Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 160 (2004) 53–78
Artificial Intelligence
www.elsevier.com/locate/artint

Learning a decision maker's utility function from (possibly) inconsistent behavior

Thomas D. Nielsen *, Finn V. Jensen

Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg Ø, Denmark

Received 9 September 2003; accepted 18 August 2004

Abstract

When modeling a decision problem using the influence diagram framework, the quantitative part rests on two principal components: probabilities for representing the decision maker's uncertainty about the domain and utilities for representing preferences. Over the last decade, several methods have been developed for learning the probabilities from a database. However, methods for learning the utilities have only received limited attention in the computer science community.

A promising approach for learning a decision maker's utility function is to take outset in the decision maker's observed behavioral patterns, and then find a utility function which (together with a domain model) can explain this behavior. That is, it is assumed that decision maker's preferences are reflected in the behavior. Standard learning algorithms also assume that the decision maker is behavioral consistent, i.e., given a model of the decision problem, there exists a utility function which can account for all the observed behavior. Unfortunately, this assumption is rarely valid in real-world decision problems, and in these situations existing learning methods may only identify a trivial utility function. In this paper we relax this consistency assumption, and propose two algorithms for learning a decision maker's utility function from possibly inconsistent behavior; inconsistent behavior is interpreted as random deviations from an underlying (true) utility function. The main difference between the two algorithms is that the first facilitates a form of batch learning whereas the second focuses on adaptation and is particularly well-suited for scenarios where the DM's preferences change over time. Empirical results demonstrate the tractability of the algorithms, and they also show that the algorithms converge toward the true utility function for even very small sets of observations.

* Corresponding author.
E-mail addresses: tdn@cs.auc.dk (T.D. Nielsen), fvj@cs.auc.dk (F.V. Jensen).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2004.08.003

Keywords: Influence diagram; Learning utility functions; Inconsistent behavior

1. Introduction

When modeling a decision problem using the influence diagram framework [10] we need to specify both a qualitative part (represented by a acyclic directed graph) and a quantitative part. The quantitative part is comprised of probabilities which represent the decision maker's (DM's) uncertainty about the domain, and utilities which represent the DM's preferences about the different outcomes of the decision problem.

The specification of the quantitative part of the model constitutes one of the main difficulties when modeling a decision problem using the influence diagram representation. Over the last decade, research has focused on learning the probabilities from a database (see, e.g., [4] for an overview), however, automatic learning of the utilities has received less attention. For sequential decision problems, this task was first addressed by Sargent [23] within the economics community. Sargent [23] considered the problem of estimating (among other parameters) the cost of adjusting the daytime and overtime labor force by observing a firm's labor demand. In a more general context, there has been an increasing interest in techniques for structural estimation of Markov decision processes (MDPs), see [22] and the references within. For instance, Keane and Wolpin [15] estimates a dynamic model for schooling, work and occupational choice of young men.

In general, current approaches for semi-automatic learning of a DM's utility function have focused on three different areas: (i) eliciting the utility function of the DM based on a database of already elicited utility functions [5,6], (ii) iterative refinement of the DM's current utility function using a value of information approach [8], and (iii) eliciting the utility function based on a database of observed behavioral patterns [7,18,22,27]. The latter approach is based on the assumption that the DM's "true" utility function is reflected in the observed behavior.

In this paper we focus on the third category within the framework of influence diagrams, i.e., learning a DM's utility function from her observed behavioral patterns. Unfortunately, the estimation techniques for MDPs cannot directly be applied to frameworks such as influence diagrams, although some of the ideas carry over. Moreover, existing ID learning methods are based on the assumptions that the model provides an accurate representation of the problem domain, and that the DM is an expected utility maximizer always with the same von Neuman–Morgenstern utility function.[1] These assumptions, however, are rarely valid in practice. For example, humans rarely behave as to maximize the expected utility w.r.t. a certain model [1], and the utility function of a DM may fluctuate and/or change permanently over time. As a consequence, traditional learning methods may only be able to identify a trivial utility function (i.e., a utility function giving the same utility to all outcomes), but this type of utility function is hardly an appropriate description of the DM's preferences.

[1] Together, these two assumptions imply that the DM is behavioral consistent, i.e., given a model of the decision problem, there exists a utility function which can account for all the observed behavior.

In this paper we relax the assumptions above, and propose two algorithms for learning the "true" utility function of a DM by observing possibly inconsistent behavior, i.e., behavior which cannot be explained by a non-trivial utility function.[2] The first algorithm can be characterized as a batch learning procedure whereas the second algorithm focuses on adaptation. The assumption underlying both of the proposed algorithms is that inconsistent behavior can be explained as random deviations from an underlying true utility function which is the one we would like to estimate.

A promising feature of the proposed algorithms follows from the fact that a given observation-decision sequence truncates the space of possible utility functions. Hence, we would in general expect that even a small database can be used to learn a reasonable approximation of the DM's true utility function. That this is also the case is confirmed by several empirical experiments, where it is also shown that the algorithms are even robust to deviations from the underlying utility model. This observation also points to another possible application area, namely adaptive agents in, e.g., computer games. That is, agents which can adapt to the behavior of other computer controlled agents as well as avatars. In these types of domains we usually have very limited amounts of data for the individual agents, thereby making it difficult to apply traditional machine learning algorithms when performing learning or adaptation.

The remainder of this paper is organized as follows. In Section 2 we describe the framework of influence diagrams and introduce some of the terms and notation used throughout the paper. In Section 3 we pose the general problem of learning a DM's utility function and we outline some existing learning algorithms. Section 4 discusses a utility model for explaining inconsistent behavior, and based on this model we propose two algorithms for learning a utility function from possibly inconsistent behavior; the first algorithm is a batch learning procedure whereas the second algorithm provides a method for doing adaptation. In Section 5 we present several empirical results which demonstrate the applicability of both algorithms. Finally, in Section 6 we discuss some optimization issues and point to areas for future research.

2. Influence diagrams

An influence diagram (ID) can be seen as a Bayesian network (BN) augmented with *decision nodes* and *value nodes*, where value nodes have no descendants. Thus, an ID is a acyclic directed graph $G = (\mathcal{U}, \mathcal{E})$, where the nodes $\mathcal{U}$ can be partitioned into three disjoint subsets; *chance nodes* $\mathcal{U}_C$, decision nodes $\mathcal{U}_D$ and value nodes $\mathcal{U}_V$.

In the remainder of this paper we assume a total ordering of the decision nodes, indicating the order in which the decisions are made (the ordering of the decision nodes is traditionally represented by a directed path which includes all decision nodes). Furthermore, we will use the concept of node and variable interchangeably if this does not introduce any inconsistency. We will also assume that no *barren nodes* are specified by the

[2] See [22] for a discussion of this problem in the context of MDPs, where inconsistent behavior is attributed to unobserved state variables.

ID since they have no impact on the decisions, see [24]; a chance node or a decision node is said to be barren if it has no children, or if all its descendants are barren.

With each chance variable and decision variable X we associate a *state space* sp(X) which denotes the set of possible outcomes/decision options for X. For a set $\mathcal{U}'$ of chance variables and decision variables we define the state space as $\mathrm{sp}(\mathcal{U}') = \times\{\mathrm{sp(X)} \mid \mathrm{X} \in \mathcal{U}'\}$, where $A \times B$ denotes the Cartesian product of A and B. The uncertainty associated with each chance variable C is represented by a *conditional probability potential* $P(C \mid \mathrm{pa}(C)) : \mathrm{sp}(\{\mathrm{C}\} \cup \mathrm{pa(C)}) \to [0; 1]$, where $\mathrm{pa}(C)$ denotes the parents of C in the ID. The domain of a conditional probability potential $\phi_C = P(C \mid \mathrm{pa}(C))$ is denoted $\mathrm{dom}(\phi_C) = \{C\} \cup \mathrm{pa}(C)$.

The DM's preferences are described by a multi-attribute utility potential, and in the remainder of this paper we shall assume that this utility potential is linearly-additive with equal weights, see, e.g., [28]. The set of value nodes $\mathcal{U}_V$ defines the set of *utility potentials* which appear as additive components in the multi-attribute utility potential.[3] Each utility potential indicates the local utility for a given configuration of the variables in its domain. The domain of a utility potential ψ_V is denoted $\mathrm{dom}(\psi_V) = \mathrm{pa}(V)$, where V is the value node associated with ψ_V. Analogously to the concepts of variable and node we shall sometimes use the terms value node and utility potential interchangeably.

A *realization* of an ID, I, is an attachment of potentials to the appropriate variables in I, i.e., a realization is a set $\{P(C \mid \mathrm{pa}(C)) \mid C \in \mathcal{U}_C\} \cup \{\psi_V(\mathrm{pa}(V)) \mid V \in \mathcal{U}_V\}$. So, a realization specifies the quantitative part of the model whereas the ID constitutes the qualitative part.

The arcs in an ID can be partitioned into three disjoint subsets, corresponding to the type of node they go into. Arcs into value nodes represent functional dependencies by indicating the domain of the associated utility potential. Arcs into chance nodes, termed *dependency arcs*, represent probabilistic dependencies, whereas arcs into decision nodes, termed *informational arcs*, imply information precedence; if there is an arc from a node X to a decision node D, then the state of X is known when decision D is made.

Let $\mathcal{U}_C$ be the set of chance variables and let $\mathcal{U}_D = \{D_1, D_2, \ldots, D_n\}$ be the set of decision variables. Assuming that the decision variables are ordered by index, the set of informational arcs induces a partitioning of $\mathcal{U}_C$ into a collection of disjoint subsets $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n$. The set $\mathcal{C}_j$ denotes the chance variables observed between decision D_j and D_{j+1}. Thus the variables in $\mathcal{C}_j$ occur as immediate predecessors of D_{j+1}. This induces a *partial order* $\prec$ on $\mathcal{U}_C \cup \mathcal{U}_D$, i.e., $\mathcal{C}_0 \prec D_1 \prec \mathcal{C}_1 \prec \cdots \prec D_n \prec \mathcal{C}_n$.

The set of variables known to the decision maker when deciding on D_j is called the *informational predecessors* of D_j and is denoted $\mathrm{pred}(D_j)$. By assuming that the decision maker remembers all previous observations and decisions, we have $\mathrm{pred}(D_i) \subseteq \mathrm{pred}(D_j)$ (for $D_i \prec D_j$) and in particular, $\mathrm{pred}(D_j)$ is the variables that occur before D_j under $\prec$. This property is known as *no-forgetting* and from this we can assume that an ID does not contain any no-forgetting arcs, i.e., $\mathrm{pa}(D_i) \cap \mathrm{pa}(D_j) = \emptyset$ if $D_i \neq D_j$.

