FUZZY SET COVERING

AS A NEW PARADIGM FOR THE

INDUCTION OF FUZZY CLASSIFICATION RULES

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Jacobus van Zyl, B.Sc. B.Eng. M.Sc.

aus Stellenbosch, Südafrika

Mannheim, 2007

2., corrected edition

Dekan: Prof. Dr. Matthias Krause, Universität Mannheim
Referent: Prof. Dr. Ian Cloete, International University in Germany
Korreferent: Prof. Dr. E. Badreddin, Universität Mannheim

Tag der mündlichen Prüfung: 16. Mai 2007

ABSTRACT

In 1965 Lofti A. Zadeh proposed fuzzy sets as a generalization of crisp (or classic) sets to address the incapability of crisp sets to model uncertainty and vagueness inherent in the real world. Initially, fuzzy sets did not receive a very warm welcome as many academics stood skeptical towards a theory of "imprecise" mathematics. In the middle to late 1980's the success of fuzzy controllers brought fuzzy sets into the limelight, and many applications using fuzzy sets started appearing.

In the early 1970's the first machine learning algorithms started appearing. The AQ (for A^q) family of algorithms pioneered by Ryszard S. Michalski is a good example of the family of set covering algorithms. This class of learning algorithm induces concept descriptions by a greedy construction of rules that describe (or cover) positive training examples but not negative training examples. The learning process is iterative, and in each iteration one rule is induced and the positive examples covered by the rule removed from the set of positive training examples. Because positive instances are separated from negative instances, the term separate-and-conquer has been used to contrast the learning strategy against decision tree induction that use a divide-and-conquer learning strategy.

This dissertation proposes fuzzy set covering as a powerful rule induction strategy. We survey existing fuzzy learning algorithms, and conclude that very few fuzzy learning algorithms follow a greedy rule construction strategy and no publications to date made the link between fuzzy sets and set covering explicit. We first develop the theoretical aspects of fuzzy set covering, and then apply these in proposing the first fuzzy learning algorithm that apply set covering and make explicit use of a partial order for fuzzy classification rule induction. We also investigate several strategies to improve upon the basic algorithm, such as better search heuristics and different rule evaluation metrics. We then continue by proposing a general unifying framework for fuzzy set covering algorithms. We demonstrate the benefits of the framework and propose several further fuzzy set covering algorithms that fit within the framework.

We compare fuzzy and crisp rule induction, and provide arguments in favour of fuzzy set covering as a rule induction strategy. We also show that our learning algorithms outperform other fuzzy rule learners on real world data. We further explore the idea of simultaneous concept learning in the fuzzy case, and continue to propose the first fuzzy decision list induction algorithm. Finally, we propose a first strategy for encoding the rule sets generated by our fuzzy set covering algorithms inside an equivalent neural network.

ABSTRACT

Im Jahre 1965 wurden unsharfe Mengen (auch Fuzzy-Menge) von Lofti A. Zadeh als Generalisierung zu scharfen (oder klassischen) Mengen eingeführt um die Fähigkeiten von scharfen Mengen zu erweitern in Richtung der Modellierung von Unsicherheit und Ungenauigkeit der Welt. Zu Anfangs waren Fuzzy-Mengen nicht besonders populär, da Akademiker einer Theorie von “unpräziser” Mathematik skeptisch gegenüberstanden. Mitte der 80er Jahre rückten Fuzzy-Controller in die allgemeine Aufmerksamkeit, gefolgt von vielen Anwendungen der Fuzzy-Mengen.

In den frühen 70er Jahren enstanden die ersten Maschinelles Lernen Algorithmen. Die AQ (von A^q) Familie von Algorithmen von Ryszard S. Michalski stellt ein Beispiel der Familie von Set-Covering Algorithmen dar. Diese Klasse von Lernalgorithmen induziert Konzeptbeschreibungen her mit Hilfe einer gierigen Regelkonstruktion, welche nur positive Training-Beispiele beschreibt. Der iterativ Lernproze leitet in jeder Iteration eine Regel her und löscht in Folge dessen die positiven Beispiele, welche mit der Regel erfasst wurden aus der Menge von positiven Traning-Beispielen. Der Term Separate-and-Conquer wurde gewählt um den Kontrast der Lernstrategie entgegen Entscheidungsbaum Induktion hevorzuheben, bei welcher eine Divide-and-Conquer Lernstrategie Anwendung findet. Hierbei bezieht sich Separate-and-Conquer auf die Tatsache, dass positive von negativen Instanzen separiert werden.

Diese Dissertation stellt Fuzzy-Set-Covering vor als eine leistungsfähige Regel-Herleitungs Regelinduktions Strategie. Wir verschaffen eine Übersicht über Fuzzy-Lernalgorithmen. Infolge dessen finden wir, dass nur wenige Fuzzy-Lernalgorithmen einer gierigen Regelkonstruktion verfolgen. Weiterhin verschaffen soweit keine Publikationen explizit den Link zwischen Fuzzy-Mengen und Set-Covering. Zuerst entwickeln wir theoretische Aspekte von Fuzzy-Set-Covering. Diese werden im folgenden angewendet auf den ersten Fuzzy Lernen Algorithmus, welcher Gebrauch macht von Set-Covering sowie explizit die partielle Ordnung für Fuzzy Klassifikation Regelinduktion berücksichtigt. Weiterhin recherchieren wir verschiedene Strategien um den zu Grunde liegenden Algorithmus zu verbessern, wie bessere Such-Heuristik und verschiedene Bewertungsfunktion um die Regeln zu evaluieren. Weiterhin schlagen wir ein allgemeines Rahmenwerk vor für die Fuzzy-Set-Covering Algorithmen. Wir zeigen die Vorteile für dieses Rahmenwerk auf zusammen mit weiteren Fuzzy-Set-Covering Algorithmen, welche auch in dieses Rahmenwerk eingepasst werden knnen.

