Probabilistic Networks and Fuzzy Clustering as Generalizations of Naive Bayes Classifiers

Christian Borgelt, Heiko Timm, and Rudolf Kruse

Dept. of Knowledge Processing and Language Engineering
Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
e-mail: {borgelt,htimm,kruse}@iws.cs.uni-magdeburg.de

Abstract. Although at first sight probabilistic networks and fuzzy clustering seem to be disparate areas of research, a closer look reveals that they can both be seen as generalizations of naive Bayes classifiers. If all attributes are numeric (except the class attribute, of course), naive Bayes classifiers often assume an axis-parallel multidimensional normal distribution for each class as the underlying model. Probabilistic networks remove the requirement that the distributions must be axis-parallel by taking the covariance of the attributes into account, where this is necessary. Fuzzy clustering is an unsupervised method that tries to find general or axis-parallel distributions to cluster the data. Although it does not take into account the class information, it can be used to improve the result of naive Bayes classifiers and probabilistic networks by removing the restriction that there can be only one distribution per class.

1 Introduction

Probabilistic networks are a method to decompose a multivariate probability distribution in order to make reasoning in multi-dimensional domains feasible. Fuzzy clustering is a method to find groups of similar objects or cases, which compared to classical (crisp) clustering has the advantage that an object or a case can belong (with a degree between 0 and 1) to more than one cluster. Thus, at first sight, there seems to be little connection between these two methods. Nevertheless, in this paper we venture to discuss them together, since they can both be seen as generalizations of naive Bayes classifiers.

Our rationale is that the three techniques mentioned above—naive Bayes classifiers, probabilistic networks, and fuzzy clustering—share the idea that underlying the dataset to process there is a model consisting of a set of probability distributions/density functions that generated the data. They differ w.r.t. the assumptions they make about the distributions/density functions and whether they take into account the value of a distinguished class attribute (supervised methods: naive Bayes classifiers, probabilistic networks) or not (unsupervised methods: fuzzy clustering). Of course, there are still other methods, for example, radial basis function neural networks [19], that can be interpreted in much the same fashion. However, a complete list of such methods and a discussion of their similarities and differences is beyond the scope of this paper. We selected the three methods mentioned above as examples, because the first two (naive Bayes classifiers, probabilistic networks) show very clearly the properties we are interested in and because the connection to fuzzy clustering points out interesting directions to improve these techniques.

To simplify the explanation of the ideas this paper tries to convey, we confine ourselves to numeric attributes (with the exception of the class attribute, of course). That is, we consider only those attributes used to characterize an object or case under consideration that can be described by real numbers. With this restriction a common assumption, that is made with all three methods we are going to discuss, is that the data to process was generated by a set of multidimensional normal distributions (also called *Gaussians*). The three methods differ in the constraints they place on this set of distributions. Naive Bayes classifiers and probabilistic networks (if the latter are used for classification tasks) restrict the number of distributions to the number of classes, since they assume exactly one distribution per class. Naive Bayes classifiers, in addition, assume that for each multivariate normal distribution, i.e., for each class, the attributes are independent, thus requiring the distributions to be axis-parallel. Of fuzzy clustering algorithms, there are also general and axis-parallel variants. In fuzzy clustering, however, the number of multivariate normal distributions is not restricted to the number of classes (it is an unsupervised method and does not take the class information into account), but can be chosen freely. This often leads to a better fit to the data and may be exploited to improve the two other methods.

The brief overview just given already fixes the order in which we discuss the methods. In section 2 we examine naive Bayes classifiers. In section 3 we turn to probabilistic networks and show how a naive Bayes classifier can be seen as a special Bayesian network and how general Bayesian networks remove the strong independence assumptions underlying naive Bayes classifiers. In section 4 we study fuzzy clustering algorithms from the specific point of view indicated above (as estimators of an underlying set of probability distributions/density functions) and show how they can improve the aforementioned methods by removing the restriction of one distribution per class. Finally, in section 5, we draw conclusions from our discussion.

2 Naive Bayes Classifiers

Naive Bayes classifiers [11, 6, 16, 17] are an old and well-known type of classifiers, i.e., of 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. They do so using 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 domain under consideration. These other attributes may be symbolic, i.e., $\mathrm{dom}(A_j) = \{a_1^{(j)}, \ldots, a_{m_j}^{(j)}\}$, or numeric, i.e., $\mathrm{dom}(A_j) = \mathbb{R}$. For simplicity, we always use the notation $a_{i_j}^{(j)}$ for a value of an attribute A_j, independent of whether it is a symbolic or a numeric one.[1] With this notation, 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 c_i for which this probability is highest. Of course, it is usually impossible to store all of these conditional probabilities explicitly, so that a simple lookup would be all that is needed to find the most probable class. If there are numeric attributes, this is obvious (we need some parameterized function then). But even if all attributes are symbolic, such an approach most often is infeasible: We would have to store a class (or a class probability distribution) for each point of the Cartesian product of the attribute domains, 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. That is, naive Bayes classifiers consider[2]

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

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

There are two observations to be made about the inversion carried out above. In the first place, 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

[1] To be able to use this notation for numeric attributes, we simply have to choose an appropriate uncountably infinite index set $\mathcal{I}_j$, from which the index i_j is to be taken.

