Utility Elicitation as a Classification Problem

Urszula Chajewska
Computer Science Department
Stanford University
Stanford, CA 94305-9010
and
Rockwell Palo Alto Laboratory
444 High Street, Suite 400
Palo Alto, CA 94301
urszula@cs.stanford.edu

Lise Getoor
Computer Science Department
Stanford University
Stanford, CA 94305-9010
getoor@cs.stanford.edu

Abstract

The majority of real-world probabilistic systems are used by more than one user, thus a utility model must be elicited separately for each new user. Utility elicitation is long and tedious, particularly if the outcome space is large and not decomposable. Most research on utility elicitation focuses on making assumptions about the decomposability of the utility function. Here we make no assumptions about the decomposability of the utility function; rather we attempt to cluster a database of existing user utility functions into a small number of prototypical utility functions. Having identified these prototypes, we can then effectively classify a new user's utility function by asking many fewer and simpler assessments than full utility model elicitation would require.

1 Introduction

Probabilistic systems, such as Bayesian Networks [Pearl, 1988] and Influence Diagrams [Howard and Matheson, 1984] have been studied for more than a decade. Many systems based on them are now in use. Some of them are designed to give advice to a large number of users. Often these users may not agree in their preferences and goals in a given decision context. Therefore, we need to elicit a utility function not once, as a part of creating a model, but many times—once for each user. This may be an extremely long and tedious process, particularly if the outcome space is large and the utility function is not known to be easily decomposable into independent components.

Utility elicitation has been studied extensively in the area of Decision Analysis (DA) [Luce and Raiffa, 1957; Keeney and Raiffa, 1976; Howard, 1984a; 1984b]. Only recently has it started to receive attention in Artificial Intelligence (AI) [Ha and Haddawy, 1997; Linden *et al.*, 1997]. Most of the research concentrates on decomposing utility functions, taking advantage of the various assumptions of independence between the attributes. Decomposed functions are easier to elicit and can allow more efficient reasoning procedures. Common approaches restrict attention to additive models [Linden *et al.*, 1997], and partial elicitation of models [Ha and Haddawy, 1997].

In many cases, however, attributes are highly correlated and thus assumptions of decomposability are suspect. If we incorrectly assume decomposability, the true utility function may differ significantly, and thus our assumption will adversely affect the choice of strategy.

However, outcome spaces may be very large and eliciting full utility functions from every user separately may be infeasible. Yet, there is still hope for effective elicitation of user's utility functions. Quite often, there are only a few qualitatively different utility functions. In many cases, we find that we can partition most users' utility functions into classes with very similar functions within each group. Having these clusters of utility functions defined and knowing their prevalence in the population can guide the process of utility elicitation from a particular user.

In our approach, we begin with a database of user utility functions. From this database, we cluster the utility functions, identifying a set of prototypical utility functions. We define a similarity measure for utility functions, and use a modification of one the popular clustering algorithms to construct the prototypes. Next, we build a decision tree for classifying a new utility function. The types of questions posed to the user during this elicitation will be easier for them to answer than

