A Naive Bayes Style Possibilistic Classifier

Christian Borgelt and Jörg Gebhardt

Dept. of Knowledge Processing and Language Engineering
Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany

e-mail: borgelt@iws.cs.uni-magdeburg.de

Abstract: Naive Bayes classifiers can be seen as special probabilistic networks with a star-like structure. They can easily be induced from a dataset of sample cases. However, as most probabilistic approaches, they run into problems, if imprecise (i.e, set-valued) information in the data to learn from has to be taken into account. An approach to handle uncertain as well imprecise information, which recently gained some attention, are possibilistic networks. Because of the close structural resemblance of possibilistic networks to probabilistic ones, the idea suggests itself to construct a possibilistic classifier as a special possibilistic network in much the same way in which a naive Bayes classifier is a special probabilistic network. Thus we obtain a classifier that can easily handle imprecise information in the data to learn from.

1 Introduction

Naive Bayes classifiers [8, 5, 13] are an old and well-known type of classifiers. They use a probabilistic approach to assign a class to a case or an object and can be seen as a special type of probabilistic networks. Probabilistic networks, which have been studied extensively in the field of graphical modeling [11] and of which Bayesian networks [19] and Markov networks [15] are the most popular, are based on the idea to exploit independence relations between attributes used to describe a domain in order to decompose a multivariate probability distribution into a set of (conditional or marginal) distributions on lower dimensional subspaces. Early efficient implementations include HUGIN [1] and PATHFINDER [9]. Naive Bayes classifiers are a special case of probabilistic networks, since they make strong assumptions about the dependence and independence relations between the class attribute and the other attributes and thus represent a specific decomposition of a given multivariate probability distribution. Naive Bayes classifiers as well as the more general probabilistic networks can easily be induced from a dataset of sample cases [14, 4, 10].

A drawback of all probabilistic approaches is that they provide a framework to handle *uncertain*, but *precise* information. However, the explicit treatment of imprecise (i.e., set-valued) information is more and more claimed to be necessary for industrial practice. Therefore it is reasonable to investigate graphical models based on other uncertainty calculi, the more so, as extending purely probabilistic approaches to the treatment of imprecise information usually renders the corresponding inference mechanisms computationally intractable. A noteworthy attempt in this direction are the so-called valuation-based networks [21, 22] which have been implemented in PULCINELLA [20]. Another are possibilistic networks, which due to their connection to fuzzy systems and their ability to deal not only with uncertainty but also with imprecision, recently gained some attention. They have been implemented in POSSINFER [7, 12].

In this paper, we focus on the latter approach, that is, on possibilistic networks. We suggest a classifier that is a special case of a possibilistic network in much the same way in which a naive Bayes classifier is a special case of a probabilistic network. In section 2 we review the ideas underlying naive Bayes classifiers and relate them to Bayesian networks in section 3. In section 4 we review possibilistic networks and in section 5 we introduce our naive Bayes style possibilistic classifier. In section 6 we briefly discuss how naive Bayes as well as possibilistic classifiers can be simplified to improve their performance. In section 7 we report experimental results and in section 8 we draw conclusions and indicate future work.

2 Naive Bayes Classifiers

Classifiers are programs that assign a class from a predefined set to an object or case under consideration based on the values of attributes used to describe this object or case. To do so, naive Bayes classifiers use a probabilistic approach, i.e., they try to compute conditional class probabilities and then predict the most probable class. To be more precise, let C denote a class attribute with a finite domain of m classes, i.e.,

$\mathrm{dom}(C) = \{c_1, \ldots, c_m\}$, and let $A_1, \ldots, A_n$ be a set of other attributes used to describe a case or an object of the universe of discourse. These other attributes may be symbolic, i.e., $\mathrm{dom}(A_j) = \{a_1^{(j)}, \ldots, a_{m_j}^{(j)}\}$, or numeric, i.e., $\mathrm{dom}(A_j) = \mathbb{R}$. In this paper, however, we confine ourselves to symbolic attributes. With this restriction, a case or an object can be described by an instantiation $\omega = (a_{i_1}^{(1)}, \ldots, a_{i_n}^{(n)})$ of the attributes $A_1, \ldots, A_n$ and thus the universe of discourse is $\Omega = \mathrm{dom}(A_1) \times \ldots \times \mathrm{dom}(A_n)$.

