observed behavior at n: $V_n[\mathbf{u}] \geq V_{n'}[\mathbf{u}]$ for all $n' \in S(n)$.

The number of constraints generated in this fashion is linear in the size of the decision tree — exponentially less than the number of possible deterministic strategies over the tree, which is the number of inequalities implied by Eq. (1). Moreover, the constraints will be at most as large and often much smaller in size. Nonetheless, both formulations describe the same region of the utility space $\mathcal{U}^*$.

3.2 Partial strategy observations

In most cases, it is unrealistic to assume that $\mathbf{s}^*$ can be observed entirely. If we have only a single observation of A's behavior in a particular decision setting, we will, by force, only observe one trajectory. For example, in the bookseller scenario of Section 2, if A signs up for email notification, B will never learn what would have happened in the other branch of the decision tree. Even if we have multiple observations of the agent in the same decision problem, some parts of the decision tree may never be visited, because the decision leading there may be dominated by another and will never be chosen. Therefore, it is critical to have an approach that allows us to deal with partial strategy observations.

To understand the difficulty from a technical perspective, consider again the backward induction algorithm of Section 3.1. In A's decision nodes we may not know which of n's children was chosen by A. Hence, we cannot determine the expression $V_n[\mathbf{u}]$, nor the associated set of constraints. When the strategy is unobserved, we can only use the fact that A is rational to conclude that:

$$V_n[\mathbf{u}] = \max_{n' \in S(n)} V_{n'}[\mathbf{u}]. \qquad (2)$$

The difficulty is that Eq. (2) is non-linear in the subutility variables, leading to a non-convex region $\mathcal{U}^*$ of feasible utilities. Thus, $\mathcal{U}^*$ can no longer be expressed using a set of linear inequalities, and in fact, can get exponentially complex. For our algorithm (below), as well as in the work of (Ng & Russell, 2000), it is critical that our feasible region be described compactly as a set of linear inequalities.

One simple approach is to relax the constraints implied by Eq. (2) in a way that is consistent with it yet gives rise to linear constraints. In detail, we derive (linear) upper and lower bounds on the expression Eq. (2), and use these bounds to specify the induced constraints on $\mathbf{u}$. Recall that each expression $V_n[\mathbf{u}]$ is a linear function of the form $\sum_{j=1}^{m} \alpha_{n,j} u_j$. As the u_j's are non-negative, one possible relaxation of Eq. (2) is to replace it with the two expressions

$$\begin{aligned} \overline{V}_n[\mathbf{u}] &= \sum_{j=1}^{m} \max\{\alpha_{n',j} \ : \ n' \in S(n)\} \cdot u_j, \quad (3) \\ \underline{V}_n[\mathbf{u}] &= \sum_{j=1}^{m} \min\{\alpha_{n',j} \ : \ n' \in S(n)\} \cdot u_j, \quad (4) \end{aligned}$$

which are linear in the u_j's. Formally, Eq. (3) and Eq. (4) satisfy $\underline{V}_n[\mathbf{u}] \leq V_n[\mathbf{u}] \leq \overline{V}_n[\mathbf{u}]$, and can hence be used to define relaxed constraints in a modified algorithm.

Our tree propagation algorithm stores two different expressions — one lower and one upper bound — at each node of the tree. The propagation of these expressions is similar to the backward induction algorithm of Section 3.1, replacing $V_n[\mathbf{u}]$ with its upper or lower bound, as follows. At each leaf ℓ, we define $\underline{V}_\ell[\mathbf{u}] = \overline{V}_\ell[\mathbf{u}] = \sum_{j=1}^{m} \alpha_{\ell,j} u_j$. For a node that A perceives to be a chance node, we take expectation for both $\overline{V}$ and $\underline{V}$, in the obvious way. At each *observed* decision node with choice $\sigma(n)$, we define

$$\overline{V}_n[\mathbf{u}] = \overline{V}_{\sigma(n)}[\mathbf{u}]; \quad \underline{V}_n[\mathbf{u}] = \underline{V}_{\sigma(n)}[\mathbf{u}]$$

and then add to the constraint set $\mathcal{C}$ the constraints:

$$\forall n' \in S(n) \ : \ \overline{V}_n[\mathbf{u}] \geq \underline{V}_{n'}[\mathbf{u}].$$

At each *unobserved* decision node, we define $\underline{V}_n[\mathbf{u}]$ and $\overline{V}_n[\mathbf{u}]$ using Eq. (3) and Eq. (4). After traversing the complete tree, the constraints in $\mathcal{C}$ define a convex region $\mathcal{U}_{\mathcal{C}}$ which is a superset of the *true* region of feasible utilities, $\mathcal{U}^*$.