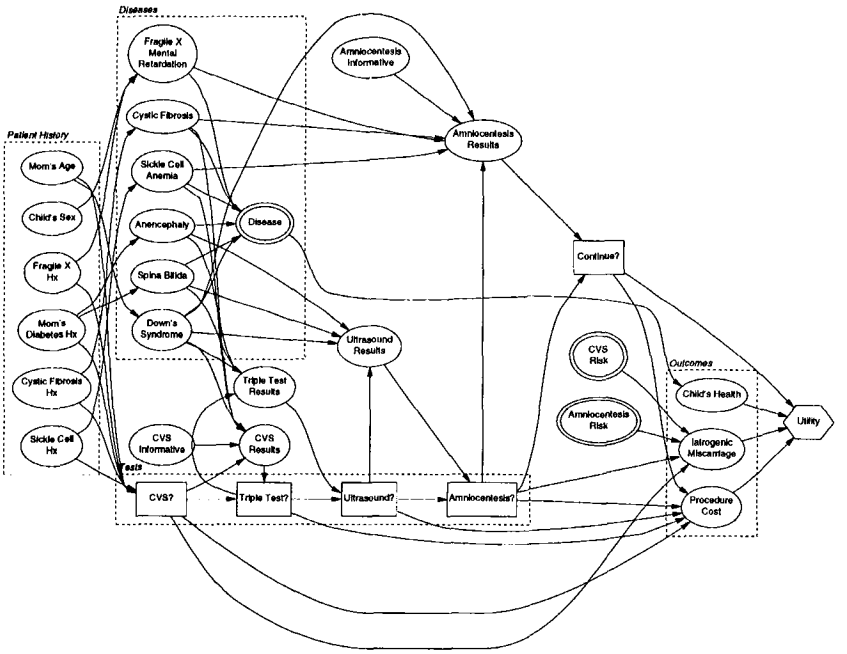

Figure 1: A simplified version of PANDA model (courtesy of Joseph Norman and Yuval Shahar, Stanford's Section on Medical Informatics).

the questions required during full elicitation. In addition, the number of questions required to classify a new utility function should be significantly fewer than the full utility function elicitation would require.

The domain we focus on is prenatal testing as shown in Figure 1. We are using a simplified version of the model developed by the PANDA project lead by Prof. Yuval Shahar in Stanford's Section on Medical Informatics[1]. PANDA is a loose acronym for "prenatal testing decision analysis." PANDA uses knowledge gained from many studies and from practicing obstetricians to advise the patients on which prenatal diagnostic tests they should choose during their pregnancies. The model includes data about six major diseases which can be diagnosed before birth, along with their prevalence and severity. It considers four tests used to diagnose these diseases. These tests have different sensitivities, specificities, costs and health risks.

The decision about the choice of tests is rarely easy. The patient's risk for having a child with a serious disease depends on the mother's age, child's sex and race and the family history. Some tests are not very accurate; others carry a significant risk of inducing miscarriages. The analysis of the model reveals the considerable influence of the utility function (especially the patient's attitude towards the risk of having a child with a serious disease and toward a miscarriage) on the final decisions.

In the following section, we review some concepts from utility theory in the context of this example. In Section 3, we describe our approach to identifying prototype utility functions. In Section 4, we show how these prototypes are used to classify a new user's utility function. Finally, in Section 5, we discuss potential benefits of our approach and future directions for this work.

2 Review: Utility Theory

The principle of maximizing expected utility has long been established as the guide to making rational decisions. The axioms of utility theory, which are stated in terms of constraints on preferences, imply the existence of utility functions. [Neumann and Morgenstern, 1947; Savage, 1954; Luce and Raiffa, 1957].

Let O be a set of possible outcomes $\{o_1, \ldots, o_n\}$. The outcomes are sometimes also called situations (in AI literature) or consequences (in DA literature). In the case of the PANDA model, the possible outcomes include: the birth of a healthy baby, a test-induced miscarriage, the birth of a baby with severe Down's syndrome, etc. The set of possible strategies, $S = \{s_1, \ldots, s_m\}$ contains all possible decision sequences (conditional plans), such as: "do not take CVS, take the triple test, if the result is negative, do not take any more tests, otherwise, take amniocentesis, if the result is negative, continue, if not, terminate the pregnancy." The patient's history, h_k, is

[1]See http://smi-web.stanford.edu/projects/panda/.