For a given instantiation ω, a naive Bayes classifier tries to compute the conditional probability

$$P(C = c_i \mid \omega) = P(C = c_i \mid A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)})$$

for all c_i and then predicts the class for which this probability is highest. Of course, it is usually impossible to store all conditional probabilities explicitly, so that a simple lookup would be all that is needed to find the most probable class: We would have to store a class (or a class probability distribution) for each point of the universe of discourse, whose size grows exponentially with the number of attributes. To circumvent this problem, naive Bayes classifiers exploit—as their name already indicates—Bayes rule and a set of conditional independence assumptions. With Bayes rule

$$P(Y \mid X) = \frac{P(X \mid Y) \cdot P(Y)}{P(X)},$$

where X and Y are events, the conditional probabilities are inverted, i.e., naive Bayes classifiers consider[1]

$$\begin{aligned} &P(C = c_i \mid A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)}) \\ &= \frac{P(A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)} \mid C = c_i) \cdot P(C = c_i)}{P(A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)})} \end{aligned}$$

Of course, for this to be possible, the probability $P(A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)})$ must be strictly positive.

Note that we can neglect the denominator of the fraction on the right, since for a given case or object to be classified, it is fixed and therefore does not have any influence on the class ranking (which is all we are interested in). In addition, it can always be restored by a normalization, i.e., we can exploit

$$\begin{aligned} &P(A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)}) \\ &= \sum_{j=1}^{m} P(A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)} \mid C = c_j) \cdot P(C = c_j). \end{aligned}$$

It follows that we only need to consider

$$\begin{aligned} &P(C = c_i \mid A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)}) \\ &= \frac{1}{S} P(A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)} \mid C = c_i) \cdot P(C = c_i), \end{aligned}$$

where S is a normalization constant.

Furthermore, note that just inverting the probabilities does not buy us anything, since the probability space is just as large as it was before the inversion. However, here the second ingredient of naive Bayes classifiers—which is responsible for the "naive" in their name—comes in, namely the conditional independence assumptions. To exploit them, we first apply the chain rule of probability:

$$\begin{aligned} &P(C = c_i \mid A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)}) \\ &= \frac{1}{S} \cdot P(A_n = a_{i_n}^{(n)} \mid A_{n-1} = a_{i_{n-1}}^{(n-1)}, \ldots, \\ &\qquad\qquad A_1 = a_{i_1}^{(1)}, C = c_i) \\ &\qquad \cdots \\ &\qquad \cdot P(A_2 = a_{i_2}^{(2)} \mid A_1 = a_{i_1}^{(1)}, C = c_i) \\ &\qquad \cdot P(A_1 = a_{i_1}^{(1)} \mid C = c_i) \\ &\qquad \cdot P(C = c_i). \end{aligned}$$

Then we assume that given the value of the class attribute, any attribute A_j is independent of any other. That is, we assume that knowing the class is enough to determine the probability of a value $a_{i_j}^{(j)}$, i.e., that we need not know the values of any other attributes. This assumption considerably simplifies the formula stated above, since with it we can cancel all attributes A_j appearing in the conditions:

$$\begin{aligned} &P(C = c_i \mid A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)}) \\ &= \frac{1}{S} \cdot P(A_1 = a_{i_1}^{(1)} \mid C = c_i) \\ &\qquad \cdot P(A_2 = a_{i_2}^{(2)} \mid C = c_i) \\ &\qquad \cdots \\ &\qquad \cdot P(A_n = a_{i_n}^{(n)} \mid C = c_i) \\ &\qquad \cdot P(C = c_i). \end{aligned}$$

This is the fundamental formula underlying naive Bayes classifiers. For a symbolic attribute A_j the conditional probabilities $P(A_j = a_{i_j}^{(j)} \mid C = c_i)$ are stored as a simple conditional probability table. This is feasible now, since there is only one condition and hence only $m \cdot m_j$ probabilities have to be stored.[2]

Naive Bayes classifiers can easily be induced from a dataset of preclassified sample cases. All we have to do is to estimate the conditional probabilities $P(A_j = a_{i_j}^{(j)} \mid C = c_i)$. We may use, for instance, maximum likelihood estimation, which yields

$$\hat{P}(A_j = a_{i_j}^{(j)} \mid C = c_i) = \frac{\#(A_j = a_{i_j}^{(j)}, C = c_i)}{\#(C = c_i)},$$

[1] Note that we can always use a probability P, since we confined ourselves to symbolic attributes.