4. Probabilistic Model of Utilities

The constraints described in the previous section specify a region of feasible utility functions (or a relaxation thereof). We can then use this region to derive an estimate of the true utility function, e.g., by finding the centroid of the region, or using the procedure of (Ng & Russell, 2000). However, any particular choice is, by force, somewhat arbitrary, and is unlikely to be the agent's actual utility function. Indeed, in the case where we are using the relaxed constraints as the basis for our estimate, we are only guaranteed that our estimate will be contained in $\mathcal{U}_C$, not in $\mathcal{U}^*$; in other words, our estimate may not even be consistent with the observed behavior of A.

A more robust approach is to explicitly represent our uncertainty about A's utility function, by maintaining a *probability distribution* over the utility space. In other words, we view the subutilities $\mathbf{u}$ as a continuous-valued random vector, and use a probability density function (PDF) $p(\mathbf{u})$ to represent our beliefs about its possible values. As shown by Chajewska and Koller (2000), we can obtain such a prior by applying density estimation to a database of partially or fully elicited utility functions of many agents. The resulting distribution is an estimate of the utility functions in the population, and can be used as a prior distribution over the utility function of a newly encountered agent.

Our observations of the agent's actions can be viewed as evidence regarding $\mathbf{u}$: Certain utility functions $\mathbf{u}$ are consistent with the agent's actions, whereas others are not. We can *condition* our prior $p(\mathbf{u})$ on this evidence, to derive a more informed *posterior* $q(\mathbf{u})$. Then we use $q(\mathbf{u})$ as the basis for reasoning about the future behavior of A, e.g., by using the posterior mean of the distribution, as an alternative to the utility estimate proposed by Ng and Russell.

Unfortunately, even if $p(\mathbf{u})$ is a "nice" distribution with a compact closed form representation, the posterior $q(\mathbf{u})$ can be quite complex. The problem is that irregular polytopes (even convex ones) are computationally difficult to deal with. For example, even estimating the volume of a polytope is a hard computational problem (Bárány & Füredy, 1986). Fortunately, we can use Markov Chain Monte Carlo (MCMC) techniques to generate a set of samples from the posterior distribution $q(\mathbf{u})$ in an efficient fashion. As we discuss in the next section, a set of samples from $q(\mathbf{u})$ can be substantially more useful than a single point estimate of the agent's utility.

Our approach is based on the MCMC algorithms of Applegate and Kannan (1991) for estimating the volume of a polytope on the one hand, and for generating samples from a log-concave density function (such as a Gaussian) on the other. Applegate and Kannan show that both of these algorithms are rapidly mixing, so that the number of sampling steps required for convergence to a stationary distribution is polynomial in the dimension of the polytope. We believe that a similar analysis can be applied to our algorithm, which simply combines the two algorithms into one.

In detail, we use a Metropolis-Hastings algorithm, which traverses the convex set $\mathcal{U}_C$ based on the distribution p. We first construct a regular grid in the m-dimensional hyper-cube $[0,1]^m$. This grid defines a partition of $[0,1]^m$ into cubes. Then, starting from some arbitrary starting point $\mathbf{x}^{(0)}$ in $\mathcal{U}_C$, we carry out a sequence of MCMC steps:

1. Starting from the current grid location $\mathbf{x}^{(t-1)}$, choose a candidate successor state $\mathbf{y}$ as follows. With probability $1/2$, $\mathbf{y} = \mathbf{x}^{(t-1)}$. With probability $1/2$, $\mathbf{y}$ is chosen uniformly from among $\mathbf{x}^{(t-1)}$'s $2m$ neighbors, so that for each possible neighbor $\mathbf{z}$, the probability that $\mathbf{y} = \mathbf{z}$ is $1/4m$.
2. If $\mathbf{y}$ is located outside $\mathcal{U}_C$, stay at the current position, i.e., set $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}$. Otherwise, accept the new location with probability $\min\left\{1, \frac{p(\mathbf{y})}{p(\mathbf{x}^{(t-1)})}\right\}$.

