Partially Supervised Learning of Fuzzy Classification Rules

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.),

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von Dipl.-Inform. Aljoscha Alexander Klose,
geboren am 1. April 1971 in Braunschweig

Magdeburg, den 23.01.2004

Zusammenfassung

Das Forschungsgebiet *Data Mining* oder auch *Wissensentdeckung in Datenbanken* ist als Antwort auf die Herausforderung entstanden, die enorm wachsenden Datenbestände zu analysieren, die heutzutage von Unternehmen und Forschungsinstitutionen gesammelt werden. Eine wichtige Aufgabe des Data Minings ist die Klassifikation, wobei Fuzzytechniken zur Extraktion von Klassifikationsregeln aus Daten wegen ihrer dem Menschen verständlichen Modellierung reizvoll sind.

Oftmals enthalten die zu analysierenden Datensätze keine Klasseninformation, und ihre Anzahl verbietet ein manuelles Kennzeichnen. Daher gibt es ein steigendes Interesse an teilüberwachten Verfahren, die auch mit nur teilweise vorhandener Klasseninformation lernen können. Leider verwenden die meisten aktuellen Data-Mining-Verfahren überwachtes Lernen, und die meisten teilüberwachten Verfahren erzeugen keine dem Menschen verständlichen Modelle.

In dieser Arbeit stelle ich die Ideen der Fuzzyklassifikation und des Lernens von Fuzzyklassifikatoren vor, wobei ich speziell deren Fähigkeiten und Interpretierbarkeit betrachte und einige ihrer Eigenarten und Probleme aufdecke. Weiterhin gebe ich einen Überblick über teilüberwachte Lernverfahren mit einem Schwerpunkt auf Fuzzymethoden und untersuche deren Schwächen insbesondere im Hinblick auf die Induktion interpretierbarer Fuzzyregeln. Die wesentlichen Leistungen dieser Arbeit bestehen in der Entwicklung eines evolutionären Algorithmus und spezialisierter Fitnessfunktionen, die das teilüberwachte Lernen interpretierbarer Fuzzyregeln erlauben.

Abstract

The research area of *Data Mining* or *Knowledge Discovery in Databases* has emerged in response to the challenges of analyzing the tremendously growing datasets gathered nowadays by companies and research institutions. Classification is one important task of data mining, where fuzzy techniques to extract classification rules from data are appealing due to their human understandable modeling.

Often, datasets to be analyzed do not contain class labels, and their size renders manual labeling infeasible. Thus, there is an increasing interest in semi-supervised methods that can learn from only partially labeled data. Unfortunately, most current data mining methods are supervised, and most current semi-supervised methods do not generate human understandable models.

In this thesis we review the key concepts of fuzzy classification and fuzzy classifier learning, with a focus on their capabilities and interpretability, and reveal some common peculiarities and pitfalls. Furthermore, we review approaches to semi-supervised learning with a stress on fuzzy methods, and show their deficiencies, particularly for inducing interpretable fuzzy rules. The main achievements of this thesis are the development of an evolutionary algorithm and specialized fitness functions that allow semi-supervised learning of interpretable fuzzy rules.

Contents

Chapter 1

Introduction

Over the past decade, dramatic advances in information and sensor technology could be witnessed. Increasing processor power, growing storage device capacities, and the emergence of the world wide web enabled companies and scientific and governmental institutions to collect, store and process large archives of all kinds of data like numbers, tables, documents, images, and sounds. However, it turned out that traditional ways to analyze data hardly scale to those database sizes, and that transforming the abundance of data into useful knowledge is therefore difficult. In reply to these challenges a new area of research has emerged, which has been named "Knowledge Discovery in Databases" or "Data Mining". Fayyad et al. (1996) give the following definition:

> Knowledge Discovery in Databases (KDD) is a research area that considers the analysis of large databases in order to identify valid, useful, meaningful, unknown, and unexpected relationships.

The stress lies on methods that help users, who often have a vague understanding of their data, to support their hypotheses and models, to find other patterns or regularities hidden in the data, and to translate these findings into human notions to give the users an understanding of their data. A number of analysis tools for data mining have been established over the past years. Some well-known methods are, for instance, statistics (regression analysis, discriminant analysis etc.), time series analysis, decision trees, cluster analysis, neural networks, inductive logic programming, and association rules.

However, characteristics and thus demands of current applications are changing and call for new algorithms. In this introduction we outline which important trends we see in the sources of the data to be analyzed, and thus which trends we expect in the characteristics of the data. We argue why fuzzy methods are well suited to deal with these changing demands, and why lately these challenges consequently lead to a considerably growing interest of the research community into *semi-supervised* learning methods.

Changing Data Characteristics

Most classical data mining methods, like decision trees and neural networks, expect an input of single uniform tables of tuples of attribute values. In many modern applications, however, the data to be analyzed come from heterogeneous information sources: Many of the archives contain images, texts, video, or even sound data. We certainly cannot expect to find data mining algorithms that are generally applicable to all mentioned kinds of information sources. The approaches will always strongly depend on some kind of application-specific preprocessing to extract characterizing features from the specific type of media. Additionally, to enable data mining in such feature spaces we suppose that it is crucial to exploit any available a priori knowledge, and thus to have algorithms that support to incorporate such information.

As the data seldomly comes from well designed experiments or measurements, it is often unevenly distributed in the input space and class frequencies are often unbalanced. Furthermore, the data is often of low quality. Algorithms must thus be able to deal with uncertainty and imprecision. Missing values are also a common problem that data mining algorithms should be able to handle. A special case of missing values are missing class labels: The focusing of data mining methods on supervised learning is a severe drawback in many real-world applications. In contrast to the abundance of data available in the archives, labeling these data is often a problem. In many cases, the labels for the training samples have to be assigned manually or determined by expensive analyses. Typical examples of such domains include speech processing, image analysis, text classification, or medical or biological applications. Labeling a complete dataset can become an at least tedious if not infeasible task when there are many objects—and in some applications "many" can easily mean tens of thousands. With increasing sizes of the databases to be analyzed, learning from data which is only partially labeled becomes more and more important.

The Role Of Fuzzy Techniques

The outlined characteristics of the data—their quantity, complexity, dimensionality and imperfection—, the essential of extracting understandable patterns from these, and the need to incorporate available background knowledge in that process make fuzzy techniques an interesting tool for data mining (Kruse and Klose, 2002). They can transform between computer representations and (naturally linguistic) human concepts, and thus allow to "compute with words". The inherent imprecision of words is not necessarily a weakness, but on the contrary, can be crucial to model complex systems. From our own experience we observed that many practical applications have this certain robustness where full precision is not necessary. In such cases, exaggerated precision can be a waste of resources, and solutions obtained using fuzzy approaches might be easier to understand and to apply. Good examples of models that gain their strengths by explicitly taking into account vagueness, imprecision or uncertainty are systems based on fuzzy rules.

One important task in data mining is classification. Fuzzy *if-then* rules have become popular for this task, as their use of linguistic variables is close to human descriptions of structure in data. For data analysis we are looking for procedures that can extract fuzzy rules from a dataset of examples. Ideally, these rules allow to accurately classify new objects and describe the structure of the data distribution in an understandable fashion. If we apply such techniques, we must be aware of a tradeoff between precision and interpretability. However, the results should not only be judged for their accuracy, but also for their interpretability, as the ultimate goal of data mining is to extract human understandable patterns. Not all fuzzy models are equally suited to accurately solve the task at hand and still generate interpretable models.

Semi-Supervised Learning

Most existing approaches to fuzzy rule extraction are supervised, i.e. they expect that all examples are labeled. Unfortunately, as initially mentioned, in many current domains it is not possible to access the labels for all objects. In such cases, one usually confines the examples to a certain—hopefully representative—fraction of the data and leaves the unlabeled data aside. In spite of being unlabeled, the additional data might still bear valuable information on the true distribution of the objects in the input space.

There have been several proposals for methods that exploit the remain-

ing, otherwise discarded data to support the learning of a classifier. With growing database sizes it is not surprising that there is an increasing interest in approaches that are able to learn from partially labeled data. Nevertheless, combinations of descriptive fuzzy classification rules and semi-supervised learning have hardly been investigated.

Outline

In Chapter 2, we review the key concepts of fuzzy rule based classifiers. There are a number of different ways to evaluate fuzzy rules. We describe the influence of these alternatives on the approximation capabilities and interpretability of fuzzy classifiers. We use these findings to motivate our choices in our own implementation. In this chapter we consider the fuzzy classifiers as given. Chapters 3 and 4 deal with approaches to automatically extract fuzzy rules from example data. Since in this thesis the induction of rules using evolutionary algorithms plays a prominent role, these methods are described in a chapter of its own (Chapter 4).

In Chapter 5, we review previous work in the field of semi-supervised learning. We focus on methods for semi-supervised learning of fuzzy models. However, little has been done on the extraction of *interpretable* fuzzy rules from partially labeled data. We discuss whether those approaches are suited for extension, and propose our own semi-supervised approaches for fuzzy rule induction based on evolutionary algorithms in Chapter 6. The functionality of all semi-supervised methods is illustrated and compared on artificial datasets and data from the UCI repository of machine learning. In Chapter 7, additionally two real-world applications are presented to show the applicability of our approaches. Limitations and open questions are discussed in Chapter 8.

Chapter 2

Classification with Fuzzy Rules

When humans are to formulate their knowledge about dependencies they will certainly use words, and often use if-then rules. If we want to build systems that allow humans to introduce their prior knowledge, and that, vice versa, produce output that is understandable to humans, it seems reasonable to work with rules and natural language concepts. On the other hand, language is always connected with vagueness, which is usually tried to be avoided in computer systems. Humans, however, can easily handle this kind of vagueness. Even more so, the achieved level of abstraction is one explicit strength of natural language and often necessary to handle complex systems. Fuzzy set theory is one approach to model natural language concepts in automated systems.

This chapter shall give the reader a survey on classification based on fuzzy if-then rules. In the next section, we introduce classification as a general functional dependence that maps points from a feature space into a set of classes. Section 2.2 briefly outlines fuzzy set theory. We argue that classic ("crisp") set theory is not able to model linguistic concepts adequately. The breakthrough of fuzzy techniques came mainly with fuzzy controllers, where fuzzy sets are used in rules to describe functional relationships. As we also show in Section 2.2, basically the same concepts can be used to represent classification functions.

An interesting—however often neglected—aspect are the theoretical capabilities of fuzzy classifiers. Especially the use of linguistic terms in fuzzy

rules leads to the erroneous belief that the system will "somehow" cope with the uncertainties and manage to classify objects in the right way. But, of course, there are exact numbers underlying the fuzzy sets. Thus, when fuzzy rules are evaluated, there are sharp decision boundaries between classes. Hence, it is important to have an intuition, how the final results of fuzzy classification rules look like. This can help to avoid some peculiarities and pitfalls. These issues are considered in Section 2.3, and are an important prerequisite for the choice of the fuzzy model used in our fuzzy classifier approach proposed in Chapter 6.

Another aspect important for our approach is an alternative interpretation of membership degrees as probability densities. In Section 2.4, we discuss the restrictions under which this interpretation is possible. We use this reading of membership degrees in Chapter 6, when we derive our measures for semi-supervised learning.

2.1 Fundamentals of Classification

Let us consider a universe $\Omega = (\mathcal{A}, \mathcal{C})$, consisting of a set of n_d attributes (or features) $\mathcal{A} = \{X_i | i = 1, \ldots, n_d\}$ and a set of n_c classes $\mathcal{C} = \{c_i | i = 1, \ldots, n_c\}$. Let $\boldsymbol{X} = X_1 \times \cdots \times X_{n_d}$ be the n_d-dimensional feature space spanned by the attributes. The objects ω of our universe are characterized by their feature vectors $\boldsymbol{x} \in \boldsymbol{X}$. We assume the classes to be mutually exclusive and exhaustive, i.e. each object ω belongs to one and only one class $c \in \mathcal{C}$. The task of classification is to reconstruct or predict the class c^ω of an object ω given its feature vector $\boldsymbol{x}^\omega$. A *classifier* can be defined as a function

$$g : \boldsymbol{X} \to \mathcal{C}. \tag{2.1}$$

For this definition, it is irrelevant how function g is represented or where it comes from. A central subject of *pattern recognition* is to discover models—i.e. families of functions, that are suited to represent g for different problems—and to develop algorithms to learn these models from data. Learning in this context means to choose an instance within a family of functions that is optimal with respect to some given example data.

Optimality usually means that the total probability of error over the feature space is minimal. Let us assume that the objects are randomly chosen from some distribution, such that a feature vector $\boldsymbol{x}$ has a probability $P(\boldsymbol{x})$. Let us further assume that we know the probability $P_g(error|\boldsymbol{x})$ that g produces an error at point $\boldsymbol{x}$ in the feature space. Then the total

probability of error is

$$P_g(error) = \int_{\boldsymbol{X}} P_g(error|\boldsymbol{x})P(\boldsymbol{x})d\boldsymbol{x}. \tag{2.2}$$

Let $P(c_i|\boldsymbol{x})$ denote the posterior class probabilities at a point $\boldsymbol{x}$ in the feature space. Then an optimal classifier should predict the class with the maximal posterior probability $c_{\max}(\boldsymbol{x}) = \text{argmax}_{c\in\mathcal{C}}P(c|\boldsymbol{x})$, and thus minimize $P(error|\boldsymbol{x}) = 1 - P(c_{\max}(\boldsymbol{x})|\boldsymbol{x})$. However, the real probabilities are generally unknown, and must thus be estimated from the example dataset.

Let us consider a dataset $\mathcal{D} = \{\omega_1, \ldots, \omega_n\}$. Each of the n objects ω_i has a corresponding pair $(\boldsymbol{x}^{\omega_i}, c^{\omega_i})$ of feature vector and class label. A reasonable estimation of $P_g(error)$ is then

$$\hat{P}_g(error) = \frac{1}{n}\sum_{\omega\in\mathcal{D}} I(g(\boldsymbol{x}_i), c^{\omega}), \tag{2.3}$$

with an indicator function I that assumes value 1 for misclassifications:

$$I : \mathcal{C}\times\mathcal{C} \to \{0,1\}, \quad I(c,c') = \begin{cases} 0 & \text{if } c = c' \\ 1 & \text{else.} \end{cases} \tag{2.4}$$

$\hat{P}_g(error)$ thus simply counts the number of errors (in relation to the total number of classified objects). This definition is reasonable, if all misclassifications cause equal costs,[1] and if we use *crisp labels* as outputs of our classifier function g, i.e. if g returns exactly one class c for every feature vector $\boldsymbol{x}$. In some situations, however, it might be desirable to get extra information about, e.g., the certainty or ambiguity of the classification. Consider, for example, a classifier in a medical domain that shall distinguish between benign and malignant tumors. There might be cases, where it is only slightly more probable that a tumor is benign. In such cases, if we knew about the narrow decision of g, we might try to make additional tests. However, g gives us no information about the certainty of the decision. Therefore, it can be advantageous to have a *soft classifier* (that is, a classifier with *soft labels*), defined as

$$\begin{gathered}\tilde{g} : \boldsymbol{X} \to \mathbb{R}^{n_c}, \quad \tilde{g}(\boldsymbol{x}) := (\tilde{g}_{c_1}(\boldsymbol{x}), \ldots, \tilde{g}_{c_{n_c}}(\boldsymbol{x}))^T, \\ \text{with } \tilde{g}_{c_i} : \boldsymbol{X} \to \mathbb{R}, i = 1, \ldots, n_c\end{gathered} \tag{2.5}$$

[1] This is also called *zero-one-loss*. If misclassifications cause different costs, one usually defines a *cost* or *loss matrix* that specifies the costs accompanied by an object ω of class c_i wrongly classified as $c_j, j \neq i$. In that case, instead of minimizing the total error probability, learning tries to minimize the corresponding expected costs.

that returns a vector of real numbers, one for every class. A crisp classifier g can be derived from $\tilde{g}$ by returning the class with the maximal output value:

$$g : \boldsymbol{X} \to \mathcal{C}, \quad g(\boldsymbol{x}) := \underset{c \in \mathcal{C}}{\operatorname{argmax}} \, (\tilde{g}(\boldsymbol{x}))_c = \underset{c \in \mathcal{C}}{\operatorname{argmax}} \, \tilde{g}_c(\boldsymbol{x}), \tag{2.6}$$

where $(\tilde{g}(\boldsymbol{x}))_c$ denotes that element of output vector returned by $\tilde{g}$ that corresponds to class c. Depending on the definition of $\tilde{g}$, the outputs for the classes can be interpreted as measures of certainty, and thus their relation as a measure of ambiguity.

If $\tilde{g}$ maps the input data points to $[0,1]^{n_c}$, it can formally be seen as *possibilistic* labeling. If additionally $\sum_{i=1}^{n_c} \tilde{g}_{c_i} = 1$ is fulfilled, the labels are formally *probabilistic*. If the objects ω of the training dataset have also possibilistic or probabilistic labels $\tilde{c}^\omega \in [0,1]^{n_c}$, it is also common to optimize the mean squared error between the true labels and the classifier output:

$$E_{\tilde{g}} = \sum_{\omega \in \mathcal{D}} \|\tilde{g}(\boldsymbol{x}^\omega) - \tilde{c}^\omega\|^2. \tag{2.7}$$

If only the crisp labels c^ω are given, usually 1-in-n-encoding[2] is used:

$$\begin{aligned} \tilde{c}_{\text{1-in-n}}(c) &= (\tilde{c}_1(c), \ldots, \tilde{c}_{n_c}(c))^T \\ \tilde{c}_i(c) &= \begin{cases} 1, & \text{if } c = c_i \\ 0, & \text{else.} \end{cases} \end{aligned} \tag{2.8}$$

Different scientific communities have proposed a variety of function classes to represent classifiers g. These function classes differ, for example, in

- the motivating ideas and the theoretical foundations,
- the flexibility in adapting to class distributions,
- the support of *soft labels* (and how these can be interpreted),
- the complexity of corresponding learning algorithms, and
- the transparency of the represented separations to humans.

Especially for the last aspect, rule-based approaches have in general some strengths. The representation of knowledge in the form of rules is intuitively understandable. Especially fuzzy rules, which abstract from numbers and

[2] As in our notation n_c denotes the number of classes, it could be called 1-in-n_c-encoding. However, this would be uncommon.

use linguistic terms instead, are suited to represent expert knowledge in a machine readable manner, and present the results of automatic methods to humans in an interpretable way. Additionally, they naturally yield soft labels that can be interpreted as similarities to prototypical cases. In the next section we review the fundamentals of fuzzy set theory and fuzzy rules, and show how they are used to represent classification functions $\tilde{g}$.

2.2 Fuzzy Rules to Represent Classification Knowledge

If humans describe objects, they effectively use linguistic terms like, for instance, *small*, *old*, *long*, *fast*. However, classical set theory is hardly suited to define sets of objects that satisfy such linguistic terms. Let us, for examples, assume a person being assigned to the set of *tall* persons. If a second person is only insignificantly smaller, it should also be assigned to this set, and thus it seems reasonable to formulate a rule like "a person who is less than 1mm smaller than a tall person is also tall" to define our set. However, if we repeatedly apply this rule, obviously persons of *any* size will be assigned to the set of tall persons. Any threshold for the concept *tall* will be hardly justifiable. On the other hand, it is easy to find persons that are *tall* or *small*, respectively. Modeling the typical cases is not the problem, but the *penumbra* between the concepts can hardly be appropriately modeled with classical sets.

The main principle of fuzzy set theory is to generalize the concept of set membership (Zadeh, 1965). In classical set theory a characteristic function

$$\begin{aligned} \mathbb{1}_A &: \Omega \to \{0,1\} \\ \mathbb{1}_A(\omega) &= \begin{cases} 1, & \text{if } \omega \in A \\ 0, & \text{else,} \end{cases} \end{aligned} \tag{2.9}$$

defines the memberships of objects $\omega \in \Omega$ to a set $A \subset \Omega$. In fuzzy set theory the characteristic function is replaced by a membership function

$$\mu_M : \Omega \to [0,1], \tag{2.10}$$

that assigns numbers to objects $\omega \in \Omega$ according to their membership degree to a fuzzy set $M \subset \boldsymbol{X}$. A membership degree of one means that an object fully belongs to the fuzzy set, zero means that it does not belong to the set. Membership degrees between zero and one correspond to partial memberships.

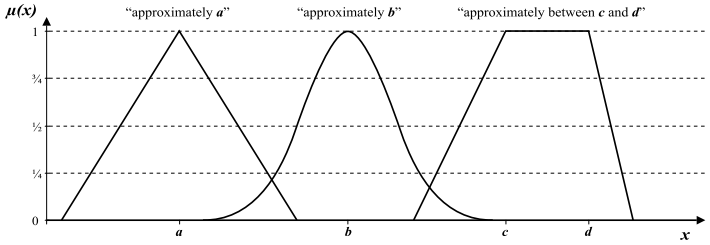


Figure 2.1: Examples of typical fuzzy sets

Membership degrees can be used to represent different kinds of imperfect knowledge, including *similarity*, *preference*, and *uncertainty*. In fuzzy classification rules, fuzzy sets are used to model similarity between attribute values and prototypes, often described by linguistic terms. On the real scale, very common fuzzy sets are so-called *fuzzy numbers* (or fuzzy intervals) that assume a value of one for a single value $a \in \mathbb{R}$ (or interval $[a, b] \subset \mathbb{R}$), and have monotonously decreasing membership degrees with increasing distance from this *core*. Fuzzy numbers can be associated with linguistic terms like, for example, "approximately a". In fuzzy rule based systems, typically parameterized membership functions are used, where these are in most cases either triangular, trapezoidal, or *Gaussian* shaped (cf. Figure 2.1):

$$\mu_{x_0,\sigma}(x) = \exp\left(-\frac{(x - x_0)^2}{2\sigma^2}\right). \tag{2.11}$$

If the complete input range is covered by overlapping fuzzy sets, this is called *fuzzy partition*. If their number is sufficiently small, the fuzzy sets M are usually associated with linguistic terms, e.g. $A_M \in \{$*small*, *medium*, *large*$\}$. In the following, fuzzy sets M, their corresponding fuzzy membership functions μ_M and the associated linguistic terms A_M will be used interchangeably, where the correspondence is clear.

The linguistic terms can be used in the *antecedents* of fuzzy rules, which in most cases are Boolean expressions of simple (fuzzy) clauses. The simple clauses use the individual features $x_1, \ldots, x_{n_d}$, and take the form "x_i IS A", where A is a linguistic term. An example of a simple clause could thus be "x_i IS *large*". The definition of the rules' consequents can vary considerably. Most types of fuzzy if-then systems use linguistic terms in the consequents,

which is close to the original idea of linguistic reasoning. The widely used model by Mamdani and Assilian (1975) uses consequents that take the form "THEN y IS B", where y is the output attribute and B is a linguistic term. In MIMO systems (multiple inputs, multiple outputs), the consequents can also be Boolean expressions of simple clauses.

A contrast to this are the fuzzy systems proposed by Takagi and Sugeno (1985). In these systems, the consequents are functions of the input, i.e. a consequent is defined as "THEN $y = f(\boldsymbol{x})$". In MIMO systems, each rules' consequent can have several independent function definitions for the output variables. The functions f are in most cases polynomials. Obviously, this definition is further away from linguistic reasoning; the model is thus sometimes called *functional* (instead of *logical*) model.

Depending on the rule structure, different inference mechanisms are necessary. Virtually all classification systems based on fuzzy if-then rules—as well as most *neuro-fuzzy* classification systems—use inference mechanisms that are either derived from Takagi-Sugeno or Mamdani-Assilian fuzzy models. Generally, interpretation of (non-fuzzy) rule bases leads to evaluation of individual rules by implication, and conjunctive aggregation of the results. However, neither the Mamdani-Assilian, nor the Takagi-Sugeno model allows such interpretation in a generalized ("fuzzified") logic sense.[3] Alternatively, they can be interpreted on the basis of equality relations, which puts them on the proper semantics of interpolation theory (Klawonn and Kruse, 1993). The antecedents of the fuzzy rules then describe vague prototypes or prototypical cases. Intuitively, this means that the more similar a case is to a prototype, the more similar should the system's output be to some output associated with the prototype. These kinds of fuzzy models have been used in fuzzy control, and are probably the commercially most successful fuzzy systems.

The underlying principle of any controller is to map (real-valued) inputs to (real-valued) outputs. In the following sections, we show how Mamdani-Assilian and Takagi-Sugeno MIMO systems are evaluated, and how they can be used to represented (soft) classification functions $\tilde{g}$.

Mamdani-Assilian Fuzzy Systems

In practice, the antecedents of Mamdani-Assilian fuzzy systems are virtually always restricted to conjunctions of simple clauses. The r-th rule of a

[3] An interpretation in a narrower sense of fuzzy *logic* is possible, but unusual in fuzzy control. The semantics, and the resulting inference calculations are, for example, discussed by Gottwald (1986); Gottwald and Pedrycz (1986); Gebhardt and Kruse (1993).

rule base that describes the relation of n_d inputs $x_1, \ldots, x_{n_d}$ to n_c outputs $\tilde{g}_1, \ldots \tilde{g}_{n_c}$ takes the form

$$\begin{aligned} R_r\text{: IF} \quad & x_1 \text{ IS } A_{1,i_{1,r}} \text{ AND} \\ & x_2 \text{ IS } A_{2,i_{2,r}} \text{ AND } \ldots \text{ AND} \\ & x_{n_d} \text{ IS } A_{n_d,i_{n_d,r}} \\ \text{THEN} \quad & \tilde{g}_1 \text{ IS } B_{1,j_{1,r}} \text{ AND } \ldots \text{ AND } \tilde{g}_{n_c} \text{ IS } B_{n_c,j_{n_c,r}} \\ \text{WITH WEIGHT } & w_r. \end{aligned}$$

The $A_{d,i}$ and $B_{d,j}$ are linguistic terms associated with fuzzy sets $\mu_{d,i}$ and $\nu_{d,j}$. The first index defines on which (input or output) axis the fuzzy sets are defined. The second index enumerates the fuzzy sets on this axis. The indices $i_{d,r}$ are needed, as fuzzy sets can be shared by several rules.[4] However, for simplicity, in the following we use $\mu_{d,r}$ to denote the fuzzy set (or linguistic term) that is used by rule r in dimension d, and $\nu_{j,r}$ to denote the j-th output fuzzy set used in the r-th rule. A rule can additionally be associated with a weight w_r that intuitively specifies the relevance of the rule. However, the use of rule weights are controversial, and thus this part is often omitted. The effects of rule weights are discussed in Sections 2.3.5 and 2.4.

The first step in the evaluation of a fuzzy rule base of Mamdani or any other type is called *fuzzification*. The attribute values $x_i \in X_i$ of the input vector $\boldsymbol{x}$ are in general crisp values. These values are fuzzified by calculating their degree of membership $\mu_{i,j}(x_i)$ to the fuzzy sets defined on that feature and used in the antecedents. These membership values are considered as truth values of the simple clauses "x_i is $A_{i,j}$".

The next step in rule base evaluation is *inference*: the calculation of each rules' influence and combination of these. The *activation* (or *firing strength*) of a rule is defined as the fuzzy truth value of its antecedent, determined by evaluating the Boolean expression of the fuzzified inputs. In the Mamdani case, we only have conjunctions of simple clauses. Conjunctions and disjunctions of fuzzy membership degrees are evaluated by so-called t-norms and t-conorms, respectively:

Definition 2.1 *A* **t-norm** $\top : [0,1]^2 \to [0,1]$ *is a commutative and associative function that satisfies* $\top(a,1) = a$ *and* $a \leq b \Rightarrow \top(a,c) \leq \top(b,c)$.

Definition 2.2 *A* **t-conorm** $\bot : [0,1]^2 \to [0,1]$ *is a commutative and associative function that satisfies* $\bot(a,0) = a$ *and* $a \leq b \Rightarrow \bot(a,c) \leq \bot(b,c)$.

[4] E.g. $i_{2,3}$ specifies the index of the fuzzy set in its enumeration that is used by the third rule and the second dimension; if $i_{d,r} = i_{d,r'}$, the corresponding fuzzy set is shared by rules r and r'.

For $a, b \in \{0, 1\}$, all t-norms (t-conorms) behave like the traditional conjunction (disjunction). For the values in between, however, different behaviors are possible.

Let us use $\top(a_1, \ldots, a_n) = \top(a_1, \top(a_2, \ldots \top(a_{n-1}, a_n) \ldots))$ as a definition of n-ary t-norms. Then we define the activation of a rule:

Definition 2.3 *The* **activation of a rule** r *is calculated as*

$$act_r(\boldsymbol{x}) = \top\left(\mu_{1,r}(x_1), \ldots, \mu_{n_d,r}(x_{n_d})\right).$$

For the common t-norms minimum ($\top_{\min}$), algebraic product ($\top_{\text{prod}}$), and Łukasiewicz t-norm ($\top_{\text{Luka}}$) we get

$$act_r^{\min}(\boldsymbol{x}) = \min\{\mu_{1,r}(x_1), \ldots, \mu_{n_d,r}(x_{n_d})\}, \tag{2.12}$$

$$act_r^{\text{prod}}(\boldsymbol{x}) = \mu_{1,r}(x_1) \cdot \cdots \cdot \mu_{n_d,r}(x_{n_d}), \tag{2.13}$$

$$act_r^{\text{Luka}}(\boldsymbol{x}) = \max\{1 - n_d + \mu_{1,r}(x_1) + \cdots + \mu_{n_d,r}(x_{n_d}), 0\}. \tag{2.14}$$

A number of other t-norms have been proposed to achieve special characteristics. However, they play a minor role in fuzzy classification.

As mentioned earlier, inference in Mamdani systems is a heuristic that cannot be interpreted in a strict logic sense (but, under some restrictions, as an interpolation between fuzzy control points). The intuitive idea is that a rule's influence on the output should increase with its activation level. Therefore, Mamdani suggested to cut the consequent fuzzy sets $\nu_{j,r}$ at the activation level, i.e. to use the minimum of both. More generally, any t-norm can be used:

Definition 2.4 *The* **output of a single fuzzy rule** r *is calculated by a t-norm of consequent fuzzy set and activation:*

$$\nu_{j,r}^{out}(y_j) = \top(act_r, \nu_{j,r}(y_j)).$$

If rule weights $w_r \in [0, 1]$ are used, the output is defined as

$$\nu_{j,r}^{out}(y_j) = \top(\top(act_r, \nu_{j,r}(y_j)), w_r).$$

The individual outputs are then aggregated over all n_k rules of the rule base. Mamdani suggested the maximum operation, however, any t-conorm could be used here:

Definition 2.5 *The* **output of a rule base** *for output j is calculated as disjunctive aggregation of the rules' outputs*

$$\nu_j^{out}(y_j) = \bot(\nu_{1,j}^{out}(y_j), \ldots, \nu_{n_k,j}^{out}(y_j)).$$

The last step in rule base evaluation is *defuzzification*, which extracts a crisp output value from the fuzzy outputs ν_j^{out} of the inference system. The most usual approach to determine a representative value from a fuzzy set is to calculate its *center of gravity* (COG)

$$\tilde{g}_j = COG(\nu_j^{out}) = \frac{\int_{-\infty}^{\infty} \nu_j^{out}(y) \cdot y \, dy}{\int_{-\infty}^{\infty} \nu_j^{out}(y) \, dy}. \tag{2.15}$$

Other defuzzification methods, like *maximum* or *mean of maximum* method, have been proposed for fuzzy controllers to achieve special control behaviors. However, they play a minor role for fuzzy classification.

Takagi-Sugeno Fuzzy Systems

A common traditional approach to controller design exploits that systems are often sufficiently simple that their behavior can be linearized in the working points. The intuitive idea of building non-linear control functions with Takagi-Sugeno fuzzy systems is to vaguely describe system states that can be associated with a simple, preferably linear, control function. Each of such states is described by a fuzzy rule. The task of the inference system is to interpolate between these control functions. The general structure of a Takagi-Sugeno fuzzy rule is

$$\begin{aligned} R_r\text{: IF} \quad & x_1 \text{ IS } A_{1,i_{1,r}} \text{ AND} \\ & x_2 \text{ IS } A_{2,i_{2,r}} \text{ AND } \ldots \text{ AND} \\ & x_{n_d} \text{ IS } A_{n_d,i_{n_d,r}} \\ \text{THEN} \quad & \tilde{g}_1 \text{ IS } f_{1,r}(\boldsymbol{x}) \text{ AND } \ldots \text{ AND } \tilde{g}_{n_c} \text{ IS } f_{n_c,r}(\boldsymbol{x}) \\ \text{WITH WEIGHT } & w_r. \end{aligned}$$

Again, the weight term is optional. The first steps of rule evaluation—fuzzification of the inputs and determination of rule activation—are identical to evaluation in Mamdani systems. However, the disjunctive aggregation of rules' outputs is replaced by calculating a weighted average of the output functions:

Definition 2.6 *The j-th output of a Takagi-Sugeno fuzzy rule based system is calculated as*

$$\tilde{g}_j(\boldsymbol{x}) = \frac{\sum_{r=1}^{n_k} act_r(\boldsymbol{x}) \cdot f_{j,r}(\boldsymbol{x})}{\sum_{r=1}^{n_k} act_r(\boldsymbol{x})}.$$

If rule weights are used, the output is calculated as

$$\tilde{g}_j(\boldsymbol{x}) = \frac{\sum_{r=1}^{n_k} act_r(\boldsymbol{x}) \cdot w_r \cdot f_{j,r}(\boldsymbol{x})}{\sum_{r=1}^{n_k} act_r(\boldsymbol{x}) \cdot w_r}.$$

Intuitively, the more one rule is activated in relation to the other rules, the more the value of its consequent function influences the output. Rules with greater weights also have increased influence. In contrast to Definition 2.4, w_r is not restricted to the unit interval, but can assume arbitrary positive values. Notice that the output from Definition 2.6 is already a crisp value, and hence defuzzification is not necessary in Takagi-Sugeno systems.

Classification with Mamdani or Takagi-Sugeno Fuzzy Systems

Although the definition of a classification function in Section 2.1 allows arbitrary values for the outputs g_j, most approaches restrict them to take values from the unit interval $[0, 1]$. In that case, the vector $(g_1, \ldots, g_{n_c})$ can be interpreted as a possibilistic label. Thus, the linguistic formulation of the consequent of a fuzzy classification rule is usually

R: IF ... THEN class of object IS c,

represented internally by the 1-in-n-encoding

R: IF ... THEN $g_1 = 0, \ldots, g_c = 1, \ldots, g_{n_c} = 0$.

In Mamdani classification systems, this is in most cases modeled by using consequent fuzzy sets $\nu_{j,r}$ that are crisp numbers, i.e. either 0 or 1, represented by membership functions $\mathbb{1}_{\{0\}}$ and $\mathbb{1}_{\{1\}}$. In Takagi-Sugeno systems, the crisp values g_j are represented by output functions that are constants, i.e. $f_{i,r} \equiv g_{i,r}$. It can easily be shown that in that case center of gravity defuzzification of the aggregated output fuzzy set in Mamdani systems (Eq. (2.15)) and weighted average in Takagi-Sugeno systems yield almost the same outputs.[5]

Using these simplifications and Takagi-Sugeno rule evaluation, we define the output of a base of fuzzy classification rules:

[5] They are only different, if several rules have identical consequents. In Mamdani systems by Definition 2.5, these are combined by a t-conorm, and thus get "absorbed". In Takagi-Sugeno systems, such consequents "count double".

Definition 2.7 *Let* $\boldsymbol{R} = \{R_1, \ldots, R_{n_k}\}$ *be a rule base of* n_k *fuzzy if-then classification rules of the form*

R_r*: IF* x_1 *IS* $A_{1,r}$ *AND ... AND* x_{n_d} *IS* $A_{n_d,r}$
THEN class of object IS c_r *WITH WEIGHT* w_r*.*

Let c_j *denote the class corresponding to the* j*-th output* $\tilde{g}_j$*, and let* $\top$ *denote the t-norm used to calculate the activation. Then the outputs of a fuzzy system with* $\top$**-sum inference** *are calculated as*

$$\tilde{g}_j(\boldsymbol{x}) = \frac{\sum\limits_{R_r \in \boldsymbol{R} \wedge c_r = c_j} act_r(\boldsymbol{x}) \cdot w_r}{\sum\limits_{R_r \in \boldsymbol{R}} act_r(\boldsymbol{x}) \cdot w_r},$$

or, using $\top$**-max inference**, *as*

$$\tilde{g}_j(\boldsymbol{x}) = \max_{R_r \in \boldsymbol{R} \wedge c_r = c_j} act_r(\boldsymbol{x}) \cdot w_r.$$

This definition also applies if no rule weights are used. In that case, we set $w_r = 1, \forall r \in \{1, \ldots, n_k\}$. For $\top$, any t-norm can be used. The most common inference schemes for fuzzy classification rules are *prod-sum* and *min-max*.

2.3 Properties of Fuzzy Rules

One reason why fuzzy rules are popular with application experts is the implicit abstraction from numbers. It seems easier to use linguistic terms, where one does not have the need to specify exact values in the first place. However, when a fuzzy classifier is applied to new data, usually a winner-takes-all principle is used to determine a single class to which a tuple is most similar. Therefore, there are always crisp borders between regions of different predicted classes. In this section we discuss the shape of these borders and show which class distributions can be represented by such a system. This section shall give an intuition of what a fuzzy classifier can do and cannot do, for which kind of datasets fuzzy classifiers are suited, and what should be considered when one is created.

In Section 2.3.1, we review some theoretical results on the capabilities of fuzzy classifiers. However, the corresponding proofs assume classifiers that are rather uncommon in practice. In the remaining sections, we discuss the decision boundaries of more realistic fuzzy classifiers. Section 2.3.2 considers the influences of different t-norms used to calculate the rule activation. In

Section 2.3.3, we show that *globally defined* (shared) linguistic terms can lead to axis parallel borders, which is an often unwanted restriction, and can even lead to an equivalence of fuzzy classifiers and lookup table classifiers with hyperbox cells, if complete rule bases are used. One possibility to circumvent these problems is to use *locally defined* fuzzy sets. Section 2.3.4 discusses the linguistic interpretability of such fuzzy sets. Finally the effects of rule weights are discussed in Section 2.3.5. The results are summarized in Section 2.3.6.

2.3.1 Representable Decision Boundaries

There are a number of proofs in fuzzy literature that fuzzy rule based systems are universal approximators. It can be shown for both, Mamdani and Takagi-Sugeno systems, that they can approximate any continuous (Riemann-integrable) function with arbitrary precision. Thus, we can also approximate g_c with arbitrary precision (for a survey, see Kuncheva, 2000a). However, most of these proofs are only interesting as a theoretical result, because they require vast numbers of fuzzy rules which is impractical in realistic applications.

Hence, in this section we restrict ourselves to more realistic rule bases with few rules only. For that case, Klawonn and Klement (1997) showed that in a two-dimensional space any classification border of the form $x_2 = f(x_1)$ (i.e. $g(x_1, x_2) = c_1$, if $x_2 > f(x_1)$, and $g(x_1, x_2) = c_2$, if $x_2 < f(x_1)$) can be represented by two rules only. Although the authors considered only monotonous functions and used several rules to assemble piecewise monotonous functions, it can easily be shown that it is possible to construct a pair of rules that can represent *any* function f. This result holds for any t-norm $\top$ for antecedent evaluation, and for $\top$-sum as well as for $\top$-max inference. In two-dimensional planes, any piecewise closed area can thus be described by fuzzy rules. Therefore the area is split into rectangles such that the boundary in the sections can either be expressed as $x_2 = f(x_1)$ or as $x_1 = g(x_2)$. Each of these rectangles is then represented by two appropriate fuzzy rules.

However, this works only in two dimensions. In higher dimensional domains, the t-norm used plays a role. Klawonn and Klement investigated the capabilities of classifiers with min-max inference. In that case, the decision boundary is located where the activations of the minimally activated rules of each class are equal. Due to the maximum aggregation, the location of the decision boundary depends locally on the activations of two rules only. For each of these rules, due to the minimum t-norm, the activation depends

on the membership degrees of one antecedent fuzzy set only. Thus, the shape of the border at this point is determined by the equality[6]

$$\mu_{i,r}(x_i) = \mu_{j,r'}(x_j). \tag{2.16}$$

If different dimensions i and j are involved, i.e. if $i \neq j$, Eq. (2.16) defines a curve in i and j. As all other dimensions yield higher (or at least equal) membership degrees, they are (locally) negligible, and the curve is extended to a cylindrical hypersurface orthogonal to $X_i \times X_j$. If $i = j$, then Eq. (2.16) holds for a fixed x_i only.[7] In that case, the decision boundary is locally a hyperplane orthogonal to X_i. The fact that fuzzy min-max-classifiers decide locally on the basis of at most two variables has been discussed in detail in (von Schmidt and Klawonn, 1999). The authors conclude that it is not possible to represent arbitrary decision boundaries in more than two-dimensional spaces. However, their claims hold only locally. Although it can be shown that the points with at most two relevant variables are dense in $\boldsymbol{X}$, it does not at all mean that the boundary depends on the same two variables everywhere. Nothing is said about the size of those local areas and the "global" appearance of the boundary. Thus the results of (von Schmidt and Klawonn, 1999) give us little intuition of the general boundary shapes of fuzzy min-max-classifiers.

One common conclusion is that it might be reasonable to replace $\top_{\min}$ with other t-norms offering more flexibility. Klawonn and Klement (1997); von Schmidt and Klawonn (2000) suggest to use the Łukasiewicz t-norm $\top_{\text{Łuka}}$, i.e.

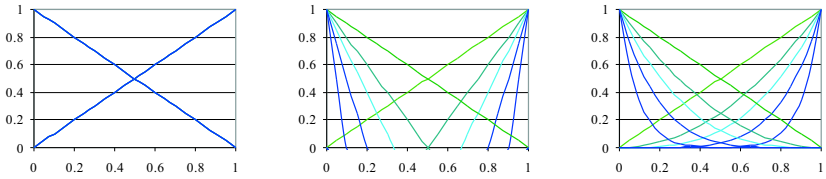
$$\begin{aligned} \top_{\text{Łuka}} : [0,1]^n &\rightarrow [0,1] \\ \top_{\text{Łuka}}(\mu_1, \ldots, \mu_n) &= \max\{1 - n + \sum_{i=1}^{n} \mu_i, 0\}. \end{aligned} \tag{2.17}$$

With appropriately constructed fuzzy sets, $\top_{\text{Łuka}}$-max classifiers allow to represent arbitrary separating hyperplanes

$$\sum_{i=1}^{n} c_i x_i = c \tag{2.18}$$

[6] For our discussion we assume continuous membership functions and t-norms, and thus between regions assigned to different classes there exists a region of equal activation (in general, this region—i.e. the decision boundary—has a thickness of exactly one point).

[7] We assume that $\mu'_{i,r} \neq \mu'_{i,r'}$ in dimension i.



a) $\top_{\min}$ b) $\top_{\text{Łuka}}$ c) $\top_{\text{prod}}$

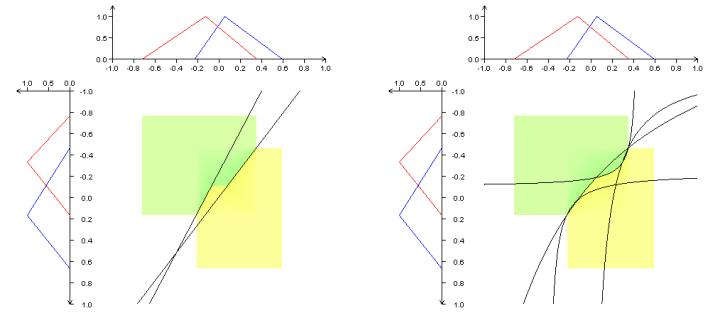
Figure 2.2: Influence of dimensionality on rule activation for different t-norms. The graphs show the activations of two rules with triangular fuzzy sets when we move from one rule center $\boldsymbol{x}$ to the other $\boldsymbol{x}'$. Dimensionalities are 1, 2, 3, 5, and 10, respectively.

in any finite domain, i.e. $x_i \in [a_i, b_i]$. The intuitive idea is to use linear membership functions for the fuzzy sets. Thus the right side of Eq. (2.17) (yielding the activation of a rule) has almost the structure of the left side of Eq. (2.18). The right side is given by a second rule (of another class) with constant membership functions. The trick of the construction of the fuzzy sets is to scale them appropriately, such that $\top_{\text{Łuka}}$ does not go into "saturation", i.e. such that it stays in the range $(0, 1)$. In that way, the decision boundary—i.e. the points of equal activations—has the shape of a hyperplane.

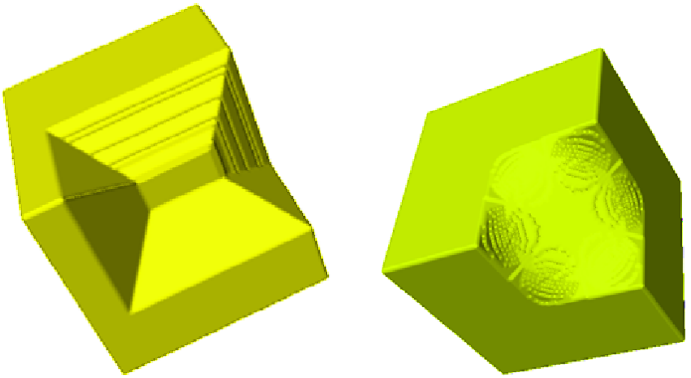
2.3.2 Influence of the t-Norm

Although the fuzzy classifiers used in the proofs need few rules only, they are of little practical use. In both cases, the constructed fuzzy sets differ extremely from those commonly used to represent the linguistic concepts in the antecedents, and thus linguistic interpretability will be lost. In the two-dimensional case, the fuzzy sets are directly derived from the function $x_2 = f(x_1)$ of the decision boundary. However, if we already know f, or at least have some mechanism to represent an arbitrary function in the fuzzy sets, why don't we use it directly? Analogously we can ask, why we might wish to represent the hyperplanes with a $\top_{\text{Łuka}}$-max fuzzy classifier, if this does not increase interpretability, but rather even obscures the structure of the hyperplane?

The original idea of fuzzy classifiers is to describe a functional dependence with linguistic terms, which are therefore usually represented by in-



a) $\top_{\min}$, two dimensions b) $\top_{\text{prod}}$, two dimensions

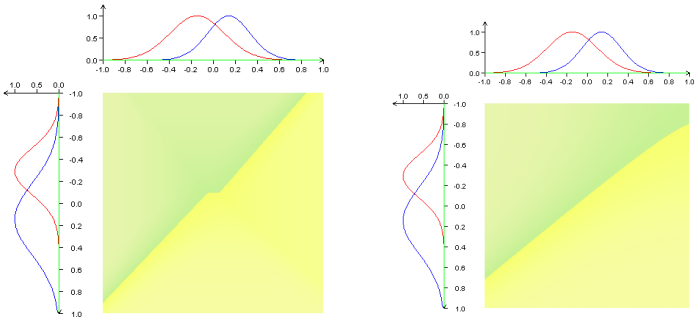


c) $\top_{\min}$, three dimensions d) $\top_{\text{prod}}$, three dimensions

Figure 2.3: Decision boundaries between partially overlapping rules.

terpretable parameterized fuzzy sets (like those in Fig. 2.1). It is common to use triangular membership functions and fuzzy partitions, where the membership degrees of any two neighboring fuzzy sets add up to one. With these assumptions, however, the Łukasiewicz t-norm is rather useless in higher dimensional input domains. With more realistic fuzzy sets, $\top_{\text{Luka}}$ goes rather quickly into saturation, i.e. there are inputs where the rule activation is zero, although all involved membership degrees are greater zero.