[2] Actually only $m \cdot (m_j - 1)$ probabilities are really necessary. Since the probabilities have to add up to one, one value can be discarded from each conditional distribution. However, in implementations it is usually much easier to store all probabilities.

where $\#(C = c_i)$ is the number of sample cases that belong to the class c_i and $\#(A_j = a_{i_j}^{(j)}, C = c_i)$ is the number of sample cases belonging to class c_i and having the value $a_{i_j}^{(j)}$ for the attribute A_j. To ensure that the probability is strictly positive, it is assumed that there is at least one example for each class in the dataset. Otherwise the class is simply removed from the domain of the class attribute. If an attribute value does not occur given some class, its probability is either set to $\frac{1}{2N}$, where N is the number of sample cases, or a uniform prior of $\frac{1}{N}$ is added to the estimated distribution, which is then renormalized (Laplace correction).

3 Probabilistic Networks

As already mentioned in the introduction, naive Bayes classifiers are a special case of probabilistic networks or, to be more specific, of Bayesian networks. A Bayesian network is a directed acyclic graph in which each node represents an attribute (interpreted as a random variable), that is used to describe some domain of interest, and each edge represents a direct dependence between two attributes. In addition, the graph represents a particular joint probability distribution, which is specified by assigning to each node in the network a (conditional) probability distribution for the values of the corresponding attribute given the parent attributes in the network (if any).

Formally, a Bayesian network describes a factorization of a multivariate probability distribution. This factorization results from applying first the chain rule of probability to the joint distribution. Then the factors are simplified by exploiting *conditional independence statements* of the form $\forall \omega \in \Omega$:

$$P(\omega_{X \cup Y} \mid \omega_Z) = P(\omega_X \mid \omega_Z) \cdot P(\omega_Y \mid \omega_Z)$$

whenever $P(\omega_Z) > 0$, where X, Y, and Z are three disjoint sets of attributes and $\omega_X = \mathrm{proj}_X(\omega)$ is the projection of an instantiation ω to the attributes in X. As one can easily verify, these statements are equivalent to statements of the form $\forall \omega \in \Omega$:

$$P(\omega_X \mid \omega_{Y \cup Z}) = P(\omega_X \mid \omega_Z).$$

To be more precise, consider a probability distribution P on the joint domain of a set of attributes $A_1, \dots A_n$ (again we assume that all attributes are symbolic). We first apply the chain rule of probability to obtain

$$\begin{aligned}
&P(A_1 = a_{i_1}^{(1)}, \dots, A_n = a_{i_n}^{(n)}) \\
&\quad = P(A_n = a_{i_n}^{(n)} \mid A_{n-1} = a_{i_{n-1}}^{(n-1)}, \dots, A_1 = a_{i_1}^{(1)}) \\
&\quad \cdot\ P(A_{n-1} = a_{i_{n-1}}^{(n-1)} \mid A_{n-2} = a_{i_{n-2}}^{(n-2)}, \dots, A_1 = a_{i_1}^{(1)}) \\
&\qquad \cdots \\
&\quad \cdot\ P(A_2 = a_{i_2}^{(2)} \mid A_1 = a_{i_1}^{(1)}) \\
&\quad \cdot\ P(A_1 = a_{i_1}^{(1)}).
\end{aligned}$$

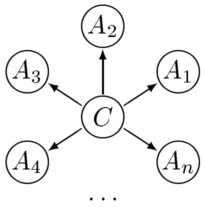

Figure 1: A naive Bayes classifier is a Bayesian network with a star-like structure.