[3] Note that Tatman and Shachter [28] also consider the case where the utility is defined as the product of a set of utility potentials; such a utility potential can be transformed to decompose additively by taking the logarithm of the utilities.

2.1. Evaluation

When evaluating an ID we identify a strategy for the decisions involved; a strategy can be seen as a prescription of responses to earlier observations and decisions. The evaluation is usually performed according to the *maximum expected utility principle*, which states that we should always choose an alternative that maximizes the expected utility.

Definition 1. Let I be an ID and let $\mathcal{U}_D$ denote the decision variables in I. A *strategy* is a set of functions $\Delta = \{\delta_D \mid D \in \mathcal{U}_D\}$, where δ_D is a *policy* given by:

$$\delta_D : \mathrm{sp}\big(\mathrm{pred(D)}\big) \to \mathrm{sp(D)}.$$

A strategy that maximizes the expected utility is termed an *optimal strategy*, and each policy in an optimal strategy is termed an *optimal policy*.

The optimal policy for the last decision variable D_n is given by:[4]

$$\delta_{D_n}(\mathcal{C}_0, D_1, \ldots, D_{n-1}, \mathcal{C}_{n-1}) = \arg\max_{D_n} \sum_{\mathcal{C}_n} P(\mathcal{C}_n \mid \mathcal{C}_0, D_1, \ldots, \mathcal{C}_{n-1}, D_n) \sum_{V \in \mathcal{U}_V} \psi_V .$$

Similarly, the maximum expected utility for decision D_n is

$$\rho_{D_n}(\mathcal{C}_0, D_1, \ldots, D_{n-1}, \mathcal{C}_{n-1}) = \max_{D_n} \sum_{\mathcal{C}_n} P(\mathcal{C}_n \mid \mathcal{C}_0, D_1, \ldots, \mathcal{C}_{n-1}, D_n) \sum_{V \in \mathcal{U}_V} \psi_V . \tag{1}$$

In general, the optimal policy for a decision variable D_k is given by:

$$\delta_{D_k}(\mathcal{C}_0, D_1, \ldots, D_{k-1}, \mathcal{C}_{k-1}) = \arg\max_{D_k} \sum_{\mathcal{C}_k} P(\mathcal{C}_k \mid \mathcal{C}_0, D_1, \ldots, \mathcal{C}_{k-1}, D_k) \rho_{D_{k+1}}, \tag{2}$$

where $\rho_{D_{k+1}}$ is the maximum expected utility potential for decision D_{k+1}:

$$\rho_{D_{k+1}}(\mathcal{C}_0, D_1, \ldots, D_k, \mathcal{C}_k) = \max_{D_{k+1}} \sum_{\mathcal{C}_{k+1}} P(\mathcal{C}_{k+1} \mid \mathcal{C}_0, D_1, \ldots, \mathcal{C}_k, D_{k+1}) \rho_{D_{k+2}}.$$

If we suppress the maximization over D_{k+1}, we obtain the *expected utility potential* for decision D_{k+1}:

$$\rho(D_{k+1}, \mathcal{C}_0, D_1, \ldots, D_k, \mathcal{C}_k) = \sum_{\mathcal{C}_{k+1}} P(\mathcal{C}_{k+1} \mid \mathcal{C}_0, D_1, \ldots, \mathcal{C}_k, D_{k+1}) \rho_{D_{k+2}}. \tag{3}$$

[4] For the sake of simplifying notation we shall assume that for all decision variables D_i there is always exactly one element in $\arg\max_{D_i}(\cdot)$.

The optimal policy for a decision variable, D_k, is in principle a function over the entire past of D_k (i.e., $\text{pred}(D_k)$). However, not all the variables observed do necessarily influence the optimal policy for D_k, hence we introduce the notion of a required variable:

Definition 2. Let I be an ID and let D be a decision variable in I. The variable $X \in \text{pred}(D)$ is said to be *required* for D if there exists a realization of I, a configuration $\bar{y}$ over $\text{dom}(\delta_D)\backslash\{X\}$, and two states x_1 and x_2 of X s.t. $\delta_D(x_1, \bar{y}) \neq \delta_D(x_2, \bar{y})$. The set of variables required for D is denoted $\text{req}(D)$.

Different algorithms for identifying the required variables have been proposed by Shachter [25], Nielsen and Jensen [19] and Lauritzen and Nilsson [16]. For example, the algorithm by Shachter [25] traverse the decision variables in reverse temporal order. When a decision variable is visited its required past is determined and the decision variable is then substituted by a chance variable having the required variables as parents:

Definition 3. Let D be a decision variable and let $\delta_D(\text{req}(D))$ be an optimal policy for D. A chance variable D' with $\text{req}(D)$ as parents and with the probability potential $P(D' \mid \text{req}(D))$ defined as:

$$P(d \mid \pi) = \begin{cases} 1 & \text{if } \delta_D(\pi) = d, \\ 0 & \text{otherwise} \end{cases}$$

is said to be the *chance variable representation of* δ_D.

Thus, we only need to (iteratively) determine the required variables for the last decision variable D_n. For this decision variable it can be shown that a variable $X \in \text{pred}(D_n)$ is required for D_n if and only if X is d-connected to a utility descendant of D_n given $\text{pred}(D_n) \setminus \{X\}$.

3. Observed behavior as utility constraints

Consider an influence diagram, I, modeling some decision problem, where the DM's preferences are expressed by a utility function ψ. Given this influence diagram representation, the DM's observed behavior can be seen as an observation-decision sequence (or *behavioral pattern*) w.r.t. the variables in I; for each decision D we have a configuration over the set of variables which is comprised of D and the variables which are observed immediately before D. Such an observation-decision sequence is necessarily consistent with at least one strategy Δ', and we therefore introduce the notion of an *instantiation* of a strategy.

Definition 4. Let I be an ID and let Δ be a strategy for the decision variables in I. Let I' be the BN obtained from I (value nodes are ignored) by replacing all decision variables in I with chance variable representations of their policies in Δ (see Definition 3). A configuration $\boldsymbol{c}$ over the variables $\bigcup_{D \in \mathcal{U}_D}(\text{req}(D) \cup \{D\})$ is then said to be an *instantiation* of Δ if $P'(\boldsymbol{c}) > 0$, where $P'(\cdot)$ is the probability distribution encoded by I'.

Definition 5. Let I be an ID and let ψ be a utility function for I. If Δ is an optimal strategy for I w.r.t. ψ and $\boldsymbol{c}$ is an instantiation of Δ, then $\boldsymbol{c}$ is said to be an instantiation of I w.r.t. ψ (or ψ *induces* $\boldsymbol{c}$). For a set $\mathcal{D} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ of instantiations we say that ψ induces $\mathcal{D}$ if ψ induces $\boldsymbol{c}_i$, for all $1 \leqslant i \leqslant N$.

Obviously, for any utility function we may have several different instantiations w.r.t. the utility function in question. Moreover, for any instantiation $\boldsymbol{c}$ we do not have a unique utility function inducing $\boldsymbol{c}$, since a utility function is only unique up to a positive affine transformation. Note that if the optimal strategy is not required to be strictly optimal, then any instantiation may be induced by a *trivial utility function* (a utility function giving the same utility to all outcomes).

Assume now that we have a database $\mathcal{D} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ of observed behavioral patterns for some DM; this corresponds to the situation where the DM is repeatedly confronted with the same decision problem, e.g., a doctor diagnosing and treating patients. Given that the DM is an expected utility maximizer, always with the same von Neuman–Morgenstern utility function, an obvious approach for identifying a representation of the utility function which induces $\mathcal{D}$ would be to investigate the set $\Psi_{\mathcal{D}}$ of candidate utility functions:

$$\Psi_{\mathcal{D}} = \{\psi \mid \psi \text{ induces } \boldsymbol{c}_i, \forall 1 \leqslant i \leqslant N\}. \tag{4}$$

If the database encodes an entire strategy, then we can easily find this set of candidate utility functions: Start off by substituting each decision variable with its chance variable representation as encoded by the cases in $\mathcal{D}$ (see Definition 3). Next, consider the last decision variable D_n and assume that case $\boldsymbol{c}$ specifies $\{D_n\} \cup \mathrm{req}(D_n) = (d, \pi_{D_n})$ that is, $\boldsymbol{c}^{\downarrow\{D_n\}\cup\mathrm{req}(D_n)} = (d, \pi_{D_n})$. This gives us:

$$\rho(d, \pi_{D_n}) \geqslant \rho(d', \pi_{D_n}), \quad \forall d' \in \mathrm{sp(D_n)} \setminus \{\mathrm{d}\}. \tag{5}$$

From Eq. (1), the expected utility function for the last decision is given by:

$$\begin{aligned}
\rho\big(D_n, \mathrm{req}(D_n)\big) &= \sum_{\mathcal{C}_n} P\big(C_n \mid D_n, \mathrm{pred}(D_n)\big) \sum_{V \in \mathcal{U}_V} \psi_V\big(\mathrm{pa}(V)\big) \\
&= \sum_{\mathcal{C}_n} P\big(C_n \mid D_n, \mathrm{req}(D_n)\big) \sum_{V \in \mathcal{U}_V} \psi_V\big(\mathrm{pa}(V)\big) \\
&= \sum_{V \in \mathcal{U}_V} \sum_{\mathcal{C}_n} P\big(C_n \mid D_n, \mathrm{req}(D_n)\big) \psi_V\big(\mathrm{pa}(V)\big) \\
&= \sum_{V \in \mathcal{U}_V} \sum_{\mathcal{C}_n \cap \mathrm{pa}(V)} \psi_V\big(\mathrm{pa}(V)\big) \sum_{\mathcal{C}_n \setminus \mathrm{pa}(V)} P\big(C_n \mid D_n, \mathrm{req}(D_n)\big) \\
&= \sum_{V \in \mathcal{U}_V} \sum_{\mathcal{C}_n \cap \mathrm{pa}(V)} \psi_V\big(\mathrm{pa}(V)\big) P\big(C_n \cap \mathrm{pa}(V) \mid D_n, \mathrm{req}(D_n)\big).
\end{aligned} \tag{6}$$

Thus, for any configuration (d, π_{D_n}) over $\{D_n\} \cup \mathrm{req}(D_n)$ the expected utility is linear in the utilities, and inequality (5) therefore defines a set of linear constraints for the utility function. Moreover all the coefficients for the inequalities can be found by instantiating the corresponding variables and performing one outward propagation in the strong junction tree representation of the ID [11,17]; from the construction of the strong junction tree

we are guaranteed that at least one clique contains $\mathrm{pa}(V)$ from which $P(C_n \cap \mathrm{pa}(V) \mid d, \pi_{D_n})$ can directly be read. Finally, as all decision variables have been substituted with their chance variable representations we can perform the analysis above for each decision variable by moving in reverse temporal order. This also implies that in order to find a set of candidate utility functions we simply need to solve the collection of linear inequalities identified during the analysis. Unfortunately this approach relies on the entire strategy to be encoded in the database which is very unlikely to happen in practice. When only a partial strategy is described by the database, then the constraints specified by the observed behavior are no longer linear and the above procedure cannot be applied. This can easily be seen by noticing that for any unobserved configuration of the required past for a decision variable, we need to apply a maximization when eliminating that variable; the expected utility function for a decision variable in the required past for that variable will therefore no longer be linear (we shall return to this discussion in Section 6).