Figure 2.2 gives a comparison of the most common t-norms. Let us assume that we have two rules each of which describes a prototype in n_d-dimensional space. Let the cluster centers be $\boldsymbol{x} = (x_1, \ldots, x_{n_d})$ and $\boldsymbol{x}' = (x'_1, \ldots, x'_{n_d})$. Let us further assume fuzzy sets that take their maximum



a) $\top_{\min}$ b) $\top_{\text{prod}}$

Figure 2.4: Decision boundaries with Gaussian membership functions.

value at the center of one prototype and linearly decrease towards the center of the other prototype. Figure 2.2 shows, how the rules' activations change when we move from $\boldsymbol{x}$ to $\boldsymbol{x}'$, and which effect the dimensionality has ($n_d \in \{1, 2, 3, 5, 10\}$). As $\top_{\min}$ is an (actually the only) idempotent t-norm, the conjunction of the $\mu_i(x_i)$ is equal to the individual membership values and dimensionality does not decrease rule activation. In contrast to this, the Łukasiewicz t-norm $\top_{\text{Luka}}$ quickly decays to zero when the dimensionality is higher. The product $\top_{\text{prod}}$ also decays rather quickly. However, for $a > 0$ and $b > 0$ it always assumes positive values $\top_{\text{prod}}(a, b) > 0$, and thus classification is still possible. Thus, in the following we consider $\top_{\min}$ and $\top_{\text{prod}}$ only, as they are more appropriate in practical applications.

The used t-norm has also an important influence on the decision boundaries. Figure 2.3 depicts the decision boundaries between two rules of different classes for $\top_{\min}$ (left column) and $\top_{\text{prod}}$ (right column) for a two-dimensional (upper row) and three-dimensional (lower row) input space. In the three-dimensional plots, we show only one rule, such that the otherwise hidden decision boundary between the rule boxes is exposed. Generally, the borders for $\top_{\text{prod}}$ are smoother than for $\top_{\min}$. For the minimum, the decision boundary is piecewise linear, just as the involved fuzzy sets. For the product, the boundary can be described by hyperbola sections (cf. the continued curves in Figure 2.3b). We discussed this in detail in (Nürnberger et al., 1999b, 2000).

Obviously, apart from the t-norm used, the shape of the fuzzy sets plays a dominant role for the shape of the decision boundary. If we use, for in-

stance, Gaussian membership functions to approximate the fuzzy sets in Figure 2.3b, we get decision boundaries as shown in Figure 2.4. The resulting boundaries are very smooth, in particular in combination with $\top_{\text{prod}}$. Notice that Gaussian membership functions—in contrast to the triangular fuzzy sets—are greater zero at any point, and thus the complete input space is classified by the rules.

2.3.3 Axis Parallelism Through Shared Fuzzy Sets

Fuzzy membership functions in fuzzy rule based systems can be defined either *locally* or *globally*. *Globally* means that a set of membership functions is defined for the rule base, and that the antecedents of the rules link to these definitions. In the *local* case, each rule has individual fuzzy set definitions, which might be different for every rule.[8] Globally partitioning the axes into fuzzy sets generally makes the assignment of linguistic labels easier. Global definitions are thus more commonly used. However, as we show in this section, they also have severe disadvantages. The following lemma and the theorem state the cause of some of the peculiarities.

Lemma 2.1 *Let r and r' be two fuzzy rules having antecedent fuzzy sets μ_i and μ'_i, $i = 1, \ldots, n_d$, and different classes in the consequents. Let μ_i and μ'_i be identical in one dimension i. Then the class boundary between the two rules is parallel to dimension i in that sense that along each line parallel to the unit vector e_i either the same class is predicted or the activations of the two rules are equal.*

Proof: Let us assume w.r.o.g. that i=1, i.e. $\mu_1 \equiv \mu'_1$, and that at point $\boldsymbol{x} = (x_1, \ldots, x_{n_d})$ $act(x_1, \ldots, x_{n_d}) > act'(x_1, \ldots, x_{n_d})$ holds. Let τ and τ' denote the conjunctive aggregation of dimensions $2, \ldots, n_d$, i.e. $\tau = \top(\mu_2(x_2), \ldots, \mu_{n_d}(x_{n_d}))$ and $\tau' = \top(\mu'_2(x_2), \ldots, \mu'_{n_d}(x_{n_d}))$. Then for any t-norm $\top$ the following holds

$$\begin{array}{rrcl} & act(x_1, \ldots, x_{n_d}) & > & act'(x_1, \ldots, x_{n_d}) \\ \Rightarrow & \top(\mu(x_1), \tau) & > & \top(\mu'(x_1), \tau') = \top(\mu(x_1), \tau') \\ \Rightarrow & \tau & > & \tau' \\ \Rightarrow & \forall a \in [0,1] : \top(a, \tau) & \geq & \top(a, \tau') \\ \Rightarrow & \forall x'_1 \in X_1 : \top(\mu(x'_1), \tau) & \geq & \top(\mu(x'_1), \tau') \\ \Rightarrow & \forall x'_1 \in X_1 : act(x'_1, \ldots, x_{n_d}) & \geq & act'(x'_1, \ldots, x_{n_d}). \end{array} \tag{2.19}$$

[8] Globally defined fuzzy sets are also called *linguistic modeling*, to stress that the globally defined fuzzy sets are usually associated with linguistic terms. In contrast to this, locally defined fuzzy sets that are known as *approximative modeling*.

This means that by moving $\boldsymbol{x}$ along dimension X_1, always the same rule (if any) has the maximum activation, and thus we can find a boundary parallel to X_1. □

As t-norms are in general not required to be strictly monotonous, we cannot exclude the case that both activations take the same value. Such *ambiguous points* can occur, if, for example, $\top_{\min}$ is used (cf. Figure 2.5).

A direct consequence of Lemma 2.1 is that rules with antecedents differing in one dimension i only have class boundaries that can be described by sets of lines parallel to any of the other dimensions $j \neq i$. These lines obviously span (one or more) hyperplanes orthogonal to e_i. As Lemma 2.1 holds independently of the used t-norm and especially the shape of the fuzzy sets, some potentially useful degrees of freedom for the class boundary can be lost in the learning or fine-tuning process.

If we consider rule bases with more than two rules, the results can be adopted in principle. As we showed in (Nürnberger et al., 1999b), this leads to axis parallel class borders for complete rule bases. The positions of the borders are determined by the points of intersection of the membership functions. However, the exact shape of the fuzzy sets does not influence the (crisp) classification.

Theorem 2.1 *Let $\mathcal{F}$ be a fuzzy classifier with a $\top$-max inference and globally defined fuzzy sets. If $\mathcal{F}$ has a complete rule base, i.e. if there is one rule for ever possible combinations of fuzzy sets, then the input space $X_1 \times \cdots \times X_{n_d}$ can be partitioned into a set of (possibly degenerate) hypercubes $A_{1,k_1} \times \cdots \times A_{n_d,k_{n_d}}$, such that for any two non-ambiguous points from one hypercube the same rule has the maximal activation.*

Proof: In each dimension i, let $A'_{i,j_i} \subset X_i$ be the subset, where fuzzy set μ_{i,j_i} has the maximal degree of membership, i.e.

$$A'_{i,j_i} = \{x \in X_i | \forall j'_i \neq j_i : \mu_{i,j'_i}(x) < \mu_{i,j_i}(x)\}. \tag{2.20}$$

The Cartesian product $A'_{1,j_1} \times \cdots \times A'_{n_d,j_{n_d}}$ can be associated with a rule

$$\text{R: IF } x_1 \text{ is } \mu_{1,j_1} \text{ AND } \ldots \text{ AND } x_{n_d} \text{ is } \mu_{n_d,j_{n_d}} \text{ THEN } \ldots$$

As the A'_{i,j_i} are pairwise disjunct by definition, the association of Cartesian products and rules is bijective. Let us assume a point $(x_1, \ldots, x_{n_d}) \in A'_{1,j_1} \times$

$\cdots \times A'_{n_d,j_{n_d}}$. It follows that

$$\begin{array}{rrcl} & \forall i \, \forall j'_i : \mu_{i,j_i}(x_i) & \geq & \mu_{i,j'_i}(x_i) \\ \Rightarrow & \top(\mu_{1,j_1}(x_1), \ldots, \mu_{n_d,j_{n_d}}(x_{n_d})) & \geq & \top(\mu_{1,j'_1}(x_1), \ldots, \mu_{n_d,j'_{n_d}}(x_{n_d})) \\ \Rightarrow & act_r(x_1, \ldots, x_{n_d}) & \geq & act_{r'}(x_1, \ldots, x_{n_d}) \end{array} \tag{2.21}$$

If the point $(x_1, \ldots, x_{n_d})$ is non-ambiguous, i.e. if exactly one rule has the maximal activation, it follows that this must be rule R_r, i.e. $act_r > act_{r'}$.

To complete the proof, we have to show that the Cartesian products can be represented by hypercubes and that the whole input space can be partitioned into such hypercubes. Let $A'_{i,0}$ denote the subset of X_i where two or more fuzzy sets take the same value, i.e.

$$A'_{i,0} = X_i \backslash \bigcup_{j_i} A'_{i,j_i}. \tag{2.22}$$

We split each A'_{i,j_i} and $A'_{i,0}$ into (closed or open, possibly degenerate) consecutively numbered intervals A_{i,k_i}. We get

$$A_{i,k_i} \cap A_{i,k'_i} = \emptyset \text{ and } \bigcup_{k_i} A_{i,k_i} = X_i, \tag{2.23}$$

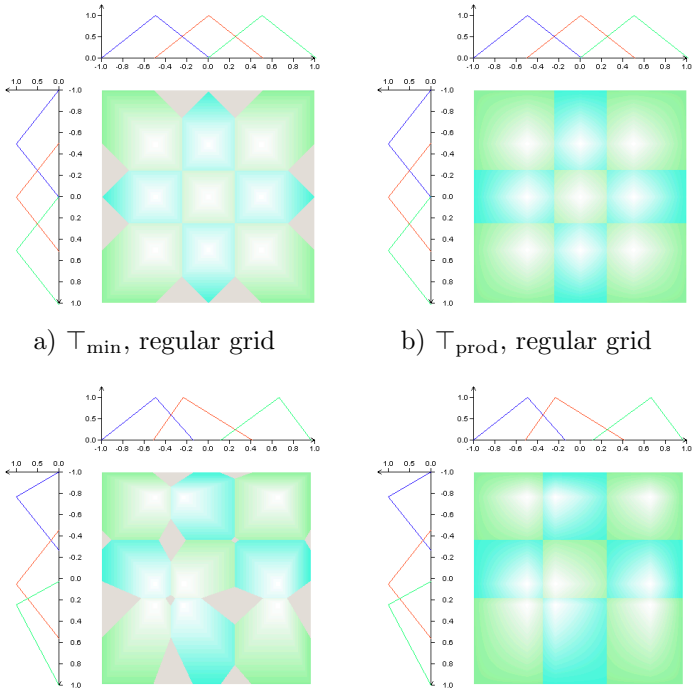
and thus

$$\begin{array}{rcl} X_1 \times \cdots \times X_{n_d} & = & (\bigcup_{k_1} A_{1,k_1}) \times \cdots \times (\bigcup_{k_{n_d}} A_{n_d,k_{n_d}}) \\ & = & \bigcup_{k_1,\ldots,k_{n_d}} (A_{1,k_1} \times \cdots \times A_{n_d,k_{n_d}}). \end{array}$$

□

Figure 2.5 illustrates the results from this section in two dimensions for t-norm $\top_{\min}$ (left column) and the strictly monotonous $\top_{\text{prod}}$ (right column), and for an equidistant fuzzy partition (upper row), and an irregular partition that might, e.g., result from neuro-fuzzy learning (lower row).

Kuncheva (2000b) gives a similar proof that fuzzy classifiers are "look-up tables in disguise", like those in the right column of Figure 2.5. However, the proof has a minor mistake, and therefore ignores the possibility of ambiguous regions, which can occur with non-strict t-norms (e.g. $\top_{\min}$), and which cannot be represented by crisp look-up tables with hyperbox cells. A grid-like structure can be found in all four figures. However, as can be seen in



a) $\top_{\min}$, regular grid b) $\top_{\text{prod}}$, regular grid

c) $\top_{\min}$, irregular grid d) $\top_{\text{prod}}$, irregular grid

Figure 2.5: Complete rule bases (i.e. all nine possible rules) with different t-norms. Notice the ambiguous regions colored in gray in a) and c).

the right column of Figure 2.5, only the fuzzy classifiers with $\top_{\text{prod}}$ have equivalent look-up tables with hyperbox cells. The $\top_{\min}$ classifier with equidistant fuzzy sets in Figure 2.5a has ambiguous regions only on the edges. These would vanish, if the outmost fuzzy sets were left- and right-shouldered, respectively.[9] By shifting the fuzzy sets, however, ambiguous regions can occur at inner points as well (cf. Figure 2.5c). The problem of ambiguity can dramatically decrease classification performance, as no unique class label can be assigned to points lying in these regions.

[9] A fuzzy set μ is called left-shouldered, if $\exists x_1 : (x \leq x_1 \Leftrightarrow \mu(x) = 1)$, and right-shouldered, if $\exists x_1 : (x \geq x_1 \Leftrightarrow \mu(x) = 1)$.

In this section we considered only maximum aggregation of rules. If $\top$-sum inference is used, the situation generally changes, as we have to consider the summation of overlapping rules of the same class. In such areas, a kind of majority voting is performed by the weighted average. Although this can lead to shifted class boundaries, it can be shown that similar effects can occur.

It should be noted that forced axis parallelism as well as ambiguous regions do not only occur in complete rule bases. Shared fuzzy sets—especially in combination with non-strict t-norms—have a general tendency to this undesirable behavior. We thus conclude that local fuzzy set definitions and strict t-norms like $\top_{\text{prod}}$ might be preferable from a perspective of classification abilities.

2.3.4 Interpretability of Local Fuzzy Sets

We argued that local (rule-wise) fuzzy set definitions have several advantages over a global definition. However, it is generally assumed to be easier to find interpretable linguistic labels for globally defined fuzzy sets, and hence these are often preferred. This is certainly true, if the fuzzy partitions have been defined in advance and associated with linguistic terms by domain experts. However, if they are learned from data, the interpretability of globally defined fuzzy sets can also be corrupted.[10]

In fact, we can always transform local and global fuzzy set definitions into each other: trivially, if fuzzy sets are defined globally, we can also define them directly (and possibly redundantly) in the rules. If, vice versa, each of the n_k rules of the rule base uses locally defined fuzzy sets, we could also define them globally and link to them in the rules. In general, this leads to n_k fuzzy sets on each axis, which might be difficult to be described linguistically (particularly if n_k is large). Thus, interpretability is not necessarily a questions of local or global fuzzy set definitions.

There have been some efforts to close the gap—or at least find good tradeoffs—between accuracy and linguistic interpretability of fuzzy rules (e.g., Casillas et al., 2002). One possibility is to use the fuzzy sets as they were learned, and generate more detailed linguistic descriptions for them. Often, linguistic hedges—like "very", "more", "above"—are used to enhance linguistic labels (Marín-Blázquez and Shen, 2002). If the learning of the fuzzy sets is, however, unrestricted, such descriptions might be difficult to extract and of limited readability.

[10] Here, learning from data means either to create fuzzy sets from scratch or modify initially defined fuzzy sets (see Section 3, and especially Section 3.1.2).

An alternative approach to circumvent this problem is to restrict the allowed modifications of locally defined fuzzy sets during learning such that linguistic interpretation is possible afterwards. The commonly used membership function types, like parameterized triangular or Gaussian shapes, can generally be seen as fuzzy numbers with associated linguistic descriptions (see Section 2.2). One basic possibility to support the description with linguistic terms of such fuzzy sets is to define a relative order on them. We could thus define a relation operator *larger* as

$$\mu > \mu' \overset{\text{def.}}{\iff} \exists x^0, \forall x : (x \geq x^0 \Rightarrow \mu(x) \geq \mu'(x)) \wedge (x \leq x^0 \Rightarrow \mu(x) \leq \mu'(x).$$

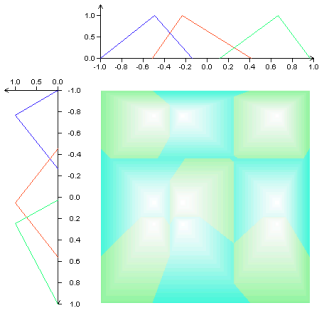
For triangular membership functions, defined by foot points and tip (a, b, c), this leads to the condition

$$\mu_{a,b,c} > \mu_{a',b',c'} \Leftrightarrow (a > a' \wedge b > b' \wedge c > c').$$

For example, the fuzzy sets in Figure 2.5 are ordered according to this condition. We can assign linguistic labels like "small", "medium", "large". If a learning algorithm ensures widths and relative positions of fuzzy sets such that they are ordered, it is generally easy to assign linguistic standard descriptions, or relative descriptions like "slightly smaller than medium".

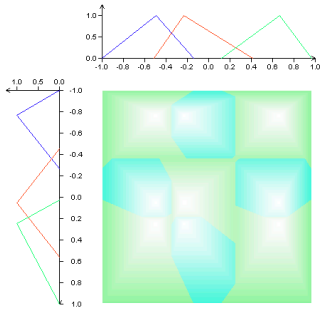
2.3.5 The Role of Rule Weights

The use of rule weights in fuzzy rule bases is controversial. For instance, Nauck and Kruse (1998); Nauck (2000) strongly argue against the use of rule weights, as they might disturb interpretability of fuzzy systems. Their main argument is that in most cases rule weights can be eliminated and incorporated into the membership function. This would lead to systems without weights, but with decreased interpretability, as the new fuzzy sets are in general no longer normal, and different fuzzy sets might represent the same linguistic terms. In contrast to this, Ishibuchi and Nakashima (2000) argue for rule weights. Ironically, they refer to the argumentation in (Nauck, 2000), but turn it 'upside-down': most neuro-fuzzy approaches fine-tune the membership functions, which might disrupt the relation of linguistic terms and fuzzy sets. Hence, they argue that when rule weights can be interpreted as changed fuzzy sets, one could as well leave the fuzzy sets unmodified and fine-tune the weights instead, which is in general the easier task. As the relation of linguistic terms to fuzzy sets is maintained, interpretability is maintained as well.



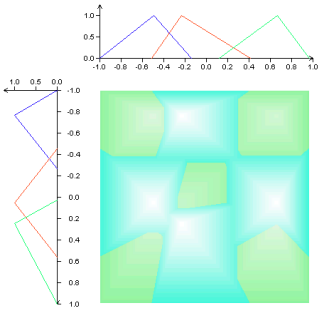
a)

0.99	1.0	0.99
1.0	0.99	1.0
0.98	0.985	0.98



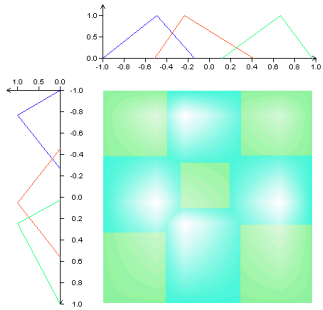
b)

0.99	0.985	0.99
0.985	0.99	0.985
0.98	0.985	0.98



c)

0.6	1.0	0.5
1.0	0.4	1.0
0.5	1.0	0.6



d)

0.6	1.0	0.5
1.0	0.4	1.0
0.5	1.0	0.6

Figure 2.6: The effects of rule weights. The tables show the weights of the nine rules, Figures a), b) and c) use min-max, Figure d) uses prod-max inference. Figures a) and b): Even very small changes lead to assignment of the ambiguous regions to one or the other class (and thus major changes compared to Figure 2.5c); Figures c) and d): Only minor changes of the decision boundaries in spite of major weight changes: compare Figures a) and c), and Figures d) and 2.5d), respectively.

Obviously, these argumentations are contradicting. Interpretability is a rather subjective and imprecise concept, and thus it is hard to say that one is "right". It should, however, be remarked that any (at least any positive) rule weights can be eliminated and incorporated into the membership functions, but only very specific changes of the fuzzy sets have their counterparts in weight changes.

As weights are always defined for individual rules, and as (as argued in Nauck, 2000) they can be seen as equivalent to modifications on the fuzzy sets, we partially circumvent the problems of shared (i.e. globally defined) fuzzy sets (cf. Section 2.3.3). Especially the possibly occurring ambiguous regions for non-strict t-norms can be eliminated using rule weights: even the slightest change of rule weights assigns the ambiguous region to the higher weighted rule (cf. Figure 2.6). It is, however, questionable, whether such impacts of slight changes can be considered interpretable. As can be seen in Figure 2.6c, further changes of the rule weights have little effect. For strictly monotonous t-norms like $\top_{\text{prod}}$ the effects are even smaller.

Apart from the relaxation of the hyperbox cell structure of complete rule bases, Ishibuchi argues that rule weights play an important role when general and specific rules shall be used. General rules, i.e. rules that contain "don't cares" in the antecedent, are an important means to reduce the number of rules in high dimensional domains. They can for example result from pruning techniques (e.g., Klose and Nürnberger, 1999). As adding terms to the antecedents can only decrease rule activation, unweighted specific rules are always dominated by the more general rules. By using rule weights, this problem can be ameliorated.

We considered this aspect in (Nürnberger et al., 1999a). However, instead of heuristically choosing rule weights, we suggest a mathematically more founded motivation. The problem is that general rules cover a greater hypervolume and a mechanism is needed to balance this. If we interpret the fuzzy sets as probability density functions, we have to normalize their integral to one. This can be done by rule weights. Because in this case the rule weights have a clear probabilistic interpretation, interpretability of the system is not disturbed. This alternative interpretation of rule weights is discussed in Section 2.4.

2.3.6 Discussion

Generally, fuzzy rule based classifiers are able to approximate any decision boundary with any precision. As, however, for the sake of interpretability and readability the number of rules and the shape of membership functions

is usually restricted, the resulting fuzzy classifiers have only limited representation capabilities in practice. With the commonly used parameterized, unimodal membership functions (i.e. fuzzy numbers), each rule vaguely describes a cluster in the input space.

We have argued for t-norms that yield positive values for positive inputs, as opposed to $\top_{\text{Luka}}$, which for higher dimensional rules easily yields zero activations in spite of positive membership degrees. For strict t-norms, ambiguous regions do not occur, and hence $\top_{\text{prod}}$ can be preferable to $\top_{\text{min}}$. Additionally, the decision boundaries for $\top_{\text{prod}}$ are often smoother and more "natural"—particularly, when prod-sum inference is combined with Gaussian membership functions.

Globally defined fuzzy sets are often preferred in fuzzy classifier designs, as they are considered to be easier described with linguistic terms. However, global definitions bear the pitfall that decision boundaries might be restricted to be axis parallel. In the extreme case of a complete rule base, this results in an equivalence of fuzzy classifiers and lookup tables with hyperbox cells. This can be avoided by using local fuzzy set definitions. To maintain the linguistic interpretability of (locally or globally defined) fuzzy sets during learning, they have to meet certain conditions. In most cases, linguistic descriptions can also be found for (a sufficiently small number of) locally defined fuzzy sets, if the fuzzy sets are ordered as defined in Section 2.3.4. Under this condition, locally defined fuzzy sets allow additional degrees of freedom without sacrificing too much interpretability.

Rule weights can also interfere with interpretability. In Section 2.3.5, we reviewed some of the arguments of flexibility versus interpretability. The next section describes an alternative interpretation of rule weights, which gives them a probabilistic foundation.

2.4 Probabilistic Interpretation of Fuzzy Classification Rules

There are obvious structural similarities between fuzzy classification rules and probabilistic classifiers like, for example, naïve Bayes or mixtures of Gaussians classifiers. In fact, they can be shown to perform identical calculations under some restrictions. Following and extending our results from (Nürnberger et al., 1999a), we show in the following sections that any naïve Bayes classifier can be represented by a fuzzy classifier. Supplementary, here we prove that—vice versa—any fuzzy classifier with t-norm $\top_{\text{min}}$ can

be represented as a naïve Bayes classifier.

In (Nürnberger et al., 1999a), our main motivation to transform a naïve Bayes classifier into a (neuro-) fuzzy classifier was to improve the classification performance of the former by the iterative learning algorithms of the latter. In the context of this thesis, the results are of interest for a different reason: showing that fuzzy classifiers can (under some weak constraints) be transformed into naïve Bayes classifiers gives them a sound probabilistic interpretation. We will need this probabilistic interpretation of the fuzzy sets when we derive our semi-supervised algorithm in Chapter 6.

The following section gives a brief introduction to naïve Bayes classifiers. Sections 2.4.2 and 2.4.3 describe how a naïve Bayes classifier can be represented by a fuzzy classification system, and vice versa.

2.4.1 Naïve Bayes Classifiers

Naïve Bayes classifiers are an old and well-known type of classifiers (Good, 1965; Duda and Hart, 1973). They use a probabilistic approach that tries to compute for a given tuple $\boldsymbol{x} \in \boldsymbol{X}$ the conditional probabilities

$$P(C = c \mid \boldsymbol{x}) = P(C = c \mid X_1 = x_1, \ldots, X_{n_d} = x_{n_d}) \tag{2.24}$$

for classes $c \in \mathcal{C}$. For a given example it predicts that class c with the maximum posterior probability. In most cases, it is not possible to store the conditional probabilities for any possible instantiation of $\boldsymbol{x}$, e.g. in a lookup table. For numeric attributes, this is obvious, and parameterized density functions have to be used. But even in case of attributes with finite domains (like symbolic attributes), the number of possible attribute combinations grows exponentially with the number of dimensions, which makes a lookup table approach infeasible in general. Therefore, Bayes rule is used to invert the conditional probabilities[11]

$$\begin{aligned} & P(C = c \mid X_1 = x_1, \ldots, X_{n_d} = x_{n_d}) \\ = \; & \frac{p(X_1 = x_1, \ldots, X_{n_d} = x_{n_d} \mid C = c) \cdot P(C = c)}{p(X_1 = x_1, \ldots, X_{n_d} = x_{n_d})} \end{aligned} \tag{2.25}$$

The probability density function $p(X_1 = x_1, \ldots, X_{n_d} = x_{n_d})$ in the denominator must be strictly positive for this inversion to be possible. However, for

[11] For simplicity, we always use a probability density function p, 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.

a given object to be classified, it must be greater zero, as we encountered at least one instance with that attribute combination. Moreover, we are mostly interested in the class ranking, and thus can completely neglect this denominator. If needed, its influence can always be restored by normalizing the distribution on the classes. It follows that we need only to consider

$$\begin{aligned} &P(C = c \mid X_1 = x_1, \ldots, X_{n_d} = x_{n_d}) \\ &= \frac{1}{S}\, p(X_1 = x_1, \ldots, X_{n_d} = x_{n_d} \mid C = c) \cdot P(C = c), \end{aligned} \tag{2.26}$$

with a normalization factor S that is constant for a fixed $\boldsymbol{x}$. However, this inversion does not decrease the size of the probability space. Therefore, the crucial assumption is made that, for a fixed value of the class attribute, any attribute X_d is independent of any other. Assuming that knowing the class is enough to determine the probability distribution for an attribute independently of any other attributes is a pretty strong, easily violated, and thus "naïve" assumption. However, this assumption allows us to simplify the above equation to

$$\begin{aligned} P(C = c \mid X_1 = x_1, \ldots, X_{n_d} = x_{n_d}) = \\ \frac{1}{S} \cdot p(X_1 = x_1 \mid C = c) \\ \cdot \qquad \ldots \\ \cdot p(X_{n_d} = x_{n_d} \mid C = c) \cdot P(C = c). \end{aligned} \tag{2.27}$$

This is the fundamental formula underlying naïve Bayes classifiers. For symbolic attributes X_d the conditional probabilities $P(X_d = x_d \mid C = c)$ can be stored as conditional probability tables, which is feasible for single attributes. Numeric attributes may be discretized and then treated like symbolic attributes (Dougherty et al., 1995). However, they are usually assumed to have a probability density that can adequately be described by a Gaussian function and hence only the expected value and the variance of the normal distribution need to be stored in this case. Naïve Bayes classifiers are induced from a dataset $\mathcal{D}$ of sample cases by estimating the conditional probabilities or probability densities by maximum likelihood estimation.

2.4.2 Converting Naïve Bayes into Fuzzy Classifiers

In this section we prove that any naïve Bayes classifier can be converted into an equivalent fuzzy classifier. The reverse direction, i.e. the proof that any fuzzy classifier can be transformed into a (possibly *extended*) naïve Bayes classifier, is shown in the subsequent section. Both proofs are constructive.

Theorem 2.2 *Any naïve Bayes classifier can be represented by a fuzzy classification system, which uses either prod-sum or prod-max inference.*

Proof: In contrast to probability densities, which can assume any positive value, fuzzy membership functions are restricted to the unit interval $[0, 1]$. Additionally, fuzzy membership functions are usually required to have at least one element which fully fulfills the concept described by the associated linguistic term, i.e. $\exists x : \mu(x) = 1$. Therefore, we define factors from the conditional probability density functions p for each attribute X_d given the class c

$$w^v_{c,d} := \sup_{x_d}\{p(X_d = x_d \mid C = c)\}, \tag{2.28}$$

and use these to define membership functions $\mu_{c,d}$ as

$$\mu_{c,d}(x_d) := \frac{1}{w^v_{c,d}} \cdot p(X_d = x_d \mid C = c). \tag{2.29}$$

Furthermore, weights w^p_c are set to the prior probability of class $c \in \mathcal{C}$:

$$w^p_c := P(C = c), \tag{2.30}$$

and a rule weight w_c is defined as the product of these weights

$$w_c := w^p_c \cdot \prod_{d=1}^{n_d} w^v_{c,d}. \tag{2.31}$$

For each class $c \in \mathcal{C}$ we define a fuzzy rule

$$\begin{array}{rlllll} R_c\text{: IF} & x_1 & \text{IS} & \mu_{1,c} \text{ AND } \ldots \text{ AND} \\ & x_{n_d} & \text{IS} & \mu_{n_d,c} \\ \text{THEN} & \text{class} & \text{IS} & c \text{ WITH WEIGHT } w_c. \end{array}$$

As we define only one rule per class, the results of prod-max and prod-sum inference are equal (except for the normalizing denominator, see Definition 2.7). Here, we complete the proof for prod-max inference:

$$\begin{aligned} \tilde{g}_c(\boldsymbol{x}) &= w_c \cdot act_{R_c}(\boldsymbol{x}) = w^p_c \cdot \prod_{d=1}^{n_d} w^v_{c,d} \cdot \mu_{c,d}(x_d) \\ &= P(C = c) \cdot \prod_{d=1}^{n_d} p(X_d = x_d \mid C = c) \\ &= P(C = c \mid X_1 = x_1, \ldots, X_{n_d} = x_{n_d}) \cdot S. \end{aligned} \tag{2.32}$$

Apart from constant S—which is negligible, as we mentioned above—the two models obviously perform the same calculations. □

2.4.3 Converting Fuzzy into Naïve Bayes Classifiers

The reversal of the Theorem 2.2 is not directly possible. However, we can prove the following:

Theorem 2.3 *A fuzzy classifier with prod-sum or prod-max inference can be transformed into an equivalent naïve Bayes classifier, if each class is described by exactly one rule.*

Proof: Let us consider a rule base with one rule R_c per class $c \in \mathcal{C}$. In that case the reverse transformation is done similar to the proof above. When we want to interpret fuzzy membership degrees as probability densities, we have to normalize them such that the integral (or sum, in case of discrete domains) over all possible values is one. Therefore, here the normalizing constant is

$$v_{c,d} := \int_{X_d} \mu_{c,d}(x)dx, \tag{2.33}$$

and the conditional probability density functions are defined as

$$p(X_d = x \mid C = c) := \frac{1}{v_{c,d}}\,\mu_{c,d}(x). \tag{2.34}$$

As the prior probabilities must sum up to one, i.e. $\sum_{c\in\mathcal{C}} P(C = c) = 1$, we can define them from the rule weights w_c as

$$P(C = c) = w_c \cdot \prod_{d=1}^{n_d} v_{c,d} \cdot \frac{1}{k}, \tag{2.35}$$

with a normalizing constant

$$k = \sum_{c\in\mathcal{C}} w_c \cdot \prod_{d=1}^{n_d} v_{c,d}. \tag{2.36}$$

With these definition we get

$$\begin{aligned} w_c \cdot act_{R_c}(\boldsymbol{x}) &= w_c \cdot \prod_{d=1}^{n_d} \mu_{c,d}(x_d) \\ &= w_c \cdot \prod_{d=1}^{n_d} v_{c,d} \cdot p(X_d = x \mid C = c) \\ &= k \cdot P(C = c) \cdot \prod_{d=1}^{n_d} p(X_d = x \mid C = c). \end{aligned} \tag{2.37}$$

As we have one rule per class, again prod-sum and prod-max inference yield equivalent results. For prod-sum inference, we have to normalize Eq. (2.37):

$$\begin{aligned}
\tilde{g}_c(\boldsymbol{x}) &= \frac{w_c \cdot act_{R_c}(\boldsymbol{x})}{\sum_{c' \in \mathcal{C}} w_{c'} \cdot act_{R_{c'}}(\boldsymbol{x})} \\
&= \frac{k \cdot P(C = c) \cdot \prod_{d=1}^{n_d} p(X_d = x \mid C = c)}{\sum_{c' \in \mathcal{C}} k \cdot P(C = c') \cdot \prod_{d=1}^{n_d} p(X_d = x \mid C = c')} \\
&= \frac{P(C = c) \cdot \prod_{d=1}^{n_d} p(X_d = x \mid C = c)}{\sum_{c' \in \mathcal{C}} P(C = c') \cdot \prod_{d=1}^{n_d} p(X_d = x \mid C = c')} \\
&= P(C = c \mid X_1 = x_1, \ldots, X_{n_d} = x_{n_d}). \qquad (2.38)
\end{aligned}$$

Hence, the fuzzy classifier output $\tilde{g}_c(\boldsymbol{x})$ is equal to the output of a naïve Bayes classifier as constructed by Eq. (2.33) to Eq. (2.36). □

The proof shows that any fuzzy classifier with one rule per class can be transformed into a naïve Bayes classifier. However, the strength of fuzzy rule bases lies in the possibility to describe more complex classes with several rules. If we consider such a rule base where classes are described with more than one rule, they cannot be handled directly by a naïve Bayes classifier. However, it is possible to split the classes into *subclasses* (or *components*). Each subclass then corresponds to the examples described by one fuzzy rule, which—according to Theorem 2.3—can be represented by a naïve Bayes classifier. The posterior probabilities of these subclasses must be combined to yield the classes' probabilities. For this purpose, we define an *extended naïve Bayes classifier*.

Definition 2.8 *An extended naïve Bayes classifier is a tuple* $(nb, \mathcal{C}, \mathcal{C}', k)$*, where* $\mathcal{C}$ *is a set of classes, and* $\mathcal{C}'$ *a set of subclasses of* $\mathcal{C}$*,* $k : \mathcal{C}' \to \mathcal{C}$ *is a surjective mapping that maps the subclasses* $c' \in \mathcal{C}'$ *to classes* $c \in \mathcal{C}$*, and* nb *is a naïve Bayes classifier that estimates* $P(C' = c' \mid \boldsymbol{X} = \boldsymbol{x})$*. The extended naïve Bayes classifier calculates the conditional probability of a class given the attribute values as*

$$\begin{aligned}
P(C = c \mid \boldsymbol{X} = \boldsymbol{x}) &= P\Big(\bigvee_{c' \in \mathcal{C}' : k(c') = c} (C' = c') \mid \boldsymbol{X} = \boldsymbol{x}\Big) \\
&= \sum_{c' \in C' : k(c') = c} P(C' = c' \mid \boldsymbol{X} = \boldsymbol{x}).
\end{aligned}$$

In the extended naïve Bayes classifier the conditional class probabilities are no longer independent, i.e. for $i \neq j$, it does in general not hold that

$P(C = c \mid X_i = x_i, X_j = x_j) = P(C = c \mid X_i = x_i) \cdot P(C = c \mid X_j = x_j)$. However, the underlying classifier is still "naïve".

Based on this definition, we can reverse Theorem 2.2. However, as the inference mechanism must now combine the results of several rules for one class, we have to restrict the type of fuzzy classifier to prod-sum inference.

Theorem 2.4 *Any fuzzy rule based classifier with prod-sum inference according to Definition 2.7 can be transformed into an equivalent extended naïve Bayes classifier* $(nb, \mathcal{C}, \mathcal{C}', k)$.

Proof: Let there be one subclass for each of the n_k rules $R_1, \ldots, R_{n_k}$, i.e. $\mathcal{C}' = \{c'_1, \ldots, c'_{n_k}\}$, and let the mapping k be defined from the rules' consequents: $k(c'_j) := c_{R_j}$. Let nb be a naïve Bayes classifier for C', constructed from the rule base as shown in the proof of Theorem 2.3.

We can rewrite the output of the resulting extended naïve Bayes classifier as follows:

$$\begin{aligned} P(C = c \mid \boldsymbol{X} = \boldsymbol{x}) &= \sum_{c' \in \mathcal{C}' : k(c') = c} P(C' = c' \mid \boldsymbol{X} = \boldsymbol{x}) \\ &= \sum_{c' \in \mathcal{C}' : k(c') = c} \frac{P(C' = c') \cdot \prod_{d=1}^{n_d} p(X_d = x_d \mid C' = c')}{\sum_{c'' \in \mathcal{C}'} P(C' = c'') \cdot \prod_{d=1}^{n_d} p(X_d = x_d \mid C' = c'')} \\ &= \frac{\sum_{c' \in \mathcal{C}' : k(c') = c} w_{c'} \cdot act_{R_{c'}}(\boldsymbol{x})}{\sum_{c' \in \mathcal{C}} w_{c'} \cdot act_{R_{c'}}(\boldsymbol{x})} = \tilde{g}_c(\boldsymbol{x}). \end{aligned} \tag{2.39}$$

Thus, the output of the constructed extended naïve Bayes classifier is equal to the output of the fuzzy classifier. □

In addition to theorem 2.2, it should be noted that also any *extended* naïve Bayes classifier has an equivalent prod-sum inference fuzzy classifier, which has one rule per subclass, i.e. in general several rules per class.

2.4.4 Discussion

We have shown that any naïve Bayes classifier can be transformed into a corresponding fuzzy rule based classifier, and that any fuzzy classifier with prod-sum inference can be transformed into a (possibly extended) naïve Bayes classifier.

Although they can be transformed into each other, it should be noted that naïve Bayes classifiers and fuzzy classifiers will in general considerably differ when trained from data. This is due to the different optimization

criteria: naïve Bayes classifiers perform a maximum likelihood estimation of the probability distribution (under the crude assumptions of conditional independence), whereas most learning algorithms for fuzzy systems[12] try to optimize the performance, and thus the decision boundaries between classes. In (Nürnberger et al., 1999a), we improved the performance of naïve Bayes classifiers by transforming them into fuzzy classifiers and applying neuro-fuzzy learning techniques.

Nonetheless, the equivalence of the approaches gives fuzzy classification systems an alternative sound probabilistic basis. As we discussed in Section 2.3.5, rule weights are usually hardly justifiable. In the transformations from naïve Bayes to fuzzy classifiers, and vice versa, rule weights get a proper interpretation. As we can see from Eq. (2.31) and Eq. (2.35), they serve two purposes. On the one hand, they are used to normalize the size and thus influence of individual rules, and on the other hand they reflect the prior probabilities of (sub-)classes.

The use of (fuzzy) rule weights as (probabilistic) normalization factors might seem questionable, and it might seem at first that rule weights are necessary to allow transformation of a fuzzy classifier. This is, however, not the case. It should be noted that Theorem 2.4 allows to transform *any* fuzzy rule based classifier with prod-sum inference into an (extended) naïve Bayes classifier with equivalent outputs—including fuzzy classifiers without rule weights (i.e. $w_j = 1, j = 1, \ldots, n_k$).

The most relevant conclusion from our considerations is that fuzzy membership functions can—after appropriate normalization—be interpreted as conditional probability density functions. This will be important in Chapter 6, as it allows us to combine classifiers based on fuzzy set theory and quality measures based on probability theory.

2.5 Conclusions

In this chapter, we reviewed classification based on fuzzy if-then rules. Although the application of such fuzzy classifiers is rather intuitive, there are a number of variants, and some pitfalls that should be avoided to get interpretable, yet flexible rule bases. Therefore, we discussed capabilities of fuzzy classifiers and the influence of alternatives in the classifier design. One goal was to motivate the choices of the fuzzy classifier model that we make in Chapter 6 when we extend fuzzy classification to semi-supervised learning.

[12] See Chapter 3

As we discussed in Section 2.3.2, it is advantageous, if t-norms are strict, i.e. continuous and strictly monotonous. Being continuous is a prerequisite for smooth decision boundaries. The strict monotonicity ensures three things. First, such t-norms assume positive (non-zero) values for any positive arguments, which is important for the coverage of the input space, particularly if its dimensionality is high. In contrast, the practical applicability of the non-strict t-norm $\top_{\text{Luka}}$ is rather limited, because it leads to a quick decay of rule activations to zero. Second, ambiguous regions cannot occur with strictly monotonous t-norms—in contrast to, e.g., $\top_{\text{min}}$. And third, the membership degrees in all dimensions affect the rule activation in any point, and thus t-norms like $\top_{\text{prod}}$ generally yield smoother, less "jaggedly" looking decision boundaries than, e.g., $\top_{\text{min}}$. This is particularly true, if $\top_{\text{prod}}$ is used in connection with Gaussian functions.

We showed that linguistic modeling, i.e. global definitions of fuzzy sets can unintendedly restrict their decision boundaries to be axis parallel. Especially if complete rule bases are defined, fuzzy classifiers can become equivalent to lookup tables. Thus less rules, with local fuzzy set definitions are preferable to achieve higher flexibility. Usually, there is a tradeoff between flexibility and interpretability. However, we argued that locally defined fuzzy sets are still interpretable if the number of rules is reasonably small and some restrictions are placed on the fuzzy membership functions.

If we have only one rule per class, $\top$-sum and $\top$-max inference yield equivalent results. Differences only occur, where rules of one class overlap. In such cases it is intuitive that the influences of several rules predicting the same class should be weighted higher, as it is done in $\top$-sum inference. Although the differences are usually only subtle, in many cases $\top$-sum yields slightly smoother boundaries.

Hence, there are a number of good reasons to use prod-sum inference. If we use prod-sum inference, this allows an alternative, probabilistic interpretation of membership functions and rule weights. This result is very important, as our semi-supervised approach in Chapter 6 requires interpretation of the membership degrees as probability densities.

Chapter 3

Extracting Fuzzy Rules from Data

In the preceding chapter we described how fuzzy rules are used for classification and which properties they have. So far, the rule bases have been considered as given. This chapter deals with the aspects where such fuzzy rules come from, and especially how they can be generated from example data.

One basic method to obtain fuzzy rules is to create them manually, i.e. an expert defines linguistic terms with adequate fuzzy sets on the attributes and describes the classes by linguistic rules. This method stands out from the other methods presented in this chapter as it is the only method that does not use example data. It is still noteworthy because many other models, like for instance, neural networks or statistical classifiers, do (in general) not allow manual creation at all. Fuzzy rules are commonly said to be adequate to model the experts' prior knowledge, as it is one explicit goal of fuzzy rule bases to serve as an interface between expert knowledge and machine representation. However, even if a fuzzy rule base has been constructed by hand, it is common—and very often necessary—to fine-tune and check the rule base against some example data.

Fuzzy rules are often seen not only as a tool to classify new datasets, but also as a tool for exploratory data analysis. In that case, transformation of example data into interpretable rules is desirable. The task of extracting rules (or other classification models) from data is commonly called learning. By far the most common learning paradigm in machine learning is

supervised learning. The input for the supervised learning task consists of a dataset of tuples together with corresponding class labels. The learning task is to generalize the regularities found in the labeled example data.

The second important learning paradigm is unsupervised learning, i.e. learning from example data without known class labels. The general task of unsupervised learning is to reveal inherent structures in the data. It is often assumed that the data is divided into a number of (a priori known or unknown) classes, and the learning task is to reconstruct the class information from the distribution of the example data. This task is known as cluster analysis.

In both cases, learning is inductive, i.e. knowledge is generated by generalizing from specific cases (Mitchell, 1997). There exist both, supervised and unsupervised, approaches to extract fuzzy rules from data. In the next sections we present some of the most important methods.

3.1 Supervised Induction of Fuzzy Rules

There is a variety of methods that have been proposed to induce fuzzy rules from data. They differ in the kind of extracted rule bases (e.g. whether they use global or local fuzzy set definitions) and in the extent of a priori fixed parts of the rule base (e.g. fixed fuzzy sets or fixed rules). In the following sections we sketch the most common strategies, namely

- structure-based rule extraction with given fuzzy sets,
- fuzzy set (parameter) tuning given a rule base,
- identification of fuzzy sets *and* rules with grid-like structure, and
- formation of fuzzy graph-like rules.

Genetic or evolutionary algorithms are also often discussed for optimizing or creating fuzzy systems. One advantage of evolution strategies is their ability to modify and optimize model structure and parameters, whereas most optimization strategies can only adapt model parameters. As we discuss in Section 6.1, this will be of some importance for the semi-supervised approach proposed in this thesis. Therefore, the induction of fuzzy rules by evolutionary algorithms will take its own chapter (cf. Chapter 4).

3.1.1 Structure-Based Approach by Wang&Mendel

If the attributes are partitioned into linguistic terms and corresponding fuzzy sets, an efficient and widely used algorithm to extract rules is that proposed by Wang and Mendel (1992). The algorithm extracts rules in conjunctive normal form with all attributes occurring in the antecedents. Because it uses globally defined fuzzy sets, the number of possible different antecedents, i.e. the number of rules in the complete rule base, is equal to the product of the number of fuzzy sets in each dimension. Of these theoretically possible rules, the algorithm creates those that yield the maximal activation for at least one tuple of the example dataset.

In its basic form, the algorithm needs just one run over the training data to find these rules. As t-norms are monotonous, the antecedent of the maximally activated rule can be constructed from the fuzzy sets that assume the maximal membership degrees in the individual dimensions. Additionally, the class of the pattern is registered for the rule. If the antecedent already exists in the rule base, only the pattern's class is counted for this rule. After all patterns have been presented, the consequents of the rules are set to the corresponding most frequent classes.

As this algorithm considers only the regions of maximal membership, the input space is structured by a multi-dimensional grid. Thus, a single pattern that falls into a grid cell triggers creation of the corresponding rule. Partial overlapping of adjacent fuzzy rules is not considered in this basic algorithm. As the number of grid cells – and with it the number of possible rules – grows exponentially with increasing dimensionality, the algorithm often produces too many, too specific rules. Therefore, a second rule learning stage is often used to choose a representative subset of rules.

3.1.2 Fuzzy Set Learning with Fixed Rules

The algorithm by Wang&Mendel expects predefined fuzzy membership functions on the attributes. These can, for example, be given by domain experts. In many cases, it is difficult to define appropriate fuzzy partitions beforehand. This is especially true if rule learning is applied in a data mining sense, and the goal is to extract previously unknown knowledge. If knowledge about the attributes (and the specific classification task) is not sufficient to define suited fuzzy sets, it is common to define fuzzy partitions equidistantly on the attributes and associate them with linguistic terms like, e.g., *small, medium, large*.

There are other approaches to find appropriate partitions, e.g. by (fuzzy)

clustering. This can be done either separately for each attribute, on the combined attributes of the input space, or even in input and output space together (the latter is more common for fuzzy control). Fuzzy clustering can not only be used to define fuzzy sets, but also for the (unsupervised) extraction of fuzzy rules, as shown in Section 3.2.4.

In any case—no matter whether the fuzzy sets are defined manually, by equidistant partitioning or by clustering—the performance of fuzzy rules can usually be significantly increased by fine-tuning the underlying fuzzy sets. Normally, parameterized membership functions are used. The optimization task is then to adjust the parameters such that some performance measure (e.g. classification rate) is maximized. There are a number of methods from numerics that have been proposed for this task. However, many of the common methods, as for example, gradient descent, rely on differentiable functions. In case of fuzzy rule bases, the widely used min-max-inference and triangular fuzzy sets cause problems due to non-differentiable points and plateau areas. These problems have been investigated by Eitzinger (2001). His proposal uses so-called *bundle methods* (Schramm and Zowe, 1992) that can deal with such cases.

Other approaches use work-arounds. For example, they use differentiable membership functions like Gaussians, and use prod-sum inference, or replace minimum and maximum operation by fuzzy variants that are differentiable, e.g. the *softmax*

$$\text{softmax}(x_1, \ldots, x_n) = \frac{\sum_i x_i \cdot e^{\alpha x_i}}{\sum_i e^{\alpha x_i}}, \tag{3.1}$$

where α determines how closely softmax approximates max. As α approaches ∞, softmax approaches the maximum function, for $\alpha = 0$, softmax calculates the mean, and if α approaches $-\infty$, softmax approaches the minimum.