Ein Vergleich zwischen Fuzzy und sharfen Regelinduktion wird hergestellt sowie Argumente für unscharfe entgegen scharfe Set Covering als Regelinduktions-Strategie vorgestellt. Wir zeigen auch die besseren Leistungen unseres Lernalgorithmus auf im Vergleich zu anderen Fuzzy-Regellerner sowie zu realen Daten. Weiterhin erforschen wir die Idee eines gleichzeitigen Konzept-Erlernens im Fall Fuzzy. Wir stellen den ersten Fuzzy Entscheidungsliste Induktions-Algorithmus weiterhin vor. Abschlieend erstellen wir eine erste Strategie um die Regelmengen, die von unserem Fuzzy-Set-Covering Algorithmus generiert werden mit einem äquivalenten neuronalen Netzwerk zu kodieren.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

In the modern world we are collecting huge amounts of data. In fact, mankind is collecting much more data than can be processed by humans. Several areas in computer science try to address this issue. In the data base field efforts are underway to process large databases in better ways, whereas computer vision, for example, focuses on the visualization and automatic processing of potentially vast amounts of image data. Machine learning methods pose one solution to automatic data analysis and processing. Machine learning is a field within computer science primarily concerned with the design of computer algorithms that improve their performance with experience [Mitchell, 1997]. Experience can be in the form of training examples, as in the case of artificial neural networks, for example, or may even be self-generated as in the case of reinforcement learning. Classification rules represent an important method of knowledge representation. Humans typically prefer reasoning by logical rules to decisions obtained from black box systems, since rule-based reasoning is comprehensible and can be validated, thereby improving confidence in the system [Duch et al., 2000; Andrews et al., 1995].

Many different methods have been proposed in the machine learning literature to induce classification rules for a concept from a set of positive and negative instances [Clark and Niblett, 1989; Michalski et al., 1986a; Theron and Cloete, 1996; Cendrowska, 1987; Quinlan, 1986]. This type of learning has commonly been called rule induction or concept learning. The induction methods induce different types of descriptions, for example decision trees [Quinlan, 1986] or propositional rules [Theron and Cloete, 1996]. The induction of fuzzy rules was proposed as an improvement over crisp rule induction.

Set covering is a successful rule induction methodology used in the crisp case. However, to date almost no algorithms employed *set covering* for the induction of *fuzzy* rules. In particular, we found no fuzzy set covering alogrithm using the properties of a partial order. In this dissertation we establishe set covering as a fuzzy classification rule induction methodology and show that set covering using a partial order can be used to create powerful fuzzy rule induction alogrithms. We propose several novel fuzzy set covering algorithms, and show that they are capable of inducing highly comprehensible rule sets with similar or better classification accuracy compared to previous fuzzy classifiers. We motivate the importance of this contribution in the next section.

1.1 Motivation

There are many reasons for wishing to extend crisp propositional classification rules to the fuzzy case. Fuzzy classification rules are more expressive and allow semantically more natural conditions to be described. They are also more comprehensible, because they allow symbolic knowledge to be formulated in a natural way using linguistic terms. The linguistic terms incorporated by fuzzy rules are defined by fuzzy sets. Members of a universe of discourse belong to a fuzzy set to certain membership degrees, which may be defined by an associated membership function. Thus fuzzy sets are commonly used to address the limitations associated with (over-) exact representations by providing support for vagueness, ambiguity and uncertainty in human understanding. A fuzzy set also addresses the problem of discretization of continuous (real valued) data. Most machine learning algorithms need to discretize continuous variables into a set of nominal attribute values defining ranges on the continuous domain which together cover a range, or all of the continuous domain. For example consider an integer variable age (in years) with domain $[0, 120]$, for which we wish to induce a condition that distinguishes "young" from "old" citizens. In the crisp case a condition must select a definitive cut-off point, e.g. age ≤ 25. We can now ask the question, what about a person 26 years of age, and how do we decide where to draw the line? Discretization does not cater to the gradual progression in the real world understanding of age from young to old, a characteristic which can be represented by a suitably chosen linguistic variable and linguistic terms for "young" and "old." The fuzzy set representation also increases the representational power of the description language. This extension also includes the crisp case (i.e. when there is no ambiguity in the source data) as a special case. Furthermore, the classification of an instance by a fuzzy rule is associated with a degree of certainty or confidence. There is no such degree of certainty for crisp rules, and an estimation of the confidence in the classification can at best be obtained from the performance of the rule on a labeled training set. Finally, contrary to the crisp case, in the fuzzy case the decision boundary of a rule need not be axis-parallel, and may even be non-linear. Because of the interpolation effect of fuzzy inference between overlapping, non-rectangular fuzzy sets, the classification boundary can be smooth, non-axis parallel.

For the reasons above, much work has already been devoted to the study of fuzzy rule learning systems [Guillaume, 2001]. The construction of a fuzzy system can be divided into two stages, parameter identification and structure identification [Pomares et al., 2002], also called knowledge base (or data base) and rule base induction, respectively [Casillas et al., 2000]. The knowledge base refers to the knowledge contained in the membership functions, whereas rule base identification concerns the issue of inducing good rules. The two stages may happen simultaneously [Peña-Reyes and Sipper, 2001; Kasabov et al., 1997] or sequentially [Hong and Chen, 2000]. However, much more attention is being paid to the parameter adjustment phase, as structure identification is a very complex task for which it is very difficult to obtain reliable procedures [Pomares et al., 2002]. There is also a definite difference between a tuning method and a learning method. Tuning methods optimise parameters in a predefined rule set, whereas learning methods perform a more elaborate search of the space of possible rule bases and/or knowledge bases, and do not depend on a predefined rule set [Cordón et al., 2004]. One successful structure identification approach is decision trees. The induction of fuzzy decision trees, as a generalization of "crisp"

decision trees, have been addressed in a number of papers [Cios and Sztandera, 1992; Yuan and Shaw, 1995; Guetova et al., 2002].

