Data Categorization Using Decision Trellises

Paolo Frasconi, Marco Gori, Senior Member, IEEE, and Giovanni Soda, Member, IEEE

Abstract—We introduce a probabilistic graphical model for supervised learning on databases with categorical attributes. The proposed belief network contains hidden variables that play a role similar to nodes in decision trees and each of their states either corresponds to a class label or to a single attribute test. As a major difference with respect to decision trees, the selection of the attribute to be tested is probabilistic. Thus, the model can be used to assess the probability that a tuple belongs to some class, given the predictive attributes. Unfolding the network along the hidden states dimension yields a trellis structure having a signal flow similar to second order connectionist networks. The network encodes context specific probabilistic independencies to reduce parametric complexity. We present a custom tailored inference algorithm and derive a learning procedure based on the expectation-maximization algorithm. We propose decision trellises as an alternative to decision trees in the context of tuple categorization in databases, which is an important step for building data mining systems. Preliminary experiments on standard machine learning databases are reported, comparing the classification accuracy of decision trellises and decision trees induced by C4.5. In particular, we show that the proposed model may offer significant advantages for sparse databases in which many predictive attributes are missing.

Index Terms—Belief networks, classification, connectionist models, context specific independence, data mining, decision trees, machine learning.

1 INTRODUCTION

The purpose of knowledge discovery from databases (KDD) is the exploration of large amounts of relatively raw data in order to extract potentially useful, valid, and understandable patterns [1]. KDD systems are therefore mainly concerned with data understanding, but they typically rely upon data mining modules that can be regarded as low level interfaces to raw data. Hence, it is not surprising that KDD combines a relatively large collection of tools that have been developed within many different (but in some sense related) research communities, such as databases, artificial intelligence, knowledge engineering, statistics, and machine learning. KDD is interesting in database technology since it allows us to exploit the inductive (learned from data) inference paradigm, as opposed to the more traditional deductive (rule-based) inference paradigm that characterizes most of the DBMS designed so far [2]. KDD is also interesting in expert systems engineering inasmuch as it is expected to help overcome the knowledge acquisition bottleneck by means of automatic or semiautomatic techniques for data understanding. Finally, algorithms and learning architectures developed within statistics and machine learning provide a valuable set of tools for solving the challenging tasks occurring at the data mining level of knowledge discovery.

In this paper, we focus on the problem of data categorization, or classification. Classification is rather important in data mining and it has been extensively investigated in statistics, machine learning, and pattern recognition. The problem of classification can be approached using statistical models, such as Bayesian classifiers [3], [4], symbolic systems, such as decision trees [5] or decision lists [6], subsymbolic systems such as artificial neural networks [7], and various hybrid architectures built by combining these systems in different ways.

The method we propose in this paper is rooted in statistics and can be seen as a particular probabilistic belief network. The interest in belief networks (a.k.a. graphical models) lies at the intersection of AI [8], statistics [9], and, more recently, machine learning [10]. In particular, Bayesian networks (BNs), a special case of belief network with directed arcs, now represent a quite popular model for reasoning under uncertainty in AI. During the last decade, most of the research on graphical models was focused on efficiently solving the problem of probabilistic inference, namely the problem of assessing the belief about some variables given the evidence entered by instantiating some other variables. Research on inference is now relatively mature and general and well-understood algorithms are available [11]. The problem of learning Bayesian networks from databases, instead, received increasing research interest during the last few years [12], [13]. Belief networks are appealing as a data mining tool because they offer a flexible way of encoding domain knowledge, are able to deal with missing data, and generate human comprehensible explanations in the form of probabilistic causal relationships [14]. Moreover, they permit many existing architectures and algorithms commonly used for learning (such as Bayesian classifiers, connectionist networks, and hidden Markov models) to be reformulated in a more general framework, thus providing a higher formal level of description that facilitates extension and creation of new or hybrid models [15], [16].

The decision trellis (DT) introduced in this paper is a generalized Bayesian network built upon a linear chain of hidden variables. These hidden variables have an
interpretation that closely follows the semantics commonly attached to nodes of decision trees: single attribute tests in the internal nodes and class labels in the leaves.

The topology is constrained so that arrows are directed from attribute nodes toward the hidden and the class nodes (i.e., no arrow emanates from the class node and no arrow is directed toward attribute nodes). This discriminant direction of modeling is opposite to the generative direction, which is commonly used when applying Bayesian networks to classification (e.g., [4], [7], [18]). One disadvantage of discriminant models is the lack of assumptions about the data. However, these models should be preferred when statistical robustness is important, as detailed in Section 2.3. Unfortunately, having arrows directed from attribute nodes to hidden and output nodes introduces a practical difficulty: The size of the conditional probability tables (CPTs) in Bayesian networks explodes exponentially with the number of attributes. Among the methods that have been proposed to limit this problem, we may cite noisy-OR gates [8] and sigmoidal belief functions [9]. In the case of decision trellises, instead, we chose to exploit an alternative model for the local CPTs, based on context specific independence (CSI) [20]. CSIs, as opposed to (global) conditional independencies, hold for particular configurations of the separating variables and can be effectively used to reduce the parametric complexity of BNs. The CSIs proposed in this paper have a quite natural interpretation that links Bayesian networks to decision trees. In our model, each hidden node has states associated to attribute tests and classification decisions, and it is connected to the previous hidden node and all the predictive attribute. However, we assume that each hidden node is contextually independent of \( n - 1 \) attributes given the state of the previous hidden node. In other words, the state of each hidden node can be interpreted as the decision of testing one particular attribute (like an internal node in decision trees) or as the decision of assigning the instance to some class (like leaves in decision trees).

Decision trellises are also related to other probabilistic decision tree models, such as hierarchical mixtures of experts (HMEs) [21]. However, while HMEs deal with numerical representations of data (as typical in connectionism), decision trellises are conceived for categorical (or discretized) data. One advantage of dealing directly with categorical data is that predictions can be better explained in symbolic terms (like decision trees that can be converted into a set of classification rules). Moreover, in the case of databases containing numerical data, discretization of numerical attributes has been shown to improve classification accuracy in several domains [22], [23].

The paper is organized as follows. In Section 2, we formalize the categorization problem and introduce some basic concepts of supervised learning and probabilistic belief networks. Context specific independencies are reviewed in Section 3. In Section 4, we describe decision trellises, the proposed model for categorization. In Section 5, we derive a custom probabilistic inference algorithm for decision trellises and describe the algorithm for learning the parameters of the model. The connectionist interpretation of decision trellises is discussed in Section 6. Experimental results on a set of standard machine learning databases are reported in Section 7. Finally, some conclusions and insight for future development are given in Section 8.