Some approaches use heuristics that simply ignore such areas, and still yield reasonable results in most cases. One such example is NEFCLASS that uses simple fuzzy set modification rules (Nauck and Kruse, 1997).

An important aspect of membership function tuning are the semantics of the fuzzy sets. Especially if the fine-tuning is used after building the rule base, and thus after fuzzy sets have been associated with linguistic terms, the correspondence of such linguistic terms and underlying fuzzy sets should be maintained. A variety of constraints is thus usually required in the learning process, e.g. fuzzy sets must keep their relative order, must overlap, or add up to one (see Nauck and Kruse, 1997).

3.1.3 Forming Fuzzy Partitions and Grid-Like Rules

A common way to induce fuzzy rule bases from scratch, i.e. to learn them completely from examples without prior definitions of rules or fuzzy sets, is to learn in two stages. The induction starts with equidistantly defined fuzzy partitions with standard linguistic labels. In the first stage, these fuzzy sets are used with the Wang&Mendel algorithm to create rules. In the second stage, the fuzzy sets are fine-tuned for this (now fixed) set of rules.

As rules and fuzzy set definition strongly depend on each other, learning in two phases is sometimes not flexible enough. Higgins and Goodman (1992, 1994) suggested an extension of the Wang&Mendel algorithm that does not need a priori defined fuzzy sets. The algorithm starts with only one fuzzy set per input dimension, and thus creates one rule only. Then a split point is chosen where the current model makes the maximal error. At this point, a new fuzzy set is introduced in every dimension. With the resulting fuzzy partitions, Wang&Mendel is executed again. These steps—finding a split point, refining the granulation of the input space, and creating new rules based on this granulation—are iterated until sufficient precision is reached or some other criterion is fulfilled.

3.1.4 Hyperbox-oriented Fuzzy Rule Learning

In Chapter 2 we have discussed properties of global granulations of the input space. The restriction to a number of fuzzy sets that have meaningful names and that are shared by all rules can make the rule base more readable. However, one loses some flexibility,[1] which might in turn make a finer granulation necessary, and thus lead to huge rule bases that are again hard to read. In such cases rule bases with local definitions of the fuzzy sets might be the better solution. Their rule-wise definition of the fuzzy sets allows to increase complexity of the rules in those regions of the input space only where extended flexibility is needed.

An algorithm that extracts such rules from data is the *Fuzzy Min-Max Neural Network (FMM)* proposed by Simpson (1992, 1993). The approach is *hyperbox-oriented*, i.e. each rule is associated with a hyperbox in input space, within which the rule fully applies. The hyperboxes are represented by pairs of minimal and maximal value in each dimension.[2] The activation of

[1] For example, forced axis parallelism of decision boundaries, cf. Section 2.3.3.

[2] The name "Min-Max" classifier refers to these corner points of the hyperboxes. It does not specify the inference mechanism, which is "average-max" in the original proposal (Simpson, 1992).

the rule outside the hyperbox is controlled by a fuzzy membership function and monotonously decreases with distance from the hyperbox. Simpson suggests to use symmetrical trapezoidal membership functions with fixed widths of the "rims". All features are scaled into the unit interval $[0, 1]$. Let $v_{r,j}$ and $w_{r,j}$ denote the minimum and maximum value of the jth dimension of the rth hyperbox (rule), and let γ be the parameter that controls the *sensitivity*, i.e. the behavior outside the interval $[v_{r,j}, w_{r,j}]$. Then the membership degree for dimension j for rule r is defined as[3]

$$\begin{aligned}\mu_{r,j}(x_j) &= \frac{1}{2}(\max\{0, 1 - \min\{0, \gamma(x_j - w_{r,j})\}\} + \\ &\quad \max\{0, 1 - \min\{0, \gamma(v_{r,j} - x_j)\}\}). \end{aligned} \tag{3.2}$$

The activation of a rule r is defined as the average membership degree

$$act_r(\boldsymbol{x}) = \frac{1}{n_d} \sum_{j=1}^{n_d} \mu_{r,j}(x_j). \tag{3.3}$$

The training algorithm, however, does not take into account the membership function, but considers the hyperboxes (i.e. the min-max pairs) only. The original algorithm iterates over the tuples once. For each tuple the following steps are performed:

- Test if the tuple is already contained in a compatible hyperbox, where a tuple is said to be *compatible* with a hyperbox if the labels of the corresponding rule and tuple match. If such a hyperbox exists, proceed to the next tuple.
- If no such hyperbox exists, find that hyperbox that is closest to the tuple, compatible, and *can be extended* to contain the tuple. The latter criterion involves a user-specified parameter θ that controls the maximal allowed size of a hyperbox. The constraint is defined as

$$\sum_{j=1}^{n_d} (\max\{w_{r,j}, x_j\} - \min\{v_{r,j}, x_j\}) \leq n_d \theta. \tag{3.4}$$

  If such a rule r exists, its minimum and maximum points for all dimensions j are updated to

$$v'_{r,j} := \min\{v_{r,j}, x_j\} \text{ and } w'_{r,j} := \max\{w_{r,j}, x_j\}. \tag{3.5}$$

[3] Eq. (3.2) defines trapezoidals that are scaled in ordinate direction, such that the membership degrees come from the range $[0.5, 1]$. The width of the rim is $\frac{1}{\gamma}$.

If no compatible and extendable rule exists, a new hyperbox is created and initialized to contain only the tuple, i.e. for all j, $v_{r,j}$ and $w_{r,j}$ are both set to x_j. The consequent of the corresponding rule is set to the tuple's label.

- If a new rule r has been added or if an existing rule r has been extended to contain the current tuple, it is checked for overlap with the other rules. As the activation inside a hyperbox is 1, hyperboxes with concurring consequents must not overlap to avoid ties. Thus for all rules r' with consequents different from rule r, it is checked if r and r' overlap. If they do, the hyperboxes are contracted based on the "minimal adjustment principle", i.e. that dimension and min-max point is chosen that allows to resolve the overlap with a minimal change.

The induced rule bases often contain many rules, which are hardly interpretable due to their number and the local fuzzy set definitions. The resulting classifiers strongly depend on the ordering of the tuples and often perform not very well on unseen data. However, the algorithm is one of the classical induction approaches, and had considerable impact on fuzzy classifier research. A wide number of modifications and extensions have been proposed, ranging from simple changes (e.g. process the tuples several times; use different membership functions and t-norms) to more sophisticated designs with adaptive resolutions (e.g., Rizzi et al., 1998). Berthold (2003) devised an advanced approach that is based on similar mechanisms. His membership functions are, however, parameterized by 4-tuples $(a_j^r, b_j^r, c_j^r, d_j^r)$ that define trapezoids, possibly degenerated to triangles (i.e. $b_j^r = c_j^r$) or "don't cares" (i.e. $a_j^r = -\infty, d_j^r = +\infty$). This allows more general rules, which can be crucial to get understandable rule bases for higher dimensional datasets.

3.1.5 Hybrid Methods of Rule Extraction

There are a number of approaches that combine several techniques to induce interpretable fuzzy rules from data. Popular combinations include, for example, fuzzy systems and neural networks (see below), or fuzzy systems and evolutionary algorithms (as treated in more detail in Chapter 4). Other hybrid approaches combine fuzzy techniques with classical machine learning models (e.g. induction of decision trees, see below).

Neuro-Fuzzy Approaches

Systems that combine neural networks and fuzzy systems have gained quite a lot of attention over the past years. The motivating idea is to exploit the learning ability of neural networks and the proximity of fuzzy systems to human thinking. There are a number of possibilities how to combine the two. The approaches can be divided into several groups (Nauck et al., 1997):

- **Fuzzy neural networks:** These neural networks use fuzzy methods to support learning—e.g. by using fuzzy rules to change the learning parameters (Halgamuge et al., 1994)—or work with fuzzy inputs (Narazaki and Ralescu, 1991; Ishibuchi et al., 1995).

- **Concurrent "neural/fuzzy systems":** neural network and fuzzy system process the same task, but are constructed independently. Usually the neural network preprocesses the inputs to, or postprocesses the outputs from the fuzzy system.

- **Cooperative neuro-fuzzy models:** A neural network is used to determine the parameters (rule weights and/or fuzzy sets) or the structure (rule base) of a fuzzy system. After the learning phase, the fuzzy system works without the neural network.

- **Hybrid neuro-fuzzy models:** Neural network and fuzzy system are combined into a homogeneous architecture, such that the system can at any time be interpreted as a (specialized) neural network or as a fuzzy system.

Most current neuro-fuzzy approaches are hybrid neuro-fuzzy models. This architecture is especially suited for data analysis. The other hybridizations either still have a black-box neural network as one part, or they lose their learning ability once transformed into a fuzzy system. Hybrid neuro-fuzzy models are transparent rule systems when interpreted as fuzzy systems, and can still be trained without modifications. For a survey on neuro-fuzzy models for data analysis see, e.g., (Klose et al., 2001).

An example of a neuro-fuzzy classification system is NEFCLASS (Nauck and Kruse, 1995, 1997). NEFCLASS uses the Wang&Mendel approach to induce fuzzy rules (and thus the structure of the neural network), and a backpropagation-like parameter learning algorithm. Additionally, the current implementations of the NEFCLASS approach have been extended by

a number of features that make it more appropriate for real-world applications. The extensions include the treatment of missing values and the ability to use data with both, numeric and symbolic attributes (Nauck and Kruse, 1999; Nauck et al., 1999). A lot of attention has been paid to the development of automatic pruning strategies to get more compact and readable rule bases. When a rule base is induced from data with the Wang&Mendel algorithm, it often has too many rules and thus gives little insight into the structure of the data. The pruning techniques implemented in NEFCLASS have been shown to be effective in both, reducing the number of rules and increasing generalization ability (Nauck et al., 1997; Klose et al., 1998; Klose and Nürnberger, 1999).

Tree-based Approaches

Decision trees are a popular classifier approach, where the classification knowledge is stored in a tree structure. The nodes are associated with tests on the features and the leaves are associated with classes. A new case is classified by descending from the root on the path that is given by the outcomes of the tests in the nodes, until a leave is reached and the associated class is assigned. What makes decision trees interesting for data analysis is the existence of very efficient divide-and-conquer induction algorithms (Breiman et al., 1984; Quinlan, 1993). The classification knowledge stored in decision trees is rather transparent and interpretable, and additionally the trees can be transformed into (crisp) rules.

This mechanism can be exploited in several ways to generate fuzzy rules. Tests on numeric attributes lead to crisp thresholds, which are often unsatisfying. A natural extension of decision trees is thus to replace the tests in the nodes with fuzzy queries. In such fuzzy decision trees, a case can descend on several paths from the node to the leaves, with different degrees of membership or compatibility according to the tests' outcomes. The induction algorithms for fuzzy decision trees are very similar to those of crisp decision trees like ID3 or C4.5, just that the test selection measures are replaced by measures that can deal with fuzzy memberships (Janikow, 1998; Chi and Yan, 1996; Yuan and Shaw, 1995; Wang et al., 1999). From the resulting fuzzy decision trees fuzzy rules can be extracted.

The second group uses the standard decision tree algorithms to induce crisp rules. These initial rules are then transformed into fuzzy rules, and eventually postprocessing steps, like pruning or fine-tuning, are applied afterwards (Jäkel et al., 1999; Maher and Clair, 1993). These approaches are often combined with neural networks or evolutionary optimization.

3.2 Unsupervised Induction of Fuzzy Rules

The second important learning paradigm is unsupervised learning, i.e. learning from example data without known class labels. Generally speaking, the task of unsupervised learning is to reveal inherent structure in the data. Typical tasks of unsupervised learning are

- **Compressing or approximating the distribution**, e.g. by kernel density estimation (Silverman, 1986), or by learning vector quantization (LVQ, Kohonen, 1986).
- **Dimensionality reduction**: find lower dimensional substructures in the data, e.g. with self organizing maps (SOMs, Kohonen, 1982, 1995).
- **Dependency analysis**: find (in)dependencies between features that may hint to causal relations and can, for example, be extracted by graphical models (Borgelt and Kruse, 2002),
- **Cluster analysis**: try to find or reconstruct the class information from the inherent structure in the data (Jain and Dubes, 1988).

As cluster analysis ultimately assigns classes to objects, it is also referred to as *automatic* or *unsupervised classification*. In the classifier context of this thesis, cluster analysis is the most relevant unsupervised learning task. It tries to find groups in the data such that objects in the same group are similar to each other. As there is often some ambiguity when assigning objects to clusters, there have been a number of fuzzy approaches to cluster analysis which take into account partially overlapping clusters. In Sections 3.2.1 and 3.2.2, we outline some of the most important fuzzy cluster algorithms. Section 3.2.3 deals with the question of cluster validity. The results of fuzzy clustering can be transformed into fuzzy rules, and can therefore be used for unsupervised extraction of fuzzy rules from data, as described in Section 3.2.4.

3.2.1 Models with Point-Prototype Clusters

The fuzzy clustering algorithms considered in the following sections use an unlabeled dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, where the features are real-valued, i.e. $\boldsymbol{x}_j \in \boldsymbol{X} = \mathbb{R}^{n_d}$. The result of the algorithms is a $n_k \times n$ matrix of cluster memberships $U = [u_{ij}]$, where n_k is the number of clusters and $u_{ij} \in [0, 1]$ specifies the degree to which object j belongs to cluster i. Usually, clusters are expected to be non-empty: $\forall i \in \{1, \ldots, n_k\} : \sum_{j=1}^{n} u_{ij} > 0$. In fuzzy

cluster analysis, we also distinguish between possibilistic or probabilistic labels, where the membership degrees of an object are called probabilistic if they sum up to one over all clusters, i.e. if $\forall j \in \{1, \ldots, n\} : \sum_{i=1}^{n_k} u_{ij} = 1$.

Additional to U, most algorithms produce a prototypical description of the clusters. In case of *point-prototype clusters*, these are solely defined by their positions in space. Thus, each cluster is associated with a cluster center, represented by a vector $v_i \in \mathbb{R}^{n_d}$. As clustering means grouping *similar* objects, similarity must be measured. In point-prototype models this is usually done with a distance measure

$$\delta_A(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_A = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T A (\boldsymbol{x} - \boldsymbol{x}')}, \tag{3.6}$$

with $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{n_d}$ and a positive-definite $n_d \times n_d$ matrix A (Mahalanobis distance). If A is the identity matrix $\mathbb{1}$, $\delta_A(\boldsymbol{x}, \boldsymbol{x}')$ becomes the Euclidean distance.

Probably the best known and most widely used fuzzy clustering algorithm is fuzzy c-means (FCM, Bezdek, 1981). It tries to minimize the objective function

$$J_m(U, \boldsymbol{V}) = \sum_{i=1}^{n_k} \sum_{j=1}^{n} u_{ij}^m \|\boldsymbol{x}_j - v_i\|_A^2, \tag{3.7}$$

where U are probabilistic cluster memberships,[4] $\boldsymbol{V} = \{v_1, \ldots, v_k\}$ is the set of cluster prototypes, and $m > 1$ is a *degree of fuzzification*. The closer m is to one, the crisper the cluster result will be. A very common choice is $m = 2$.

The minimization of J_m is usually done by *alternating optimization*. From a fixed set of prototypes V, the optimal cluster membership U can be derived, as well as the optimal positions of the prototypes V for given memberships U. Thus the optimization comprises the following steps:

1. Choose an appropriate number of clusters n_k and a fuzzifier m.
2. Randomly choose initial cluster centers V.
3. Estimate new memberships U from V as

$$u_{ij} = \left(\sum_{i'=1}^{n_k} \left(\frac{\|v_i - \boldsymbol{x}_j\|_A}{\|v_{i'} - \boldsymbol{x}_j\|_A} \right)^{\frac{2}{m-1}} \right)^{-1}. \tag{3.8}$$

[4] Fuzzy c-means is an extension of the hard c-means (or k-means) algorithm (MacQueen, 1967). They basically minimize the same objective function. However, in k-means the labels are crisp (i.e. $u_{ij} \in \{0, 1\}$), and thus constant m can be ignored.

4. Estimate new cluster centers V from updated membership degrees U as

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^m \boldsymbol{x}_j}{\sum_{j=1}^{n} u_{ij}^m}. \tag{3.9}$$

5. Loop steps 3. and 4. until the change of the cluster centers is below a threshold or the number of iterations exceeds some limit.

The algorithm can be guaranteed to converge, although not necessarily to a global minimum of J_m.

Possibilistic c-means (PCM), a variant of FCM, assigns possibilistic labels to the objects (Krisnapuram and Keller, 1993). The idea is that possibilistic labels allow outliers to have low membership degrees to all clusters. The objective function J is extended to

$$J_m^{\text{PCM}}(U, \boldsymbol{V}) = J_m(U, \boldsymbol{V}) + \sum_{i=1}^{n_k} w_i \sum_{j=1}^{n} (1 - u_{ij})^m, \tag{3.10}$$

where the added term rewards high memberships and thus prevents $u_{ij} = 0$ as a trivial minimum of J. However, as has been shown in (Timm and Kruse, 2002; Timm, 2002), the global minimum of PCM puts all cluster centers to the same point, and the algorithm thus depends on an optimization that stops in local minima.

3.2.2 Models with Non Point-Prototype Clusters

Clusters that are only represented by their centers tend to be hyperspherical and of similar size. If however, the clusters in the data are hyperellipsoidal or of varying size, the results of FCM or PCM can be unsatisfying. An obvious extension is to represent a cluster not only by its position, but also by its shape. Gustafson and Kessel (1979) proposed a clustering algorithm (GK) that for every cluster optimizes a positive definite $n_d \times n_d$ matrix A_i which is used for the calculation of distances. With $\boldsymbol{A} = \{A_1, \ldots, A_{n_k}\}$ being the set of matrices, the objective function Eq. (3.7) is modified to

$$J_m^{GK}(U, \boldsymbol{V}, \boldsymbol{A}) = \sum_{i=1}^{n_k} \sum_{j=1}^{n} u_{ij}^m \|\boldsymbol{x}_j - v_i\|_{A_i}^2. \tag{3.11}$$

$J_m^{GK}(U, \boldsymbol{V}, \boldsymbol{A})$ is minimized with the additional constraint $\det(A_i) = \rho_i$, where $\rho_i > 0$ are user specified constants that regulate cluster sizes and ensure that the A_i are positive definite.

From fixed cluster memberships U and prototype centers $\boldsymbol{V}$ the optimal A_i can be calculated through the *fuzzy covariance matrices* C_i defined as

$$C_i = \frac{\sum_{j=1}^{n} u_{ij}^m (\boldsymbol{x}_j - v_i)(\boldsymbol{x}_j - v_i)^T}{\sum_{j=1}^{n} u_{ij}^m}. \tag{3.12}$$

The optimal A_i for given U and $\boldsymbol{V}$ is

$$A_i = (\rho_i \det(C_i))^{\frac{1}{n_d}} C_i^{-1}. \tag{3.13}$$

$J_m^{GK}(U, \boldsymbol{V}, \boldsymbol{A})$ is minimized with alternating optimization similar to FCM. U is updated by Eq. (3.8) with the distances replaced by $\| \cdot \|_{A_i}$, $\boldsymbol{V}$ is updated by Eq. (3.9) and $\boldsymbol{A}$ is calculated by Eq. (3.13). As the objective function J_m^{GK} is more complex due to its high number of free parameters, it is common to run FCM first and use its results to initialize U and $\boldsymbol{V}$.

Gath and Geva (GG) suggested a different approach that replaces the Mahalanobis distance $d^2(v_i, x_j) = \|x_j - v_i\|_A^2$ with an exponential distance measure (Gath and Geva, 1989)

$$d_{C_i,GG}^2(v_i, \boldsymbol{x}_j) = \left(\frac{\sqrt{\det(C_i)}}{\rho_i} \right) \exp(\frac{1}{2} \|\boldsymbol{x}_j - v_i\|_{C_i^{-1}}^2), \tag{3.14}$$

where C_i are the fuzzy covariance matrices calculated from Eq. (3.12) with $m = 1$. In spite of this, this distance is used in Eq. (3.8) with $m = 2$:

$$u_{ij} = \left(\sum_{i'=1}^{n_k} \left(\frac{d_{C_i,GG}(v_i, \boldsymbol{x}_j)}{d_{C_{i'},GG}(v_{i'}, \boldsymbol{x}_j)} \right)^2 \right)^{-1}. \tag{3.15}$$

There is a close relation of this distance measure to normal distributions, and it is possible to interpret GG as maximum likelihood fitting of a Gaussian mixture model to the data and thus allows probabilistic interpretation of the cluster memberships (Bezdek et al., 1999).

3.2.3 Cluster Quality Measures

The unsupervised nature of clustering, i.e. its lack of an a priori, externally defined objective makes it difficult to assess its results algorithmically. Clustering algorithms partition the data no matter whether there are marked clusters or not. Therefore the choice of the cluster shape and especially of an appropriate number of clusters is quite important.

Cluster quality measures try to quantify how good the discovered clusters represent the data. The sum of squared distances between points $\boldsymbol{x}$ and corresponding (closest) rule centers $v_{i(\boldsymbol{x})}$ is one measure of cluster quality:

$$\mathcal{Q}^{ssd} = \sum_{\boldsymbol{x} \in \boldsymbol{X}} \|\boldsymbol{x} - v_{i(\boldsymbol{x})}\|^2. \tag{3.16}$$

A reasonable way to judge cluster quality is to optimize the clusters with respect to a given objective function J and number of clusters n_k, and then assess the result with a separate cluster quality measure. This second measure should be independent of J in the sense that it is based on a different rationale. $\mathcal{Q}^{ssd}$ is hardly suited as a cluster quality measure for the algorithms described in this chapter because it has obvious similarities to objective function J in Eq. (3.7). Actually, $\mathcal{Q}^{ssd}$ is minimized by (hard) c-means clustering.

In the following sections we present two measures of cluster quality, the Davies-Bouldin index and Dunn's index. As we show in Chapter 5, they also play a role in semi-supervised learning.

Davies-Bouldin Index

The measure proposed by Davies and Bouldin (1979) measures the ratio of intra-cluster to inter-cluster distances. The measure thus gives good values for compact clusters that are well separated. Let $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ be a partition of the unlabeled dataset $\mathcal{D}$ into n_k crisp clusters generated by winner-takes-all defuzzification of U. Inter-cluster distance is the Euclidean distance between two cluster centers v_i and $v_{i'}$

$$\delta_{ii'}^{DB} = \|v_i - v_{i'}\|, \tag{3.17}$$

and intra-cluster distance is defined as the average distance of the tuples $\mathcal{D}_i$ assigned to a cluster center i

$$\alpha_i^{DB} = \sqrt{\frac{1}{|\mathcal{D}_i|} \sum_{\boldsymbol{x} \in \mathcal{D}_i} \|\boldsymbol{x} - v_i\|^2}. \tag{3.18}$$

The intra-inter-cluster ratio for two clusters is defined as

$$\rho_{ii'}^{DB} = \frac{\alpha_i^{DB} + \alpha_{i'}^{DB}}{\delta_{ii'}^{DB}}. \tag{3.19}$$

The Davies-Bouldin index is calculated from the ratios of the n_k clusters as

$$\mathcal{Q}^{DB} = \frac{1}{n_k} \sum_{i=1}^{n_k} \max_{i' \neq i} \rho_{ii'}^{DB}. \tag{3.20}$$

Low values of $\rho_{ii'}^{DB}$ indicate that the clusters are small in comparison to their distance. $\mathcal{Q}^{DB}$ sums up these values for each cluster and its nearest or most overlapping neighbor. The better the clusters are separated, the smaller the value of $\mathcal{Q}^{DB}$ will be.

Dunn's Index

The idea of the index proposed by Dunn (1974) is similar to the Davies-Bouldin index. However, intra- and inter-cluster distances are defined differently. Intra-cluster distance is measured by the *diameter* of the cluster, defined as[5]

$$\alpha_i^D = \max_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{D}_i} \|\boldsymbol{x} - \boldsymbol{x}'\|. \tag{3.21}$$

Inter-cluster distance is defined as the *set distance*

$$\delta_{ii'}^D = \min_{\boldsymbol{x} \in \mathcal{D}_i, \boldsymbol{x}' \in \mathcal{D}_{i'}} \|\boldsymbol{x} - \boldsymbol{x}'\|. \tag{3.22}$$

Dunn's index uses the ratio of smallest inter- to greatest intra-cluster distance

$$\mathcal{Q}^D = \frac{\min_{i \neq i'} \delta_{ii'}^D}{\max_i \alpha_i^D}. \tag{3.23}$$

Good clusters are indicated by large values of $\mathcal{Q}^D$. The main problem of Dunn's index is its sensitivity to noise. Due to the definition of diameter and set distance, a single outlier can massively change the results. This problem has been addressed by *Dunn-like* indices that replace the definitions of diameter and set distance by more robust measures based on graph theoretic concepts, like minimum spanning tree or relative neighborhood graph (Pal and Biswas, 1997; Bezdek and Pal, 1998).

3.2.4 Converting Fuzzy Clusters to Fuzzy Rules

Fuzzy clustering can be used to reveal or reconstruct class information from unlabeled data. The result is a soft assignment of the objects to the clusters. In the scope of this thesis, however, we are interested in descriptions of the

[5] Note that the Euclidean distance in Eq. (3.21) can be replaced by any other metric.

structure in the form of interpretable fuzzy classification rules. Especially in fuzzy control it has become popular to combine fuzzy clustering with subsequent transformation of the clusters into fuzzy if-then rules. Every cluster is turned into one fuzzy rule. The fuzzy sets are derived by projecting the clusters onto the axes (Sugeno and Yasukawa, 1993; Klawonn and Kruse, 1997). For classification rules, only the antecedent fuzzy sets have to be derived. The consequent, i.e. the predicted class cannot be derived from unlabeled data. Thus, either an arbitrary label is used (e.g. the cluster number), or the labels are chosen from additional background knowledge.

Strictly speaking, projecting a cluster j to axis i to get the corresponding antecedent fuzzy set $\mu_{i,j}$ means to compute

$$\mu_{i,j}(y) = \sup\{u_j(\boldsymbol{x}) | \boldsymbol{x} \in \boldsymbol{X} \wedge x_i = y\}, \tag{3.24}$$

where membership degree $u_j(\boldsymbol{x})$ is calculated according to the update equations, i.e. let $d_j(\cdot,\cdot)$ denote the distance function used in the corresponding objective function and cluster j, then $u_j(\boldsymbol{x})$ is defined as (cf. Eq. (3.8) and Eq. (3.15))

$$u_j(\boldsymbol{x}) = \left(\sum_{j'=1}^{n_k} \left(\frac{d_j(v_j, \boldsymbol{x})}{d_{j'}(v_{j'}, \boldsymbol{x})} \right)^{\frac{2}{m-1}} \right)^{-1}. \tag{3.25}$$

However, computing Eq. (3.24) is rather cumbersome, and hence we usually project only the given data and the corresponding cluster memberships, i.e.

$$\mu_{i,j}(y) = \sup\{u_{cj} | c = 1, \ldots, n_k \wedge (\boldsymbol{x}_j)_i = y\}, \tag{3.26}$$

with $\sup(\emptyset) := 0$. The singular peaks that are defined by this equation are either connected by computing the convex hull, or are approximated by simpler parameterized functions, like triangles, trapezoids, or Gaussians. Details of the approximation can, for example, be found in (Sugeno and Yasukawa, 1993).

This projection works with any fuzzy clustering algorithm. However, in most cases projecting leads to a certain loss of information, i.e. the Cartesian product of the projected fuzzy sets is not equal to the original cluster memberships. The smallest projection error can usually be achieved using FCM, due to its spherical cluster shape. If, however, more flexibility is needed, and thus GK or GG are used, the projection error can become rather large. This error stems mostly from the rotation of the hyperellipsoidal clusters. A reasonable compromise between flexibility of cluster shapes and projection accuracy can be achieved by modifying the GK and GG models to extract

axis parallel clusters, i.e. by restricting the fuzzy covariance matrices C_i to diagonal matrices. As shown by Klawonn and Kruse (1995, 1997), this also significantly simplifies the update scheme, as the matrix inversions are no longer necessary.

In case of the GK algorithm, A_i in Eq. (3.13) becomes a diagonal matrix with

$$a_{dd}^{(i)} = \frac{(\rho_i \prod_{d'=1}^{n_d} \sum_{j=1}^{n} u_{ij}^m (x_{j,d'} - v_{i,d'})^2)^{\frac{1}{n_d}}}{\sum_{j=1}^{n} u_{ij}^m (x_{j,d} - v_{i,d})^2}, \tag{3.27}$$

and $a_{dd'}^{(i)} = 0$ for $d \neq d'$. With A_i defined in this way, cluster memberships and prototype centers are computed as in the original version of GK (Klawonn and Kruse, 1997).

In the GG model, which is—as mentioned above—closely related to mixtures of Gaussians, it can be shown that the diagonal entries of the matrices C_i are computed as (*fuzzy*) variances

$$c_{dd}^{(i)} = \frac{\sum_{j=1}^{n} u_{ij}^m (x_{j,d} - v_{i,d})^2}{\sum_{j=1}^{n} u_{ij}^m}, \tag{3.28}$$

and $c_{dd'}^{(i)} = 0$ for $d \neq d'$. Of course, the inverse matrices C_i^{-1} needed in Eq. (3.14) are also diagonal, and the diagonal elements are simply the reciprocal values of $c_{dd}^{(i)}$. The update equations for U and $\boldsymbol{V}$ are again kept as in the original algorithm (Klawonn and Kruse, 1997).

If the axis parallel version of the GG algorithm is used to cluster the data and Gaussian fuzzy sets are used in the rules, the fuzzy set parameters can be directly derived from cluster centers v_i and fuzzy variances C_i.

Apart from projection errors, another problem of fuzzy rule bases obtained from cluster analysis is their decreased comprehensibility. Since the resulting fuzzy sets are not restricted to match any semantic interpretation, they are often hardly linguistically interpretable.

3.3 Conclusions

In this chapter, we gave a survey on the most common techniques to induce fuzzy rules from a dataset of examples. The most obvious difference of the presented approaches is that between supervised and unsupervised approaches. The algorithms of these two groups are applied to completely different problems. Although we can in principle turn a supervised into an unsupervised learning task by discarding the labels, this will in general not

yield satisfying results. The other direction is not possible at all, as the label information of the training data is not available. The main problem of unsupervised learning is that its results strongly depend on the distance measure used and the scaling of the data. It has commonly problems in finding the right number of clusters. Thus, we can usually not guarantee that the clusters in the data space correspond to meaningful classes of the objects. Especially if the dimensionality is high, similarity functions—playing a central role in clustering—show strange behavior (see, e.q., Jiminez and Landgrebe, 1998). This makes pure unsupervised learning only moderately applicable to high-dimensional datasets. Supervised learning is generally less sensitive to the number of dimensions.

In Chapter 5, we present existing approaches to semi-supervised learning. One goal of the previous sections was to see, which fuzzy rule induction algorithms are most promising for the extension to semi-supervised learning. Semi-supervised approaches depend on an ability to reveal structure from the distribution of the data points without consideration of the class labels. Thus it is not surprising that unsupervised approaches have been proposed as a basis for semi-supervised learning, and there are some extensions of unsupervised (fuzzy) clustering to semi-supervised (fuzzy) clustering. However, there are also semi-supervised extensions of supervised algorithms, like the *generalized fuzzy min-max classifier* (see Section 5.2.6). The suitability of the rule learning approaches for extension to semi-supervision is considered in Chapter 6, where we propose our semi-supervised algorithm for fuzzy classification rule learning.

The next chapter deals with evolutionary algorithms, which are also popular for fuzzy rule learning. We describe them in a separate chapter, because they play a special role: depending simply on the fitness function used, these algorithms can learn either supervised or unsupervised. As we discuss in Section 6.1, this makes them especially interesting for semi-supervised learning.

Chapter 4

Evolutionary Algorithms for Fuzzy Rule Learning

Evolutionary algorithms (EA) have been established as a versatile class of search and optimization techniques (Goldberg, 1989; Holland, 1975; Mitchell, 1998). Like artificial neural networks, they mimic mechanisms from nature. However, the simulation of biological reproduction schemes in EA allows—in contrast to, e.g., backpropagation learning in neural network—to optimize not only parameters, but also model structure. This makes them well suited for complex search spaces.

As they perform stochastic search using only a performance function, evolutionary algorithms are largely problem independent. They can easily be integrated with fuzzy techniques. However, it is necessary to understand how optimization works in evolutionary algorithms to choose appropriate encodings of potential solutions as *chromosomes*, to define a function to assess the fitness of these chromosomes, and to choose reasonable operators for selection, recombination, and mutation. Thus, we will briefly review the relevant concepts of evolutionary algorithms in Section 4.1. In Section 4.2, we review previous work on evolutionary algorithms for fuzzy rule learning. Section 4.3 summarizes and discusses the concepts and sets them into relation to the fuzzy rule induction algorithms presented in the previous chapter.

Our measures for semi-supervised learning of fuzzy rules call for a rather capable non-linear optimization technique. As we show in Chapter 6, evolutionary algorithms are well suited for this task.

4.1 Introduction

All evolutionary algorithms share the same underlying ideas. They simulate the development of a population of individuals over a number of generations. Each individual can be interpreted as candidate solution for the problem to be solved. Starting with a randomly initialized population, individuals which are fitter with respect to the problem have a higher probability to survive or reproduce. By recombinations of good partial solutions and by occasional mutations, evolutionary algorithms perform a stochastic, directed and parallel search. However, the details of how candidate solutions are encoded and how reproduction is modeled can be very different. There are two distinct paradigms in evolutionary algorithms:

- **Evolution strategies (ES)** have been proposed by Rechenberg (1965, 1994) and Schwefel (1965). They focus on the *phenotype* in the sense that the individuals directly encode the values of the parameters to be optimized, and that mutation and recombination work directly on these.
- **Genetic algorithms (GA)** have been introduced by Holland (1975) and Goldberg (1989). In contrast to ES, they focus on the *genotype* of the individuals, i.e. the chromosomes that encode the candidate solutions are strings over a discrete (often binary) alphabet. GA operations are closer to biological genetics, i.e. mutation and recombination directly affect the genes, ignoring the effect on the phenotype at first.

Both types will be described in the next two sections. Section 4.1.4 will discuss differences and connections.

4.1.1 Evolution Strategies

The individuals in the original variants of evolution strategies hold a vector of genes which are (most often real-valued) parameters that are to be optimized:

$$\bar{g} = (g_1, \ldots, g_n) \in \boldsymbol{G}, \text{ where usually } \boldsymbol{G} = \mathbb{R}^n. \tag{4.1}$$

The optimization task is formalized in a fitness function

$$\mathcal{Q} : \boldsymbol{G} \to \mathbb{R} \tag{4.2}$$

that assesses the quality of a candidate solution on a numeric scale. In ES, the best solutions according to that fitness function are taken to the next generation as seed points for better solutions.

The most important operation in ES is mutation, i.e. random variation of the parameter values. In nature, small deviations are more probable than large changes. Thus, mutation in ES is performed by adding Gaussian distributed noise with zero mean and standard deviation σ:

$$\mathcal{N}_{0,\sigma} = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2\sigma^2}}. \tag{4.3}$$

Usually, the genes of all children are modified in this way before they are included in next generation's population.

Parameters like σ that control the search characteristics are called *strategy parameters*. Small values of *sigma* lead to a *local* exploration that helps to find the optimum near the current values. Large values might help to escape local minima. It has thus become usual to control the strategy parameters themselves in the evolution strategy. Each gene g_i is associated with a mutation parameter σ_i. In a mutation step, a gene is modified according to its own step size:

$$g_i^{(t+1)} = g_i^{(t)} + \mathcal{N}_{0,\sigma_i}. \tag{4.4}$$

Additionally, the strategy parameters are mutated. This can, for example, be done by adding Gaussian noise with zero mean and a certain deviation δ (e.g., Bäck et al., 1991):

$$\sigma_i^{(t+1)} = \sigma_i^{(t)} + \mathcal{N}_{0,\delta}. \tag{4.5}$$

The idea is that strategy parameters which are well suited for the search space lead more often to good solutions than others, and thus have a higher chance to survive. This helps to balance between *exploration* of the search space (*global* search) and *exploitation* of good, already found solutions (*local* search).

Recombination plays a central role in evolutionary algorithms, as it allows to transfer information about good subsolutions to (and combine these in) the next generation. In ES, there are two main variants:

- Discrete recombination: the genes of an offspring are copied from either of the parents with equal probability.
- Intermediate recombination: the genes of an offspring are set to the mean values of the corresponding parents' genes.

The strategy parameters are modified in the same way.

It has been shown that even the simplest (1+1)-ES—i.e. one parent and one offspring compete—, even without mutation adaptation, is able to find the global optimum in search space with probability one (see, e.g., Bäck et al., 1991). Necessary prerequisite is that the mutation operation can reach any point in search space with non-zero probability—and that the algorithm runs long enough. However, since the proof says nothing about the number of generations that are needed, it is not very helpful.

4.1.2 Genetic Algorithms

As in evolution strategies, the individuals in genetic algorithms are represented by a gene string $\bar{g} = (g_1, \ldots, g_n)$. However, similar to the four bases Adenin, Thymin, Cytosin, and Guanin that nature uses to encode the genetic information in the DNA, the g_i stem from a discrete, usually even binary alphabet $g_i \in \{0, 1\}$. To get a candidate solution, the genotype must first be translated into its corresponding phenotype. This usually means that the parameters, which shall be be optimized, correspond to fixed substrings of the chromosomes. Like in ES, the problem-specific fitness function $\mathcal{Q}$ is defined on the phenotype.

Apart from the significantly different encoding of individuals in ES and GA, they use other definitions of the genetic operators. Mutation, for example, is specialized to real valued parameters in ES. In GA, mutation affects the genes of the genotype without consideration of their meaning for the phenotype. Thus, every gene can be affected by mutation with equal probability p_{mut}. Let $\chi \in [0, 1]$ denote a uniformly distributed random variable. Then mutation of a gene g_i is defined as

$$g_i^{(t+1)} = \begin{cases} \neg g_i^{(t)}, & \text{if } \chi < p_{mut} \\ g_i^{(t)}, & \text{else.} \end{cases} \tag{4.6}$$

While the purpose of mutation is to bring in new solutions to the gene pool, recombination shall ideally pass the parents' information about good partial solutions to an offspring. The most common recombination operator is crossover. Let the bit strings of two parents be denoted by $\bar{g}_1 = (g_{1,1}, \ldots, g_{1,n})$ and $\bar{g}_2 = (g_{2,1}, \ldots, g_{2,n})$, respectively. Then the so-called *one-point* crossover, the most basic variant, randomly chooses a split point $i \in \{1, n-1\}$ and produces an offspring with a bit string

$$\bar{g}_c = (g_{1,1}, \ldots, g_{1,i}, g_{2,i+1}, \ldots, g_{2,n}). \tag{4.7}$$

In one-point crossover the probability of two genes to be copied into a child depends on the position on the chromosome: the closer genes are to the

ends of the bit string, the higher the probability that they are divided. E.g. one-point crossover always separates a parent's first and last gene $g_{j,1}$ and $g_{j,n}$. To avoid this unwanted bias, two-point crossover can be used, where two split points $1 \leq i_1 < i_2 \leq n$ are used to define an offspring

$$\bar{g}_c = (g_{1,1}, \ldots, g_{1,i_1}, g_{2,i_1+1}, \ldots, g_{2,i_2-1}, g_{1,i_2}, \ldots, g_{1,n}). \tag{4.8}$$

Some proposals extend this to multi-point crossover, and some suggest to involve more than two parents.

In contrast to ES, where simply the best individuals reproduce, genetic algorithms use more sophisticated selection schemes that also allow less fit individuals to survive, although with a lower probability. Apart from being biologically more plausible, the advantage of these selection schemes is increased exploration of the search space and thus higher probabilities to escape from local optima. The most widely used schemes include *fitness proportionate*, *rank-based*, and *tournament* selection.

As the name suggest, in fitness proportionate selection the probability for an individual to be selected for reproduction is proportional to the quality of the solution it represents. Let $\mathcal{Q}(\bar{g}_j)$ denote the fitness of the j-th individual. Then its selection probability $P(\bar{g}_j)$ is defined as

$$P(\bar{g}_j) = \frac{\mathcal{Q}(\bar{g}_j)}{\sum_{j'} \mathcal{Q}(\bar{g}_{j'})}. \tag{4.9}$$

Obviously, the values of the fitness function $\mathcal{Q}$ have to non-negative. This can always be achieved by shifting ($\mathcal{Q}' = \mathcal{Q} + const$) or by exponentiation ($\mathcal{Q}' = e^{\mathcal{Q}}$). This reveals a major problem of fitness proportionate selection: the selection probabilities strongly depend on the exact definition of the fitness function. Additionally, it depends on the variance in the population. Because this variance usually decreases over time, selection pressure also decreases.

Rank-based selection tries to avoid these problems by considering solely the relative ordering of the candidate solutions' fitness values. It can be seen as a transformation of $\mathcal{Q}$: the candidate solutions are sorted by their fitness $\mathcal{Q}$ in increasing order. The least fit individual is assigned a new fitness $\mathcal{Q}_{rank} = b$ with a bias b, the next best a value of $b+1$, and so on. Selection is then performed fitness proportionate with respect to $\mathcal{Q}_{rank}$. The bias b controls selection pressure.

Good results, which are independent of transformations of $\mathcal{Q}$ and which additionally do not require sorting of the population, can be achieved with tournament selection (Goldberg, 1990; Mühlenbein and Schlierkamp-Voosen,

1993). In tournament selection, two (or more) individuals are randomly chosen from the population, and the fitter of them is selected. It often depends on the problem—and the characteristics of the fitness function—, which selection strategy yields the right balance between selection pressure and population diversity. Comparisons of several strategies can, for example, be found in (Goldberg and Deb, 1991; Blickle and Thiele, 1995).

4.1.3 Building Blocks and the Schema Theorem

The schema theorem is often used as an explanation of how genetic algorithms function (Holland, 1975). A schema H is defined as a template with fixed instantiations for a subset of the bit string, while the remaining bits can take arbitrary values. The idea is that if a schema encodes information about a good (partial) solution, the individuals that share this schema will have above-average fitness, and thus a higher reproduction probability.

The evolution of schemata can be formally described (Holland, 1975; Schaffer, 1987). Let L denote the length of the chromosomes, $O(H)$ the *order* of the schema (i.e. the number of fixed bits), and $L(H)$ the *defining length* (i.e. the distance between the first and last fixed bit position). Let $m(H,t)$ be the frequency of a schema at generation t. The averaged fitness of all individuals at generation t is denoted with $\bar{f}(t)$, that of the individuals with schema H with $f(H,t)$. If we assume fitness proportionate selection, the frequency of individuals with schema H after selection is

$$m_{sel}(H,t+1) = m(H,t) \cdot \frac{f(H,t)}{\bar{f}(t)}. \tag{4.10}$$

The frequency of above average schemata will grow exponentially due to $\frac{f(H,t)}{\bar{f}(t)} > 1$.[1] Schemes might be disrupted by crossover or mutation. H is disrupted by crossover, if the second parent does not contain the schema (probability $1 - m_{sel}(H,t+1)$) *and* the operator affects the scheme. For one-point crossover this means that the split point falls between first and last bit of the schema, thus this probability is $\frac{L(H)}{L-1}$. Crossover is applied with probability p_c. Mutation does not disrupt H if none of the $O(H)$ bits is changed. Because the probability of a bit change is the mutation probability p_m, H survives mutation with probability $(1-p_m)^{O(H)}$. Putting all together, we can estimate the probability that schema H survives from one

[1] Obviously, frequency is bound by 1. This is, however, no contradiction, as the average fitness $\bar{f}(t)$ increases with time.

generation to the next, i.e. that it is selected and not disrupted. Considering that H might also be generated from other schemata by crossover or mutation, Eq. (4.10) can be extended to the inequality

$$\begin{aligned} m(H,t+1) &\geq m_{sel}(H,t+1)\cdot(1-p_m)^{O(H)} \\ &\quad \cdot\left(1-p_c\frac{L(H)}{L-1}\left(1-m_{sel}(H,t+1)\right)\right) \\ &= m(H,t)\cdot\frac{f(H,t)}{\bar{f}(t)}\cdot(1-p_m)^{O(H)} \\ &\quad \cdot\left(1-p_c\frac{L(H)}{L-1}\left(1-m(H,t)\cdot\frac{f(H,t)}{\bar{f}(t)}\right)\right). \end{aligned} \tag{4.11}$$

The effectiveness of genetic algorithms is often explained by the large number of schemata that are tested in parallel in every generation (i.e. there are by far more schemata than individuals), and the convergence to good solutions. As can be seen in Eq. (4.11), necessary condition for the success of a schema is not only its above-average fitness, but also a low order $O(H)$ (so mutation does no harm) and low defining length $L(H)$ (so crossover does not affect it). Schemata with low defining length and high average fitness are called building blocks, as the solution is ideally assembled from these.

This explanation has been criticized for several reasons. For example, in an initial population, schemata of order O occur with a probability of $\frac{1}{2}^O$. If we consider complex problems with large L, above-average schemata will have accordingly large orders $O(H)$, and their probability to occur initially will be diminishingly small for realistic population sizes. For a survey of other points of criticism see, e.g., (Whitley, 2001).

However, we can learn from the schema theorem and Eq. (4.11) what might obstruct successful application of genetic algorithms. Although GA do not take the meaning of the bits into consideration, an intelligent encoding can support them by keeping the relevant bits together and thus keeping defining lengths of schemata short. Additionally, crossover and mutation can be selected specifically for a problem, such that they do not destroy partial solutions.

4.1.4 Discussion

The question which of the paradigms is superior is rather futile. Although followers of both schools sometimes fiercely defend their model, there is—as often—nothing like a free-lunch: It depends on the problem at hand, which

strategy will work better. As a rule of thumb, genetic algorithms have their strengths in discrete search spaces, like e.g. combinatorial problems. For structure optimization, GA with variable chromosome lengths have been proposed. For such cases, no direct ES counterpart exists. However, evolution strategies are generally better suited for (real-valued) parameter identification. As a matter of fact, there have been proposals to combine ideas of both paradigms. The *real-coded* genetic algorithms are rather successful examples (Eshelman, 1991; Schaffer and Eshelman, 1993). Besides real-valued genes, they use specialized mutation and crossover operations. As the induction of fuzzy rules implies learning of structure *and* parameters, such hybrid, specialized evolutionary algorithms are highly interesting.

Although EA are sometimes promoted as problem independent optimization algorithms that need nothing but a fitness function, they do in general not work *out of the box* in more complex search spaces. This is certainly the case for a challenging task like fuzzy rule base learning. Thus well-considered design of encoding and genetic operators is necessary. The next sections will present the most common approaches to evolutionary induction of fuzzy rule-based systems.

4.2 Evolutionary Fuzzy Rule-Based Systems

The task of learning a rule base with evolutionary algorithms has been considered in the machine learning community since the late 70's (Grefenstette, 1994). Although since then a large number of chromosome encoding schemes and genetic operators have been proposed, most of the approaches can be classified into three main categories: the "Pittsburgh-style" (Smith, 1980), the "Michigan-style" (Holland and Reitman, 1978), and the iterative rule learning (IRL) approach (Venturini, 1993). These approaches were first proposed for crisp rule induction, however, they have been extended to the learning of fuzzy rules.

Of these three approaches, the Pittsburgh-style approach is closest to the basic genetic algorithm: the goal is to optimize a rule base, and thus each chromosome represents an entire rule base. The performance of the rule bases is assessed by a fitness function. By using appropriate genetic operators, the population evolves towards better solutions. Finally, the candidate solution with the highest fitness is chosen. Examples of fuzzy rule-based systems that use Pittsburgh-style genetic learning include (Hoffmann and Pfister, 1997; Carse et al., 1996; Pham and Karaboga, 1991; Thrift, 1991; Kinzel et al., 1994). The main drawback of this approach are the

rather high computational demands due to the necessary evaluation of all rule bases in every generation. Both other approaches, Michigan-style and iterative rule learning, thus evolve individual rules.