[2] For simplicity, we always use a probability density function f, although this is strictly correct only, if there is at least one numeric attribute. If all attributes are symbolic, this should be a probability P. The only exception is the class attribute, since it necessarily has a finite domain.

addition, its influence can always be restored by normalizing the distribution on the classes, i.e., we can exploit

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

It follows that we only need to consider

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

where S is a normalization constant.[3]

Secondly, we can see 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:[4]

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

Now we make the crucial assumption 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 (density) for a value $a^{(j)}_{i_j}$, i.e., that we need not know the values of any other attributes. Of course, this is a pretty strong assumption, which is very likely to fail. It is truly "naive" to make it nevertheless. However, it considerably simplifies the formula stated above, since with it we can cancel all attributes A_j appearing in the conditions:

$$P(C = c_i \mid A_1 = a^{(1)}_{i_1}, \ldots, A_n = a^{(n)}_{i_n}) = \frac{1}{S} \cdot P(C = c_i) \cdot \prod_{j=1}^{n} f(A_j = a^{(j)}_{i_j} \mid C = c_i)$$

[3] Strictly speaking, the constant S depends on the instantiation $(a^{(1)}_{i_1}, \ldots, a^{(n)}_{i_n})$. However, as already said above, when classifying a given case or object, this instantiation is fixed and hence we need to consider only one value S.

[4] Again we always use a probability density function f, although this is strictly correct only, if the conditioned attribute is numeric.

This is the fundamental formula underlying naive Bayes classifiers. For a symbolic attribute A_j the conditional probabilities $P(A_j = a^{(j)}_{i_j} \mid C = c_i)$ 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.[5] For numeric attributes it is usually assumed that the probability density is a Gaussian function (a normal distribution) and hence only the expected values $\mu_j(c_i)$ and the variances $\sigma^2_j(c_i)$ need to be stored in this case. Alternatively, numeric attributes may be discretized [5] and then treated like symbolic attributes. In this paper, however, we make the normal distribution assumption, since we need it for the connection to fuzzy clustering.

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/probability densities $f(A_j = a^{(j)}_{i_j} \mid C = c_i)$ using, for instance, maximum likelihood estimation. For symbolic attributes, this yields

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

where $\#(C = c_i)$ is the number of sample cases that belong to the class c_i and $\#(A_j = a^{(j)}_{i_j}, C = c_i)$ is the number of sample cases that belong to class c_i and have the value $a^{(j)}_{i_j}$ for the attribute A_j. To ensure that the probability is strictly positive (see above), 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). For a numeric attribute A_j the standard maximum likelihood estimation functions

$$\hat{\mu}_j(c_i) = \frac{1}{\#(C = c_i)} \sum_{k=1}^{\#(C=c_i)} a^{(j)}_{i_j(k)}$$

for the expected value, where $a^{(j)}_{i_j(k)}$ is the value of the attribute A_j in the k-th sample case belonging to class c_i, and

$$\hat{\sigma}^2_j(c_i) = \frac{1}{\#(C = c_i)} \sum_{k=1}^{\#(C=c_i)} \left(a^{(j)}_{i_j(k)} - \hat{\mu}_j(c_i) \right)^2$$

for the variance can be used.

[5] 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.

3 Probabilistic Networks

Probabilistic inference networks—especially Bayesian networks [20], but also Markov networks [18]—are well-known tools for reasoning under uncertainty in multidimensional domains. The idea underlying them is to exploit independence relations between the 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 [12].

Dependence and independence relations have been studied extensively in the field of graphical modeling [14, 24] and though using them to facilitate reasoning in multidimensional domains has originated in the probabilistic setting, this approach has been generalized to be usable with other uncertainty calculi [22], for instance, in the so-called valuation-based networks [23] and has been implemented, for example, in PULCINELLA [21]. Due to their connection to fuzzy systems, which in the past have successfully been applied to solve control problems, and due to their ability to deal not only with uncertainty but also with imprecision, recently possibilistic networks also gained some attention. They can be based on the context-model interpretation of a degree of possibility, which focuses on imprecision [9], and have been implemented, for example, in POSSINFER [10, 15]. In this paper, however, we focus on Bayesian networks, since they are closest to naive Bayes classifiers and thus to fuzzy clustering.

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. The structure of the directed graph encodes a set of conditional independence statements that can be read from the graph using a graph theoretic criterion called *d-separation* [20]. 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 its parent attributes in the network (if any).

Formally, a Bayesian network describes a factorization of a multivariate probability distribution/density function. This factorization results from applying first the chain rule of probability to the joint distribution/density function. 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 $\omega = (A_1 = a_{i_1}^{(1)}, \ldots, A_n = a_{i_n}^{(n)})$ to the attributes in X. As one can easily verify, these statements are equivalent to statements of the form $\forall \omega \in \Omega$:

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

From the description given up to now one can already guess the connection to naive Bayes classifiers. To be more precise, consider a probability distribution/density function f on the joint domain of a set of attributes $A_1, \dots A_n$. We first apply the chain rule of probability to obtain (we use the same notation as in the preceding section):