Now assume that we are given a set of candidate utility functions, and we would like to reason about the DM preferences. One approach would be to select one of the candidate utility functions as a representation of the DM's unknown utility function; by the assumptions above we are guaranteed that such a utility function exists. This approach has been pursued by, e.g., Suryadi and Gmytrasiewicz [27] who propose an adaptation scheme where a single utility function is found by using a method similar to the delta rule for learning neural networks. Ng and Russell [18] propose heuristics for discriminating among the different candidate utility functions in the context of Markov decision processes. Alternatively, Chajewska et al. [7] derive a procedure for estimating a probability distribution over the candidate utility functions. The approach by Chajewska et al. [7] is based on specifying a prior probability distribution for the utilities, and then applying a Markov Chain Monte Carlo technique for sampling from this truncated distribution; the distribution is truncated according to the constraints imposed by the observed behavior. Note that both Ng and Russell [18] and Chajewska et al. [7] consider the situation when the optimal strategy is fully observed as well as when it is only partially observed.

Unfortunately, the assumptions underlying the methods referenced above are rarely valid in real world decision problems: (i) the ID model may not provide an accurate description of the decision problem, see, e.g., [22,26], (ii) the DM's preferences may fluctuate and/or change permanently over time, and (iii) human DMs do not always behave as to maximize the expected utility [1,14]. In these situations, the observed behavior of the DM may appear inconsistent w.r.t. the ID in question, i.e., there may not exist a non-trivial utility function for the ID which satisfies all the constraints implied by the observed behavior. As a consequence, existing learning methods would only identify a trivial utility function which is inadequate for, e.g., reasoning about the future behavior of the DM. Note that this problem has previously been considered in the context of MDPs [22], where discrepancies in the observation-decision sequences are attributed to the occurrence of unobserved state variables. In this setting, the parameters are estimated by maximizing (using a nested fixed point algorithm) the likelihood function for the observed behavior.

In what follows we establish a utility model which accommodates situation (i) and (ii); as a special case, the model also includes the situation where the previous stated assumptions do hold. Based on this model we propose two methods for learning a DM's utility function by observing (possibly) inconsistent behavioral patterns. The discussion of (iii) is

postponed until Section 6, where the model is briefly discussed in relation to the *rank dependent utility model* by Quiggin [21].

4. Learning a utility function from inconsistent behavior

In the proposed utility model, it is assumed that the DM has an underlying (true) utility function (denoted ψ^*) and that any type of inconsistent behavior can be explained as deviations from ψ^*.[5] More precisely, consider a database of cases $\mathcal{D} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ from an ID I. In the proposed model, each case $\boldsymbol{c}_i$ is assumed to be an instantiation of I w.r.t. some utility function $\psi^{\boldsymbol{c}_i}$ obtained from ψ^* by adding some white noise. Hence, each case, $\boldsymbol{c}_i$, is an instantiation of an optimal strategy found w.r.t. the utility function $\psi^{\boldsymbol{c}_i}$ generated from ψ^*; for each outcome o, we have $\psi^{\boldsymbol{c}_i}(o) = \psi^*(o) + \varepsilon_o$, where ε_o has a normal distribution with zero mean and variance σ_o^2. Hence, by assuming that the noise associated with the utilities are marginally independent, and that the variance only depends on the outcome in question we get

$$\psi^{\boldsymbol{c}_i} = \sum_{V \in \mathcal{U}_V} \psi_V^{\boldsymbol{c}_i} = \sum_{V \in \mathcal{U}_V} \left(\psi_V^* + \varepsilon_V^{\boldsymbol{c}_i}\right), \tag{7}$$

where $\varepsilon_V^{\boldsymbol{c}_i}(o^{\downarrow \mathrm{pa}(V)}) \sim N(0, \sigma^2_{o^{\downarrow \mathrm{pa}(V)}})$. Note that we implicit assume that $\psi^{\boldsymbol{c}_i}$ decomposes additively in the same factors as ψ^*. A graphical representation of the utility model (with m utility parameters in the true utility function) is illustrated in Fig. 1, where the nodes $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N$ are information nodes that encode the observed behavior; the independence statements can be deduced in the usual fashion.

Thus, for each parameter, ψ_j^*, in the true utility function we have a variance, σ_j^2, and a corresponding utility parameter, $\psi_j^{\boldsymbol{c}_i}$, for each case $\boldsymbol{c}_i$. Note that the model does not make any assumptions about the type of prior probability distribution of the true utility function and the variance of the parameters. Moreover, the model facilitates an easy way to encode prior information about the DM's preferences. For instance, given that we have the prior information that $o_i \prec o_j$, then we can take this information into account by simply

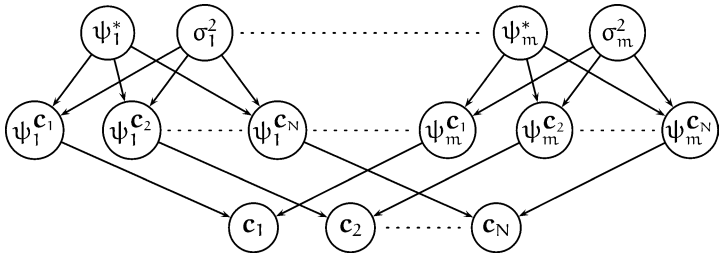

Fig. 1. A graphical representation of the utility model for encoding inconsistent behavior.

[5] This assumption can also be seen in relation to (i) as well as the discussion by [26] who argue that uncertainties in the model can be encoded in the utility function.

specifying $\psi^*(o_i) < \psi^*(o_j)$; thus changing the independence statements in the model, making it a chain graph.

Each constraint induced by an observed behavioral pattern specifies the set of candidate utility functions for the corresponding case. Given a prior probability distribution for the true utility function, we can then, in principle, use the set, $\mathcal{D}$, of cases to obtain a posterior probability distribution $P(\psi^* \mid \mathcal{D})$ for the true utility function. In turn, this probability distribution can be used to calculate the expected value of each parameter in the true utility function, which could be used as a representation of the DM's actual utility function.

Unfortunately, even if the prior distribution for the true utility function has a "nice" closed form expression, the posterior may not be on closed form nor may it belong to some specific family of probability distributions. To overcome this problem we can instead apply a Markov Chain Monte Carlo technique, where we generate samples from the desired distribution. For instance, assume that the database $\mathcal{D}$ consists of N cases $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ and that we are interested in computing the expected value of each utility parameter. That is, we seek:[6]

$$\mathbb{E}[\psi_j^* \mid \mathcal{D}] = \int_{\psi_j^*=0}^{1} \psi_j^* P(\psi_j^* \mid \mathcal{D}) \, \mathrm{d}\psi_j^*, \quad \forall 1 \leqslant j \leqslant m, \tag{8}$$

and this expectation can be approximated by:

$$\mathbb{E}[\psi_j^* \mid \mathcal{D}] \approx \frac{1}{n} \sum_{k=1}^{n} x_k, \tag{9}$$

where $\{x_1, \ldots, x_n\}$ are sampled from $P(\psi_j^* \mid \mathcal{D})$.

A naïve approach to generating these samples would be to construct a Markov Chain according to the model in Fig. 1. Thus the state space of the chain would correspond to the hyper-cube $[0; 1]^d$, where $d = m \cdot (N + 1)$. That is, the dimensionality of the hyper-cube is proportional to the number utilities; for each utility parameter ψ_j^* in the true utility function we have the set $\{\sigma_j^2, \psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N}\}$ of parameters. When sampling from the Markov Chain we can then discard any sample that does not satisfy the instantiations; this can be determined (without considering computational complexity) by inserting the proposed utility function in the ID and then testing whether the associated case is included in the optimal strategy. Thus, using this state space we can directly generate samples from the posterior distribution of the true utility function which can then be used to, e.g., approximate the expected value of each utility parameter in the true utility function (see Eq. (9)). Observe, that even though we are only interested in sampling from the true utility function we still need to include all the utility parameters associated with the different cases in the database. This also reveals the main computational problem with this approach: the hyper-cube grows exponentially in the number of cases, and we therefore need a large set of samples in order to get an adequate approximation of the posterior probability.

[6] The assumption that the utility parameters only take on values in the interval $[0; 1]$ is not a restriction, since any utility function can be transformed to this scale.

In the following sections we present two alternative methods for calculating the expected value of each parameter in the true utility function as well as updating the prior distributions for the parameters in the true utility function and the variance. The first method considers the situation where we calculate the expectations by considering all the cases simultaneously (a form of batch learning), and the second method deals with the situation where the cases are considered in sequence, thereby facilitating an adaptation scheme. Both methods rely on a Markov Chain Monte Carlo technique, and details hereof are described in Section 4.3.