There are several aspects which make separate-and-conquer rule learners attractive [Fürnkranz, 1999]. Decision trees are often hard to understand, and Quinlan [1993a] noted that even pruned decision trees may be too cumbersome and complex to provide insight into the domain. Rivest [1987] also showed that decision lists with at most k conditions per rule are strictly more expressive than decision trees of depth k. Decision trees encode all the information contained in a whole rule set. Thus, to humans they are less comprehensible than rule sets as the whole tree must be considered at once, while only a single rule from a rule set need to be considered at a time. Another aspect is that there are certain concepts that cannot be represented by a concise tree [Cendrowska, 1987]—the restriction of decision trees to non-overlapping rules imposes a strong constraint on learnable rules [Fürnkranz, 1999]. This results in the replicated subtree problem—due to the fragmentation of the example space imposed by the restriction to non-overlapping rules, it often happens that the same subtree has to be learned at various places in a decision tree [Pagallo and Hassler, 1990]. Consider, for example, the following two rules,

$$a_1 \wedge b_3 \rightarrow \delta_1$$
$$c_5 \wedge d_1 \rightarrow \delta_1$$

If these two rules cover all instances belonging to class δ_1, then a single decision tree cannot represent the concept in this precise form. The root node of the tree must split on a single attribute, and there is no attribute common to both rules. Thus, if an extra attribute can be used to form a smaller decision tree that covers the training set, this tree will be preferred over others due to decision tree learners' bias of preferring shorter trees. This may not be significant overhead for a computer, however this unnecessary attribute may be costly to obtain, e.g. if the knowledge of a patient's temperature and blood pressure is enough to make a decision, then requiring an additional blood test is a serious consideration. If no such extra attribute exists, then the rule set obtained by first representing the concept as a decision tree and then extracting rules will be much more complex. If each attribute in the rule set above had five different values, the equivalent decision tree contains 73 leaf nodes and 90 edges. Thus the extracted rule set must be radically simplified to obtain the original concept (see Appendix F for a simplified example).

Set covering is a very successful methodology in the crisp case that applies the separate-and-conquer strategy to crisp inductive learning [Fürnkranz, 1999]. Set covering algorithms construct concept descriptions by the induction of a conjunctive expression which covers (or matches) a subset of the positive examples, removing the covered positive examples, and then repeating this process until all the positive examples are covered. Examples of set covering concept learners are the AQR[1] family of algorithms first introduced by Michalski, CN2[2], PRISM, and the Basic EXclusion Algorithm (BEXA) by Theron and Cloete [Michalski et al., 1986b; Michalski, 1969; Clark and Niblett, 1989; Cendrowska, 1987; Theron and Cloete, 1996].

The success of crisp set covering algorithms make set covering an attractive proposition for concept learning. Many other methods have been proposed for the induction of fuzzy rules, including fuzzy

[1] AQ stands for A^q algorithm.

[2] The C and N are the first letters of the algorithm's authors, P. Clark and T. Niblett

neural networks, fuzzy clustering algorithms, and fuzzy decision trees. However, there are very few fuzzy rule induction algorithms that apply greedy incremental rule construction. We found only two methods that use a separate-and-conquer strategy by removing covered instances from the training set, and the relation to the set covering methodology is not made explicit. This dissertation establishes set covering as an important fuzzy rule induction methodology, relates fuzzy to crisp set covering, and proposes fuzzy set covering using a partial order as a powerful new fuzzy rule induction strategy. Fuzzy sets are an intermediary between the symbolic and sub-symbolic knowledge representations—we can reason with fuzzy sets symbolically as linguistic terms, and we can also relate instances to fuzzy sets by a numerical value (membership degree). Thus, the introduction of fuzzy set covering for classification rule induction is an important contribution to machine learning since it narrows the gap between the symbolic and sub-symbolic knowledge representations, thereby bringing together the fuzzy and symbolic machine learning communities. Since the proof of the pudding is in the eating, we provide several experiments with fuzzy set covering algorithms to demonstrate that they perform extremely well with respect to rule set comprehensibility and classification accuracy, also compared to other state of the art algorithms. This dissertation also introduces the first ever use of simultaneous concept learning for fuzzy rules, thereby allowing the induction of fuzzy decision lists. We show that under the right conditions, decision lists can provide extremely compact but still accurate concept descriptions. In the next section we summarize the complete problem statement, and then we provide our objectives for achieving this goal.

1.2 Problem Statement

Rule induction is a very important subclass of machine learning methods since it provides insight into the model learned from data. Humans often reason using a set of rules, and thus a rule based classifier is intuitive and often more readily accepted than black box classifiers. A limitation of classic rule learners is their inability to deal with uncertainty and ambiguity. Rules based on fuzzy sets can address this issue. However, there are very few rule learners using fuzzy sets that apply an incremental rule induction strategy such as set covering (a very successful crisp rule induction strategy). In a study of several different fuzzy inference systems obtained from data, Guillaume [2001] concluded that the blind improvement of performance (e.g. by generating meaningless domain partitions purely for performance reasons) may degrade the interpretability of the induced fuzzy rules, and thereby invalidate the explicit assumption that fuzzy rules are by nature easy to interpret. Guillaume gives three conditions for interpretable rules, (a) fuzzy sets should be interpreted as linguistic labels, (b) the rule sets should be as small as possible, and (c) the rules should be incomplete rules. This means that for a high level of interpretability, the rules should be as general as possible, allowing each rule to cover a high number of instances, therefore resulting in a small rule set. In this dissertation we investigate fuzzy rule induction strategies capable of inducing accurate and, very importantly, comprehensible concept descriptions in the form of fuzzy classification rules. We are specifically not concerned with algorithms for function approximation or fuzzy control.

