In *Proceedings of the Workshop "Learning About Users"* at IJCAI99, the Sixteenth International Joint Conference on Artificial Intelligence. Stockholm, July 31st, 1999. (This paper avaliable from http://w5.cs.uni-sb.de/ ready/.)

Learning Bayesian Networks With Hidden Variables for User Modeling

Barbara Großmann-Hutter, Anthony Jameson, Frank Wittig*
Department of Computer Science, University of Saarbrücken
P.O. Box 151150, D-66041 Saarbrücken, Germany

Abstract

We present issues and initial results of our research into methods for learning Bayesian networks for user modeling on the basis of empirical data, focusing on issues that are especially important in the context of user modeling. These issues include the treatment of theoretically interpretable hidden variables, ways of learning partial networks and combining them into a single network, ways of taking into account the special properties of datasets acquired through psychological experiments, and ways of increasing the efficiency and effectiveness of the learning algorithms.

1 Introduction

1.1 Goals

This paper focuses on a type of machine learning technique that has so far seldom been employed for learning about users: the learning of Bayesian networks (BNs) on the basis of empirical data. The general problem of learning BNs has been a topic of intensive study in recent years.[1] And BNs have frequently been employed to model users (see [Jameson, 1996] for a survey of this work up through 1995). But so far there have been only a few steps in the direction of combining these two lines of research.

In several systems, the conditional probability tables (CPTs) of a BN have been computed more or less straightforwardly on the basis of empirical data (see, e.g., [de Rosis *et al.*, 1992; Albrecht *et al.*, 1998; Lau and Horvitz, 1999]). Although these efforts have had to deal with some tricky issues (e.g., the potentially huge size of the CPTs of [Albrecht *et al.*, 1998]), they have not involved the application of the sophisticated BN learning techniques that are discussed in the overviews cited above.

To see this work in the context of the present workshop, note that a BN for user modeling almost always represents general relationships among variables, which are assumed to apply to all users that the system is likely to encounter. The learning of BNs therefore differs in an important way from most other applications of machine learning techniques to user modeling: Normally, a machine-learning-based user modeling system includes a *learning component* that processes a large amount of data about an individual user. The system's *performance component* then uses the results from the learning component to make decision-relevant predictions about that user in a particular situation.

In the approach pursued here, the data-hungry learning component processes data from a sample of users to learn a BN that applies to users in general. It is only the performance component (i.e., the learned BN) that requires data about the individual user for whom predictions are to be made. A Bayesian network is in principle capable of deriving useful (though uncertain) predictions on the basis of a very small number of observations. Consequently, this approach can be used in situations in which only very limited data about the individual user are available. An example would be a situation in which an assistance system offers help to a user who is using the system for the first (and perhaps only) time.

In the long run, the contrast just discussed may become less sharp: It may prove worthwhile in some situations to learn a different BN for each individual user. But for the initial steps at learning BNs for user modeling, it seems best to focus on user-independent networks—both because of the advantage just mentioned and because there already exist many well-understood examples of such networks.

1.2 Focus on Networks With Hidden Variables and Known Structure

Learning techniques for Bayesian networks address one or both of two basic questions:

1. What structure is the network supposed to have—i.e., what variables are to be represented, and what causal links exist among them?
2. How can the causal relationships be quantified in terms of CPTs?

*This research is being supported by the German Science Foundation (DFG) in its Collaborative Research Center on Resource-Adaptive Cognitive Processes, SFB 378, Project B2, READY. The experiment described in Section 2 was designed and conducted in collaboration with Leonie March and Ralf Rummer of the Department of Psychology, University of Saarbrücken. We thank the reviewers for their helpful suggestions.

[1] The many overviews include those of Buntine [1996], Heckerman [1995], and Russell and Norvig [1995, chap. 5].