an instantiation of the observable variables in the model representing information specific to the patient and her pregnancy: patient's age, child's sex and race, family history of diabetes, etc. The given decision strategy together with the patient's history induces a probability distribution over the set of outcomes $P(\mathbf{O}|\mathbf{H},\mathbf{S})$. Given a probability distribution P and the user's utility function defined over the outcomes, $U(\mathbf{O})$, we can compute the expected utility for the given patient and the chosen strategy:

$$\mathrm{EU}(s|h) = \sum_o P(o|h,s)U(o).$$

Von Neumann and Morgenstern developed an approach to utility elicitation based on measuring the strength of a person's preference for an outcome by the risks he or she is willing to take to obtain it. Consider the outcome space consisting of three events o_1, o_2, and o_3 and a user with the preference ordering $o_1 \succ o_2 \succ o_3$. If he or she is offered a choice between o_2 for sure and a gamble in which o_1 will be received with probability π and o_3 with probability $(1-\pi)$, denoted $< \pi, o_1; (1-\pi), o_3 >$, then, according to the theory, there exists a value π for which the user will be indifferent. The outcome o_2 can then be assigned the value $\pi U(o_1) + (1-\pi)U(o_3)$.

The utility function was shown by von Neumann and Morgenstern [Neumann and Morgenstern, 1947] to be uniquely determined up to an increasing linear transformation, i.e., for any utility function $U(O)$ and constants a and b, such that $a > 0$, $aU(O) + b$ is also a utility function encoding the same preferences. The constant a changes the scale of the utility function. The constant b changes its zero point.

In order to compare two utility functions, we have to make sure that they are normalized, i.e., their zero points and scales are the same. Usually this is done by finding two endpoints of the scale—the best and worst possible outcomes, $o_\top$ and $o_\bot$—and assigning them the values of 1 and 0 respectively. However, the worst outcome in any given set does not necessarily have the same value for every person. It is a common practice to include the death of the decision maker in the set of outcomes for that reason[2].

In this paper, we will assume that the utility functions in our database have been normalized. In the PANDA domain, the two outcomes chosen for the endpoints could be the birth of a healthy baby, $o_\top$, and the death of the pregnant woman herself, $o_\bot$; the utility functions will take the values in the interval $[0, 1]$.

3 Identifying Prototype Utility Models

We will assume that we have a database of N normalized utility functions[3] over our outcome space O, where $|O| = D$. Our data points—the utility functions—will be represented as vectors of values, one value for each outcome, $\{u(o_1), \ldots, u(o_D)\}$. We will also assume that the population sample used in creating the database is drawn from some distribution $\mathcal{U}$; our new users will also come from the same distribution.

In order to create the utility models or prototypes, we can use any of the popular clustering algorithms. Clustering involves dividing a set of data points into non-overlapping groups, or clusters, of points, where points in a cluster are "more similar" to one another than to points in other clusters. (We will return to the term "more similar" shortly.) When a dataset is clustered, every point is assigned to some cluster, and every cluster can be characterized by a single reference point (we will call these points *prototypes*), which may be a particular instance in the cluster (as in the *k-mediods* algorithm [Vinod, 1969]), or some constructed exemplar, for example the mean of the points in the cluster (as in the popular *k-means* algorithm [MacQueen, 1967]).

The algorithm starts by picking a set $\mathcal{P}$ of k prototypes at random. Next we assign each data point to the closest prototype. Once we have our new clusters, we choose a new prototype for each cluster according to some scoring criteria. We repeat until convergence, i.e. until there is no change in cluster membership, or for some prespecified computation limit.

As we mentioned while describing the clustering algorithm, we are trying to cluster together "similar" utility functions. The choice of the distance measure used in clustering is very important. As we will see later, we will use the prototype's utility function for our computation of the best strategy. In order to do so, we must have some guarantees as to the variance in expected utility for each cluster.

We begin by defining the expected utility of a particular strategy s_i with respect to a particular utility function u_j and a particular history h_k:

$$\mathrm{EU}_{u_j}(s_i|h_k) = \sum_{o_l} P(o_l|s_i,h_k)u_j(o_l)$$

[2]The problem is non-trivial. In a recent paper Druzdzel [Druzdzel, 1997] pointed out the importance of a fixed zero point in explaining the recommendations of probabilistic systems. He considers this matter in some detail and suggests setting the zero point of the utility function at the status quo (with respect to the elements affected by the decision process). Note that this would allow the assignment of a negative utility value to some outcomes. In case of normalized utility functions with death as the worst outcome, no negative values are allowed.