The origin of Michigan-style approaches, namely the *Cognitive System One (CS-1)* proposed by Holland and Reitman (1978), takes a special position in genetic algorithm learning: it does not only consider competition amongst individuals with the goal of finding the single *best* individual. Instead, the individuals (i.e. the rules) are also cooperating, and the final solution (i.e. the rule base) is assembled from individuals of the whole population. Obviously, specialized fitness functions are necessary that prevent convergence of all rules to the same optimum. CS-1 uses an epoch-based credit apportionment algorithm to control cooperation and competition amongst rules. The fitness of a rule depends on the number of covered tuples and on the misclassification rate. Concurring rules, i.e. rules with antecedents that describe intersecting areas, share their covered tuples and thus get lower rewards. Thus, it pays for rules to evade to less covered areas of the search space. An adequately implemented credit apportionment system should balance between cooperation and competition, and thus evolve rules with high (joint) coverage and low misclassification rate. As the interaction between individual rules is sometimes hardly predictable, the credit apportionment system is the sore point of Michigan-style rule learning. It thus remains the focus of research. Michigan-style approaches to fuzzy rule learning can, for instance, be found in (Valenzuela-Rendón, 1991; Ishibuchi et al., 1999; Bonarini, 1996; Parodi and Bonelli, 1993).

Like Michigan-style approaches, iterative rule learning tries to reduce the search space by evolving individual rules instead of entire rule bases. However, there is only competition amongst rules of one population, and a single best rule according to a fitness measure is chosen. The rule is inserted into a rule base, and the learning task is modified to take the already covered examples into account. As the name suggests, this rule creation and selection procedure is iterated until a satisfying rule base is assembled (Venturini, 1993). Iterative rule learning divides the problem of competition and cooperation into two stages: competition between rules of a population and cooperation between new and already chosen rules. This avoids the problems of credit apportionment. However, it is not trivial to formalize optimality of a single rule in a fitness function. Fuzzy approaches to evolutionary iterative rule learning can, for example, be found in (Cordón et al., 1999, 1998; González and Pérez, 1999).

A fourth approach has been established in evolutionary induction of fuzzy rule systems: genetic optimization of the fuzzy partitions in combina-

tion with deterministic rule induction. The algorithm by Wang&Mendel (cf. Section 3.1.1) is a rather efficient method for fuzzy rule induction. However, it depends on prior definitions of fuzzy sets. Rule bases induced with inappropriate fuzzy partitions can afterwards hardly be healed by fine-tuning. An alternative approach is proposed by Cordón et al. (2001): the chromosomes do not directly represent rules, but fuzzy partitions only, i.e. they encode the granularity of the attributes, and scaling factors that allow nonlinear dilatations of the (otherwise equidistant) fuzzy partitions. Thus local sensitivities can be varied under preservation of standard linguistic labels. To measure the fitness of a chromosome, a rulebase is deterministically generated by the Wang&Mendel algorithm using the encoded fuzzy partitions. The quality of the obtained rule base is used as a fitness measure of the chromosome. Due to the one-to-one relation of chromosomes and rule bases, the algorithm is actually comparable to Pittsburgh-style learning. However, search space size is significantly reduced, though at the cost of decreased flexibility.

For the purpose of semi-supervised learning, the Pittsburgh-style approaches are most promising. In Michigan-style and iterative rule learning, there is no objective function that is minimized for the rule base. Instead, the final rule base results from the interplay of competition and cooperation. This will become more critical, if labeled and unlabeled data are considered in rule learning. Furthermore, these approaches hardly allow to incorporate a quality measure that is independent of the learning mechanism. Because our approach, that we present in Chapter 6, is based on the Pittsburgh-approach, the section below discusses the chromosome encodings and genetic operators of such rule learners in more detail.

4.2.1 Pittsburgh-Style Rule Induction

Pittsburgh-style learning of fuzzy if-then rule-based systems can be performed on several levels of complexity:

- Optimization of fuzzy sets with fixed rule base,
- Optimization of fuzzy rules with fixed fuzzy partitions,
- Optimization of fuzzy rules and fixed fuzzy sets in stages, or
- Parallel optimization of fuzzy rules and fuzzy sets.

Obviously, these learning tasks place very different demands on the evolutionary algorithm. In the following sections we present some of the most

common chromosome encodings and genetic operators that have been proposed in literature.

When judging the suitability of the approaches for our problem at hand, we have to keep in mind that many of them deal with fuzzy control. Fuzzy rules for control and classification have rather similar structure (cf. Section 2.2), and thus learning of fuzzy sets and antecedent clauses is transferable. The fuzzy sets in the consequents of control rules are simply replaced by class labels. If the defuzzified classification results, i.e. the crisp labels, are used to estimate the misclassification rate of a rule base, fitness functions defined on the basis of these misclassifications will usually have discontinuities. This can complicate optimization in comparison to smoother fitness functions in fuzzy control. Additionally, the problems in data analysis are usually of much higher dimensionality than control applications, which again makes classification harder in comparison to control. On the other hand, the best consequent for a classification rule with a given antecedent can be directly determined from the labeled data. Thus, in contrast to fuzzy control, the consequents need not to be optimized, which decreases search space size. Furthermore, classification problems can almost always be solved in *batch mode*—in contrast to control problems where learning must often be performed *online*, and sometimes even with a real plant in the control loop. Thus, bigger populations and more generations are often affordable in classification.

Optimization of Fuzzy Sets with Fixed Rule Base

If the rule base is given (e.g. by an expert), optimization of the fuzzy sets refers to fine-tuning of the membership functions. This is a relatively easy task, because setting up the rule base makes only sense within an approximately fixed context, and thus the fuzzy set positions already must be almost correct. In such cases, other methods like e.g. gradient descent might be more appropriate (cf. Section 3.1.2). However, evolutionary learning can be a feasible alternative, if involved membership functions or learning objectives prohibit gradient descent. This can, for instance, be the case for non-differentiable membership functions.

In almost all proposals parameterized fuzzy sets are used, and thus a chromosome represents these parameters. In (Klose et al., 2000), we used real-valued encodings of the parameters, but binary encodings have also been successfully applied (see, e.g., Karr, 1991). The usual triangular, trapezoidal, or Gaussian shapes of the fuzzy sets can be found. Mutation affects either the bits in binary encodings or directly the parameter values

in real-valued encodings. Accordingly, crossover exchanges bits or complete parameters.

An alternative encoding of fuzzy sets has been devised by Kinzel et al. (1994). They use a *vertical* view on the fuzzy sets, i.e. their fuzzy sets are characterized by real-valued membership values at a fixed number of sampling points of a domain. Crossover of two chromosomes thus transfers the information, where a rule will or will not hold, from parents to child. However, although a repair mechanism ensures convexity, linguistic interpretability might be lost. Additionally, precision is limited by the number of sampling points.

Optimization of Fuzzy Rules with Fixed Fuzzy Partitions

Because fixed fuzzy partitions are defined globally and a priori, these approaches always induce descriptive rule bases, i.e. rules that share the fuzzy sets. For low-dimensional problems it is common to represent the rules in a decision table with an entry for every possible antecedent (i.e. every fuzzy set combination). Rule induction then means to assign appropriate consequents to the antecedents. Additionally, a null value can be assigned to deactivate a rule. An early fuzzy control example can be found in (Thrift, 1991). In any case, rule learning is a combinatorial problem that is generally more challenging than fuzzy set tuning.

Mutation is performed by randomly replacing the values of the consequents. If the decision tables are represented in a one-dimensional gene-string, standard one- or two-point-crossover can be applied. However, adjacent rules, i.e. rules with intersecting antecedents, can lie at quite different positions on a one-dimensional chromosome. In the terminology of the schema theorem, such a schema with rules adjacent in the input space, but separated on the gene-string, has a large defining length and thus high probability to be destroyed by crossover. To support the genetic algorithm in evolving building blocks of adjacent rules, specialized crossover operators have been proposed that take spatial relations into account, e.g. by considering more-dimensional chromosome structures, where adjacent rules lie at adjacent loci. For instance, Kinzel et al. (1994) suggest to randomly choose a crossover-point in the decision table, and exchange all entries within an also randomly chosen (Manhattan-) distance from that point. Thus adjacent rules have a higher probability to end up in the same offspring (see Figure 4.1).

As the number of rules grows exponentially with the number of dimensions, representing rules in decision tables is not feasible in higher-

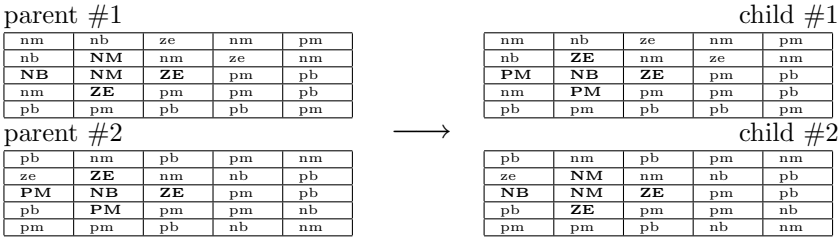


Figure 4.1: Point-radius crossover for two-dimensional decision tables. Each table cell represents a rule, its value is the rule's consequent. The letter combinations stand for linguistic terms (n=negative, p=positive, b=big, m=medium, ze=approx. zero). The entries with capital letters are chosen for crossover and exchanged (cf. Kinzel et al., 1994).

dimensional spaces and thus other encodings are needed. In contrast to fuzzy control where complete rule bases are commonly used, we are interested in small numbers of rules in classification. This often allows better interpretability and in addition avoids the pitfall of look-up tables (cf. Section 2.3.3). Partial rule bases are usually encoded as (variable sized) sets of rules. Many approaches use a variable number of rules in the chromosomes, but complete antecedents in each rule (e.g., Magdalena, 1997a).

To achieve even more flexibility, antecedents with variable numbers of clauses can be used. For instance, Hoffmann and Pfister (1997) devised a chromosome encoding inspired from *messy* GA (Goldberg et al., 1989). The genes are pairs (i, j) of indexes, interpreted as fuzzy clauses "x_i IS $\mu_{i,j}$". Any set of such clauses where at least one input and one output variable is included can be translated into a fuzzy rule: the input variables build the antecedent, the output variable(s) build the consequent. Multiple occurances of one input are interpreted as disjunction. A rule base is encoded as a set of such rules, i.e. a chromosome is a set of sets of index pairs.

The specialized crossover operators are called *cut* and *splice*. The cut operation randomly splits the parent sets into two parts,[2] and the offspring is assembled from two such subsets in the splice operation. Cut and splice are performed on the level of rules and of clauses. Hoffman and Pfister report the problem that cut and splice are rather destructive operations and that the recombined rules are thus rather random. Similar to the point-radius-crossover in decision tables, the authors suggest to match the rules before

[2] Internally, the sets are stored as lists, although interpretation as rules is position independent. However, the position in the lists is important in the genetic operations.

mating. By restricting splits to "similar" rules, information can better be passed from the parent rules to the offspring. Furthermore, recombination of sets of clauses is performed less often than split and splice of complete rule bases. The GA also has mutation operators: random changes are applied to the linguistic terms, and additionally, random clauses can be deleted from or inserted into rules.

Magdalena (1997b) proposed a different solution to the problem of protecting adjacent rules in crossover operations. He assumes the set of rules to be in "virtual decision tables", and then exchanges the rules in a variable sized range of the tables between parents. The range is described by a conjunction (over attributes) of disjunctions of linguistic terms (within an attribute) that serves as a filter for the set of rules.

Optimization of Fuzzy Rules and Fixed Fuzzy Sets in Stages

A basic way two learn both, fuzzy sets and fuzzy rules, is subsequent application of the approaches of the last sections. For instance, in (Kinzel et al., 1994), the rules of an initial rule base are optimized with fixed fuzzy sets. Afterwards, the set of rules is fixed and the fuzzy sets are fine-tuned. Encodings and genetic operators are usually different in the two phases. The separation into two stages reduces the complexity of the search space in comparison to parallel optimization. However, due to the complex interactions between rules and fuzzy sets, the globally optimal rule base might not be reachable. This might be one of the reasons why Kinzel et al. found that fuzzy set optimization after rule optimization does not have large effects on the final results.

Parallel Optimization of Fuzzy Rules and Fuzzy Sets

Parallel optimization of fuzzy rules and fuzzy sets is certainly the most challenging learning task. However, it allows to generate a rule base from scratch. There have been proposals for both, descriptive or approximative fuzzy rules. The Pittsburgh-style approaches for the induction of descriptive fuzzy rules basically combine an encoding for fuzzy partitions and one for rule bases in one chromosome. The genetic operators are adapted from those of the separate approaches. Examples can be found in (Liska and Melsheimer, 1994; Shi et al., 1999).

If approximative fuzzy rules are to be induced, the fuzzy set definitions have to be stored with every rule. Cooper and Vidal (1994) suggest variable length chromosomes, composed of fixed length rules. Each rule stores for

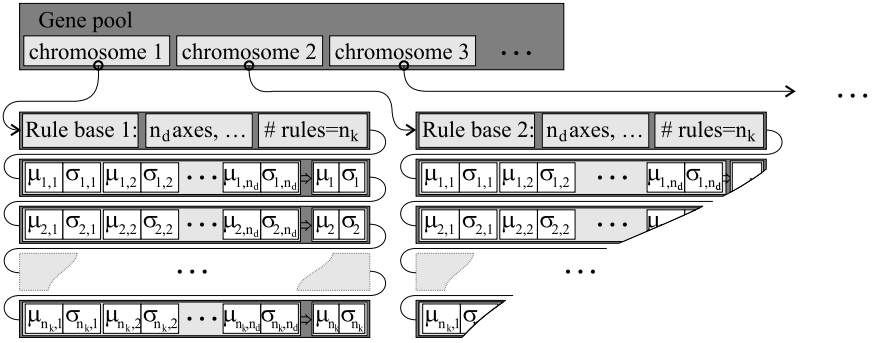


Figure 4.2: Chromosome encoding proposed by Carse et al. (1996). The centers and widths of the fuzzy sets, denoted by μ and σ, are defined locally in the rules.

every input and output dimension widths and centers of isosceles triangular fuzzy sets, binary coded with 8 bits of precision. The authors claim that genetic learning of approximative fuzzy rules failed due to a lack of appropriate reproduction operators. As crossover makes only sense between similar genes,[3] they therefore suggest to crossbreed a rule of one parent with the closest rule of the other parent (according to the Manhattan distances of the rules' centers). Mutation affects the bits as usual. Additionally, rules can be added or deleted.

Many authors prefer real-valued to binary parameter encodings. They report that convergence can often be reached faster, more robust and with higher precision. The approach presented by Carse et al. (1996) uses a chromosome encoding similar to (Cooper and Vidal, 1994), however with centers and widths encoded as float values (cf. Figure 4.2). This allows mutation operations in favor of small, less destructive changes.[4]

The crossover operation, called *one (or two) point ordered crossover*, can be seen as a continuous version of the "virtual decision table" approach in (Magdalena, 1997b). Let $[a_i, b_i]$ be the range of attribute i, and let χ denote a random number drawn from $[0, 1]$. A split point $(c_1, \ldots, c_{n_d}) \in [a_1, b_1] \times [a_{n_d}, b_{n_d}]$ is randomly chosen. Let C_{ir} denote the center of a rule

[3] "In nature it would not make much sense for the mother's gene for good vision to combine with the father's gene for curly hair." (Cooper and Vidal, 1994)

[4] The authors *multiply* the values with a factor uniformly drawn from the range $[0.9, 1.1]$, which gives different results than the *adding* of Gaussian noise common in evolution strategies.

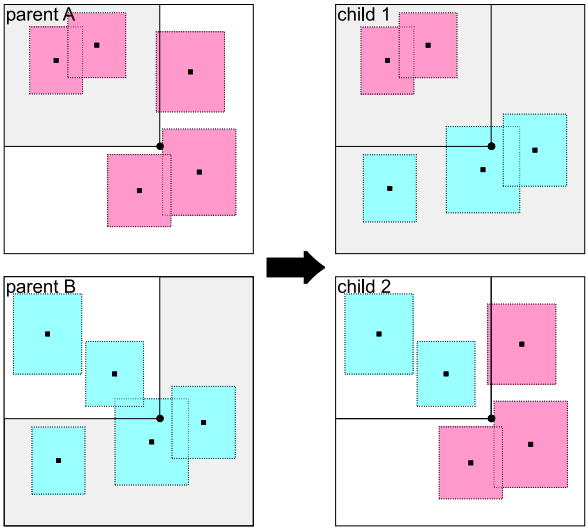


Figure 4.3: One point ordered crossover (Carse et al., 1996).

r in dimension i. Then the parents' rules are filtered according to the condition

$$\forall i : C_{ir} < c_i. \tag{4.12}$$

The first child is assembled from the rules of parent **A** that fulfill, and of rules from parent **B** that do not fulfill this condition. The second child is assembled from the remaining rules, as illustrated in Figure 4.3.

If the split point is uniformly drawn from $[a_1, b_1] \times \ldots \times [a_{n_d}, b_{n_d}]$ with $n_d \geq 2$, the hypervolume of the exchanged corner will in average be smaller than that of the remainder. Therefore, Carse et al. suggest to compensate the effects of growing dimensionality by choosing the split point as

$$c_i = a_i + (b_i - a_i) \cdot \chi^{\frac{1}{n_d}}. \tag{4.13}$$

The approach can be extended to two point ordered crossover by choosing a second split point $(c'_1, \ldots, c'_{n_d})$

$$c'_i = c_i + (b_i - a_i) \cdot \chi^{\frac{1}{n_d}}, \tag{4.14}$$

and replacing Eq. (4.12) with the following condition

$$\forall i : (c_i < C_{ir} < c'_i) \vee (C_{ir} + b_i - a_i < c'_i). \tag{4.15}$$

Instead of corners, hypercubes are exchanged in the two point version. Obviously, in both cases rules with nearby centers will be split with lower probability and can thus persist as building blocks.

4.2.2 Discussion

The evolution of approximative rules is in general significantly easier than the induction of descriptive rules. The rules, i.e. the areas described by the antecedents, are natural building blocks for good solutions. This is especially true for classification, where rules describe areas with homogeneous class distributions. An approximative rule, which locally holds its fuzzy set definitions, is thus a rather compact building block (and therefore stable with respect to crossover). A descriptive rule, on the other hand, makes only sense in connection with the fuzzy sets it refers to. If these are exchanged during crossover, the represented area in input space is very likely to change substantially. Especially if more rules refer to the same fuzzy sets—which is the idea of global fuzzy partitions—, interactions between rules can be rather complex and may obstruct optimization.

As we discussed in Section 2.3.4, global fuzzy set definitions lead to a loss of expressiveness which might be disproportionate in comparison to the gain in comprehensibility. We therefore strongly argue for approximative rules. It might be reasonable to abandon some degrees of freedom to support (or gain) interpretability.

Most of the approaches in evolutionary fuzzy rule-based systems have been devised in the early and mid 90's. Some of the recent research activities focus on special requirements of data analysis, like for instance, rule reduction in high-dimensional spaces (Cordón et al., 1999; Gómez-Skarmeta and Jiménez, 1999; Ishibuchi et al., 1997). Another interesting topic treated by Ishibuchi et al. (1997) is multi-objective optimization: if several, partially contradicting objectives shall be optimized, traditional search methods fail. Here evolutionary algorithms can be a feasible alternative (Ishibuchi et al., 1997, 2001).

4.3 Conclusions

In this chapter we outlined the basic concepts of the most common evolutionary algorithms, i.e. evolution strategies and genetic algorithms, and how these powerful search mechanisms can be used for fuzzy rule-based systems. We reviewed appropriate encodings and genetic operators, and

stressed their importance for reliable learning and fast runtimes in spite of complex search spaces.

One might wonder, why we did not divide this chapter into supervised and unsupervised approaches, like we did in Chapter 3. Actually, there has been a number proposals for unsupervised learning (i.e. clustering) with evolutionary algorithms (e.g., Hall et al., 1994; Klawonn and Keller, 1998). The reason not to make the distinction in this chapter is simply that it basically depends on the fitness function only, whether the search is supervised or unsupervised. The encodings and genetic operators are identical. This is one reason why evolutionary fuzzy rule induction seems to be suited for semi-supervised learning: this kind of supervision can be seen as a blend of supervised and unsupervised learning. In Chapter 6, we present measures for semi-supervised learning that can directly be used as fitness functions for an evolutionary fuzzy rule learner.

Chapter 5

Semi-Supervised Learning

The preceding chapters reviewed a number of common methods for the extraction of fuzzy classification rules from data. The algorithms can be divided into supervised and unsupervised approaches. Both paradigms for learning have their drawbacks. The main drawback of supervised learning obviously is its need for supervision, i.e. the need to present labels together with the objects. The result of unsupervised learning, however, strongly depends on a number of prior assumptions (explicit or implicit). Thus it depends on an appropriate choice of e.g. attribute scaling, distance measure, distribution function and expected number of classes or clusters, whether the clusters found in the data space correspond to any meaningful classes of objects. Hence in many cases unsupervised learning does not yield satisfactory results and supervised learning is much more common in practice. If labeling all objects is impractical, one usually confines the examples to a certain—hopefully representative—fraction of the data and leaves the unlabeled data aside.

If however, the remaining and otherwise discarded unlabeled data contained some additional useful information, it would be an appealing idea to use it to support the learning of the classifier. The idea of exploiting the information in both labeled and unlabeled data is not new, and early approaches of semi-supervised learning date back to the 1980's (e.g. Pedrycz, 1985). However, as argued above, with tremendously growing sizes of datasets in real-world applications labeling all of the data becomes more and more infeasible and the exploitation of additional unlabeled examples gets increasingly interesting. Thus a growing number of publications, workshops and conference tracks on semi-supervised learning could be observed

over the past years.

In this chapter, we deal with theoretical and practical aspects of semi-supervised learning. In Section 5.1, we present explanations, why learning from partially labeled data can work at all, and which preconditions are necessary. The explanations are based on theoretical considerations on the properties of the underlying structure of the data, and are supplemented with intuitive examples. Section 5.2 reviews literature on semi-supervised learning. As the goal of this thesis is semi-supervised learning of fuzzy classification rules, we focus on methods that allow to induce fuzzy models, and present details of a number of approaches which are relevant in this context. The capabilities and restrictions of the considered semi-supervised approaches are demonstrated on a number of illustrative examples (Section 5.3). In Section 5.4, we discuss their suitability to induce interpretable fuzzy classification rules, and point out the need for a specialized approach.

5.1 Learning From Examples Without Labels?

One might wonder how a learner in a supervised scenario can benefit from unlabeled examples. If their class is unknown, how should the learner adjust its model? Supervised learning can be seen as the task to reconstruct (at least partially) a probability distribution $P(\boldsymbol{x}, c)$ between objects $\boldsymbol{x} \in \boldsymbol{X}$ and class labels $c \in \mathcal{C}$ from a set of labeled examples $\mathcal{D}^l = \{\omega_i = (\boldsymbol{x}^{\omega_i}, c^{\omega_i}) | i = 1, \ldots, n_l\}$. As n_l is finite, i.e. as we have only a limited set of examples, this actually is an ill-posed problem. If we do not assume anything about P, the only fact about P which we can derive from $\mathcal{D}^l$ is that $P(\omega_i) > 0$, i.e. the examples ω_i are possible. Apart from this trivial finding, the example data do not contain any information on how to generalize on unseen data. Therefore, we always have to assume that P comes from some family of probability distributions. Learning—supervised or unsupervised—then means to estimate a specific $\hat{P}$ given a family and given the example data.

Which family of probability distributions we choose depends on our assumptions about the underlying processes that generated the data. In classification two major groups can be distinguished based on the way P can be more efficiently represented (Ripley, 1996):

- The *sampling paradigm* assumes that $P(\boldsymbol{x}, c)$ can be efficiently represented as $P(\boldsymbol{x} \mid c)$. Methods following this paradigm—so-called *generative methods*—model the probability distributions of each class individually. A simple generative method is, for instance, the naïve Bayes classifier.

- The *diagnostic paradigm* assumes that $P(\boldsymbol{x}, c)$ can be efficiently represented as $P(c \mid \boldsymbol{x})$. The associated *diagnostic* or *discriminative methods* are closer to regression, i.e. they model c as a (noisy) function of $\boldsymbol{x}$. An example of a simple classifier following the diagnostic paradigm is logistic regression.

As we will see in the following sections, the assumptions about the underlying structures play an important role for the question, whether and how additional unlabeled data can be beneficial. The theoretical considerations of the next two subsections will be illustrated with two corresponding intuitive examples in Section 5.1.3.

5.1.1 The Sampling Paradigm

In the sampling paradigm it is assumed that $P(\boldsymbol{x} \mid c)$ can be efficiently represented, i.e. it is assumed that objects of a class c *generate* a distinct distribution in the input space $\boldsymbol{X}$. Let us assume parameters π and θ that determine class priors $P(c \mid \pi)$ and conditional probability distributions $P(\boldsymbol{x} \mid c, \theta)$. With dependencies as in Figure 5.1a, we get the joint density model

$$P(\boldsymbol{x}, c \mid \pi, \theta) = P(\boldsymbol{x} \mid c, \theta) \cdot P(c \mid \pi). \tag{5.1}$$

If π and θ are given, we can classify a pattern $\boldsymbol{x}$ using Bayes' rule:

$$P(c \mid \boldsymbol{x}, \pi, \theta) = \frac{P(c \mid \pi) \cdot P(\boldsymbol{x} \mid c, \theta)}{\sum_{c' \in \mathcal{C}} P(c' \mid \pi) \cdot P(\boldsymbol{x} \mid c', \theta)}. \tag{5.2}$$

For the task of learning, i.e. the task of estimating π and θ, we could consider the marginal distribution

$$P(\boldsymbol{x} \mid \pi, \theta) = \sum_{c \in \mathcal{C}} P(\boldsymbol{x} \mid c, \theta) \cdot P(c \mid \pi), \tag{5.3}$$

and maximize the likelihood function

$$L(\mathcal{D}, \pi, \theta) = \prod_{\omega \in \mathcal{D}} P(\boldsymbol{x}^\omega, c^\omega \mid \pi, \theta). \tag{5.4}$$

One possible extension to partially supervised learning is to consider the *joint likelihood* over the labeled dataset $\mathcal{D}^l$ and the unlabeled dataset $\mathcal{D}^u$

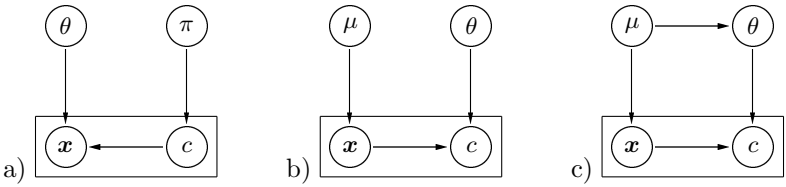


Figure 5.1: Dependency assumptions between input and class label a) in the sampling paradigm, b) in the diagnostic paradigm, and c) in the diagnostic paradigm with regularization; from (Seeger, 2001).

(Seeger, 2001), defined as

$$L(\mathcal{D}^l \cup \mathcal{D}^u, \pi, \theta) = \prod_{\omega \in \mathcal{D}^l} P(\boldsymbol{x}^\omega, c^\omega \mid \pi, \theta) \cdot \prod_{\omega \in \mathcal{D}^u} P(\boldsymbol{x}^\omega \mid \pi, \theta)$$
$$= \prod_{\omega \in \mathcal{D}^l} P(\boldsymbol{x}^\omega \mid c^\omega, \theta) P(c^\omega \mid \pi) \cdot \prod_{\omega \in \mathcal{D}^u} \sum_{c \in \mathcal{C}} P(\boldsymbol{x}^\omega \mid c, \theta) P(c \mid \pi), \quad (5.5)$$

and try to maximize it, e.g. with the expectation-maximization (EM) algorithm. There are a number of approaches that are in principle based on this joint likelihood (Miller and Uyar, 1998; Larsen et al., 2002; Nigam et al., 2000). In Section 5.2.2, we present the approach by Miller and Uyar.

5.1.2 The Diagnostic Paradigm

In the second model—the diagnostic paradigm—it is assumed that $P(c \mid \boldsymbol{x})$ can be efficiently modeled, i.e. that there is a (noisy) functional relationship between inputs $\boldsymbol{x}$ and output c. This functional dependence is given by a parameterized family of models $P(c \mid \boldsymbol{x}, \theta)$. As can be seen in Figure 5.1b, a parameter μ controls the distribution $P(\boldsymbol{x} \mid \mu)$ of the input points $\boldsymbol{x}$. Thus we can reconstruct the complete distribution as

$$P(\boldsymbol{x}, c \mid \mu, \theta) = P(c \mid \boldsymbol{x}, \theta) \cdot P(\boldsymbol{x} \mid \mu). \quad (5.6)$$

However, the dependence between μ and $\boldsymbol{x}$ is obviously not needed to predict class c for a given input $\boldsymbol{x}$.

The joint likelihood of the complete dataset $\mathcal{D}^l \cup \mathcal{D}^u$ is

$$\begin{aligned} L(\mathcal{D}^l \cup \mathcal{D}^u, \mu, \theta) &= L(\mathcal{D}^l, \mu, \theta) \cdot L(\mathcal{D}^u, \mu, \theta) \\ &= \prod_{\omega \in \mathcal{D}^l} P(\boldsymbol{x}^\omega, c^\omega \mid \mu, \theta) \cdot \prod_{\omega \in \mathcal{D}^u} P(\boldsymbol{x}^\omega \mid \mu, \theta) \\ &= \prod_{\omega \in \mathcal{D}^l} P(c^\omega \mid \boldsymbol{x}^\omega, \theta) P(\boldsymbol{x}^\omega \mid \mu) \cdot \prod_{\omega \in \mathcal{D}^u} \underbrace{\sum_{c \in \mathcal{C}} P(c \mid \boldsymbol{x}^\omega, \theta)}_{=1} P(\boldsymbol{x}^\omega \mid \mu) \\ &= \prod_{\omega \in \mathcal{D}^l} P(c^\omega \mid \boldsymbol{x}^\omega, \theta) \cdot \prod_{\omega \in \mathcal{D}^l} P(\boldsymbol{x}^\omega \mid \mu) \cdot \prod_{\omega \in \mathcal{D}^u} P(\boldsymbol{x}^\omega \mid \mu) \\ &= \prod_{\omega \in \mathcal{D}^l} P(c^\omega \mid \boldsymbol{x}^\omega, \theta) \cdot \prod_{\omega \in \mathcal{D}^l \cup \mathcal{D}^u} P(\boldsymbol{x}^\omega \mid \mu). \end{aligned} \tag{5.7}$$

From Eq. (5.7), we can conclude that $P(\theta \mid \mathcal{D}^l, \mathcal{D}^u) = P(\theta \mid \mathcal{D}^l)$ and that $P(\theta \mid \mathcal{D}^l, \mu) = P(\theta \mid \mathcal{D}^l)$. This means that neither existence of unlabeled data $\mathcal{D}^u$ nor any knowledge about the distribution of the input points $P(\boldsymbol{x})$ influence the estimation of the model parameters. Thus semi-supervised learning is not directly applicable in this paradigm.

The problem of models following the diagnostic paradigm is apparently the a priori independence of θ and μ. To remedy this problem, Seeger (2001) suggests to modify the traditional diagnostic paradigm by allowing dependencies between θ and μ as shown in Figure 5.1c. The conditional prior $P(\theta \mid \mu)$ allows to transfer information from μ—i.e. information we extracted from $\mathcal{D}^u$—to θ, and thus makes the additional use of unlabeled data feasible. However, the choice of an appropriate regularization is not trivial. One idea of incorporating unlabeled data into diagnostic methods could be to try to exploit redundancies between different views on the data.

5.1.3 An Intuitive Explanation

The preceding sections reflected on theoretical considerations, how an additional unlabeled dataset $\mathcal{D}^u$ can help to improve estimation of $P(c \mid \boldsymbol{x})$. In this section, we give some more intuitive explanations how examples without class label can be exploited.

In the sampling paradigm the class is assumed to be central in the underlying data-generating process: an object of a certain class c (e.g. "type of iris flower") is generated with a certain probability. This object will then form observable features $\boldsymbol{x}$ (e.g. "petal length"). If the probability distributions of these features are class-specific, we will find characteristic accumulations

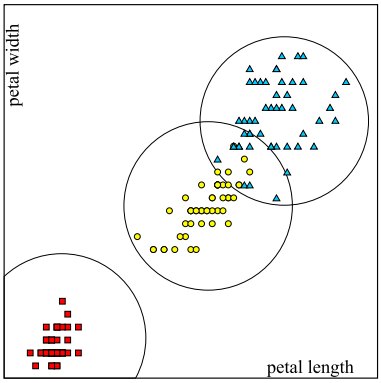
(i.e. clusters) in input space (Figure 5.2a). Therefore, the information about the distribution in the input space, extracted from unlabeled data $\mathcal{D}^u$, can be exploited to improve the estimation of cluster positions and shapes, and thus accuracy of $P(c \mid \boldsymbol{x})$ for unseen points.

How could a data-generating process falling into the diagnostic paradigm look like? For a very simple example, let us assume we gather company data, including the features *(total) income* and *(total) expenses*. Let the class label c be *profitability*, with the (noiseless) functional dependence "c is profitable, if *income* > *expenses*". However, this relationship is assumed to be unknown and shall be learned from example data. Obviously, the class does not affect the distribution of the features (Figure 5.2b). Thus, if we present random pairs *income/expenses* but withhold the corresponding class labels, the classifier will have no chance to improve its model.

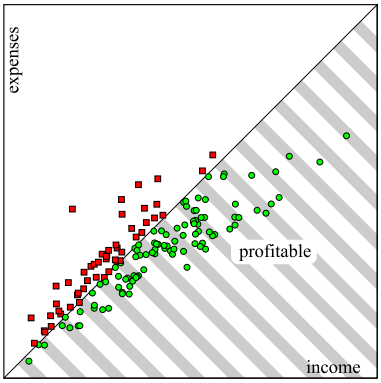
However, the data for randomly chosen companies is very probably not uniformly distributed in the input space. More likely, most companies will be distributed around the line $\frac{income}{expenses} = const$ (with $const = 1 + average\ yield$). Considerably less companies will have $\frac{income}{expenses} < 1$. We thus might be able to find a lower dimensional substructure in the data. For this extraction of redundancy, both labeled and unlabeled data can be exploited. From our updated knowledge about μ, we might also update our expectations of class distributions.[1]

The artifice of modeling a dependence $P(\theta \mid \mu)$ allows semi-supervised diagnostic methods. Although it can be helpful in some situations, it will very likely be misleading in situations where the modeled dependence does not exist, and thus the use of $\mathcal{D}^u$ can even decrease classifier performance. However, the data generating process assumed in the diagnostic paradigm seems to be more tailored to function approximation. In many classification problems, an object's class does have causal influence on the instantiations of its features, and thus the sampling paradigm is appropriate. As shown in Section 5.1.1, the corresponding *generative methods* can more naturally be extended to partially supervised learning. Fortunately, the fuzzy rule based classifiers considered in this thesis are generative, and thus, they can be expected to profit from additional unlabeled data.

[1] Obviously, extraction of lower dimensional substructures makes more sense in a higher (than two) dimensional feature space. In such cases, dimensionality reduction by, for instance, principal component analysis (PCA) is a rather common preprocessing step. It could be interpreted as an update of $P(c \mid \boldsymbol{x}, \theta)$, such that $P(c \mid \boldsymbol{x}_1, \theta) = P(c \mid \boldsymbol{x}_2, \theta)$ iff $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are projected on the same point in the subspace of the chosen principal components. As PCA is performed on unlabeled data, it can exploit $\mathcal{D}^l \cup \mathcal{D}^u$, and therefore PCA with subsequent supervised learning (using $\mathcal{D}^l$ only) is a way of partially supervised learning.



a) Iris data: each class is generated by a characteristic probability distribution (sampling paradigm).



b) Company data: the distribution of the examples is completely independent from the class information (diagnostic paradigm).

Figure 5.2: Intuitive examples for sampling and diagnostic paradigm.

5.2 Approaches to Semi-Supervised Learning

In most publications, Pedrycz's approach of 1985 is cited as the first work in the area of semi-supervised clustering. Twelve years later he revisited the problem and published some extended results and more detailed discussion of his 85's ideas. In (Pedrycz and Waletzky, 1997) he stated that

> "surprisingly, limited attention has been paid to the mechanisms of partial supervision."

Recently, partial supervision has obviously come into the focus of current research in computational intelligence. There is a growing number of publications in this field, and successful applications of semi-supervised approaches have been reported, for example in the field of image processing (Bensaid et al., 1996) and especially in text classification (Nigam et al., 2000; Lanquillon, 2001). A number of different ideas have been proposed how to combine the information of labeled and unlabeled data. Theses ideas can be categorized into four main groups:

1. **Labeled examples as seed points:** a supervised classifier is used to build a model from the class information of labeled points. The model

is then used to apply labels to the unlabeled points, and is iteratively re-learned. Approaches differ in the type of model used (e.g. point prototypes, naïve Bayes classifiers or neural networks), and the speed of applying the new labels (from one pattern per iteration to labeling all unlabeled patterns in one step) (Bensaid et al., 1996; Gabrys and Petrakieva, 2002; Lanquillon, 2001; Verikas et al., 2002).

2. **Labeled examples as cluster labels:** an unsupervised algorithm is used to find structure in the dataset, e.g. by cluster analysis. Then the clusters are labeled using the given labeled points. This can be done in various ways (Bensaid and Bezdek, 1998; Park and Yae, 2002, see also Section 5.2.5). The labeled points can also be used to guide the clustering, e.g. the number of clusters (Amar et al., 1997; Gabrys and Petrakieva, 2002). Dara et al. (2002) proposed to find low-dimensional structures by training a self-organizing map from all available data in an unsupervised manner. The map nodes are then labeled from the labeled dataset.

3. **Unlabeled examples for density estimation:** the abundance of unlabeled examples can be used for a more reliable estimation of the probability density function in the input space. This is similar to the second group, as cluster analysis also performs a kind of density estimation. However, approaches like (Skabar, 2002; Kothari and Jain, 2002) explicitly model and use the probability density function. A different approach has been proposed in (Bennett and Demiriz, 1998; Fung and Mangasarian, 1999) for the semi-supervised learning of support vector machines. Instead of using regions of high density to find clusters, regions of data scarcity are used to find the optimal class borders. Verikas et al. (2002) propose a similar method for the learning of feed-forward neural networks.

4. **Specialized objective functions:** there is a variety of approaches which have specialized objective functions that can take into account labeled *and* unlabeled examples, for example by adding a penalty term for labeled examples that are assigned to wrong clusters (Pedrycz, 1985; Pedrycz and Waletzky, 1997; Timm, 2002). In (Demiriz et al., 2002) a mixture of cluster dispersion and cluster impurity is optimized. Our approach presented in Chapter 6 also belongs into this group.

This categorization can only give a rough overview. The borders between the categories are not crisp, and many approaches could be assigned to more than one category, depending on the point of view.

A number of approaches have been proposed for models like neural networks or support vector machines, that are generally hardly human understandable. Little or no work has been done on the semi-supervised extraction of (descriptive) fuzzy rules. In the following sections, we describe details of a number of semi-supervised algorithms. The described methods have been chosen with respect to the goal of this thesis to construct a semi-supervised fuzzy rule learner. Therefore, on one hand, we describe generic methods that can be used with many—and thus *also* with fuzzy rule-based—classifiers (Section 5.2.1). On the other hand, we describe methods closely related to fuzzy rules, for example fuzzy clustering or mixture of Gaussians classifiers (Sections 5.2.2 and 5.2.3).

5.2.1 Wrapper Methods

Some methods of semi-supervised learning can be seen as "wrappers" around fully supervised classifiers, i.e. they are built on top of a supervised classifier which is basically used without modifications. One method of handling a set of partially labeled data is to discard the unlabeled data and use an arbitrary supervised classifier on the remaining labeled data. Although this form of learning obviously cannot really be called semi-supervised, currently it is the most common method to deal with partially labeled data in practice. In our experiments, this approach will be considered as a benchmark for other methods.

Static labeling (Gabrys and Petrakieva, 2002) is another method that can be used with any supervised classifier. Initially, the classifier is also induced on the labeled data only. Then the obtained classifier is used to label the unlabeled data. The resulting completely labeled dataset is used to induce the final classifier. It should be noted that, although static labeling can be performed with any classifier, it does not seem to make sense with every classifier, as initial errors will likely be reproduced in the final classifier. In (Gabrys and Petrakieva, 2002) this approach is used in combination with a nearest neighbor classifier.[2]

As mentioned in Section 5.1.3, the unlabeled data can be used for *regularization* in preprocessing steps, like, for example, for dimensionality reduction. All available data could be processed by principle component analysis (PCA) or self-organizing maps (SOM), and an arbitrary supervised classifier be applied to the labeled data in the resulting subspace (see, for example,

[2]Nearest neighbor classification seems not to profit from static labeling in a semi-supervised setting: In (Gabrys and Petrakieva, 2002), it performs equally or even worse than the benchmark method of discarding the unlabeled data.

Dara et al., 2002). However, as the meaning of the attributes is usually lost in the subspace transformation, this method is of limited interest for the induction of understandable fuzzy rules.

The most promising wrapper methods—and those which really deserve the name "wrapper"—are the (non-static) *iterative labeling approaches*. They rely on a fully supervised classifier, which is required to return some kind of confidence information about its decisions.[3] The classifier, called the *base classifier*, is enclosed in a loop with the following steps:

1. Start with a labeled dataset $\mathcal{D}^l$, an unlabeled dataset $\mathcal{D}^u$, and an initially empty set $\mathcal{D}^{l'}$. Build an initial classifier from $\mathcal{D}^l$ only.
2. Apply the current classifier to $\mathcal{D}^u$.
3. Use predicted labels and confidence values to assign labels to one or more points from $\mathcal{D}^u$. Remove these points from $\mathcal{D}^u$ and add them to $\mathcal{D}^{l'}$.
4. Build a classifier from $\mathcal{D}^l \cup \mathcal{D}^{l'}$.
5. While $\mathcal{D}^u$ is not empty, jump to 2.

There are several alternative ways how to choose the patterns to be labeled in step 3. Gabrys and Petrakieva suggest to use only one pattern at a time, i.e. that pattern which has been classified with the highest degree of confidence (Gabrys and Petrakieva, 2002).[4] This approach can be varied by selecting the b most confidently classified examples, where $b > 1$ is a fixed number. Instead of a fixed number of examples, the algorithm can also choose examples based on the relative confidence values. For example, with $\text{conf}_{\max}$ as the highest confidence obtained in step 2, and a constant $\alpha \in [0, 1]$, the algorithm could select all $\omega \in \mathcal{D}^u$ with $\text{conf}(\omega) \geq \alpha \cdot c_{\max}$ for labeling in step 3.

If the base classifier assigns *soft* labels (e.g. posterior class probabilities) and if it can also learn from input data with soft labels, the iterative labeling can be slightly modified. In contrast to the algorithm described above, all unlabeled examples are (re-)labeled in each iteration based on the current classifier (see, for example, Lanquillon, 2001):

[3] Depending on the type of classifier, this confidence can be derived from, for example, the distance to the nearest neighbor, a posterior probability, a fuzzy membership degree, etc.

[4] As they combine this rule with a nearest neighbor classifier, this approach becomes similar to agglomerative single-linkage clustering (Jain and Dubes, 1988) in the sense that it tends to build chain-like instead of compact structures. This makes this approach rather sensitive to noise.

1. Start with a labeled dataset $\mathcal{D}^l$, an unlabeled dataset $\mathcal{D}^u$. Build an initial classifier from $\mathcal{D}^l$ only.
2. Apply the current classifier to $\mathcal{D}^u$.
3. Assign the predicted *soft* labels to *all* points from $\mathcal{D}^u$.
4. Build a classifier from $\mathcal{D}^l \cup \mathcal{D}^u$.
5. Repeat from 2, until the labels of $\mathcal{D}^u$ converge (or another stopping criterion is fulfilled).

An extension of this scheme is presented by Verikas et al. (2002). Instead of directly using the predicted soft labels in step 3, they build an average of the labels over the k nearest neighbors of an unlabeled example. For this averaging, they use the (crisp) labels of the labeled examples and the soft labels determined by the base classifier (a feed-forward neural network).

5.2.2 Gaussian Mixtures and Joint Likelihood

Miller and Uyar proposed a semi-supervised learning algorithm for a generalized *mixture of Gaussians* classifier (Miller and Uyar, 1997, 1998). Mixture of Gaussians classifiers model posterior class probabilities by superimposing a number of Gaussian density distributions each of which is associated with a soft (probabilistic) class label. The posterior class probabilities of these classifiers are defined as

$$P(c \mid \boldsymbol{x}) = \sum_{m=1}^{n_k} \left(\frac{P(m)p(\boldsymbol{x} \mid \theta_m)}{\sum_{m'=1}^{n_k} P(m')p(\boldsymbol{x} \mid \theta_{m'})} \right) P(c \mid m), \tag{5.8}$$

where p is a Gaussian density function, parameterized by component specific mean values and covariance matrices $\theta_m = (\mu_m, \Sigma_m)$, i.e.

$$p(\boldsymbol{x} \mid \theta_m) = \frac{1}{\sqrt{\det(2\pi\Sigma_m)}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \mu_m)^T \Sigma_m^{-1} (\boldsymbol{x} - \mu_m) \right). \tag{5.9}$$

The n_k components are mixed with non-negative priors $P(m)$, $\sum_m P(m) = 1$. The relation of components m to classes c is controlled by the posteriors $P(c \mid m)$, $\sum_c P(c \mid m) = 1$.[5] These can be interpreted as soft labels of the components.

[5] In contrast to this generalized model, in the original model the components are crisply assigned to the classes, i.e. $P(c \mid m) \in \{0, 1\}$.

Basis of the semi-supervision is the joint log likelihood over labeled and unlabeled data, as introduced in Section 5.1.1 (cf. Eq. (5.5)). In the following, we present the method proposed by Larsen et al. (2002), which uses a subtle modification of the joint log likelihood. Larsen et al. extended the joint likelihood by a factor α to balance between labeled and unlabeled data. They then try to maximize the joint log likelihood

$$\begin{aligned}
&\log L(\mathcal{D}^l \cup \mathcal{D}^u, \theta) \\
&= \sum_{\omega \in \mathcal{D}^l} \log p(\boldsymbol{x}^\omega, c^\omega \mid \theta) + \alpha \sum_{\omega \in \mathcal{D}^u} \log p(\boldsymbol{x}^\omega \mid \theta) \\
&= \sum_{\omega \in \mathcal{D}^l} \log \sum_{m=1}^{n_k} p(\boldsymbol{x}^\omega \mid m, \theta) P(c^\omega \mid m, \theta) P(m) \\
&\quad + \alpha \sum_{\omega \in \mathcal{D}^u} \log \sum_{m=1}^{n_k} p(\boldsymbol{x}^\omega \mid m, \theta) P(m). \qquad (5.10)
\end{aligned}$$

They use the EM algorithm initialized as followed:

- The input vectors $\boldsymbol{x}$ of n_k randomly chosen examples are taken as initial mean vectors. All initial covariance matrices Σ_m are estimated from the combined data $\mathcal{D}^l \cup \mathcal{D}^u$.
- The component priors $P(m)$ are set to $\frac{1}{n_k}$.
- The class priors $P(c)$ are set from the class frequencies in $\mathcal{D}^l$, the class-component posteriors are set to the same values $P(c \mid m) = P(c)$.