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

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 distribution/density function can be computed from $\forall i_1, \dots, i_n$:

$$f(\omega) = f(A_1 = a^{(1)}_{i_1}, \dots, A_n = a^{(n)}_{i_n}) = \prod_{i=1}^{n} P(A_j = a^{(j)}_{i_j} \mid \omega_{\text{parents}(A_i)}),$$

where parents(A_j) is the set of attributes of which to know the instantiations is sufficient to determine the probability (density) of the values of attribute A_j. The name "parents(A_j)" stems from the fact that in a 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. An example is shown on the left in figure 1, which represents the factorization $\forall i_1, \dots, i_7$:

$$\begin{aligned}
&f(A_1 = a^{(1)}_{i_1}, \dots, A_7 = a^{(7)}_{i_7}) \\
&= f(A_1 = a^{(1)}_{i_1}) \cdot f(A_2 = a^{(2)}_{i_2} \mid A_1 = a^{(1)}_{i_1}) \cdot f(A_3 = a^{(3)}_{i_3}) \\
&\cdot\ f(A_4 = a^{(4)}_{i_4} \mid A_1 = a^{(1)}_{i_1}, A_2 = a^{(2)}_{i_2}) \cdot f(A_5 = a^{(5)}_{i_5} \mid A_2 = a^{(2)}_{i_2}, A_3 = a^{(3)}_{i_3}) \\
&\cdot\ f(A_6 = a^{(6)}_{i_6} \mid A_4 = a^{(4)}_{i_4}, A_5 = a^{(5)}_{i_5}) \cdot f(A_7 = a^{(7)}_{i_7} \mid A_5 = a^{(5)}_{i_5})
\end{aligned}$$

It is obvious that a sparse graph is desirable to obtain a factorization with "small" factors. Whether a sparse graph can be found sometimes depends on the order of the attributes, but it cannot be guaranteed that a sparse graph exists for a given domain. In such cases usually an approximation is accepted.

Bayesian networks can be used for probabilistic reasoning by fixing the values of some (observed) attributes and then propagating this information in the

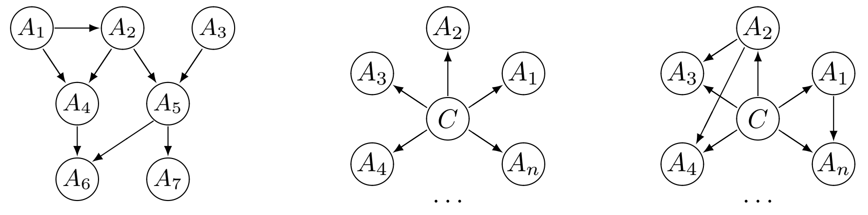

Fig. 1. A simple Bayesian network on a domain consisting of seven attributes (left). A naive Bayes classifier is a Bayesian network with a star-like structure (middle). It can easily be extended by adding edges between attributes that are still dependent given the class (right).

network to obtain the probabilities/densities for values of other (unobserved) attributes. This process, which is usually called evidence propagation, basically consists in replacing the prior probability distribution/density function with the posterior one, that is, the one conditioned on the values of the observed attributes. To make it efficient, a Bayesian network is often transformed into a clique tree for which a simple propagation scheme exists. The evidence is propagated along the edges of this clique tree using the marginal probability distributions/density functions associated with the nodes that represent the cliques. For details on clique tree construction and the clique tree propagation (CTP) algorithm, see e.g. [18].

It is easy to see that Bayesian networks are directly related to naive Bayes classifiers. In fact, a naive Bayes classifier is just a special Bayesian network with a star-like structure as shown in the middle of 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.

The main drawback of naive Bayes classifiers are the very strong conditional independence assumptions underlying them (see above). Although these assumptions necessarily lead to sparse graph, a lot of information can get lost. Fortunately, exploiting the more general approach underlying Bayesian networks, this severe constraint can be relaxed. That is, we may add edges between those of the attributes $A_1, \dots, A_j$ which are still dependent given the class (see figure 1 on the right). This can lead to improved classification results, since the extended conditional probability distributions are better suited to capture the dependence structure of the domain. To keep the resulting graph sparse, one may introduce the restriction that no attribute may have more than a fixed number of parents. Probabilistic networks of this type have been successfully applied in telecommunication [7].

As an illustrative example, let us take a look at the well-known iris data. The classification problem here is to predict the iris type (iris setosa, iris versicolor,

iris type	iris setosa	iris versicolor	iris virginica
prior probability	0.333	0.333	0.333
petal length	1.46 ± 0.17	4.26 ± 0.46	5.55 ± 0.55
petal width	0.24 ± 0.11	1.33 ± 0.20	2.03 ± 0.27

Table 1. A naive Bayes classifier for the iris data. The normal distributions are described by stating $\hat{\mu} \pm \hat{\sigma}$. It is easy to see from this table how different petal lengths and widths provide evidence for the different types of iris flowers.

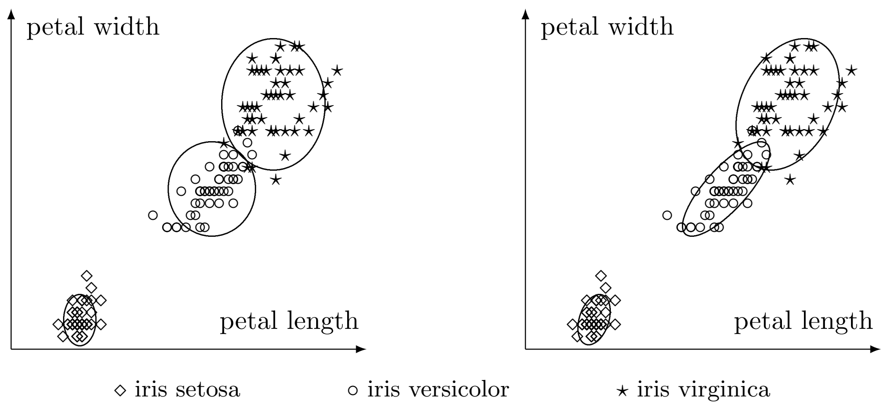

Fig. 2. Naive Bayes density functions for the iris data (axis-parallel ellipses, left) and density functions that take into account the covariance of the two measures (general ellipses, right). The ellipses are the 2σ-boundaries of the probability density functions.