2 Background

In this paper, we focus on classification tasks for discrete domains. We consider a single relation \( R \) with attributes \( X_1, \ldots, X_N, Y \) (if the database is made of many distinguished relations, we may assume that \( R \) is the corresponding universal relation [24]). Each row in the table is referred to as an instance or case. One particular attribute, denoted by \( Y \), represents the category or class of the instance. We denote by \( C \) the total number of classes. The variables \( X_1, \ldots, X_N \) are referred to as predictive attributes. Note that, in practice, \( R \) is often obtained by projecting a relation with a larger scheme onto a subset of attributes which are believed to be relevant to classification. This attribute selection step is often critical in data mining [25], as well as in pattern recognition [3].

In this paper, we shall adopt the standard convention of denoting variables by uppercase letters and realizations by the corresponding lowercase letters. The domain of a categorical attribute \( X_i \) will be denoted by

\[
\text{dom}(X_i) = \{x_{i}^{1}, x_{i}^{2}, \ldots, x_{i}^{n_i}\}
\]

being \( n_i = |\text{dom}(X_i)| \); the whole set of predictive attributes will be denoted as \( X = (X_1, \ldots, X_N) \). Moreover, the domain of \( Y \) (i.e., the set of classes) will be denoted by \( \{y^1, \ldots, y^m\} \). We shall use the table notation for probabilities. For example, \( P(X) \) is a shorthand for the table \( [P(X = x^1), \ldots, P(X = x^n)] \) and \( P(X, y|Z) \) denotes the two-dimensional table with entries \( P(X = x^i, Y = y|Z = z) \). Finally, \( \sum_Z P(X, Y, Z) = P(Y, Z) \) denotes the marginalization operator on probability tables.

2.1 Tuple Categorization

A supervised learning algorithm in the aforementioned context is any algorithm that takes as input a finite set of training examples (i.e., a table with columns \( \{X_1, \ldots, X_N, Y\} \) in which the class attribute \( Y \) is never missing) and outputs a function \( h(x) \) that approximates, as closely as possible, the "true" dependency between \( x \) and \( y \). The generalization accuracy of a classifier is the expected error rate for predictions made on the whole instance space. Good classifiers are required to yield high accuracy. Although generalization accuracy is usually the main measure of a classifier's performance, the extraction of understandable patterns is also an important issues in knowledge discovery [1]. Hence, less accurate classifiers that facilitate human comprehensible explanations may be preferred to slightly more accurate classifiers that behave as "black-boxes."

Probabilistic approaches to classification are easily motivated to account for different sources of uncertainty that may affect databases, due to systematic or occasional flaws in the data gathering process (e.g., noisy or missing attributes, contradictions, uncertainty about attribute
relevance, etc.). When using a supervised probabilistic approach, classification is performed by evaluating the conditional distribution \( P(Y|x) \) and then applying some decision criterion to predict a class label (in the simplest case, predict the class \( y \) that maximizes \( P(Y = y|x) \)). In the case of unsupervised learning (related to clustering), the learner is trained to estimate \( P(Y,X) \) (i.e., the joint density) without being informed that \( Y \) is the class attribute. In principle, unsupervised learning is more general since the conditional density \( P(Y|X) \) can be easily obtained as \( P(Y,X)/\sum_x P(Y,X) \). However, supervised techniques typically yield better generalization to new instances as they solve a more specific problem and are therefore less subject to idiosyncrasies in the data. Unsupervised learning is particularly useful in KDD when the classes of tuples are unknown and, therefore, also the partition of training data into homogeneous groups must be inferred [17].

In general domains predictive attributes may be numerical (i.e., real numbers) or categorical (i.e., symbols chosen from a finite set). The model we are going to present is designed for categorical attributes only. At first this may appear as a limitation, as most real world databases (for example, in economics, medicine and many other domains) do contain numerical attributes. However, dealing with numerical attributes is typically more difficult than dealing with categorical ones. For example, decision trees deal with numerical attributes by generating elementary axis-perpendicular splits, which corresponds to comparing each attribute with respect to a given threshold. However, it has been shown that decision trees trained on preprocessed data, in which the numerical attributes have been previously discretized, achieve higher generalization accuracy in several domains [22]. Moreover, in some domains, simple classifiers such as naive-Bayes have been shown to yield accuracy comparable to the accuracy of decision trees when discretizing numerical attributes [22]. Finally, classifiers as simple as 1R (equivalent to a decision tree with only one decision node) have been shown to yield surprisingly good accuracy when coupled to proper data discretization algorithms [23].

2.2 Probabilistic Belief Networks

Belief or conditional independence networks became popular in artificial intelligence as a tool for reasoning in probabilistic expert systems [8]. More generally, belief networks are effectively used in statistics for representing and manipulating complex probability distributions [9]. As a matter of facts, many learning systems, such as Boltzmann machines [30], multilayered perceptrons [7], hidden Markov models (HMMs) [31], and input/output HMMs (IOHMMs) [32] (just to mention some of them), can be regarded as particular graphical models.

A belief network is an annotated graph in which nodes represent random variables\(^3\) in the universe of discourse, and missing edges encode a set of conditional independence statements amongst these variables. Given a particular state of knowledge, the semantics of belief networks determine whether collecting evidence about a set of variables does modify one’s belief about some other set of variables. Specifically, let \( U \) denote the universe of discourse and let \((X,Y,Z)\) be disjoint subsets of \(U\). The variables \( X \) and \( Y \) are said to be conditionally independent given \( Z \), denoted \( X \perp Y|Z \), if \( P(X,Y,Z) = P(X|Z) \) whenever \( P(Y,Z) > 0 \). Intuitively, if \( Z \) is known, then further knowledge about \( Y \) does not affect one’s belief about \( X \). A conditional independence model is a collection of triplets \((X,Y,Z)\) such that \( X \perp Y|Z \) holds true. A graph with nodes associated to variables in \( U \) is an independency map for the independence model if the nodes associated to \( X \) and \( Y \) are graphically separated by the subset of nodes \( Z \) for each triplet such that \( X \perp Y|Z \). Graphical separation criteria for verifying conditional independency can be defined for undirected graphs (a.k.a. Markov networks), directed acyclic graphs (DAGs) (a.k.a. Bayesian networks), and chain graphs [33]. These criteria are referred to as \( u \)-separation, \( d \)-separation, and \( e \)-separation, respectively.

Belief networks, however, are not limited to quantitatively encoding conditional independencies, but they also quantitatively specify the parameters of the probability distribution over the universe of discourse. In particular, in the case of Bayesian networks (BNs), it can be shown that the table \( P(U) \) can be factorized as

\[
P(U) = \prod_i P(U_i|Pa_i)
\]

where \( Pa_i \) denotes the parents of \( U_i \). Hence, BNs are specified by a DAG with local density models \( P(U_i|Pa_i, \theta_i) \) attached to each node, where \( \theta_i \) is a set of parameters for the local density. In the case of unrestricted multinomial models, \( P(U_i|Pa_i, \theta_i) \) is simply a conditional probability table (CPT) with entries \( \theta_{ijk} = P(U_i = u_i|Pa_i = p_{ai}) \). These parameters can either be elicited from experts or learned from examples.