Starting from these initial values, the final values are gained by iterating a number of updating steps. First it is determined how the probability mass of the examples is distributed to the components. The corresponding component posteriors for the labeled data $\omega \in \mathcal{D}^l$ are defined as

$$P(m \mid \boldsymbol{x}^\omega, c^\omega) = \frac{p(\boldsymbol{x}^\omega \mid \theta_m) P(c^\omega \mid m) P(m)}{\sum_{m'=1}^{n_c} p(\boldsymbol{x}^\omega \mid \theta_{m'}) P(c^\omega \mid m') P(m')}, \qquad (5.11)$$

and those for the unlabeled data $\omega \in \mathcal{D}^u$ as

$$P(m \mid \boldsymbol{x}^\omega) = \frac{p(\boldsymbol{x}^\omega \mid \theta_m) P(m)}{\sum_{m'=1}^{n_c} p(\boldsymbol{x}^\omega \mid \theta_{m'}) P(m')}. \qquad (5.12)$$

Next, the mean vectors are updated:

$$\mu_m = \frac{\sum_{\omega \in \mathcal{D}^l} \boldsymbol{x}^\omega P(m \mid \boldsymbol{x}^\omega, c^\omega) + \alpha \sum_{\omega \in \mathcal{D}^u} \boldsymbol{x}^\omega P(m \mid \boldsymbol{x}^\omega)}{\sum_{\omega \in \mathcal{D}^l} P(m \mid \boldsymbol{x}^\omega, c^\omega) + \alpha \sum_{\omega \in \mathcal{D}^u} P(m \mid \boldsymbol{x}^\omega)}. \qquad (5.13)$$

With these new mean vectors and $S_m^\omega = (\boldsymbol{x}^\omega - \mu^m)(\boldsymbol{x}^\omega - \mu^m)^T$ the new covariance matrices are estimated as[6]

$$\Sigma_m = \frac{\sum_{\omega \in \mathcal{D}^l} S_m^\omega P(m \mid \boldsymbol{x}^\omega, c^\omega) + \alpha \sum_{\omega \in \mathcal{D}^u} S_m^\omega P(m \mid \boldsymbol{x}^\omega)}{\sum_{\omega \in \mathcal{D}^l} P(m \mid \boldsymbol{x}^\omega, c^\omega) + \alpha \sum_{\omega \in \mathcal{D}^u} P(m \mid \boldsymbol{x}^\omega)}. \tag{5.14}$$

Finally, the component priors and class-component posteriors are updated as

$$P(m) = \frac{\sum_{\omega \in \mathcal{D}^l} P(m \mid \boldsymbol{x}^\omega, c^\omega) + \alpha \sum_{\omega \in \mathcal{D}^u} P(m \mid \boldsymbol{x}^\omega)}{n_l + \alpha n_u} \text{ and} \tag{5.15}$$

$$P(c \mid m) = \frac{\sum_{\omega \in \mathcal{D}^l \wedge c^\omega = c} P(m \mid \boldsymbol{x}^\omega, c^\omega)}{\sum_{\omega \in \mathcal{D}^l} P(m \mid \boldsymbol{x}^\omega, c^\omega)}. \tag{5.16}$$

These steps are repeated until convergence.

The core equations of this semi-supervised learning algorithm are obviously Eq. (5.11) and (5.12). The unlabeled examples are distributed to the components based solely on the components' priors and density functions. For the labeled examples these probabilities degrees are increased or decreased depending on $P(c^\omega \mid m)$, i.e. based on the compatibility of the class labels of examples and components.

5.2.3 Extensions to Fuzzy Clustering

Semi-Supervised FCM: ssFCM

Bensaid et al. (1996) proposed an extension of the fuzzy c-means algorithm (cf. Section 3.2.1). Datasets $\mathcal{D}^u$ and $\mathcal{D}^l$ are concatenated with a common membership matrix U with $n_l + n_u$ columns and n_c rows (i.e. one row for each class $c \in \mathcal{C}$). The labels of $\mathcal{D}^l$ are presumed to be correct. This is taken into account by setting the corresponding columns in U to the 1-in-n encoded class labels, i.e. let j_ω denote the column index corresponding to object ω, then

$$u_{ij_\omega} = \begin{cases} 1, & \text{if } c^\omega = c_i \\ 0, & \text{else.} \end{cases} \tag{5.17}$$

These columns are fixed, i.e. they are ignored while updating U from the current cluster prototypes V (cf. Chapter 3.2, Eq. (3.8)). With the fixed

[6] In (Larsen et al., 2002), the EM algorithm of (Miller and Uyar, 1997) is extended in two ways to improve robustness and generalizability. First, means and covariances are estimated from independent splits of the data. Second, the covariances are slightly regularized towards the initial covariance matrix estimated from the complete dataset.

memberships for $\mathcal{D}^l$, Eq. (3.9) for updating the prototypes V remains unchanged. Instead of randomly initializing V, the labeled examples are used as initial centers, i.e. an initial V is calculated from Eq. (3.9) with $u_{ij_\omega} = 0$ for rows corresponding to unlabeled objects $\omega \in \mathcal{D}^u$.

As the influence of the labeled examples can easily be overwhelmed if $|\mathcal{D}^u| \ll |\mathcal{D}^l|$, Bensaid et al. suggest to use weights α^ω for the labeled examples ω to balance the influences of $\mathcal{D}^u$ and $\mathcal{D}^l$. Thus Eq. (3.9) is replaced by

$$v_i = \frac{\sum_{\omega \in \mathcal{D}^l \wedge c^\omega = c_i} \alpha^\omega \boldsymbol{x}^\omega + \sum_{\omega \in \mathcal{D}^u} u_{ij_\omega}^m \boldsymbol{x}^\omega}{\sum_{\omega \in \mathcal{D}^l \wedge c^\omega = c_i} \alpha^\omega \quad + \sum_{\omega \in \mathcal{D}^u} u_{ij_\omega}^m}. \tag{5.18}$$

If nothing is known about the reliability of individual examples or classes, the same α^ω is chosen for all $\omega \in \mathcal{D}^l$. Obviously, if α^ω are positive integers, Eq. (5.18) is equal to Eq. (3.9) for a dataset with α^ω copies of each labeled example ω. The authors give only some vague hints on how to choose these weights: α^ω should be large, if $\mathcal{D}^l$ is small, and small, if the reliability of the labels is low.

The idea of this approach simply is to trust the labeled examples, and fix them in the (otherwise almost unmodified) FCM clustering algorithm. Trusting the labeled examples of course is reasonable. The problem is that, as the authors also remark, this approach expects the labeled examples to be good estimations of the (final) cluster prototypes. If the chosen (or given) labeled examples are untypical for their cluster, they will unwantedly attract the prototype. On the other hand, if they already were good estimations of the cluster centers, it would not be necessary to take the additional unlabeled data into account, and we could simply use any supervised approach on them.

The authors note that the straightforward modification of the update equations interferes with the minimization of an objective function, like J from Eq. (3.7) that the original FCM is based on. Alternatively, the approach of the following section puts the focus on the extension of the objective function, and derives modified update rules.

Partially Supervised Gustafson and Kessel

The approach by Pedrycz (1985); Pedrycz and Waletzky (1997) extends the objective function of Gustafson and Kessel (see Eq. (3.11)). In the same way as in the previous section, the (input vectors of the) datasets $\mathcal{D}^u$ and $\mathcal{D}^l$ are assumed to be joined in a matrix X. Additionally, a vector $\vec{b} = [b_j]$

with binary entries

$$b_{j_\omega} = \begin{cases} 1, & \text{if } \omega \in \mathcal{D}^l \\ 0, & \text{else} \end{cases} \tag{5.19}$$

marks the labeled examples. A matrix F contains the known labels in 1-in-n encoding (cf. Eq. (5.17)). The original Gustafson&Kessel objective function J_m^{GK} is extended by adding a penalty term

$$J_m(U, \boldsymbol{V}, \boldsymbol{A}) = J_m^{GK}(U, \boldsymbol{V}, \boldsymbol{A}) + \alpha \sum_{i=1}^{n_k} \sum_{j=1}^{n} (u_{ij} - f_{ij} b_j)^m \|\boldsymbol{x}_j - v_i\|_{A_i}^2, \tag{5.20}$$

where α is intended to balance the influence of the labeled data. Let us define two weighting constants α^u and α^l for the unsupervised and supervised part of the objective function, with $\alpha^u = \frac{1}{1+\alpha}$, $\alpha^l = \frac{\alpha}{1+\alpha}$. Then minimizing Eq. (5.20) is equal to minimizing

$$J_m(U, \boldsymbol{V}, \boldsymbol{A}) = \alpha^u J_m^{GK}(U, \boldsymbol{V}, \boldsymbol{A}) + \alpha^l \sum_{i=1}^{n_k} \sum_{j=1}^{n} (u_{ij} - f_{ij} b_j)^m \|\boldsymbol{x}_j - v_i\|_{A_i}^2. \tag{5.21}$$

Intuitively, the additional term penalizes labeled points that have low membership degrees to the cluster suggested by their label.

The update rules for $\boldsymbol{V}$ and $\boldsymbol{A}$ which result from minimizing J_m remain identical to those of minimizing J_m^{GK} (cf. Eq. (3.9), Eq. (3.12), and Eq. (3.13)). The modified update rule for U (for $m = 2$) can be shown to be (Pedrycz and Waletzky, 1997)

$$u_{ij} = \alpha^u \frac{1 + \alpha(1 - b_j \sum_{i'=1}^{n_k} f_{i'j})}{\sum_{i'=1}^{n_k} \frac{\|v_i - x_j\|_{A_i}^2}{\|v_{i'} - x_j\|_{A_{i'}}^2}} + \alpha^l f_{ij} b_j. \tag{5.22}$$

For unlabeled examples (i.e. for $b_j = 0$) this leads to the original update rule (cf. Eq. (3.8))

$$u_{ij} = \frac{1}{\sum_{i'=1}^{n_k} \frac{\|v_i - x_j\|_{A_i}^2}{\|v_{i'} - x_j\|_{A_{i'}}^2}}. \tag{5.23}$$

Because we assume 1-in-n encoding, it follows that $\forall j : \sum_{i=1}^{n_k} f_{ij} = 1$, and thus the labeled examples are updated with

$$u_{ij} = \alpha^u \frac{1}{\sum_{i'=1}^{n_k} \frac{\|v_i - x_j\|_{A_i}^2}{\|v_{i'} - x_j\|_{A_{i'}}^2}} + \alpha^l f_{ij}. \tag{5.24}$$

As expected, for $\alpha = 0$ (and thus $\alpha^u = 1$, $\alpha^l = 0$), the labels of $\mathcal{D}^l$ are completely ignored. Interestingly, for $\alpha \to \infty$ (and thus $\alpha^u \to 0$, $\alpha^l \to 1$), the fuzzy ISODATA with partial supervision becomes equal to ssFCM (except that the A_i are identity matrices in ssFCM). For intermediate values of α, the memberships of the labeled data are more or less pushed towards their corresponding clusters (by multiplying the normally resulting memberships with α^u and adding the differences to the membership corresponding to the label).

Both approaches expect that each class can be represented by one cluster, i.e. $n_k = n_c$. If a class cannot be appropriately described by one cluster, the algorithms will perform significantly worse (see Section 5.3.1 for an example). Timm (2002) proposed a similar approach, with a modified objective function taking the form

$$J_m(U, \boldsymbol{V}, \boldsymbol{A}) = J_m^{GK}(U, \boldsymbol{V}, \boldsymbol{A}) + \alpha \sum_{i=1}^{n_k} \sum_{j=1}^{n} (1 - \text{class}(i,j))\, u_{ij}^m, \tag{5.25}$$

where $\text{class}(i,j)$ is defined to take value 1, if the j-th example is labeled and its class is represented by the i-th cluster, or value 0 else. This leads to the update rule

$$u_{ij} = \frac{1}{\sum_{i'=1}^{n_k} \left(\frac{\|v_i - x_j\|^2_{A_i} + \alpha(1-\text{class}(i,j))}{\|v_{i'} - x_j\|^2_{A_{i'}} + \alpha(1-\text{class}(i',j))} \right)^{\frac{1}{m-1}}} \tag{5.26}$$

for the membership matrix U. The updating scheme for the cluster prototypes $\boldsymbol{V}$ and $\boldsymbol{A}$ remains unchanged. Timm used this approach for clustering of fully labeled datasets, but he mentions that it can also be applied to semi-supervised learning. Although the function $class(\cdot,\cdot)$ allows several clusters to belong to one class, correspondence has to be defined in advance.

Gaussian Mixtures Revisited

The approach described in Section 5.2.2 was based on an EM optimization of the joint log likelihood derived from the sampling paradigm assumption (cf. Section 5.1.1). The performed update steps are very close to those of the alternating optimization in fuzzy clustering. For a better comparison of the generalized Gaussian mixtures approach to the semi-supervised clustering approaches described in this section, we rewrite Eqs. (5.11) to (5.16) on the basis of the Gaussian distance function used in Gath&Geva fuzzy clustering (cf. Eq. (3.14) in Section 3.2.2).

Let us replace mean vectors μ_m by prototype vectors v_m and covariance matrices Σ_m by C_m. Using Eq. (5.9) we get

$$\begin{aligned}
& p(\boldsymbol{x} \mid \theta_m) P(m) \\
&= \frac{1}{\sqrt{\det(2\pi C_m)}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-v_m)^T C_m^{-1}(\boldsymbol{x}-v_m)\right) P(m) \\
&= \left(\frac{\sqrt{(2\pi)^{n_d}\det(C_m)}}{P(m)} \exp\left(\frac{1}{2}(\boldsymbol{x}-v_m)^T C_m^{-1}(\boldsymbol{x}-v_m)\right)\right)^{-1} \\
&= \left(\frac{(2\pi)^{\frac{n_d}{2}}}{P(m)} \sqrt{\det(C_m)} \exp\left(\frac{1}{2}(\boldsymbol{x}-v_m)^T C_m^{-1}(\boldsymbol{x}-v_m)\right)\right)^{-1} \\
&= \left(d^2_{C_m,GG}(v_m,\boldsymbol{x})\right)^{-1}, \text{ with } \rho_m = \frac{P(m)}{(2\pi)^{\frac{n_d}{2}}},
\end{aligned} \tag{5.27}$$

i.e. the reciprocal value of the distance function used in Gath&Geva clustering is equal to the component contribution $p(\boldsymbol{x} \mid \theta_m)P(m)$. We can now insert this equivalence into the membership update equation Eq. (3.15):

$$\begin{aligned}
u_{mj} &= \left(\sum_{m'=1}^{n_k} \frac{d^2_{C_m,GG}(v_m,x_j)}{d^2_{C_{m'},GG}(v_{m'},x_j)}\right)^{-1} \\
&= \frac{\frac{1}{d^2_{C_m,GG}(v_m,x_j)}}{\sum_{m'=1}^{n_k} \frac{1}{d^2_{C_{m'},GG}(v_{m'},x_j)}} \\
&= \frac{p(\boldsymbol{x}_j \mid \theta_m)P(m)}{\sum_{m'=1}^{n_k} p(\boldsymbol{x}_j \mid \theta_{m'})P(m')} \\
&\overset{\text{Eq. (5.12)}}{=} P(m \mid \boldsymbol{x}_j).
\end{aligned} \tag{5.28}$$

We see that the membership degrees u_{mj} of tuples j to clusters m calculated by Gath&Geva are identical to Eq. (5.12), the EM estimation of the probability $P(m \mid \boldsymbol{x}_j)$ that an unlabeled tuple j belongs to component m.

If we want to extend Gath&Geva to semi-supervised clustering, we consequently can define the membership update function for the labeled data as $u_{mj} = P(m \mid \boldsymbol{x}_j, c_j)$, according to Eq. (5.11). It can easily be shown that with these definitions the updating of prototypes (i.e. mean values and covariances) remains identical in Gaussian mixture fitting and (then semi-supervised) Gath&Geva. The balancing factor α can be represented by tuple weights for the unlabeled data. In contrast to the algorithm pre-

sented in Section 3.2.2, the cluster sizes ρ_m are not fixed in advance, but change with updated component priors $P(m)$.

We have seen in the previous sections that for FCM and Gustafson&Kessel the membership update equations for the unlabeled data remain equal in the unsupervised and the semi-supervised versions. Thus we now compare the update equations for the labeled examples with those resulting for the semi-supervised Gath&Geva clustering.

The membership degrees for labeled data are updated in two steps. First, the class-cluster correspondences are estimated, which can be seen as determining (soft) labels l_{mc} of the clusters:

$$\begin{aligned} l_{mc} &:= P(c \mid m) = \frac{\sum_{\omega \in \mathcal{D}^l \wedge c^\omega = c} P(m \mid \boldsymbol{x}^\omega, c^\omega)}{\sum_{\omega \in \mathcal{D}^l} P(m \mid \boldsymbol{x}^\omega, c^\omega)} \\ &= \frac{\sum_{j=1}^{n_l} I(c, c_j) u_{mj}}{\sum_{j=1}^{n_l} u_{mj}}, \end{aligned} \tag{5.29}$$

with an indicator function $I(c, c')$ that assumes value one if $c = c'$ and zero otherwise. These labels then are used to update the membership degrees of the labeled examples as follows:

$$\begin{aligned} u_{mj} &= P(m \mid \boldsymbol{x}_j, c_j) \\ &= \frac{p(\boldsymbol{x}_j \mid m) P(m) P(c_j \mid m)}{\sum_{m'=1}^{n_k} p(\boldsymbol{x}_j \mid m') P(m') P(c_j \mid m')} \\ &= \frac{p(\boldsymbol{x}_j \mid m) P(m) l_{mj}}{\sum_{m'=1}^{n_k} p(\boldsymbol{x}_j \mid m') P(m') l_{mj'}} \\ &= \frac{\frac{1}{d^2_{C_m, GG}(v_m, x_j)} l_{mj}}{\sum_{m'=1}^{n_k} \frac{1}{d^2_{C_{m'}, GG}(v_{m'}, x_j)} l_{m'j}} \\ &= \left(\sum_{m'=1}^{n_k} \frac{d^2_{C_m, GG}(v_m, x_j) l^{-1}_{mj}}{d^2_{C_{m'}, GG}(v_{m'}, x_j) l^{-1}_{m'j}} \right)^{-1}. \end{aligned} \tag{5.30}$$

This can be seen as another variant of semi-supervised fuzzy clustering. Notice that if we replace the soft labeling of Eq. (5.29) by crisp labeling fixed to a one-to-one correspondence of clusters and classes, i.e. $l_{mc} = I(m, c)$, Eq. (5.30) becomes equal to the ssFCM approach (cf. Section 5.2.3). However, the semi-supervised mixtures of Gaussians approach allows to use more clusters than classes and does not need initial assignments of clusters to classes (in contrast to, e.g., the approach by Timm (2002)).

5.2.4 An Approach Based on Dispersion and Impurity

All previously presented approaches have in common that their objective functions basically use the same term for the unsupervised part of learning: the customary weighted sum of squared distances $\sum_{i=1}^{n_k} \sum_{j=1}^{n} u_{ij}^m \|x_j - v_i\|_A^2$ (cf. Section 3.2.1, Eq. (3.7)). Then this term is combined with a second term that incorporates the information from the labeled examples.

Demiriz et al. (1999, 2002) explicitly combine a measure of cluster dispersion and a measure of cluster impurity for their semi-supervised clustering approach. The cluster dispersion is calculated from all tuples $\mathcal{D}^l \cup \mathcal{D}^u$, ignoring the known labels. Demiriz et al. propose to use the inverse Davies-Bouldin index, which is described in Section 3.2.3. The Davies-Bouldin index is more common for cluster validation. However, here it is used to measure how well the rules cover the tuples in input space, which can be characterized by small, well separated clusters.

Cluster impurity, on the other hand, measures how the known labels are distributed to the clusters (i.e. the influence regions of the rules). A very simple measure of impurity would be the number of misclassifications. However, more sophisticated measures take the distribution of misclassification into account, and prefer configurations where the misclassifications are concentrated in a few clusters. Demiriz et al. use the Gini-Index which is well known for the induction of decision trees (Breiman et al., 1984). Let $\boldsymbol{X}_{il}$ denote the labeled examples of class l assigned to cluster i (i.e. cluster i yields maximal activation), and let $\boldsymbol{X}_i$ be the total number of labeled examples assigned to cluster i. Then the Gini-index for a cluster i is defined as

$$\text{gini}_i = 1 - \sum_{l=1}^{n_c} \left(\frac{|\boldsymbol{X}_{il}|}{|\boldsymbol{X}_i|} \right)^2 . \tag{5.31}$$

The impurity measure is combined from the n_k rules' Gini-indexes:

$$\text{impurity} = \sum_{i=1}^{n_k} \frac{|\boldsymbol{X}_i|}{|\boldsymbol{X}|} \cdot \text{gini}_i . \tag{5.32}$$

The objective function for the semi-supervised algorithm is a linear combination of the two measures:

$$J^{DB} = \alpha \cdot \text{impurity} + (1 - \alpha) \cdot \text{dispersion}, \alpha \in [0, 1]. \tag{5.33}$$

To use the found clusters for classification, the label for any non-empty cluster is chosen as the majority class of labeled points belonging to that

cluster. Each unlabeled object is supposed to belong to the nearest cluster prototype.

As these assignments of objects to clusters are crisp, the objective function J^{DB} has many steps and is not differentiable. Thus alternating optimization as in the other approaches is not feasible. The authors propose a evolutionary algorithm for minimization of J^{DB}. They use Pittsburgh-style chromosomes that represent the n_k n_d-dimensional cluster centers as floats. The algorithm is rather straightforward, with unspecialized operators taken from the GAlib (Wall, 1999). Details can be found in (Demiriz et al., 2002).

5.2.5 Semi-Supervised Point-Prototype Clustering

Bensaid and Bezdek (1998); Labzour et al. (1998) propose a semi-supervised point-prototype clustering algorithm that is based on the fuzzy c-means algorithm. The algorithm, called ssPPC, first overpartitions the unlabeled input patterns in a fully unsupervised manner using fuzzy c-means (although the authors remark that any point-prototype cluster algorithm can be used). Then the resulting clusters are labeled based on the labeled examples. In a final step, the unlabeled tuples are labeled based on their memberships to the clusters.

The algorithm performs the following steps:

- Cluster $\mathcal{D}^u$ with fuzzy c-means. The number of clusters n_k is heuristically chosen as n_l, i.e. there is one cluster for each labeled example. Let $\boldsymbol{V} = \{v_1, \ldots, v_{n_k}\}$ denote the resulting cluster prototypes and $U_{(n_k, n_u)}$ the matrix of cluster memberships.

- Assign class labels to the cluster prototypes v_i. The labels of the prototypes $L_{(n_k, n_c)}$ can be possibilistic, i.e. $l_{ij} \in [0, 1]$. Three alternative strategies have been proposed to find labels (Labzour et al., 1998):

  A. For each prototype v_i find the nearest labeled example from $\mathcal{D}^l$ and adopt its (crisp) label.

  B. Assign each labeled example to its nearest prototype. Let ${\mathcal{D}^l}_j$ be the labeled examples of class j, and ${\mathcal{D}^l}_{ij} \subset {\mathcal{D}^l}_j$ the subset assigned to prototype v_i. Use their fraction to define possibilistic class labels:

$$l_{ij} = \frac{{\mathcal{D}^l}_{ij}}{{\mathcal{D}^l}_j}. \tag{5.34}$$

C. The distance between a prototype v_i and ${\mathcal{D}^l}_j$ is measured as $d_{ij} = \min_{\omega \in {\mathcal{D}^l}_j} \{\|v_i - \boldsymbol{x}^\omega\|\}$. The labels for the prototype are defined as the ratio between the class distances:

$$l_{ij} = \frac{1}{n_c - 1} \left(1 - \frac{d_{ij}}{\sum_{i'=1}^{n_k} d_{i'j}} \right). \tag{5.35}$$

- Compute the labels $\hat{u}_{lj}, j = 1, \ldots, n_u$ for the unlabeled tuples $\omega \in \mathcal{D}^u$ by aggregating the prototype labels and the tuples' cluster memberships:

$$\hat{u}_{lj} = \min \left\{ 1, \sum_{k=1}^{n_k} l_{kl} u_{kj} \right\}. \tag{5.36}$$

If necessary, crisp labels are generated from $\hat{u}_{lj}$ by winner-takes-all defuzzification.

Bensaid et al. use this approach for transduction, i.e. they label only the unlabeled fraction of the example data. Application to new data was not intended, however, it is possible by calculating memberships U for new data from the given prototype positions.

As this approach allows multiple clusters for each class, it performs better on datasets that need this flexibility. Therefore, the authors call their approach the "successor" of their proposal in (Bensaid et al., 1996, see Section 5.2.3), although the underlying mechanisms are obviously rather different. Additionally, due to the initial overpartitioning, ssPPC cannot appropriately take the density information of the unlabeled data into account. The experiments on artificial datasets in Section 5.3 underline this problem.

5.2.6 Generalized Fuzzy Min-Max Classifier

In Section 3.1.4, we presented the *fuzzy min-max classifier* as the prototype of hyperbox oriented fuzzy rule learners (Simpson, 1992). Gabrys and Bargiela (2000) extended that scheme to semi-supervised learning.[7] The performed modifications to the original operations (*initialization* of new

[7] The extension to semi-supervised learning is only one aspect among several proposed modifications and enhancements. For instance, the authors suggest to use hyperboxes as input patterns. However, as we consider only input *points*, i.e. the "special case" of zero-volume hyperboxes, we leave these considerations apart. Furthermore, the membership function from Eq. (3.2) is modified to a "normal" trapezoid, and the average operator from Eq. (3.3) is replaced by the more common t-norm $\top_{\min}$. These changes affect fuzzy inference, but have no impact on the rule induction. The maximum hyperbox size

rules, *expansion*, *overlap test*, and *contraction*) are astonishingly straightforward:

- First, the definition of *compatibility* between hyperboxes and tuples is modified. If the label of a tuple is unknown, or if a hyperbox contains only unlabeled tuples and thus its consequent label is unknown, then there is no information that the tuple is *not* compatible to the hyperbox. Therefore, these cases are defined as compatible.
- The first labeled example that is added to a hyperbox defines the corresponding rule's consequent.
- Unlabeled hyperboxes are tested for overlap with any other hyperbox because they might assume any label. The expansion and contraction operations remain unchanged.

The algorithm is successfully applied a "toy dataset" and the Iris data. A similar semi-supervised extension of the original fuzzy min-max classifier scheme is presented by Kulkarni et al. (2002). The authors basically employ the same semi-supervised operations, but use hyperspheres instead of hyperboxes as prototype shapes.

5.3 Capabilities and Limitations

In the previous sections, we presented a variety of methods which learn from labeled and unlabeled data. They have in common that they either induce fuzzy rules or fuzzy cluster prototypes, or are general enough to be extended to fuzzy classifier learning. Before we further discuss their suitability for the semi-supervised learning of interpretable fuzzy rules on realistic problems, we demonstrate and compare their classification abilities on several illustrative example datasets. The results from our empirical evaluation and the discussion in Section 5.4 show the need for our specialized approach proposed in Chapter 6.

So far, there are no publicly available and commonly agreed on benchmark datasets for semi-supervised classifiers. The usual way of testing the methods uses the datasets of the UCI machine learning repository (Blake and Merz, 1998), and treats an arbitrarily chosen random subset as unlabeled. However, evenly distributed missing labels are neither the most

constraint Eq. (3.4) is modified to limit the maximum instead of the average width of a hyperbox. Finally, the dataset is processed several times, with adapted (decreasing) maximum size from run to run. Thus, more specific rules will be learned in later epochs.

Abbr.	Method	Section
GG	axis-parallel Gath&Geva fuzzy clustering	3.2.2
NEFCLASS	NEuro Fuzzy CLASSification	3.1.5
ssFCM	semi-supervised FCM	5.2.3
ssPPC	semi-supervised point-prototype clustering	5.2.5
ssGK	semi-supervised Gustafson&Kessel	5.2.3
ssMoG	semi-supervised mixtures of Gaussians	5.2.2
DBG	Davies-Bouldin / Gini index based approach	5.2.4
GFMM	generalized fuzzy min-max classifier	5.2.6

Table 5.1: Compared algorithms: GG is unsupervised, NEFCLASS is supervised, the other six methods are semi-supervised.

realistic case in practice, nor the case in which semi-supervised learning can be expected to yield its best results. Another problem is that most UCI datasets are rather small. Therefore, it is not uncommon that in publications the labels of 50 (or more) percent of the data are used for learning. From the practical point of view, the main motivation for semi-supervised learning algorithms is of course their capability to work in cases where $|\mathcal{D}^u| \gg |\mathcal{D}^l|$.[8] Unfortunately, for real-world problems with small numbers of possibly less representative labeled examples the underlying ground truth for the unlabeled data is often not available, and fair comparison of algorithms by e.g. the number of misclassifications thus is not possible.

Therefore, we compare the algorithms on two kinds of test datasets. First, we use several artificial datasets. In these examples, the distribution of the data as well as the choice of the labeled subset are designed to raise several difficulties for supervised (on the labeled subset) or unsupervised methods. These examples help to illustrate how the presented semi-supervised methods cope with these problems and which deficiencies they have. Additionally, we adopt the common test procedure of applying the algorithms to datasets from the UCI machine learning repository with randomly hidden labels.

Table 5.1 lists all methods considered in the following experiments. As a benchmark for the semi-supervised approaches, we additionally present the results of applying a pure supervised and a pure unsupervised algorithm to

[8] If someone has already labeled 50% of the data, he or she can probably as well label all of it. Or, from the learner's side: if 50% of the data are not sufficient to learn the model in a fully supervised manner, then the remaining 50% will probably not be of much help either (especially as they are unlabeled).

the datasets. As a supervised algorithm we use the well-known neuro-fuzzy classifier NEFCLASS (Nauck and Kruse, 1997, cf. Section 3.1.5). The unlabeled examples cannot be used for supervised learning and thus they are discarded when inducing a fuzzy classifier with NEFCLASS. As a benchmark for an unsupervised method we use the axis-parallel Gath&Geva (GG) fuzzy clustering algorithm (cf. Section 3.2.2). It is applied to the union of unlabeled examples plus labeled examples with ignored labels. The result of GG can in principle be transformed into a set of fuzzy rules, e.g. as described in Section 3.2.4.

The semi-supervised algorithms have been implemented according to the descriptions by their authors. One relevant modification refers to the implementation of Pedrycz's approach (ssGK), where we restricted the clusters to be axis-parallel, i.e. only the variances, but not the covariances are updated from the membership matrix. This helps to keep projection loss small when generating fuzzy rules (cf. Section 3.2.4). The parameters of the algorithms have been chosen according to their authors' recommendations. We tried to find robust sets of parameters. For each method, we tested the suitability of the parameters on a few test runs and kept set of parameters for all runs.

For the unsupervised GG, we assumed a one-to-one relation of clusters to classes, like in its semi-supervised relatives ssFCM and ssGK. As ssMoG allows to learn the labels of clusters, we generally set the number of clusters to two times the number of classes. For ssPPC, we follow the authors' suggestion to set the number of clusters to the number of labeled examples. For the labeling of the clusters, we implemented the second proposed scheme (see Eq. (5.34) in Section 5.2.5), because the authors reported the best results for it (Labzour et al., 1998).

5.3.1 Artificial Benchmark Datasets

In this section, we present three artificial datasets and the results of the algorithms described in this chapter. All three datasets have two input dimensions and two classes, marked by "○" and "△".

The first artificial dataset demonstrates how unsupervised learning fails due to the lack of guidance. The structure in the data does contain almost no information about the class distribution.[9] The second dataset shows a situation where unsupervised learning works even better than supervised learning, i.e. we could say that hiding the class information improves the

[9] In the terminology of Seeger (2001)—cf. Section 5.1—this distribution could be better modeled with a diagnostic than with a generative model.

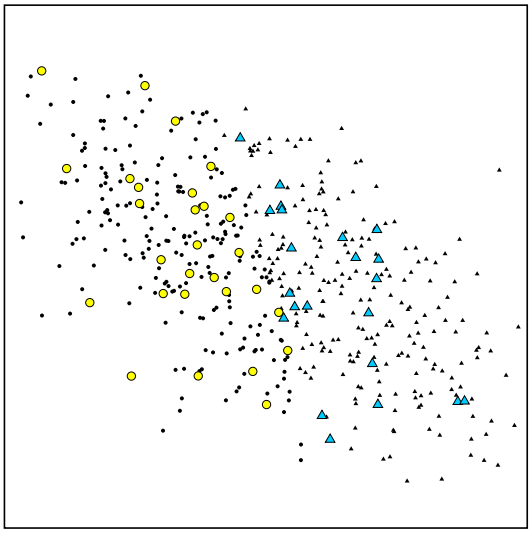


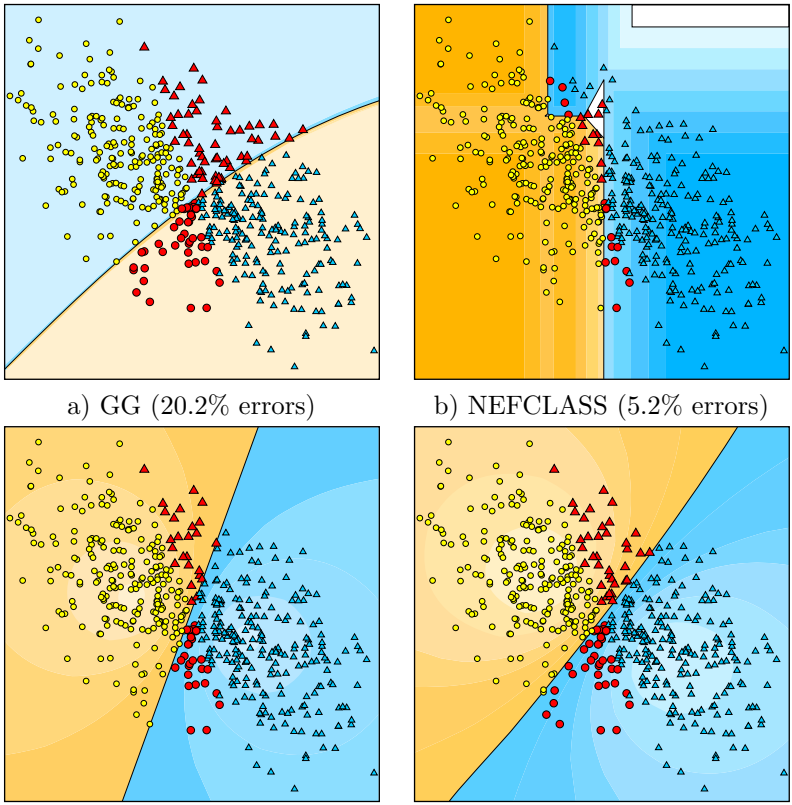
Figure 5.3: First artificial dataset: the distribution of the unlabeled examples hardly reflects the class structure.

results.[10] There are two well separated clusters that will be found by most unsupervised clustering algorithms. The problem of the given class labels is that they are not very representative for the classes, and thus mislead the supervised classifier. The third dataset increases the complexity of the problem by splitting one of the classes into two clusters. Therefore, at least two rules (i.e. prototypes) are needed to represent this class. This dataset will obviously be a challenge to those algorithms that allow to learn only one prototype per class. Additionally, the chosen labeled examples are not very representative and the clusters overlap.

Artificial Dataset I

Figure 5.3 shows 500 tuples from the first dataset that are used to train the classifiers. The two classes "○" and "△" are equally probable. The classes lie in two clusters side by side. 10% of the tuples have randomly been chosen

[10] This claim is, of course, a little ostentatious: We use two completely different classifiers and learn from different parts of the available data. Thus, we can not objectively measure the effect of hiding the class labels.



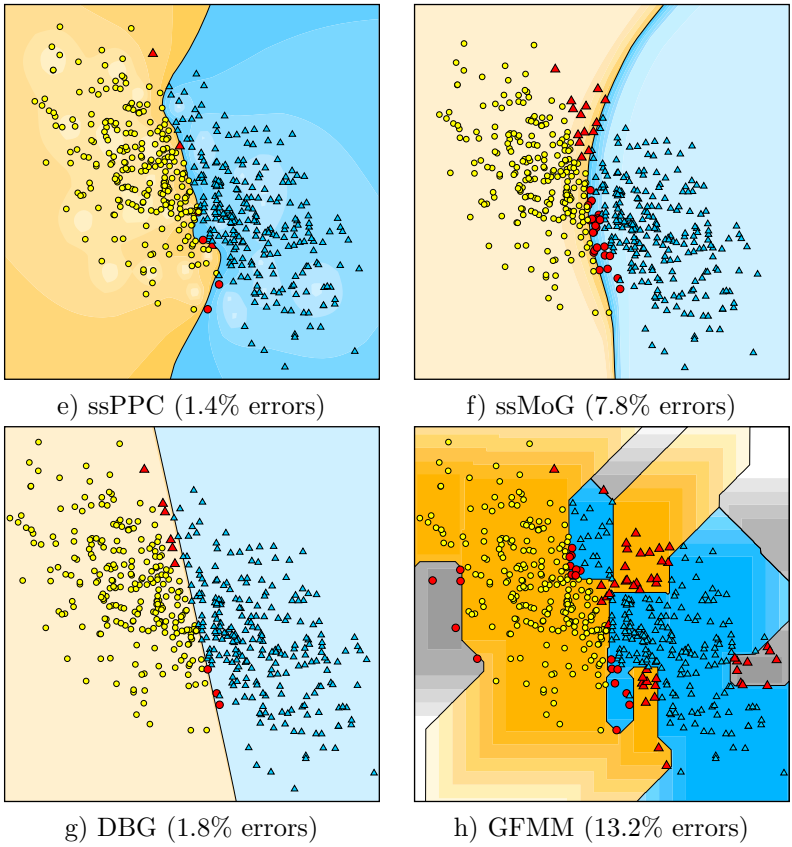
a) GG (20.2% errors) b) NEFCLASS (5.2% errors)

c) ssFCM (10.6% errors) d) ssGK (14.8% errors)

Figure 5.4: Experimental results on 1st artificial dataset.

to build the labeled dataset $\mathcal{D}^l$.[11] The corresponding points are drawn colored and slightly bigger. The class labels of the remaining 450 tuples have been discarded to form the unlabeled data $\mathcal{D}^u$ (however, the labels are shown by black symbols in Figure 5.3). The given labeled examples are sufficient to solve the classification task in a supervised manner from $\mathcal{D}^l$ only. However, as the clusters lie very close and even slightly overlap, they can hardly be found in the density distribution of the points. Hence, we can expect unsupervised learning to perform rather bad on this dataset.

[11] "○" occurs 28 times, "△" 22 times.



e) ssPPC (1.4% errors) f) ssMoG (7.8% errors)

g) DBG (1.8% errors) h) GFMM (13.2% errors)

Figure 5.4: Experimental results on 1st artificial dataset (continued).

The joint partially labeled dataset $\mathcal{D}^l \cup \mathcal{D}^u$ was used to induce the classifiers. The induced models have been applied to an independent test dataset of 500 tuples from the same distribution. The results are shown in Figure 5.4. Misclassified points are shown in red with the symbol that corresponds to their true class label. Additionally, the decision boundaries are shown. The background color of the diagrams relates to the activation of a rule or to the distance from the prototype center.

The unsupervised GG located two clusters on the principle axis of the point cloud (Figure 5.4a). However, the resulting decision boundary only hardly relates to the underlying true class boundary. Thus GG performs

rather bad on the test data. The supervised NEFCLASS is able to construct quite a good decision boundary from the labeled data $\mathcal{D}^l$ only (Figure 5.4b). Due to its global fuzzy set definitions and the min-max inference scheme, we can observe that the decision boundary is forced to be axis-parallel and that (shown in white) there is an ambiguous triangular region with a tie of two or more rules.[12] In Figures 5.4c and 5.4d, it can be seen that the influence of the additionally used labels allows ssFCM and ssGK to slightly rotate the decision boundary, and thus the number of misclassifications is reduced compared to GG. It should be noted that further increasing the weight of the labeled examples by changing the parameters does not help to rotate the boundary any further because the cluster centers are already close to the centers of the labeled examples, i.e. the center of class "○" is above the center of class "△".

The result of ssPPC, shown in Figure 5.4e, is one of the best on this first dataset. The boundary is very flexible, due to the high number of cluster prototypes (i.e. $n_k = |\mathcal{D}^l| = 50$). The idea of the ssMoG approach is close to that of the semi-supervised clustering approaches. However, it performs better because the probability of misclassifications is explicitly taken into account (Figure 5.4f). Because the Gini index also explicitly measures impurity, DBG also successfully separates the points from $\mathcal{D}^l$ and thus also generates a good decision boundary for this test dataset (Figure 5.4g). The measure induces a model with two prototypes only. GFMM produces the most jagged decision boundary of the compared methods (Figure 5.4h). The two yellow, class "○" hyperboxes that lie completely in the blue, class "△" area significantly degrade the performance. Such hyperboxes can result from the semi-supervised learning algorithm. It builds hyperboxes from the unlabeled data and then labels them from the first labeled tuple. If this seed tuple is an outlier, large hyperboxes with wrong labels can be created. And, even worse: once labeled incorrectly the hyperbox will shrink if it contains contradicting tuples, and grow again if unlabeled tuples lie next to it. However, it can never disappear. Actually, the two hyperboxes in Figure 5.4h that degrade performance do not contain a single example of class "△". Additionally, if no labeled tuple falls into a hyperbox, it remains unlabeled (as for example the shaded gray areas in Figure 5.4h).

[12] Notice, however, that this might partially be overcome by the pruning strategies of the more recent NEFCLASS versions.

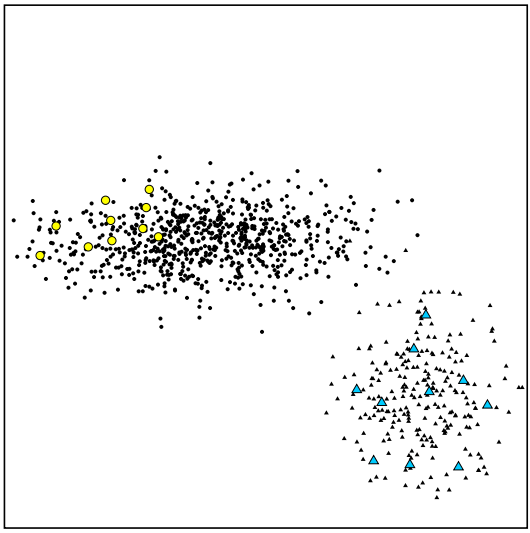


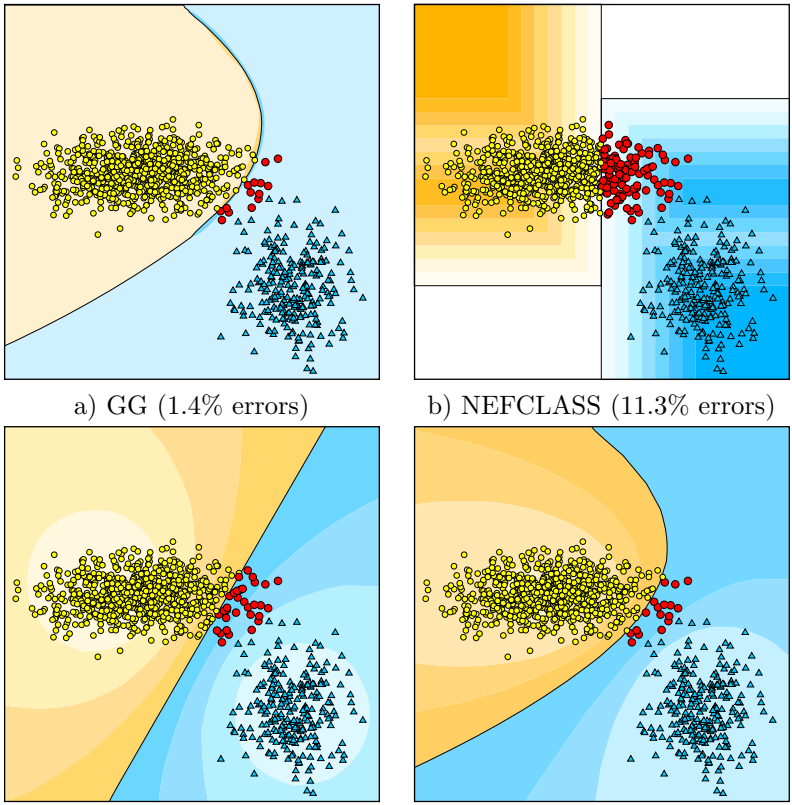
Figure 5.5: Second artificial dataset: untypical labeled examples.

Artificial Dataset II

The second dataset, depicted in Figure 5.5, demonstrates a situation where the unlabeled data contains more information on the optimal decision boundaries than the labeled fraction of the data. As unsupervised learning can exploit $\mathcal{D}^u$ and $\mathcal{D}^l$ (with discarded labels), but pure supervised methods can only make use of $\mathcal{D}^l$, we have the rather uncommon situation that unsupervised learning works even better than supervised learning.

The training dataset has 1000 points, 750 of class "○" and 250 of class "△". The classes build two well separated clusters that should easily be identified by most unsupervised clustering algorithms. 2% of the labels are assumed to be known, i.e. 10 examples of each class. The problem of the given class labels is that they do not characterize the classes very well. A supervised classifier which can only learn from $\mathcal{D}^l$ thus is easily misled.

Figure 5.6 shows the results of applying the induced models to an independent test dataset. As can be seen in Figure 5.6a, the unsupervised GG has no problems in identifying the cluster structure and performs very well on this dataset. In contrast to this, the performance of NEFCLASS is significantly worse (Figure 5.6b). Using the few labeled points only, the

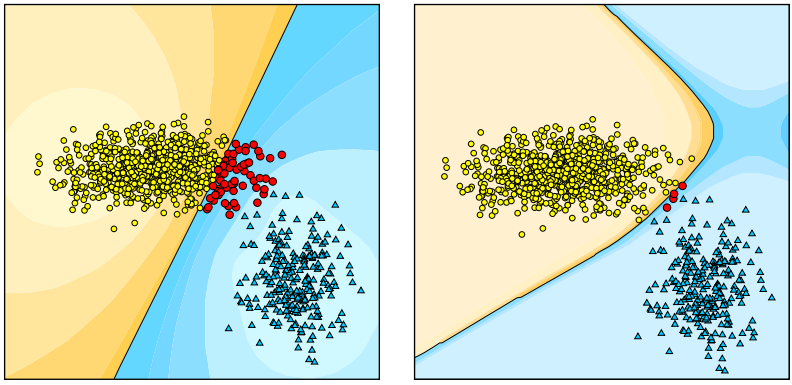


a) GG (1.4% errors) b) NEFCLASS (11.3% errors)

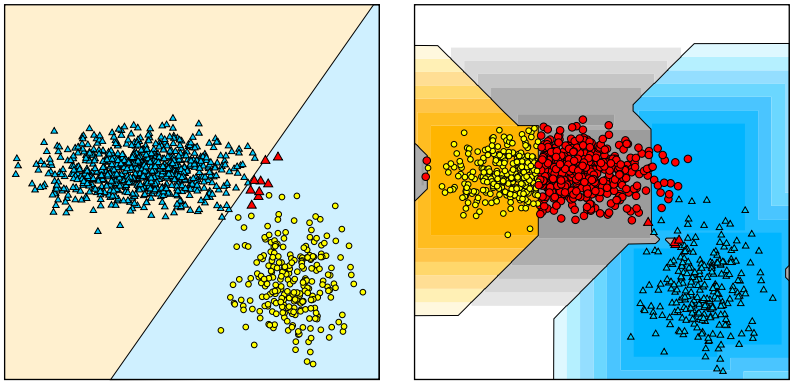
c) ssFCM (3.1% errors) d) ssGK (1.7% errors)

Figure 5.6: Experimental results on 2nd artificial dataset.

best a supervised classifier can do is to split the data in the middle between the observed classes. Actually, NEFCLASS creates four fuzzy rules for the labeled examples and keeps these initial rules without any fine-tuning, as they already perform perfect on $\mathcal{D}^l$. However, this leads to many misclassified points on the whole dataset because the decision boundary should not be set to the middle between the shown labeled examples, but rather at the line of the lowest data density. This second example dataset might seem to be "unfair" for the supervised NEFCLASS in the sense that no supervised algorithm would have a chance to build a proper separating line. However, in practice such situations might be rather common if the data is labeled by



e) ssPPC (6.2% errors) f) ssMoG (0.4% errors)



g) DBG (0.9% errors) h) GFMM (46.7% errors)

Figure 5.6: Experimental results on 2nd artificial dataset (continued).

hand. Manually given labels are often defined for those objects that differ most significantly from the other classes, and not for those that are typical for their own class.