Some of the techniques presuppose that all of the variables in the network are *observable*, i.e., that information about their values is available when the network is learned. As we will argue below, in the context of user modeling there are several advantages to the introduction of *hidden* (unobservable) variables (e.g., variables that refer to the user's level of knowledge, specific beliefs, general interests and/or current goals): Their inclusion can lead to more parsimonious and interpretable networks, although it also entails a number of technical problems.

We consider here only the case where the structure of the to-be-learned network is specified in advance, so that only the CPTs have to be learned. One reason is that this situation is less complex than situations in which some aspects of the structure of the network are learned as well (see, e.g., [Cooper and Herskovits, 1992]; [Heckerman, 1995]). A second reason is that specifying the structure in advance is one way of enhancing the interpretability of the resulting networks.

1.3 Issues

The job of learning BNs for user modeling raises a number of issues that appear to be more important in this particular context than for BN learning in general:

1. How can methods borrowed from experimental psychology be applied so as to yield suitable input for BN learning techniques? When we are learning about human cognitive processes, it makes sense to leverage the rich methodology of experimental psychology when collecting the data that are to serve as input to the learning techniques. The same methods of controlling and manipulating variables that enhance the interpretability of experimental data can similarly make the learning of BNs more efficient.

2. How can the large individual differences that typically exist between users (and between experimental subjects) be taken into account, so that the learned network will be applicable to users other than those who supplied the data for the learning?

3. How can we ensure that the learned networks are interpretable? Especially when hidden variables are involved, a learning technique may come up with a solution that yields satisfactory results but which doesn't fit with any theoretical interpretation of the variables involved. But theoretical interpretability is desirable in that it allows us to relate the learned networks to large amounts of relevant existing knowledge, such as that from psychological and educational research.

4. How can we learn partial networks and then combine the results? When dealing with most realistic scenarios in user modeling, it will be impossible to gather empirical data for entire networks at once. In particular, psychological experiments are typically designed to investigate only a few variables at a time. Therefore, an interesting problem is that of learning partial networks and putting these parts together into a single BN. Similarly, it may happen that previous research has already yielded adequate information concerning one part of a network. In that case, it may be possible to restrict learning to the rest of the network and subsequently to combine the parts.

5. How can the learning techniques be adapted so that they work with acceptable efficiency? Since techniques for learning networks with hidden variables involve iterative search through a space with very high dimensionality and with many local maxima, they can raise serious problems of computational complexity that can threaten their applicability, even if they only need to be applied off-line. Some approaches to solving these problems are not specific to user modeling, but others are connected with the above-mentioned issues of ensuring interpretability and combinability of networks.

Since the work described here is still in progress, we will not offer firm answers to all of these questions here. Instead, we will discuss the results achieved so far and the ideas that we are currently pursuing.

1.4 Example Domain

Although the issues we are addressing are not specific to any particular application domain, for concreteness we are using a particular application scenario, that of the system READY (see, e.g., [Jameson *et al.*, 1999]). In this domain, the focus of the user modeling is on the user's situationally determined *resource constraints*—specifically, limitations of time and working memory (WM). The key hidden variables in READY's dynamic BNs refer to these limitations. Until now, the entries of the conditional probability tables (CPTs) of these BNs have been based on a combination of (a) analysis of relevant previous experiments, (b) our own largely qualitative empirical studies, and (c) theoretical considerations. Our goal in applying BN learning techniques is not simply to replace these other sources of knowledge but to combine them with a more direct use of empirical data.

In the full READY system, the BNs are rather complex dynamic networks that include dozens of variables. To get a grip on the problems of BN learning in this context, we began with the data of an experiment that focused on only a few of these variables.

2 Summary of Experiment and Results

It only makes sense for a system $\mathcal{S}$ to try to adapt to the working memory load of a user $\mathcal{U}$ if different system actions can be most appropriate given different levels of working memory load. For example, intuitively one would expect that when $\mathcal{U}$ is suffering from high WM load, $\mathcal{S}$'s instructions should be designed to be simpler and easier to execute, even if this simplicity leads to a longer execution time for the instructions.