2.3 A Discriminant Approach to Categorization

When using a graphical model for categorization, one important distinction that can be made concerning the orientation of the arcs in the network is whether the class node \( Y \) is a root node or a leaf. These two extreme alternatives correspond to two distinct directions for probabilistic categorization: the generative direction and the discriminant direction, respectively. Taking the generative (or causal: \( Y \rightarrow X \)) direction corresponds to the assumption that the category causes the observed attributes (e.g., in a medical domain, a disease is the cause of its symptoms). The causal direction is familiar in Bayesian statistics and, in fact, the naive-Bayes classifier [3, 4] is a prototypical example of generative model. Class-conditional and prior densities (i.e., \( P(X|Y) \) and

\[
P(X) \]

1. A much better method would use oblique splits, i.e., by means of arbitrary hyperplanes; however, finding optimal oblique splits is intractable [26].

2. The scope of the independence semantics of these networks goes beyond probability distributions and there exist other domains, such as relational algebra, in which formal properties of conditional independence can be defined and exploited by means of graphical representations [27, 28, 29].

3. Hence, the terms “node” and “variable” are often used interchangeably in the context of belief networks.
$P(Y)$, respectively) need to be specified at the quantitative level of description, requiring statistical assumptions about the form of these densities. A wrong assumption may significantly degrade performance.

Taking the discriminant (or diagnostic: $X \rightarrow Y$) direction, on the other hand, strong assumptions about class-conditional densities need not to be made. As an interesting example, a single neuron with logistic output (a log-linear model in statistics) is a model of $P(Y|X)$, which is consistent with every class-conditional density belonging to the exponential family, provided that "dispersion parameters" (such as the covariance matrix, in the case of Gaussian densities) are independent of the class $y$ [34]. Since diagnostic modeling is not strongly committed to any particular choice of the class-conditional densities, more robustness may be expected when prior knowledge about data is scarce. Decision trees [5] and hierarchical mixtures of experts [21] are examples of models pursuing the diagnostic direction.

3 ASYMMETRIC BAYESIAN NETWORKS

An independence statement of the form $X \perp Y|Z$ expresses a global independence relationship, since $X$ and $Y$ are independent for every configuration of the variables in $Z$. As shown in Section 2.2, such independencies allow us to simplify the form of the distribution over the universe of discourse and, for example, in the case of BNs, the domain model is reduced to a set of local CPTs. In some cases, however, there is much more structure that can be exploited. In particular, it is possible that only partial knowledge about $Z$ is sufficient to make $X$ and $Y$ independent. For example, consider the Bayesian network in Fig. 1. The network contains a binary variable $Y$ with three binary parents $\Omega$, $X_1$, and $X_2$. The local CPT for node $Y$ would contain $2^3 = 8$ parameters (the number of parameters grows exponentially with the number of parents). However, it can be observed that given $\Omega = \omega^j$, the probability that $Y = 1$ only depends on $X_1$ and, given $\Omega = \omega^k$, the probability that $Y = 1$ only depends on $X_2$. Hence, only four parameters are needed to specify the CPT attached to node $Y$. Such independencies are sometimes called local, or asymmetric, or context-specific, since they only hold for particular configurations of the separating variables. Traditional BNs fail to capture these regularities at qualitative level (these independencies are buried into the values of the parameters and cannot be read off in the graphical representation, nor efficiently used to perform inference). A systematic attempt to formalize and exploit context specific independency (CSI) is reported in a recent paper by Boullier et al. [20]. Following their formalism, let $X, Y, Z$, and $C$ be four disjoint subsets of the universe of discourse, and let $c$ be a configuration over the variables $C$. Then, $X, Y$ and $Z$ are said to be conditionally independent given $Z$ and the context $c$, denoted $X \perp Y|Z, c$ if $P(X|Z, c, Y) = P(X|Z, c)$ whenever $P(Y, Z, c) > 0$. Clearly, if $X \perp Y|Z, c$ holds for every possible configuration $c$, then also $X \perp Y|Z \cup C$ holds. However, contextual independence permits us to express a finer level of detail than conditional independence.

CSI can be graphically described by augmenting a traditional BN with labels on the edges. These labels may be assimilated to Boolean expressions on the states of the parents of the node on which the edge being considered is incident to. Given a particular context $c$, all the edges labeled by expressions which are satisfied by the configuration $c$ are called vacuous or inhibited by $c$ (edges labeled by the constant expression FALSE cannot be inhibited and thus play the same role as edges in standard BNs). Then, CSIs are simply checked by means of the usual d-separation criterion [8] on the traditional BN $D$ obtained by deleting all the edges inhibited by $c$. In [20], it is argued that such separation criterion is sound and complete. In this paper, networks that encode CSIs are called asymmetric Bayesian networks (ABNs). Such networks are pictorially represented by adding dashed edges pointing to edges. Dashed edges indicate which context variables may inhibit the regular edge they point to. As in standard BNs, ABNs store, for each node $U$, the conditional probability table $P(U|Pa_U)$. The size of these tables, however, may be significantly reduced by context specific independencies.

**Example.** Let $U = X_1, X_2, X_3, X_4$ and consider the asymmetric Bayesian network depicted in Fig. 2. Suppose

```
<table>
<thead>
<tr>
<th>$\Omega$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$P(Y = 0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega^1$</td>
<td>0</td>
<td>0</td>
<td>$p_0$</td>
</tr>
<tr>
<td>$\omega^1$</td>
<td>0</td>
<td>1</td>
<td>$p_0$</td>
</tr>
<tr>
<td>$\omega^1$</td>
<td>1</td>
<td>0</td>
<td>$p_1$</td>
</tr>
<tr>
<td>$\omega^1$</td>
<td>1</td>
<td>1</td>
<td>$p_1$</td>
</tr>
<tr>
<td>$\omega^2$</td>
<td>0</td>
<td>0</td>
<td>$p_2$</td>
</tr>
<tr>
<td>$\omega^2$</td>
<td>0</td>
<td>1</td>
<td>$p_3$</td>
</tr>
<tr>
<td>$\omega^3$</td>
<td>1</td>
<td>0</td>
<td>$p_2$</td>
</tr>
<tr>
<td>$\omega^3$</td>
<td>1</td>
<td>1</td>
<td>$p_3$</td>
</tr>
</tbody>
</table>
```

Fig. 1. Example of context specific independencies.
The decision trellis model

The belief network for the proposed classifier is shown in Fig. 3. Nodes are associated to three groups of variables: a set of input nodes $X_j, j = 1, \ldots, N$ is associated to the predictive attributes, an output node $Y$ is associated to the class, and a chain of hidden nodes is associated to a set of latent decision variables $\Omega_h, h = 0, \ldots, H - 1$. The network is connected so that $\Omega_0$ is a parent of $\Omega_h$, all the attributes are parents of $Y$ and $\Omega_h$ for $h = 1, \ldots, H - 1$, and $\Omega_{H-1}$ is a parent of $Y$. Hence, the network follows the discriminant direction of modeling in the sense that arrows emanate from attribute nodes and converge toward the class node.