1.3 Objectives

We approach the probem stated above by persuing several objectives during the course of this dissertation,

- review and characterise other fuzzy concept learners and study examples of successful crisp rule learners,
- adapt the crisp learning strategy for the fuzzy case, and create a fuzzy learner using this strategy,
- summarize the characteristics of other fuzzy rule learners and compare with that of the new learner,
- investigate pruning and stop growth measures in the fuzzy case,
- develop a fuzzification method for continuous data,
- examine the influence of different rule evaluation mechanisms,
- analyse the learner's training parameters,
- investigate different description languages,
- measure the performance of the learner and compare to other fuzzy learners,
- investigate the induction of ordered rule sets (decision lists),
- investigate the possibility to develop a framework within which the learner and its various extensions can be understood and characterised.

1.4 Accomplishments

During the course of this dissertation we will demonstrate that all the objectives set out in the previous section are met. We will show that our proposed fuzzy rule learning methodology adheres to all three conditions for high interpretability as stated in the problem statement. We will also show that the proposed algorithms are at the same time capable of inducing highly accurate fuzzy rule sets. We list several contributions made by the dissertation.

1. We prove the feasibility of fuzzy set covering [Cloete and van Zyl, 2004b] as a new methodology for the induction of fuzzy classification rules [Cloete and van Zyl, 2004c].
2. We introduce FUZZYBEXA [Cloete and van Zyl, 2006], the fuzzy generalization and improvement of BEXA, and provide various efficiency measures and stop growth criteria applicable in the fuzzy case.
3. We introduce various rule evaluation functions [van Zyl and Cloete, 2004c], and show that the heuristics they employ play an important role in the induction process and the overall performance of the algorithm [Cloete and van Zyl, 2004a].
4. We provide preliminary results for a method for encoding the extracted fuzzy rules in an artificial neural network, and show that there is a one to one mapping from rule set to network [van Zyl and Cloete, 2004e].
5. We also provide a method for the induction of ordered fuzzy rule sets [van Zyl and Cloete, 2004f].

6. We propose FCF (Fuzzy Covering Framework), a new unifying framework based on fuzzy set covering for the induction of classification rules, where crisp set covering is included as a special case of fuzzy set covering [van Zyl and Cloete, 2006].

7. We provide four different specialization models for this framework: specialization by exclusion [Cloete and van Zyl, 2006], FUZZYSEEDSEARCH [van Zyl and Cloete, 2006], FUZZCONRI [van Zyl and Cloete, 2004d,a], and Fuzzy PRISM [van Zyl and Cloete, 2004b].

8. We present theoretical arguments for fuzzy set covering as opposed to crisp concept learning, and provide experiments comparing FCF to state of the art concept learners.

9. We provide a comparison between fuzzy set covering and other fuzzy learners on a set of benchmark data sets to demonstrate FCF's superior performance [Cloete and van Zyl, 2006].

10. Finally, we provide results on two real world applications, the detection of SPAM and the prediction of mortality in septic shock patients. FCF was able to outperform previously used methods convincingly, both with respect to classification performance and especially with respect to rule set comprehensibility.

1.5 Dissertation Outline

Since the goal of the dissertation is to develop a new fuzzy rule induction methodology, in Chapter 2 we provide a survey of fuzzy concept learners in general. We show that there are indeed very few algorithms that apply a greedy incremental rule construction strategy, and that none apply the set covering rule induction strategy using a partial order for the induction of fuzzy classification rules. In Chapter 3 we study the details of the succesful crisp rule learner BEXA. In the following chapter we develop fuzzy set covering for classification rule induction, and introduce FUZZYBEXA as an instantiation of such algorithms. FUZZYBEXA makes explicit use of partial ordering and lattice theory by arranging the concepts in its description language in a lattice, and drawing conclusions for induction strategies, efficiency, and pre-pruning.

FUZZYBEXA can be adjusted to the specific problem by a set of learning parameters, for example, the beam width. We provide an empirical evaluation of the influence of these parameters on the algorithm in Chapter 5. The following chapter discusses the importance of the rule evaluation function, introduces several new evaluation functions, and provides an empirical comparison. To establish FUZZYBEXA's uniqueness, Chapter 7 provides a comparison between FUZZYBEXA and other fuzzy concept learners at the hand of FUZZYBEXA's various characteristics. In the following chapter we propose FUZZCONRI, a fuzzy set covering algorithm that induces conjunctive fuzzy rules. Chapter 9 introduces FCF (Fuzzy Covering Framework), a general fuzzy set covering framework that allows the use of arbitrary specialization models employing different description languages. We also propose several different specialization models for the framework. In Chapter 11 we provide arguments for fuzzy set covering as opposed to crisp concept learning, and we present empirical proof that FCF outperforms comparable fuzzy learners on real world data. We present concluding remarks and directions for further research in Chapter 12.

CHAPTER 2

Fuzzy Concept Learners

2.1 Introduction

A typical empirical learning algorithm receives a set of examples, where each example is described by a vector of attributes, and each attribute consists of attribute values. In the case of a neural network, for example, all attributes have attribute values from the domain of real numbers. The task of a concept learner is to build a mapping from attribute values to concepts (or classes). ID3[1], for example, builds a decision tree to classify examples based on their attribute values, where attributes are tested at the node, and different branches represents different attribute values [Quinlan, 1986]. A concept learner is said to induce a concept description from a set of positive and negative instances (examples) of the concept. Covering algorithms are a class of concept learning algorithms that produce concept descriptions by iteratively generating concept descriptions, and at each step removing the positive instances covered (or classified) while retaining all negative instances, until all positive instances are covered. AQ15 and CN2, for example, induce a set of IF-THEN propositional rules, where the antecedent is built by a boolean expression of the attribute values, and the THEN part represents a concept [Michalski et al., 1986b; Clark and Niblett, 1989]. The expressiveness of the learner is determined by its description language. Learners with a more expressive description language can represent more complex concepts, and are therefore more powerful. The description language of BEXA [Theron and Cloete, 1996], for example, can be represented using Michalski's Variable Valued Logic System 1 (VL_1) [Michalski, 1972], and may contain internal disjunction.