or iris virginica) from measurements of the sepal length and width and the petal length and width. Due to the limited number of dimensions of a sheet of paper we confine ourselves to the latter two measures. The naive Bayes classifier induced from these two measures and all 150 cases is shown in table 1. The conditional probability density functions used by this classifier to predict the iris type are shown graphically in figure 2 on the left. The ellipses are the 2σ-boundaries of the (bivariate) normal distribution. These ellipses are axis-parallel, which is a consequence of the strong conditional independence assumptions made by a naive Bayes classifier: The normal distributions are estimated separately for each dimension and no covariance is taken into account. However, even a superficial glance at the data points reveals that the two measures are far from independent given the iris type. Especially for iris versicolor the density function is a rather bad estimate. However, if we allow for an additional edge between the petal length and the petal width, which, in this case, is most easily implemented by estimating the covariance matrix of the two measures, a much better fit to the data can be achieved (see figure 2 on the right, again the ellipses are the 2σ-boundaries of the probability density function). As a consequence the number of misclassifications drops from six to three (which can easily be made out in figure 2).

To summarize, probabilistic networks generalize naive Bayes classifiers in two ways. In the first place, by additional edges, the restriction to axis-parallel density functions can be removed and thus conditional dependences between the attributes can be taken into account. Secondly, in probabilistic networks there is usually no distinguished class attribute. Any attribute (or any set of attributes) can be made the focus of inferences. Thus several, quite different reasoning tasks can be solved with the same probabilistic network.

However, there is still the restriction that only one density function is estimated for each class. This is not always appropriate, especially under the normal distribution assumption. A better fit can often be achieved, if more than one "normal distribution cluster" per class is assumed. Such a generalization may be achieved by exploiting ideas from fuzzy clustering, which we study in the next section.

4 Fuzzy Clustering

The terms "classification" and "to classify" are ambiguous. In the preceding sections they are used to describe the process of assigning a class from a *predefined* set to an object or case under consideration. In classical statistics, however, these terms usually have a different meaning: They are used to describe the process of dividing a dataset of sample cases into groups of similar cases, with the groups *not* predefined, but to be found by the classification algorithm. This process is also called classification, because the groups to be found are usually (and confusingly) called *classes*. To avoid the confusion that may result from this ambiguity, the latter process, i.e., dividing a dataset into groups of similar cases, is often called *clustering* or *cluster analysis*, thus replacing the ambiguous term *class* with the less ambiguous *cluster*. Nevertheless a reader should keep in mind that in this section "to classify" has a different meaning than in the preceding ones (except where explicitly indicated otherwise).

Cluster analysis is, as already mentioned, a technique to classify data, i.e., to divide a given dataset of sample cases into a set of classes or *clusters*. The goal is to divide the dataset in such a way that two cases from the same cluster are as similar as possible and two cases from different clusters are as dissimilar as possible. Thus one tries to model the human ability to group similar objects or cases into classes and categories.

In classical cluster analysis [2] each case or object is assigned to exactly one cluster. That is, classical cluster analysis yields a crisp partitioning of a dataset with "sharp" boundaries between the clusters. It is therefore also called *crisp cluster analysis*. A crisp partitioning of the dataset, however, though often undisputedly successful, is not always appropriate. If the "clouds" formed by the data points corresponding to the cases or objects under consideration are not clearly separated by regions bare of any data points, but if, in contrast, in the joint domain of the attributes there are only regions of higher and lesser data point density, then the boundaries between the clusters can only be drawn with a certain amount of arbitrariness. Due to this arbitrariness it may be doubted, at least for data points close to the boundaries, whether a definite assignment to one class is justified.

An intuitive approach to deal with such situations is to make it possible that a data point belongs in part to one cluster, in part to a second etc. *Fuzzy cluster analysis* does just this: It relaxes the requirement that a data point must be assigned to exactly one cluster by allowing gradual memberships, thus offering the opportunity to deal with data points that do not belong definitely to one cluster [3, 4]. In general the performance of fuzzy clustering algorithms is superior to that of the corresponding crisp clustering algorithms [3].

Most fuzzy clustering algorithms are objective function based: They determine an optimal classification by minimizing an objective function. In objective function based clustering usually each cluster is represented by a *cluster prototype*. This prototype consists of a *cluster center* (whose name already indicates its meaning) and maybe some additional information about the size and the shape of the cluster. The cluster center is an instantiation of the attributes used to describe the domain, just as the data points in the dataset to divide. However, the cluster center is computed by the clustering algorithm and may or may not appear in the dataset. The size and shape parameters determine the extension of the cluster in different directions of the underlying domain.

The degrees of membership to which a given data point belongs to the different clusters are computed from the distances of the data point to the cluster centers w.r.t. the size and the shape of the cluster as stated by the additional prototype information. The closer a data point lies to the center of a cluster (w.r.t. size and shape), the higher is its degree of membership to this cluster. Hence the problem to divide a dataset $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_r\} \subseteq \mathbb{R}^n$ into m clusters can be stated as the task to minimize the distances of the data points to the cluster centers, since, of course, we want to maximize the degrees of membership.