Latent (unobserved) variables are often found in belief networks used for supervised and unsupervised learning (for example, in mixture models, multilayered perceptrons, and hidden Markov models). Latent variables can be justified by assuming the existence of unobservable states in the process responsible of generating the data. In the present model, hidden variables are associated to decisions⁴ about the instance to be classified, either to test one attribute or to decide for the class of the instance. The states for hidden variables are picked from the set $\{r^0, \ldots, r^N, y^1, \ldots, y^f\}$ so that $\Omega_h = r^i$ means that attribute $X_i$ is tested at level $h + 1$, and $\Omega_h = y^j$ means that the instance is classified as belonging to the $i$th class. Hence, the states of hidden variables have an interpretation that closely follows the semantics of nodes in a decision tree (i.e., attribute tests in the internal nodes and class labels in the leaves). The depth of the model $H$ (the number of decision levels) clearly affects model complexity.

Note that “decisions” in the context of this model are not related to decisions in influence diagrams (where they are typically under the control of an agent).

According to (1), the CPTs for the model contain the probabilities $P(\Omega_h | \Omega_{h-1}, X)$ and $P(Y | \Omega_{H-1}, X)$. Without any constraints, the size of these tables would grow exponentially with $N$, the number of predictive attributes. This is a problem common to diagnostic approaches to classification: Arrows in the belief network emanate from the attribute nodes and are directed toward the class node so that the class node has as many parents as the number of predictive attributes. Generative approaches (such as the naïve-Bayes classifier) do not suffer this problem as arrows are directed in the opposite direction (and, hence, nodes tend to have a smaller number of parents). Solutions to this “many parents” problem, among the others, are the introduction of noisy OR-gates [8], in which multiple causes are assumed to exert independent contributions on a common effect; or using sigmoidal belief networks [19], in which the conditional probabilities are generated by a log-linear model. The approach we propose here is based on the “decision” semantics attached to nodes (either decide to test one attribute or decide to assign class $y^j$ to the instance) and takes advantage of the following CSI:

$$\Omega_h \perp X^i | X_j, \omega^f_{h-1},$$

where $X^i$ denotes the set of attributes

$$\{X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_N\}.$$

The interpretation of relation (2) is straightforward: If $\Omega_{h-1}$ is in state $\omega^i_{h-1}$, signifying that attribute $X_i$ will be tested at the next level $h$, then the next hidden variable $\Omega_h$ is probabilistically independent of the values of the other attributes $X_k, k \neq i$. A similar CSI is assumed for the class variable $Y$:

$$Y \perp X^i | X_j, \omega^f_{H-1}.$$

The model is further constrained as follows:

$$P(\Omega_h = \omega^i_{h} | \Omega_{h-1} = y^j, x) = 0 \quad \forall \ i = 1, \ldots, N$$

$$P(\Omega_h = y^j | \Omega_{h-1} = y^j, x) = \delta_{j, c} \quad \forall \ \ell = 1, \ldots, C,$$

where $\delta$ being the Kronecker symbol defined as $\delta_{c, c} = 1$ if $\ell = c$ and $\delta_{c, c} = 0$, otherwise. Equations (4) mean that as soon as a decision node $\Omega_h$ is in state $y^j$, signifying that the instance is classified as belonging to class $y^j$, no other attribute tests are made at the successive decision levels.
In other words, when a hidden node $\Omega_h$ is instantiated with a class label $y_c$, predictions are made as if the chain was effectively pruned at level $h$. Node $\Omega_h$ has no parents. Hence, instantiating $\Omega_0$ with a class label corresponds to deciding the class without observing the attributes (resembling a classifier that predicts according to the majority of the class labels observed in the training set). Instantiating $\Omega_0$ with a class label corresponds to deciding the class using only one attribute, like in Holte's 1R classifier [23].

The CSI assumptions in (2), (3), (4) drastically simplify the complexity of the model since each decision node needs to store only $N(N + C) \sum_{i=1}^{N} r_i$ conditional probabilities, instead of $N(N + C) \prod_{i=1}^{N} r_i$ (as it would be required by considering the joint effect of all attributes on the state of hidden decision variables). An interesting consequence of (2) is that the knowledge of all the hidden variables makes the attributes independent. We may relate this property to a similar property of the naive-Bayes classifier [4], in which knowledge of the class makes the attributes independent.

The parameters for a decision trellis are defined as

$$\theta_{0ijk} = P(\Omega_0 = \omega|\Omega_{h-1} = \omega^j, X_j = x^k_j)$$

$h = 1, ..., H - 1; i = 1, ..., N; j = 1, ..., N; k = 1, ..., r_j;

\theta_{0i} = P(\Omega_0 = \omega^i)$

$i = 1, ..., N_i$;

$$\theta_{0ijk} = P(Y = y|\Omega_{h-1} = \omega^j, X_j = x^k_j)$$

$h = 2, ..., H; c = 1, ..., C; j = 1, ..., N; k = 1, ..., r_j;

\theta_{i} = P(Y = y^i)$

c = 1, ..., C.

5. Inference and Learning

Let $X$ and $Y$ be two disjoint subsets of the universe of discourse $U$, and suppose the state of $X$ is known. The problem of probabilistic inference in BNs consists of computing the conditional probability $P(Y|X, D)$, where $D$ is the network being considered. The inference problem can be solved in different ways, depending on the topology of the graph. In the case of singly connected networks (polytrees), inference can be solved by algorithms based on local propagation of messages, sometimes referred to as $\pi - \lambda$ propagation [8]. In these algorithms, for every node (or variable) $V$, the evidence (entered as values for the variables in $X$) can be separated in two disjoint sets $X^+(V)$ (the intersection between $X$ and the nondescendants of $V$) and $X^-(V) = X \setminus X^+(V)$. Since $D$ is singly connected, $V$ separates $X^+(V)$ and $X^-(V)$. Hence, belief propagation is performed by having each node $u$ pass a message to its children and a $\lambda$ message to its parents, where $\pi(V) = P(V|X^+(V))$ and $\lambda(V) = P(X^-(V)|V)$. In the case of multiply connected networks (i.e., networks with undirected cycles), clustering approaches are generally needed. In these cases, the network is first compiled into a so-called junction tree, whose nodes are associated to clusters of variables. Then, inference proceeds by local message passing among clusters [11]. One of the difficulties with clustering approaches is that inference is less and less efficient as the connectivity of the network increases.\(^5\) Now, decision trellises are not polytrees. However, mainly because of the context specific independencies encoded in the network, it is possible to derive a simple and efficient customized tailored inference algorithm. Such an algorithm is detailed below, and it has an interesting connectionist interpretation described in Section 6.1. The learning algorithm for decision trellises is just a particular case of parametric learning in BNs with hidden variables. In Section 5.2, we propose a simple solution based on the expectation-maximization (EM) algorithm, which relies on the custom inference algorithm for computing the expected sufficient statistics in the E-step.