Based on their method of learning, fuzzy concept learners can be divided into roughly seven major classes: (1) those employing greedy incremental rule construction, (2) those following a divide-and-conquer strategy, such as fuzzy decision trees, (3) those that use similarity search, (4) those that employ stochastic search, (5) those that derive a fuzzy partition, (6) those that build hierarchical systems, and (7) those based on gradient descent. There are also a few exceptions that cannot be put into any of these major classes, for example fuzzy Bayesian learning.

To date, no algorithms have been proposed that use the set covering approach to fuzzy rule induction in the way presented in this dissertation. This chapter provides a survey of existing fuzzy classification methodologies, and we review several algorithms of each as examples. The reviewed algorithms are rep-

[1] ID3 is an acronym for Iterative Dichotomiser 3.

resentative of their groups, and other similar algorithms follow the same strategy with only incremental differences. Since concept learners induce rules with different description languages, we first discuss different possible description languages in Section 2.2. We then continue to discuss greedy incremental rule learners, divide-and-conquer search, similarity search, stochastic search, fuzzy partitioning methods, hierarchical fuzzy systems, and gradient descent search, respectively in Sections 2.3 to 2.9. The discussion is thus ordered from most related to more distant work. Within each section, the survey of literature on the respective rule induction method is presented in chronological order (except where not appropriate), and more emphasis is placed on work introducing new concepts. Some significant methods that do not fall into any of the major categories are discussed in Section 2.10. Section 2.11 concludes the chapter.

2.2 Description Languages

Propositional rules are of the form

$$\text{IF } antecedent \text{ THEN } consequent \tag{2.1}$$

In Mamdani-type fuzzy controllers the antecedent takes the form

$$\eta_1 \text{ is } \mu_1 \text{ AND } \ldots \text{ AND } \eta_{n-1} \text{ is } \mu_{n-1} \tag{2.2}$$

where $\eta_1, \ldots, \eta_{n-1}$ are input variables. The consequent takes the form

$$\eta_n \text{ is } \mu_n \tag{2.3}$$

where η_n is the output variable. The antecedent of Takagi-Sugeno fuzzy rules have the same form as that of Mamdani rules, but the consequent is a linear function of the input variables [Takagi and Sugeno, 1985],

$$\text{IF } \mathbf{x} \text{ is } A_i \text{ THEN } y_i = \mathbf{a}_i^T \mathbf{x} + b_i, \qquad i = 1, 2, \ldots, M \tag{2.4}$$

where i is the rule index, $\mathbf{x} \in \mathbb{R}^n$ the antecedent, and $y_i \in \mathbb{R}$ the consequent. The antecedent fuzzy set of the i$^{\text{th}}$ rule is A_i,

$$A_i(\mathbf{x}) : \mathbb{R}^n \rightarrow [0, 1] \tag{2.5}$$

and is typically defined as an AND-operation by means of the product operator [Setnes, 2000]. A *complete* rule contains linguistic terms from all input domains, whereas *incomplete* rules do not.

A possibility rule involving fuzzy sets A and B is a special kind of fuzzy rule corresponding to the statement "the more X is A, the more possibility B is a range for Y," where X and Y are two variables [Dubois et al., 2002]. The possibility rule guarantees a certain lower bound $\pi(x, y)$ that (x, y) is an admissible instantiation of (X, Y),

$$\pi(x, y) \geq \min\{A(x), B(y)\} \tag{2.6}$$

Typically X and Y are input and output variables, respectively, and we are interested in the values of Y given X. Assuming $\pi(y|x) = \pi(x, y)$, Eq (2.6) is a lower bound to a conditional possibility distribution, i.e. given the value $X = x$ the possibility that $Y = y$ is lower bounded by $\pi(x, y)$.

FUZZYBEXA, to be introduced in Chapter 4, will express its rule antecedents in FuzzyAL, a fuzzy attributional logic that contains disjunction, conjunction, and internal disjunction. The rule consequent may either be a single variable, or a conjunction of output variables (see Section 4.3.1 for a detailed treatment of FuzzyAL). Most rule learners uses either Mamdani or Takagi-Sugeno type rules, and with the exception of fuzzy decision trees, few induce incomplete rules.

2.3 Greedy Incremental Rule Construction

Inductive rule learners induce IF-THEN classification rules from data by identifying features that empirically distinguish positive from negative training examples [Mitchell, 1997]. The class of fuzzy rule learners most related to our work induce rules by a greedy incremental rule construction process. These rule learners construct rules by expanding one or more rules incrementally at each step. The search process employs a greedy search by choosing the expansion leading to the (local) best improvement. There exist very few such rule learning algorithms (we have found only four to date) compared to the large number of fuzzy decision tree learners, fuzzy neural networks, and fuzzy genetic algorithms. In this section we review all the algorithms following this approach in relative detail. In Chapter 9 we will propose several specialization models for fuzzy rule induction, and we will provide a comparative discussion of the relevant algorithms surveyed in this section.