[3]In fact, the assumption is not unreasonable. Medical informatics centers are collecting user's preferences for medical decision analysis. See, for example, http://preferences.stanford.edu.

where o_l ranges over possible outcomes. We can then look at the *best strategy* for a particular utility function and history,

$$s^*_{u_j|h_k} = \text{argmax}_{s_i} \text{EU}_{u_j}(s_i|h_k).$$

In the following, let u_p be the utility function for some prototype p and let $s^*_{u_p|h_k}$ be the best strategy for this prototype.

Eventually, we will be giving advice to new users based on the expected utility computation for the prototype to whose cluster this user's utility function is determined to belong. We wouldn't like the result to differ significantly from what users could expect based on the full utility elicitation. We need to consider two possibly different strategies: the strategy that we will pick for u_p, $s^*_{u_p|h_k}$, versus the strategy that we would pick for the true utility function, $\tilde{u}$, $s^*_{\tilde{u}|h_k}$. We will evaluate both of these strategies relative to $\tilde{u}$ and for a particular history h_k. The difference between the expected utilities of these two strategies is

$$\text{EU}_{\tilde{u}}(s^*_{\tilde{u}|h_k}) - \text{EU}_{\tilde{u}}(s^*_{u_p|h_k}).$$

We begin by computing for every utility function u_j in our database the optimal strategy for the current history h_k: $s^*_{u_j|h_k}$. We also compute the expected utility of each of these strategies: $\text{EU}_{u_j}(s^*_{u_j|h_k})$.

Now, for each utility function we will want to compare the expected utility of its best strategy to the expected utility of the strategy for each of the k prototypes $\mathcal{P}$. Thus, in addition, for each u_j, we compute the expected utility of the best strategy for each prototype, relative to u_j: $EU_{u_j}(s^*_{u_p|h_k})$.

Now, we cluster so as to minimize

$$\min_{p \in \mathcal{P}} \left(\text{EU}_{u_j}(s^*_{u_j|h_k}) - \text{EU}_{u_j}(s^*_{u_p|h_k}) \right).$$

We assign each utility function to the cluster of the nearest prototype. Next we choose a new utility function as the prototype for this cluster by picking the utility function for which the sum of these differences is the smallest, i.e., for each u_j in this cluster compute

$$\text{Score}(u_j) = \sum_{\substack{u_i \in \text{cluster} \\ i \neq j}} \left(\text{EU}_{u_i}(s^*_{u_i|h_k}) - \text{EU}_{u_i}(s^*_{u_j|h_k}) \right)$$

and choose the u_j with the lowest score as the new prototype for this cluster. We repeat this process until we either converge, or reach some computation limit. The algorithm is presented in Figure 2.

Note that we are creating clusters for a particular history. We can do this in two ways. We can do this online, at the point at which we are presented with a new user whose utility function we want to classify. At that point, we know the particular history h_k that we are interested in, and we create the clusters relative to that history. Alternatively, we may do this offline, and create prototypes for each potential history. The choice will obviously depend on the size of the problem, the response requirements for the online utility elicitation and the storage availability.

Procedure Clustering

Inputs: N = number of data points
D = number of outcomes
k = number of classes
u_j utility functions to be clustered; $j = 1..N$
h_k the current history
Outputs: p_i prototypes; $i = 1..k$

Choose an initial set of k prototypes p_i at random
Repeat
  For each u_j
    For each p_i
      Compute $d(u_j, u_{p_i}) = \text{EU}_{u_j}(s^*_{u_j|h_k}) - \text{EU}_{u_j}(s^*_{u_{p_i}|h_k})$
  Assign each data point u_j to cluster of nearest prototype
  For each p_i
    For each $u_j \in$ cluster i
      $\text{Score}(u_j) = \sum_{\substack{u_i \in \text{cluster} \\ i \neq j}} \text{EU}_{u_i}(s^*_{u_i|h_k}) - \text{EU}_{u_i}(s^*_{u_j|h_k})$
    Replace p_i with the u_j with lowest score
until there are no changes or time limit exceeded

Figure 2: Clustering Procedure.