4.1. Method 1

In order to avoid the computational difficulty of the method described above, we can exploit some of the independence properties of the model in Fig. 1 when calculating

$$\mathbb{E}[\psi_j^* \mid \mathcal{D}] = \int_{\psi_j^*} \psi_j^* P(\psi_j^* \mid \mathcal{D}) \, \mathrm{d}\psi_j^*, \quad \forall 1 \leqslant j \leqslant m.$$

First of all, we shall assume that the prior distribution of each parameter, ψ_j^*, in the true utility function and its associated precision, $\tau_j = 1/\sigma_j^2$, follows a normal-gamma distribution (which is a conjugate distribution for the normal distribution): The conditional distribution of ψ_j^* given τ_j and the (user specified) parameter λ_j^0 is a normal distribution with mean μ_j^0 and precision $\lambda_j^0 \tau_j$, whereas the marginal distribution for τ_j is a gamma distribution with parameters α_j^0 and β_j^0. Note that λ_j^0 can be seen as a virtual sample size, which can be used to control the variance of the distribution. We shall also assume that $P(\psi_j^{\boldsymbol{c}_i} \mid \mathcal{D}) \approx P(\psi_j^{\boldsymbol{c}_i} \mid \boldsymbol{c}_i)$ is approximately correct. The accuracy of this approximation is heavily dependent on the strength of the dependence between ψ_k^* and $\psi_k^{\boldsymbol{c}_k}$ as well as σ_k^2 and $\psi_k^{\boldsymbol{c}_k}$. When only a small amount of prior knowledge is encoded in the model we will in general have a weak dependence between these parameters, and for this situation we conjecture that the assumption is approximately correct. Now, based on these two assumptions we get (see Appendix A for the derivations as well as the expressions (Eqs. (A.5)–(A.6)) for the updated prior distributions):

$$\mathbb{E}(\psi_j^* \mid \mathcal{D}) \approx \mathbb{E}\big(\psi_j^* \mid \hat{\psi}_j^{\boldsymbol{c}_1}, \ldots, \hat{\psi}_j^{\boldsymbol{c}_N}\big) = \frac{\lambda_j^0 \cdot \mu_j^0 + \sum_{i=1}^N \hat{\psi}_j^{\boldsymbol{c}_i}}{\lambda_j^0 + N}, \tag{10}$$

where $\hat{\psi}_j^{\boldsymbol{c}_i} = \mathbb{E}(\psi_j^{\boldsymbol{c}_i} \mid \boldsymbol{c}_i)$.

In order to calculate $\hat{\psi}_j^{\boldsymbol{c}_i} = \mathbb{E}(\psi_j^{\boldsymbol{c}_i} \mid \boldsymbol{c}_i)$ we construct a Markov chain for each of the cases. The state space of each of the chains corresponds to $(\psi^*, \sigma^2, \psi^{\boldsymbol{c}_i})$ and can be represented by the hyper-cube $[0; 1]^{3m}$, where m is the number of parameters in the true utility function. From a given chain we can generate a set of samples for the utility function corre-

sponding to the associated case $\boldsymbol{c}_i$. These samples are then used to approximate the values in each utility function $\psi^{\boldsymbol{c}_i}$:

$$\mathbb{E}\big[\psi_j^{\boldsymbol{c}_i} \mid \boldsymbol{c}_i\big] = \int_{\psi_j^{\boldsymbol{c}_i}=0}^{1} \psi_j^{\boldsymbol{c}_i} P\big(\psi_j^{\boldsymbol{c}_i} \mid \bar{c}_i\big)\, \mathrm{d}\psi_j^{\boldsymbol{c}_i} \approx \frac{1}{n}\sum_{j=1}^{n} x_j, \tag{11}$$

where $\{x_1, \ldots, x_n\}$ are samples from $P(\psi_j^{\boldsymbol{c}_i} \mid \boldsymbol{c}_i)$. The approximated values are then treated as actual observations of $\psi^{\boldsymbol{c}_i}$ (i.e., they can be seen as a representation of the case), and used to determine the expected value of each parameter in the true utility function. It is important to note that the approximated values may *not* specify a utility function which is consistent with the case in question. The problem is that the constraints imposed by a case do not necessarily specify a convex region. Finally, as also described above, the approximated values can also be used to simply estimate the posterior probability distributions $P(\psi_j^*, \sigma_j^2 \mid \mathcal{D}) \approx P(\psi_j^*, \sigma_j^2 \mid \hat{\psi}_j^{\boldsymbol{c}_1}, \ldots, \hat{\psi}_j^{\boldsymbol{c}_N})$ and $P(\psi_j^* \mid \mathcal{D}) \approx P(\psi_j^* \mid \hat{\psi}_j^{\boldsymbol{c}_1}, \ldots, \hat{\psi}_j^{\boldsymbol{c}_N})$, see Appendix A for the derivation.

4.2. Method 2

As an alternative to the method above, we can instead iteratively update the prior distribution, over the true utility function and the variance. This approach facilitates an adaptation of the probability distributions over the parameters as new cases arrive. However, instead of updating the joint distribution over the parameters we shall use the set of estimated posteriors as the new updated prior distributions.

As we only consider one case at a time, we can construct a Markov chain as described above, i.e., a chain with a state space corresponding to the hyper-cube $[0; 1]^{3m}$. An immediate approach for iteratively updating the prior distribution would then be to use the generated samples to calculate the maximum likelihood parameters for the joint normal-gamma distribution. That is, by assuming that the samples for each utility parameter in the true utility function and the associated precision come from a normal-gamma distribution, these samples could then be used for calculating the ML parameters of this distribution. For the utility parameters this is straightforward as we simply need the sample mean. For the gamma distribution, we need to consider the ML-estimators for both α and β: Assume that $x_1, \ldots, x_n$ form a sample from a gamma distribution with parameters α and β:

$$f(x) = \begin{cases} \dfrac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x) & \text{for } x > 0, \\ 0 & \text{for } x \leqslant 0. \end{cases} \tag{12}$$

Hence, for the joint distribution we have:

$$L(\alpha, \beta) = \prod_{i=1}^{n} f(x_i) = \frac{\beta^{n\alpha}}{[\Gamma(\alpha)]^n} \left(\prod_{i=1}^{n} x_i\right)^{\alpha-1} \exp\left(-\beta \sum_{i=1}^{n} x_i\right). \tag{13}$$

When determining the ML estimates for α and β we need to solve the following equation:

$$\ln(\alpha) - \frac{\partial}{\partial \alpha} \ln\big(\Gamma(\alpha)\big) = \ln(\bar{x}_n) - \frac{1}{n}\sum_{i=1}^{n} \ln(x_i). \tag{14}$$

The digamma function, $\frac{\partial}{\partial\alpha}\ln(\Gamma(\alpha))$, cannot be evaluated analytically, however, there exists several accurate approximations, see [13] for an overview. Based on such an approximation we can solve the equation using numerical methods, e.g., Newton–Raphson iteration; as a staring point, the search can be initialized based on the first and second moment:

$$\mathbb{E}(X) = \frac{\alpha}{\beta} \quad \text{and} \quad \mathbb{E}(X^2) = \frac{\alpha(\alpha+1)}{\beta^2}.$$

Alternatively, we could directly use the values for α and β s.t. they correspond to the first and second moment of the samples from the estimated distribution. This avoids the numerical approximation of the digamma function.

The accuracy of the above two approaches is heavily dependent on $P(\tau_j \mid \boldsymbol{c})$ being (approximately) a gamma distribution. To determine how close $P(\tau_j \mid \boldsymbol{c})$ is at being a gamma distribution, we performed a χ^2 goodness-of-fit test w.r.t. samples generated for the Mildew model in Section 5. As a null-hypothesis we assumed that the samples were generated from a gamma distribution (using the ML parameters described above), and this hypothesis was accepted with probability less than 10^{-30}; that the discrepancy also has an impact on the convergence properties of the algorithm was confirmed by empirical experiments. As a consequence, we instead calculate the expected value of each utility parameter for the associate case and used these values to update the distribution over each parameter in the true utility function and the variance, i.e., we basically apply method 1 iteratively, but only with one case at a time keeping the factorization of the probability distribution; the actual procedure for updating the probabilities is given by Eqs. (A.5)–(A.6) in Appendix A. Note that this is an approximation since the distribution does not factorize in this way after evidence has been inserted. The approximation can, however, be avoided by using the normal-Wishart prior distribution, but in this case we cannot exploit any independence relations during sampling and we would therefore need a larger set of samples to get an accurate estimate (see Section 4.3).[7] It is important to emphasize that even though this approach can be used to learn a utility function from a database by considering the cases one at a time, an error from the initial cases (introduced by the approximation) can propagate to subsequent cases. After convergence, however, we expect that such errors will have less influence since we are working with a more informed prior distribution; this is also confirmed by the empirical experiments (see Section 5).

Finally, it should be noted that λ_j can be used as a fading factor that controls how much the past should be taken into account when new cases arrive. This could for instance be used to model that the DM's utility function shifts over time.

4.3. Constructing the Markov chains

The construction of the Markov Chain now remains to be discussed. First of all, note that Applegate and Kannan [2] propose a method for sampling from a log-concave function and approximating the volume of a convex body. However, even though we are (initially) working with log-concave functions (both the normal distribution and the

[7] Note that the updating rules derived in Appendix A can be adapted to the normal-Wishart distribution.

gamma distribution are log-concave and this class of functions is closed under multiplication) this no longer holds in the truncated case defined by the instantiations. Instead we use a standard Metropolis-Hastings algorithm, over the state space for all the parameters $(\psi_1^*, \ldots, \psi_m^*, \sigma_1^2, \ldots, \sigma_m^2, \psi_1^{\boldsymbol{c}}, \ldots, \psi_m^{\boldsymbol{c}})$; we shall use $\bar{x}_t$ to denote a configuration for these parameters at step t and use $X^i = x_t^i$ (for $1 \leqslant i \leqslant 3m$) to denote the value of the ith parameter at step t. We apply a proposal function $q(\cdot \mid \cdot)$ that randomly (with uniform probability) picks one parameter to change, and then proposes a new value for this parameter according to a random walk:

$$\begin{aligned}\bar{x}_t &= \big[\psi_1^*, \ldots, \psi_m^*, \sigma_1^2, \ldots, \sigma_m^2, \psi_1^{\boldsymbol{c}}, \ldots, \psi_k^{\boldsymbol{c}}, \ldots, \psi_m^{\boldsymbol{c}}\big] \\ &\mapsto \bar{x}' = \big[\psi_1^*, \ldots, \psi_m^*, \sigma_1^2, \ldots, \sigma_m^2, \psi_1^{\boldsymbol{c}}, \ldots, x_k, \ldots, \psi_m^{\boldsymbol{c}}\big]. \end{aligned} \tag{15}$$

Thus, at most one parameter is changed at each step, and the algorithm accepts this move with probability:

$$\min\left\{1, \frac{P(\bar{x}')q(\bar{x}_t \mid \bar{x}')}{P(\bar{x}_t)q(\bar{x}' \mid \bar{x}_t)}\right\}, \tag{16}$$

given that the state $\bar{x}'$ satisfies the constraints encoded by $\boldsymbol{c}$; if these constraints are not satisfied we stay at the current state $\bar{x}_t$, i.e., $\bar{x}_{t+1} = \bar{x}_t$. Observe that we need not compute the joint probability $P(\bar{x})$ as we can exploit the factorization of the probability distribution when calculating the acceptance probability: E.g., if $X^i = \psi_j^{\boldsymbol{c}}$, then:

$$\frac{P(\bar{x}')}{P(\bar{x}_t)} = \frac{P(X^i = x' \mid \psi_{j,t}^*, \sigma_{j,t}^2)}{P(X^i = x_t^i \mid \psi_{j,t}^*, \sigma_{j,t}^2)}.$$