Several fuzzy clustering algorithms can be distinguished depending on the additional size and shape information contained in the cluster prototypes, the way in which the distances are determined, and the restrictions that are placed on the membership degrees. We confine ourselves to a subset of all possible algorithms that is best suited to demonstrate the ideas we are interested in. To be more precise, we consider the task to minimize the objective function

$$J(\mathbf{X}, \mathbf{U}, \mathbf{B}) = \sum_{i=1}^{m} \sum_{j=1}^{r} u_{ij}^{\alpha} d^2(\boldsymbol{\beta}_i, \boldsymbol{x}_j) \tag{1}$$

subject to

$$\sum_{j=1}^{r} u_{ij} > 0, \text{ for all } i \in \{1, \dots, m\}, \tag{2}$$

$$\sum_{i=1}^{m} u_{ij} = 1, \text{ for all } j \in \{1, \dots, r\}, \tag{3}$$

where $u_{ij} \in [0,1]$ is the membership degree of datum $\boldsymbol{x}_j$ to cluster c_i, $\boldsymbol{\beta}_i$ is the prototype of cluster c_i, and $d(\boldsymbol{\beta}_i, \boldsymbol{x}_j)$ is the distance between datum $\boldsymbol{x}_j$ and prototype $\boldsymbol{\beta}_i$. $\mathbf{B}$ is the set of all m cluster prototypes $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m$. The $m \times r$ matrix $\mathbf{U} = [u_{ij}]$ is called the fuzzy partition matrix and the parameter α is called the fuzzifier. This parameter determines the "fuzziness" of the classification. With higher values for α the boundaries between the clusters become softer, with lower values they get harder. Usually $\alpha = 2$ is chosen.

Constraint (2) guarantees that no cluster is empty and constraint (3) ensures that the sum of the membership degrees for each datum equals 1. Fuzzy clustering algorithms which minimize the objective function J subject to these constraints are usually called *probabilistic clustering algorithms*, since the membership degrees for a given datum formally resemble the probabilities of its being a member of the corresponding cluster.

The objective function $J(\mathbf{X}, \mathbf{U}, \mathbf{B})$ is usually minimized by updating the membership degrees u_{ij} and the prototypes $\boldsymbol{\beta}_i$ in an alternating fashion, until the change $\Delta\mathbf{U}$ of the membership degrees is less than a given tolerance ε. This approach is also known as the *alternating optimization method.*

Skeleton of a Fuzzy Clustering Algorithm

```
Fix the number of clusters $m$
Fix $\alpha$, $\alpha \in (1, \infty)$
Initialize the fuzzy $m$-partition $\mathbf{U}$
REPEAT
    Update the parameters of each clusters prototype
    Update the fuzzy $m$-partition $\mathbf{U}$ using equation (4) (see below)
UNTIL $|\Delta\mathbf{U}| < \varepsilon$
```

To minimize the objective function J, the membership degrees are updated using equation (4) below. This equation can be derived by differentiating the objective function J.

$$u_{ij} = \begin{cases} \dfrac{1}{\displaystyle\sum_{k=1}^{m} \left(\frac{d^2(\boldsymbol{x}_j, \boldsymbol{\beta}_i)}{d^2(\boldsymbol{x}_j, \boldsymbol{\beta}_k)} \right)^{\frac{1}{\alpha - 1}}}, & \text{if } I_j = \emptyset, \\ 0, & \text{if } I_j \neq \emptyset \text{ and } i \notin I_j, \\ x, x \in [0,1] \text{ such that } \sum_{i \in I_j} u_{ij} = 1, & \text{if } I_j \neq \emptyset \text{ and } i \in I_j, \end{cases} \tag{4}$$

where $I_j = \{i | 1 \leq i \leq m, d^2(\boldsymbol{x}_j, \boldsymbol{\beta}_i) = 0\}$, i.e., I_j represents (by their indices) the set of all clusters, to whose centers the datum $\boldsymbol{x}_i$ is identical.

Equation 4 is used to update the membership degrees in all probabilistic clustering algorithms. In contrast to this, the formulae for computing the prototypes vary depending on what additional information is included in the prototypes (size and shape parameters) and how the distances are determined. Each choice leads to a different algorithm.

The simplest choice, of course, is to include in the cluster prototypes only the cluster centers and to use a Euclidean distance function (thus implicitely fixing that the clusters are spheres of equal size). The result is the well-known fuzzy C means algorithm, which was developed by Bezdek [3]. This algorithm, however, is very inflexible and thus often leads to an insufficient fit to the data. In addition, it cannot easily be interpreted probabilistically, which is important for our considerations. Therefore, in the following, we discuss a more flexible algorithm that is explicitely based on a probabilistic model.

In [8] Gath and Geva suggested a fuzzy clustering algorithm (the *FMLE*—Fuzzy Maximum Likelihood Estimation) which is based on the assumption that the dataset to be classified was generated by m n-dimensional normal distributions, where m is the number of clusters. To represent the necessary parameters, each cluster prototype is a triple $\boldsymbol{\beta}_i = (\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, p_i)$, where $\boldsymbol{\mu}_i$ is the expected value of the multivariate normal distribution, $\boldsymbol{\Sigma}_i$ is the $n \times n$ covariance matrix, and p_i is the probability of the cluster c_i, such that $\sum_{i=1}^{m} p_i = 1$. Intuitively, $\boldsymbol{\mu}_i$ is the cluster center, $\boldsymbol{\Sigma}_i$ describes the size and shape of the cluster (the determinant of $\boldsymbol{\Sigma}_i$, for example, is a measure of the cluster size), and p_i determines the relative frequency of data points that are generated by the cluster c_i. The set of all cluster prototypes defines a complex probability density function on the n-dimensional domain under consideration, from which the probability densities at the data points in the dataset X can be determined.