5.1 Inference in Decision Trellises

The inference algorithm for DTs is vaguely inspired by the Baum-Welch algorithm\(^6\) used for HMMs and IOHMMs. In fact, hidden variables in DTs are organized in a linear chain as in HMMs and IOHMMs. Since DTs are used as classifiers, evidence is always inserted into attribute nodes (during prediction) and also into the class node (during learning). Hence, only a limited number of inference queries to the network need to be answered. Following [8], let us decompose the belief about hidden variables $\Omega_h$ into a causal support

$$\pi(\Omega_h) = P(\Omega_h|x)$$

and a diagnostic support

$$\lambda(\Omega_h) = P(y|\Omega_h, x),$$

using Bayes' theorem:

$$P(\Omega_h|x, y) = \frac{P(y|\Omega_h, x)P(\Omega_h|x)}{P(y|x)} \propto \pi(\Omega_h)\lambda(\Omega_h). \tag{6}$$

In the following derivation, we shall denote by $\pi_i(\Omega_h)$ the $i$th entry of the $N + C$-dimensional table $\pi(\Omega_h)$ (an attribute test probability for $i = 1, ..., N$ and a class decision for $i = N + 1, ..., N + C$) and by $\pi_{i, y}(Y)$ the $i$th entry of the $C$-dimensional table $\pi(Y)$. A similar notation will be used for $\lambda$ tables. The causal support can be recursively computed by forward propagation along the chain, using the CSI (2):

$$\pi_i(\Omega_h) = \theta_{ih}, \quad i = 1, ..., N + C$$

$$\pi_i(\Omega_h) = \sum_{j=1}^{N} \theta_{ij}\pi_j(\Omega_{h-1})$$

$$i = 1, ..., N$$

$$\pi_{N+C}(\Omega_h) = \sum_{j=1}^{N} \theta_{ij}\pi_j(\Omega_{h-1}) + \pi_{N+C}(\Omega_{h-1})$$

c = 1, ..., C.

\(^5\) Formally, it can be shown that (even approximate) inference in BNs is NP-hard [36]. In practice, inference can be computationally afforded provided that the network is not too densely connected.

\(^6\) The Baum-Welch algorithm has been recently shown to be a special case of the junction tree algorithm [97].
where \( x_j \) is the realization of attribute \( X_j \) in the instance being considered. Similarly,

\[
\pi_c(Y) = \sum_{i=1}^{N} \lambda_{i}(\Omega_{h-1}) \theta_{k,i} \pi_{j}(\Omega_{h-1}) + \pi_{N+1,c}(\Omega_{h-1}) \quad c = 1, \ldots, C.
\]

(8)

Since \( \pi_c(Y) = P(Y = y | x) \), this last equation will also be used to obtain the class predicted by the model.

The diagnostic support is recursively computed by backward propagation along the chain:

\[
\lambda_c(\Omega) = \delta_c y \quad c = 1, \ldots, C
\]

\[
\lambda_{N+1,c}(\Omega) = \delta_c y
\]

\[
\lambda_i(\Omega_{h-1}) = \theta_{y \Omega_{h-1}} y
\]

\[
\lambda_i(\Omega) = \sum_{i=1}^{N} \lambda_i(\Omega_{h-1}) \theta_{k,i} \pi_{j}(\Omega_{h-1}) + \pi_{N+1,c}(\Omega_{h-1}) \theta_{k,i} \pi_{j}(\Omega_{h-1})
\]

\[ i = 1, \ldots, N; h = 1, \ldots, H \]

(9)

where \( y \) is the class (target) of the instance. It can be noted that the above equations have a striking resemblance to the Baum-Welch equations for inference in HMMs (see, e.g., [31]).

### 5.2 A Learning Procedure for Decision Trellises

Since the topology of the network is deterministically known, the learning problem is simply reduced to quantitative learning, i.e., estimating the parameters of the model given a dataset \( D \). Quantitative learning in Bayesian networks is straightforward assuming parameters independence and data completeness [13]. In the case of decision trellises, however, the problem is complicated by the presence of hidden variables. A common approach for learning in the presence of missing data is to estimate the parameters according to the maximum likelihood (ML) principle. In this way, learning is cast into an optimization procedure applied to the likelihood function \( P(D | \theta) \). Common approaches for solving this optimization problem are gradient ascent and the expectation-maximization (EM) algorithm [38]. Both methods are iterative and, under certain regularity conditions, converge to a local maximum of the likelihood function.

Early uses of EM were mainly in unsupervised learning (e.g., for training hidden Markov models [31] or mixture-based density models [39]). More recently, EM algorithms have been applied to supervised learning problems [21], [32]. The algorithm iteratively fills in missing values in the data by effectively assigning credit to the hidden state variables given the observed data.

The generic \( t \)th iteration of EM consists of an expectation step followed by a maximization step. The expectation step consists of computing the expected sufficient statistics \( \tau \) for the parameters, given the observed data and the previous parameters \( \theta^{(t-1)} \).

In the case of generic BNs unrestricted multinomial distributions, the sufficient statistics are simply the counts \( N_{ijk} \): How many times the generic variable \( U_i \) was found in the \( k \)th state \( \omega^i_k \) while its parents \( P_h \) were found in the \( j \)th configuration \( \omega^{j}_h \). The intuition behind the algorithm is that these counts are not available because the state of hidden variables is unknown; however, their expectations can be computed by solving a probabilistic inference problem. An EM algorithm starts with some initial parameters \( \theta^{(0)} \) and iterates the following two steps for \( t = 1, 2, \ldots \):

- **E-step**: Compute the expected sufficient statistics given the observed data and the previous parameters:

\[
\tau_{ijk}^{(t)} = E[N_{ijk} | D, \theta^{(t-1)}] \quad (10)
\]

The computation proceeds by solving a probabilistic inference problem with evidence entered into the attribute nodes and the class node.

- **M-step**: Update the parameters using the expected sufficient statistics:

\[
\theta_{ijk}^{(t)} \leftarrow \frac{\tau_{ijk}^{(t)}}{\sum_{i,j,k} \tau_{ijk}^{(t)}}
\]

(11)

Dempster et al. [38] showed that, under some regularity conditions, EM converges to a local maximum of the likelihood function.

In the case of decision trellises, the expected sufficient statistics for the parameters are the expected counts \( N_{hijk} \) of the number of times \( \Omega_h = \omega^i_h, \Omega_{h-1} = \omega^j_{h-1} \), and \( X_j = \omega^k_j \). These quantities can be simply computed by summing over the training examples the probabilities \( P(\omega^i_h, \omega^j_{h-1} | y, x) \). For a generic training example, these probabilities are computed as follows:

\[
P(\omega^i_h, \omega^j_{h-1} | y, x, \theta)
\]
5.3 Missing Attributes

The presence of corruption in data is quite an important issue for learning systems. Two different kinds of corruption can be considered. Either some attributes are noisy (i.e., their observed value differs from their true value) or missing (i.e., they are not observed). The problem of missing attributes has been approached in different ways in machine learning and pattern recognition. Quinlan's C4.5 [5], for example, deals with missing data by considering all the possible outcomes of a test on an unknown attribute (as many as the cardinality of the domain of the missing attribute) and weighting each alternative with the relative frequency, as measured from the training set.