This also applies for the other parameters except that we also have to take their children into account. That is, for, e.g., $X^i = \psi_j^*$ we get:

$$\frac{P(\bar{x}')}{P(\bar{x}_t)} = \frac{P(\psi_{j,t}^{\boldsymbol{c}} \mid X_i = x', \sigma_{j,t}^2)P(X_i = x' \mid \sigma_{j,t}^2)}{P(\psi_{j,t}^{\boldsymbol{c}} \mid X_i = x_t^i, \sigma_{j,t}^2)P(X_i = x_t^i \mid \sigma_{j,t}^2)}.$$

The algorithm can now be summarized as follows:

Algorithm 1. Initialize the Markov Chain at an arbitrary starting point $\bar{x}^0$ s.t. for case $\boldsymbol{c}_i$, the utilities $\psi_1^{\boldsymbol{c}_i}, \ldots, \psi_m^{\boldsymbol{c}_i}$ satisfy the constraints induced by that case.[8]

1. Sample a parameter index i from a uniform distribution over $\{1, 2, \ldots, 3m\}$.
2. Sample a point x' from $q(\cdot \mid x_t^i)$, where
$$q\big(\cdot \mid x_t^i\big) \sim N\big(x_t^i, \sigma^2\big).$$
3. Set $\bar{x}_{t+1} = \bar{x}' = (x_t^1, \ldots, x_t^{i-1}, x', x_t^{i+1}, \ldots, x_t^{3\cdot m})$ with probability:
$$\min\left\{1, \frac{P(\bar{x}')q(\bar{x}_t \mid \bar{x}')}{P(\bar{x}_t)q(\bar{x}' \mid \bar{x}_t)}\right\},$$
given that $\bar{x}'$ satisfies the constraints; otherwise stay at the current position.

[8] This starting point may simply correspond to any trivial utility function.

4. Set $t := t + 1$ and goto step 1.

In the algorithm above, it should be noted that we only need to test whether $\bar{x}'$ satisfies the constraints if X_i corresponds to a parameter in the utility function for case $\boldsymbol{c}_i$.

4.3.1. Testing the proposed utility function

The most computationally intensive step of the algorithms is to determine whether a proposed parameter change leads to a utility function which induces the case in question; on average, this test should be performed for every third generated sample since the candidate parameter is sampled from a uniform probability distribution over $\{1, 2, \ldots, 3m\}$ (assuming that the candidate point is initially accepted in step 3 of Algorithm 1).

A straightforward approach for performing this test is simply to solve the influence diagram w.r.t. the proposed utility function, and then determine if the case is consistent with the optimal strategy. However, even though this may be computationally feasible for some domains we can actually avoid some of these calculations. First of all, we need not evaluate the influence diagram if the change in parameter value is less than the difference in expected utility between the optimal decision option and the next best decision option for each decision variable; if this is the case, then changing the parameter will have no impact on the optimal strategy (note that this difference should be stored for subsequent evaluations). Secondly, by analyzing the structure of the ID and by reusing previous calculations we need not perform a complete propagation when testing a new utility function:

- We only need to consider decision variables for which the proposed utility parameter is relevant, see [19,25].
- No probability potential needs to be updated as only utility parameters are changed, see also [20].

Note that for the second case we may simply store the probability potentials calculated in the first propagation, and then reuse these probability potentials in all subsequent propagations. For example, the lazy propagation architecture [17] facilitates such type of propagation by maintaining the sets of potentials.

5. Empirical results

In order to test the proposed algorithms we have applied them to the Mildew network and the Poker network. The Mildew network (depicted in Fig. 2(a)) contains 21 utility parameters in the range $[-1; 13]$, and the Poker network (depicted in Fig. 2(b)) contains 4 utility parameters in the range $[-1; 3]$; the number of possible observation-decision sequences for the two networks are 1344 and 186624, respectively (ignoring zero probabilities). A more detailed description of the two networks can be found in [12].

For each of the networks we generated a database of instantiations by first adding white noise to the utility parameters in the original utility function. Each modified utility function was then used to sample a single instantiation of the corresponding ID; when generating a dataset of size N this procedure was therefore performed N times. To measure the ro-

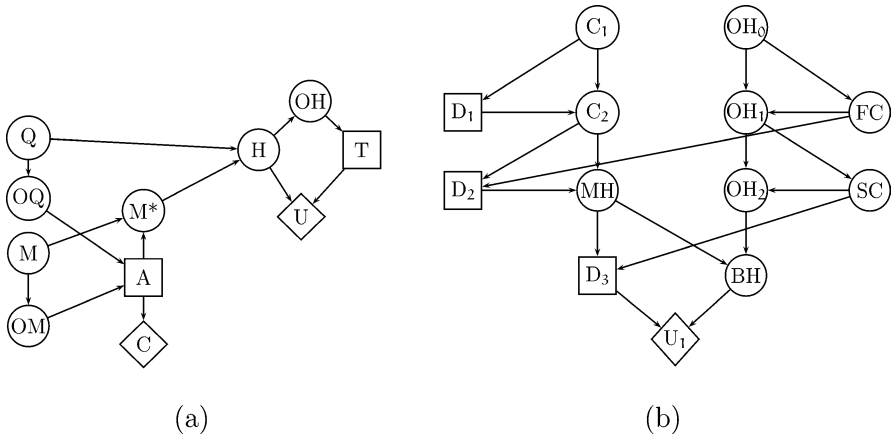

Fig. 2. Figure (a) shows an influence diagram representation of the Mildew problem, and figure (b) shows an influence diagram representation of the Poker problem.

bustness of the algorithms w.r.t. inconsistent behavior we performed three tests where the noise added to the utilities were sampled from a normal distribution with mean 0 and with variance 1, 2 and 3, respectively. We used a burn-in phase of 2000 steps and afterwards we ran the Markov chains for 98000 steps. For the batch learning approach, the results (averaged over 10 runs) of the experiments for the Mildew network are shown in Fig. 3. More precisely, the figure depicts the average expected utility of the learned strategy (i.e., the optimal strategy induced by the learned utility function) w.r.t. the true utility function; the error-bars correspond to the empirical standard deviation and the horizontal lines show the maximum and minimum expected utility.

For the adaption approach, the results (averaged for 10 runs) of the experiments are shown in Fig. 4; the figures depict the average expected utility of the learned strategy w.r.t. the true utility function. As was expected, and which can be seen from the figures, the amount of noise introduced in the data has an impact on the convergence properties of the algorithm. Moreover, after convergence, the algorithm exhibit (on average) larger fluctuations as more noise is introduced: for more than 16 cases, the average empirical standard deviation is 0.065, 0.079 and 0.194 for the three experiments, respectively.

We also tested how robust the learning algorithms are in situations where the data has not been generated according to the model. Specifically, we generated databases as described previously, however, instead of modifying the values of the utility function with noise sampled from a normal distribution with fixed variance, we added noise where the variance of the noise was sampled from a uniform distribution in the interval $[0; 3]$. The results of the test are depicted in Fig. 5(a). Moreover, we also tested how the adaptation procedure performs when the DM's behavior shifts (drastically) over time. The test was performed by employing the adaptation procedure for 20 cases generated as above with variance 2. Afterwards the values in the original utility function were "reversed" (producing a radical different behavior) and an additionally 30 cases were generated in a similar fashion but with the new utility function. The result of the experiment is depicted in Fig. 5(b), which shows that the algorithm is able to adapt to the changed behavior.

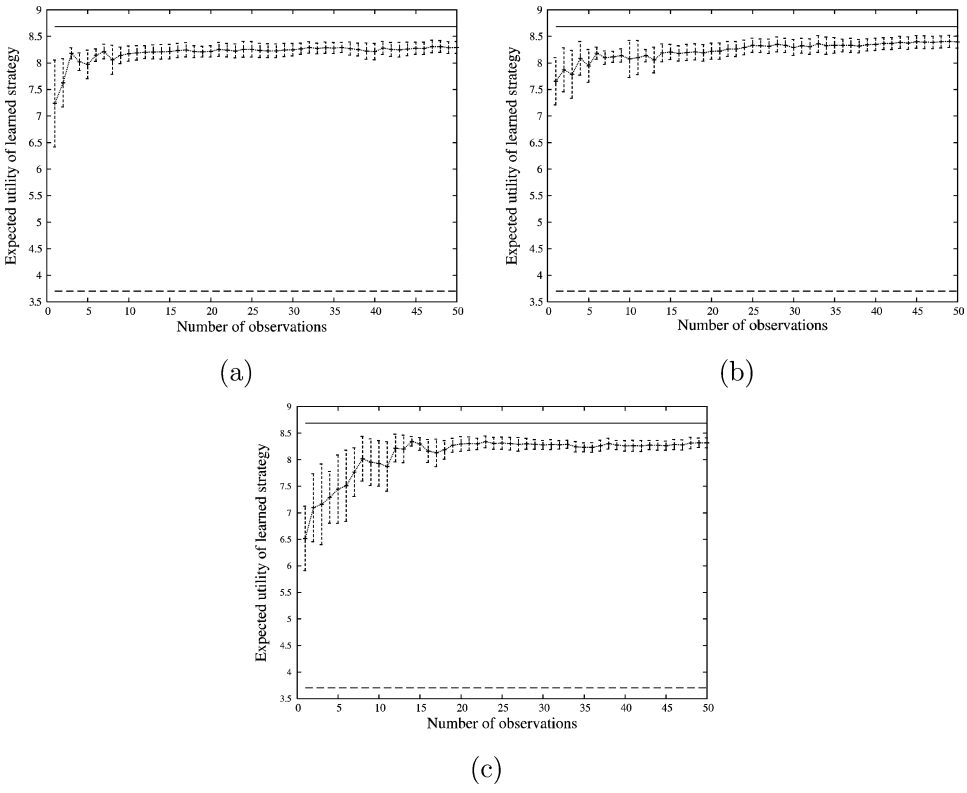

Fig. 3. The figures show the expected utility of the learned strategy for the Mildew network as a function of the size of the database (using batch learning); the results are averaged over 10 runs and there are 1344 possible observation-decision sequences. For figures (a), (b) and (c) the databases were generated by modifying the original utility function by adding white noise with variance 1, 2 and 3, respectively.