After an initial "mixing phase", we store samples $\mathbf{u}^{(t)}$ from the Markov chain at regular intervals b.[1] We choose $\mathbf{u}^{(t)}$ from the subcube corresponding to $\mathbf{x}^{(t)}$. The stationary distribution of this Markov chain is $p(\mathbf{u})$ constrained to $\mathcal{U}_C$. In the case where we have fully observed strategies, and $\mathcal{U}_C = \mathcal{U}^*$, the stationary distribution is $q(\mathbf{u})$. Thus, after some number of steps, we will start collecting samples from approximately the desired posterior.

In the case where we observe only partial trajectories, our points are in $\mathcal{U}_C$ but may not be in $\mathcal{U}^*$. However, we can test each utility function $\mathbf{u}$, to see whether it is compatible with A's observed behavior, and reject the ones that are not. This elimination can be realized by explicitly solving the decision problem for the sample utility values and checking their consistency with the observed behavior. Note that we are using the convex set $\mathcal{U}_C$ merely as a computational tool to construct a rapidly mixing Markov chain. We have no theoretical guarantees about the fraction of points that will be rejected in the subsequent selection process. (See Section 6 for some empirical results.)

5. Optimizing relative to utility samples

The set of samples generated from the posterior density $q(\mathbf{u})$ may be useful for several purposes: First, it

[1]We can also store all of the samples after the mixing phase, but the computational cost of using all of them in our algorithm is too large.

could be used to approximate the posterior mean, simply by averaging the sample points $\mathbf{u}^{(t)}$. The mean is a point estimate of agent A's utility. It can be used as an alternative to the point estimate provided by the approach of Ng and Russell. Rather than using heuristics to select among the otherwise indistinguishable elements of $\mathcal{U}^*$, it uses the prior density $p(\mathbf{u})$.

However, a single point $\mathbf{u}^*$ loses the information concerning our certainty about the estimate. This information might be critical in a situation where the agent B needs to act based on her estimate of A's future behavior: An action that might be optimal with respect to A's behavior if his utility is $\mathbf{u}^*$ might be highly suboptimal relative to slightly different utilities.

A more robust approach for B is to try and plan her actions, explicitly taking into consideration her uncertainty about A's utility function. While this approach is infeasible relative to the entire distribution $q(\mathbf{u})$, we can use the samples from the distribution as a computationally convenient approximation. We now show how these samples can be used effectively to optimize B's behavior, in the context of the asymmetric games discussed in Section 2.

An appropriate formal framework for representing B's decision problem is a game with imperfect information, where an initial move of "nature" determines the utility function of A according to the probability distribution $q(\mathbf{u})$. To avoid dealing with infinite trees, we use our samples $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(K)}$ as a discrete set of initial states, representing our approximation to $q(\mathbf{u})$. Nature therefore selects each of our K MCMC samples with probability $1/K$. Each choice $\mathbf{u}^{(k)}$ is associated with a first-level subtree T_k, which is precisely our original game tree, but using $\mathbf{u}^{(k)}$ to specify the subutility variables for A.

The informed agent B does not observe the choice of nature, and hence does not know her position in the game tree. Formally, there is a collection of *information sets*, representing B's uncertainty in this case. Each information set contains a set of nodes among which B cannot distinguish. In this case, each information set will contain K nodes — one from each first-level subtree T_k — the set of nodes whose path (aside from the initial nature move) is the same.

Given the asymmetric information, players A and B have different perspectives on the resulting game. From A's perspective, the game reduces to a set of single-player decision trees, each of which can be solved in isolation by the backward induction algorithm. The situation for the informed agent is more complex. Since B is aware of the structure of the game tree and of the utility function $\mathbf{u}^{(k)}$ in each subtree T_k, she can predict A's actions in each subtree; however, she does not know which subtree she is actually in. As B cannot distinguish between the nodes in an information set, she must choose the same action in each of them. To optimize her strategy, she must compute the expected utility of each of her actions at each information set, where the expectation is taken relative to the possible utility values for A.