2.1 Method

We investigated this hypothesis using the experimental task shown in Figure 1. In the middle and bottom of the screen we see the buttons used for the subjects' primary task, whereas at the top there is a "lamp" for the secondary task.

In the primary task the subject had to click on 2 to 4 buttons, according to (prerecorded) spoken instructions of the system (e.g. "Set *X* to 3 ..."). A similar situation could arise in a real application if $\mathcal{S}$, for example, offered assistance via a telephone hot-line. In this case, the buttons on the screen would contain meaningful labels; abstract labels were used in the experiment to eliminate any effects of differences in subjects' prior knowledge.

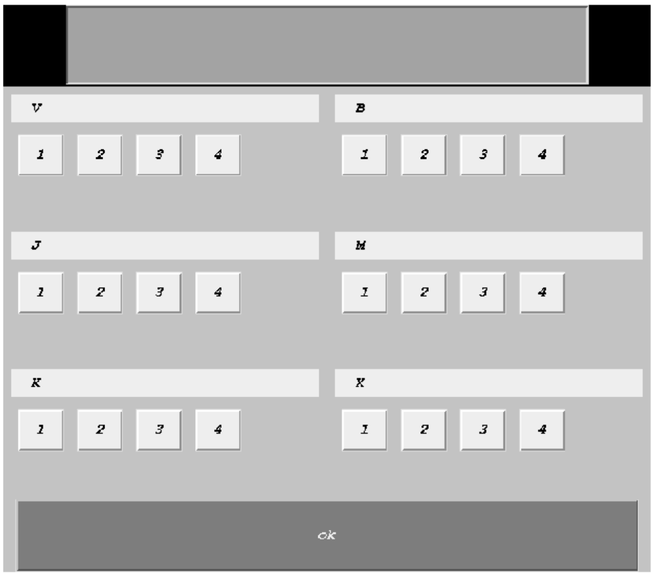

Figure 1. *Main screen used for the experiment described in Section 2.*

Stepwise:	*Bundled:*
$\mathcal{S}$: Set *X* to *3*.	$\mathcal{S}$: Set *X* to *3*, set *M* to *1*, set *V* to *4*.
$\mathcal{U}$: ... [OK]	$\mathcal{U}$: ... ... [OK]
$\mathcal{S}$: Set *M* to *1*.	
$\mathcal{U}$: ... [OK]	
$\mathcal{S}$: Set *V* to *4*.	
$\mathcal{U}$: ... [OK]	

Figure 2. *Illustration of the two presentation modes for instructions.*

The secondary task was performed concurrently with the primary task: The "lamp" flashed intermittently; whenever two successive flashes had the same color, the subject was to press the space bar. A comparable distraction in a realistic situation might involve, for example, the need to monitor some process running on the computer or to communicate with another person.

Three independent variables were manipulated:

1. *Number of steps in each task:* In each task, either 2, 3, or 4 settings had to be made in succession.

2. *Presentation mode:* As is illustrated in Figure 2, in the *stepwise* mode, the instruction for each step was spoken separately; the next instruction was given only after the subject had executed the instruction and clicked on the "OK" button. In the *bundled* mode, the instructions for all 2 to 4 steps of a given task were spoken without interruption; only after completing the last step did the subject click on the "OK" button.

3. *Presence of secondary task:* The secondary task only had to be performed in half of the blocks of trials; in the other half, the lamp did not change color.

Two dependent variables will be considered here:

1. *Error in the primary task:* This binary variable has the value "True" when the subject pressed all of the right buttons for a given task.

2. *Execution time for the primary task:* This is the time between the moment the system began speaking the instructions for a task and the moment the subject pressed the "OK" button to signal completion of the task.