Then we exploit conditional independence statements to simplify the conditions by removing those attributes of which the conditioned attribute is independent given the values of the remaining attributes. Thus the joint probability distribution can be computed as

$$P(A_1, \dots, A_n) = \prod_{j=1}^{n} P(A_j \mid \mathrm{parents}(A_j)),$$

where $\mathrm{parents}(A_j)$ is the set of attributes that is sufficient to determine the probability (density) of the values of attribute A_j. The name "$\mathrm{parents}(A_j)$" stems from the fact that in the Bayesian network the conditioning attributes are connected by directed edges to the conditioned attributes and hence are the parents of this attribute in the graph. This makes it very simple to read the factorization formula from a Bayesian network: For each attribute (node) there is exactly one factor in which it is the conditioned attribute, and the conditions of this factor are the attributes corresponding to the attribute's parent nodes in the graph.

It is easy to see that Bayesian networks are directly related to naive Bayes classifiers. In fact, a naive Bayes classifier is a Bayesian network with a star-like structure as shown in figure 1. That is, there is a distinguished attribute, namely the class attribute. It is the only unconditioned attribute (the only one without parents). All other attributes are conditioned on the class attribute and on the class attribute only. Reasoning consists in propagating the evidence about the values of the attributes $A_1, \dots, A_n$ along the edges to obtain information about the class. This information is then accumulated.

4 Possibilistic Networks

As already indicated in the introduction, the development of possibilistic networks was triggered by the fact that probabilistic networks are well suited to represent and process *uncertain* information, but cannot easily be extended to handle *imprecise* information. Maybe the best way to explain the difference between uncertain and imprecise information is to consider the notion of a degree of possibility. The interpretation we prefer is based on the context model [6, 12]. In this model possibility distributions are seen as information-compressed representations of (not necessarily nested) random sets and a degree of possibility as the one-point coverage of a random set [18].

To be more precise: Let ω_0 be the actual, but unknown state of a domain of interest, which is contained in a set Ω of possible states. Let $(T, 2^T, P)$, $T = \{t_1, t_2, \ldots, t_r\}$, be a finite probability space and $\gamma : T \to 2^\Omega$ a set-valued mapping. T is seen as a set of contexts that have to be distinguished for a set-valued specification of ω_0. The contexts are supposed to describe different physical and observation-related frame conditions. $P(\{t\})$ is the (subjective) probability of the (occurrence or selection of the) context t.

A set $\gamma(t)$ is assumed to be the *most specific correct set-valued specification* of ω_0, which is implied by the frame conditions that characterize the context t. By 'most specific set-valued specification' we mean that $\omega_0 \in \gamma(t)$ is guaranteed to be true for $\gamma(t)$, but is not guaranteed for any proper subset of $\gamma(t)$. The resulting *random set* $\Gamma = (\gamma, P)$ is an imperfect (i.e. imprecise *and* uncertain) specification of ω_0. Let π_Γ denote the *one-point coverage of* Γ (the *possibility distribution induced by* Γ*)*, which is defined as

$$\pi_\Gamma : \Omega \to [0, 1], \quad \pi_\Gamma(\omega) = P\left(\{t \in T \mid \omega \in \gamma(t)\}\right).$$

In a complete modeling the contexts in T must be specified in detail, so that the relationships between all contexts t_j and their corresponding specifications $\gamma(t_j)$ are made explicit. But if the contexts are unknown or ignored, then $\pi_\Gamma(\omega)$ is the total mass of all contexts c that provide a specification $\gamma(t)$ in which ω_0 is contained, and this quantifies the *possibility of truth* of the statement "$\omega = \omega_0$" [6].

It is easy to see that a possibility distribution induced by a random set Γ degenerates to a probability distribution (namely the one on the set of contexts), if in each context t there is only one possible value, i.e., if $\forall t \in T : |\gamma(t)| = 1$. This is what we meant by saying that probabilistic approaches can handle *uncertain*, but *precise* information.