Models based on probabilistic belief networks have the advantage that missing information can be dealt with in a simple and consistent way: Evidence is only entered into nodes associated with visible attributes and then inference is run in the normal way. Unfortunately, in the case of decision trellises, the exact inference procedure in the presence of missing attributes is computationally expensive since there is the need of propagating the evidence entered into the visible attribute nodes into the nodes associated to unobserved attributes. In fact, in the presence of missing attributes, independence relation (2) does not hold. Let us consider, for example, the network shown in Fig. 4. If $X_j$ is not observed, the path $X_j \rightarrow \Omega_{h-1} \rightarrow X_j \rightarrow \Omega_h$ in the above network is active and, thus, $\Omega_h$ and $X_j$ are not d-separated by $\omega_j$. Therefore, $\Omega_h$ is conditionally dependent of $X_j$, even though $\Omega_h$ is instantiated to $\omega^j$, meaning that $X_j$ should be tested. The consequence is that, in the presence of missing attributes, exact inference is intractable.

In the following, we introduce a heuristic inference procedure which is tractable, although it involves drastic simplifying assumptions. The method is inspired by a work by Ghahramani and Jordan [41], who consider joint density modeling for supervised and unsupervised learning. In order to avoid the complexity of modeling a joint density of attributes that can (possibly) be all statistically dependent, they propose a mixture model along with the assumption that attributes are independent given a particular component of the mixture. In this method, each mixture maintain a separate (factorized) model and maximum likelihood estimation is obtained by using the EM algorithm.

As explained in Section 4, the probability of hidden variables $\Omega_h$ in decision trellises only depends on the attribute $X_j$ being tested. If we assume that $X_j$ is independent of other attributes once test $\omega^j$ has been decided at some level $h$, then, when attribute $X_j$ is not observed, we have:

$$P(\omega^j_h|\omega^j_{h-1}, x^j) = \sum_{x^j} P(\omega^j_h, x^j|\omega^j_{h-1}, x^j)$$

$$= \sum_{x^j} P(\omega^j_h|\omega^j_{h-1})P(x^j|\omega^j_{h-1})$$

$$= \sum_{x^j} \theta^j_{h,k} p_{i|j}.$$  \hspace{1cm} (13)

Assuming a multinomial distribution for categorical attributes, the model for the missing datum $X_j$ in the node $\Omega_j$ has the simple form:

$$P(x_j|\omega^j_j) = \prod_{h=1}^{r_j} p_{i|j}$$

where $z_{ik}$ is an indicator variable such that $z_{ik} = 1$ if $X_j = x^j_i$. As a further simplification, the local models for missing data are estimated independently of $h$ from the training data (simply replacing $\mu_{ijk}$ with the frequency of occurrence of value $x^j_i$ for the attribute $X_j$). Clearly, this method of dealing with missing data is not fully rigorous because of the independency assumption in (13). However, it can be justified as a heuristic and the experimental evaluation (see Section 7.4) is very satisfactory.

6 RELATIONSHIPS TO NEURAL NETWORKS AND DECISION TREES

6.1 A Connectionist Interpretation

Inference equations (7) for decision trellises can be interpreted as propagation of neural activities in a connectionist network in which nodes are associated to the states of the variables in the universe of discourse. In order to understand the connectionist interpretation of decision trellises, let us introduce the following state-unfolding operation. Given a decision trellis $D$, the state-unfolding graph $N(D)$ associated with $D$ is a second-order neural network [42] whose architecture is defined as follows:

- Nodes in $N(D)$ are associated with the states of the variables $X_1, \ldots, X_N, \Omega_0, \ldots, \Omega_{H-1}, Y$.
- Each node in $N(D)$ is labeled with the belief for the state of the corresponding variable, conditioned on the predictive attributes. For example, node $\omega^j_i$ is labeled with $P(\Omega_h = \omega^j_i | x)$ and node $y^j_i$ is labeled with $P(Y = y^j_i | x)$. Hence, units in the unfolded network correspond to local “one-hot” encoding of domain variables.
- For each variable $\Omega_h$, $h = 1, \ldots, H - 1$ the network $N(D)$ contains a set of multiplier nodes $m_{ijk}$ receiving from $\omega^j_{i-1}$ and $x^j_i$ and sending to nodes $\omega^j_k$, $i = 1, \ldots, N$ with connection weight $\theta_{ijk}$, and to

![Fig. 4. Independence relation (2) does not hold in the presence of missing attributes (see text for explanation).](image-url)
TABLE 1
Scheme for the Sample “Credit” Domain

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Housing</td>
<td>own, rent, for free</td>
</tr>
<tr>
<td>Car</td>
<td>yes, no</td>
</tr>
<tr>
<td>Education</td>
<td>low, high school, master, doctoral degree</td>
</tr>
<tr>
<td>Employed</td>
<td>unemp, ≤ 20 yrs, &gt; 20 yrs</td>
</tr>
<tr>
<td>Customer class</td>
<td>bad, good</td>
</tr>
</tbody>
</table>

nodes $y^c_0, c = 1, \ldots, C$ with connection weight $\theta_{(\text{mfc})}$; moreover, the network $N(D)$ contains a set of multiplier nodes $m_{jk}$ receiving from $x^j$ and $x_0^f$, and sending to $y^c$ with connection weight $\theta_{jk}$.

Example. In order to illustrate the relationships between decision trellises and connectionist networks, we introduce a toy financial domain that we shall refer to as “credit.” The task consists of predicting whether a customer should be granted a credit by a bank, depending on a very simplified database scheme in which predictive attribute measure economic wealth and instruction of the customer. The scheme of the database and the admissible values for the attributes are detailed in Table 1.

Fig. 6 and Fig. 7 show the state-unfolding graph in the case of the “credit” example of Fig. 5. The model proposed in this paper was named after the state-unfolding structure, which has the form of a trellis. Each level in the trellis corresponds to one hidden variable in the chain. The last level contains only class labels since the last variable of the chain has states in \{1,...,C\}. Arcs in the trellis correspond to “transitions" from states at level $h-1$ to states at level $h$ and paths from level 0 to level $H$ are probabilistically selected according to the outcomes of attribute tests. For the sake of simplicity, attribute nodes have not been shown in the trellis of Fig. 6. A partial detail of the trellis connectivity pattern is given in Fig. 7, in which attribute nodes have also been expanded.