In addition, another variation of our algorithm would instead of clustering for each history, create clusters based on averaging over histories, by minimizing the following distance:

$$d(u_j, u_p) = \sum_{h_k} P(h_k) \cdot (\text{EU}_{u_j}(s^*_{u_j|h_k}) - \text{EU}_{u_j}(s^*_{u_p|h_k}))$$

Here the algorithm choice will again depend on the characteristics of the problem. In the case where using a strategy that often is a good one, in an unlikely situation, does not have serious consequences, this latter approach may be satisfactory. However, in a medical setting, where it is imperative to follow the correct strategy in the case of unlikely situation, our first approach is appropriate.

4 Using Default Models for Utility Elicitation

Given that we have found our k utility prototypes, we now build a decision tree, which we will use to classify new users' utility function. There are two types of splits we allow in the tree: preference splits and fixed lottery splits. Preference splits are of the form "Is outcome o_i

preferred to outcome o_j?" Fixed lottery splits are of the form "Is outcome o_i preferred to the standard lottery, $[c, o_\top; (1-c), o_\bot]$?". Note that each of these questions are in some sense "easier" to answer than the question asked during full model elicitation: "For what value of c is the user indifferent between outcome o_i and the standard lottery, $[c, o_\top; (1-c), o_\bot]$?".

How do we compute the best split? To keep the number of expected questions to a minimum, at every stage we want to find the question that results in the greatest information gain. At first, the information content of the answer to our classification question is

$$I(P(p_1), \ldots, P(p_k)) = \sum_{i=1}^{k} -P(p_i) \log_2 P(p_i),$$

where $P(p_i)$ is the probability that any given instance belongs to cluster i. To find the split with the greatest information gain, we need to compute (for all possible questions) the information we will still need after the split, and subtract it from the information content we need at the current node. This will be given by

$$I(P(p_1), \ldots, P(p_k)) - \left(\sum_{i=1}^{k} P(p_i|yes) I(P(p_1|yes), \ldots, P(p_k|yes)) + \sum_{i=1}^{k} P(p_i|no) I(P(p_1|no), \ldots, P(p_k|no)) \right),$$

where $P(p_i|yes)$ is the fraction of all utility functions belonging to cluster i which are consistent with the question we are considering.

In case of preference questions, all we need to do is to compute for each utility prototype the transitive closure of the ordering relation it defines. Then, we can easily compute the information gain resulting from each possible preference question.

In case of a lottery split, we need to choose the constant c. Considering many possible constants for every outcome would be quite expensive. To simplify matters, here we consider the following heuristic for choosing the constant c: for every outcome, set c to be the average utility of o_i among all the utility functions in the database.

It is tempting to introduce a slight bias towards preference questions, since they are much easier to answer for the user. We could use the lottery splits only if the information gain provided by them is "much higher" than the one provided by the best preference split. The precise definition of what "much higher" should mean needs to be determined by empirical evaluation.

At every node in the tree, we would be able to specify the probability that a utility function at this node in the tree belongs to the cluster i, for all $i = 1 \ldots k$. We should stop splitting when this probability exceeds some threshold θ for one of the clusters.

Note that in general, for full utility elicitation, there are $|O|$ lottery questions that must be asked of the form "For what value of c is the user indifferent between outcome o_i and the standard lottery, $[c, o_\top; (1-c), o_\bot]$?". In our decision tree classification, there will be at most k questions (and in many cases, much fewer than that; hopefully closer to $\log k$) asked, and each of the questions will be a preference question of the form, "Is outcome o_i preferred to outcome o_j?" or, "Is outcome o_i preferred to the standard lottery, $[c, o_\top; (1-c), o_\bot]$?", for fixed c.