The partially supervised algorithms perform rather well on this dataset. The result of the semi-supervised ssGK (Figure 5.6d) is almost identical to that of the unsupervised GG. The result of ssFCM is slightly worse (Figure 5.6c). This can partially be explained by the more rigid cluster shape of the underlying FCM. However, it can be observed that the center of the left (yellow) cluster is not located at the center of the points of class "○". Instead, it is rather strongly attracted by the labeled examples of "○", an

effect of setting the membership of labeled points to corresponding clusters to one. Actually, the original fully unsupervised FCM would perform better than ssFCM on this example because it yields better estimates of the cluster centers. The objective function of ssFCM can be seen as a special case of the one ssGK uses. Accordingly, the performance of ssGK degrades, when the influence of the labeled points is increased. Hence, both algorithms depend on "good-natured" labeled points.

When we compare the results of ssPPC in Figure 5.4e and Figure 5.6e, we might be irritated by the simplicity of the decision boundary in the latter, where it is almost a straight line. Actually, to 8 out of $n_k = |\mathcal{D}^l| = 20$ clusters Eq. (5.34) assigns labels with $\forall j \in \mathcal{C} : l_{ij} = 0$. This means that most clusters that are positioned at unlabeled points are, in fact, completely ignored. The remaining clusters lie close to the labeled points. The resulting decision boundary is close to, e.g., that of naïve Bayes classification from $\mathcal{D}^l$ only. Obviously, ssPPC depends on labeled examples that are well distributed in $\mathcal{D}^u$. However, it is less important that their center coincides with the center of the class.

The class boundary which results from ssMoG is almost perfect (Figure 5.6f). The better performance in comparison to GG can mostly be attributed to the higher flexibility due to four mixtures instead of two clusters, and the ability of ssMoG to adapt a priori probabilities (i.e. cluster sizes). As on the first dataset, DBG again yields one of the best results (Figure 5.6g). As practically any reasonable set of clusters allows to separate the two classes for the 20 labeled examples only, the impurity measure does not contribute to the specific solution.[13] Thus, the (almost perfect) location of the decision boundary is determined mostly by the Davies-Bouldin index.

The weakest result on this dataset is generated by GFMM. As can be seen in Figure 5.6h, GFMM failed to label the hyperboxes in between the labeled points. Thus 45.2% of the data are not classified at all. As a matter of fact, the original FMM applied to the labeled data only would have performed better on this dataset.

Artificial Dataset III

The third artificial dataset, shown in Figure 5.7, combines the challenges of the first two datasets, i.e. overlapping, hard-to-separate clusters and moderately representative labeled examples. Additionally, the complexity is

[13] Notice that impurity, i.e. the Gini-index, is calculated from the labeled examples only. All faultless solutions yield the same minimal impurity and thus the same contribution to the fitness function independently of the chosen weight in Eq. (5.33).

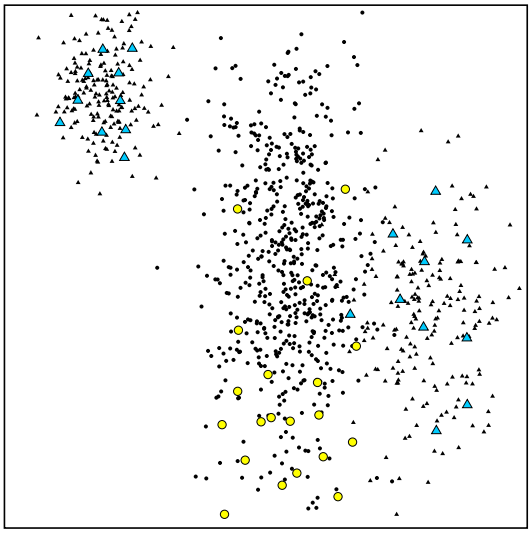


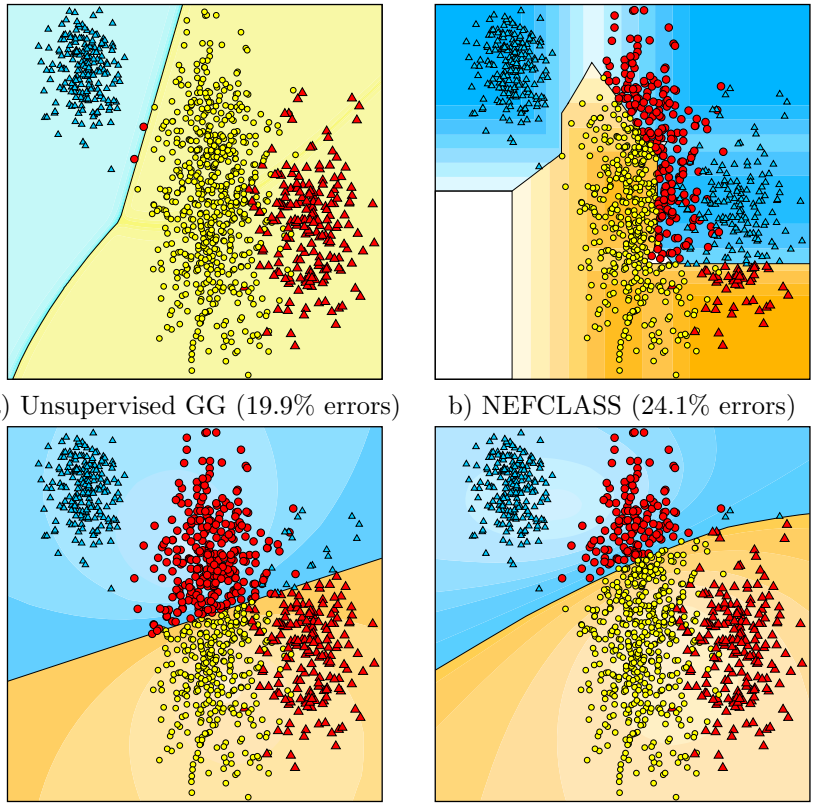
Figure 5.7: Third artificial dataset: untypical examples, overlapping clusters, and two clusters for one class.

increased. Class "△" is split into two clusters and thus requires at least two rules to be adequately described. It should be noted that this example will mostly illustrate a principle problem of ssFCM and ssGK, namely the inability to model classes with more than one cluster. However, the 40 given labeled examples allow relatively good results with pure supervised methods.

Although generally, we assumed a one-to-one relationship of clusters to classes for the unsupervised GG, the results presented in Figure 5.8a are generated with three clusters to help to identify the true clusters.[14] However, even with three clusters, GG fails to find the underlying structure in the data. Obviously, the overlapping clusters are hard to find without the guidance of labeled examples. The supervised NEFCLASS has problems to find appropriate rules in areas without presented (labeled) examples (Figure 5.8b).

The partial supervision of ssFCM and ssGK could potentially help to

[14] With two clusters, the results look rather similar to Figure 5.8d and the error raises to about 30%.



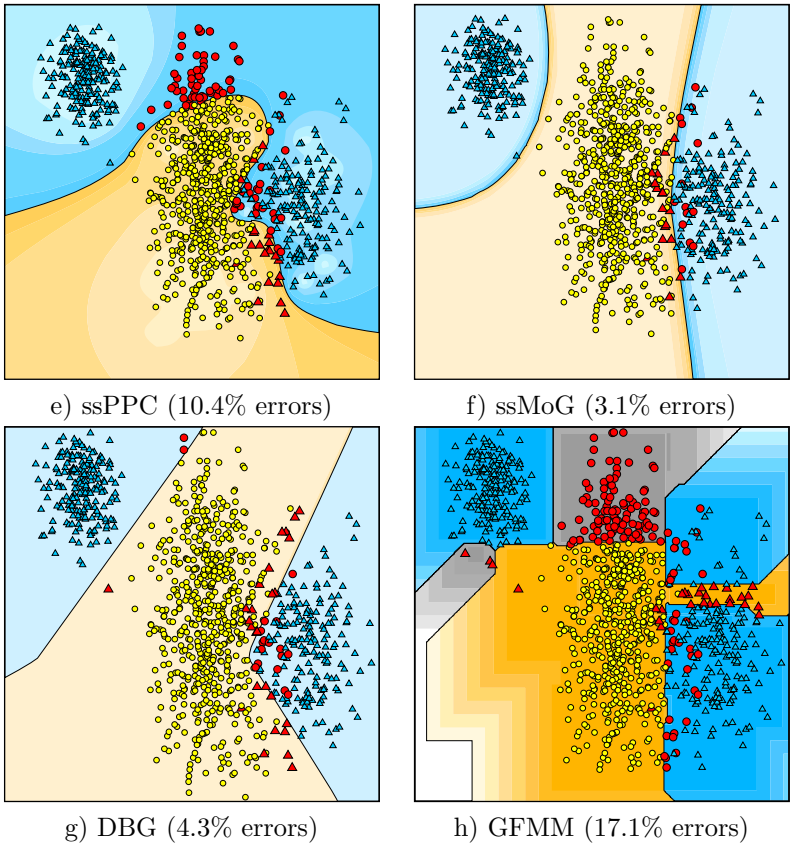
a) Unsupervised GG (19.9% errors) b) NEFCLASS (24.1% errors)

c) ssFCM (40.0% errors) d) ssGK (32.9% errors)

Figure 5.8: Experimental results on 3rd artificial dataset.

separate the two overlapping clusters. However, both approaches do not allow to induce more than one cluster per class. As can be seen in Figures 5.8c and 5.8d, this significantly deteriorates the performances of these semi-supervised algorithms on this dataset. Notice that in both approaches the blue cluster is attracted to a certain degree by the labeled examples of the lower right cluster.

In comparison to the second dataset, the labeled examples are spread better within the unlabeled examples, and the effect of "empty" clusters in ssPPC is less apparent. The result shown in Figure 5.8e is rather good. However, it shows a tendency of ssPPC to overfitting due to the highly



e) ssPPC (10.4% errors) f) ssMoG (3.1% errors)

g) DBG (4.3% errors) h) GFMM (17.1% errors)

Figure 5.8: Experimental results on 3rd artificial dataset (continued).

flexible decision boundary. Obviously, this boundary is pretty close to that resulting from nearest neighbor classification from $\mathcal{D}^l$ alone. This is not surprising, as ssPPC does not really seek for structure, but effectively uses the distance from the labeled examples.

Figure 5.8f depicts the results of ssMoG, which nicely approximates the underlying distribution of the example dataset. Again, this approach yields the best result of the compared methods. The result of DBG is slightly worse (Figure 5.8g). However, considering the more restricted cluster shape, the result is very good. The last compared semi-supervised method, GFMM, performed better than on the second dataset. It still generates hyperboxes

from unlabeled examples that remain unlabeled and thus unavoidably produces errors. The error for these rejected tuples could be reduced simply by *guessing* their class label (or that of the containing hyperbox). However, this does obviously not help to reveal the structure of the dataset.

5.3.2 Empirical Results on the Iris Dataset

Additionally to the artificial examples, we compared the performance of the algorithms on Fisher's well-known "real-world" Iris dataset, which can be found in the UCI machine learning repository (Blake and Merz, 1998). The dataset contains three classes of iris plant that are described by four numeric attributes, namely petal and sepal lengths and widths. The dataset has 150 examples. In the experiments, different random fractions of the labels have been hidden to the learning algorithms. The fractions have been chosen to be 4%, 8%, 20%, 40% and 100%. For the experiments the following steps are performed:

for $run = 1 \ldots 20$ **do**
  Randomly split the data into training and test set (75 cases each).
  for all percentages $p \in \{4, 8, 20, 40, 100\}$ **do**
    Randomly choose p% labeled examples of the training dataset as $\mathcal{D}^l$, discard the labels of the remaining tuples to form $\mathcal{D}^u$.
    for all classifier approaches CA **do**
      Induce a classifier with CA (for the semi-supervised approaches, use $\mathcal{D}^l \cup \mathcal{D}^u$; for NEFCLASS, use $\mathcal{D}^l$ only; for GG use the complete training data with discarded labels).
      Apply the classifier to the test dataset.
    end for
  end for
end for

Figure 5.9 summarizes the results. The abscissa shows the percentage of labeled examples, the ordinate shows the number of misclassifications. The black lines mark the mean error averaged over the twenty runs. The darker corridors mark 1st and 3rd quartiles, the brighter corridors are spanned by the minimal and maximal errors observed during the 20 runs.

The two extremes—pure unsupervised and pure supervised learning—mark the poles in between which most semi-supervised results lie. The unsupervised GG has a fair performance. As it does not depend on the chosen labeled subset, its performance is constant over the size of $\mathcal{D}^l$ (Figure 5.9a). The fully supervised NEFCLASS clearly outperforms GG when it

gets enough—10% or more—labeled examples. However, with less examples performance quickly degrades (Figure 5.9b).

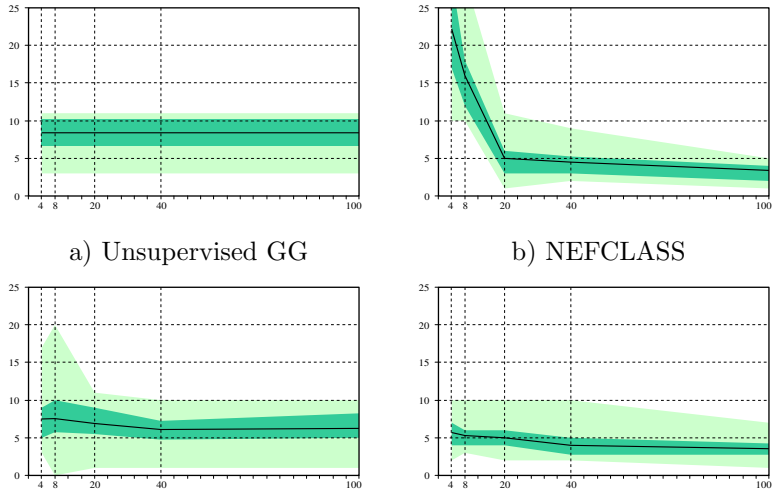
Almost all considered semi-supervised approaches succeed to induce classifiers which are better than NEFCLASS for few labeled examples, and better than GG for greater fractions of available labels. The only clear exception is GFMM, which performs mediocre for 100% known labels, but unacceptably bad in the presence of many unknown labels (Figure 5.9h). The other approaches are rather similar and generally suited to point out the advantage of semi-supervised learning.

Although most of the approaches have a better average error than GG, the spread from minimal to maximal error is quite large for some of them. The narrowest corridor can be observed for ssGK, which shows the best overall performance on this dataset (Figure 5.9d). DBG and ssMoG, which both did very well on the artificial datasets, perform slightly worse than ssGK. Especially DBG seems to have problems with 20% or less labeled examples (Figure 5.9g). As argued above, finding solutions with no training errors (and thus minimal impurity) is easily possible for the Iris data and small labeled subsets $\mathcal{D}^l$. Hence, the cluster positions will mainly be determined by the Davies-Bouldin index, which seems to be less adequate for this dataset.

5.4 Discussion

This chapter gave a survey of existing approaches to semi-supervised learning. As the goal of this thesis is to develop an algorithm for the semi-supervised induction of fuzzy classification rules, the survey was focused on methods that induce—or can be modified to induce—fuzzy classifiers. To our knowledge, in spite of a number of different proposals, GFMM seems to be the only semi-supervised method that explicitly generates a set of fuzzy rules. The overall performance of GFMM, however, was rather disappointing. The other methods have been designed to extract non-rule models like fuzzy (point- or non point-prototype) clusters or mixtures of Gaussians. Some of these algorithms showed rather good performances, and all of their induced models can in principle be transformed into fuzzy rules. The ultimate objective of fuzzy rule-based classification is the learning of readable and interpretable rule bases. In the following, we therefore discuss the suitability of the models for this goal. The following aspects are considered:

- **Expressiveness:** How flexible is the induced model? What kind of distribution and decision boundaries can be modeled? How capable



a) Unsupervised GG b) NEFCLASS

c) ssFCM d) ssGK

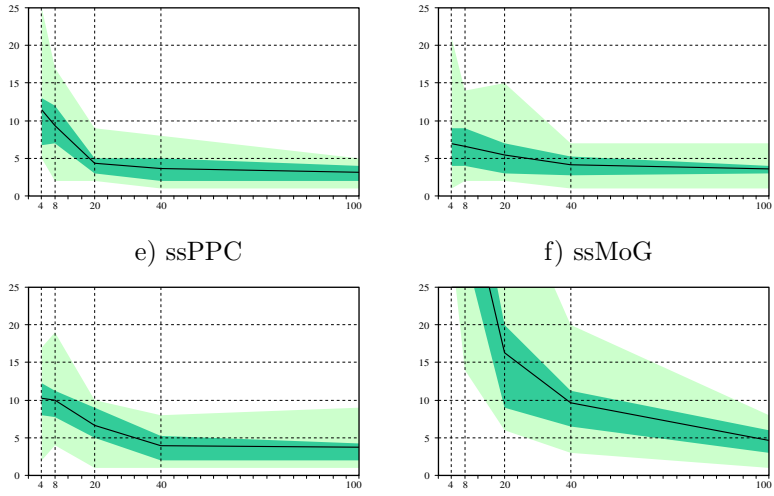
Figure 5.9: Experimental results on Iris dataset. The graphs show the error vs. the percentage of labeled examples; black line: mean error, dark corridor: 1st/3rd quartile, bright corridor: minimal/maximal error.

are single rules/clusters? How flexible is the model induced by the interaction of rules/clusters?

- **Interpretability:** How difficult is it to transform the model into a fuzzy rule base? How readable will this rule base be? How many rules does it have? Will it let the user gain insight into the data?

- **Semi-supervised learning:** How capable was the semi-supervised learning algorithm? Does it depend on the representativeness of the examples? How strongly does it respect the information of the labeled data $\mathcal{D}^l$?

The results which are discussed in the next sections are summarized in Table 5.2.



e) ssPPC f) ssMoG

g) DBG h) GFMM

Figure 5.9: Experimental results on Iris dataset (continued).

5.4.1 Expressiveness of Induced Model

As mentioned in Chapter 2, fuzzy classifiers can approximate arbitrary class boundaries with arbitrary precision. The proofs require that either the number of fuzzy rules must be unrestricted or that the membership functions must be flexible enough. In practice both are limited—and thus also the approximation capabilities. However, as there are significant differences between the models, we compare the flexibility of individual clusters (or membership functions), and the flexibility of the class boundaries, i.e. of the results of the interacting rules.

Three of the models, namely ssFCM, ssPPC and DBG, consider only point-prototypes and isotropic distances. They are not able to model shape or size of clusters. The remaining models allow more flexible cluster shapes, either as fuzzy hyperellipsoids (ssGK and ssMoG) or fuzzy hyperboxes (GFMM). However, even point-prototypes can yield very complex cluster shapes, as we observed for ssPPC; the aggregation of a large number of simple clusters allows this method to generate very flexible decision boundaries. GFMM is also able to create a large number of rules and thus to represent rather arbitrary boundaries. DBG and ssMoG normally use a

	ssFCM	ssGK	ssPPC	ssMoG	DBG	GFMM
Expressiveness (5.4.1):						
Cluster shapes	○	+	○	+	○	+
Complex clusters	−	○	++	++	+	++
Interpretability (5.4.2):						
Convert to fuzzy sets	+	○	+	++	+	++
Linguistic interpretability	+	+	−	○	+	−
Interpretab. of rule base	+	+	−−	−	+	−
Semi-supervised learning (5.4.3):						
Overall performance	○	+	+	++	++	−−
Untypical examples	○	+	−	++	++	−−
Respecting errors on $\mathcal{D}^l$	−	−	+	+	++	(++)

++: very good +: good ○: neutral −: bad −−: very bad

Table 5.2: Comparison of the considered approaches

smaller number of clusters (or Gaussians, respectively). However, as the maximum number of clusters to be used is specified by the user, it can be adapted to the required expressiveness. Since the cluster shape of DBG is more restricted than that of ssMoG or GFMM, and since its number of clusters is generally smaller and their aggregation is crisp,[15] the overall flexibility of DBG is slightly smaller than that of ssPPC, ssMoG or GFMM.

The strict cluster-class correspondence of ssFCM and ssGK wastes much of the flexibility of fuzzy classifiers. While ssGK can partially compensate for that weakness by its more flexible cluster shape, ssFCM can only represent Euclidean Voronoi cells. This means that for any problem to be perfectly modeled with ssFCM, each pair of classes has to be linear separable, what is obviously a strong restriction.

5.4.2 Interpretability of Transformed Rule Base

One common motivation for using fuzzy methods is the demand for explicable data analysis results that can be understood by human experts and, for example, be checked for plausibility. However, the models generated by fuzzy methods somewhat differ in their interpretability.

As we want classification knowledge represented in form of fuzzy rules, for all models except GFMM the first problem is to transform clusters into

[15] Both, ssPPC and ssMoG, use soft labels of the clusters, and thus aggregation of rules is weighted.

fuzzy membership functions and rules. In GFMM each hyperbox corresponds to a fuzzy rule with trapezoidal membership function and $\top_{\min}$-conjunction. For the cluster approaches, the projection method described in Section 3.2.4 can be used. As we restricted the non point-prototype methods ssGK and ssMoG to axis-parallel clusters, the projection errors should be reasonably small. However, instead of projecting the clusters in combination with $\top_{\min}$-conjunction, the mixtures found by ssMoG can be directly transformed into fuzzy rules with prod-sum inference (analogous to the transformation of naïve Bayes classifiers discussed in Section 2.4). In this case, the Gaussian distributions in the input dimensions can be interpreted as membership functions.

As we will show in Chapter 6, models with isotropic distances can be transformed into products of Gaussians with (arbitrary but) constant variances. Thus the clusters of ssFCM, ssPPC, and DBG can also be directly transformed into fuzzy rules with $\top_{\text{prod}}$-conjunction, without the need of projecting them. However, the constant width of the resulting membership functions does not reflect different cluster sizes or shapes.

All compared approaches yield local (i.e. rule-wise) definitions of membership functions. In general, it is much easier to find linguistic descriptions for globally defined membership functions shared by all rules. However, for ssFCM and ssGK, $n_k = n_c$ membership functions are created for each input dimension, i.e. one for each class. If the number of classes is small, generating linguistic description—like "small", "medium", "large"—will be possible in most cases. DBG and ssMoG also usually induce small numbers of clusters (and thus membership functions). Especially DBG autonomously reduces the number of clusters. Additionally, as all resulting membership functions share the same constant width, there are no situations where one fuzzy sets is included in another. Such situations are much harder to be described linguistically than just relative orderings. Although ssPPC also uses fuzzy sets with constant widths, due to the extreme overpartitioning with $n_k = |\mathcal{D}^l|$ clusters it generates far too many membership functions to be described linguistically. The same holds for GFMM, where additionally inclusions of fuzzy sets frequently occur.

The interpretability of the data analysis result—i.e. the interpretability of the rule base as a whole—depends to a great extent on the interpretability of single rules and on their number. Hence, the high number of rules in ssPPC and GFMM massively degrades interpretability and readability of the rule base. With respect to this, approaches like ssFCM, ssGK, ssMoG, or DBG have clear advantages. The interpretability of ssMoG is, however, massively spoiled by the posteriors $P(c \mid m)$, which have to be

translated into fuzzy consequents. This means that one rule does in general not describe a single class, but a mixture of classes. The decision boundaries that result from aggregation of such rules become rather unpredictable. The same problem arises for the soft labels that ssPPC calculates for its clusters.

Obviously, none of the approaches was intended to generate membership functions that are suited for linguistic description. Another common, yet very potent means to achieve interpretable, readable rule bases is the use of "don't cares", i.e. specifying only a subset of the dimensions in the antecedents. This helps to make rules more general on the one hand and easier graspable for humans on the other hand. However, this possibility is not used in any of the presented approaches.

5.4.3 Semi-Supervised Learning Capabilities

An aspect that is even more relevant than expressiveness and interpretability in the context of this thesis is the ability of the methods to learn from partially labeled data. We pointed out that semi-supervised learning can be expected to be most effective in situations where many unlabeled, but only few labeled, possibly untypical examples are available for inducing a classifier. Thus one question is whether an algorithm is able to cope with such situations. The unsupervised component of semi-supervised learning (as well as any pure unsupervised learning) relies on assumptions about the nature of the data (cf. Section 5.1). Consequently, another important question is how a semi-supervised learning algorithm will behave if these assumptions are false. Although a performance decrease can hardly be avoided in such situations, it is desirable that the result of learning from $\mathcal{D}^l \cup \mathcal{D}^u$ is not (or at least not much) worse than pure supervised learning from $\mathcal{D}^l$ alone.

The overall performance of the algorithms can hardly be judged, as it obviously strongly depends on the data at hand which algorithm will be best. In our case, it depends on the weighting of the considered example datasets. However, we can say that ssMoG and DBG yielded rather good results on each example. On the Iris dataset, ssGK was in front. The only clear exception was GFMM, which regularly performed rather bad.

The second artificial dataset illustrates a scenario of learning from examples which might be characteristic for their class, however not prototypical, i.e. which do not lie near the center of their class. On this dataset ssMoG and DBG performed best,[16] followed by ssGK, ssFCM, and ssPPC (in that

[16] The slight performance difference between ssMoG and DBG on the second artificial

order). GFMM again lies clearly behind. The sequence of the other approaches is not surprising, when we look at the underlying methods. DBG crisply assigns the examples to the clusters. The labels are only taken into consideration in the impurity measure. Once the labeled examples have been assigned to their correct corresponding clusters, impurity is constant and the exact position of the examples does not influence the position of the clusters. The mechanism used in ssMoG can be seen as a similar, but softened version of DBG. In contrast to these algorithms, in ssFCM the labeled examples always directly influence the prototype estimation, because their membership is fixed at one. As the labeled tuples are often weighted to compensate for the smaller size of $\mathcal{D}^l$, untypical examples can significantly worsen the result. The same idea is basically used in ssGK. As, however, a factor is introduced in ssGK to balance between pure unsupervised and ssFCM-like semi-supervised learning, the effect is less marked. Lastly, the decision boundary of ssPPC is mostly determined by the distances from the labeled examples. Hence, it is obvious that untypical examples affect the result rather severely.

In Table 5.2, the ability of a semi-supervised algorithm to perform at least as good as pure supervised learning on the labeled examples only is named the ability to "respect the errors on $\mathcal{D}^l$". Although this means not necessarily the same, we wanted to express whether an approach is able to extract the information contained in $\mathcal{D}^l$. The difference between the two abilities is most obvious for GFMM. It is probably the best of the presented algorithm to avoid errors on $\mathcal{D}^l$ (hence the "++" in Table 5.2), because it does not stop splitting the hyperboxes before the error on $\mathcal{D}^l$ is zero. However, looking at the result on the first artificial dataset shows that it can create (almost absurd) rules with a generalizing performance much worse than pure supervised learning.

The reason that DBG performs extremely well on the considered examples lies in the combination of impurity and dispersion measure. The Gini index as impurity measure, which is usually weighted higher than dispersion, explicitly respects the errors on $\mathcal{D}^l$. As long as the clusters are consistent with $\mathcal{D}^l$, DBG tries to fit their position to the unlabeled data $\mathcal{D}^u$ by using the Davies-Bouldin dispersion measure. In ssMoG, the error for the labeled examples is minimized by maximizing the probability that the labels were generated by the modeled distribution—a direct consequence of the maximum likelihood principle used. As long as these probabilities are high, the distribution is adapted to the densities of the unlabeled points

dataset is mostly due to the different cluster shapes.

(which is why the closeness of the labeled points to the cluster centers is of secondary importance).

The mechanisms in DBG or ssMoG to exploit the labeled data are quite different than those for ssFCM or ssGK. Intuitively, in ssFCM and ssGK any labeled point tries to influence its corresponding cluster, such that it belongs to this cluster with a higher membership degree. Although this seems reasonable to reduce the error for a single example, it can lead to higher error rates for other involved tuples, especially for ssFCM where only the centers are updated. The effect of this can be seen in the first artificial dataset. To model the linear decision boundary, the center of the first class ("△") would have to be right of and slightly above the center of the second class ("○"). However, the labeled examples—and thus the cluster centers—of the first class lie below that of the second class, and therefore the decision boundary becomes tilted to the wrong side. Similar effects can be observed on the third dataset.

The last considered algorithm, ssPPC, performs almost perfect on the first dataset. Actually, ssPPC performs more like a supervised than like a semi-supervised algorithm: the extreme overpartitioning is closer to subsampling than to revealing underlying cluster structure; as the soft cluster labels are based on the isotropic distances of $\mathcal{D}^l$ only, the structural information is hardly exploited. The clusters from the overpartitioning define a set of Voronoi cells. As they are learned unsupervised, there is no guarantee that the cell borders coincide with the class borders.[17] Thus ssPPC does not guarantee small error rates on $\mathcal{D}^l$, especially if n_k is small.

5.5 Conclusions

This chapter dealt with the current research area of semi-supervised learning. We gave theoretical considerations, why it is possible that examples without class labels can support a learning algorithm in finding better decision boundaries, and which characteristics the underlying data-generating process must possess. The artificial datasets give an idea of the characteristics that allow successful learning from partially labeled data. The best improvements can be expected, if the labeled examples are too few and too unrepresentative for pure supervised learning, and if the density distribution of the (unlabeled) data corresponds with the classes (i.e. if the *sampling paradigm* is appropriate for the data).

[17] Notice, however, that the resulting decision boundary is generally smooth due to the soft labels.

We presented a number of existing approaches to semi-supervised learning, with a stress on methods related to fuzzy set theory. On our examples, the methods were able to outperform pure supervised or unsupervised learning in most cases. However, in spite of the diversity of approaches, we argued in Section 5.4 that none of these algorithms is fully appropriate for the semi-supervised induction of readable and interpretable rule based fuzzy classifiers. In the next chapter, we formulate the requirements for a semi-supervised learning algorithm for fuzzy classification rules, and present our approach to this task.

Chapter 6

Evolutionary Semi-Supervised Fuzzy Classification

The preceding chapter gave a survey and discussion of existing algorithms for semi-supervised classifier learning, with a focus on fuzzy methods. However, as we pointed out in Section 5.4, none of these approaches is fully suited for the semi-supervised induction of interpretable fuzzy classification rules. In Section 6.1, we list requirements that should be met by a semi-supervised fuzzy rule learner. From these requirements we derive characteristics that an improved model should possess, taking into account the findings from

- Chapter 2 about properties of fuzzy classification rules, from
- Chapter 5 about potential approaches to semi-supervised learning, and from
- Chapters 3 and 4 about appropriate rule learning methods.

We come to the conclusion that the requirements are matched best with an evolutionary algorithm for rule learning. The details of our rule learning framework are described in Section 6.2.

One appealing feature of evolutionary algorithms is their flexibility regarding the optimized fitness function. In Sections 6.3 and 6.4, we propose two alternative fitness functions that assess the quality of a fuzzy rule base with respect to a set of partially labeled data. Although the approaches

are rather different in their details, they share similar basic ideas how to incorporate labeled and unlabeled examples into the fitness function. The first approach is an extension of the algorithm by Demiriz et al. (cf. Section 5.2.4). The second approach is based on the *Minimum Description Length* (MDL) principle. We evaluate the performance of the approaches in Section 6.5 and compare it to those of the algorithms presented in Chapter 5.

6.1 Requirements and Preferences

Ultimately, we want to develop an algorithm that allows to induce interpretable fuzzy classification rules in a semi-supervised manner. Therefore, as in Section 5.4, we consider three aspects: expressiveness, interpretability, and semi-supervised learning capabilities. The underlying fuzzy models should be flexible enough to cope with the problems discussed in the previous chapter. The semi-supervised learning method should be able to induce a model from partially labeled datasets with various characteristics. The resulting fuzzy rule bases should remain interpretable, i.e. they should let a human expert gain insights into the analyzed data.

In Chapter 5, we pointed out that semi-supervised learning can have significant advantages over pure supervised learning, when few labeled examples $\mathcal{D}^l$ are available and when the abundantly available unlabeled data $\mathcal{D}^u$ have a certain inherent cluster structure. Especially, if the labeled examples are unrepresentative, supervised learning from $\mathcal{D}^l$ alone yields unsatisfying results. If an algorithm is able to cope with such cases—like, for example ssMoG or DBG—, the benefits of semi-supervised learning become most evident.

On the other hand, we can generally not rule out the possibility that there is no—or, even worse, misleading—inherent structure in the example data. Ideally, a semi-supervised algorithm should in no case perform worse than supervised learning from $\mathcal{D}^l$ alone. Although this cannot be guaranteed in general, it seems to be reasonable to reduce the errors on the given labeled examples $\mathcal{D}^l$, and use $\mathcal{D}^u$ only to an extent that is consistent with $\mathcal{D}^l$. In DBG, this has been realized by explicitly modeling and combining measures of impurity and dispersion. We adopt this procedure in our approaches, as it allows fine and intuitive control of the use of $\mathcal{D}^l$ and $\mathcal{D}^u$. Especially the dispersion measure must obviously be suited for the underlying fuzzy model.

In Section 2.3, we discussed the characteristics of fuzzy rule bases that help to maintain interpretability without losing too much expressiveness. Some of the characteristics to improve interpretability might at the same

time even improve the performance of the rule base. Rule bases with small numbers of rules, for example, are generally more readable, and often generalize better on unseen data. Similarly, the use of "don't cares" eliminates attributes from the antecedents, which makes rules much easier to read, and often improves generalization ability. An algorithm should be able to adapt the size of the rule base and the use of "don't cares" to the complexity of the learning task.

Fuzzy consequents—corresponding to soft cluster labels or component posteriors—can be a means of more flexibility for the decision boundary (cf., for example, ssPPC or ssMoG). However, they can easily destroy interpretability. Similarly, rule weights should be used with deliberation. However, if they are used with the proper semantic of normalization factors as shown in Section 2.4, they can help to compensate effects of different numbers of attributes in the antecedents.

For effective semi-supervision it is necessary that the rules can adequately capture the cluster structure. The model should allow smooth "natural" cluster shapes. For interpretability it is also advisable that the rules are fitted to the data. Additionally the resulting decision boundaries should behave intuitively predictable. From our findings in Section 2.3, this suggests to use Gaussian membership functions together with the product t-norm $\top_{\text{prod}}$.

Globally defined fuzzy sets are usually easier described linguistically. However, this can often significantly limit flexibility. As we assume that a high measure of flexibility might be needed to sufficiently capture cluster structure, we prefer locally defined fuzzy sets. With appropriate restrictions on their sizes and relative positions, they can usually still be represented by linguistic terms (cf. Section 2.3.4).

As mentioned earlier, we decided to use an evolutionary algorithm for fuzzy rule induction. The use of an EA is another reason to use locally instead of globally defined fuzzy sets, since these can be more efficiently encoded in chromosomes (cf. Section 4.2.2).

Generally, evolutionary algorithms are computationally more demanding than other approaches. Why did we still decide to use an evolutionary algorithm? In Chapter 3, a variety of non-evolutionary methods has been presented, some of which have been extended to semi-supervised learning in Chapter 5. However, not all the alternative methods are equally appropriate to be extended to semi-supervision. The initial fuzzy partitions, for instance, that are a priori defined in structure based approaches (cf. Section 3.1.1) might lead to premature assignment of unlabeled examples to classes and thus make it difficult to identify the underlying cluster structure. Fuzzy

decision trees (cf. Section 3.1.5) consider only single attributes in one stage. This often allows reasonable splits for labeled examples. However, this procedure is less suited to identify cluster structure. GFMM is a semi-supervised extension of a hyperbox oriented approach (cf. Section 3.1.4). As shown on the example datasets of Section 5.3, the iterative processing of examples leads to premature assignments of unlabeled examples to labeled hyperboxes, which often cannot be restored in later stages. We conclude that these approaches to fuzzy rule induction are less suitable for semi-supervised learning.

Many of the semi-supervised approaches for learning fuzzy models have been extended from fuzzy cluster analysis. Nevertheless, there are a number of reasons that speak for evolutionary rule learning in comparison to search techniques like, for example, alternating optimization. One advantage of using an evolutionary algorithm for fuzzy rule learning is that it can directly use quite different quality measures. On one hand, this is rather convenient, because in contrast to gradient descent or alternating optimization, it is not necessary to build the derivatives of the objective functions. On the other hand, this aspect becomes indispensable if the objective function is non-differentiable, as those presented in Sections 6.3 and 6.4.

Seeger (2001) hypothesized that semi-supervised learning is much more prone to convergence in local minima. The "strong" guidance from the labeled examples might overwhelm the "weak" guidance from the unlabeled examples. The random search of evolutionary algorithms might increase chances to escape local minima and thus to cope with that problem.

Although we considered only numeric attributes, practical applications often have mixed numeric and symbolic features. An evolutionary algorithm can in principle be extended to symbolic features, if the genetic operators and the objective function are modified to take such features into account. This is in general more difficult for methods like alternating optimization.

The most important advantage of evolutionary algorithms is their ability to optimize not only (numeric) parameters, but to learn also structural aspects of the model, like the number of rules or the use of "don't cares" in the antecedents. Hence, they can adapt the flexibility of the model to the complexity of the task. They easily allow to incorporate constraints on the fuzzy sets or rules to maintain interpretability without modifying the objective function. For our goal of understandable fuzzy rules, an evolutionary algorithm thus offers the learning capabilities necessary for the semi-supervision, *and* the possibility to control the interpretability of the resulting rule base.

6.2 An Evolutionary Algorithm for Rule Induction

In this section, we describe implementation aspects of our rule learning framework based on an evolutionary algorithm. We describe how the fuzzy rule base is encoded in the chromosomes and which genetic operators have been chosen. The measures that are presented in Sections 6.3 and 6.4 for the semi-supervised assessment of fuzzy rule bases can be used directly as fitness functions in the evolutionary rule learner.

As discussed in Section 4.2, Michigan-style and iterative rule learning approaches require specially adapted fitness functions to control cooperation between rules. This restriction is dropped in Pittsburgh-style approaches, which simplifies the design of objective functions specialized for semi-supervised learning. We thus implemented a Pittsburgh-style rule learner similar to the approach by Carse et al. (cf. Section 4.2.1). It simultaneously learns fuzzy sets and fuzzy rules, which allows very flexible adaptation of the candidate solutions. In the following sections, we describe similarities and differences of the proposal by Carse et al. and our implemented approach.

Encoding of the Rule Bases

In our Pittsburgh-style approach, each chromosome encodes a complete rule base as depicted in Figure 6.1. We use approximative fuzzy rules, i.e. each rule holds its own fuzzy set definitions. We pointed out before that approximative fuzzy rules have certain appealing properties for a semi-supervised fuzzy model. Additionally, the local definition of fuzzy sets significantly supports evolutionary learning. Each rule consists of the parameters of the antecedent fuzzy sets for every dimension and an integer for the consequent class.[1] We use Gaussian membership functions, requiring two parameters center and width. A special value is used to signal a "don't care" in a dimension of an antecedent. The rules are simply concatenated to build the rule base. Hence, the chromosomes can vary in length, depending on the number of rules.

The initial population of chromosomes is created by generating a user-specified number of rule bases. The number of rules in a rule base is randomly chosen in some given limits. The centers and widths are randomly

[1] The approach by Carse et al. was proposed for fuzzy control. Hence, their consequents are fuzzy sets.

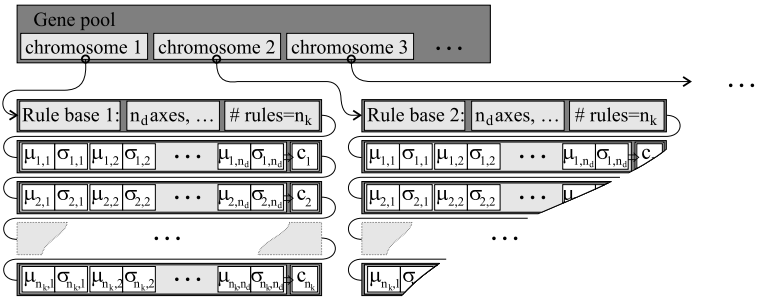


Figure 6.1: Chromosome encoding used in our approach.

chosen such that the fuzzy sets lie in the attribute ranges.

Selection Operator

The selection operator determines the probability for an individual of the current population to reproduce, i.e. to be transferred to the next generation, eventually after mating with another individual and/or mutation.

We chose *tournament selection* as selection operator: two chromosomes are randomly chosen from the pool and the fitter of the two is taken. This operator is more robust compared to fitness proportionate approaches, as it does not depend on the scaling of the fitness function, and computationally cheaper than rank based selection, which implies sorting the chromosomes by fitness (cf. Section 4.1.2).

Additionally, an elitist strategy is used, i.e. the best rule base of the population is copied to the next generation without modifications.

Crossover Operator

In genetic algorithms, recombination of individuals is the key mechanism to sift through the search space for good combinations of partial solutions. Formal considerations of these mechanisms in the schema theorem show the importance of *building blocks*, i.e. of partial solutions that have high average fitness and high probability to remain intact during crossover (cf. Section 4.1.3).

In accordance with this, a good crossover operator should produce valid offsprings that preserve good partial solutions from their parents. In our

case of approximative fuzzy rules, obviously the rules or combinations of rules lend themselves as natural building blocks. The ordinary (one point or two point) crossover most likely preserves information from genes that lie close to each other on the chromosome. As discussed in Section 4.2, rules generally describe multi-dimensional regions in input space and thus cannot be ordered on one-dimensional chromosomes. Good combinations of adjacent rules are therefore easily separated by ordinary crossover.

Carse et al. (1996) proposed so-called one or two point *ordered* crossover as a solution to that problem (compare Figure 4.3 in Section 4.2.1). In this approach every axis is randomly split into two intervals and one of them is chosen. In the child's rule base we copy the rules of one parent lying inside the intersection of these intervals, and the rules of the other parent lying outside. The resulting rule base of the child chromosome is covered with rules from the whole rule space, and neighboring rules have a higher probability to survive together. The number of rules may change by this procedure.

We initially used this crossover operator (Klose and Kruse, 2002). However, we found that it bears some pitfalls and thus modified it (Klose, 2003b). First, it is not rotation invariant. This can easily be seen for the one-point version (cf. Eq (4.12)): rules with centers in $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ are *always* separated, as $a_i < c_i$ is true for all i (and $c_i \neq a_i$), and $b_i < c_i$ is false for all i. Other pairs of diametric hyperbox corners are, however, *never* separated, because they have at least one dimension with $C_{ir} = b_i$, thus $C_{ir} \not< c_i$, and consequently condition Eq. (4.12) is false for all hyperbox corners except $(a_1, \ldots, a_n)$. This rotation dependence is not cured by the two-point version, as the exchanged hyperboxes still lie along the main diagonal (from $(a_1, \ldots, a_n)$ to $(b_1, \ldots, b_n)$).

Additionally, the exchanged subspaces have rather different shapes, especially if dimensionality is high. The normalization of the random variable in Eq. (4.13) is reasonable to balance the sizes of the exchanged hyperbox and the remainder of the input space. It leads, however, to other side effects. Let us for example, assume ten input dimensions. Then the average length for an edge of the hyperbox will be $0.5^{0.1} > 93\%$ of the attribute range. Eighty percent of the edges will be longer than 85%, and twenty percent even longer than 97% of their attribute range. Thus, rules from a rather large contiguous and compact area (namely a hypercube) of the input space are copied from one parent. The remainder of the input space is made up of the "margins". While its average hypervolume is equal to that of the hyperbox, adjacent rules from the second parent are more likely to be separated.

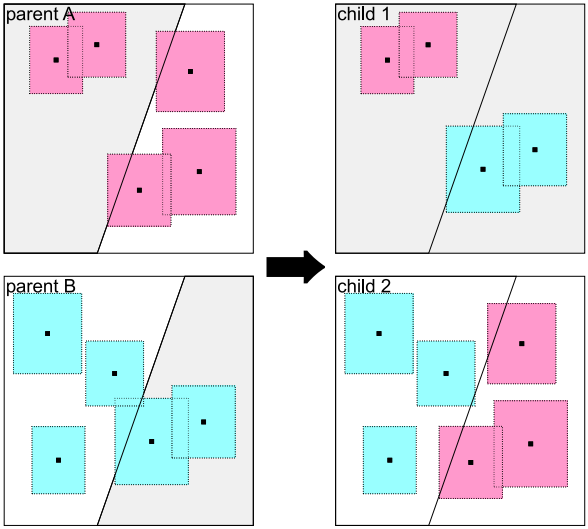


Figure 6.2: Modified one point ordered crossover (cf. Figure 4.3).

Although some of the problems could be eliminated,[2] the asymmetry of hyperbox subspace and its complement is a general problem. We suggest to use shapes for the subspaces that have similar probabilities for nearby rules of both parents to be jointly copied to the offspring. We therefore modified the crossover by defining a random hyperplane and choosing the rules that have their center on one or the other side (see Figure 6.2). The hyperplane is defined by a random normal vector and a random distance of the hyperplane to the center of the input space. The approach is obviously rotation invariant due to the random orientation of the hyperplane. The distance to the center is restricted such that the hypervolumes of the exchanged subspaces are approximately balanced.

Mutation

Our algorithm optimizes both, structure of the rule base and parameters of the fuzzy sets, and hence the mutation operator also affects structure and parameters. Structural mutation changes the number of rules. Rules are

[2] Rotation invariance, for example, could be achieved by choosing for each dimension a random relation operator $<$ or $>$ for Eq. (4.12).

deleted with a certain small probability. Subsequently, new random rules might be created and added to the rule base. For the fuzzy sets, mutation is performed by overlaying the parameters, i.e. centers and width of the fuzzy sets, with small Gaussian distributed noise (cf. Section 4.1.1). Additionally, there is a small chance that fuzzy sets are set to the special value for "don't care" (or that "don't cares" are reverted to fuzzy sets). The probability of parameter mutations is much higher than that of structure mutations.

Repair Mechanism

To enable linguistic interpretation of the fuzzy sets after learning, the positions and sizes of the fuzzy sets must be restricted. because crossover and mutation can violate these restriction, we implemented a repair mechanism that checks the fuzzy sets of every dimension after those operations.

We pointed out in Section 2.3.4 that linguistic interpretability can be achieved by requiring that the fuzzy sets are ordered. The order of the fuzzy sets is given by a definition of *larger*, i.e. a fuzzy set μ_A is larger than a fuzzy set μ_B, if there is an intersection point x_0 such that $\mu_A(x)$ is larger for all points x left of x_0 and $\mu_B(x)$ is larger for all points x right of x_0. For fuzzy sets ordered according to this relation we can assign the usual labels (e.g. small, medium, large) or mixtures of these (e.g. small to medium). Our repair mechanism subsequently tests pairs of fuzzy sets in the order of their centers. If a conflict between adjacent fuzzy set occurs, its parameters are shifted to restore the ordering. To avoid directional bias, in every generation the fuzzy sets are either tested and corrected from right to left, or from left to right.

6.3 Modified Davies-Bouldin Index

The semi-supervised approach by Demiriz et al., described in Section 5.2.4, performed rather good on our examples (cf. Section 5.4). The combination of the Davies-Bouldin index as explicit measure of dispersion and the Gini index as explicit impurity measure gave the algorithm balanced control between supervised and unsupervised learning. The authors used this measure in combination with a simple genetic algorithm. However, the approach is restricted to point-prototypes and isotropic distances. In this section, we propose an extension of this approach that is suited to serve as a fitness function for our proposed evolutionary fuzzy rule learner.

