

Lying on the Web: Implications for Expert Systems Redesign

Zhengrui Jiang, Vijay S. Mookerjee, Sumit Sarkar

School of Management, University of Texas at Dallas, Richardson, Texas 75083-0688
{zjxj011000@utdallas.edu, vijaym@utdallas.edu, sumit@utdallas.edu}

We consider a new variety of sequential information gathering problems that are applicable for Web-based applications in which data provided as input may be distorted by the system user, such as an applicant for a credit card. We propose two methods to compensate for input distortion. The first method, termed *knowledge base modification*, considers redesigning the knowledge base of an expert system to best account for distortion in the input provided by the user. The second method, termed *input modification*, modifies the input directly to account for distortion and uses the modified input in the existing (unmodified) knowledge base of the system. These methods are compared with an approach where input noise is ignored. Experimental results indicate that both types of modification substantially improve the accuracy of recommendations, with knowledge base modification outperforming input modification in most cases. Knowledge base modification is, however, more computationally intensive than input modification. Therefore, when computational resources are adequate, the knowledge base modification approach is preferred; when such resources are very limited, input modification may be the only viable alternative.

Key words: sequential information gathering; expert systems; input distortion; noise handling

History: Salvatore March, Senior Editor; H. R. Rao, Associate Editor. This paper was received on February 13, 2004, and was with the authors 5 months for 2 revisions.

1. Introduction

Over the last few decades, expert systems have been implemented for a wide range of business applications. These include, among others, financial planning, providing help desk support, recommending products to customers, software debugging, configuring equipment, and the diagnosis of illnesses in the medical domain (Turban and Aronson 2000). These systems provide several benefits such as improved decision making, consistent and reproducible decisions, shorter decision-making times, and the ability to provide expertise in many places at the same time. The variety of applications for which expert systems are used continues to grow as the costs to develop and deploy such systems reduce over time.

During operation, an expert system typically acts as a consultant to the user. The system asks questions to the user, responses to which help the system identify the cause of a problem (or the solution to a problem) by examining its knowledge base. This consultation process is usually interactive in nature, as this allows the system to efficiently and cost-effectively

collect information, i.e., collect items of information pertinent to the current problem instance.

Recently, many expert systems are being deployed over the Web. For instance, many financial institutions, through their websites, allow customers to apply for credit cards (e.g., www.credit-cards-comparison-charts.com, www.creditcardmall.com). Other websites elicit personal user information and preferences that are fed to expert systems that help target potential customers of a product or service in online marketing campaigns.

1.1. Problem Motivation

Expert systems that are deployed over the Web are facing an important challenge in the form of noisy input data supplied to the system during its use. A variety of factors contribute to the presence of noisy input data. The most significant cause is probably the deliberate falsification of input data by Web users. Tapscott (1999) has reported that “at least 4 out of 10 Internet users admit to giving false answers to website questionnaires to *capture a preferred benefit*,”

and those who lie on the Web “routinely give bogus names, incomes, ages, and gender or say they live somewhere they don’t.” Web users also lie to protect their privacy and possible misuse by firms of any personal data they provide (Fox et al. 2000). Another factor that contributes to lying is that there is no face-to-face interaction between the user and the organizations’ agents in an online environment. Thus, there are no visual or nonverbal cues that could potentially help an agent recognize that a user is lying. The likelihood of input data falsification is further increased because of the relative anonymity of the Web user, combined with the fact that users do not need to expend any significant effort to consult the system. Besides intentional distortion of input data, factors such as poor interface design or lack of computing and typing skills could lead to random mistakes in the input data supplied by a user.

Regardless of the cause of noise, incorrect input data can lead to errors in a firm’s decision-making process. In the case of credit card applications, high-risk customers may be wrongly approved while deserving customers may be denied. Similarly, in the case of data collected to support a marketing campaign, a firm may solicit someone who is not interested or miss someone who may have been willing to purchase. In all these situations, it becomes important for the organization to take measures that either reduce noise (e.g., through better interface design, by providing incentives to reduce lying) or compensate for it after it has occurred.

Noise handling methods are needed because reduction measures are not likely to completely eliminate input noise. While warning users of the severe consequences of lying is easy, enforcement is usually not. If the disincentives of lying are not enforced (e.g., because the verification process to establish the correct information is costly), users may eventually begin to ignore them. Furthermore, users may perceive that they benefit more from lying than from the incentive provided by the firm to be truthful. Thus, noise compensation techniques are still necessary to improve the decisions recommended by expert systems, in particular those that operate in a Web environment.

1.2. Previous Research

Depending on how the initial knowledge base is obtained, expert systems can broadly be classified into

inductive systems and deductive systems. For inductive expert systems, decision rules (e.g., a decision tree) are learned from training data. Deductive systems, on the other hand, are usually based on expert-provided decision rules.

Existing literature on handling noise in expert systems has mainly focused on the development of inductive systems to operate in a noisy environment. Within this area, the most popular noise handling technique is decision tree pruning (Quinlan 1986, Mingers 1989, Breslow and Aha 1997). Quinlan (1986) found that pruned decision trees perform better than unpruned decision trees in the presence of input data noise. While pruning is usually applied as a post processing technique (i.e., the tree is pruned after it is induced), other approaches prune the decision tree during its construction; i.e., the induction process itself is modified to cope with noise (Clark and Niblett 1989, Mookerjee et al. 1995). In addition to pruning, a number of fuzzy learning methods have been used to derive fuzzy rules that perform well with noise and/or incomplete training data (Lee 1990, Hong and Chen 2000, Wu et al. 2003). Finally, version space-based learning strategies have also been shown to perform well with noisy and uncertain data sets (Hirsh 1994, Hong and Tsang 1997). Typically, all the above-mentioned techniques work well only when the noise in the training and test data have similar patterns.

In contrast to inductive systems, the rules of a deductive system are provided by experts. Hence, noisy input data does not affect the creation of rules, but is relevant to consider when the system is consulted. When an expert system operates in an offline mode, it may be possible to remove noise in the input data before the data is supplied to the system. Due to these reasons (i.e., noise has no impact on the rules, inputs can usually be cleaned before consulting the system), noise handling in deductive systems has not been an active research area. However, with more expert systems being deployed over the Web, noise resulting from deliberate falsification (or from other random causes) becomes a real obstacle. Even in offline systems, noise may still be an issue if the data cannot be cleaned (or is too costly to clean) before consultation. For example, offline expert systems have been developed for audit planning and audit risk analysis (Graham et al. 1991, Delisio et al.

1993). In these systems, certain inputs may be costly to obtain accurately. For example, accurately estimating the value of inventory assets may require a physical inspection of the material held at various factory sites. Thus, the system may be required to work with inventory figures that are noisy. This study attempts to address the problem of designing deductive expert systems that need to operate with noisy input data.

1.3. Contributions

We propose two methods to cope with noise. The first method, which we term *knowledge base modification* (KM), considers modifying the knowledge base (specifically, a decision tree) to account for distortion in the inputs provided by the user. This method involves a one-time computational effort to modify the knowledge base; however it is very efficient during operation. It is appropriate when the distortions in inputs are relatively stationary (i.e., input noise levels do not change much over time), and the improvement in decision making is significant. The second method—termed *input modification* (IM)—involves a preprocessing step during which the observed inputs are modified to account for distortions. This method involves modifying an observed input to the most likely true value of the input given the observation(s) made by the system. The modified input is then fed into the existing (unmodified) knowledge base. The IM method may sometimes be preferable because the KM method requires frequent knowledge base redesign if the input noise levels fluctuate.

The problem is formulated as one where an organization has available to it (1) a decision tree that leads to the desired outcomes (recommendations) when accurate input data is available, and (2) a set of beliefs about the accuracy of each possible input data item provided by users. In the KM method, a revised decision tree is obtained, which specifies optimal recommendations for all feasible combinations of observed input values. The IM method does not require any modification of the original decision tree. The KM and IM methods are compared along several dimensions with the original decision tree. Experimental results indicate that both types of modification substantially improve the accuracy of recommendations, with knowledge base modification outperforming input modification in most cases.

Input modification, however, requires less computational effort than knowledge modification. Therefore, knowledge modification is the preferred approach when computational resources are not a limiting factor; when such resources are very limited, input modification may be the only viable alternative.

The rest of the paper is organized as follows. Section 2 describes the overall model of noise and assumptions. Section 3 provides details of the two proposed noise handling methods. Section 4 presents results of experiments conducted using the proposed methods. In §5, we consider two variations of the proposed methods and provide overall recommendations concerning the choice of a noise handling method. Section 6 summarizes the contributions of this work and identifies opportunities for future research.

2. Input Noise Model

We illustrate our model of noise using decision rules for a hypothetical credit rating application shown in Table 1 below. We propose this example in the context of credit screening for online credit card applications, where the objective is to decide whether an application should be approved, and, if so, the kind of card (e.g., platinum, gold, or regular) that should be issued to a particular customer. Table 1 shows four inputs that could be required from a user: *Income* (high, medium, low),¹ *Bachelor's Degree* (yes, no), *Employment* (yes, no), and *Bankruptcy* (yes, no). Bankruptcy refers to whether the applicant has ever filed for bankruptcy. For expositional simplicity, we assume in this example that the four inputs are independent. Based on a consultation session, the system recommends a credit risk category: high risk (HR), medium risk (MR), or low risk (LR) that could be used to determine the kind of card to be offered to the customer. The last column of this table shows the joint probability of the inputs in each row of the table. A dash entry in the table represents that the input could be "Y" or "N."