The fuzzy maximum likelihood estimation algorithm classifies the data using a maximum likelihood approach. That is, it tries to determine the parameters of the cluster prototypes in such a way that the probability of the dataset (or, to be more precise, the sum of the probability densities at the data points in the dataset) is maximized. The rationale underlying this is that before observing the data all sets of prototypes are equally likely. With this assumption, the posterior probability of the dataset given the prototypes is a direct measure of the probability of the prototypes given the dataset (simply apply Bayes rule).

To maximize the likelihood of the data, the distance measure used in the fuzzy maximum likelihood estimation algorithm is inversely proportional to the probability density as defined by a cluster prototype. To be more precise, the distance is computed as

$$d(\boldsymbol{x}_j, \boldsymbol{\beta}_i) = \text{const.} \cdot \left(\frac{p_i}{\sqrt{|\boldsymbol{\Sigma}_i|(2\pi)^n}} \;\; \exp\left(-\frac{1}{2} (\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x}_j - \boldsymbol{\mu}_i) \right) \right)^{-1}.$$

Based on this distance measure the membership degrees are computed using equation (4).

However, if the fuzzy maximum likelihood estimation algorithm is applied exactly in the way outlined above, it tends to be unstable, mainly because of the large number of degrees of freedom. To make it more stable, it is advisable to introduce some restrictions. A serious problem that occurred frequently during our experiments was that one of the clusters became very small, with the shape either a sphere or a very thin and long ellipsoid. Therefore, in some experiments, we restricted the relative size of the clusters by introducing a constraint on the relative values of the determinants: If they deviate more than by a factor of three from the average, they are forced back into the range defined by the average and this factor. This lead to a much more stable behaviour and better results.

Let us now compare the fuzzy maximum likelihood estimation algorithm to a naive Bayes classifier. If we assume that the attributes are independent of each other given the clusters, just as we did for the naive Bayes classifier, then the clusters are defined by their probability, their centers, and the variances for each dimension (or, in other words, in the covariance matrix all elements but the diagonal elements are zero). Intuitively, with this assumption, the clusters are axis-parallel (see above). In this case the degree of membership of a datum to a clusters is computed in much the same way as a naive Bayes classifier computes the conditional class probabilities. Thus, an axis-parallel variant of the fuzzy maximum likelihood estimation algorithm [13] can be seen as a direct analogon of a naive Bayes classifier. The only difference, of course, is that a naive Bayes classifier already knows the classes the cases in the dataset belong to, whereas the clustering algorithm tries to find a good partitioning into classes. Nevertheless, if there is class information, and if the attributes convey information about the class, the class information can often be used to assess the quality of a clustering result.

As an illustrative example we turn again to the iris data we already used above. Figure 3 shows the result of the axis-parallel variant of the fuzzy maximum likelihood estimation algorithm on the iris data, if all four attributes are used (although—for technical reasons—only three dimensions are shown). On the left the data points are connected to the centers of the clusters for which their degree of membership is highest. The ellipsoids on the right indicate the 3σ-boundaries of the multivariate normal distributions.[6] It is easy to see that the result closely resembles the result of the naive Bayes classifier.

If the assumption that the attributes are independent given the class does not hold, the normal version of the fuzzy maximum likelihood estimation algorithm can be applied. Since it uses a full covariance matrix, dependencies between the attributes can be taken into account. Again we illustrate this with the help of the iris data. Figure 4 shows the result of the normal version of the fuzzy maximum likelihood estimation algorithm, if all four attributes are used (although only three dimensions are shown). On the left the data points are connected to the center of the cluster for which their degree of membership is highest. The ellipsoids on the right indicate the 3σ-boundaries of the multivariate normal distributions. Although this figure, especially the ellipsoids on the right, are a little harder to visualize in three dimensions, it is fairly obvious that the fit to the data is better than in figure 3.

As indicated, the results shown in figure 3 and 4 are computed using all attributes of the iris data set. However, usually the iris data set is classified based on the petal length and width only, since these are the two most informative attributes. In addition, comparing the results to the results of the preceding sections is easier, if we confine ourselves to these two dimensions. Of course,

[6] The fuzzy data analysis program `fcluster` which was used to create these screen shots can be obtained free of charge from our WWW-site: `http://fuzzy.cs.uni-magdeburg.de`.

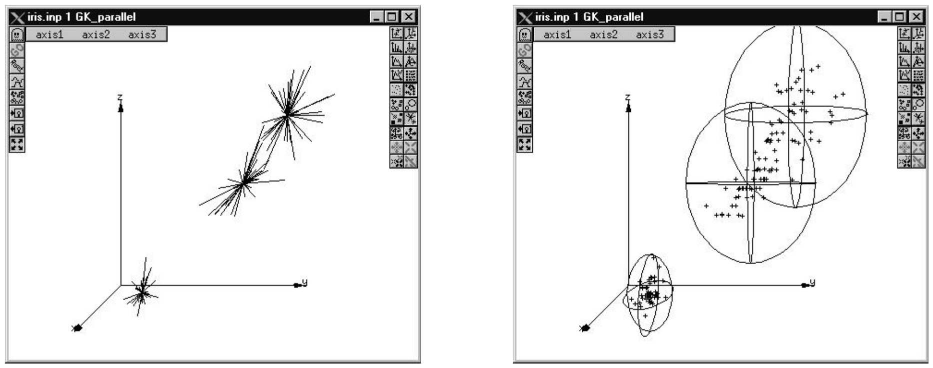

Fig. 3. The iris dataset classified with the axis-parallel variant of the fuzzy maximum likelihood algorithm, all attributes used. The vertical axis is the petal width, the horizontal the petal length and the depth is the sepal width. On the left each data point is connected to the center of that cluster to which it has the highest degree of membership. The ellipsoids on the right indicate the 3σ-boundaries of the multivariate normal distribution.