A dispersion measure is calculated from the coordinates of all available

tuples $\mathcal{D}^l \cup \mathcal{D}^u$, ignoring the known class labels of $\mathcal{D}^l$. Its purpose is to measure how well the rules are matched to the tuples in the input space. The inverse Davies-Bouldin index proposed by Demiriz et al. (1999, 2002) measures the relation of inner-cluster to inter-cluster distances. The definitions are based on Euclidean distances, and are thus not suited to consider the variable sizes and shapes of fuzzy rules. However, for Gaussian membership functions with fixed widths σ and $\top_{\text{prod}}$ as conjunction operator, there is a close relationship between distance from rule center $\mathbf{c}$ and (negative logarithm of) rule activation:

$$\begin{aligned} -\ln act(\boldsymbol{x}) &= -\ln\left(\mu_1(x_1) \cdot \cdots \cdot \mu_{n_d}(x_{n_d})\right) \\ &= -\ln\left(\exp\left(-\frac{(x_1 - c_1)^2}{2\sigma^2}\right) \cdot \cdots \cdot \exp\left(-\frac{(x_{n_d} - c_{n_d})^2}{2\sigma^2}\right)\right) \\ &= -\ln\left(\exp\left(-\frac{(x_1 - c_1)^2 + \cdots + (x_{n_d} - c_{n_d})^2}{2\sigma^2}\right)\right) = \frac{1}{2\sigma^2}\,\|\boldsymbol{x} - \mathbf{c}\|^2. \end{aligned} \tag{6.1}$$

Generally, activations take shape and size of fuzzy rules into account. Thus, we define a modified Davies-Bouldin index based on activations (Klose, 2003a). Let X_i be the set of tuples assigned to cluster v_i. The inner-cluster distance, originally defined as the average distance of the tuples to their cluster

$$\alpha_i = \sqrt{\frac{1}{|X_i|} \sum_{\boldsymbol{x} \in X_i} \|\boldsymbol{x} - v_i\|^2}, \tag{6.2}$$

is replaced by the average activation

$$\hat{\alpha}_i = -\frac{1}{|X_i|} \sum_{\boldsymbol{x} \in X_i} \ln \operatorname{act}_i(\boldsymbol{x}). \tag{6.3}$$

The original inter-cluster distance is simply the Euclidean distance between two cluster centers v_i and $v_{i'}$

$$\delta_{ii'} = \|v_i - v_{i'}\|. \tag{6.4}$$

Due to the rule shapes, we cannot use this distance. However, the closer—and thus the more overlapping—two rules i and i' are, the higher the activations will be that tuples of rule i yield for rule i'. Hence we define the average activation that tuples from rule i yield for rule i':

$$\hat{\alpha}_{ii'} = -\frac{1}{|X_i|} \sum_{\boldsymbol{x} \in X_i} \ln \operatorname{act}_{i'}(\boldsymbol{x}). \tag{6.5}$$

We then approximate the intra-cluster distance based on the combined average activations

$$\hat{\delta}_{ii'} = \frac{1}{2}(\hat{\alpha}_{ii'} + \hat{\alpha}_{i'i}). \tag{6.6}$$

The inner-cluster to inter-cluster ratio for two clusters is defined as

$$\hat{\rho}_{ii'} = \frac{\hat{\alpha}_i + \hat{\alpha}_{i'}}{\hat{\delta}_{ii'}}. \tag{6.7}$$

From this we get an alternative dispersion measure

$$\text{dispersion} = \widehat{\text{DB}} = \frac{1}{k}\sum_{i=1}^{k} \max_{i' \neq i} \hat{\rho}_{ii'}. \tag{6.8}$$

The impurity, i.e. the Gini index, is calculated from the assignment of labeled examples to rules. Because only the crisp class after winner-takes-all defuzzification, but not the shape of rules, is considered to calculate impurity, the Gini index can be used without modifications as described in Section 5.2.4.

The fitness function is a linear combination of the two measures:

$$\text{fitness} = \alpha \cdot \text{impurity} + (1 - \alpha) \cdot \text{dispersion}, \alpha \in [0, 1]. \tag{6.9}$$

This fitness can directly be inserted into our evolutionary framework.

6.4 A Measure Based on the MDL Principle

Although the performance of the DBG approach was rather good, we have some reservations about the combination of impurity and dispersion measure. The measures are derived from rather different underlying theories. As our evolutionary framework uses tournament selection, monotonic transformations of the fitness function—like, for example, scaling, squaring or taking the logarithm—do not affect the results. However, as we linearly combine two measures, such transformations of the individual measures *do* matter. Therefore, we prefer to use measures that are derived within the same theoretic framework.

The Minimum Description Length (MDL) principle is an information theoretic approach for statistical model selection that allows to compare models of different complexity (Rissanen, 1983, 1989). In this section, we propose a fitness function that is based on the MDL principle. This makes it possible to measure impurity and dispersion, and additionally regularize model complexity in one single theoretic context.

Minimum Description Length Principle (MDL)

Statistical model selection tries to find for a given dataset the optimal model within competing families of models. Maximum likelihood is not well suited for this problem as it generally chooses the model with the greatest flexibility. This model, however, will tend to overfit the data and have poor generalization ability.

The Minimum Description Length Principle suggests to choose the model that yields the shortest description of data. The idea of the MDL principle is that the data has to be transmitted from an (imaginary) sender to an (imaginary) receiver across a communication channel. Structure in the data can be used for more efficient codes that result in shorter messages. However, both sender and transmitter need to know the encoding scheme of the data. Thus the message is compound by first transmitting the coding scheme, and then—using this scheme—the data. Complex models need a longer coding scheme, as more free parameters have to be transmitted. However, the resulting data part of the message will usually be shorter. The model with the shortest overall message length is chosen, as it is assumed to give the best description of the structure in the data.

In the following we describe how a fuzzy rule base can be interpreted as an encoding scheme and used to efficiently transmit data. The transmission of labeled and unlabeled data naturally leads to a measure that allows to assess the quality of rule bases with varying numbers of rules in the context of partially labeled data.

In our case, the competing models are rule bases, i.e. the structure in the data is described by sets of fuzzy rules. This structure can be exploited to transmit the tuples. Instead of directly transmitting a tuple's coordinates and its class label, we first transmit which rule is used to transmit the tuple. For each rule, sender and receiver agree on an encoding where more probable coordinate values and class labels are associated with shorter codes. If the structure defined by the code matches the distribution of the points better, the average code length per tuple will be smaller.

If symbols from a finite set $\mathcal{A}$ with a probability distribution P shall be transmitted across a (noiseless) communication channel, Huffman's algorithm allows to construct the corresponding optimal binary prefix code (Cover and Thomas, 1991). The length of a transmission can be calculated by summing up the code lengths for the symbols of the transmitted message.

We should note that the MDL principle is used only as a measure to compare models. We are generally not interested in actually transmitting the data. Thus, we do not construct a code, but for simplicity estimate the

code lengths of symbols $a \in \mathcal{A}$ as

$$l_P(a) = -\log_2 P(a), \tag{6.10}$$

which can be shown to be the theoretic lower bound for any code and which is usually close to the length resulting from Huffman coding. Thus, for each part of a message we have to choose an appropriate probability distribution over all possible alternatives. The code length in bits of an instantiation a is calculated as $l_P(a)$.

In our case, the complete message consists of the following parts:

- the rule base (the *encoding*), e.g. the number of dimensions, the number of rules, the fuzzy sets used in a rule. This information is used to transmit
- the data tuples (the *data* itself), e.g. the index of the rule that is used to encode this tuple, the class labels and the exact values of the tuple using a rule specific code.

The following sections will detail the parts of the message. It is advantageous for the encoding to treat labels and coordinates separately. Thus we describe the lengths of rule base, class labels, and tuple coordinates in individual sections. These three parts of the message correspond to measures of complexity, impurity, and dispersion, respectively.

Message Length of the Rule Base

The first part of the message is the definition of the rule base, i.e. the specification of the structure which would enable the receiver to reconstruct the optimal code used to transmit the data itself. However, as we mentioned before, we do not actually want to transmit the data, and thus we can make some simplifications. Some parts of the description—like, for example, the domain description—have equal length for any competing model. Hence it can be ignored when we compare message lengths.

The variable part of the rule base consists of fuzzy sets and rules. We assume that the parameter values of the fuzzy sets are equally probable within their given ranges, and that these ranges have a resolution of s steps.[3] Thus, every value has a probability of $p_{par} = \frac{1}{s}$, and the corresponding code length is $l_{par} = -\log_2 p_{par} = \log_2 s$. As s is constant and the number

[3] A quantization into s steps means that we define borders $x_i = a + \frac{b-a}{s}\, i, i \in [0, s]$, and consider intervals $z_i = (x_{i-1}, x_i], i \in [1, s]$.

of parameters per fuzzy set is constant, the length of a fuzzy set l_{fset} is constant, too.

We encode whether a specific dimension of a rule is given by a fuzzy set or set to the special symbol "don't care". If we assume a probability p_{dc} for "don't cares", the length for a single dimension of a rule is

$$l_{dim}(r,d) = \begin{cases} -\log_2 p_{dc}, & \text{if dim } d \text{ in rule } r \text{ is "don't care"} \\ l_{fset} - \log_2(1-p_{dc}) & \text{else.} \end{cases} \tag{6.11}$$

The lengths of a single rule and a complete rule base are thus

$$l_{rule}(r) = \sum_{d=1}^{n_d} l_{dim}(r,d), \text{ and} \tag{6.12}$$

$$l_{base} = \sum_{r=1}^{n_k} l_{rule}(r), \tag{6.13}$$

respectively. Obviously, this part of the message measures complexity of the rule base, i.e. the number of rules and the complexity of the antecedents.

Message Length of the Data

As shown in Section 2.4, we can interpret fuzzy membership values as probability densities. We use this interpretation to transmit the tuple coordinates.

Let us first assume that we want to transmit a value x without any knowledge of the distribution of values. Analogous to the parameters of the fuzzy sets, we set up a precision of s steps for the range $[a,b]$, and thus need $\log_2 s$ bits to encode x. However, the code lengths can be optimized if we assume that the values come from a probability distribution with probabilities proportional to the membership function. The probability of a certain value x, or more precisely the probability that x lies in its containing interval $z_x = (x_{i-1}, x_i], x_{i-1} < x \leq x_i$, is

$$P(z_x) = \frac{\int_{x_{i-1}}^{x_i} \mu(t)dt}{\int_a^b \mu(t)dt}. \tag{6.14}$$

For large s, we have small intervals z_x and can thus approximate the integral

$$\int_{x_{i-1}}^{x_i} \mu(t)dt \approx (x_i - x_{i-1}) \cdot \mu(x_i) = \frac{b-a}{s} \cdot \mu(x_i) \approx \frac{b-a}{s} \cdot \mu(x). \tag{6.15}$$

The second approximation uses that $x \approx x_i$ and assumes that the membership function μ is sufficiently smooth. We thus approximate the code length to transmit a value x using a fuzzy set μ by

$$l_\mu(x) = -log_2 P(z_x) = \log_2 s + \log_2 \frac{\int_a^b \mu(t)dt}{b-a} - \log_2 \mu(x). \tag{6.16}$$

Obviously, for "don't cares", i.e. for $\mu \equiv 1$, we get $l_\mu(x) = \log_2 s$, as we already argued above. Thus we can use Eq. (6.16) for all fuzzy sets, including "don't cares". To transmit a tuple $\boldsymbol{x}$ using a rule r we need a message of length

$$\begin{aligned} l_{tuple}(\boldsymbol{x}, r) &= \sum_{d=1}^{n_d} l_{\mu_{d,r}}(x_d) \\ &= n_d \log_2 s + \sum_{d=1}^{n_d} \log_2 \frac{\int_a^b \mu_{d,r}(t)dt}{b-a} - \sum_{d=1}^{n_d} \log_2 \mu_{d,r}(x_d) \\ &= n_d \log_2 s + \sum_{d=1}^{n_d} \log_2 \frac{\int_a^b \mu_{d,r}(t)dt}{b-a} - \log_2 act_r(\boldsymbol{x}) \end{aligned} \tag{6.17}$$

This part measures cluster dispersion. The closer the tuples lie to the centers of the membership functions of their rules (i.e. the better the rules are adapted to the data), the higher the (average) rule activation, and hence the shorter the (average) code length. Eq. (6.17) is used for encoding the coordinates of both, labeled and unlabeled tuples. For each tuple ω, we use rule r_ω that yields the shortest code:

$$l_{tuples}(\mathcal{D}^u \cup \mathcal{D}^l) = \sum_{\omega \in \mathcal{D}^u \cup \mathcal{D}^l} l_{tuple}(\boldsymbol{x}^\omega, r_\omega). \tag{6.18}$$

Strictly speaking, we also have to transmit the indices of the used rules. However, as the corresponding message length is small and almost constant, we neglect it. Notice that although the chosen precision s affects the message length, it adds a constant term only and can thus be ignored.

Message Length of the Class Labels

The next part of our message are the class labels of the tuples. We first consider the labeled examples $\mathcal{D}^l$. A basic way to encode their labels would be to assume equal probabilities $P(c) = \frac{1}{n_c}$, which leads to a description

length of $n \cdot \log_2 n_c$ for n examples. However, this encoding can usually be improved by first transmitting the class frequencies and use these for an optimized code. We assume that any division of the n objects into n_c classes is equally probable. From combinatorics we know that there are $\frac{(n+n_c-1)!}{n!(n_c-1)!}$ possible divisions. We assume these divisions to be enumerated, such that by simply transmitting the index of the division, the receiver can reconstruct the class frequencies N_{c_1} to $N_{c_{n_c}}$. To assign the labels to the tuples with knowledge of the class frequencies, we can permute the tuples appropriately and the assign class c_1 to the first N_{c_1} tuples, class c_2 to the next N_{c_2} tuples, and so on. Because permutations of tuples within a class are irrelevant, only $\frac{n!}{N_{c_1}!\,N_{c_2}!\,\ldots\,N_{c_{n_c}}!}$ permutations are different. If we use this scheme to transmit the tuples the necessary length is

$$l'_{labels} = log_2 \frac{(n+n_c-1)!}{n!(n_c-1)!} + log_2 \frac{n!}{N_{c_1}!\,N_{c_2}!\,\ldots\,N_{c_{n_c}}!}. \tag{6.19}$$

The second term of this description length grows for heterogeneous class frequencies, and takes the minimal value zero, if only one class is present in the considered dataset. We have not yet exploited that we already transmitted which rule is used for which tuples, and that in good rules the majority of tuples belongs the same class. Instead of using l'_{labels} for the complete set of tuples, we can significantly reduce the description length, if we apply Eq. (6.19) to the tuples of each rule individually. Thus, the number of objects n becomes N_r, i.e. the number of objects assigned to rule r; class frequencies N_c are replaced by $N_{r,c}$, denoting the number of objects of class c assigned to rule r. The description length is then calculated as

$$l''_{labels} = \sum_{r=1}^{n_k} log_2 \frac{(N_r+n_c-1)!}{N_r!(n_c-1)!} + \sum_{r=1}^{n_k} log_2 \frac{N_r!}{N_{r,c_1}!\,N_{r,c_2}!\,\ldots\,N_{r,c_{n_c}}!}. \tag{6.20}$$

Eq. (6.20) describes only labeled examples. For the examples from $\mathcal{D}^u$, the labels are unknown. We assume that these examples should be transmitted with the minimal possible effort, and thus assume them to belong to the majority class $c_{\max}(r)$ of a rule r. With $N_{r,?}$ denoting the number of unlabeled tuples assigned to rule r, we get the final description length for the class labels as

$$\begin{aligned} l_{labels} &= \sum_{r=1}^{n_k} log_2 \frac{(N_r+n_c-1)!}{N_r!(n_c-1)!} \\ &+ \sum_{r=1}^{n_k} log_2 \frac{N_r!}{(N_{r,c_{\max}(r)}+N_{r,?})! \prod_{c\in\mathcal{C}, c\neq c_{\max}(r)} (N_{r,c}!)}. \end{aligned} \tag{6.21}$$

Misclassified labeled tuples make that part of the message longer, as the probability distribution of the corresponding rule becomes more heterogeneous, and thus this part of the measure quantifies rule impurity.

Message Length as Fitness Function

The total description length of the (considered parts of the) message is simply the sum of the single lengths:

$$l_{total} = l_{base} + l_{tuples} + l_{labels}. \tag{6.22}$$

One initial motivation for using the MDL principle was to avoid seemingly subjective weights. However, this leaves the user with little control over the learning. For instance, more dimensions inevitably increase l_{tuples}, and can thus overwhelm the influence of the labels. Similarly, the relation of the lengths of the three parts depends on the sizes of $\mathcal{D}^l$ and $\mathcal{D}^u$.

As the different influences, like the number of dimensions or the size of the unlabeled dataset, basically linearly affect the respective message parts, we suppose that multiplicative scaling of the parts does not interfere too much with the unified theoretic context of MDL. For the evolutionary learning it is only important that scaling yields an appropriate ranking of the solutions.

As we mentioned in the descriptions of the message parts, their lengths can also be seen individually as measures of complexity, cluster dispersion and cluster impurity. Since weighting of the individual parts of the message can be important to balance their influence on the learning, we define the fitness function as

$$fitness(\mathcal{R}, \mathcal{D}) = w_{base} \cdot l_{base} + w_{tuples} \cdot l_{tuples} + w_{labels} \cdot l_{labels}. \tag{6.23}$$

Empirically we found that a choice of $w_{base} \approx w_{labels} \approx 1$ and $w_{tuples} \approx \frac{l_{labels}}{l_{tuples}}$ yields reasonable balance between the measures. Obviously, we do not know l_{labels} or l_{tuples} in advance. In practice, we thus performed several runs with different values for w_{tuples} to see in which range l_{labels} and l_{tuples} lie. Based on these observations, we chose w_{tuples}. In most cases, the results were rather robust with regard to the exact values of the weights.

6.5 Empirical Evaluation

In this section, we apply the semi-supervised algorithms proposed in this chapter to the datasets of Section 5.3, and compare it to the approaches of Chapter 5.

We use the evolutionary rule learning framework with the two measures proposed. Additionally, we use two variants, once with fuzzy membership functions of fixed widths, and once with variable widths (i.e. only regularized by the repair mechanism). Accordingly, we compare the following methods:

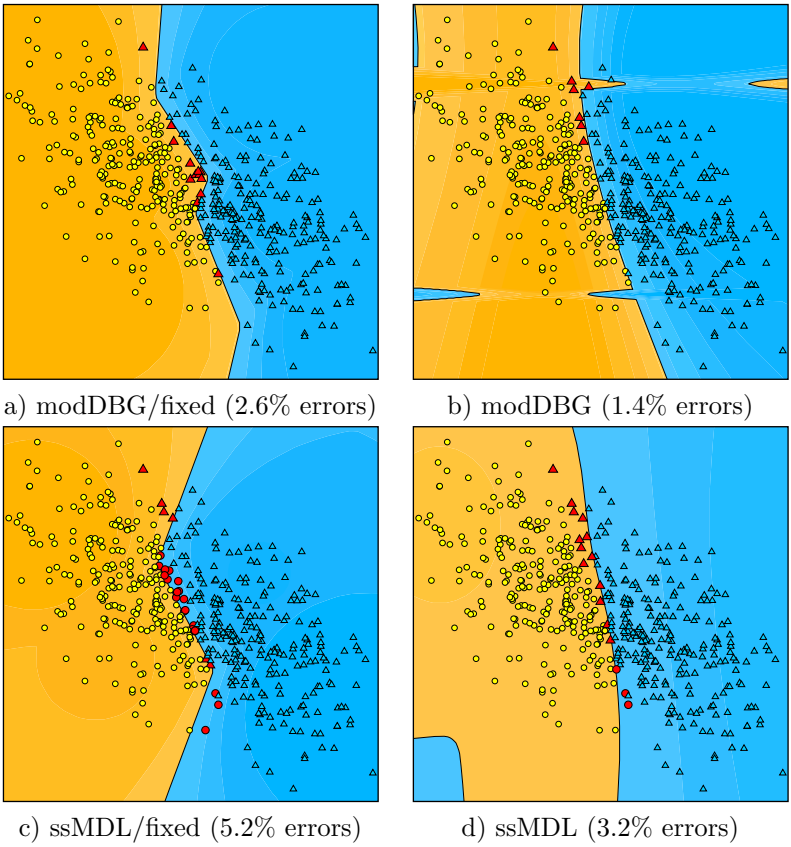
- **modDBG:** the fitness function based on the Gini index and the modified Davies-Bouldin index (Section 6.3).
- **modDBG/fixed:** same as modDBG, however with fixed widths of fuzzy sets.
- **ssMDL:** the fitness function derived from the Minimum Description Length Principle (Section 6.4).
- **ssMDL/fixed:** same as ssMDL, however with fixed widths of fuzzy sets.

6.5.1 Artificial Benchmark Datasets

The purpose of the artificial datasets, which we described in detail in Section 5.3, was to illustrate situations where pure supervised or unsupervised methods perform poorly, and to compare how the different semi-supervised approaches can cope with these situations. In the following, we present the results from applying our semi-supervised algorithms to these artificial datasets.

Artificial Dataset I

The first artificial dataset contains almost no cluster structure that unsupervised methods could exploit to find the class boundaries. However, given the labeled examples, it is a rather easy supervised learning task. Not all semi-supervised algorithms are equally capable of dealing with such data. For instance, ssFCM and ssGK even made errors on the (linearly separable) labeled examples. All approaches presented in this chapter succeed to separate the labeled examples. However, they differ in the shape of the decision boundaries, and thus in their generalization errors. Both approaches based on the modified Davies Bouldin index (modDBG) are close to the optimal separating line. Figure 6.3b shows a general problem of modDBG: it tends to learn degenerated rules and thus, apart from two rules that almost perfectly fit the problem, creates two further rules which are much wider than high. They lead to "bulges" in the decision boundary. Although they do not degrade performance on this dataset, they are certainly not desirable. The version with fixed widths of the fuzzy sets (modDBG/fixed) is safe from this

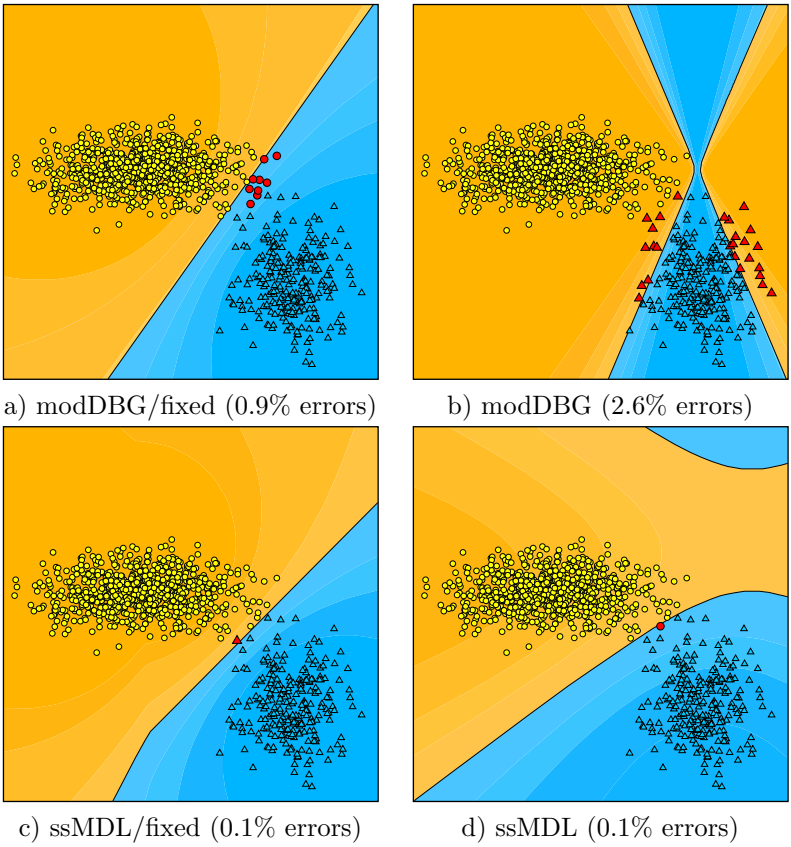


a) modDBG/fixed (2.6% errors) b) modDBG (1.4% errors)

c) ssMDL/fixed (5.2% errors) d) ssMDL (3.2% errors)

Figure 6.3: Experimental results on 1st artificial dataset.

behavior. However, it needs more rules to approximate the distribution. In spite of a slightly higher error, ssMDL uses only three rules on this dataset to describe a rather smooth and close to optimal decision boundary. Its fixed widths version (ssMDL/fixed) has more difficulties to generalize the linearly separable decision boundary.



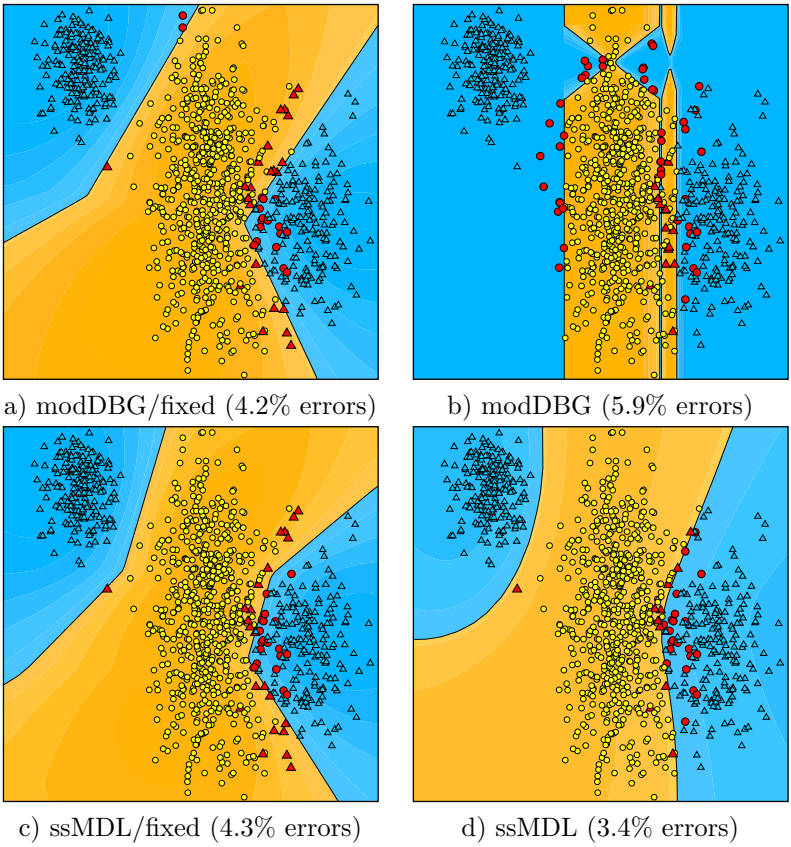
a) modDBG/fixed (0.9% errors) b) modDBG (2.6% errors)

c) ssMDL/fixed (0.1% errors) d) ssMDL (0.1% errors)

Figure 6.4: Experimental results on 2nd artificial dataset.

Artificial Dataset II

The second artificial dataset demonstrates a situation where semi-supervised learning has supposedly the highest potential: there is some cluster structure in the unlabeled data, but the labeled examples given are untypical for the cluster prototypes. In such situations, pure supervised learning will perform poorly. This semi-supervised learning task is solved virtually perfect by both versions of ssMDL (Figure 6.4c and 6.4d), and only slightly worse by modDBG/fixed (Figure 6.4a). Although the modDBG version



a) modDBG/fixed (4.2% errors) b) modDBG (5.9% errors)

c) ssMDL/fixed (4.3% errors) d) ssMDL (3.4% errors)

Figure 6.5: Experimental results on 3rd artificial dataset.

with variable widths produces more errors near the decision boundary, it still succeeds to adequately identify the cluster structure (Figure 6.4b).

Artificial Dataset III

The third dataset combines the challenges of the first two datasets and adds complexity, because one of the classes is split into two clusters. In spite of the restrictions that we imposed on the fuzzy sets, all four tested variants perform rather well. The error rates are similar to those of the

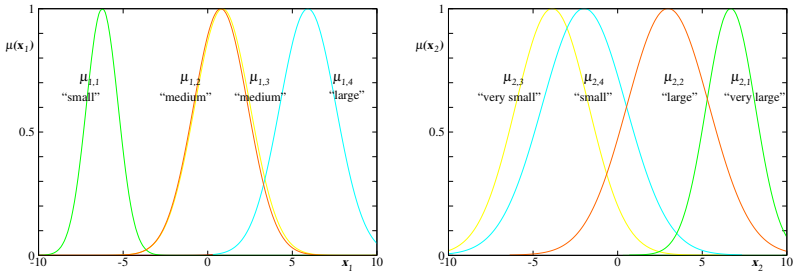


Figure 6.6: Fuzzy sets of the rule base learned by ssMDL (cf. Figure 6.4d) and possible linguistic descriptions.

R_1:	IF x_1 IS small	AND x_2 IS very large	THEN class IS △
R_2:	IF x_1 IS medium	AND x_2 IS large	THEN class IS ○
R_3:	IF x_1 IS medium	AND x_2 IS very small	THEN class IS ○
R_4:	IF x_1 IS large	AND x_2 IS small	THEN class IS △

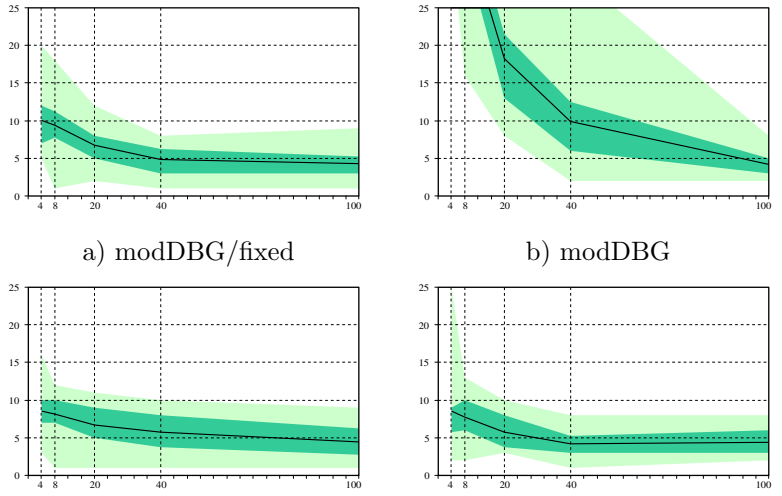
Table 6.1: Rule base learned by ssMDL.

best approaches of the previous chapter (namely DBG and ssMoG). The best results are achieved by both versions of ssMDL, whereas the variable widths of modDBG lead to degenerated fuzzy sets and thus deformations of the decision boundary.

Figure 6.6 and Table 6.1 exemplary show the rule base learned on the third artificial dataset by ssMDL (variable widths). The restrictions applied to the fuzzy sets allow to assign linguistic terms derived from the standard partitioning. Although this has been done manually in this example, the ordering of the fuzzy sets greatly simplifies the algorithms mentioned in Section 2.3.4.

6.5.2 Empirical Results on the Iris Dataset

As can be seen in Figure 6.7, the performance of our proposed semi-supervised fuzzy classifier approaches on the Iris dataset is similar to the better approaches of Chapter 5. There is one exception: the performance of modDBG is disappointing on this dataset. Obviously, the tendency to degenerated fuzzy sets is more critical with the higher number of dimensions (and probably also the rather low number of tuples). When we compare the



a) modDBG/fixed b) modDBG

c) ssMDL/fixed d) ssMDL

Figure 6.7: Experimental results on Iris dataset. The graphs show the error vs. the percentage of labeled examples; black line: mean error, dark corridor: 1st/3rd quartile, bright corridor: minimal/maximal error.

two variants of ssMDL, the increased flexibility of fuzzy models with membership functions of variable width pays only for more labeled examples. Although the average performance of ssMDL is equal or better than that of ssMDL/fixed, the former has a tendency to overfitting when learning from few labeled examples.[4]

6.6 Discussion

In Section 5.4 of the previous chapter, we compared advantages and drawbacks of existing approaches and discussed their suitability for semi-supervised induction of fuzzy rule based classifiers. In this chapter, we listed requirements for a specialized approach, and proposed an evolutionary learning scheme and two alternative semi-supervised fitness functions. Conse-

[4]Recall that labeling 4% and 8% of the examples result in only one or two examples per class, respectively.

	modDBG (fixed)	modDBG (variable)	ssMDL (fixed)	ssMDL (variable)
Expressiveness (6.6):				
Cluster shapes	∘	+	∘	+
Complex clusters	+	++	+	++
Interpretability (6.6):				
Convert to fuzzy sets	+	++	+	++
Linguistic interpretability	+	+	+	+
Interpretab. of rule base	+	−	+	++
Semi-supervised learning (6.6):				
Overall performance	++	∘	++	++
Untypical examples	++	+	++	++
Respecting errors on $\mathcal{D}^l$	++	++	++	++

++: very good +: good ∘: neutral −: bad −−: very bad

Table 6.2: Comparison of the proposed approaches (cf. Table 5.2)

quently, we have to assess the approaches in the same categories—*expressiveness*, *interpretability*, and *semi-supervised learning capabilities*—to see how close we came to our objective of semi-supervised fuzzy classifier learning. The results discussed in the next sections are summarized in Table 6.2.

Expressiveness of Induced Model

The flexibility of the cluster shapes of the approaches depends on the restrictions that are put on the fuzzy sets. The approaches with constant widths have generally less flexible cluster shapes. As argued in Section 6.3, fixed widths lead to the calculation of Euclidean distances. The decision boundary is defined by the interplay of individual rules. The use of Euclidean distances leads to piecewise linear decision boundaries (i.e. Voronoi cells). This still allows to model rather flexible and complex clusters. The flexibility can be increased by allowing fuzzy sets of variable widths. However, the flexibility of the two non-fixed variants is less than that of, for instance, ssMoG, which does not restrict the fuzzy sets for linguistic interpretability, and additionally allows soft labels. Still, we consider the flexibility of the models representable by modDBG and ssMDL to be very good.

Interpretability of Transformed Rule Base

As the proposed evolutionary algorithm directly generates rules with local fuzzy set definitions, the problem of converting clusters to fuzzy sets does not appear. This is also true for the approaches with fixed widths. However, because constant, but arbitrary widths are chosen for all fuzzy sets, the resulting membership functions do not reflect different cluster sizes or shapes in the data.

The evolutionary algorithm generates local definitions of fuzzy sets, which are in general more difficult to describe linguistically. On the other hand, the relative order of the fuzzy sets is guaranteed during learning by the repair mechanism. Together with the usually small number of rules—and thus fuzzy sets per dimension—it is usually possible to find adequate linguistic descriptions (cf. Section 2.3.4).

The small number of fuzzy sets generally also improves readability and thus interpretability of the complete rule base. As we consider interpretability as the degree to which a human expert gains insight into the data at hand by looking at the rules and the fuzzy set definitions, the models differ. The ssMDL approach with variable widths yields rather good cluster descriptions, with fuzzy sets adapted to the shape of the data distribution. This is not possible for the fixed widths approaches. Because the fuzzy sets induced by modDBG tend to be degenerated, the interpretability of the resulting rule base is significantly degraded.

Although the ssMDL algorithms had the possibility to use "don't cares", this has not been used in the presented results. The length of the tuple description can in in most cases be reduced by adapting the corresponding fuzzy sets to the data. Thus, "don't cares" reduce the length of the rule base description but generally increase the length of the tuple description. They are only useful, if either the tuples described by one rule cover almost the complete input range, or if the weight of the rule base w_{base} is excessively high. In this case, however, classification results are often substantially degraded, since for semi-supervision the mechanism of fitting the rules to the data is crucial.

Semi-Supervised Learning Capabilities

With the exception of the variable widths modDBG, all observed performances where similar to those of DBG or ssMoG, and hence we also grade them as very good. The overall performance of the variable widths modDBG is obviously impaired by the weak performance on the Iris dataset. This

example might be less adequate to test semi-supervised learning, since it is rather small and we use randomly chosen labeled examples. However, the other algorithms prove that better performances are possible.

The proposed approaches use fitness functions that can be seen as combinations of measures of cluster impurity and dispersion. We argued in Section 6.1 that this can be useful to ensure good exploitation of the information in $\mathcal{D}^l$ on the one hand, and adaptation of the clusters to $\mathcal{D}^u$ on the other hand, as long as it complies with the labeled examples. Accordingly, the proposed methods showed good performance on the artificial examples, with the exception of the variable width modDBG that had problems to adapt its rules to the data.

6.7 Conclusions

All algorithms that we proposed in this chapter were successful in learning fuzzy classification rules from partially labeled data. The most promising approach seems to be ssMDL with variable fuzzy set widths. It constantly yielded very good results. In contrast to the fixed width approaches, which learn only cluster centers, and in contrast to modDBG with flexible widths, the resulting fuzzy sets and rules of ssMDL correspond nicely with the distribution of the points. This supports interpretability of the rule base.

Although the flexibility of the fuzzy sets was restricted to maintain interpretable fuzzy sets, the performance is competitive or even superior to the semi-supervised approaches presented in the previous chapter. We suppose that this is mainly due to the combination of the underlying (restricted, but still) flexible fuzzy model, the adaptable mixture of impurity, dispersion and complexity measures, and the powerful evolutionary search algorithm.

The examples of this chapter were rather simple and mainly intended to be illustrative. In the next chapter, we show the applicability of the proposed algorithms on two more complex semi-supervised real-world problems.

Chapter 7

Applications of Semi-Supervised Fuzzy Classification

In this chapter, we present two real-world applications of the methods proposed in previous chapter. Both applications come from the field of image processing. In spite of their quite different characteristics, they have in common that large numbers of examples can be generated, but labels are not readily available. This problem frequently occurs in many image processing tasks. In cases where examples are chosen and labeled manually, these are usually scarce and often untypical for the class prototypes. Hence, supervised learning can be expected to perform suboptimal and there might be room for improvements by applying semi-supervised learning.

The application presented first is targeted at supporting object tracking in images. In this context, semi-supervised learning is used for image segmentation (Section 7.1). The second application supports an aerial image analysis system by filtering object primitives and only pass the most promising primitives to the subsequent analysis chain (Section 7.2).

7.1 Semi-Supervision in a Man-Machine Interface

The system described in the following sections has been designed and implemented in a Ph.D. thesis by Schneider (2001) as a computer-based interactive medium to convey spatial information to blind people. The first usable prototype of the system was presented in (Schneider and Strothotte, 2000). The prototype can be used by a blind pedestrian to learn information needed by him or her to travel through an unfamiliar part of a city. The system enables the user to explore and learn the area. Firstly, the system allows a user to get an overview of routes, intersections and buildings. The main focus, however, is on letting the user learn routes after selecting them.

The original man-machine interface of the system is based on a manually adjusted segmentation algorithm. However, changing lighting conditions make an automatic adjustment preferable. To keep the necessary efforts of manual labeling low, we tried to apply semi-supervised learning to the problem. The following sections describe our collaborative work on a semi-supervised extension of the system (Klose and Schneider, 2001).

7.1.1 Application Domain

The idea of the system described here is to convey spatial information by *constructive exploration*. Schneider defined constructive exploration as a method for people to learn spatial information by (partially) reconstructing it with building blocks, guided by an interactive computer system. Methods currently used, such as verbal descriptions or tactile maps, are not fully adapted to the users' needs. In comparison to these methods, the examined "hands-on" approach should give blind people a better understanding of spatial layout.

A constructive exploration system was developed which lets users build routes from game piece-like objects called "route bricks." The route bricks are placed on a pad with a tactile grid representing the map. The map itself is only present in digital form inside of the computer and is conveyed to the user during interaction through speech and sound. After selecting a route by placing a start and a target object on the pad, the user can place one route brick after the other, guided by the system which has calculated the route from map data. One side of a brick can be snapped to a peg at the end of the previous one. A newly placed brick can be rotated with the peg of the previous one as the axis, so the system only has to convey the angle

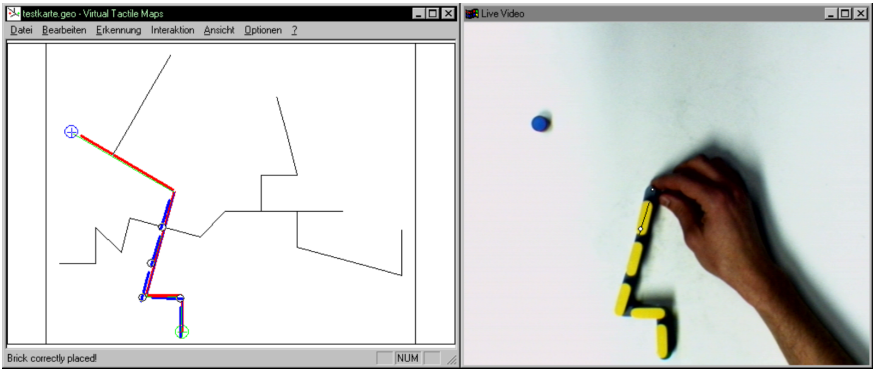


Figure 7.1: The planned path on the computer and the exploration interface

to the user by sound.

In order to react to users placing or moving objects, the system has to track the objects. It does so by image processing: A camera is placed over the pad facing downwards. A simple, but robust approach based on color segmentation is used. Since the system processes color rather than grey-scale images, the objects can be discriminated solely by their respective color. Since the object shapes are already known and can be reduced to points in case of the route end objects and lines in case of the route bricks, segmentation and measurement of areas can be done in one sweep, leading to a highly interactive system on current PC hardware.

Color segmentation also facilitates the discrimination of areas belonging to interaction objects versus those caused by shadows or belonging to other objects, such as the hand. This, in turn, increases the recognition rate, which needs to be rather high, since blind users are not able to detect recognition errors by visual feedback in form of, e.g., graphical cursors on a screen. In the same vein, calibration must be automatic, possibly after setting certain parameters when the system is set up for the first time.

In the first prototype of the system, each color to segment was preset manually to one subspace of the color space. To the human who is setting up the system, each subspace corresponded conceptually to the color of a certain brick, e.g., to the perceived "yellow" of the route bricks. With this approach, the color subspaces had to be readjusted when the lighting condition changed. Even with light constant in time, it varies over the space of the pad due to shadows and the light source not being positioned

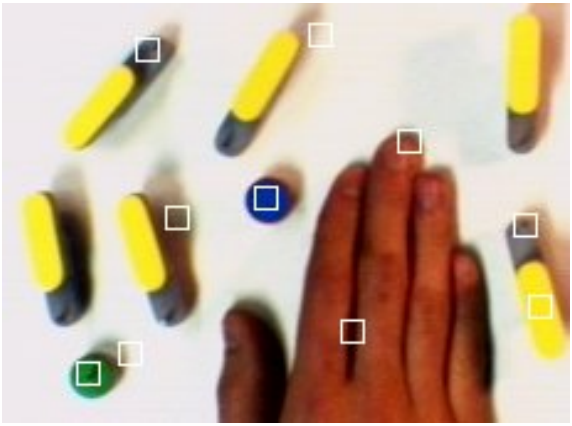


Figure 7.2: The areas of the sample image used as labeled data.

directly over the pad. We therefore investigated ways to let the system learn color values for segmentation in the current set-up after giving it only a few prototypical color values.

7.1.2 Semi-Supervised Image Segmentation

Image segmentation is often performed in an unsupervised manner (cf. Bezdek et al., 1993, for a survey). Pixels are usually treated as individual objects. Their color values in an appropriate color space are used as attributes. In domains with well differentiable objects (and object appearances) this is a feasible approach. However, if clusters overlap, unsupervised segmentation has difficulties in finding the correct class borders. Additionally, the labels of the clusters still have to be assigned manually.

On the other hand, we could use supervised classification. The main drawback of supervised classification is the need for labeled training data. The images taken from the camera in the application have a resolution of 300×300 pixels, i.e. we have a maximum of $90'000$ pixels per image that require class labels. Additionally, we can gather arbitrary many sample images with different lighting and camera conditions. Obviously, not all the data can be labeled manually, and thus suggest to use a semi-supervised approach.

The task is to identify green and blue markers, and yellow building blocks and distinguish them from the background. Bensaid et al. (1996) applied

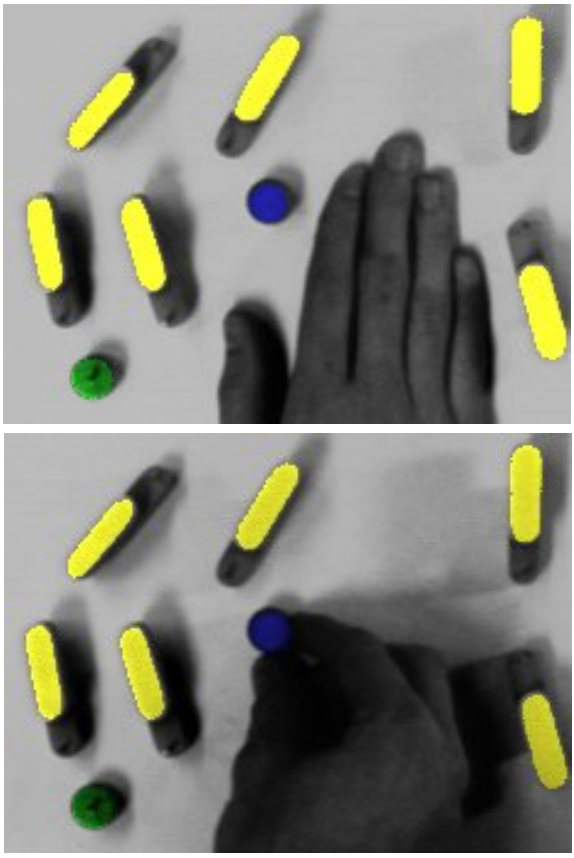


Figure 7.3: (a) The segmentation result on one of the training images, (b) and the result on a test image.

their ssFCM approach[1] to image segmentation. However, as discussed in Section 5.4, ssFCM lacks the flexibility to model multi-cluster classes. As we expect rather complex classes—especially for the background class—, we apply ssMDL to this problem (cf. Section 6.4).

Our test dataset comprises three images under varying lighting conditions, which we clipped to 230×170 pixels. Figure 7.2 shows one example

[1] cf. Section 5.2.3

image. The regions shown in that image are cut out as examples. That is, we used one example region per class (green, blue, yellow), and seven background examples. Each region has a size of 8×8 pixels, totaling in 640 labeled examples. Notice that we labeled examples from one image only. The remaining pixels of that image and all pixels of the other examples images were used as unlabeled examples. For performance reasons we reduced the unlabeled examples to a random subset of 15′000 pixels.

To balance the influences of labeled and unlabeled examples, we set $w_{labels} = 2 \cdot w_{base} = 100 \cdot w_{tuples}$ (cf. Section 6.4). Fig. 7.3a shows the classification result on the image that we used to extract the labeled examples. Pixels classified as *background* are depicted in black and white, the other classes are tinted in their respective colors. Fig. 7.3b shows the result of applying the induced model to an independent test image. As can be seen, the results are both excellent, although the pixels in the test image are generally darker, and there are more shadows.

The learned rule base has eight rules: one for each marker class, and five to describe the background. This demonstrates the complexity of this class, which could not adequately be modeled with ssFCM or ssGK.

The evolutionary learning from the total of 15′640 examples takes a while,[2] and is thus not suited for interactive application. It should, however, be noted that the speed of applying the learned rule bases is much faster. Hence, they can be used for image segmentation in realtime.

7.2 Filtering Object Primitives in Image Analysis

This application considered in the following sections has been studied in a cooperation with the *Research Institute for Optronics and Pattern Recognition* (FGAN/FOM) in Ettlingen/Germany. The focus of this project was the analysis of aerial images to extract man-made structures, such as airfields. Results of the cooperation are described in (Klose et al., 1999, 2000). In these publications we had a different view on the problem and applied pure supervised learning. However, as we explain in the following, it might be more appropriate to consider the task as semi-supervised.