Finally, we have made a similar set of tests for the Poker domain. The results for this domain exhibit the same type of behavior as discussed above. For example, Fig. 6(a) shows the results (averaged over 10 runs) of applying the batch learning algorithm in the Poker domain; the database was generated by modifying the original utility function by adding noise with mean 0 and variance 1.[9] Fig. 6(b) shows the application of the adaptation algorithm using the same setting as for the batch learning algorithm. Figs. 6(c) and 6(d) show the results for the situation where the database was generated by modifying the original utility function by adding noise with mean 0 and variance 3.

Analogously to the Mildew network, we also tried to simulate a shift in the DM's preferences after 20 observations. As before, each observation was sampled after having modified the original utility function by adding noise with mean 0 and variance 1, and after 20 cases the values in the original utility function were switched. The result is shown in Fig. 7 which illustrates that the algorithm is able to take the changed behavior into account during adap-

[9] Note that the y-axis has been scaled for illustration purposes.

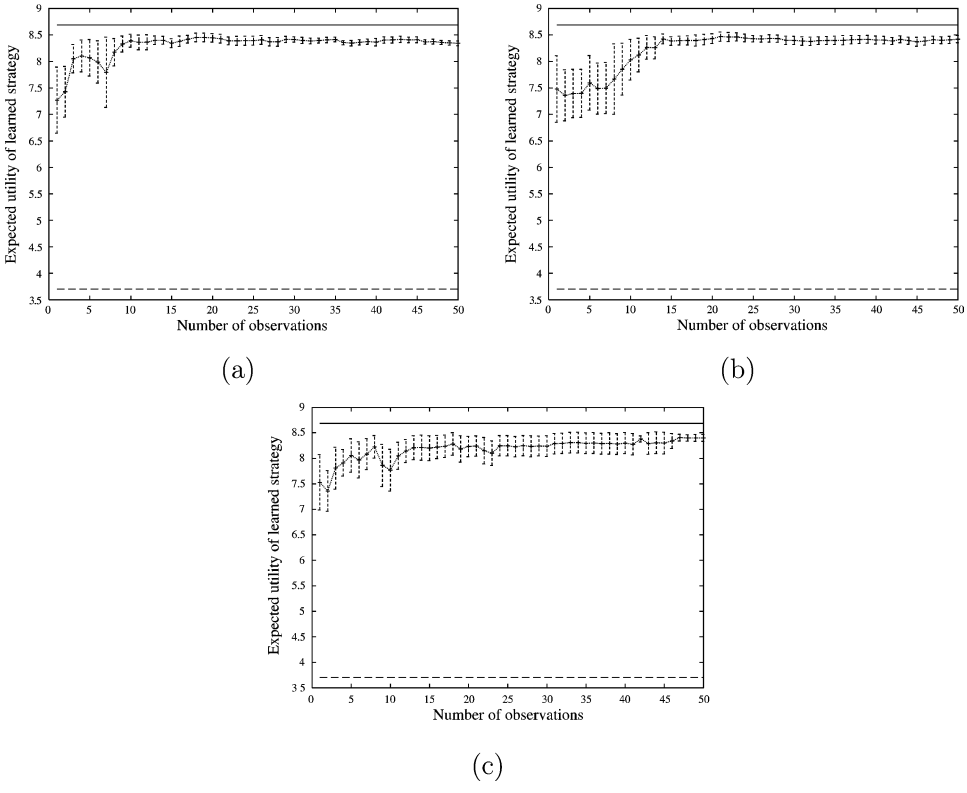

Fig. 4. The figures show the expected utility of the learned strategy for the Mildew network as a function of the number of cases (using adaptation); the results are averaged over 10 runs and there are 1344 possible observation-decision sequences. For figures (a), (b) and (c) the cases were generated by modifying the original utility function by adding white noise with variance 1, 2 and 3, respectively.

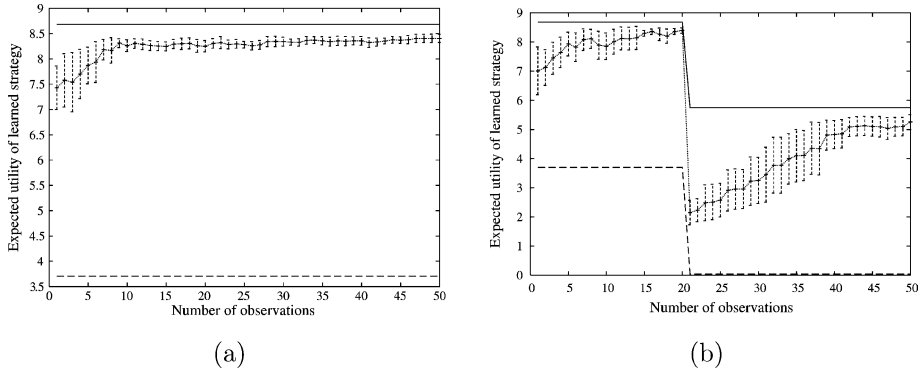

Fig. 5. Figure (a) shows the expected utility of the learned strategy for the Mildew network averaged over 10 runs (using adaptation). The databases were generated by modifying the original utility function with noise from a normal distribution with mean 0 and variance sampled from a uniform distribution in the interval [0; 3]. Figure (b) shows the impact of changing the behavior after 20 cases.

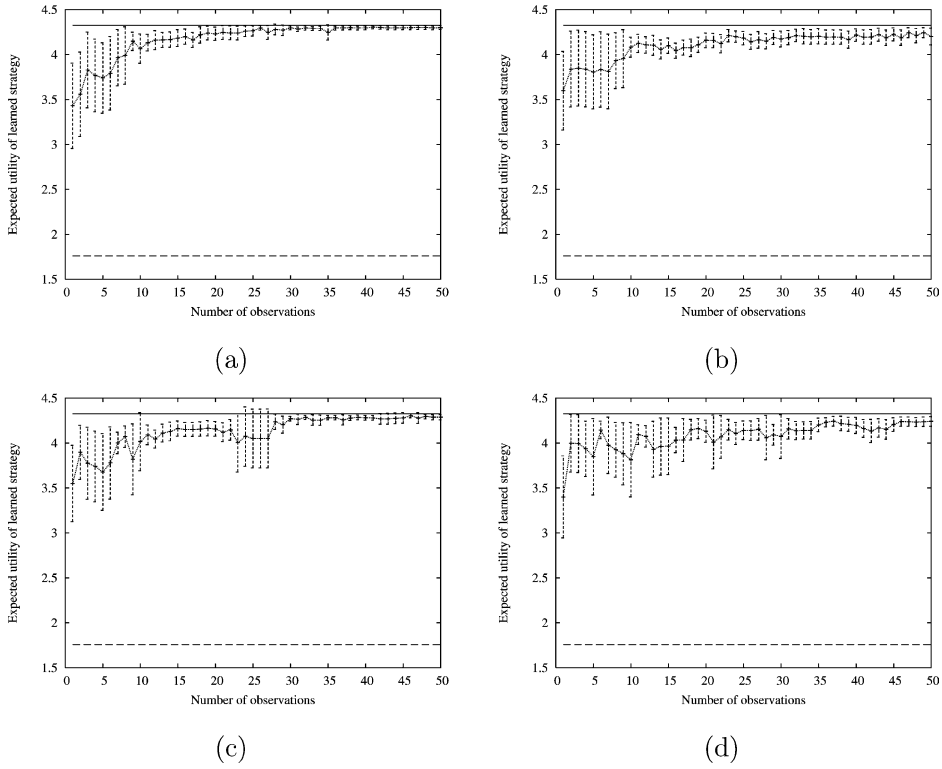

Fig. 6. The figures show the expected utility of the learned strategy for the Poker network averaged over 10 runs; there are 186624 possible observation-decision sequences. The two databases were generated by modifying the original utility function with noise from a normal distribution with mean 0 and variance 1. The results in figure (a) were generated by the batch learning algorithm, and the results in figure (b) were generated by the adaptation algorithm. Figures (c) and (d) show the results for the situation where the database was generated by modifying the original utility function by adding noise with mean 0 and variance 3.

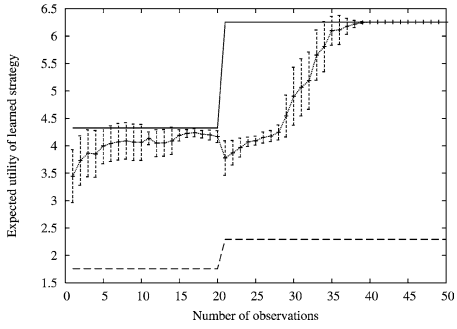

Fig. 7. The figure show the expected utility of the learned strategy for the Poker network as a function of the number of cases (using adaptation); there are 186624 possible observation-decision sequences. After 20 cases the values in the original utility function were shifted to simulate a change in the DM's preference. All observations were generated by modifying the original utility function by adding noise with mean 0 and variance 1.

tation. In the following section we shall discuss different methods for detecting such shifts in preferences (to obtain a faster convergence) which could be taken into account during adaptation.

6. Discussion and future work

As previously mentioned, the most computationally intensive step of the algorithms is to test whether a proposed move leads to a utility function which is consistent with the case in question. As an alternative to the methods proposed in Section 4.3.1, one could instead pose the problem as performing a type of sensitivity analysis, see, e.g., [9,20]. However, it is important to notice that there is a fundamental difference in the two problems: we seek to determine whether changing a parameter leads to a utility function which induces an optimal strategy including the case in question, whereas sensitivity analysis has to do with changes in the optimal strategy. This also implies that the methods proposed in, e.g., [20] cannot immediately be applied; these methods rely on the expected utility functions being linear in the utilities, and this only holds when the optimal strategy is known (i.e., in our case, fully encoded in the database).

Thus, in order to apply the methods described by Nielsen and Jensen [20] we could instead induce the "missing" cases: Start with the last decision, D_n, and determine the linear inequalities corresponding to each case by restricting them to D_n, see Eq. (6). Next, use these inequalities for checking the constraints in step 3 of Algorithm 1 and determine a utility function for each of the cases (again restricted to D_n). After the utility functions have been found we calculate an optimal policy for each case, and continue to decision D_{n-1}. Now, in order to compute the required inequalities for decision D_{n-1} (for a given case) we first substitute decision D_n with its chance variable representation (determined by the optimal policy induced by the utility function identified in the previous step), and decision D_{n-1} is then treated as the last decision. This procedure then continues to the first decision variable, where all unrestricted cases are considered.

Algorithm 2. Let $\mathcal{D}$ be a database consisting of N cases $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$, where each case specifies a realization of a strategy for the influence diagram I with decision variables $\{D_1, \ldots, D_n\}$ ordered by their indexes.