2.1. Estimating Probabilities for True Input States

To account for the presence of noise in the observations, it is necessary to estimate the probability of true

¹ We assume that the inputs in the expert system are discrete. In reality, a continuous value of an input (e.g., age or income) may be collected from the user and mapped to a discrete value before being supplied to the system.

Table 1 Decision Table for the Credit-Granting System Example

Rules	Inputs				Outcome	Joint prob.
	Income	Bachelor's degree	Employment	Bankruptcy		
CR0	H	—	—	—	LR	0.600
CR1	M	Y	Y	—	LR	0.126
CR2	M	—	N	—	MR	0.090
CR3	M	N	Y	—	MR	0.084
CR4	L	—	—	Y	HR	0.020
CR5	L	—	—	N	MR	0.080

input values based on noisy observations. Let **True** refer to the true input vector underlying an observed input vector **Observed**; the observations could refer to all the possible inputs that could be obtained for a given problem, or they could refer to a subset of the possible inputs. We are interested in estimating $P(\mathbf{True} | \mathbf{Observed})$ for all feasible sets of input values, expressed as:

$$P(\mathbf{True} | \mathbf{Observed}) = \frac{P(\mathbf{Observed} | \mathbf{True}) \cdot P(\mathbf{True})}{P(\mathbf{Observed})}. \quad (1)$$

When the number of inputs under consideration is even moderately high, it becomes difficult to obtain reliable estimates for $P(\mathbf{Observed} | \mathbf{True})$ directly as a very large number of probability parameters have to be estimated. For example, if there were n binary inputs, then we would need to estimate probability terms for **True** vectors corresponding to each of the 2^n possible **Observed** vectors. Because there would be 2^n different **True** vectors for which such parameters would need to be estimated for each **Observed** vector, we would need to estimate a total of 2^{2n} different probability parameters to calculate the above expression for all feasible observations. It is clearly impossible for domain experts to provide so many estimates. Even when the estimates are obtained from data, the amount of data required to make reliable estimates could be prohibitively high. To make tractable the number of parameters to be estimated, we make two assumptions:

ASSUMPTION 1. $P(\mathbf{Observed} | \mathbf{True}) = \prod_i P(\text{Observed}_i | \mathbf{True}_i)$, where i is an index over the inputs.

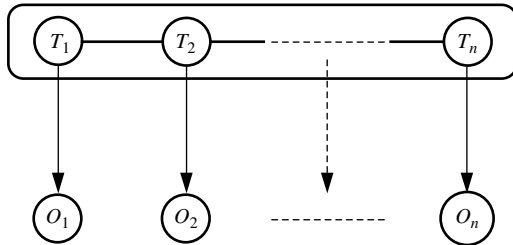
ASSUMPTION 2. $P(\text{Observed}_i | \mathbf{True}) = P(\text{Observed}_i | \mathbf{True}_i)$ for all i .

Assumption 1 implies that each individual input observation is conditionally independent of other input observations given the set of true input values. In other words, the probability associated with each input observation is dependent on the true state of the input vector, and not on other noisy observations. We expect this to be reasonable in most circumstances. For instance, a user may falsify her income to obtain a favorable recommendation, but at the same time she may be honest about her education background if she perceives it to be beneficial for the desired outcome. Assumption 2 can be interpreted to mean that the observed state of an input i is conditionally independent of the true states of other inputs, given the true state of input i . In other words, applicants distort data at the level of the individual input level rather than at the aggregate profile level. It is possible that some applicants are able to foresee the inputs they will be asked to provide in the sequential decision process, and may make the extra cognitive effort of manipulating their response for a current input based on the set of their true states for inputs already provided as well as the inputs they may be asked to provide subsequently. Typically, though, we expect respondents to distort data on each input based primarily on the true value of that input and their expectation of the impact of the distortion on the eventual recommendation by the system.

Figure 1 provides a graphical representation of the dependencies (and conditional independencies) between the observed and true input states. The box circumscribing the T_i s is used to denote the dependencies across all true input states.² While it is possible in some situations that an input state is independent of other inputs, to keep our model fairly general we do not make this assumption. Assumption 1 eliminates the need for arcs connecting the O_i 's (observed states) directly. Assumption 2 eliminates arcs from T_j to O_i for all $j \neq i$. The assumptions taken together imply that observation O_i is conditionally independent of all other observed and true states, given the true state T_i .

² If we were to use the traditional representation for probability networks, we would need to show arcs across all pairs of input states, which would make the figure very cluttered. Instead, we use the box for simplicity.

Figure 1 Graphical Representation of Dependencies Between Observed and True States



The above assumptions are used to simplify the term $P(\mathbf{Observed} | \mathbf{True})$ as follows:

$$P(\mathbf{Observed} | \mathbf{True}) = \prod_i P(\text{Observed}_i | \text{True}_i). \quad (2)$$

Equation 1 can now be rewritten as:

$$P(\mathbf{True} | \mathbf{Observed}) = \frac{\prod_i P(\text{Observed}_i | \text{True}_i) \cdot P(\mathbf{True})}{P(\mathbf{Observed})}. \quad (3)$$

The task of estimating $P(\mathbf{True} | \mathbf{Observed})$, which is essentially one of determining the likelihood of a true vector given the observations (Berger 1985, pp. 27–31), is considerably simplified as the parameters needed in the revised expression are the terms $P(\text{Observed}_i | \text{True}_i)$ for each i , and the term $P(\mathbf{True})$. The terms $P(\text{Observed}_i | \text{True}_i)$ for each i can be obtained either from domain experts, or from sample data collected for that purpose. In contrast to Equation (1), for n binary inputs only $2n$ independent parameters would need to be obtained (two for each input) instead of 2^n . Given the relatively fewer number of parameters needed, reliable estimates may be obtained with

reasonable amounts of data. $P(\mathbf{True})$, the second term in the numerator, is simply the joint probability of a true input vector. This can be estimated from historical data. Based on (2) and the distribution of the true vectors, we can calculate the probability that a vector is observed:

$$P(\mathbf{Observed}) = \sum_r P(\mathbf{Observed} | \mathbf{True}^r) P(\mathbf{True}^r), \quad (4)$$

where r is the index over all possible true vectors.

Figure 2 illustrates the distortion matrices and marginal distributions for each input in the credit rating application. Each entry in a distortion matrix stands for the probability of an observed input state (column) given the true input state (row). To illustrate the process of calculating the left-hand-side quantity in Equation (3), consider an observed input vector ($I^O = "M," D^O = "Y," E^O = "N," B^O = "N"$) corresponding to the observed states of the inputs *Income*, *Bachelor's Degree*, *Employed*, and *Bankruptcy*. To obtain the probability that the true input vector is ($I^T = "H," D^T = "Y," E^T = "Y," B^T = "Y"$), we first use Equation (2) to calculate

$$\begin{aligned} &P(I^O = "M," D^O = "Y," E^O = "N," B^O = "N" | I^T = "H," \\ &\quad D^T = "Y," E^T = "Y," B^T = "Y") \\ &= P(I^O = "M" | I^T = "H") \cdot P(D^O = "Y" | D^T = "Y") \\ &\quad \cdot P(E^O = "N" | E^T = "Y") \cdot P(B^O = "N" | B^T = "Y") \\ &= 0.0038. \end{aligned}$$

Based on the distortion matrices and marginal distributions shown in Figure 2, we have $P(I^T = "H," D^T = "Y," E^T = "Y," B^T = "Y") = 0.054$ and $P(I^O = "M,"$

Figure 2 (a) Input Distortion Matrices; (b) Input Marginal Distributions

(a)	<i>Income (I)</i>	<i>Bachelor's degree (D)</i>	<i>Employed (E)</i>	<i>Bankruptcy (B)</i>
	High Medium Low	Yes No	Yes No	Yes No
High	0.9 0.05 0.05	0.95 0.05	0.9 0.1	0.2 0.8
Medium	0.35 0.6 0.05	0.35 0.65	0.45 0.55	0.01 0.99
Low	0.25 0.35 0.4			

(b)	<i>Income (I)</i>	<i>Bachelor's degree (D)</i>	<i>Employed (E)</i>	<i>Bankruptcy (B)</i>
	High Medium Low	Yes No	Yes No	Yes No
High	0.6	0.6	0.7	0.2
Medium	0.3	0.4	0.3	0.8
Low	0.1			

$O^O = "Y," E^O = "N," B^O = "N") = 0.0389$. From (3), we obtain

$$P(I^T = "H," D^T = "Y," E^T = "Y," B^T = "Y" | I^O = "M," \\ D^O = "Y," E^O = "N," B^O = "N") = 0.0049.$$

The likelihood of all other feasible true input vectors can be calculated analogously.

3. Solution Approaches

As briefly discussed in §1, there are two methods to address noise in input data, the *knowledge base modification* (KM) method and the *input modification* (IM) method. In the KM method, the original knowledge base is modified based on the most likely outcome given each possible observed vector. In the IM method, the observed inputs are used to infer the most likely true values of the inputs that are then fed into the existing (unmodified) knowledge base. In this section, we begin with a detailed description of the two methods, and then illustrate them using the credit rating system example introduced in the previous section. We finally highlight the conceptual differences between KM and IM.

3.1. Knowledge Modification

The inputs to the KM method include the original decision tree (or *True tree* for simplicity), the joint input probability distributions, and the distortion matrices for all variables. The output to the KM method is a modified decision tree that we name a *KM tree*. Before the KM tree can be generated, we need to first compute the fully enumerated *KM table*, which includes the *KM recommendations* for all possible observed vectors. The KM recommendations for a given observed input vector, in turn, depend on the conditional probabilities of all true vectors given the observed vector. The KM method consists of the following five steps.