6.2 Relationships to Hierarchical Mixtures of Experts

Adaptive mixtures of experts and hierarchical mixtures of experts (HME) [21] are a divide and conquer approach to supervised learning in connectionist models. A mixture of experts is composed by a set of subnetworks (experts) that compete to gain responsibility in modeling outputs in a given region of input space. The system output $y$ is obtained as a convex combination of the experts’ outputs $y_i$: $y = \sum_i y_i \pi_i$, where the weights $y_i$ are computed as a parametric function of the inputs by a separate subnetwork (gating network) that assigns responsibility to different experts for different regions of the input space.

In the HME architecture, each expert is recursively decomposed as another mixture of (sub)experts, yielding a structure which can be easily interpreted as a probabilistic decision tree.

There are, however, several differences between decision trellises and HME, which are worth being discussed. First, decision trellises are conceived for categorical data while HME, like neural networks, deals with numerical representations of data. A second difference (related to the first) is that HME do not perform tests on single attributes. Instead, every gating network operates on the whole real valued vector that represent each instance. Moreover, when used for classification, HME classifies the input pattern only at the bottom of the tree using the expert networks attached to the leaves of the tree. In decision trellises, classification decisions are permitted at every level of the model.

6.3 Decision Trellises with 0-1 Parameters Are Decision Trees

If all conditional probabilities degenerate to the extreme values 0 or 1, all nodes compute a deterministic function of the values of their parents. In this case, arcs of the unfolded network with null weight can be eliminated, resulting in a tree structure which is equivalent to a decision tree. An example of this reduction is shown in Fig. 8 for the "credit” domain. Notice, however, that the converse is not true, i.e., there exist decision trees that cannot be represented as decision trellises with 0-1 probabilities: specifically, all those trees that test the same attribute at the same level in two or more distinct paths.

---

Fig. 5. Decision trellis for the credit domain represented as an asymmetric Bayesian network.
7 EXPERIMENTAL RESULTS

7.1 Datasets

We applied the induction algorithm to 16 real-world databases from the University of California at Irvine Repository [43]. Databases "breast-w," "cleve," "colic," "diabetes," "heart," "pima," and "sick" are medical databases. For example, "heart" is a database of clinical records about heart disease with 14 attributes including "age," "sex," "chest pain type" (angina, abnang, stang, asympt), "cholesterol," etc.; the task is to predict whether heart disease is present or not. Databases "australian," "crx," and "german" contain information relevant to the problem of bank credit approval. Data in "australian" and "crx" are confidential, so schema and attribute values have been withheld and replaced by meaningless symbols. "german" contains attributes such as "status of existing checking account," "credit history," "duration of present employment," etc. The task is to determine whether credit should be granted. Full details about the databases can be found in [43]. Note that only one of the databases has categorical attributes only ("soybean"), so the proposed algorithm can be applied directly. The other

Fig. 6. The trellis obtained by state-unfolding the network of Fig. 5. For simplicity, attribute nodes are not shown (see Fig. 7 for details).

level $h - 1$

level $h$

Fig. 7. Detailed topology of decision trellises for the example of Fig. 5. The unfolded network can be interpreted as a connectionist network with second-order connections.
databases contain numerical attributes. In these cases, we applied the entropy-based global discretization method described by Fayyad and Irani [44] to convert numerical attributes into categorical ones.

### 7.2 Selection of Model Complexity

The complexity of the model is controlled by the height \( H \). A good choice of \( H \) should take into account the number of jointly relevant attributes. In fact, each path in the trellis can be interpreted as a probabilistic conjunctive rule that involves all the attributes tested in \( \Omega_0, \ldots, \Omega_{H-1} \). If the
TABLE 2
Experimental Results on Some Databases from the Repository at the University of California at Irvine

<table>
<thead>
<tr>
<th>Dataset</th>
<th>size</th>
<th>N</th>
<th>C</th>
<th>Average trellis height</th>
<th>Accuracies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Trellis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Error reduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C4.5 Discretized</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Error reduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C4.5 Continuous</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>size</th>
<th>N</th>
<th>C</th>
<th>Average trellis height</th>
<th>Accuracies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Error reduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C4.5 Discretized</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Error reduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C4.5 Continuous</td>
</tr>
</tbody>
</table>

trellis is too short, longer rules cannot be represented. However, the amount of training data is also relevant to the choice of H since too high trellises may exhibit overfitting. In the experiments reported here, we employed a simple incremental algorithm that trades expressiveness for model simplicity. Starting with H = 2, a decision trellis is trained until EM reaches a local maximum of the likelihood function. Let \( e(H) \) be the empirical error (evaluated on the training set) obtained with maximum likelihood parameters. The algorithm proceeds by adding a new level \( H + 1 \) with random parameters and keeping the parameters associated with previous levels. The augmented model is then trained until convergence and the errors \( e(H + 1) \) and \( e(H) \) compared. If \( e(H + 1) < e(H) \), the process is iterated by increasing again the height. Otherwise, the trellis with height \( H \) is returned. The algorithm searches in a greedy way for a short trellis that can obtain good performance on the training data. Stopping the search as soon as an increment for \( H \) does not improve classification on the training set can be seen as a form of inductive bias to prevent overfitting.

7.3 Results
The decision trellis learning procedure was integrated into the MLC++ library by Kohavi et al. [45] in order to take advantage of the library routines for accuracy estimation and continuous attribute discretization. Accuracy for smaller databases (less than 1,000 instances) was estimated by a complete 10-fold cross-validation procedure. Accuracy for "sick" and "segment" databases was estimated by training on two-thirds of the available instances and testing on the remaining third (splits were randomly chosen to obtain a uniform class distribution).

In Table 2, we report the estimated accuracies for the decision trellis, compared to the estimated accuracy for decision trees induced by Quinlan's C4.5 [5]. Since C4.5 can deal with both nominal and continuous attributes, we measured both the performance obtained using entropy discretization (column "C4.5 Discretized") and using no discretization (column "C4.5 Continuous"). The relative error reduction in both cases is computed as \( (\text{err}_{\text{MLC}} - \text{err}_{\text{C4.5}})/(1 - \text{err}_{\text{C4.5}}) \). The second column reports the total number of instances in each dataset, while the number of attributes \( N \) and the number of classes \( C \) are reported in the next two columns. The fifth column is the average trellis height (as obtained from the algorithm described above) over the 10 cross-validation trials.

7.4 Experiments with Missing Data
In this set of experiments, we have tested the learning capabilities of decision trellises in the presence of missing data. Starting from the original datasets, we have randomly deleted a fixed amount of predictive attributes, in each training set and in each test set. Fig. 9 and Fig. 10 show the results obtained on the various datasets, where the accuracies are estimated by 10-fold cross-validation. As expected, the performance of both algorithms tends to decrease with the percentage of deleted attributes. However, it can be noted that C4.5 generally exhibits more severe degradation. For clean data (with no removed attributes), 4.5 outperforms the trellis in five out of the 16 datasets but, when a large enough portion of attributes is removed, the trellis outperforms C4.5 in almost every dataset (except in the case of the "glass" dataset, where the difference in performance between the two algorithms
is rather irregular). In Table 3, we compare the average performance of the two algorithms for a percentage of corruption ranging from 0 percent to 90 percent.