1. Set $i := n$.
2. Repeat
3. For each case $\boldsymbol{c}$, calculate the expected value for all parameters $\psi_j^{\boldsymbol{c}}$.
4. Use the estimated values to complete the optimal policy for decision D_i in each case.
5. For each case substitute decision D_i with its chance variable policy.
6. Set $i := i - 1$.
7. Until $i = 0$.

Note that as all the inequalities encode convex sets we are always working with a convex set during sampling; these sets are closed under intersection. Thus we can apply the sampling method proposed by Applegate and Kannan [2].

An interesting approach for making the learning algorithm more robust, could be to weight the contribution from each case (i.e., changing the updating rule for λ_j) in proportion to the probability of actually seeing that case. By weighting each case we can ensure that cases with low probability have less impact on the learning procedure; this is consistent with their contribution to the maximum expected utility. Moreover, by observing the "case probabilities" during adaptation we could, e.g., use the occurrence of low probability cases as an indication of a shift in the DM's preferences (producing a different behavior); thereby being able to change the value of λ_j accordingly and allowing a more rapid adaptation. Furthermore, it could be interesting to extend the proposed algorithm to handle cases with missing values. One obvious approach to this problem would be to induce the missing data, however, it is not completely apparent how these (partially induced) cases should be weighted as compared to fully observed cases.

Finally, as also discussed in Section 3, we could base the learning methods on a descriptive model of the DM rather than a prescriptive. That is, acknowledging that humans rarely behave as to maximize the expected utility, we could for instance use the rank dependent utility (RDU) model by Quiggin [21] during learning. In the RDU model, there are two parameters: (i) a utility function $u(\cdot)$ which is cardinal and represents the DM's preferences under certainty, and (ii) a *probability transformation* (or *weighting*) function $q(\cdot)$, which is strictly increasing from $[0, 1]$ onto itself. The utility $V(L)_{u,q}$ of a single stage lottery L is determined as follows.[10] First, the components of L are re-indexed in increasing order of their utility:

$$L = (c_1, p_1; \ldots; c_i, p_i; \ldots; c_n, p_n) \quad \text{with } u(c_1) \leqslant \cdots \leqslant u(c_i) \leqslant \cdots \leqslant u(c_n). \tag{17}$$

Then, its utility is evaluated as:

$$V(L)_{u,q} = \sum_{i=1}^{n} u(c_i) \left[q\left(\sum_{j=i}^{n} p_j \right) - q\left(\sum_{j=i+1}^{n} p_j \right) \right]$$

or, equivalently as

$$V(L)_{u,q} = u(c_1) + \sum_{i=2}^{n} \left[u(c_i) - u(c_{i-1}) \right] q\left(\sum_{j=i}^{n} p_j \right).$$

The crucial idea underlying the RDU model is that the utility of each consequence is given a variable weight, which depends on both its probability and on its ranking w.r.t. the other consequences of the lottery. The second expression suggests that the DM bases his evaluation on the probabilities of achieving at least such or such utility level. Note, however, that RDU theory no longer satisfies the independence axiom, and we cannot directly exploit the properties of dynamic programming [3] when dealing with sequential decision problems. That is, the RDU model is computationally more intensive to work with as compared to the EU model, and the integration of the model into the learning procedure is a subject for future research.

[10] Note that under the reduction of compound lotteries assumption, any sequential decision problem can be transformed into a single stage lottery.

7. Conclusion

In this paper we have proposed two algorithms for learning a DM's utility function based on observed (possible inconsistent) behavior. The algorithms extend previous proposed algorithms by explicitly taking into account that there may not exist a non-trivial utility function which can account for all the observed behavioral patterns, i.e., behavior which jointly appear inconsistent w.r.t. any utility function. The proposed algorithms are based on the assumption that inconsistent behavior can be explained as deviations from an underlying true utility function, and as such it can, e.g., (i) represent (or encode) inaccuracies in the model being applied [26], and (ii) model a DM whose utility function fluctuates and/or changes permanently over time. The main difference in the two algorithms is that the first algorithm can be seen as a form of batch learning whereas the second algorithm focuses on adaptation. Empirical results demonstrated the applicability of both algorithms, which converged to the true utility function even for very small databases. Additionally, it was shown that rapid permanent shifts in the utility functions can also be taken into account during adaptation.

Acknowledgements

This paper has benefited from discussions with the members of the Decision Support Systems group at Aalborg University, and in particular Helge Langseth. We would also like to thank `Hugin Expert` (www.hugin.com) for giving us access to the *Hugin Decision Engine* which forms the basis for our implementation. Finally, we would like to thank the anonymous reviewers for their constructive comments.

Appendix A. Proofs

In order to estimate $\mathbb{E}(\psi_j^* \mid \mathcal{D})$ we first note that:

$$
\begin{aligned}
&P\big(\psi^*, \sigma^2 \mid \mathcal{D}\big) \\
&\quad = \int_{\psi^{\boldsymbol{c}_1}} \cdots \int_{\psi^{\boldsymbol{c}_N}} P\big(\psi^*, \sigma^2, \psi^{\boldsymbol{c}_1}, \ldots, \psi^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi^{\boldsymbol{c}_1} \ldots \mathrm{d}\psi^{\boldsymbol{c}_N} \\
&\quad = \int_{\psi^{\boldsymbol{c}_1}} \cdots \int_{\psi^{\boldsymbol{c}_N}} P\big(\psi^*, \sigma^2 \mid \psi^{\boldsymbol{c}_1}, \ldots, \psi^{\boldsymbol{c}_N}, \mathcal{D}\big) P\big(\psi^{\boldsymbol{c}_1}, \ldots, \psi^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi^{\boldsymbol{c}_1} \ldots \mathrm{d}\psi^{\boldsymbol{c}_N} \\
&\quad = \int_{\psi^{\boldsymbol{c}_1}} \cdots \int_{\psi^{\boldsymbol{c}_N}} P\big(\psi^*, \sigma^2 \mid \psi^{\boldsymbol{c}_1}, \ldots, \psi^{\boldsymbol{c}_N}\big) P\big(\psi^{\boldsymbol{c}_1}, \ldots, \psi^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi^{\boldsymbol{c}_1} \ldots \mathrm{d}\psi^{\boldsymbol{c}_N} \\
&\quad = \int_{\psi_{\tilde{1}}} \cdots \int_{\psi_{\tilde{m}}} \left(\prod_{j=1}^{m} P\big(\psi_j^*, \sigma_j^2 \mid \psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N}\big) \right) \\
&\qquad \times P\big(\psi^{\boldsymbol{c}_1}, \ldots, \psi^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi_{\tilde{1}} \ldots \mathrm{d}\psi_{\tilde{m}}. \qquad \text{(A.1)}
\end{aligned}
$$

In particular

$$P\big(\psi_j^*, \sigma_j^2 \mid \mathcal{D}\big) = \int_{\psi_j^{\boldsymbol{c}_1}} \cdots \int_{\psi_j^{\boldsymbol{c}_N}} P\big(\psi_j^*, \sigma_j^2 \mid \psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N}\big) \times P\big(\psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi_j^{\boldsymbol{c}_1} \ldots \mathrm{d}\psi_j^{\boldsymbol{c}_N}. \tag{A.2}$$

In order to calculate the term $P(\psi_j^*, \sigma_j^2 \mid \psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N})$ on closed form we need a conjugate prior distribution for the true utility function and the variance. First, recall that the conditional probability distribution for the utility parameter $\psi_j^{\boldsymbol{c}_i}$ of some case $\boldsymbol{c}_i$ is a normal distribution which, for notational convenience, can be described in terms of its precision $\tau_j = 1/\sigma_j^2$:

$$\psi_j^{\boldsymbol{c}_i} \sim N(\psi_j^*, \tau_j) = \left(\frac{\tau_j}{2\pi}\right)^{1/2} \exp\left[-\frac{1}{2}\tau_j\big(\psi_j^{\boldsymbol{c}_i} - \psi_j^*\big)^2\right]. \tag{A.3}$$

A possible conjugate family of joint prior distributions for ψ_j^* (the mean) and τ_j (the precision), is the joint normal-gamma distribution specified by the conditional distribution of ψ_j^* given τ_j and the marginal distribution of τ_j:

$$P\big(\psi_j^* \mid \tau_j, \mu_j^0, \lambda_j^0\big) = \left(\frac{\lambda_j^0 \tau_j}{2\pi}\right)^{1/2} \exp\left[-\frac{1}{2}\lambda_j^0 \tau_j\big(\psi_j^* - \mu_j^0\big)^2\right],$$

$$P\big(\tau_j \mid \alpha_j^0, \beta_j^0\big) = \begin{cases} \dfrac{(\beta_j^0)^{\alpha_j^0}}{\Gamma(\alpha_j^0)} \tau_j^{\alpha_j^0 - 1} \mathrm{e}^{-\beta_j^0 \tau_j} & \text{for } \tau_j > 0, \\ 0 & \text{for } \tau_j \leqslant 0 \end{cases} \tag{A.4}$$

for $-\infty < \mu_j^0 < \infty$, $\lambda_j^0 > 0$, $\alpha_j^0 > 0$ and $\beta_j^0 > 0$. That is, the conditional distribution of ψ_j^* given τ_j is a normal distribution with mean μ_j^0 and precision $\lambda_j^0 \tau_j$, whereas the marginal distribution for τ_j is a gamma distribution with parameters α_j^0 and β_j^0. Note that λ_j^0 can be see as a virtual sample size, which can be used to control how certain we are about the distribution.