Step 1. Find the probabilities of all possible true input vectors given an observed vector.

Using (3), we find the conditional probabilities of all true input vectors given an observed vector, denoted by **Observed**. $P(\mathbf{Observed})$ is obtained based on (4).

Step 2. Find the KM recommendation for the given observed vector.

Sum up the conditional probabilities of all true vectors that have the same recommendation. The recommendation with the highest total probability becomes the KM recommendation.

Step 3. Construct the fully enumerated KM table.

This is obtained by iterating Step 1 and Step 2 for all possible observed vectors.

Step 4. Construct the condensed KM table.

The KM table is compressed to create the *condensed KM table*. The compression logic for starting with a fully enumerated decision table to remove redundant input entries has been outlined in prior studies (e.g., Reinwald and Soland 1966, Schwayder 1974).

Step 5. Generate the KM decision tree.

Using the condensed decision table and the probability distribution of observed vectors, the optimum KM tree can be found using AO* search (Mookerjee and Mannino 1997), a variant of branch and bound search. The output of this step is the KM tree.

3.2. Input Modification

In the IM method, the True tree, the marginal distributions of the variables, and the distortion matrices for the variables are needed to generate the *IM recommendations* for observed input vectors. The partial order of the variables acquired is determined by the True tree structure. For each variable acquired, its most likely true input state is chosen based on the following Bayes formula:

$$P(X^T | X^O) = \frac{P(X^O | X^T)P(X^T)}{P(X^O)}, \quad (5)$$

where X^O and X^T represent the observed state and the true state of a variable, respectively. The procedure for the IM method consists of the following steps.

Step 1. Find the most likely true state from the observed value of the variable represented by the root node, and traverse the tree based on this state.

Step 2. If the current node is a decision node, then it represents the IM recommendation for the observed inputs. If not, compute the most likely true state for the current variable, and traverse the tree accordingly.

Step 3. Repeat Step 2 until a decision node is reached.

3.3. Illustration Using the Credit Rating System Example

We illustrate the KM and IM methods using the credit rating system example.

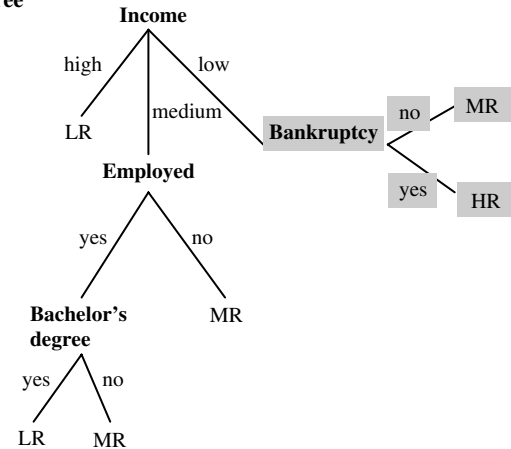
3.3.1. Knowledge Modification. To illustrate how the first two steps work, consider **Observed** = (*Income* = “L,” *Bachelor’s Degree* = “Y,” *Employed* = “Y,” *Bankruptcy* = “N”). First, we compute the conditional probabilities of all true vectors given this observed vector. Second, we add the conditional probabilities of all true input vectors that result in the same outcome. The sum of these probabilities for the low-risk category is 0.470, and the sums for the medium- and high-risk categories are 0.451 and 0.079, respectively. Therefore, we pick LR as the KM recommendation for the observed vector (“L,” “Y,” “Y,” “N”).

In the third step, by repeating Step 1 and Step 2 for all other possible observed vectors, we get the KM table. The fully enumerated KM table is compressed in the fourth step. In the final step, the KM tree is generated from the condensed KM table. For this example, the KM tree is different from the True tree only in one branch. In Figure 3, we show on the right the tree fragment in the KM tree that replaces the shaded fragment in the True tree. It is interesting to observe some similarities and differences between the True tree and the KM tree. Clearly, in the KM tree more information is sometimes needed to come up with a recommendation. For example, when income is observed to be low in the KM tree, additional inputs have to be gathered before a recommendation can be made.

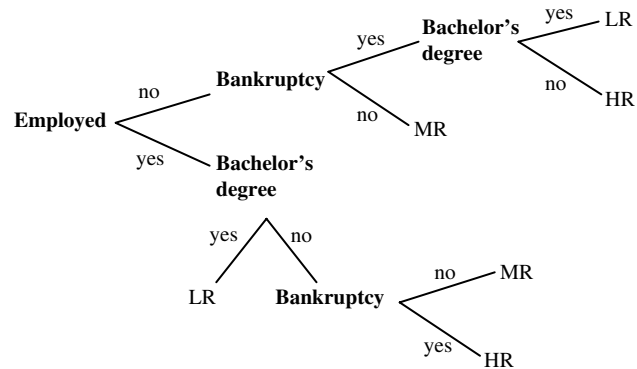
3.3.2. Input Modification. We now illustrate the IM method using the True tree shown in Figure 3(a). Assume that “L” is the observed state for *Income*, which is the root node variable. To find the most likely true state, we calculate the conditional probabilities (based on Equation (5)) and find that $P(\text{Income}^T = \text{“H”} \mid \text{Income}^O = \text{“L”}) = 0.353$, $P(\text{Income}^T = \text{“M”} \mid \text{Income}^O = \text{“L”}) = 0.177$, and $P(\text{Income}^T = \text{“L”} \mid \text{Income}^O = \text{“L”}) = 0.471$. We conclude that the most likely true state is “L” itself. Following the right-most branch marked “low,” the variable *Bankruptcy* is reached. We assume that the observed state for *Bankruptcy* is “N.” To determine the most likely true state for *Bankruptcy*, the following

Figure 3 True Tree and KM Tree for the Credit Granting System

(a) True tree



(b) KM tree fragment replacing the shaded fragment in the true tree



probabilities are calculated: $P(\text{Bankruptcy}^T = \text{“Y”} \mid \text{Bankruptcy}^O = \text{“N”}) = 0.168$ and $P(\text{Bankruptcy}^T = \text{“N”} \mid \text{Bankruptcy}^O = \text{“N”}) = 0.832$. We therefore conclude that the most likely true state for *Bankruptcy* is “N.” Following the “no” branch of this node, a decision node “MR” is reached. Therefore the IM recommendation for the observed vector (“L,” “Y,” “Y,” “N”) is “MR.”

3.4. Conceptual Comparison

IM and KM differ in terms of the constraints imposed on the methods. In IM, we are given a decision tree and are required to traverse the tree in a manner so as to compensate for noise in the inputs. Therefore, the partial order described by the given decision tree must be preserved by the IM method. In KM, there is no constraint imposed on the decision

tree produced by the method. In IM a noisy input (in the extreme case, even one with no information value) may get acquired if this input appears in a path that is being traversed in the original decision tree. The KM method, on the other hand, would eliminate inputs that were noisy in favor of others that were less affected by noise. IM would be especially disadvantaged if, for instance, an input with considerable noise appeared at the root of the decision tree. Because such an input would likely provide very little discrimination, and because new inputs cannot be added to the tree, the accuracy of decisions recommended by the IM decision tree would likely suffer.

The IM method, however, can be beneficial in some situations. If the noise level is relatively low, it may not affect the decision of whether or not a certain input should be gathered. Then, to account for noise, the only adjustment needed is to map the observed state of an input to the most likely true state and use the original decision tree as if the input was observed in this state. Another situation in which the use of IM may be appropriate is one in which the level of input noise changes frequently. In KM, the entire decision tree would need to be recalculated if the noise level changed; in IM only the strategy to traverse the original decision tree would need to be changed. We show in §5.4 that the computational overhead for IM is significantly smaller than that for KM.

4. Experimental Study

In this section, we report the results of a series of numerical experiments conducted to compare the performance of KM and IM versus that of the True tree. Some analytical findings concerning these methods are also presented here. The performance of a method is primarily measured in terms of its accuracy, i.e., the percentage of correct recommendations made in the consultation phase. We conduct a variety of experiments based on simulated as well as real-world data. We first discuss the main factors that affect performance, followed by the experimental design, the experimental procedures, and finally the discussion of experimental results.

4.1. Factors and Hypotheses

We consider three factors that might affect the performance of the proposed methods: algorithm, noise,

and dependency. Each factor and its anticipated effects are discussed below.

4.1.1. Algorithm. We are interested in the difference in performance of the two proposed methods (KM and IM). To provide a baseline measure of performance, we report the performance of the True tree, i.e., the original decision tree with no accommodation made for input noise. We expect KM to outperform IM over a wide range of situations because KM does not need to satisfy the constraint of preserving the partial order of the True tree. We also expect IM to perform better than the True tree because IM makes some effort to address noise in the input while the True tree does not. The following two hypotheses summarize our expectations about the performance of the three algorithms.