2.2 Summary of Results

A traditional analysis of variance revealed the following significant main effects:

1. A larger number of steps per task leads to longer times (obviously) and more errors.

2. The presence of a distracting secondary task likewise increases both execution time and errors.

3. The stepwise presentation mode leads to longer execution times (mainly because of the additional overhead involved in having to confirm the completion of each step), but it also reduces the number of errors. A plausible explanation for the latter result is that the subject needs to store less information in working memory and so runs less risk of forgetting an instruction.

More interesting than these main effects is the significant interaction between presentation mode and secondary task: The increase in errors associated with bundled presentation is much greater when there is a secondary task; that is, without a secondary task subjects are able to manage the more demanding bundled presentation quite well.

Roughly speaking, the practical implication is that a system should tend to give stepwise instructions when $\mathcal{U}$ is performing a secondary task (to avoid an excessive number of errors) but should tend to give bundled instructions otherwise (to save time). The choice that should be made in each individual case depends on the relative importance of speed and accuracy, among other factors.

A traditional analysis of variance exposes the major causal relationships and helps to assess their generality. But it does not directly yield a model that would allow a system to make decisions or to interpret a user's behavior. These goals can be attained when learning techniques for Bayesian networks are applied to the same data.

3 Learning a Bayesian Network Without Hidden Variables

In a first, straightforward effort, we constructed a BN without hidden variables (see Figure 3). The independent variables of the experiment are represented by the nodes NUMBER OF STEPS, PRESENTATION MODE and SECONDARY TASK. Two of the other three nodes represent the dependent variables of the experiment. Because of the large interindividual variation in execution times (represented by the node EXECUTION TIME) we introduced a variable that represents a user's average task execution time (INDIV AVE EXECUTION TIME), which in turn reflects their overall speed of processing. Before learning took place, the values of this variable were computed for each subject straightforwardly on the basis of the raw data. Consequently, this variable served as an observable variable for the purpose of learning, although it would not be observable in an application context.

Generally speaking, one advantage of arranging for learning to take place in an experimental context is that some variables which are (partially) hidden in an application context

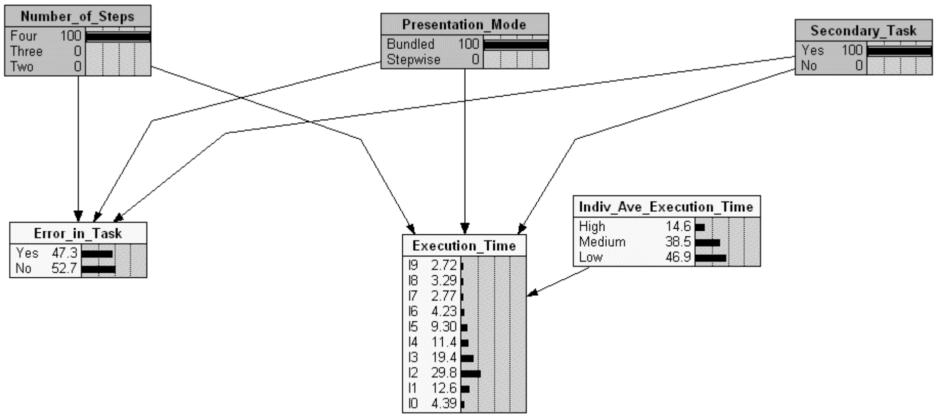

Figure 3. *Example instantiation of a Bayesian network without hidden variables learned on the basis of the experimental data. (The three nodes with a darker gray background have been instantiated.)*

can be made observable. Once the CPTs have been learned, the resulting network can be used in situations in which these variables are not observable. For example, while modeling a particular user $\mathcal{U}$ in an application situation, $\mathcal{S}$ will know that $\mathcal{U}$ has some particular INDIV AVE EXECUTION TIME, but $\mathcal{S}$ will initially have only a highly uncertain estimate of the value of this variable. This estimate will be updated in the course of the interaction.