8 CONCLUSIONS

We have proposed decision trellises, a novel architecture for tuple categorization in databases. Experimental results on some standard machine learning databases are encouraging enough. In particular, we have obtained experimental evidence that decision trellises are more robust than decision trees when learning from sparse databases in which a significant portion of the predictive attributes are missing. When uncorrupted datasets are used, decision trellis outperforms decision trees in several domains, but there are still cases in which decision trees are better. This leaves room for further improvements.

By monitoring the generalization accuracy during the likelihood optimization procedure, we often noticed an overfitting phenomenon similar to the rather common behavior observed when training neural networks: After a number of EM iterations, the generalization accuracy begins
to decrease, while the accuracy on the training set increases. This behavior could be corrected by reserving a portion of the training data to monitor generalization accuracy. However, this approach wastes training examples and, in the case of small datasets, we were not able to obtain significant improvements. Using appropriate Bayesian priors or postpruning techniques could be more helpful to limit overfitting. The choice of $H$ appears to be important and, in the experiments, we observed that the accuracy may significantly change with $H$ for some datasets. This problem might be overcome by introducing some prepruning or postpruning techniques. Another improvement might come from the optimization of the local models that are in charge of predicting missing attributes. As it stands, the algorithm substitutes missing values with their unconditional expectation, as obtained from the training relation. Local attribute models, however, may be trained, for example, using the EM algorithm as done in [41]. Other important aspects to be investigated are the extraction of (probabilistic) rules from the trained trellis and the possibility of inserting prior knowledge. In the case of neural networks, for example, it has been demonstrated that using learning as a refinement
TABLE 3
Average Performances of C4.5 and Decision Trees in Experiments on Missing Data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>C4.5</th>
<th>O.4.5</th>
<th>Error reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>anneal</td>
<td>81.94</td>
<td>81.67</td>
<td>1.5</td>
</tr>
<tr>
<td>australiasian</td>
<td>76.61</td>
<td>72.68</td>
<td>14.4</td>
</tr>
<tr>
<td>breast</td>
<td>89.46</td>
<td>83.39</td>
<td>36.5</td>
</tr>
<tr>
<td>cleve</td>
<td>76.63</td>
<td>70.25</td>
<td>21.4</td>
</tr>
<tr>
<td>crx</td>
<td>78.36</td>
<td>73.55</td>
<td>18.2</td>
</tr>
<tr>
<td>diabetes</td>
<td>70.60</td>
<td>69.44</td>
<td>3.6</td>
</tr>
<tr>
<td>german</td>
<td>70.92</td>
<td>70.72</td>
<td>0.7</td>
</tr>
<tr>
<td>glass</td>
<td>47.90</td>
<td>48.65</td>
<td>-1.5</td>
</tr>
<tr>
<td>heart</td>
<td>75.44</td>
<td>71.78</td>
<td>13.0</td>
</tr>
<tr>
<td>horse colic</td>
<td>76.35</td>
<td>73.87</td>
<td>10.2</td>
</tr>
<tr>
<td>iris</td>
<td>80.27</td>
<td>74.87</td>
<td>21.5</td>
</tr>
<tr>
<td>pima</td>
<td>70.68</td>
<td>69.45</td>
<td>4.1</td>
</tr>
<tr>
<td>segment</td>
<td>75.71</td>
<td>66.51</td>
<td>27.5</td>
</tr>
<tr>
<td>sick</td>
<td>92.43</td>
<td>92.21</td>
<td>2.8</td>
</tr>
<tr>
<td>soybean</td>
<td>55.73</td>
<td>49.11</td>
<td>13.0</td>
</tr>
<tr>
<td>vehicle</td>
<td>54.95</td>
<td>50.53</td>
<td>9.3</td>
</tr>
</tbody>
</table>

of prior knowledge can significantly improve prediction accuracy [47]. In the present model, prior knowledge might be inserted in the form of Bayesian priors for the parameters and accompanied by maximum a posteriori estimation. Finally, a further improvement might be obtained by directly exploiting numerical attributes without having recourse to quantization. For example, we are attempting to extend attribute tests using log-linear models on groups of numerical attributes.

REFERENCES


Marco Gori received the Laurea degree in electronic engineering from Università di Firenze, Italy, in 1984; and the PhD degree from Università di Bologna, Italy, in 1990. From October 1988 to June 1989, he was a visiting student at the School of Computer Science, McGill University, Montreal, Canada. In 1992, he became an associate professor of computer science at Università di Firenze and, in November 1995, he joined the University of Siena, Italy. His main research interests are in pattern recognition (especially document processing) and artificial intelligence, emphasizing connectionist models. He was general chair of the Second Workshop on Neural Networks for Speech Processing, held in Firenze in 1992, organized the NIPS '96 postconference workshop on "Artificial Neural Networks and Continuous Optimization: Local Minima and Computational Complexity"; and co-organized the Calabro Summer School on "Adapting Processing of Sequences and Data Structures," held in Salerno, Italy, in September, 1997. He serves as a program committee member for several workshops and conferences, mainly in the area of connectionist models. He acted as guest co-editor of the Neurocomputing Journal special issue on recurrent neural networks (July 1997). He is an associate editor of IEEE Transactions on Neural Networks, Neurocomputing, and Neural Computing Survey. He is the Italian chair of the IEEE Neural Network Council (R.I.G.), a member of the executive board of the European Neural Network Society, and a senior member of the IEEE.

Giovanni Soda received his degree in mathematics from the University of Florence, Italy, in 1969. From 1971 to 1975, he was a researcher at the National Council of Research, where his activity included formal systems for language manipulation. Since 1975, he has been at the University of Florence, where he is presently an associate professor of artificial intelligence in the Department of Systems and Computer Science (DISI). His current research interests include knowledge representation systems, integration of artificial intelligence techniques with neural networks, and document processing. He was general chair of A*IA '95, held in Florence in 1995. He is a member of the steering committee of A*IA, the Italian Association of Artificial Intelligence; the IEEE; the ACM; and the IAPR Societies.

Paolo Frasconi received the MS degree in electronic engineering in 1989 and the PhD degree in computer science in 1994, both from the University of Florence, Italy. He was recently appointed an associate professor of computer science with the Department of Electrical and Electronic Engineering at the University of Genova, Italy. From 1990 to 1993 he was an assistant professor with the Department of Systems and Computer Science (DISI) at the University of Florence, Italy. He was a visiting scholar in the Department of Brain and Cognitive Science at the Massachusetts Institute of Technology in 1992; a visiting scientist at Centro Studi e Laboratori Telecommunicazioni (CSELT) in Turin, Italy, in 1994; and a visiting lecturer with the School of Information Technology and Computer Science at the University of Wollongong, Australia, in 1996. His current research interests include learning in neural networks, Markovian models, and belief networks, with particular emphasis on problems involving learning about sequential and structured information.