HYPOTHESIS 1. *KM outperforms IM over a wide range of factors.*

HYPOTHESIS 2. *IM outperforms True tree over a wide range of factors.*

4.1.2. Noise. The noise (or *distortion level*) in an input variable is defined as the probability of its observed state being different from its true state. Noise in an input can be computed directly from the input's *distortion matrix* and the marginal distribution of the input. For example, if a ternary variable has the distortion matrix and marginal distribution shown in Figure 4, then the distortion level for this variable, denoted by α , equals

$$\sum_i P(\text{True}_i)(1 - P(\text{Observed}_i | \text{True}_i)) \\ = P(H)(1 - P_{11}) + P(M)(1 - P_{22}) + P(L)(1 - P_{33}). \quad (6)$$

In our experiments, the first $(n - 1)$ diagonal elements for an $(n \times n)$ distortion matrix are randomly generated. The last one is calculated based on the distortion level α and the marginal distribution of the

Figure 4 Distortion Matrix and Marginal Distribution of a Ternary Variable

		(Observed)			Marginal probability
		H	M	L	
(True)	H	P_{11}	P_{12}	P_{13}	$P(H)$
	M	P_{21}	P_{22}	P_{23}	$P(M)$
	L	P_{31}	P_{32}	P_{33}	$P(L)$

variable. For a nonbinary variable, all nondiagonal elements in the i th row equal $(1 - P_{ii})/(n - 1)$.

With respect to the effect of the level of noise on the accuracy of the methods, we have two analytical findings (please refer to Appendices A and B for proofs):

PROPOSITION 1. *For any expert system, there exists a distortion level α_0 , such that if the distortion level for all variables is less than α_0 , then given an observed vector, the KM method and IM method will always provide the same recommendation as the one given by the True tree.*

PROPOSITION 2. *If all elements in the same column of the distortion matrix of an input variable are equal, then (a) the variable will not appear in the KM tree; and (b) if the variable appears in some branches of the True decision tree, the final recommendation made by the IM method will not be affected by the observed state of the variable.*

The above two propositions capture the behavior of the two methods at two extremes. Proposition 1 considers the case where noise is sufficiently low for all variables. From this proposition it is clear that if the noise in all input variables is sufficiently low, neither KM nor IM is needed. The second proposition deals with the other extreme: noise in a particular variable is so high that no information can be inferred from its observed value. It can be shown that if the distortion matrix of a variable satisfies the “if condition” described in Proposition 2, then the posterior distribution of a variable given its observed state is the same as its prior distribution. For this case, the KM method is more efficient than the IM method because the KM method drops such a variable from the KM tree, while the IM method acquires the input if it is in the True tree.

Because the above propositions only consider extreme values of noise (very low and very high), we need to study the impact of noise over a wide range of values. For simplicity and ease of exposition, the experiments reported in this section use the same distortion level (α) for all inputs. As discussed in §3.4, the KM approach is better able to adapt to noise than the IM approach because the partial order of the True tree does not need to be preserved by the KM approach. Therefore, we expect the performance difference between KM and IM to increase as noise increases. We also expect IM to perform better than

the True tree because the True tree does not account for noise at all. Our expectations concerning noise are formulated in Hypotheses 3 and 4 below.

HYPOTHESIS 3. *The performance of IM relative to KM deteriorates with noise.*

HYPOTHESIS 4. *The performance of the True tree relative to KM and IM deteriorates with noise.*

4.1.3. Dependency. The dependency across input variables may also affect the performance of the proposed methods. This is because with higher dependency across the inputs, it may be possible to use the states of other inputs to infer the state of a noisy input. We measure dependency by the *mutual information* (MI) across the input variables. For example, if \bar{V} is a vector of five variables V_0, V_1, \dots, V_4 , then the mutual information across the input variables is computed as follows:

$$\text{MI}(V_0, V_1, \dots, V_4) = \sum_{\bar{V}} P(\bar{V}) \ln \frac{P(\bar{V})}{\prod_{i=0}^4 P(V_i)}. \quad (7)$$

Dependency can be expected to improve the performance of KM relative to IM. With higher dependency, KM can collect additional inputs as corroborative evidence to compensate for the noise in a given input; e.g., if income is noisy, then education and/or homeownership can be additionally acquired to compensate for the noise in the income variable. Because IM cannot acquire any inputs outside of those in the True tree, the performance gap between KM and IM should increase as dependency increases. When comparing the performance difference between IM and the True tree, because both methods are restricted to the same set of inputs, we do not expect that dependency across the inputs will affect the relative performance of the two methods.

HYPOTHESIS 5. *The performance difference of IM relative to KM deteriorates with an increase in variable dependency.*

HYPOTHESIS 6. *The performance of IM relative to the True tree is not affected by variable dependency.*

4.2. Experimental Design

The polynomial regression model shown in (8) is used. Here, we are primarily interested in examining

what factors affect the performance difference, rather than predicting the magnitude of the difference.

$$\begin{aligned} & \begin{Bmatrix} E(\Delta ACC_{KM-TRUE}) \\ E(\Delta ACC_{KM-IM}) \\ E(\Delta ACC_{IM-TRUE}) \end{Bmatrix} \\ & = \beta_0 + \beta_1 Distortion + \beta_2 MI + \beta_3 Distortion \cdot MI, \quad (8) \end{aligned}$$

where

$\Delta ACC_{KM-TRUE}$ is the performance difference between KM and the True tree,

ΔACC_{KM-IM} is the performance difference between KM and IM, and

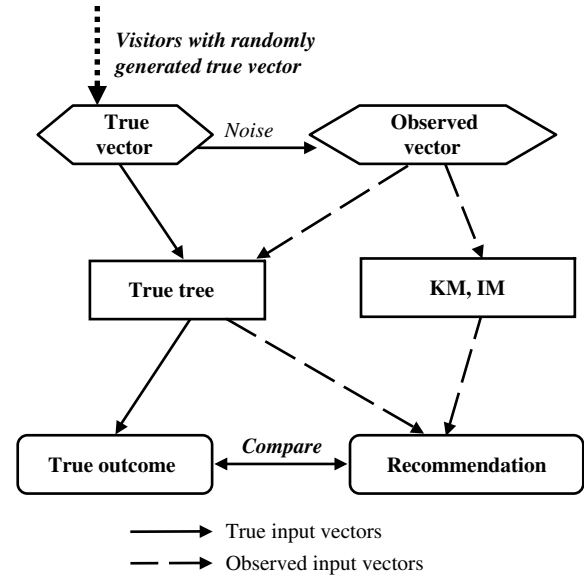
$\Delta ACC_{IM-TRUE}$ is the performance difference between IM and the True tree.

In the experiments, all the simulated knowledge bases contain five binary input variables and three possible outcomes. The MI across these variables is controlled by choosing appropriate conditional probabilities (probability of an input variable state given another input variable state) across the input variables. The *Distortion* level is randomly chosen to be within $[0, 0.5]$ and is equal for all inputs. Note that because we have used binary variables, a distortion level of 0 implies no noise whereas a distortion level of 0.5 represents maximum noise. The response variables in (8) are computed directly from the accuracy of the different methods, which can be obtained by following the experimental procedures described below.

4.3. Experimental Procedure

The overall procedure is as follows. The program first generates a True decision tree by following these three steps: (1) a *True decision table* is generated by randomly choosing one of the three possible outcomes for all possible combinations of input vectors; (2) the distributions for the input vectors are generated by randomly setting some marginal distributions and some conditional probabilities (which determine the dependency across variables); (3) the True decision tree is constructed from the True decision table. After the original knowledge base is obtained, a distortion level is randomly chosen from the interval $[0, 0.5]$. The KM tree is then constructed by following the procedure given in §3. Finally, the accuracy of the different methods is determined by processing a large number of randomly generated test cases.

Figure 5 Test Case Generation and Processing



The procedures for test case generation and processing are shown in Figure 5. For each simulated online visitor, we first randomly generate the true input vector. The *true outcome* for the visitor is determined by feeding the true vector into the True tree. The observed vector for this visitor is next obtained by distorting the true vector according to the chosen level of distortion. The KM, IM, and True tree recommendations are then found. The process is repeated 100,000 times to simulate 100,000 online visitors. The *accuracy* of a method is the percentage of correct recommendations made by the method.

To generate sufficient observations for the regression analysis, we randomly generate 100 different True trees. For each True tree, 10 different levels of distortion are used. The final data set therefore contains 1,000 rows, and each row includes the values of *Distortion*, MI, and the accuracy of the three methods.

4.4. Experimental Results

The following two *t*-tests were conducted to test Hypothesis 1 and Hypothesis 2:

$$\begin{cases} H_0: \Delta ACC_{KM-IM} = 0, \\ H_1: \Delta ACC_{KM-IM} > 0, \end{cases} \quad (T1)$$

$$\begin{cases} H_0: \Delta ACC_{IM-TRUE} = 0, \\ H_1: \Delta ACC_{IM-TRUE} > 0. \end{cases} \quad (T2)$$

Table 2 Actual Classifications of the Methods

True outcome	Classification		
	A	B	C
A	61.6% (KM)	21.5% (KM)	16.9% (KM)
	57.8% (IM)	22.9% (IM)	19.2% (IM)
	54.3% (True)	23.3% (True)	22.4% (True)
B	14.4% (KM)	68.6% (KM)	17.0% (KM)
	17.4% (IM)	65.1% (IM)	17.5% (IM)
	21.8% (True)	56.1% (True)	22.1% (True)
C	16.7% (KM)	18.1% (KM)	65.2% (KM)
	18.3% (IM)	22.2% (IM)	59.4% (IM)
	21.1% (True)	23.5% (True)	55.4% (True)

In both tests, H_0 is strongly rejected with a p -value less than 0.001. We therefore conclude that KM performs better than IM, and IM, in turn, performs better than the True tree. In Table 2, we report the actual classification provided by the three methods. The first row of this table shows that for test cases with a true class of “A,” KM correctly classified 61.6% of them as class “A,” and incorrectly classified 21.5% and 16.9% of them as class “B” and class “C,” respectively. Thus KM is not only the most accurate, but has the lowest misclassification rate within each misclassification category. IM dominates the True tree both in terms of overall accuracy and misclassifications. The experiment indicates that KM is a better classifier than IM, which is a better classifier than the True tree.