Note that, if this variable were not included in the network structure, the learning algorithm would not be able to take into account individual differences between subjects. By supplying the subject's base-rate speed as part of each observation, we are helping the algorithm to explain the observed variation in the execution times. Also, by explicitly modeling individual differences, the learned network will be better able to deal with unknown users, whether it knows their base-rate speed in advance or has to learn it in the course of the interaction.

In this initial study, we used the NETICA[2] built-in-algorithm for learning BNs. This algorithm presupposes that no variables are hidden. It computes the (conditional) probabilities on the basis of frequencies in the data.

Not surprisingly, the CPTs learned by this algorithm reflect the same patterns that were uncovered by the analysis of variance summarized above. The average negative log-likelihood describing the fit between the network and the 1728 cases processed is 4.89. Although there is no compact way of displaying all of the learned CPTs, Figure 3 illustrates how the network reproduces one of the important results: the high frequency of errors that is caused by a secondary task when the primary task is complex and the presentation mode is bundled.

[2]NETICA is a commercial tool from Norsys Software Corp. (http://www.norsys.com) for working with Bayesian networks and influence diagrams.

4 Learning a Bayesian Network With Hidden Variables

The network shown in Figure 3 simply reproduces the patterns in the empirical data, without reflecting any sort of theoretical explanation. As was suggested above, a plausible theoretical explanation might refer to limitations in the subjects' ability to store instructions in working memory while performing a secondary task: The combination of bundled presentation and a secondary task tends to overload working memory, leading to a breakdown of performance.

A network that explicitly modeled the hidden variable of WM load could have the following advantages:

1. A hidden variable can make the postulated relationships in the network more interpretable by relating them to prior theoretical and empirical knowledge.

2. A hidden variable can increase the parsimony of the network and facilitate the addition of new variables. There are many other factors that influence WM load and many other symptoms of WM overload. A network that includes a link between each cause and each symptom may be acceptable as long as it is as small as the one shown in Figure 3. But with an increasing number of variables, the number of links would at some point exceed an acceptable limit, from both a theoretical and a practical point of view.

3. A hidden variable can serve as a point at which two independently learned networks can be joined. To take an especially simple case, suppose that each of two independent learning studies has yielded a network that expresses relationships between WORKING MEMORY LOAD and several symptoms of high load (which figure as children in the learned network). It may then be possible to combine these two networks straightforwardly into a single network that includes WORKING MEMORY LOAD together with all of the symptoms. Therefore, hidden variables appear to be an important part of a solution to the problem of how to combine independently learned partial networks.

4. A hidden variable may be decision-relevant in its own right. For example, it may be desirable for $\mathcal{S}$ to avoid high