Wang *et al* proposed a learning strategy for incomplete rules using fuzzy information gain [Wang et al., 1999]. Inductive learning is generalized to learn a concept description $\tilde{R}$,

$$\tilde{\forall}\tilde{e} \in_\alpha \tilde{P} \Rightarrow \tilde{e}\tilde{\subset}_\alpha\tilde{R} \text{ and } \tilde{\forall}\tilde{e} \in_\alpha \tilde{N} \Rightarrow \tilde{e}\tilde{\not\subset}_\alpha\tilde{R} \tag{2.7}$$

where $\tilde{R}$ is a fuzzy concept description, $\tilde{\forall}$ is a linguistic quantifier such as "almost all" or "most," $\tilde{e}$ is a soft instance, $\tilde{P}$ and $\tilde{N}$ are sets of soft positive and negative instances respectively, and $\tilde{\subset}_\alpha$ and $\tilde{\not\subset}_\alpha$ are fuzzy relationship descriptors that denote α-*covered* and α-*not covered*, respectively. A soft (or fuzzy) instance is an instance that has class membership to linguistic terms in the range $[0, 1]$. Mamdani-type rules of the form of Eq (2.1) are learned, and an instance is said to be α-*covered* by a description $\tilde{R}$ if the rule strength is above a predefined threshold α, i.e. non-zero after applying an α-cut. This inductive strategy is borrowed from the crisp inductive algorithm PRISM [Cendrowska, 1987]. The algorithm receives a training set T, and learn descriptions for concepts $c_k \in C$, $k = 1, \ldots, K$, where C is a set of concepts. The rule learning begins by considering $c_k \in C$, and initialises the concept description $\tilde{R}$ to null. It then measures the fuzzy information gain for each linguistic term for the current class and chooses the term l that results in the highest gain. It then adds the term to the description, $\tilde{R} = \tilde{R} \wedge l$. The rule is then evaluated according to the fuzzy Bayes measure [Yuan and Shaw, 1995],

$$B(C_k|\tilde{R}) = \frac{\sum_{e\in T} \mu_{C_k}(e) \tau \mu_{\tilde{R}}(e)}{\sum_{e\in T} \mu_{C_k}(e)} \tag{2.8}$$

Table 2.1: The Fuzzy Beam Search algorithm.

```
    PROCEDURE FuzzyBeamSearch(maxdepth, w)
1     depth = 0
2     ruleset = a single rule with no conditions
3     REPEAT
4       FOR EACH r_i ∈ ruleset
5         FOR EACH attribute a_j not used in r_i
6           FOR EACH attribute value v_k
7             let r_ijk = r_i with [a_j = v_jk] added to the condition
8             specializations = specializations ∪ {r_ijk}
9             Compute a rule quality measure for r_ijk
10          END FOR
11        END FOR
12      END FOR
13      ruleset = best w rules among current rules
14      depth = depth + 1
15    UNTIL no rule created in this iteration outperforms rules
        of previous iterations OR depth = maxdepth
16    RETURN ruleset
    END PROCEDURE
```

where τ is a t-norm such as minimum. If the rule strength is above a user-defined level β then the rule is added to the rule set, all the α-*covered* instances removed from the training set, and the procedure repeated. When all the instances are α-*covered*, the procedure is repeated for the next class.

Fertig *et al* developed a fuzzy beam search induction algorithm [Fertig et al., 1999]. The algorithm is given in Table 2.1. It receives two parameters, $maxdepth$ is the maximal search depth, and w is the number of simultaneous paths explored, or beam width. A top-down induction is performed by adding conditions $[a_j = v_k]$ to existing rule antecedents, where a_j is an attribute that was not present in the rule antecedent before, and v_k is an attribute value from a_j. Thus, the description language allows the conjunction of different attribute-value pairs, and an attribute may only occur once, thus similar to the description language of CN2 [Clark and Niblett, 1989]. The procedure creates all possible specializations of the current rules, and evaluates them according to the evaluation function E,

$$E = \frac{|A \wedge C| - \frac{1}{2}}{|A|} \tag{2.9}$$

where A is the rule antecedent, C is the class attribute, and $|\cdot|$ is the summation of membership degrees over all examples with membership higher than a predefined value α. Once all the specializations are created, the best w rules are picked from both the specializations and previous rules. The search terminates when exactly the previous rule set is selected again after specialization, or when $maxdepth$ terms were added to the rule antecedent. The fuzzy sets may either have trapezoidal membership functions or assume crisp 0 or 1 membership degrees. The choice of the class predicted by rules are not specified in Table 2.1. The authors suggest to run the algorithm k times for a k-class problem, and let rules have each class as consequent in turn. In a two class problem, only the minority class is predicted, since such rules are more interesting. The majority class is then set as the default class.

In reference [Wang et al., 2003] Wang *et al* propose a fuzzy learning strategy based on the AQR inductive learner by Michalski [Michalski, 1969; Michalski et al., 1986a] called fuzzy AQR (FAQR). Wang *et al* state that their method focuses on learning fuzzy rules from soft training examples, and do not consider the acquisition of membership functions, which could be obtained by any of a number of methods. FAQR induces rules in disjunctive normal form, $C_1 \vee C_2 \vee \ldots \vee C_x$ where each C_i is a purely conjunctive expression. The fuzzy measurement function μ defines the degree to which a set of soft positive instances P are included by the concept description R,

$$\mu_{include}(R) = \frac{\sum_{e \in_\beta P} (\mu_P(e) \; \tau \; \mu_R(e))}{\sum_{e \in_\beta P} \mu_P(e)} \tag{2.10}$$

where τ is a t-norm, and $e \in_\beta P$ means that instance e *β-belongs* to P, i.e. $\mu_P(e) \geq \beta$. The fuzzy measurement function $\mu_{exclude}(R)$ defines the degree to which R excludes soft negative instances,

$$\mu_{exclude}(R) = \frac{\sum_{e \in_\beta N} (\mu_N(e) \; \tau \; (1 - \mu_R(e)))}{\sum_{e \in_\beta N} \mu_N(e)} \tag{2.11}$$

A concept description R is evaluated by

$$\mu_{\tilde{\forall}^+ \tilde{\forall}^-}(R) = \mu_{include}(R) \; \rho \; \mu_{exclude}(R) \tag{2.12}$$

where ρ is a union or an addition operator, and the subscript $\tilde{\forall}^+ \tilde{\forall}^-$ represents soft include positives and soft exclude negatives. If the minimum and maximum functions are used for the intersection and union operators, the costs for $\mu_{include}$ and $\mu_{exclude}$ are proportional to the numbers of positive and negative instances. The fuzzy measurement function for a complex C is defined similarly to that for R by simply replacing R with C in the equations above.