To examine the impacts of noise and dependency on the relative performance of the different methods, we ran three separate step-wise regressions based on model (8). The fitted response functions are listed in the first row of Table 3. The significance level was set to 0.10 in the regressions; i.e., only those variables that were found to be significant with a p -value of 0.10 or below were kept in the response functions. From the results, we conclude that both noise, dependency, and their interaction significantly affect the performance of the three methods.

We also examined the isolated effect of a factor (MI or *Distortion*) on the performance differences. To do this, we separately regressed the accuracy difference between pairs of methods on each factor (i.e., ignoring the other factors and interaction effects). The regression coefficients and the associated t -values are reported in the last two rows of Table 3. We find that except for the coefficient in the right-bottom cell of Table 3, all other coefficients are positive and statistically significant.

From the regression results, we find that Hypotheses 3 and 4 are supported. Thus the relative advantage of KM over IM (IM over True tree) is higher at higher levels of input distortion. Hypotheses 5 and 6 are also supported. This confirms our expectation that KM should benefit from variable dependency, while IM is not benefited from variable dependency.

4.5. Graphical Results

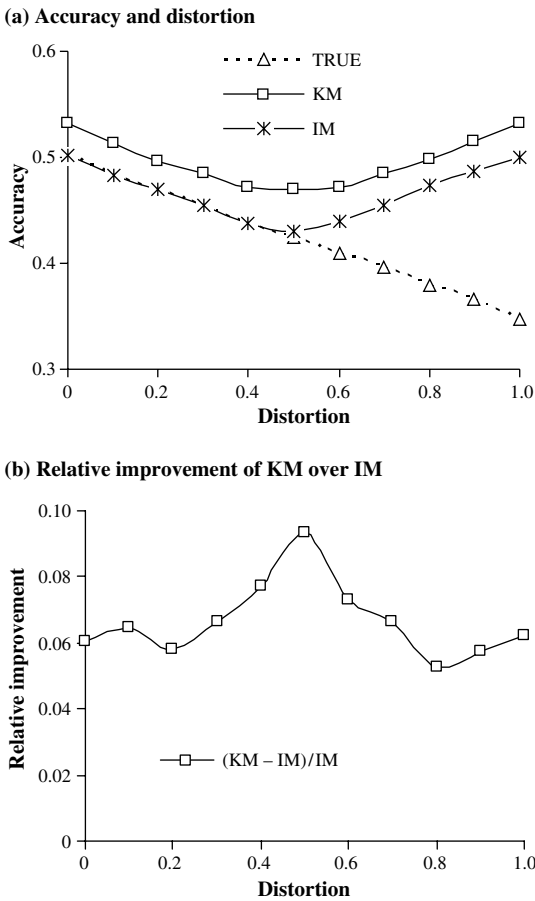
We conducted a number of controlled experiments to graphically depict the nature of impact of the various factors on performance. In these experiments, a factor of interest was varied while the others were kept constant.

4.5.1. Input Distortion. Figure 6 displays the impact of input distortion on the performance of the various methods. In this experiment, the distortion of one variable was varied from 0 to 1 in steps of 0.1, while the distortions of all other variables were kept at 0.2. The symmetric nature of the KM and IM curves (Figure 6(a)) is expected. As the level of distortion increases beyond 0.5 (note all variables are binary), the impact of noise diminishes because the amount of information contained in the input increases. Hence, the accuracy of KM and IM is the lowest at a distortion level of 0.5 and the accuracy is symmetric around this level of distortion. Note, however, that the accuracy of the True tree continues to degrade beyond the 0.5 distortion level because the True tree does not

Table 3 Impact of Noise and Dependency on the Three Methods (t Value in Parenthesis)

	$\Delta ACC_{KM-TRUE}$	ΔACC_{KM-IM}	$\Delta ACC_{IM-TRUE}$
Response functions	$0.00188 + 0.0339MI + 0.309Dist + 0.2004MI \cdot Dist$	$-0.0014 + 0.0916MI + 0.0676Dist - 0.068MI \cdot Dist$	$0.00328 - 0.0577MI + 0.241Dist + 0.268MI \cdot Dist$
$\partial \Delta ACC / \partial Dist$	0.38 (24.98)	0.034 (3.16)	0.35 (21.12)
$\partial \Delta ACC / \partial MI$	0.072 (6.95)	0.074 (13.83)	-0.0022 (0.20)

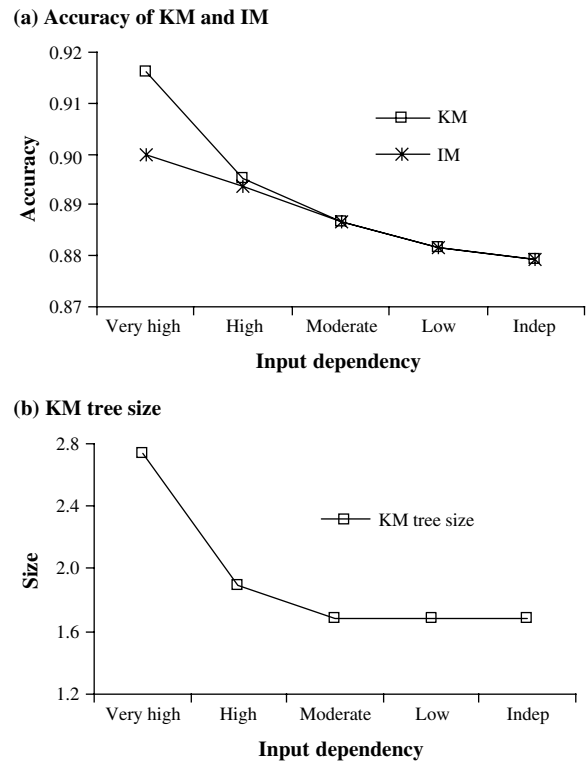
Figure 6 Impact of Distortion on Accuracy



account for noise in the inputs. In Figure 6(b) we show the relative improvement in accuracy achieved as a result of using KM over IM. As can be seen from this figure, the percent improvement tends to increase as the distortion changes from 0 to 0.5 and decrease afterward.

4.5.2. Input Dependency. To study input dependency in a realistic context, we use a modified version of the credit rating system example in §3. Within the four input variables, we assume that both *Employment* and *Income* depend on *Bachelor's Degree*, while *Bankruptcy* depends on *Income*. Using this context, five dependency levels (*very high*, *high*, *moderate*, *low*, and *zero dependency* or *independent*) are created. The distortion levels for all variables are kept at 0.1. The results are plotted in Figure 7. Figure 7(a) shows that as input dependency increases, the accuracy of the

Figure 7 Impact of Input Dependency on Accuracy and KM Tree Size



KM and IM methods increases and their performance gap widens. From Figure 7(b), we observe that the KM tree size increases at high and very high levels of input dependency. These patterns can be explained by the fact the KM method is able to collect additional inputs to compensate for noise and therefore better able to adapt to a noisy environment. However, this ability is not very useful when the inputs are not correlated, i.e., observing corroborative inputs is only useful when the inputs are dependent.

4.6. Testing with Real-World Data

To test the robustness of the methods, we conducted a series of experiments using a credit card application approval data set taken from the University of California, Irvine, machine learning repository (<http://www.ics.uci.edu/~mllearn/MLRepository.html>). The data has 2 classes, 15 variables, and 690 instances. The instances were treated as decision rules in our experiments. To make the data appropriate for the proposed methods, we preprocessed the original data as follows. First, we removed the duplicate and conflicting

instances as well as instances with missing values (490 instances remained after preprocessing). Second, we converted continuous attributes to binary attributes based on the interval split provided by the decision tree induction algorithm C4.5. Third, because the proposed methods require a complete True decision tree or fully enumerated decision table, we first built a decision tree using C4.5 and used this tree to determine the classes for those input vectors that were not in the data set. Finally, the joint distribution of input vectors was computed as the product of the marginal distributions of all input variables that were obtained from the real data.

After the fully enumerated True decision table was obtained, we varied the distortion level of one variable from 0 to 1 in steps of 0.1, and fixed the distortion level for all other variables at 0.1. The impact of distortion on accuracy is shown in Figure 8. From

the figure, we find that KM always performs better than IM, and IM, in turn, generally outperforms the True tree. Statistical tests confirmed that both performance differences are significant with a p -value of less than 0.01. Also, the advantage of KM over IM is generally higher when the distortion is around 0.5 than when it is near 0 or 1. These conclusions are consistent with our findings using simulated data.