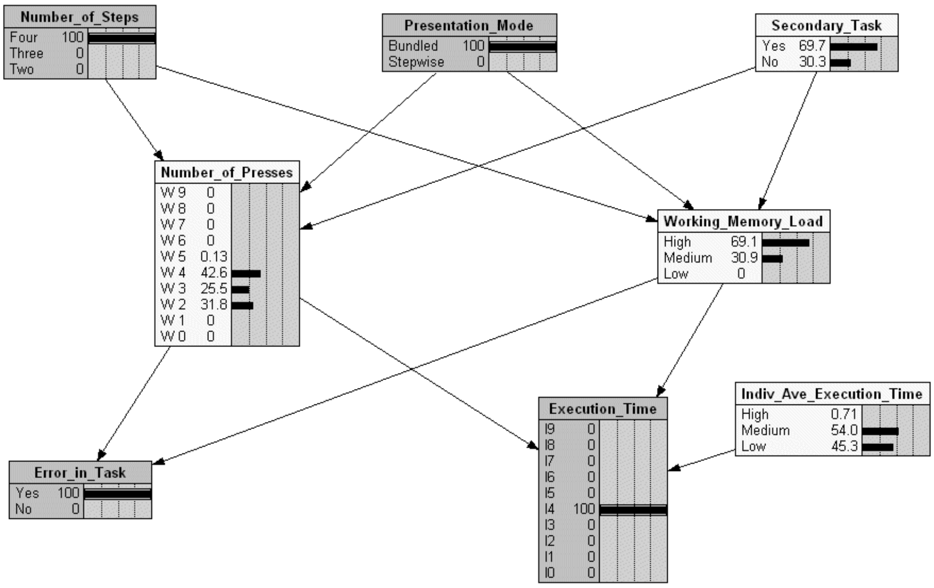

Figure 4. *Example instantiation of a Bayesian network with a hidden variable learned with the EM algorithm on the basis of the same data as the network in Figure 3.*
(The four nodes with a darker gray background have been instantiated.)

WM load in $\mathcal{U}$ simply because high WM load is subjectively unpleasant for $\mathcal{U}$. In that case we will want $\mathcal{S}$'s decision procedure to refer to its belief about $\mathcal{U}$'s WM load. This is not possible if there is no corresponding variable in the network.

To explore the possibilities of learning a network with such a hidden variable, we specified a structure that included the variable WORKING MEMORY LOAD (see Figure 4). It would be simplest if *all* effects of the independent variables on the dependent variables were mediated by WORKING MEMORY LOAD. In reality, there is an independent factor that affects execution time and error frequency: simply the number of actions that $\mathcal{U}$ has to perform. (More actions require more time and give more opportunity to make errors.) This variable, labeled NUMBER OF PRESSES in Figure 4, essentially indexes the number of button presses (on the mouse and the keyboard) that the user has to make.

In the context of our experiment, NUMBER OF PRESSES is an observable variable, since we know exactly what actions the subject was required to perform and what actions in fact were performed. In some application contexts, the corresponding variable might be only partly observable. For example, $\mathcal{U}$ might be performing a secondary task about which $\mathcal{S}$ has no information. Just as with the variable INDIV AVE EXECUTION TIME (see Section 3), it is convenient to be able to treat NUMBER OF PRESSES as an observable variable for the purpose of learning.

4.1 Learning With the EM Algorithm

The algorithm which is apparently most widely used for learning the CPTs of BNs with hidden variables is the EM (*expectation maximization*) algorithm (see, e.g., [Mitchell, 1997]). We implemented this algorithm[3] and had it learn a network with the structure shown in Figure 4. The networks learned with EM are somewhat less accurate than the learned fully observable networks. (The average negative log-likelihood, over 1728 cases, is 5.15, compared with 4.89 for the network discussed Section 3.)

4.2 Learning With the APN Algorithm

For purposes of comparison, we also implemented the Adaptive Probabilistic Networks algorithm developed by Binder *et al.* [1997], which is a gradient descent method for determining a local optimum regarding the CPT entries. The results were roughly similar to those obtained with EM. (The average negative log-likelihood, over 1728 cases, is 5.26, compared with 5.15 for EM.)

4.3 Example Use of a Learned Network

Figure 4 illustrates how the learned network makes plausible inferences when interpreting the behavior of an individual user. Consider the situation where $\mathcal{S}$ is giving instructions to $\mathcal{U}$ but $\mathcal{S}$ does not know whether $\mathcal{U}$ happens to be working on some secondary task at the same time. $\mathcal{S}$ will then want to judge the likelihood of such a secondary task, so as to be able to adapt the form of its instructions. Relevant evidence is available if $\mathcal{S}$ gets feedback on the accuracy and speed with which $\mathcal{U}$ performs the main task. In that case $\mathcal{S}$ can instantiate the variables ERROR IN TASK and EXECUTION TIME. Figure 4 shows $\mathcal{S}$'s assessments after the observation of an error by $\mathcal{U}$.

[3]The implemented EM algorithm uses NETICA's facilities to perform computations on intermediate states of the to-be-learned network. For example NETICA's inference algorithm is heavily used to compute beliefs of network nodes.