As a concept of independence, which is fundamental to the technique of graphical modeling, we use *possibilistic non-interactivity* [2]. Let X, Y, and Z be three disjoint subsets of attributes. Then X is called *conditionally independent* of Y given Z w.r.t. π, if $\forall \omega \in \Omega$:

$$\pi(\omega_{X \cup Y} \mid \omega_Z) = \min\{\pi(\omega_X \mid \omega_Z), \pi(\omega_Y \mid \omega_Z)\}$$

whenever $\pi(\omega_Z) > 0$, where $\omega_X = \mathrm{proj}_X(\omega)$ is the projection of a tuple ω to the attributes in X and $\pi(\cdot \mid \cdot)$ is a non-normalized conditional possibility distribution

$$\pi(\omega_X \mid \omega_Z) = \max\{\pi(\omega) \mid \omega \in \Omega \wedge \mathrm{proj}\,_X(\omega) = \omega_X \wedge \mathrm{proj}\,_Z(\omega) = \omega_Z\}.$$

A *possibilistic network* represents a decomposition of a multi-variate possibility distribution according to

$$\pi(A_1, \ldots, A_n) = \min{}_{j=1}^{n} \pi(A_j \mid \mathrm{parents}(A_j)),$$

where $\mathrm{parents}(A_j)$ is the set of parents of attribute A_j, which is made as small as possible by exploiting conditional independences of the type indicated above. Just as a Bayesian network, a possibilistic network is usually represented as a directed graph in which there is an edge from each of the conditioning attributes to the conditioned attribute.

5 A Possibilistic Classifier

Due to the structural equivalence of probabilistic and possibilistic networks, a naive Bayes style possibilistic classifier can be constructed in strict analogy to the probabilistic case: Let π be a possibility distribution on the attributes $A_1, \ldots, A_n$ and C. Because of the symmetry in the definition of conditional possibility distributions, we have

$$\begin{aligned} &\pi(C = c_i \mid A_1 = a^{(1)}_{i_1}, \ldots, A_n = a^{(n)}_{i_n}) \\ &= \pi(A_1 = a^{(1)}_{i_1}, \ldots, A_n = a^{(n)}_{i_n} \mid C = c_i). \end{aligned}$$

This equation takes the place of Bayes rule. It has the advantage of being much simpler than Bayes rule, and thus we need not take account of a normalization constant or of prior class probabilities.

The next step is that, in analogy to the probabilistic conditional independence assumptions, we assume that given the value of the class attribute all other attributes do not interact possibilistically. With this assumption we arrive at

$$\begin{aligned} &\pi(C = c_i \mid A_1 = a^{(1)}_{i_1}, \ldots, A_n = a^{(n)}_{i_n}) \\ &= \min\{\pi(A_1 = a^{(1)}_{i_1} \mid C = c_i), \\ &\qquad \pi(A_2 = a^{(2)}_{i_2} \mid C = c_i), \\ &\qquad \ldots \\ &\qquad \pi(A_n = a^{(n)}_{i_n} \mid C = c_i)\}. \end{aligned}$$

This is the fundamental equation underlying our possibilistic classifier. Given an instantiation $\omega = (a^{(1)}_{i_1}, \ldots, a^{(n)}_{i_n})$ we predict the class c_i for which this equation yields the highest (conditional) degree of possibility. It is obvious that, just as a naive Bayes classifier is a special Bayesian network, our possibilistic classifier is a special possibilistic network and, just as a naive Bayes classifier, it has a star-like structure.

To induce our possibilistic classifier from data, we must estimate the conditional possibility distributions. This is not as simple as it may appear at first sight, but poses some problems of efficiency. In contrast to conditional probabilities, which can easily be computed from case counters (see above), maximum projections are much harder to obtain. However, in [3] we suggested a solution to this problem. This solution has proven to be efficient for many applications (although pathological cases can be constructed). It is based on