The FAQR learning strategy consist of two phases: generation and testing. The algorithm is shown in Table 2.2. In step 5 of the procedure *GenComplex* a description is specialized as follows. Let S be the set of all single term expressions (single attribute-value pairs) that *α-cover* the SEED but not the negative instance e, then the new C_{set} is the set $\{C_j \wedge S_k | C_j \in$ old C_{set} and $S_k \in S\}$. Finally, remove all the complexes in C_{set} that are subsumed by other complexes, i.e. if C_i subsumes C_j and $\mu_{\forall^+\forall^-(C_i)} \geq \mu_{\forall^+\forall^-(C_j)}$, remove C_j from C_{set}.

FS-FOIL is a fuzzy extension of FOIL (for First Order Inductive Learner) [Quinlan, 1990; Quinlan and Cameron-Jones, 1993], and thus uses first-order logic to induce a set of fuzzy predicates describing a goal predicate [Drobics et al., 2003]. The final induced predicate $\bar{A}$ is the disjunction of the set of predicates S,

$$t(\bar{A}(\mathbf{x})) = \bigvee_{A \in S} t(A(\mathbf{x})) = S_L(x, y) t(A(\mathbf{x})) \tag{2.13}$$

where S_L is the Łukasiwich t-conorm. The objective of the algorithm is to find a set of predicates with high significance and accuracy, where significance is defined as the common support of a predicate A and the goal predicate $\bar{C}$,

$$\text{supp}(A, \bar{C}) = \frac{|(A \wedge C)(X)|}{|X|} = \frac{1}{K} \sum_{i=1}^{K} T_L(t(A(\mathbf{x}^i)), t(\bar{C}(\mathbf{x}^i))) \tag{2.14}$$

Table 2.2: The Fuzzy AQR algorithm.

PROCEDURE Fuzzy AQR(P, N)
1 Let R be an empty set
2 While the rule set R does not α-*cover* all positive instances in P_β, do the following steps. Otherwise, return R.
3 Select the positive instance SEED that is not α-*covered* by R, and has the highest $\mu_P(e)$ among the positive instances.
4 Call the procedure GenComplex to generate C_{set}, a set of complexes that α-*cover* SEED and no negative instances in N
5 Select the complex C_{best} that has the highest $\mu_{\forall^+\forall^-}$ value in C_{set}.
6 Add C_{best} as an extra rule to the rule set R (i.e. $R = R \vee C_{best}$), and go to step 2.
END PROCEDURE

PROCEDURE GenComplex (SEED)
1 Let C_{set} be a set of single-selector complexes that α-*cover* SEED
2 While at least one complex in C_{set} α-*covers* a negative instances in N, do the following steps. Otherwise, return C_{set}.
3 Select C_j from C_{set} such that $\mu_{exclude}(C_j)$ has the smallest value
4 Select a negative instance e with the highest value $\mu_N(e)$ among those α-*covered* by C_j.
5 Specialize all complexes in C_{set} to not α-*cover* the instance e.
6 Remove the worst complexes from C_{set} until $|C_{set}| \leq \theta$, where θ is a user-defined parameter.
END PROCEDURE

and T_L is the Łukasiwich t-norm. Accuracy is defined as the confidence of predicate A with respect to $\bar{C}$, where $\text{conf}(A, \bar{C}) = \frac{\text{supp}(A,\bar{C})}{\text{supp}(A)}$, $\text{supp}(A) = \frac{1}{K}\sum_{i=1}^{K} t(A(\mathbf{x}^i))$, and thus

$$\text{conf}(A, \bar{C}) = \frac{|(A \wedge \bar{C})(X)|}{|A(X)|} = \frac{\sum_{i=1}^{K} t((A \wedge \bar{C})(\mathbf{x}^i))}{\sum_{i=1}^{K} t(A(\mathbf{x}^i))} \tag{2.15}$$

The FS-FOIL algorithm is shown in Table 2.3. It starts with the predicate $\top$ that always gives truth value 1. In each iteration an intermediate set of predicates P is expanded by forming conjunctions between members of P' and members from the set E, the set of atomic predicates that may be used for expansion. The best k predicates according to the information gain G are kept, where

$$G(A) = |(A \wedge \bar{C})(X)| \left(\log_2 \frac{|(A \wedge \bar{C})(X)|}{|A(X)|} - \log_2 \frac{|\bar{C}(X)|}{|X|} \right) \tag{2.16}$$

Pruning proceeds by removing all predicates $A \in P$ for which $\text{supp}(A, \bar{C}) < \text{supp}_{\text{min}}$, and all predicates in $B \in E$ for which $\text{supp}(A \wedge B, \bar{C}) < \text{supp}_{\text{min}}$. If a predicate A has at least a minimum significance and accuracy, it is added to S. The open nodes from O covered by A are then removed by replacing O with the intersection of the fuzzy set O and the fuzzy set of elements in X that have not been described by the predicate A. The induction process terminates when $|O|/K$ falls below a certain threshold, or when no new significant and accurate predicates can be found by expansion anymore. Although FS-FOIL is said to use first-order logic, only a single variable is allowed, and thus no variable binding is necessary. FS-FOIL does not perform substitution or $\theta-$subsumption, and thus cannot learn relations

Table 2.3: The FS-FOIL algorithm.

```
    PROCEDURE FS-FOIL(C̄, X, A)
1   S = ∅, P = {⊤}, E = A, open nodes O = C̄(A)
2   REPEAT
3     P′ = best k predicates of P according to G
4     P = expansion of P′ using E
5     Prune P and E
6     IF predicate p ∈ P is accurate and significant THEN
7       add p to S
8       remove nodes covered by A from open nodes O
9       P = {⊤}, E = A
10    END IF
11  UNTIL stopping condition
    END PROCEDURE
```

between attributes or recursive functions like FOIL [Quinlan, 1990; Quinlan and Cameron-Jones, 1993; Mitchell, 1997].