Note that $\mathcal{S}$'s assessment of $\mathcal{U}$'s WM load is high and that the likelihood that $\mathcal{U}$ is performing a secondary task is also seen as fairly high.

4.4 Summary of Experience With Both Algorithms

Experience with the EM and APN algorithms showed the importance of exploiting prior knowledge about the quantitative causal relationships among the network variables. The initial CPTs with which the algorithm started on its search strongly influenced the final result, especially with the APN algorithm. This effect is due to the presence of a large number of local optima in the search space. We hope that the problem can be alleviated in part by the use of prior knowledge to restrict the search space: In addition to specifying initial CPTs that correspond to our theoretical interpretation of the hidden variables, we can specify constraints on the CPTs that the learning algorithm will be required to respect.

A somewhat analogous problem is dealt with by Druzdzel and van der Gaag [1995]. In the context of knowledge elicitation for Bayesian networks, they present a method for integrating various types of knowledge about the relevant probabilities. One type of knowledge is the qualitative opinions of experts concerning cases where one variable has a positive or negative influence on another variable. Druzdzel and van der Gaag show how such judgments can be represented in terms of constraints on the corresponding conditional probabilities. We are now exploring the following approach:

1. Constraints for our networks are represented mathematically in similar way.
2. Using the APN algorithm, we add to the negative log-likelihood metric for the evaluation of a network a penalty term that reflects the extent to which the network violates the constraints.
3. The search algorithm should then tend to avoid solutions that violate the constraints. (How hard the constraints are will depend on the relative weight of the penalty term.)

These measures to increase the accuracy of the learned networks should also tend to increase their intepretability, because they should ensure that the learning algorithm will come up with a "theory" that is compatible with our theoretical preconceptions.

5 Concluding Remarks

This paper has discussed a number of results and ideas concerning the issues formulated in Section 1.3. In keeping with the purpose of a workshop, the discussion raises more questions than it answers.

On the whole, we can see that applying BN learning techniques in a user modeling context is not a straightforward technical problem. It requires attention to the characteristic features of (a) data concerning human performance and (b) the contexts in which user modeling components are developed and applied. Doing justice to these features in turn requires a certain amount of technical innovation.

References

[Albrecht *et al.*, 1998] David W. Albrecht, Ingrid Zukerman, and Anne E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8:5–47, 1998.

[Binder *et al.*, 1997] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

[Buntine, 1996] Wray Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8:195–210, 1996.

[Cooper and Herskovits, 1992] Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[de Rosis *et al.*, 1992] Fiorella de Rosis, Sebastiano Pizzutilo, Alesandra Russo, Diane C. Berry, and F. Javier Nicolau Molina. Modeling the user knowledge by belief networks. *User Modeling and User-Adapted Interaction*, 2:367–388, 1992.

[Druzdzel and van der Gaag, 1995] Marek J. Druzdzel and Linda C. van der Gaag. Elicitation of probabilities for belief networks: Combining qualitative and quantitative information. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 141–148. Morgan Kaufmann, San Francisco, 1995.

[Heckerman, 1995] David Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1995. Revised November 1996.

[Jameson *et al.*, 1999] Anthony Jameson, Ralph Schäfer, Thomas Weis, André Berthold, and Thomas Weyrath. Making systems sensitive to the user's time and working memory constraints. In Mark T. Maybury, editor, *IUI99: International Conference on Intelligent User Interfaces*, pages 79–86. ACM, New York, 1999.

[Jameson, 1996] Anthony Jameson. Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interaction*, 5:193–251, 1996.

[Lau and Horvitz, 1999] Tessa Lau and Eric Horvitz. Patterns of search: Analyzing and modeling Web query refinement. In Judy Kay, editor, *User Modeling: Proceedings of the Seventh International Conference, UM99*. Springer Wien New York, Vienna, New York, 1999.

[Mitchell, 1997] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, Boston, 1997.

[Russell and Norvig, 1995] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.