4.7. Dependent Noise

So far, we have assumed that the observed state of each input is conditionally independent of the observed state of other inputs given the true states of the inputs. Under this assumption, the probability that a user lies about one input is independent of whether she lies about other inputs. To test the robustness of the methods in situations when the above assumption is violated, we conducted a series of experiments using the credit approval data set. The KM and IM methods were used assuming that the distortions in the input variables were independent. However, in the testing phase, the distortions of the 5 of the 15 variables were made to be dependent. Specifically, if a user lied about the first of these five input variables, then she is more likely to lie about the other four input variables. We define R , the *distortion dependence ratio*, as α_L/α_T , where the distortion level is α_L for the four inputs when the user lied on the first input, and α_T otherwise. The weighted average of α_L and α_T was used in the design phase for KM and IM. We vary R from 1 to 10 in the experiments. Figure 9 shows the accuracy of the three methods when the

Figure 8 Impact of Distortion Using Real-World Data

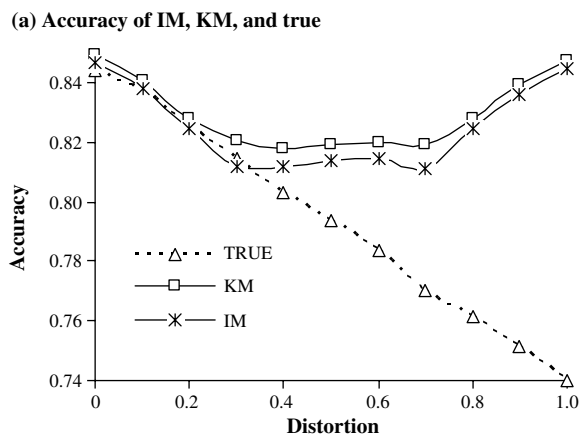


Figure 8(b) Relative improvement of KM over IM

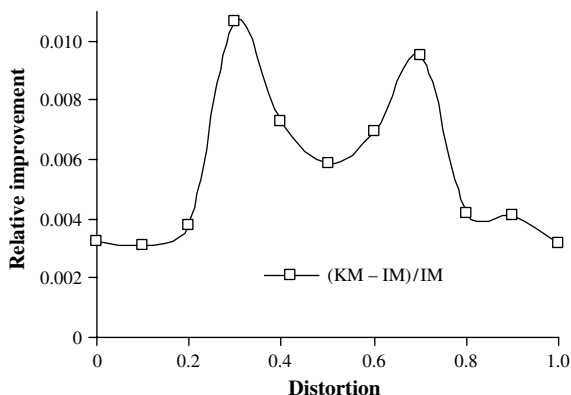
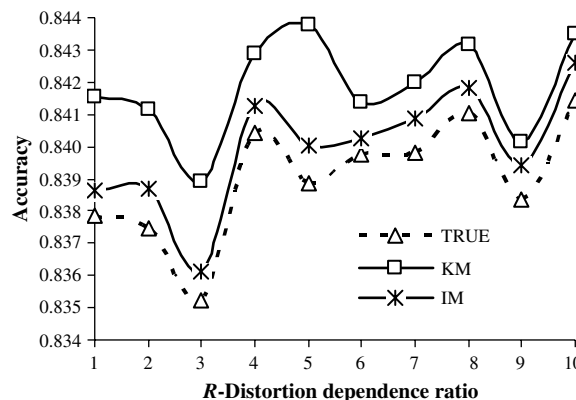


Figure 9 Impact of Dependent Error Generation



distortions in the first five variables are interdependent. From the figure we find that the performance ordering $KM > IM > True$ is preserved under dependent noise. We repeated this experiment 10 more times by selecting other sets of five variables and found that the ordering $KM > IM > True$ is still preserved in most cases. The inequalities $KM > IM$ and $IM > True$ are statistically significant with a p -value of less than 0.001.

5. Variations of KM and IM

In addition to the KM and IM methods, we consider two variations of these methods, denoted by KM-V and IM-H. We first present these variations and then compare all four methods along two dimensions: accuracy and complexity.

5.1. KM-V

Similar to KM, the KM-V method also modifies the knowledge base to account for user input noise. The difference between the KM and KM-V methods lies in how the recommendation is generated for a given observed vector. As discussed in §3, the KM method modifies the knowledge base using the most likely outcome given an observed vector. The KM-V method, on the other hand, modifies the knowledge base using the outcome of the most likely true vector given an observed vector. Thus, Step 2 of the KM method is replaced by the following step in KM-V:

Step 2 (KM-V). Given an observed input vector, find the true vector with the highest conditional probability, and choose the recommendation associated with this most likely true vector as the KM-V recommendation for the observed vector.

We expect KM-V to be less accurate than KM but, in some situations, require less computational effort.

5.2. IM-H

Similar to the IM method, the IM-H method directly modifies input data to account for noise. The IM-H method also acquires inputs based on the True tree structure, and after observing an input follows the branch that corresponds to the most likely true state for that input. The difference between IM and IM-H lies in the conditioning event: IM determines the most likely true state for the current variable conditioned

only on the observed state of the current variable, while IM-H finds the most likely true state based on the observed state of the current variable as well as the observed states of previously acquired variables in the path of the tree. The detailed probability calculation formulas for both IM and IM-H are shown in Appendix C. We expect IM-H to be more accurate than IM but also require more computational effort.

5.3. Accuracy

We conduct additional experiments to compare the performance of all four methods with the True decision tree. The noise and dependency level are randomly chosen. Table 4 summarizes the performance differences across the four proposed methods based on 1,000 observations. The results strongly (with a p -value of less than 0.001) support the following accuracy ordering: $KM > KM-V > IM-H > IM$. From Table 4, we find that the differences within the two knowledge modification methods and within the two input modification methods are not substantial.

When all input variables are mutually independent, we are able to show the following (proof in Appendix D):

PROPOSITION 3. *If all input variables are mutually independent, then KM-V, IM, and IM-H always provide the same recommendation.*

In Proposition 3, the conclusion concerning IM and IM-H is intuitive. If all variables are independent, then the observed states of other variables will not provide any information on the true state of a given variable. With respect to the KM-V method, it can be shown that if the input variables are mutually independent, then the most likely true vector can be obtained by choosing the most likely true state for each variable based only on the observed state of that variable. As shown in Appendix D, this simplification makes KM-V essentially the same as the IM method.

Table 4 Performance Comparison Across Methods

Method	TRUE	KM	KM-V	IM	IM-H
Mean accuracy	0.543	0.647	0.632	0.601	0.612
Std. dev.	0.186	0.160	0.172	0.174	0.171

5.4. Complexity

We examine the computational complexity involved in the knowledge base modification for KM and KM-V, and the effort required to compute the IM and IM-H recommendations for all possible observed vectors. If the number of variables is n and the maximum number of states for all variables is r , then it can be shown that the worst-case complexity associated with IM-H is $O(n^2r^{3n})$, that of KM and KM-V is $O(nr^{2n})$, and that of IM is just $O(nr^2)$. Therefore, the IM-H method is computationally more expensive than the two knowledge base modification methods, and the two knowledge modification methods are, in turn, more complex than the IM method.

We conducted experiments to study the average computational effort required for redesign. The program was coded in C++ and run on a PC with an Intel Celeron 1.80 GHz processor. We varied the number of binary variables from 5 to 12, and recorded the computation times for each method. The results are summarized in Table 5. We find that the redesign times for KM, KM-V, and IM-H all increase exponentially with the number of variables. For example, when the number of variables increases from 11 to 12, the redesign times for KM and KM-V increases by a factor of 2^2 (approximately), and that for IM-H increases by a factor of about 2^3 . As expected, IM takes very little time. From Table 5, we can see that in the 12-variable case, IM-H is close to three orders of magnitude more time-consuming than KM and KM-V, and KM and KM-V are almost three orders of magnitude more time-consuming than IM. The experimental complexity ordering indicated in Table 5 is consistent with the theoretical complexity ordering.

5.5. Summary of Results

Based on our analyses, we conclude that the knowledge base modification methods outperform the input

modification methods in terms of accuracy. Within the two knowledge base modification methods, KM performs better than KM-V. Except for mutually independent inputs, the computational effort required for the two methods is similar. When the inputs are mutually independent, KM-V requires much less effort than KM; however, for this condition, KM-V only performs as well as IM. Therefore, we recommend KM over KM-V. In general, IM-H does not show much superiority over IM in terms of accuracy, while the computation overhead of IM-H is several orders of magnitude higher. Therefore, we do not recommend IM-H. In summary, in choosing an appropriate method for expert system redesign, we only need to consider the KM and IM methods. The advantage of KM over IM is its accuracy, while the advantage of IM over KM is its simplicity. Therefore, in situations where computational cost of redesign is not a major concern, KM should be used. On the other hand, in situations where frequent knowledge base recomputation is costly or infeasible, IM may be preferable.

6. Discussion and Future Research

Although uncertainty in input data is a real obstacle to the widespread deployment of expert systems that acquire information from users in a sequential manner, we know of no previous research that has addressed this problem. This research analyzes the situation where the organization wishes to compensate for noisy inputs to make the best possible recommendation. There are several related issues in the context of noisy inputs that merit further research. First, a more general objective would be to maximize the overall system value to the organization by explicitly considering trade-offs between the costs associated with acquiring the noisy input data with the error costs associated with possible incorrect decisions that may result from making recommendations without fully resolving all the uncertainty in the decision-making process. Second, a more general model of noise could allow errors in one input to be related to errors in another. This could happen where the input variables are not independent of each other, and users provide (inadvertently or purposefully) incorrect input values. Finally, users could attempt to “beat” the system by providing incorrect inputs. Our future research will address these issues.