How is this procedure helping us to advise new users? First, we collect the history data h_k from the user. We find the appropriate clustering for that history, then we classify the new user's utility function. When a user's utility function is determined to belong to a given cluster characterized by the prototype p_i, we find the best strategy for the patient's history h_k and the prototype's utility function u_{p_i}.

5 Future Work

At this point, we have described a proposal for how one might approach utility elicitation. There is clearly a lot of work to be done to both implement and validate the approach.

On the practical side, we would like to test our hypothesis that the number of utility prototypes is typically small. We would like to experiment with different databases of utility functions to see for what domains this approach is feasible. It would also be useful to compare traditional utility elicitation procedure with the one presented here to evaluate the efficiency gains.

On the theoretical side, we would like to continue our research in several directions:

- We would like to more precisely characterize the savings that our utility function elicitation can provide, under particular assumptions about the models.
- We would like to use the diversity of strategies in a particular cluster to guide splitting and merging of prototypes.
- Obviously the splits in the decision tree can also be done on the features of the outcome space, rather than preference rankings on the full outcomes. Building trees in this manner may aid in the identification of general utility function decompositions.
- In many cases, there may be some obvious constraints that all utility functions will obey. For example, in the PANDA model, we can assume that a woman seeking advice on the choice of diagnostic tests would consider having a healthy baby more

desirable than having a baby with a severe disability. We would like to explore the possibility of using such constraints, or background knowledge, to increase the effectiveness of our procedure. These constraints may limit the number of utility prototypes required and increase the accuracy and efficiency of the utility elicitation procedure.

In addition, as mentioned earlier, it would be interesting to compare different clustering methods. In particular, we would like to compare the results of using a Bayesian clustering approach.

6 Acknowledgments

We thank Yuval Shahar and Joseph Norman, the creators of the PANDA model, for many inspiring and useful discussions about the problems created by their domain. Urszula Chajewska's work was supported in part by the ARPI grant F30602-95-C-0251 and by the NSF, under grant IRI-95-03109. Lise Getoor's work was supported by a National Physical Sciences Consortium fellowship.

References

[Druzdzel, 1997] M. J. Druzdzel. An incompatibility between preferential ordering and the decision-theoretic notion of utility. In *Working notes of the AAAI 1997 Spring Symposium on Qualitative Preferences in Deliberation and Practical Reasoning*, pages 35–40, Stanford, CA, 1997.

[Ha and Haddawy, 1997] V. Ha and P. Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 215–222, Providence, Rhode Island, 1997.

[Howard and Matheson, 1984] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *The Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, CA, 1984.

[Howard, 1984a] R. A. Howard. The foundations of decision analysis. In R. A. Howard and J. E. Matheson, editors, *The Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, CA, 1984.

[Howard, 1984b] R. A. Howard. Risk preference. In R. A. Howard and J. E. Matheson, editors, *The Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, CA, 1984.

[Keeney and Raiffa, 1976] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley & Sons, Inc., New York, 1976.

[Linden *et al.*, 1997] G. Linden, S. Hanks, and N. Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings, User Modelling '97*, 1997.

[Luce and Raiffa, 1957] R. D. Luce and H. Raiffa. *Games and Decisions*. John Wiley & Sons, New York, 1957.

[MacQueen, 1967] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, Berkeley, CA, 1967. University of California Press.

[Neumann and Morgenstern, 1947] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, N.J., 2nd edition, 1947.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, CA, 1988.

[Savage, 1954] L. J. Savage. *Foundations of Statistics*. John Wiley & Sons, New York, 1954.

[Shahar *et al.*, 1992] Y. Shahar, J. W. Egar, and R. Pichumani. Decision analysis of amniocentesis for prenatal diagnosis. In *Proceedings of the Annual Meeting of the Society for Medical Decision Analysis*, Portland, Oregon, 1992.

[Vinod, 1969] H. Vinod. Integer programming and the thoery of grouping. *Journal of the American Statistical Association*, 64:506–517, 1969.