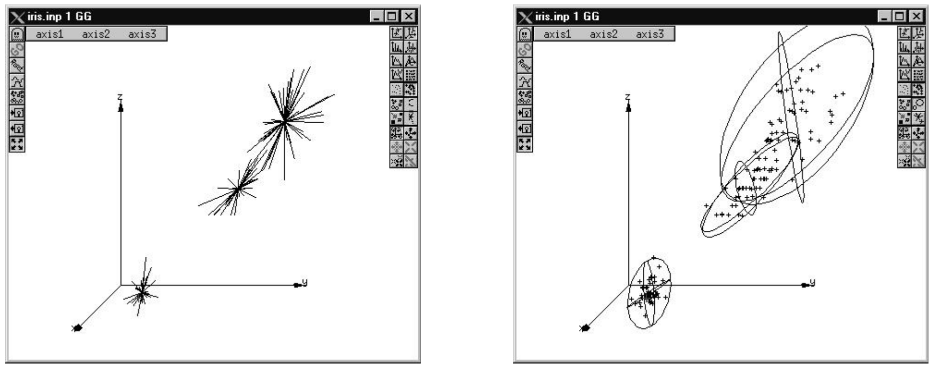

Fig. 4. The iris dataset classified with the normal fuzzy maximum likelihood algorithm, all attributes used. The vertical axis is the petal width, the horizontal the petal length and the depth is the sepal width. On the left each data point is connected to the center of that cluster to which it has the highest degree of membership. The ellipsoids on the right indicate the 3σ-boundaries of the multivariate normal distribution.

this changes the results of the clustering algorithms, since the distance functions change. Figure 5 shows the result of the axis-parallel variant of the fuzzy maximum likelihood estimation algorithm on the iris data, if only the petal length and width are used. The clusters found are hardly distinguishable from the naive Bayes clusters shown on the left in figure 2. If the three iris types are assigned to the clusters and the dataset is classified (in the sense of predicting the iris type), the number of errors is the same as for a naive Bayes classifier.

The result of the fuzzy maximum likelihood estimation algorithm, shown on the right in figure 5, however, does not resemble the one obtained by a proba-

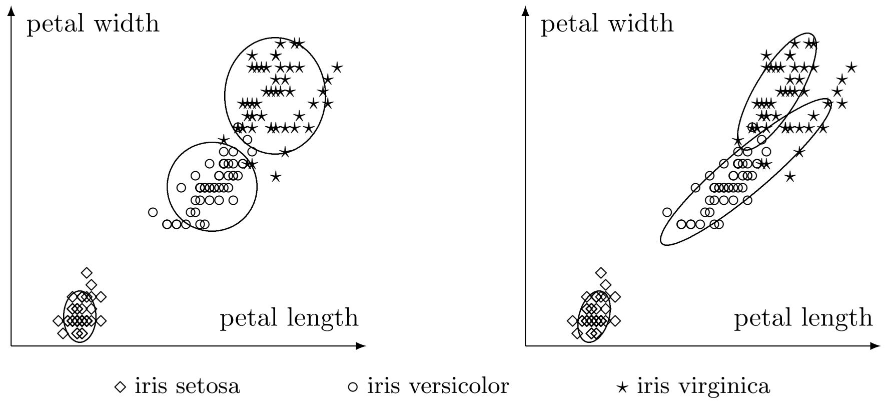

Fig. 5. Density functions generated by the fuzzy maximum likelihood estimation algorithm for the iris data, three clusters, axis-parallel version (left) and normal version (right). The ellipses are the 2σ-boundaries of the probability density functions. Petal width and petal length used only.

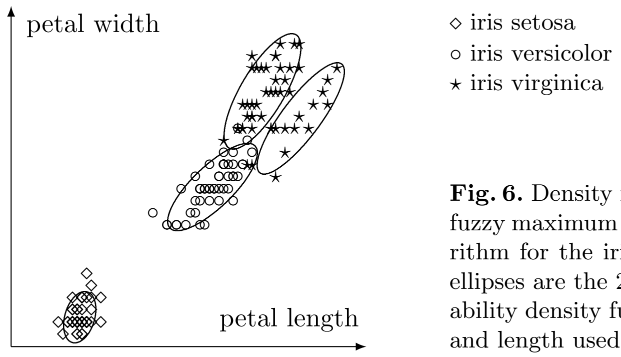

Fig. 6. Density functions generated by the fuzzy maximum likelihood estimation algorithm for the iris data, four clusters. The ellipses are the 2σ-boundaries of the probability density functions. Only petal width and length used.

bilistic network that takes into account the covariance of the two measures. Obviously the problem is that the fuzzy maximum likelihood estimation algorithm does not use any class information: Without such information the partitioning found is much more likely than the the probabilistic network clusters.