Moreover, B, being able to predict A's behavior in each subtree T_k, can conclude that certain actions are incompatible with $\mathbf{u}^{(k)}$. In this case, by the time she gets to an information set, she might be able to eliminate nodes inconsistent with the assumption of rationality for A. Her choice of action should be optimal relative only to the remaining consistent nodes.

```
Traverse the tree bottom-up computing each node's $n$ value $v_n^B$ for the
strategic player $B$:
Leaf node:             $v_n^B$ is given
Chance node:           $v_n^B = \sum_{n' \in S(n)} p(n') v_{n'}^B$
$A$'s decision node:   $v_n^B = v_{\sigma(n)}^B$ $(\sigma(n) = \arg\max_{n' \in S(n)} (v_{n'}^A))$
$B$'s decision node:
    for all nodes $n_k$ in $B$'s information set $N$
        check consistency of $\mathbf{u}^{(k)}$ with $A$'s behavior on path to $n_k$
        if not consistent, eliminate $n_k$ from $N$
    for each decision $d$ available for nodes in $N$
        $EV(d) = \frac{1}{|N|} \sum_{n_k \in N} v_{\rho(n_k, d)}^B$
        where $\rho(n_k, d)$ is the node reached from $n_k$ by taking $d$
    choose $d^* = \arg\max EV(d)$
    $v_{n_k}^B = v_{\rho(n_k, d^*)}^B$
```

Figure 2. B's strategy computation

The algorithm for computing the optimal strategy for agent B is shown in Figure 2. At the end of each game in our repeated interaction, A's observed behavior in that game is used to define new constraints, which are added to the constraint set $\mathcal{C}$ guiding the selection of sampled utility functions for the next interaction.

6. Experiments

To test the described approach in practice, we implemented a simulation of the bookseller example described above. This game has 5 subutility variables, shown in Fig. 1, so $\mathbf{u}$ is a five-dimensional vector. The game is played repeatedly, with each repetition having different parameters: item price (full and discounted), and probability that the user will enjoy the book. This scenario reflects a situation where the user enters the web-site of the bookseller repeatedly; each time he is confronted with a different book at a different price. A practical advantage of this experimental design is that the constraints on the utilities change slightly during each repetition of the game, providing additional information about the set $\mathcal{U}^*$.

We experimented both with fully observed and partially observed strategies. In both cases, we simulated several instances of the bookseller game at each step. For each instance we formulated an incremental set of linear constraints derived from the play of the game. In

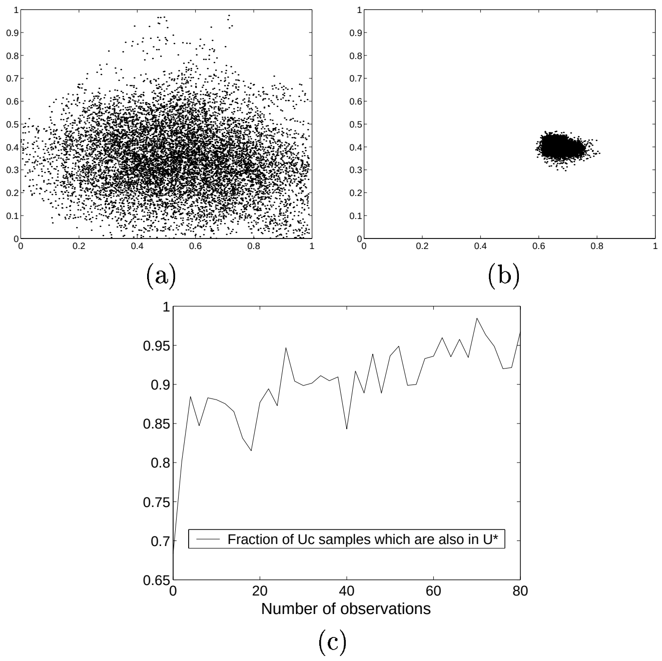


Figure 3. (a) & (b): Projection of the MCMC samples onto the $u_1 - u_2$ plane ('enjoy' and 'hate' subvariables) — (a) after 1 observation; (b) after 17 observations. (c) Fraction of MCMC samples within $\mathcal{U}^*$.