Table 5 Total KB Redesign Time vs. Number of Binary Variables (in Milliseconds)

Method	# of var.							
	5	6	7	8	9	10	11	12
KM	0	0	8	24	180	711	3,930	14,227
KM-V	0	0	8	40	165	695	4,165	14,335
IM	0	0	0	0	0	0	16	16
IM-H	0	16	110	750	17,937	170,110	1,578,885	11,893,787

Two further variants of the KM method may be considered. Under the first variant, we would still pick the most likely outcome, but as an additional step require that the probability of this outcome be higher than a certain threshold. The use of a threshold becomes especially relevant when the different recommendations are evaluated to be more or less uniformly distributed. For example, with three possible recommendations, if the probability of each recommendation is close to one-third, then it may be unacceptable to use any of these recommendations. The question of course arises as to what needs to be done under such circumstances. The most obvious solution is to flag this case as one that requires more information to solve; i.e., information outside the existing set of inputs must be brought to bear on the problem. Another solution is to try to improve the measurement quality within the existing set of inputs (e.g., attempt to reduce the level of distortion through resampling or other such means). The second variant is to use the recommendation that minimizes error cost. To use an error cost approach, it is necessary to come up with misclassification costs of the kind $C(R_i, R_j)$, where R_i is the correct recommendation for a given observed state and R_j is the chosen recommendation. With the use of such misclassification costs, the recommendation that minimizes the expected misclassification cost can be found. We are, however, somewhat ambivalent about the use of the error cost minimizing recommendation because it may be controversial to apply misclassification costs in a problem domain where human expertise is the “gold standard” for the knowledge. On the other hand, if the rules are being induced from data, the use of misclassification costs to arrive at the least cost recommendation is more acceptable.

Appendix

A. Proof of Proposition 1

We denote the true input vector by \mathbf{True} and observed input vector by \mathbf{Obs} . The state of the i th variable in \mathbf{True} and \mathbf{Obs} is denoted by $True_i$ and Obs_i , respectively. If all of the diagonal elements of a distortion matrix are $(1 - \alpha)$, then the distortion of the variable is α (assume $\alpha < 1/2$).

KM. We first find a distortion level α_1 such that the KM method will always produce the same recommendation as the one given by the True tree. Based on Assumptions 1

and 2 discussed in §2, we obtain the following conditional probabilities when \mathbf{V}^O is observed:

$$P(\mathbf{True} = \mathbf{V}^O | \mathbf{Obs} = \mathbf{V}^O) = \frac{P(\mathbf{True} = \mathbf{V}^O)}{P(\mathbf{Obs} = \mathbf{V}^O)} (1 - \alpha)^N. \quad (A1)$$

$$\begin{aligned} & P(\mathbf{True} \neq \mathbf{V}^O | \mathbf{Obs} = \mathbf{V}^O) \\ &= \sum_{\mathbf{V}^r \neq \mathbf{V}^O} \frac{P(\mathbf{True} = \mathbf{V}^r)}{P(\mathbf{Obs} = \mathbf{V}^O)} \prod_i P(Obs_i = V_i^O | True_i = V_i^r) \\ &< \frac{1}{P(\mathbf{Obs} = \mathbf{V}^O)} \sum_{\mathbf{V}^r \neq \mathbf{V}^O} [P(\mathbf{True} = \mathbf{V}^r) (1 - \alpha)^{N-1} \alpha] \\ &= \frac{1}{P(\mathbf{Obs} = \mathbf{V}^O)} [1 - P(\mathbf{True} = \mathbf{V}^O)] (1 - \alpha)^{N-1} \alpha. \quad (A2) \end{aligned}$$

Apparently, a sufficient condition for the KM recommendation to be the same as that given by the True tree when \mathbf{V}^O is observed is $(A1) \geq (A2)$, which is equivalent to

$$\begin{aligned} P(\mathbf{True} = \mathbf{V}^O) (1 - \alpha) &\geq [1 - P(\mathbf{True} = \mathbf{V}^O)] \alpha \\ &\Rightarrow \alpha \leq P(\mathbf{True} = \mathbf{V}^O). \quad (A3) \end{aligned}$$

If we let $\alpha_1 = P(\mathbf{True} = \mathbf{V}^{\min})$, where \mathbf{V}^{\min} satisfies $P(\mathbf{True} = \mathbf{V}^{\min}) \leq P(\mathbf{True} = \mathbf{V}^r)$, $\forall \mathbf{V}^r \neq \mathbf{V}^{\min}$, then based on the above derivation, we conclude that regardless of what is observed, the KM method will always give the same recommendation as given by True tree.

IM. If we can find a distortion level α_2 such that regardless of what is observed, the most likely true state for each variable is always the observed state itself, then it is guaranteed the IM method will always produce the same recommendation as the one given by the True tree. We first obtain the two conditional probabilities given an observed state for the i th variable:

$$P(True_i = V_i^O | Obs_i = V_i^O) = \frac{(1 - \alpha) P(True_i = V_i^O)}{P(Obs_i = V_i^O)}, \quad \text{and} \quad (A4)$$

$$P(True_i = V_i^j \neq V_i^O | Obs_i = V_i^O) \leq \frac{\alpha \cdot P(True_i = V_i^j)}{P(Obs_i = V_i^O)}. \quad (A5)$$

For the most likely true state to be the same as the observed state, the following is necessary:

$$\alpha < \frac{P(True_i = V_i^O)}{P(True_i = V_i^O) + P(True_i = V_i^j)}, \quad \forall V_i^j \neq V_i^O. \quad (A6)$$

We denote the least and most likely state for the i th variable by S_i^{LL} and S_i^{ML} , and define α_i as:

$$\alpha_i = \frac{P(True_i = S_i^{\text{LL}})}{P(True_i = S_i^{\text{LL}}) + P(True_i = S_i^{\text{ML}})}. \quad (A7)$$

By enforcing $\alpha < \alpha_2 = \min_i \{\alpha_i\}$, (A6) holds for all variables. Therefore, the condition $\alpha < \alpha_2$ ensures that IM and the True tree always produce the same recommendation.

In summary, if the distortions of all variables are below $\alpha_0 = \min\{\alpha_1, \alpha_2\}$, it is guaranteed that KM and IM always produce the same recommendation as the one given by the True tree.

We have assumed that the diagonal elements of each distortion matrix are equal in the discussion. Even if this is not the case, as long as the smallest diagonal element is greater than $(1 - \alpha_0)$, KM and IM always give the same recommendation as the one provided by the True tree.

B. Proof of Proposition 2

Suppose the v th input variable has S possible states and its distortion matrix satisfies the condition specified in Proposition 2. Let O_k, T_l ($k, l = 0, 1, \dots, S-1$) denote the observed states and the true states for this variable. Then, for all O_k , the following always holds:

$$P(O_k | T_l) = P(O_k | T_n), \quad \forall l \neq n, l, n = 0, 1, \dots, S-1. \quad (\text{A8})$$

Therefore, we can denote the conditional probability in the k th column by $P(O_k | T)$ such that

$$P(O_k | T) = P(O_k | T_l), \quad \forall l = 0, 1, \dots, S-1. \quad (\text{A9})$$

PROOF OF (a). We divide all true input vectors into M groups so that all rules in the same group have the same outcome in the True decision table. Let G_m represent the set of rules that have the same outcome, where $m = 0, 1, \dots, M-1$ is the index on different outcomes. We define $P_m(\mathbf{Obs}^O)$ as the probability that one of the rules in G_m is the true input vector if \mathbf{Obs}^O is the observed input vector. Based on the two assumptions given in §2, we have

$$\begin{aligned} P_m(\mathbf{Obs}^O) &= \sum_{r \in G_m} P(\mathbf{True}^r | \mathbf{Obs}^O) \\ &= \frac{1}{P(\mathbf{Obs}^O)} \sum_{r \in G_m} \left[P(\mathbf{True}^r) \prod_i P(Obs_i^O | True_i^r) \right] \\ &= \frac{P(Obs_v^O | T)}{P(\mathbf{Obs}^O)} \sum_{r \in G_m} \left[P(\mathbf{True}^r) \prod_{i \neq v} P(Obs_i^O | True_i^r) \right]. \end{aligned} \quad (\text{A10})$$

The KM recommendation for \mathbf{Obs}^O is the same as the outcome for G_B (the group of rules with the highest probability sum) if $P_B(\mathbf{Obs}^O) \geq P_m(\mathbf{Obs}^O), \forall m$, which is equivalent to:

$$\begin{aligned} &\sum_{r \in G_B} \left[P(\mathbf{True}^r) \prod_{i \neq v} P(Obs_i^O | True_i^r) \right] \\ &\geq \sum_{r \in G_m} \left[P(\mathbf{True}^r) \prod_{i \neq v} P(Obs_i^O | True_i^r) \right], \quad \forall m. \end{aligned} \quad (\text{A11})$$

By varying only the state of the v th variable of \mathbf{Obs}^O , we can obtain a total of $S-1$ different observed input vectors. These vectors have the same value for all variables except

the v th variable. For any one of them, denoted by \mathbf{Obs}^P , we have the following:

$$\begin{aligned} P_m(\mathbf{Obs}^P) &= \frac{P(Obs_v^P | T)}{P(\mathbf{Obs}^P)} \sum_{r \in G_m} \left[P(\mathbf{True}^r) \prod_{i \neq v} P(Obs_i^P | True_i^r) \right] \\ &= \frac{P(Obs_v^P | T)}{P(\mathbf{Obs}^P)} \sum_{r \in G_m} \left[P(\mathbf{True}^r) \prod_{i \neq v} P(Obs_i^O | True_i^r) \right] \\ &\quad (\text{since } Obs_i^O = Obs_i^P \text{ for } i \neq v). \end{aligned} \quad (\text{A12})$$

Multiplying both sides of (A11) by $P(Obs_v^P | T)/P(\mathbf{Obs}^P)$, we have:

$$P_B(\mathbf{Obs}^P) \geq P_m(\mathbf{Obs}^P), \quad \forall m. \quad (\text{A13})$$

Therefore, the KM outcome for \mathbf{Obs}^P is the same as that for \mathbf{Obs}^O . Analogously, we find that all observed vectors that differ from \mathbf{Obs}^O only in the v th variable receive the same KM recommendation as that for \mathbf{Obs}^O . Consequently, the S rules that differ only in the state of the v th variable in the KM table can be compressed into one rule with a “–” value for that variable.