In the general case, suppose now that $X_1, X_2, \ldots, X_n$ form a random sample from a normal distribution for which both the mean μ and the precision τ are unknown. If the joint prior distribution of μ and τ is as specified above, then the joint posterior of μ and τ, given that $X_i = x_i$ $(\forall 1 \leqslant i \leqslant n)$ is as follows: The conditional distribution of μ given τ is a normal distribution with mean μ^1 and precision $\lambda^1 \tau$, where:

$$\mu^1 = \frac{\lambda^0 \mu^0 + n\bar{x}_n}{\lambda^0 + n} \quad \text{and} \quad \lambda^1 = \lambda^0 + n; \tag{A.5}$$

$\bar{x}_n$ is the sample mean, and the marginal distribution of τ is a gamma distribution with parameters α^1 and β^1:[11]

$$\alpha^1 = \alpha^0 + \frac{n}{2} \quad \text{and} \quad \beta^1 = \beta^0 + \frac{1}{2}\sum_{i=1}^{n}(x_i - \bar{x}_n)^2 + \frac{n\lambda^0(\bar{x}_n - \mu^0)^2}{2(\lambda^0 + n)}. \tag{A.6}$$

In particular, we can find the expected value of μ (corresponding to the parameters of the true utility function) since $\mathrm{E}(\mu) = \mu^1$ and $\mathrm{Var}(\mu) = \frac{\beta^1}{\lambda^1(\alpha^1 - 1)}$.[12] Thus, we have:

$$\begin{aligned}
&\mathbb{E}\big(\psi_j^* \mid \mathcal{D}\big) \\
&\quad = \int_{\psi_j^{\boldsymbol{c}_1}} \cdots \int_{\psi_j^{\boldsymbol{c}_N}} \mathbb{E}\big(\psi_j^* \mid \psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N}\big) P\big(\psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi_j^{\boldsymbol{c}_1} \ldots \mathrm{d}\psi_j^{\boldsymbol{c}_N} \\
&\quad = \int_{\psi_j^{\boldsymbol{c}_1}} \cdots \int_{\psi_j^{\boldsymbol{c}_N}} \left(\frac{\lambda_j^0\mu_j^0 + \sum_{i=1}^{N}\psi_j^{\boldsymbol{c}_i}}{\lambda_j^0 + N}\right) P\big(\psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi_j^{\boldsymbol{c}_1} \ldots \mathrm{d}\psi_j^{\boldsymbol{c}_N} \\
&\quad = \frac{\lambda_j^0\mu_j^0}{\lambda_j^0 + N} + \frac{1}{\lambda_j^0 + N}\int_{\psi_j^{\boldsymbol{c}_1}} \cdots \int_{\psi_j^{\boldsymbol{c}_N}} \sum_{i=1}^{N}\psi_j^{\boldsymbol{c}_i} P\big(\psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi_j^{\boldsymbol{c}_1} \ldots \mathrm{d}\psi_j^{\boldsymbol{c}_N} \\
&\quad = \frac{\lambda_j^0\mu_j^0}{\lambda_j^0 + N} + \frac{1}{\lambda_j^0 + N}\sum_{i=1}^{N}\int_{\psi_j^{\boldsymbol{c}_1}} \cdots \int_{\psi_j^{\boldsymbol{c}_N}} \psi_j^{\boldsymbol{c}_i} P\big(\psi_j^{\boldsymbol{c}_1}, \ldots, \psi_j^{\boldsymbol{c}_N} \mid \mathcal{D}\big)\, \mathrm{d}\psi_j^{\boldsymbol{c}_1} \ldots \mathrm{d}\psi_j^{\boldsymbol{c}_N} \\
&\quad = \frac{\lambda_j^0\mu_j^0}{\lambda_j^0 + N} + \frac{1}{\lambda_j^0 + N}\sum_{i=1}^{N}\int_{\psi_j^{\boldsymbol{c}_i}} \psi_j^{\boldsymbol{c}_i} P\big(\psi_j^{\boldsymbol{c}_i} \mid \mathcal{D}\big)\, \mathrm{d}\psi_j^{\boldsymbol{c}_i}.
\end{aligned} \tag{A.7}$$

We shall approximate this expression by assuming that $P(\psi_j^{\boldsymbol{c}_i} \mid \mathcal{D}) \approx P(\psi_j^{\boldsymbol{c}_i} \mid \boldsymbol{c}_i)$. The accuracy of this approximation is heavily dependent on the strength of the dependence between ψ_k^* and $\psi_k^{\boldsymbol{c}_i}$ as well as σ_k^2 and $\psi_k^{\boldsymbol{c}_k}$; with little prior knowledge encoded in the model these dependencies will be weak (encoded by a large variance or small λ). Given this assumption, $\mathbb{E}(\psi_j^* \mid \mathcal{D})$ can now be expressed as:

$$\mathbb{E}(\psi_j^* \mid \mathcal{D}) \approx \frac{\lambda_j^0\mu_j^0}{\lambda_j^0 + N} + \frac{1}{\lambda_j^0 + N}\sum_{i=1}^{N}\int_{\psi_j^{\boldsymbol{c}_i}} \psi_j^{\boldsymbol{c}_i} P\big(\psi_j^{\boldsymbol{c}_i} \mid \boldsymbol{c}_i\big)\, \mathrm{d}\psi_j^{\boldsymbol{c}_i}$$

[11] Note also that when updating the prior distribution based on a random sample of size n we also increase the virtual sample size λ^0 with n.

[12] This is actually an approximation as the normal-gamma probability distribution only serves as a conjugate prior for the unrestricted normal distribution and not when it is truncated.

$$= \frac{\lambda_j^0 \mu_j^0}{\lambda_j^0 + N} + \frac{1}{\lambda_j^0 + N} \sum_{i=1}^{N} \hat{\psi}_j^{\boldsymbol{c}_i}$$
$$= \frac{\lambda_j^0 \mu_j^0 + \sum_{i=1}^{N} \hat{\psi}_j^{\boldsymbol{c}_i}}{\lambda_j^0 + N} = \mathbb{E}\big(\psi_j^* \mid \hat{\psi}_j^{\boldsymbol{c}_1}, \ldots, \hat{\psi}_j^{\boldsymbol{c}_N}\big), \tag{A.8}$$

where $\hat{\psi}^{\boldsymbol{c}_i} = \mathbb{E}(\psi^{\boldsymbol{c}_i} \mid \boldsymbol{c}_i)$.

References

[1] M. Allais, The foundation of a positive theory of choice involving risk and a criticism of the postulate and axioms of the American school, in: Expected Utility Hypotheses and the Allais Paradox, Dordrecht, Holland, 1979, pp. 27–145. Original work published in 1952.

[2] D. Applegate, R. Kannan, Sampling and integration of near log-concave functions, in: Proceedings of the 23rd Annual ACM Symposium on Theory and Computing, New Orleans, LA, 1991, pp. 156–163.

[3] R.E. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.

[4] W.L. Buntine, A guide to the literature on learning probabilistic networks from data, IEEE Trans. Knowledge Data Engrg. 8 (1996) 195–210.

[5] U. Chajewska, L. Getoor, J. Norman, Y. Shahar, Utility elicitation as a classification problem, in: G.F. Cooper, S. Moral (Eds.), Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, 1998, pp. 79–88.

[6] U. Chajewska, D. Koller, Utilities as random variables: density estimation and structure discovery, in: C. Boutilier, M. Goldszmidt (Eds.), Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, Stanford, CA, 2000, pp. 63–71.

[7] U. Chajewska, D. Koller, D. Ormoneit, Learning an agent's utility function by observing behavior, in: Proceedings of the Eighteenth International Conference on Machine Learning, Williamstown, MA, 2001, pp. 35–42.

[8] U. Chajewska, D. Koller, R. Parr, Making rational decisions using adaptive utility elicitation, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence, Austin, TX, 2000, pp. 363–369.

[9] J.C. Felli, G.B. Hazen, Sensitivity analysis and the expected value of perfect information, Medical Decision Making 18 (1999) 95–109.

[10] R.A. Howard, J.E. Matheson, Influence diagrams, in: R.A. Howard, J.E. Matheson (Eds.), The Principles and Applications of Decision Analysis, vol. 2, Strategic Decision Group, 1981, pp. 721–762, Chapter 37.

[11] F. Jensen, F.V. Jensen, S.L. Dittmer, From influence diagrams to junction trees, in: R.L. de Mantaras, D. Poole (Eds.), Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, WA, Morgan Kaufmann, San Mateo, CA, 1994, pp. 367–373.

[12] F.V. Jensen, Bayesian Networks and Decision Graphs, Springer, New York, ISBN: 0-387-95259-4, 2001.

[13] N.L. Johnson, S. Kotz, N. Balakrishnan, Continuous Univariate Distributions, second ed., Wiley Series in Probability and Statistics, vol. 1, Wiley, New York, 1994.

[14] D. Kahneman, A. Tversky, Prospect theory: an analysis of decision under risk, Econometrica 47 (1979) 263–291.

[15] M.P. Keane, K.I. Wolpin, The career decisions of young men, J. Political Economy 105 (3) (1997) 473–522.

[16] S.L. Lauritzen, D. Nilsson, Representing and solving decision problems with limited information, Management Sci. 47 (9) (2001) 1235–1251.

[17] A.L. Madsen, F.V. Jensen, Lazy evaluation of symmetric Bayesian decision problems, in: K.B. Laskey, H. Prade (Eds.), Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, Morgan Kaufmann, San Mateo, CA, 1999, pp. 382–390.

[18] A.Y. Ng, S. Russell, Algorithms for inverse reinforcement learning, in: Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, 2000, pp. 663–670.

[19] T.D. Nielsen, F.V. Jensen, Welldefined decision scenarios, in: K.B. Laskey, H. Prade (Eds.), Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, Morgan Kaufmann, San Mateo, CA, 1999, pp. 502–511.

[20] T.D. Nielsen, F.V. Jensen, Sensitivity analysis in influence diagrams, IEEE Trans. Systems Man Cybernet. Part A: Systems and Humans 33 (2) (2003) 223–234.
[21] J. Quiggin, A theory of anticipated utility, J. Economic Behavior and Organisation 3 (4) (1982) 323–343.
[22] J. Rust, Structural estimation of Markov decision processes, in: R.F. Engle, D.L. McFadden (Eds.), Handbook of Econometrics, vol. IV, Elsevier Science, Amsterdam, 1994, pp. 3081–3143.
[23] T.J. Sargent, Estimation of dynamic labor demand schedules under rational expectations, J. Political Economy 86 (6) (1978) 1009–1044.
[24] R.D. Shachter, Evaluating influence diagrams, Oper. Res. Soc. Amer. 34 (6) (1986) 79–90.
[25] R.D. Shachter, Efficient value of information computation, in: K.B. Laskey, H. Prade (Eds.), Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, Morgan Kaufmann, San Mateo, CA, 1999, pp. 594–601.
[26] R.D. Shachter, M. Mandelbaum, A measure of decision flexibility, in: E. Horvitz, F.V. Jensen (Eds.), Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence, Portland, OR, Morgan Kaufmann, San Mateo, CA, 1996, pp. 485–491.
[27] D. Suryadi, P.J. Gmytrasiewicz, Learning models of other agents using influence diagrams, in: Proceedings of the Seventh International Conference on User Modeling, 1999, pp. 223–232.
[28] J.A. Tatman, R.D. Shachter, Dynamic programming and influence diagrams, IEEE Trans. Systems Man Cybernet. 20 (2) (1990) 365–379.