[2] The learning of the reported result used a pool of 200 chromosomes and 50 generations, which took about 15 min on an AMD Athlon 1GHz

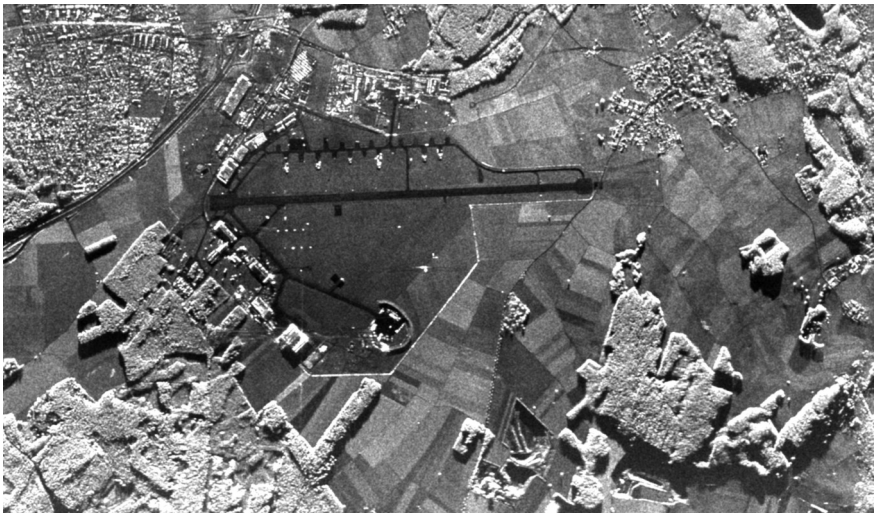


Figure 7.4: Example SAR image

7.2.1 Application Domain

The automatic identification of man-made objects in remotely sensed images is still a challenging task. In the framework of structural analysis of complex scenes a blackboard-based production system (BPI) has been developed at FGAN-FOM (Lütjen, 1986). In this system transformations of the simple objects extracted from SAR (synthetic aperture radar) images into more complex objects are controlled by production rules. A production net proceeds stepwise according to a model and produces intermediate results with an increasing degree of abstraction (Schwan et al., 1998; Schärf et al., 1998). For instance, the abstraction degrees for the construction of a runway are

$$\textbf{edges} \Rightarrow \textbf{lines} \Rightarrow \textbf{long lines} \Rightarrow \textbf{parallel lines} \Rightarrow \textbf{runways.}$$

The image analysis is based on line segments as object primitives, which is typical for the extraction of man-made objects. Figure 7.4 shows a typical SAR image of an airfield. The result of gradient-based edge detection[3]

[3] The edge extraction algorithm used was proposed by Burns et al. (1986). Especially in noisy images, this operator has a tendency of extracting high numbers of short line segments.

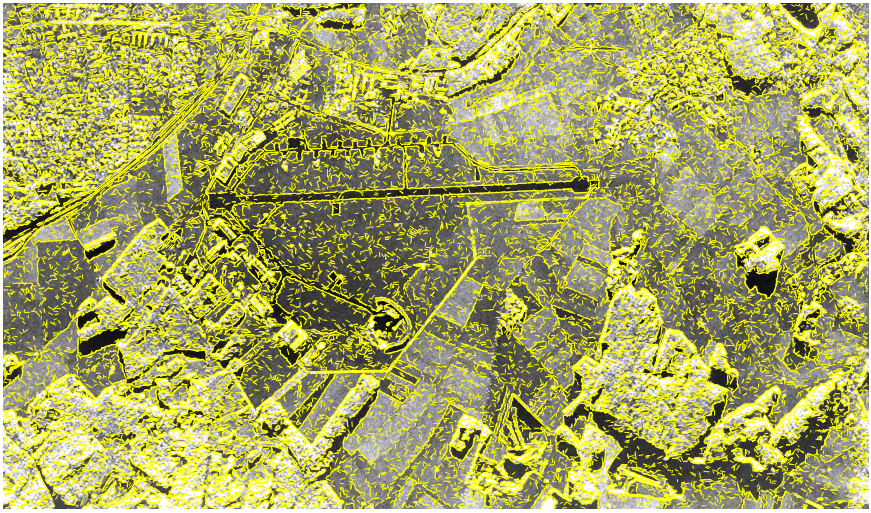


Figure 7.5: 37′659 line segments extracted from the SAR image in Figure 7.4 by Burns' edge detector

applied to the SAR image is shown in Figure 7.5. As the resolution of the images is rather high, the edge detector extracts more than 37′000 edges on this image, which have to be considered during the structural analysis. Although only a fraction of the lines are used to construct, e.g., the runway, the analyzing system has to take all of the lines into account. Unfortunately, time consumption is typically at least $O(n^2)$. Although the system is usually successful in extracting the desired structures from single images, the runtimes are too high to process image data of larger areas.

The idea was that the production process could significantly be sped up if only the most promising primitive objects are identified and the analysis is started with them. This was done by extracting features from the image that describe the primitive objects and that allow to train a classifier that decides which lines can be discarded. Experiments showed that in the case of line primitives the regions next to the lines bear useful information. Therefore rectangular windows with an orthogonal distance d adjacent to the lines are constructed. The gradient across the edge is used to define the line direction and to uniquely distinguish between the left and right window (see figure 7.6). For each line segment a set of statistical (e.g. mean, standard

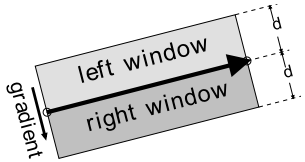


Figure 7.6: The regions next to line used to calculate texture features.

deviation) and textural features (e.g. energy, entropy) is calculated from the gray values in the region. From this set of features, we chose the 8 most important ones.

The initial idea in our project was to apply the analysis process for a number of images on the complete set of object primitives. The runways extracted this way are used to divide the lines into those that were used for the construction of complex structures (the *positive* class) and those that were not (the *negative* class). We tried to learn to differentiate between the classes with a number of classifier approaches. However, it turned out that the problem can hardly be solved directly by most classifiers. The classes were extremely unbalanced. For the image shown, only 20 lines were used to build the runway and are thus used as positive examples. Moreover, the classes were strongly overlapping, and thus almost any classifier approach simply predicts the majority class for any input, because this leads to an extremely low error rate of $\frac{20}{37'659} \approx 0.05\%$. Although it is seemingly perfect, this result is obviously completely useless, as it filters out all object primitives, and thus hinders any object recognition.

We concluded that a classifier has to take into account the special semantics of the task. Misclassifications of the positive and negative class have to be treated differently. As a matter of fact, every missed positive can turn out to be very expensive. Too many false negatives[4] can completely prevent the correct recognition of objects, whereas false positives[5] lead 'only' to considerably longer execution times. We considered this asymmetry in a misclassification cost matrix. This matrix was integrated into the NEFCLASS rule learning and pruning algorithms, which allowed successful application of NEFCLASS on this task. Although the classification

[4]False negatives, i.e. positives classified as negatives, refer to runway primitives that are discarded.

[5]False positives, i.e. negatives classified as positives, result in superfluous primitives that have to be considered in the production process.

was not perfect, in most cases all (or at least enough) relevant object primitives were classified as positive, while the total number of line segments to be processed was significantly reduced (Klose et al., 2000).

7.2.2 Semi-Supervised Line Filtering

Although the obtained error rates seemed to be rather high for a classification problem, the image processing experts were quite content.[6] It turned out that they were actually glad about some of the false positives: the edges of taxiways, parking lots, or roads—having characteristics very similar to those of runway edges—are also needed in later processing stages. However, this means that the classification problem was actually ill-posed: although these additional object primitives should in fact be positive examples, they appear as negative examples in the training dataset. This makes the learning task harder for the classifier, as it—in spite of the included asymmetric misclassification costs—still tries to separate positives from negatives.

We thus conclude that it might be more appropriate to understand the problem as one of semi-supervised learning. The runway segments *plus* similar primitives like taxiways make up the positive examples, although only a part of them is labeled. Additionally, we manually label some negative examples, i.e. line segments that should be discarded during the analysis process. A semi-supervised learning algorithm is then applied to the corpus of unlabeled data plus the few labeled examples. The algorithm should learn to separate the labeled examples, and additionally locate the decision boundary according to the structure of the unlabeled examples.

Figure 7.7 shows the line segments that we manually labeled as positive and negative examples, respectively. Altogether, we labeled only 155 examples (out of 37′659). Obviously, the dataset has the mentioned challenges for semi-supervised learning: extremely few labeled examples, and manually given labels that are probably neither prototypical for their respective classes, nor do they cover the complete spectrum of occurring edge segment characteristics. We applied our semi-supervised evolutionary fuzzy rule learner with the MDL based fitness function and variable widths of fuzzy membership functions (ssMDL), which was presented in the previous chapter.

As the number of unlabeled examples exceeds the number of labeled examples by far, we have to balance the influences of the parts of the quality

[6] On some images about 50% of the examples were classified as positive, and thus the (unweighted) error rate was about the same. However, this still means halving the number of the object primitives.

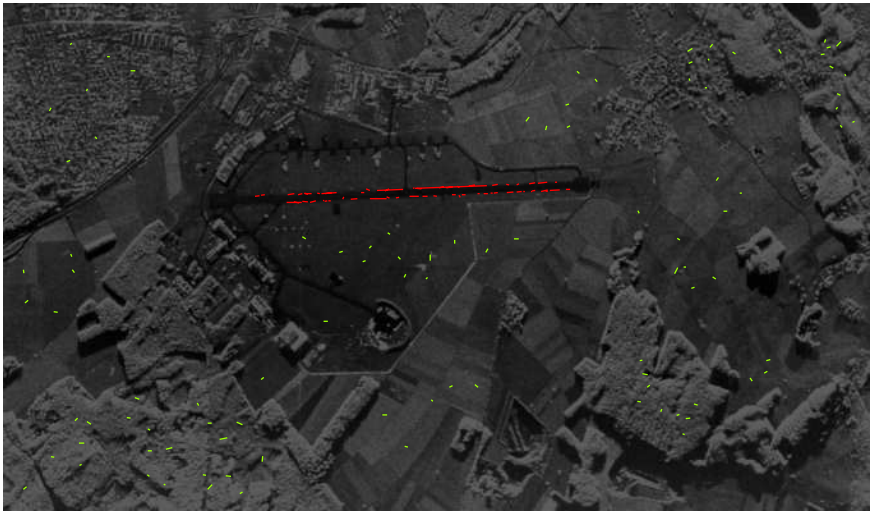


Figure 7.7: Manually labeled examples: 57 positive examples (runway line segments, in yellow), 98 negative examples (in cyan).

measure (cf. Section 6.4). For the presented result, we set the weight w_{labels} of the impurity measure to ten times the weight w_{base} of the rule base length, and to 2000 times the weight w_{tuples} of the dispersion measure, which brings the individual message lengths to roughly the same ranges. On this dataset, ssMDL was not too sensitive to changes of the weights. Figure 7.8 depicts the 3′878 line segments that ssMDL classified as positive. On the labeled examples, it produced 5 errors (3.2%). It can be seen that the line segments necessary to construct runway, taxiways, and traffic system have been successfully identified. The number of potentially relevant object primitives was reduced to 10.3%, which leads to a significantly decreased processing time.

As a comparison, we alternatively treated the problem by pure supervised learning. To reduce the influence of the learning algorithm, we again applied ssMDL to the problem. However, we presented only the 155 labeled examples for learning, and still used a higher weight for the impurity measure. In this way, ssMDL works as a pure supervised rule learner. It produced the same number of misclassifications, i.e. 5, or 3.2%. However, it could not take the unlabeled data into account for locating the decision

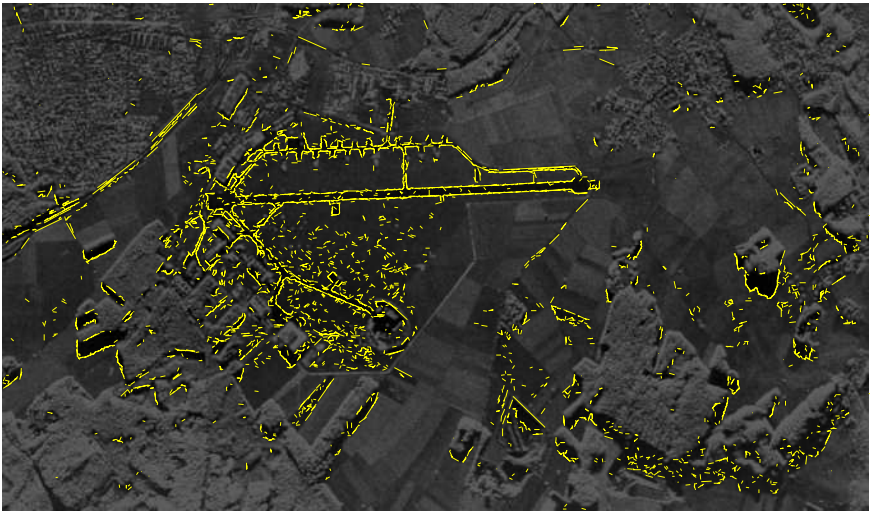


Figure 7.8: 3′878 lines classified as positive by semi-supervised ssMDL.

boundary. On this image, it extracted 6′994 lines as possible runway or traffic system primitives, only a reduction to 18.6%. The resulting positive lines are shown in Figure 7.9. Obviously, the segments extracted by this classifier are also sufficient to construct the desired objects. Although the results of the two classifiers look rather similar, it can clearly be seen that there are many more spurious positive edge segments in Figure 7.9.

It should be noted that we presented the results of reclassification, i.e. we applied the classifier to the data it was trained on. Error rates estimated from reclassification are usually optimistically biased, and thus less credible. However, we suppose that our application is less prone to this optimistic bias. The main problem of reclassification is that of overfitting: a classifier could 'memorize' the examples' labels instead of revealing the underlying regularities. However, most of the data in our application do not have a class label. In case of the supervised application of ssMDL, the unlabeled data were not used at all. In case of semi-supervised learning, only their positional information is used. Thus, the results are still meaningful (apart from the certainly optimistically estimated 3.2% errors on the labeled data). Another typical problem of semi-supervised learning can also be seen: since the labels of the unlabeled data are indeed unknown, we can hardly estimate

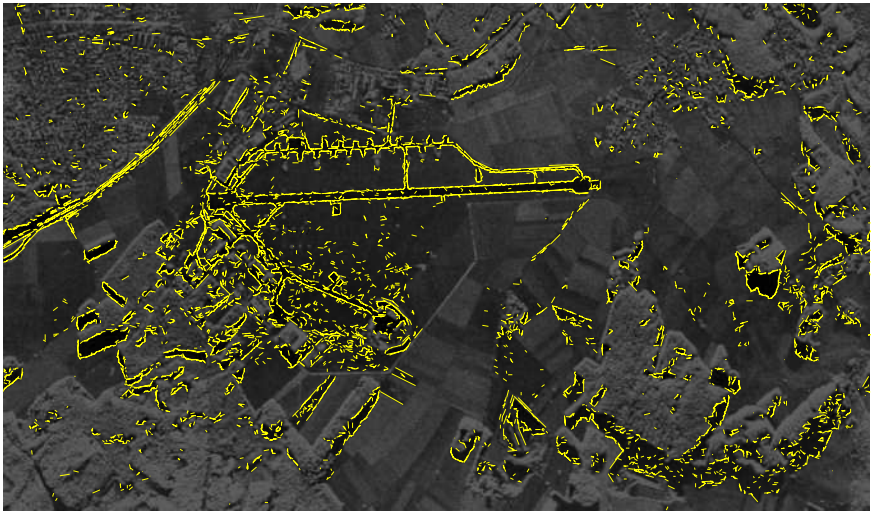


Figure 7.9: 6′994 lines classified as positive by pure supervised learning.

the overall error rate. However, in image analysis applications like ours, we can often at least visually inspect the results.

The alternative interpretation of the line filtering problem as a semi-supervised learning task yields a feasible solution, comparable to the results of our previous approach using asymmetric error costs (Klose et al., 1999, 2000). The semi-supervised learning from labeled and unlabeled data resulted in a much more selective classifier than the pure supervised approach. However, as already mentioned, we can only visually assess the classification results, because we do not have true class labels.

7.3 Conclusions

We have shown the applicability of our proposed semi-supervised rule learner on two real-world problems. In the first example, the goal was to learn an image segmentation. Using a semi-supervised technique should help to reduce the number of labeled examples required. Although one class was rather complex, ssMDL accomplished this goal. In the second example, we managed to filter relevant object primitives in a semi-supervised manner,

in spite of unbalanced class frequencies and strong overlap.

The examples showed that the evolutionary learning algorithm is also applicable to large datasets as they occur in real-world datasets, and that the MDL based measure allows to exploit the structural information in the unlabeled data.

Chapter 8

Conclusions

In this chapter, we summarize the contributions of this thesis to the field of semi-supervised fuzzy classification. Additionally, we discuss open questions and possible directions for further work.

8.1 Contributions

Fuzzy if-then rules are a well-known way to represent classification knowledge. In projects we regularly experienced that fuzzy rule-based classifiers are popular with application experts for their intuitive mode of operation, and the quality and interpretability of the results. Our contributions to this field of research are two-fold. On one hand, we analyzed capabilities and properties of fuzzy classification rules (Chapter 2). On the other hand, we investigated approaches for semi-supervised learning, and devised an approach for inducing interpretable fuzzy classification rules from partially labeled data (Chapters 5 and 6).

Fuzzy Rule-Based Classifiers

The two main aspects considered in this thesis might seem to be only loosely coupled at first sight. However, they bear substantial relation. For instance, the choice of the fuzzy classifier model that we extended to semi-supervised learning in Chapter 6 depended on our findings of Chapter 2.

Although the application of fuzzy classifiers is rather intuitive, there are some pitfalls that should be avoided to get interpretable, yet flexible rule

bases. For instance, we showed that global definitions of fuzzy sets can unintendedly restrict their decision boundaries to be axis parallel. Especially if complete rule bases are defined, fuzzy classifiers can become equivalent to lookup tables with hyperbox cells. Thus less rules, and local fuzzy set definitions are preferable to achieve higher flexibility. Usually, there is a tradeoff between flexibility and interpretability. However, we showed that locally defined fuzzy sets are still interpretable if the number of rules is reasonably small and some weak restrictions are placed on the fuzzy membership functions.

We showed an alternative interpretation of rule weights. If we use $\top_{\text{prod}}$ as t-norm, the calculations of fuzzy rule based classifiers and naïve Bayes classifiers have strong similarities. This allows an interpretation of rule weights that is based on probability theory. The role of weights to compensate for different sizes of membership functions and different class frequencies gets a clear foundation in probability theory.

This result was very important for our semi-supervised approach, as the minimum description length principle requires that we can interpret the membership degrees as probabilities.

Semi-Supervised Fuzzy Classification

In this thesis, our investigations in capabilities and characteristics of fuzzy classifiers served as a prerequisite for the development of our rule learning approach. However, the main focus of this thesis lies on semi-supervised methods.

We discussed theoretical and intuitive considerations in which situations semi-supervised learning can be expected to improve the results of supervised or unsupervised learning. Important aspects include:

- Does the underlying process that generates the data falls in the sampling or in the diagnostic paradigm? The data distribution must bear some inherent structure to make semi-supervised learning feasible.

- How many labeled and unlabeled examples are available? As there is substantially less information contained in the unlabeled examples, their number should be much higher than that of the labeled fraction of the dataset.

- Are the labeled examples already representative for their classes? In that case pure supervised learning might work as well, and the room for improvement by semi-supervised techniques is much smaller. In

general, manually labeled examples can usually be expected to be less representative.

The theoretical and intuitive considerations were empirically backed by artificial datasets that illustrated the challenges for semi-supervised learning.

Although a number of approaches have been proposed for learning from partially labeled data, little has been done on the induction of models that are interpretable in a sense of knowledge discovery. Therefore, we were interested in a semi-supervised algorithm for learning fuzzy classification rules. We reviewed a number of existing approaches that either induce fuzzy models or that use models related to fuzzy techniques. We discussed the capabilities and shortcomings of these approaches on several (artificial and real-world) datasets with respect to

- Expressiveness, i.e. the kind of decision boundaries that can be represented by the models,
- Interpretability, i.e. the readability of the rules, when the models are transformed to (or interpreted as) fuzzy rule bases, and
- Semi-supervised learning capabilities, i.e. how the approaches cope with the challenges of partially labeled datasets.

We found that none of the approaches is in all aspects suitable for the semi-supervised learning of interpretable fuzzy classification rules. From these findings we derived requirements for a semi-supervised fuzzy classification rule learner.

We implemented an evolutionary fuzzy rule learning framework. Two alternative measures were developed that can be used as fitness functions for the rule learner. One measure is based on a semi-supervised (non-fuzzy) clustering approach by Demiriz et al. (2002), which was generalized to work with fuzzy rules. The other approach is based on the Minimum Description Length principle, which allowed to measure dispersion, impurity, and complexity in one theoretic framework.

Especially the MDL based measure yielded convincing results on the example datasets. In Chapter 7, we also successfully applied it to two real-world problems.

8.2 Limitations and Further Work

Although a variety of alternative possibilities have been considered and empirically evaluated in the course of this dissertation, there are a number

of questions and topics that remain open or newly arose:

- Initially, we thought that the unifying theoretic framework of MDL allows to define a measure for semi-supervised learning that—in contrast to, e.g., the Davies-Bouldin/Gini index approach—needs no balancing weights. However, it turned out that weights are necessary for several reasons: It depends on the dimensionality of the dataset, the cardinality of the unlabeled data, the distinctiveness of the inherent structures (with respect to the class information), and, last but not least, prior knowledge about the quality of the labeled and unlabeled examples, which influence the unlabeled data should have on the classifier learning. We suppose that some kind of balancing is thus unavoidable.

  It might be interesting to apply multi-objective learning methods to induce alternative pareto-optimal results the user can choose from. Evolutionary algorithms can be much easier extended to this kind of search than other learning techniques. As a first step, it might also be interesting to vary the weights over time. It could be analyzed, whether the algorithm produces better results or converges faster, if it, for instance, starts with a higher weight on impurity to learn to classify the labeled examples first, and then gradually increase the dispersion or complexity weights to try to adapt to the unlabeled examples or to reduce complexity.

- Our evolutionary algorithm relies on parameter mutation and the improved one-point ordered crossover operator to evolve its candidate solutions. In our current implementation the strategy parameters, i.e. the operator probabilities and mutation widths, are empirically chosen in advance and not changed during learning. Especially in evolution strategies, it is common to evolve these strategy parameters, too. Although our implementation never needed more than a few hundred generations until convergence, such more advanced techniques might improve learning speed.

- Although "don't cares" are an important means for cutting down the complexity of rule bases, they have merely been used by the algorithms on our example. We suppose that the reason for this lies in a different meaning of "don't cares" in our learning algorithm and in pure supervised learning: in supervised learning, "don't cares" are generally introduced, if an attribute can be removed from the antecedent

without (substantially) degrading the classification performance. The MDL based measure introduces a "don't care" only where the assumption of a uniform distribution is more appropriate than that of a Gaussian distribution. If more compact rule bases are required, it might worth considering to label the points with the semi-supervised model first (with no or few "don't cares"), and then continue learning with the now labeled dataset and a higher weight on complexity punishment.

- On the given examples, our algorithms outperformed most of the compared supervised, unsupervised, or semi-supervised methods. However, we are well aware that there is "no such thing as a free lunch". There are always other situations, where our algorithms will perform inferior. As we have shown in Chapter 2.4, fuzzy classifiers use the same naïve assumptions as their probabilistic counterparts. If, for example, these assumptions are massively violated for the unlabeled examples, the algorithm will hardly be able to improve its results by learning from them.

  Cozman and Cohen (2002) investigated situations, where additional unlabeled examples even degraded the classification result in comparison to pure supervised learning from the labeled examples. They considered, however, randomly drawn and thus more or less prototypical labeled examples. Nevertheless, one should be aware of such problems as well.

  Hence, before semi-supervised techniques are applied, a domain expert should try to answer the following questions from his prior knowledge: Is the underlying process generative—i.e. do the classes influence the attributes, or do the attributes determine the class? How have the examples been labeled, and how representative are they? And, can the data distribution be appropriately described by the used classifier model? In this thesis, we tried to impart an understanding of semi-supervised learning that enables these questions to be intuitively answered. It is an issue for future research to further analyze and more formally define situations and circumstances under which the application of semi-supervised learning is promising.

Bibliography

A. Amar, N. T. Labzour, and A. M. Bensaid. Semi-supervised hierarchical clustering algorithms. In *Proc. 6th Scandinavian Conference on Artificial Intelligence (SCAI'97), Helsinki, Finland*, pages 232–239, 1997.

T. Bäck, F. Hoffmeister, and H. Schwefel. A survey of evolution strategies, 1991.

K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In D. A. Cohn, M. S. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems (NIPS-10)*, pages 368–374. MIT Press, Cambridge, MA, 1998.

A. M. Bensaid and J. C. Bezdek. Semi-supervised point prototype clustering. *Pattern Recognition and Artificial Intelligence*, 12(2):625–643, 1998.

A. M. Bensaid, L. O. Hall, J. C. Bezdek, and L. P. Clarke. Partially supervised clustering for image segmentation. *Pattern Recognition*, 29(2): 859–871, 1996.

M. R. Berthold. Mixed fuzzy rule formation. *International Journal of Approximate Reasoning*, 32:67–84, 2003.

J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

J. C. Bezdek, L. Hall, and L. Clarke. Review of MR image segmentation techniques using pattern recognition. *Medical Physics*, 20(4):1033–1048, 1993.

J. C. Bezdek, J. Keller, R. Krisnapuram, and N. R. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Handbooks of Fuzzy Sets. Kluwer Academic Publishers, Norwell, MA, 1999.

J. C. Bezdek and N. R. Pal. Some new indices for cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28 (2):301–315, 1998.

C. Blake and C. Merz. UCI repository of machine learning databases, 1998. (`http://www.ics.uci.edu/~mlearn/MLRepository.html`).

T. Blickle and L. Thiele. A comparison of selection schemes used in genetic algorithms. Technical report, ETH Zürich, Switzerland, 1995.

A. Bonarini. Evolutionary learning of fuzzy rules: Competition and cooperation. In W. Pedrycz, editor, *Fuzzy Modelling: Paradigms and Practice*, pages 265–284. Kluwer Academic Publishers, Norwell, MA, 1996.

C. Borgelt and R. Kruse. *Graphical Models – Methods for Data Analysis and Mining*. J. Wiley & Sons, Chichester, 2002.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

J. Burns, A. Hanson, and E. Riseman. Extracting straight lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:425–455, 1986.

B. Carse, T. C. Fogarty, and A. Munro. Evolving fuzzy rule based controllers using genetic algorithms. *Fuzzy Sets and Systems*, 80:273–293, 1996.

J. Casillas, O. Cordon, F. Herrera, and L. Magdalena, editors. *Trade-off between Accuracy and Interpretability in Fuzzy Rule-based Modelling*. Physica-Verlag, Heidelberg, 2002.

Z. Chi and H. Yan. ID3-derived fuzzy rules and optimal defuzzication for handwritten numeral recognition. *IEEE Transactions Fuzzy Systems*, 4 (2):24–31, 1996.

M. G. Cooper and J. J. Vidal. Genetic design of fuzzy controllers: the cart and jointed pole problem. In *Proc. 3rd IEEE International Conference on Fuzzy Systems (FUZZ-IEEE94)*, pages 1332–1337. IEEE Press, Piscataway, NJ, 1994.

O. Cordón, M. J. del Jesús, and F. Herrera. Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods. *International Journal of Intelligent Systems*, 3(10–11):1025–1053, 1998.

O. Cordón, M. J. del Jesús, F. Herrera, and M. Lozano. MOGUL: A methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. *International Journal of Intelligent Systems*, 14(2): 1123–1153, 1999.

O. Cordón, F. Herrera, L. Magdalena, and P. Villar. A genetic learning process for the scaling factors, granularity and contexts of the fuzzy rule-based system data base. *Information Sciences*, 136:85–107, 2001.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. J. Wiley & Sons, New York, 1991.

F. G. Cozman and I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. In *15th International Florida Artificial Intelligence Society Conference (FLAIRS 2002), Pensacola, FL*, pages 327–331, 2002.

R. Dara, S. C. Kremer, and D. A. Stacey. Clustering unlabeled data with SOMs improves classification of labeled real-world data. In *Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'02, Honolulu, HI)*, 2002. (published on CD-ROM).

D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(4):224–227, 1979.

A. Demiriz, K. P. Bennett, and M. J. Embrechts. Semi-supervised clustering using genetic algorithms. In *Proc. Artificial Neural Networks in Applications (ANNIE'99)*, 1999.

A. Demiriz, K. P. Bennett, and M. J. Embrechts. A genetic algorithm approach for semi-supervised clustering. *Journal of Smart Engineering System Design*, 4:35–44, 2002.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proc. 12th International Conference on Machine Learning (ICML'95), Lake Tahoe, CA*, pages 194–202. Morgan Kaufmann, San Mateo, CA, 1995.

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

J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(2):32–57, 1974.

C. Eitzinger. *Second-order and Nonsmooth Training Methods for Fuzzy Neural Networks*. PhD thesis, Universitaet Linz, 2001.

L. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. Rawlins, editor, *Foundations of Genetic Algorithms (FOGA 1)*, pages 265–283. Morgan Kaufmann, San Mateo, CA, 1991.

U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press / MIT Press, Menlo Park / Cambridge, USA, 1996.

G. Fung and O. L. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. Technical report, Data Mining Institute, 1999.

B. Gabrys and A. Bargiela. General fuzzy min-max neural network for clustering and classification. *IEEE Transactions on Neural Networks*, 11 (2):769–783, 2000.

B. Gabrys and L. Petrakieva. Combining labelled and unlabelled data in the design of pattern classification systems. In *Proc. EUNITE'2002*, pages 441–449, 2002.

I. Gath and A. B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:773–781, 1989.

J. Gebhardt and R. Kruse. A new approach to semantic aspects of possibilistic reasoning. In M. Clarke, R. Kruse, and S. Moral, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 151–159. Springer-Verlag, Berlin, 1993.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

D. E. Goldberg. A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. Technical report, Engineering Mechanics, University of Alabama, 1990.

D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1991.

D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989.

A. F. Gómez-Skarmeta and F. Jiménez. Fuzzy modeling with hybrid systems. *Fuzzy Sets and Systems*, 104(2):199–208, 1999.

A. González and R. Pérez. SLAVE: a genetic learning system based on an iterative approach. *IEEE Transactions on Fuzzy Systems*, 7(2):176–191, 1999.

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

S. Gottwald. Characterizations of the solvability of fuzzy equations. *Elektronische Informationsverarbeitung und Kybernetik*, 22:67–91, 1986.

S. Gottwald and W. Pedrycz. Solvability of fuzzy relational equations and manipulation of fuzzy data. *Fuzzy Sets and Systems*, 18:45–65, 1986.

J. J. Grefenstette, editor. *Genetic Algorithms for Machine Learning*. Kluwer Academic Publishers, Norwell, MA, 1994.

E. E. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proc. IEEE Conference on Decision and Control, San Diego*, pages 761–766. IEEE Press, Piscataway, NJ, 1979.

S. K. Halgamuge, A. Mari, and M. Glesner. Fast perceptron learning by fuzzy controlled dynamic adaption of network parameters. In R. Kruse, J. Gebhardt, and R. Palm, editors, *Fuzzy Systems in Computer Science*. Vieweg-Verlag, Braunschweig, 1994.

L. O. Hall, J. C. Bezdek, S. Boggavarapu, and A. Bensaid. Genetic fuzzy clustering. In *Proc. 13th International Conference of the North American Fuzzy Information Processing Society (NAFIPS94), San Antonio, Texas*, pages 411–415, 1994.

C. M. Higgins and R. Goodman. Learning fuzzy rule-based neural networks for control. In *Advances in Neural Information Processing Systems (NIPS-5)*, pages 350—357. Morgan Kaufmann, San Mateo, CA, 1992.

C. M. Higgins and R. Goodman. Fuzzy rule-based networks for control. *IEEE Transactions on Fuzzy Systems*, 2(1), 1994.

F. Hoffmann and G. Pfister. Evolutionary design of a fuzzy knowledge base for a mobile robot. *International Journal of Approximate Reasoning*, 17 (2):447–469, 1997.

J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, 1975. (reprint by MIT Press, Cambridge, MA, 1992).

J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. A. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems*, pages 313–329. Academic Press, New York, 1978.

H. Ishibuchi, K. Morioka, and I. B. Turksen. Learning by fuzzified neural networks. *International Journal of Approximate Reasoning*, 13(4):327–358, 1995.

H. Ishibuchi and T. Nakashima. Effect of rule weights in fuzzy rule-based classification systems. In *Proc. 9th IEEE International Conference on Fuzzy Systems, San Antonio*, pages 59–64, 2000.

H. Ishibuchi, T. Nakashima, and T. Murata. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29:601–618, 1999.

H. Ishibuchi, T. Nakashima, and T. Murata. Three-objective genetics-based machine learning for linguistic rule extraction. *Information Sciences*, 136 (1–4):109–133, 2001.

H. Ishibuchi, T. Nakashima, and I. B. Türksen. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy Sets and Systems*, 89(2):135–150, 1997.

A. Jain and R. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

J. Jäkel, L. Gröll, and R. Mikut. Automatic generation and evaluation of interpretable rule bases for fuzzy systems. In *Proc. CIMCA'99*, pages 192–197. IOS Press, Amsterdam, 1999.

C. Janikow. Fuzzy decision trees: Issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(2):1–14, 1998.

L. O. Jiminez and D. A. Landgrebe. Supervised classification in high-dimensional space: Geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 28(2):39–54, 1998.

C. Karr. Genetic algorithms for fuzzy controllers. *AI Expert*, 6(2):26–33, 1991.

J. Kinzel, F. Klawonn, and R. Kruse. Modifications of genetic algorithms for designing and optimizing fuzzy controllers. In *Proc. 1st IEEE Conference on Evolutionary Computation (ICEC94)*, pages 28–33. IEEE Press, Piscataway, NJ, 1994.

F. Klawonn and A. Keller. Fuzzy clustering with evolutionary algorithms. *International Journal of Intelligent Systems*, 13:975–991, 1998.

F. Klawonn and E. P. Klement. Mathematical analysis of fuzzy classifiers. In X. Liu, P. Cohen, and M. Berthold, editors, *Advances in Intelligent Data Analysis*, pages 359–370. Springer-Verlag, Berlin, 1997.

F. Klawonn and R. Kruse. Equality relations as a basis for fuzzy control. *Fuzzy Sets and Systems*, 54:147–156, 1993.

F. Klawonn and R. Kruse. Derivation of fuzzy classification rules from multidimensional data. In G. E. Lasker and X. Liu, editors, *Advances in Intelligent Data Analysis*, pages 90–94. The International Institute for Advanced Studies in Systems Research and Cybernetics, Windsor, Ontario, 1995.

F. Klawonn and R. Kruse. Constructing a fuzzy controller from data. *Fuzzy Sets and Systems*, 85:177–193, 1997.

A. Klose. Approaches to semi-supervised learning of fuzzy classifiers. In A. Günter, R. Kruse, and B. Neumann, editors, *Proc. 26th German Conference on Artificial Intelligence (KI 2003), Lecture Notes in Artificial Intelligence No. 2821*, pages 436–449. Springer-Verlag, Berlin, 2003a.

A. Klose. Extracting fuzzy classification rules from partially labeled data. *Soft Computing Journal*, 2003b. (available as "online first" at `http://www.springerlink.com`, paper version pending).

A. Klose and R. Kruse. Enabling neuro-fuzzy classification to learn from partially labelled data. In *Proc. IEEE International Conference on Fuzzy*

Systems, FUZZ-IEEE'02 (Honolulu, HI). IEEE Press, Piscataway, NJ, 2002. (published on CD-ROM).

A. Klose, R. Kruse, K. Schulz, and U. Thönnessen. Controlling asymmetric errors in neuro-fuzzy classification. In *Proc. ACM SAC'00*, pages 505–509. ACM Press, New York, 2000.

A. Klose, D. Nauck, K. Schulz, and U. Thönnessen. Learning a neuro-fuzzy classifier from unbalanced data in a machine vision domain. In *Proc. 6th International Workshop on Fuzzy-Neuro Systems FNS'99*, pages 23–32. Leipziger Universitätsverlag, Leipzig, 1999.

A. Klose and A. Nürnberger. Applying boolean transformations to fuzzy rule bases. In *Proc. 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, 1999. (published on CD-ROM).

A. Klose, A. Nürnberger, and D. Nauck. Some approaches to improve the interpretability of neuro-fuzzy classifiers. In *Proc. 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT'98)*, pages 629–633, 1998.

A. Klose, A. Nürnberger, D. Nauck, and R. Kruse. Data mining with neuro-fuzzy models. In A. Kandel, M. Last, and H. Bunke, editors, *Data Mining and Computational Intelligence*, pages 1–36. Physica-Verlag, Heidelberg, 2001.

A. Klose and J. Schneider. Semi-supervised induction of fuzzy rules applied to image segmentation. In *Proc. Joint 9th IFSA World Congress and 20th NAFIPS, Vancouver, Canada*, 2001. (published on CD-ROM).

T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

T. Kohonen. Learning vector quantization for pattern recognition. Technical report, Helsinki University of Technology, Finland, 1986.

T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Heidelberg, 1995. (3rd ext. ed. 2001).

R. Kothari and V. Jain. Learning from labeled and unlabeled data. In *Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'02, Honolulu, HI)*, 2002. (published on CD-ROM).

R. Krisnapuram and J. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98–110, 1993.

R. Kruse and A. Klose. Information mining with fuzzy methods: Trends and current challenges. In *Proc. Methods and Models in Automation and Robotics (MMAR'02), Szezecin*, pages 117–120, 2002.

U. V. Kulkarni, D. D. Doye, and T. R. Sontakke. General fuzzy hypersphere neural network. In *Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'02, Honolulu, HI)*, 2002. (published on CD-ROM).

L. I. Kuncheva. *Fuzzy Classifier Design*. Physica-Verlag, Heidelberg, 2000a.

L. I. Kuncheva. How good are fuzzy if-then classifiers? *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 30(4):501–509, 2000b.

T. Labzour, A. Bensaid, and J. Bezdek. Improved semi-supervised point-prototype clustering algorithms. In *Proc. International Conference on Fuzzy Systems*, pages 1383–1387, 1998.

C. Lanquillon. *Enhancing Text Classification to Improve Information Filtering*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Germany, 2001.

J. Larsen, A. Szymkowiak, and L. Hansen. Probabilistic hierarchical clustering with labeled and unlabeled data. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 6(2):56–62, 2002.

J. Liska and S. S. Melsheimer. Complete design of fuzzy logic system using genetic algorithms. In *Proc. 3rd IEEE International Conference on Fuzzy Systems (FUZZ-IEEE94)*, pages 1377–1382. IEEE Press, Piscataway, NJ, 1994.

K. Lütjen. BPI: Ein Blackboard-basiertes Produktionssystem für die automatische Bildauswertung. In G. Hartmann, editor, *Mustererkennung 1986, 8. DAGM-Symposium*, pages 164–168. Springer-Verlag, Berlin, 1986.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. L. Cam and J. Neyman, editors, *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, volume 1*, pages 281–297, 1967.

L. Magdalena. Adapting the gain of an FLC with genetic algorithms. *International Journal of Approximate Reasoning*, 17(2):327–349, 1997a.

L. Magdalena. Crossing unordered sets of rules in evolutionary fuzzy controllers. In *Proc. 7th International Fuzzy Systems Association World Congress (IFSA'97)*, pages 440–445, 1997b.

P. E. Maher and D. S. Clair. Uncertain reasoning in an ID3 machine learning framework. In *Proc. 2nd IEEE International Conference on Fuzzy Systems (San Francisco)*, pages 7–12, 1993.

E. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7 (2):1–13, 1975.

J. G. Marín-Blázquez and Q. Shen. Regaining comprehensibility of approximative fuzzy models via the use of linguistic hedges. In J. Casillas, O. Cordon, F. Herrera, and L. Magdalena, editors, *Trade-off between Accuracy and Interpretability in Fuzzy Rule-based Modelling*. Physica-Verlag, Heidelberg, 2002.

D. J. Miller and H. S. Uyar. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing Systems (NIPS-9)*, pages 571–577. MIT Press, Cambridge, MA, 1997.

D. J. Miller and H. S. Uyar. Combined learning and use for a mixture model equivalent to the RBF classifier. *Neural Computation*, 10:281–293, 1998.

M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1998.

T. Mitchell. *Machine Learning*. McGraw Hill, New York, 1997.

H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm I: Continuous parameter optimization. *Evolutionary Computation*, 1(2):25–49, 1993.

H. Narazaki and A. L. Ralescu. A synthesis method for multi-layered neural network using fuzzy sets. In *Proc. IJCAI-91: Workshop on Fuzzy Logic in Artificial Intelligence*, pages 54–66, Sydney, 1991.

D. Nauck. Adaptive rule weights in neuro-fuzzy systems. *Neural Computing and Applications*, 9:60–70, 2000.

D. Nauck, F. Klawonn, and R. Kruse. *Foundations of Neuro-Fuzzy Systems.* J. Wiley & Sons, Chichester, 1997.

D. Nauck and R. Kruse. NEFCLASS – a neuro-fuzzy approach for the classification of data. In K. M. George, J. H. Carrol, E. Deaton, D. Oppenheim, and J. Hightower, editors, *Proc. 1995 ACM Symposium on Applied Computing, Nashville*, pages 461–465. ACM Press, New York, 1995.

D. Nauck and R. Kruse. A neuro-fuzzy method to learn fuzzy classification rules from data. *Fuzzy Sets and Systems*, 89:277–288, 1997.

D. Nauck and R. Kruse. How the learning of rule weights affects the interpretability of fuzzy systems. In *Proc. 7th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'98), Anchorage*, pages 1235–1240, 1998.

D. Nauck and R. Kruse. Fuzzy classification rules using categorical and metric variables. In *Proc. 6th International Workshop on Fuzzy-Neuro Systems FNS'99*, pages 133–144. Leipziger Universitätsverlag, Leipzig, 1999.

D. Nauck, U. Nauck, and R. Kruse. NEFCLASS for JAVA - new learning algorithms. In *Proc. 18th International Conference of the North American Fuzzy Information Processing Society NAFIPS'99*, pages 474–476. IEEE Press, New York, 1999.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39 (2/3):103–134, 2000.

A. Nürnberger, C. Borgelt, and A. Klose. Improving naive bayes classifiers using neuro-fuzzy learning. In *Proc. 6th International Conference on Neural Information Processing (ICONIP'99), Perth, Australia*, pages 154–159, 1999a.

A. Nürnberger, A. Klose, and R. Kruse. Discussing cluster shapes of fuzzy classifiers. In *Proc. 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS'99), New York*, pages 546–550, 1999b.

A. Nürnberger, A. Klose, and R. Kruse. Analyzing borders between partially contradicting fuzzy classification rules. In *Proc. 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2000), Atlanta*, pages 59–63, 2000.

N. R. Pal and J. Biswas. Cluster validation using graph theoretic concepts. *Pattern Recognition*, 30(2):847–857, 1997.

J.-M. Park and H.-D. Yae. Analysis of active feature selection in optic nerve data using labeled fuzzy c-means clustering. In *Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'02, Honolulu, HI)*, 2002. (published on CD-ROM).

A. Parodi and P. Bonelli. A new approach to fuzzy classifier systems. In *Proc. 5th International Conference on Genetic Algorithms (ICGA93)*, pages 223–230. Morgan Kaufmann, San Mateo, CA, 1993.

W. Pedrycz. Algorithms of fuzzy clustering with partial supervision. *Pattern Recognition Letters*, 3:13–20, 1985.

W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 27(2):787–795, 1997.

D. T. Pham and D. Karaboga. Optimum design of fuzzy logic controllers using genetic algorithms. *Journal of Systems Engineering*, 1:114–118, 1991.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

I. Rechenberg. Cybernetic solution path of an experimental problem. Technical report, Royal Aircraft Establishment, Farnborough, Hants., UK, 1965.

I. Rechenberg, editor. *Evolutionsstrategie '94*. Frommann-Holzboog, Stuttgart, 1994.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11:416–431, 1983.

J. Rissanen. *Stochastic Complexity and Statistical Inquiry*. World Scientific, Singapore, 1989.

A. Rizzi, F. M. F. Mascioli, and G. Martinelli. Adaptive resolution min-max classifier. In *Proc. FUZZ-IEEE'98, Anchorage, Alaska*, pages 1435–1440, 1998.

J. D. Schaffer. Some effects of selection procedures on hyperplane sampling by genetic algorithms. In L. Davies, editor, *Genetic Algorithms and Simulated Annealing*, pages 89–130. Morgan Kaufmann, San Mateo, CA, 1987.

J. D. Schaffer and L. Eshelman. Real-coded genetic algorithms and interval schemata. In L. D. Whitley, editor, *Foundations of Genetic Algorithms (FOGA 2)*, pages 187–202. Morgan Kaufmann, San Mateo, CA, 1993.

R. Schärf, H. Schwan, and U. Thönnessen. Reconnaissance in SAR images. In *Proc. European Conference on Synthetic Aperture Radar*, pages 343–346, 1998.

B. von Schmidt and F. Klawonn. Fuzzy max-min classifiers decide locally on the basis of two attributes. *Mathware & Softcomputing*, 6:91–108, 1999.

B. von Schmidt and F. Klawonn. Construction of fuzzy classification systems with the Łukasiewicz-t-norm. In *Proc. 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS'2000)*, pages 109–113, 2000.

J. Schneider. *Konstruktive Exploration räumlicher Daten*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Germany, 2001.

J. Schneider and T. Strothotte. Constructive exploration of spatial information by blind users. In *Proc. 4th International ACM Conference on Assistive Technologies (ASSETS 2000), Arlington*, pages 188–192. ACM Press, New York, 2000.

H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM Journal on Optimization*, 2(1):121–152, 1992.

H. Schwan, R. Schärf, and U. Thönnessen. Reconnaissance of extended targets in SAR image data. In *Proc. European Symposium on Remote Sensing, Barcelona*, pages 164–171, 1998.

H.-P. Schwefel. Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik. Master's thesis, TU Berlin, 1965.

M. Seeger. Learning with labeled and unlabeled data. Technical report, Edinburgh University, 2001.

Y. Shi, R. Eberhart, and Y. Chen. Implementations of evolutionary fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 7(2):109–119, 1999.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Monographs on statistics and applied probability. Chapman and Hall, London, UK, 1986.

P. K. Simpson. Fuzzy min-max neural networks – part 1: Classification. *IEEE Transactions on Neural Networks*, 3(2):776–786, 1992.

P. K. Simpson. Fuzzy min-max neural networks – part 2: Clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):32–45, 1993.

A. Skabar. Augmenting supervised neural classifier training using a corpus of unlabeled data. In *Proc. 25th German Conference on Artificial Intelligence (KI-2002), Aachen, Germany*, pages 174–185, 2002.

S. F. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, Department of Computer Science, University of Pittsburgh, 1980.

M. Sugeno and T. Yasukawa. A fuzzy logic based approach to qualitative modelling. *IEEE Transactions on Fuzzy Systems*, 1(2):7–31, 1993.

T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 15(2):116–132, 1985.

P. Thrift. Fuzzy logic synthesis with genetic algorithms. In *Proc. 4th International Conference on Genetic Algorithms (ICGA91), San Diego, CA*, pages 509–513. Morgan Kaufmann, San Mateo, CA, 1991.

H. Timm. *Fuzzy-Clusteranalyse: Methoden zur Exploration von Daten mit fehlenden Werten sowie klassifizierten Daten*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Germany, 2002.

H. Timm and R. Kruse. A modification to improve possibilistic fuzzy cluster analysis. In *Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'02, Honolulu, HI)*. IEEE Press, Piscataway, NJ, 2002. (published on CD-ROM).

M. Valenzuela-Rendón. The fuzzy classifier system: Motivations and first results. In H. P. Schwefel and R. Männer, editors, *Proc. 1st International Conference on Parallel Problem Solving from Nature (PPSN I)*, pages 330–334. Springer-Verlag, Berlin, 1991.

G. Venturini. SIA: A supervised inductive algorithm with genetic search for learning attribute based concepts. In *Proc. European Conference on Machine Learning, Vienna*, pages 280–296, 1993.

A. Verikas, A. Gelzinis, and K. Malmquist. Using unlabeled data for learning classification problems. In L. C. Jain and J. Kacprzyk, editors, *New Learning Paradigms in Soft Computing*, pages 368–403. Physica-Verlag, Heidelberg, 2002.

M. Wall. GAlib: A C++ library of genetic algorithm components, 1999. (`http://lancet.mit.edu/ga/`).

C.-H. Wang, J.-F. Liu, T.-P. Hong, and S.-S. Tseng. A fuzzy inductive learning strategy for modular rules. *Fuzzy Sets and Systems*, 103:91–105, 1999.

L.-X. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2): 1414–1427, 1992.

D. Whitley. An overview of evolutionary algorithms: Practical issues and common pitfalls. *Journal of Information and Software Technology*, 43: 817–831, 2001.

Y. Yuan and M. J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69(2):125–139, 1995.

L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.