the fully observed case, we used A's true full strategy. In the partially observed case, we used the "trajectory" observed in the actual play of the game. After adding each new set of constraints for the observed instances, we ran the MCMC algorithm of Section 4 to produce a posterior mean estimate $\mathbf{u}^*$ of the customer's utility function. We used a burn-in phase of 10,000 steps, to allow convergence to the stationary distribution; afterwards, we ran the Markov chain for 100,000 steps, selecting samples at intervals of 10. (These parameters were selected after we experimentally determined they are sufficient to adequately cover the space.)

We first consider the behavior of our MCMC algorithm. Figure 3 illustrates the admissible utility region $\mathcal{U}^*$ at different stages, in the fully observable case. We show the samples generated by the MCMC algorithm, projected onto the u_1–u_2 plane: (a) shows the samples in a relatively early stage of learning where only few constraints are present, whereas (b) shows the samples towards the end of learning. Note that the utility region is narrowly constrained and densely sampled at this stage. In the partially observable case, the MCMC samples are generated from the relaxation $\mathcal{U}_C$. These samples are not necessarily consistent with A's behavior, so we tested each generated sample for consistency with observed behavior, ensuring that our actual sample set contains only samples from $\mathcal{U}^*$. In Figure 3(c), we show the fraction of the generated samples in $\mathcal{U}_C$ that are also in $\mathcal{U}^*$. As can be seen, the fraction of sample points that are preserved is relatively large, and approaches one as the number of steps increases. We suspect that this improvement is due to the fact that observations later in the learning process help compensate for over-relaxation in the early periods.

Next, we tested the performance of our overall learning framework. Figure 4(a) and (b) shows the error as the number of observed game instances increases, for the fully and partially observable case. In both plots, the x axis is the number of observed game instances. The solid curve represents the average Euclidean distance between the true utility function $\mathbf{u}$ and the posterior mean estimate $\mathbf{u}^*$. The dotted curve is the strategy difference: the average number of states with deviating decisions between the strategy we predicted based on $\mathbf{u}^*$ (applied to 20 game instances we did not observe) and $\mathbf{s}^*$ — the best strategy for $\mathbf{u}$. We also show the variance as error bars.

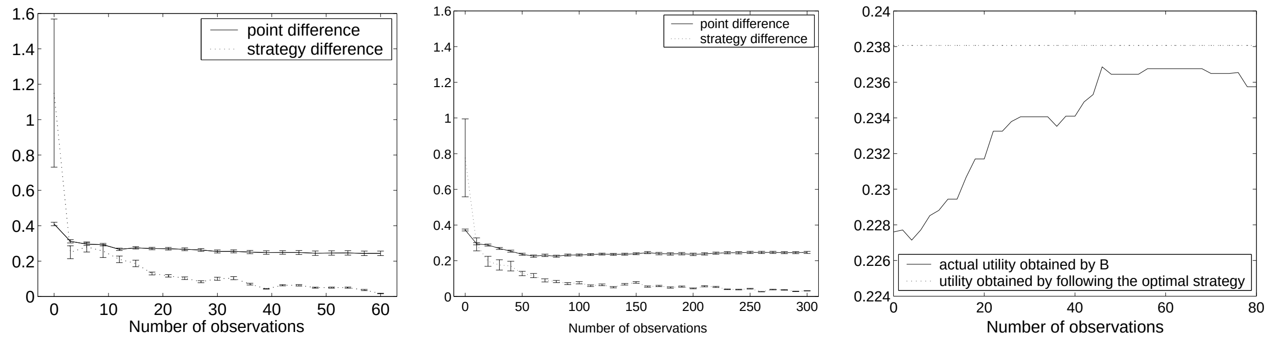
We can see that the errors decrease drastically over time. In both cases, the error in the utilities remains bounded away from zero (at about 0.22) whereas the error in the strategies goes to zero. This can be explained by the fact that there will always be some residual uncertainty about the utilities that cannot be resolved by strategy observations. Insofar as these ambiguities lead to different behavior, however, we will eventually detect them and hence identify the optimal strategy $\mathbf{s}^*$ asymptotically. In the partially observable case the strategy prediction error takes considerably longer to converge to zero.