Fortunately, we can exploit the fact that in fuzzy clustering, since no class information is taken into account, we are not bound to using just one cluster per class (as already mentioned above). We may choose freely, and if we take a closer look at the iris data, a choice of four clusters suggests itself. Indeed, with this number of clusters the algorithm yields a model that excellently fits the data as shown in figure 6. The iris virginica cases have been divided into two clusters, which, indeed, is what a human would do under these circumstances. It has to be admitted though that even with the constraint on the cluster sizes introduced above, the fuzzy maximum likelihood estimation algorithm is not completely stable and that this is not the only classification we obtained. Fortunately, the different results can easily be ranked by simply computing the value of the objective function. Since this function has to be minimized, a smaller value indicates a better solution. The value of the objective function for the result shown in figure 6 is only half as large as the value for any other result we obtained and thus this solution can clearly be regarded as the one to be chosen.

This example indicates how naive Bayes classifiers and maybe also probabilistic networks can profit from fuzzy clustering. Using more than one cluster per class can often improve the fit to the data and thus in the future we plan to investigate combinations of the discussed methods.

5 Conclusions

In this paper we discussed the relationship between naive Bayes classifiers, probabilistic networks, and fuzzy cluster analysis. As we hope to have made clear, both probabilistic networks and the fuzzy maximum likelihood estimation algorithm can be seen as generalizations of naive Bayes classifiers. However, they generalize them to different degrees. Whereas probabilistic networks only remove the requirement that the multivariate normal distributions have to be axis-parallel (by taking covariances into account), fuzzy clustering does not only this, but also lets us use more than one cluster per class. Since the normal distribution assumption, even if covariances are taken into account, is not always appropriate, this opens up a route to enhance the capabilities of the former methods. The idea is simply to split one or more classes into pseudo-subclasses, each with a multivariate normal distribution of its own. To find a good split into subclasses, fuzzy clustering methods may be used, as the example shown clearly indicates.

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 (IJCAI'89, Detroit, MI, USA)*, 1080–1085. Morgan Kaufman, San Mateo, CA, USA 1989
2. M.J.A. Berry and G. Linoff. *Data Mining Techniques — For Marketing, Sales and Customer Support.* J. Wiley & Sons, Chichester, England 1997
3. J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, NY, USA 1981
4. J.C. Bezdek and S.K. Pal. *Fuzzy Models for Pattern Recognition — Methods that Search for Structures in Data.* IEEE Press, Piscataway, NJ, USA 1992
5. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. *Proc. 12th Int. Conf. on Machine Learning (ICML'95, Lake Tahoe, CA, USA)*, 194–202. Morgan Kaufman, San Mateo, CA, USA 1995
6. R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* J. Wiley & Sons, New York, NY, USA 1973
7. K.J. Ezawa and S.W. Norton. Knowledge Discovery in Telecommunication Services Data Using Bayesian Network Models. *Proc. 1st Int. Conf. on Knowledge Discovery*

and Data Mining (KDD'95, Montreal, Canada), 100–105. AAAI Press, Menlo Park, CA, USA 1995

8. I. Gath and A.B. Geva. Unsupervised Optimal Fuzzy Clustering. *IEEE Trans. Pattern Anal. Mach. Intelligence* 11:773–781. IEEE Press, Piscataway, NJ, USA, 1989
9. J. Gebhardt and R. Kruse. The Context Model — An Integrating View of Vagueness and Uncertainty *Int. Journal of Approximate Reasoning* 9:283–314. North-Holland, Amsterdam, Netherlands 1993
10. 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*, 407–418. J. Wiley & Sons, New York, NY, USA 1996
11. I.J. Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.* MIT Press, Cambridge, MA, USA 1965
12. D. Heckerman. *Probabilistic Similarity Networks.* MIT Press, Cambridge, MA, USA 1991
13. F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis.* J. Wiley & Sons, Chichester, England 1999
14. R. Kruse, E. Schwecke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge-based Systems: Numerical Methods (Series Artificial Intelligence).* Springer, Berlin, Germany 1991
15. R. Kruse, J. Gebhardt, and F. Klawonn. *Foundations of Fuzzy Systems.* J. Wiley & Sons, Chichester, England 1994
16. P. Langley, W. Iba, and K. Thompson. An Analysis of Bayesian Classifiers. *Proc. 10th Nat. Conf. on Artificial Intelligence (AAAI'92, San Jose, CA, USA)*, 223–228. AAAI Press and MIT Press, Menlo Park and Cambridge, CA, USA 1992
17. P. Langley and S. Sage. Induction of Selective Bayesian Classifiers. *Proc. 10th Conf. on Uncertainty in Artificial Intelligence (UAI'94, Seattle, WA, USA)*, 399–406. Morgan Kaufman, San Mateo, CA, USA 1994
18. 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. Blackwell, Oxford, United Kingdom 1988
19. D. Nauck, F. Klawonn, and R. Kruse. *Foundations of Neuro-Fuzzy Systems.* J. Wiley & Sons, Chichester, England 1997
20. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (2nd edition).* Morgan Kaufman, San Mateo, CA, USA 1992
21. A. Saffiotti and E. Umkehrer. PULCINELLA: A General Tool for Propagating Uncertainty in Valuation Networks. *Proc. 7th Conf. on Uncertainty in Artificial Intelligence (UAI'91, Los Angeles, CA, USA)*, 323–331. Morgan Kaufman, San Mateo, CA, USA 1991
22. G. Shafer and P.P. Shenoy. *Local Computations in Hypertrees (Working Paper 201).* School of Business, University of Kansas, Lawrence, KS, USA 1988
23. P.P. Shenoy. *Valuation-based Systems: A Framework for Managing Uncertainty in Expert Systems (Working Paper 226).* School of Business, University of Kansas, Lawrence, KS, USA 1991
24. J. Whittaker. *Graphical Models in Applied Multivariate Statistics.* J. Wiley & Sons, Chichester, England 1990