dataset		num. of tuples	possibilistic classifier		naive Bayes classifier		decision tree	
			add. att.	rem. att.	add. att.	rem. att.	unpruned	pruned
audio	train	113	7(6.2%)	2(1.8%)	12(10.6%)	16(14.2%)	13(11.5%)	16(14.2%)
	test	113	33(29.2%)	36(31.9%)	35(31.0%)	31(27.4%)	25(22.1%)	25(22.1%)
69 atts.	selected		15	21	9	42	14	12
bridges	train	54	8(14.8%)	8(14.8%)	10(18.5%)	7(13.0%)	9(16.7%)	9(16.7%)
	test	54	23(42.6%)	23(42.6%)	24(44.4%)	19(35.2%)	24(44.4%)	24(44.4%)
10 atts.	selected		6	6	5	8	8	6
soybean	train	342	18(5.3%)	20(5.9%)	17(5.0%)	14(4.1%)	16(4.7%)	22(6.4%)
	test	341	59(17.3%)	57(16.7%)	48(14.1%)	45(13.2%)	47(13.8%)	39(11.4%)
36 atts.	selected		15	17	14	14	19	16
vote	train	300	9(3.0%)	8(2.7%)	9(3.0%)	8(2.7%)	6(2.0%)	7(2.3%)
	test	135	11(8.2%)	10(7.4%)	11(8.2%)	8(5.9%)	11(8.2%)	8(5.9%)
16 atts.	selected		2	3	2	4	6	4

Table 1: Experimental results on four datasets from the UCI machine learning repository.

the observation that the problems encountered when one tries to compute maximum projections are mainly due to the fact two imprecise tuples over the attributes used to describe the domain of interest can intersect and that this intersection may differ from both original tuples. If such intersections exist, simple methods to compute maximum projections fail. But if the *closure under tuple intersection* is computed for the dataset to learn from (that is, if all tuples that can be constructed by intersecting a subset of tuples from the dataset are added to the dataset) and if the tuples in this closure are properly weighted, then maximum projections can be computed by simply determining maxima [3].

6 Classifier Simplification

Both naive Bayes classifiers and naive Bayes style possibilistic classifiers make strong independence assumptions (see above). It is not surprising that these assumptions are likely to fail. If they fail, the classifier may be worse than necessary. To cope with this problem, we tried to simplify the classifiers, naive Bayes as well as possibilistic, using simple greedy attribute selection. With this procedure we hoped to find a (sub)set of attributes for which the strong assumptions hold at least approximately. The experimental results we report in the next section prove that this approach seems to be successful.

The attribute selection methods we used are the following: In the first method we start with a classifier that simply predicts the majority class. That is, we start with a classifier that does not use any attribute information. Then we add attributes one by one. In each step we select that attribute which, if added, leads to the smallest number of misclassifications on the training data. We stop adding attributes when adding any of the remaining attributes does not reduce the number of errors.

The second method is a reversal of the first. We start with a classifier that takes into account all available attributes and then removes attributes step by step. In each step we select that attribute which, if removed, leads to the smallest number of misclassifications on the training data. We stop removing attributes when removing any of the remaining attributes leads to a larger number of errors.

7 Experimental Results

We implemented the suggested possibilistic classifier along with a normal naive Bayes classifier and tested both on four datasets from the UC Irvine machine learning repository [16]. In both cases we used the simplification procedures described in the preceding section. The results are shown in table 1, together with the results obtained with a decision tree classifier. The columns "add att." contain the results obtained by stepwise adding, the columns "rem. att." the results obtained by removing attributes. The decision tree program used is a C4.5 clone written by the first author. The attribute selection measure was information gain ratio and the pruning method was confidence level pruning with a confidence level of 50%.

It can be seen that the possibilistic classifier performs equally well as (audio, bridges, vote) or only slightly worse (soybean) than the naive Bayes classifier. This is encouraging, since none of the datasets is especially suited to demonstrate the strengths of a possibilistic classifier. Although all of the datasets contain missing values (which can be modeled as imprecise information), the relative frequency of these missing values is rather low. None of the datasets contains true set valued information, which to treat possibility theory is so well designed.

8 Conclusions and Future Work

In this paper we suggested a possibilistic classifier that is a special case of a possibilistic network in much the same way in which a naive Bayes classifier is a special case of a probabilistic network. Our experimental results show that it can hold its own against a naive Bayes classifier, even though the datasets we used to induce the classifiers do not possess the properties (set valued information) which could demonstrate the strengths of a possibilistic approach. Therefore we are confident that it will prove successful when applied to true imprecise data.