Finally, Figure 4(c) shows the results for the optimized game play from the perspective of the informed player B. The graph shows the average utility B derives from playing a set of 20 games as a function of the number of games observed up to that point. We can see that B's performance improves over time. The dotted line indicates the optimal realizable utility if the true utility of A were known and is used as a benchmark. The fact that the actual utility approaches the benchmark indicates that the strategy of A is eventually predicted with high accuracy.

7. Conclusion and Extensions

We presented a new algorithm to learn the utility function of an agent A who acts in a sequential decision problem. This inference allows us to solicit utilities simply by observation, or it may be used in an asymmetric two-agent setting, with one informed agent optimizing her actions relative to her uncertainty about the utility of the oblivious agent A.

Our approach consists of two parts: We use the observations to formulate a set of linear constraints on the utility space (as in (Ng & Russell, 2000)). We then



(a) (b) (c)

Figure 4. Results for the bookseller example. (a) Estimation error for fully observed strategies. (b) Estimation error for partially observed strategies. (c) Realized utility for B as a function of the number of observations.

use these constraints to transform a given prior distribution over utilities into a more informative posterior. We can then obtain "candidate" utility functions for A by generating samples from the posterior distribution. These samples can be used to compute the mean of the posterior, to obtain a point estimate of the agent's utility function, or as an estimate of the posterior distribution as a whole, capturing our uncertainty about A's utility. As we have shown, these samples are particularly useful in providing a feasible algorithm for the two-agent "game" described above.

There are many interesting possible extensions to this work. Our ideas can be used in the setting of infinite horizon MDPs, simply by "unrolling" the MDP into trees with a longer and longer horizon. While this naive approach will lead to an exponential blowup, we can address the problem by using standard dynamic programming techniques. An open question, however, is the extent to which the quality of our bounds will decay with the length of the horizon. More globally, we can consider extending our results to a richer class of games, e.g., ones where the probabilities (beliefs) are not necessarily common knowledge, or ones where both players are strategic.

We can also consider extensions of the ability of the informed player to learn from the interaction. We have shown that we can view the interaction between the informed and oblivious player as a repeated game, with each interaction giving the informed agent more information about the utility function of the oblivious agent. This view immediately leads to an interesting exploration/exploitation tradeoff. We have also assumed so far that the prior over utility functions was learned in advance, and then applied as is to each new user. It would be interesting to see if the very partial data obtained by viewing only the behavior of the different agents can also be used as training data to improve our prior distribution over utility functions in the population.

Acknowledgments

This work was supported by Air Force contract F30602-00-2-0598 under DARPA's TASK program and by ONR MURI N00014-00-1-0637.

References

Applegate, D., & Kannan, R. (1991). Sampling and integration of near log-concave functions. *Proc. STOC* (pp. 156–163).

Bárány, I., & Füredy, Z. (1986). Computing the volume is difficult. *Proc. 18th ACM STOC* (pp. 442–447).

Chajewska, U., & Koller, D. (2000). Utilities as random variables: Density estimation and structure discovery. *Proc. UAI* (pp. 63–71).

Chajewska, U., Koller, D., & Parr, R. (2000). Making rational decisions using adaptive utility elicitation. *Proc. AAAI* (pp. 363–369).

Fromberg, D. G., & Kane, R. L. (1989). Methodology for measuring health-state preferences—II: Scaling methods. *Journal of Clinical Epidemiology*, *42*, 459–471.

Fudenberg, D., & Levine, D. K. (1998). *The theory of learning in games.* The MIT Press.

Keeney, R. L., & Raiffa, H. (1976). *Decisions with multiple objectives: Preferences and value tradeoffs.* John Wiley & Sons, Inc.

Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. *Proc. ICML.*

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems.* San Francisco, CA: Morgan Kaufmann.

Selten, R. (1991). Anticipatory learning in two-person games. In *Game equilibrium models*, vol. 1, 98–154. Springer-Verlag.

von Neumann, J., & Morgenstern, O. (1947). *Theory of games and economic behavior.* Princeton, N.J.: Princeton University Press. 2nd edition.