Similarly, all other sets of observed vectors that only differ in the state of the v th variable can also be condensed into one rule. Consequently, in the final condensed KM table, there will be only “–” values for variable v and thus the variable will not appear in the KM tree.

PROOF OF (b). From (A14) below, we conclude that under this case the most likely true state is always the one with the highest prior distribution, regardless of what is observed. Therefore, the IM recommendation is independent of the observed state of the v th variable.

$$P(T_l | O_0) = \frac{P(O_0 | T_l)P(T_l)}{P(O_0)} = \frac{P(O_0 | T)}{P(O_0)} P(T_l). \quad (\text{A14})$$

C. Calculating the Most Likely True State for IM and IM-H

We denote the i th variable reached in a consultation process by $X_{(i)}$. The variables examined before $X_{(i)}$ are therefore $X_{(0)}, X_{(1)}, \dots, X_{(i-1)}$. The true value of $X_{(i)}$ is denoted by $X_{(i)}^T$, and the observed value of $X_{(i)}$ is denoted by $X_{(i)}^O$. At each node, IM chooses the most likely $X_{(i)}^T$ based on $P(X_{(i)}^T | X_{(i)}^O)$, which is obtained as shown below.

$$P(X_{(i)}^T = x_j | X_{(i)}^O) = \frac{P(X_{(i)}^O | X_{(i)}^T = x_j)P(X_{(i)}^T = x_j)}{\sum_{X_{(i)}^T} P(X_{(i)}^O | X_{(i)}^T)P(X_{(i)}^T)} \quad (\text{A15})$$

IM-H makes the decision based on $P(X_{(i)}^T | X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O)$, calculated as shown below.

$$\begin{aligned} &P(X_{(i)}^T = x_j | X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O) \\ &= \frac{P(X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O | X_{(i)}^T = x_j)P(X_{(i)}^T = x_j)}{\sum_{X_{(i)}^T} P(X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O | X_{(i)}^T)P(X_{(i)}^T)}, \end{aligned} \quad (\text{A16})$$

where

$$\begin{aligned}
& P(X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O | X_{(i)}^T = x_j) P(X_{(i)}^T = x_j) \\
&= P(X_{(i)}^T = x_j) \sum_{X_{(i-1)}^T, \dots, X_{(0)}^T} P(X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O, \\
&\quad X_{(i-1)}^T, \dots, X_{(0)}^T | X_{(i)}^T = x_j) \\
&= P(X_{(i)}^T = x_j) \sum_{X_{(i-1)}^T, \dots, X_{(0)}^T} [P(X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O | X_{(i-1)}^T, \dots, X_{(0)}^T, \\
&\quad X_{(i)}^T = x_j) P(X_{(i-1)}^T, \dots, X_{(0)}^T | X_{(i)}^T = x_j)] \\
&= P(X_{(i)}^T = x_j) P(X_{(i)}^O | X_{(i)}^T = x_j) \sum_{X_{(i-1)}^T, \dots, X_{(0)}^T} [P(X_{(i-1)}^O | X_{(i-1)}^T) \\
&\quad \times \dots \times P(X_{(0)}^O | X_{(0)}^T) P(X_{(i-1)}^T, \dots, X_{(0)}^T | X_{(i)}^T = x_j)]
\end{aligned}$$

(the last equality follows from Assumptions 1 and 2).

If the variables are not independent, the conditional probability $P(X_{(i-1)}^T, \dots, X_{(0)}^T | X_{(i)}^T)$ can be calculated using the True decision table.

D. Proof of Proposition 3

IM and IM-H Will Always Give the Same Recommendation. We adopt the same notation in Appendix C. To prove that IM-H and IM always give the same recommendation when the variables are independent, we only need to prove $P(X_{(i)}^T | X_{(i)}^O) = P(X_{(i)}^T | X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O)$ for every node in every path. We first examine the conditional probability for IM-H. The denominator in the right-hand side of (A16) can be further simplified if the variables are independent:

$$\begin{aligned}
& P(X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O | X_{(i)}^T = x_j) P(X_{(i)}^T = x_j) \\
&= P(X_{(i)}^O | X_{(i)}^T = x_j) P(X_{(i)}^T = x_j) \\
&\quad \cdot \sum_{X_{(i-1)}^T, \dots, X_{(0)}^T} [P(X_{(i-1)}^O | X_{(i-1)}^T) P(X_{(i-1)}^T)] \\
&\quad \times \dots \times [P(X_{(0)}^O | X_{(0)}^T) P(X_{(0)}^T)] \\
&= P(X_{(i)}^O | X_{(i)}^T = x_j) P(X_{(i)}^T = x_j) \prod_{k=0}^{i-1} P(X_{(k)}^O). \quad (A17)
\end{aligned}$$

Substituting (A17) into the right-hand side of (A16) and canceling the common term $\prod_{k=0}^{i-1} P(X_{(k)}^O)$ in the numerator and denominator, we have

$$\begin{aligned}
& P(X_{(i)}^T = x_j | X_{(i)}^O, X_{(i-1)}^O, \dots, X_{(0)}^O) \\
&= \frac{P(X_{(i)}^O | X_{(i)}^T = x_j) P(X_{(i)}^T = x_j)}{\sum_{X_{(i)}^T} P(X_{(i)}^O | X_{(i)}^T) P(X_{(i)}^T)} = P(X_{(i)}^T = x_j | X_{(i)}^O). \quad (A18)
\end{aligned}$$

IM and KM-V Will Always Give the Same Recommendation. It can be shown that if the input variables are mutually independent, then the most likely true vector can be obtained by choosing the most likely true state based on the observed state for every variable. If we feed this most

likely true vector into the True tree, we will find that the tree path traversed is exactly the same as the path traversed under the IM method if the same observed vector is used. Therefore we conclude that IM and KM-V will always give the same recommendation if the variables are independent.

References

- Berger, J. O. 1985. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer, New York.
- Breslow, L. A., D. W. Aha. 1997. Simplifying decision trees: A survey. *Knowledge Engrg. Rev.* 12(1) 1–40.
- Clark, P., T. Niblett. 1989. The CN2 induction algorithm. *Machine Learning* 3(4) 261–283.
- Delisio, J., M. McGowan, W. Hamscher. 1993. PLANET: An expert system for audit risk assessment and planning. *5th Annual Conf. Intelligent Systems Accounting, Finance Management*, Stanford University, Stanford, CA (November 11–13).
- Fox, Susannah, L. Rainie, J. Horrigan, A. Lenhart, T. Spooner, C. Carter. 2000. Trust and privacy online: Why Americans want to rewrite the rules. *The Pew Internet and American Life Project Report* (August 20). http://www.pewinternet.org/reports/pdfs/PIP_Trust_Privacy_Report.pdf.
- Graham, L. E., J. Damens, G. Van Ness. 1991. Developing risk advisor: An expert system for risk identification. *Auditing* 10(1) 69–96.
- Hirsh, H. 1994. Generalizing version space. *Machine Learning* 17 5–46.
- Hong, T. P., J. B. Chen. 2000. Processing individual fuzzy attributes for fuzzy rule induction. *Fuzzy Sets Systems* 112(1) 127–140.
- Hong, T. Z., S. S. Tsang. 1997. A generalized version space learning algorithm for noisy and uncertain data. *IEEE Trans. Knowledge Data Engrg.* 9(2) 336–340.
- Lee, C. C. 1990. Fuzzy logic in control systems: Fuzzy logic controller—Part I and Part II. *IEEE Trans. Systems, Man, Cybernetics* 20(2) 404–435.
- Mingers, J. 1989. An empirical comparison of pruning methods for decision tree induction. *Machine Learning* 4(2) 227–243.
- Mookerjee, V., M. Mannino. 1997. Sequential decision models for expert system optimization. *IEEE Trans. Knowledge Data Engrg.* 9(5) 675–687.
- Mookerjee, V., M. Mannino, R. Gilson. 1995. Improving the performance stability of inductive expert systems under input noise. *Inform. Systems Res.* 6(4) 328–356.
- Quinlan, J. R. 1986. The effect of noise on concept learning. R. S. Michalski, J. G. Carbonell, T. M. Mitchell, eds. *Machine Learning*, Vol. 2. Morgan Kaufmann, Los Altos, CA, 149–166.
- Reinwald, L., R. Soland. 1966. Conversion of limited-entry decision tables to optimal computer programs. *J. ACM* 13(3) 339–358.
- Schwayder, K. 1974. Extending the information theory approach to converting limited-entry decision tables to computer programs. *Comm. ACM* 17(9) 532–537.
- Tapscott, D. 1999. IBM is showing leadership on the privacy issue. *Computerworld* 33(17) 34 (April 26).
- Turban, E., J. E. Aronson. 2000. *Decision Support Systems and Intelligent Systems*. Prentice Hall, Englewood Cliffs, NJ.
- Wu, W. Z., W. X. Zhang, H. Z. Li. 2003. Knowledge acquisition in incomplete fuzzy information systems via the rough set approach. *Expert Systems* 20(5) 280–286.