In the future we plan to extend our possibilistic classifier to handle numeric attributes. The main problem with such an extension is that it is not entirely clear how to estimate a possibility distribution on a continuous domain from a dataset of sample cases, if some cases possess precise values for the attribute under consideration (strange enough, for possibilistic approaches *precision* poses a problem). A simple idea would be to fix the size of a small interval to be used in such cases.

In another line of research we plan to explore the connections of our possibilistic classifier to fuzzy classifiers like those generated by neuro-fuzzy systems like NEFCLASS [17]. The connection to such classifiers is that our possibilistic classifier can be seen as fuzzy classifier with min-max-inference and one rule per class.

References

[1] S.K. Andersen, K.G. Olesen, F.V. Jensen, and F. Jensen. HUGIN — A shell for building Bayesian belief universes for expert systems. *Proc. 11th Int. J. Conf. on Artificial Intelligence*, 1080–1085, 1989

[2] C. Borgelt, J. Gebhardt, and R. Kruse. Chapter F1.2: Inference Methods. In: E. Ruspini, P. Bonissone, and W. Pedrycz, eds. *Handbook of Fuzzy Computation.* Institute of Physics Publishing Ltd., Bristol, United Kingdom 1998

[3] C. Borgelt and R. Kruse. Efficient Maximum Projection of Database-Induced Multivariate Possibility Distributions. *Proc. 7th IEEE Int. Conf. on Fuzzy Systems*, Anchorage, Alaska, USA 1998

[4] G.F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning* 9:309–347, Kluwer 1992

[5] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* J. Wiley & Sons, New York, NY, USA 1973

[6] J. Gebhardt and R. Kruse. The context model — an integrating view of vagueness and uncertainty *Int. Journal of Approximate Reasoning* 9:283–314, 1993

[7] J. Gebhardt and R. Kruse. POSSINFER — A Software Tool for Possibilistic Inference. In: D. Dubois, H. Prade, and R. Yager, eds. *Fuzzy Set Methods in Information Engineering: A Guided Tour of Applications*, Wiley 1995

[8] I.J. Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.* MIT Press, MA, USA, 1965

[9] D. Heckerman. *Probabilistic Similarity Networks.* MIT Press 1991

[10] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* 20:197–243, Kluwer 1995

[11] R. Kruse, E. Schwecke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge-based Systems: Numerical Methods.* Series: Artificial Intelligence, Springer, Berlin 1991

[12] R. Kruse, J. Gebhardt, and F. Klawonn. *Foundations of Fuzzy Systems*, John Wiley & Sons, Chichester, England 1994

[13] P. Langley, W. Iba, and K. Thompson. An Analysis of Bayesian Classifiers. *Proc. 10th Nat. Conf. on Artificial Intelligence*, 223–228, AAAI Press and MIT Press, USA 1992

[14] P. Langley and S. Sage. Induction of Selective Bayesian Classifiers. *Proc. 10th Conf. on Artificial Intelligence*, 1994

[15] S.L. Lauritzen and D.J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society, Series B*, 2(50):157–224, 1988

[16] P.M. Murphy and D. Aha, UCI Repository of Machine Learning Databases, ftp from ics.uci.edu, directory pub/machine-learning-databases, 1994

[17] D. Nauck and R. Kruse. A Neuro-Fuzzy Method to Learn Fuzzy Classification Rules from Data. *Fuzzy Sets and Systems* 89:277–288, 1997

[18] H.T. Nguyen. Using Random Sets. *Information Science* 34:265–274, 1984

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (2nd edition).* Morgan Kaufman, New York 1992

[20] A. Saffiotti and E. Umkehrer. PULCINELLA: A General Tool for Propagating Uncertainty in Valuation Networks. *Proc. 7th Conf. on Uncertainty in AI*, 323–331, San Mateo 1991

[21] G. Shafer and P.P. Shenoy. Local Computations in Hypertrees. Working Paper 201, School of Business, University of Kansas, Lawrence 1988

[22] P.P. Shenoy. Valuation-based Systems: A Framework for Managing Uncertainty in Expert Systems. Working Paper 226, School of Business, University of Kansas, Lawrence, 1991