2.4 Divide-and-Conquer Strategies

The divide-and-conquer strategy is primarily implemented by decision tree learning, as exemplified by ID3 [Quinlan, 1986] and C4.5 [Quinlan, 1993a]. Decision trees, like most other symbolic machine learning methods, cannot handle continuous values in a natural way, and can at most suggest threshold values (boundaries) for decision making. The first fuzzy decision tree induction method was proposed by Chang and Pavlidis [Chang and Pavlidis, 1977]. Their method builds a binary fuzzy decision tree using a branch-bound-backtrack algorithm. Recently, fuzzy decision trees received much attention from several authors. F-ID3, for example, is a fuzzy counterpart of the ID3 algorithm, where a fuzzy version of the entropy function based on the cardinality of fuzzy sets is used instead of the classical entropy function [Cios and Sztandera, 1992].

Instead of using fuzzy entropy, Yuan and Shaw assume that membership functions are known a-priori, and induction proceeds based on the classification ambiguity. At each node in the tree the attribute that reduces the classification ambiguity most is chosen for expansion [Yuan and Shaw, 1995]. Zeidler and Schlosser suggested to compute membership functions for continuous domains [Zeidler and Schlosser, 1995]. First the domain is partitioned, and then trapezoidal membership functions are placed on each partition, where the corner points of two adjacent trapezoids are chosen to be the first instance values left and right of the partition division. In reference [Zheru and Hong, 1996] an algorithm is proposed to fuzzify the rules deduced from a decision tree induced by the ID3 algorithm. Crisp ranges are fuzzified by placing trapezoidal membership functions over the range, and choosing the end points to extend a user defined amount over the range. The attributes are then replaced by linguistic variables, and a two-layer perceptron is used as a defuzzification method.

Janikow suggests that fuzzy decision trees are usually used when the objective of learning is high

comprehensibility, rather than "best" fuzzy partitioning of the description space [Janikow, 1996; Janikow and Fajfer, 1999]. His fuzzy decision tree is induced using the ID3 method, but the information utility of individual attributes is evaluated using fuzzy sets. During classification, the inference routine must determine to what degree the example satisfies each of the leaves, and several inference methods have been suggested [Janikow, 1998]. In reference [Janikow, 1996] exemplar learning is used in the inference method. Exemplars are special examples selected from the training data, and used as a proximity measure, i.e. the decision procedure returns the class assigned to the "closest" exemplar. This relates the fuzzy decision tree induction method by Janikow to the class of similarity search methods discussed in Section 2.5.

Marsala used the star entropy to induce fuzzy decision trees applied to data mining [Marsala, 1998], where the star entropy is an extension of the Shannon entropy [Bouchon-Meunier et al., 1996]:

$$H_S^*(C|A_j) = -\sum_{l=1}^{m_j} P^*(V_{jl}) \sum_{k=1}^{K} P^*(C_k|V_{jl}) \log(P^*(C_k|V_{jl})) \tag{2.17}$$

where V_{jl} is the set of instances from the training set that has the l^{th} attribute value for attribute A_j, C_k is the set of instances from the training set belonging to concept c_k, and P^* is the Zadeh probability measure of fuzzy events [Zadeh, 1968]. If $\nu = \{x_1, \ldots, x_n\}$ is a fuzzy set and with each element x_i the classical probability of occurrence $P(x_i)$ is associated, ν is called a fuzzy event. The probability of the fuzzy event is then defined by,

$$P^*(\nu) = \sum_{i=1}^{n} \mu(x_i) P(x_i) \tag{2.18}$$

Membership functions are obtained beforehand by using a method based on the utilization of mathematical morphology theory [Marsala and Bouchon-Meunier, 1996]. In reference [Marsala, 2000] Marsala showed that classification by fuzzy decision trees is equivalent to generalized modus ponens. Generalized modus ponens is an extension of the classical modus ponens capable of handling fuzzy data,

Rule:	$P \Rightarrow C$
Observation:	P'
Deduction:	C'

Observing a value P' close to the antecedent P of a rule allows the construction of a consequent C' close to C. It was shown that given a measure of satisfiability, e.g. fuzzy subsethood, and a process of inference by means of a fuzzy decision tree, a continuity in the value of the decision is obtained relative to the values of the description. This continuity of fuzzy decision trees results in stability when classifying evolving observations.

In reference [Boyen and Wehenkel, 1999] a fuzzy decision tree was used with application to the security assessment of a power system. The induction method is restricted to binary trees, and consists of three steps. In the first step the tree is grown, and in the second it is post-pruned using cross-validation. In the third step a non-linear optimisation method is used to refit the parameters (the transition regions of tests, and the labels attached to leaf nodes). The derived fuzzy tree represents the function $\mu_c(o)$ which associates any object o of known attribute values to the output class c by a certain membership. The

value $\mu_c(o)$ is the average of the labels attached to the leaves, weighted by the degree of membership of the object to the corresponding fuzzy subsets.

In reference [Tsang et al., 2000] Tsang *et al* states that fuzzy decision trees have the advantage that they produce comprehensible knowledge, but that they are often criticized for poor learning accuracy. A hybrid neural network is proposed to refine a fuzzy decision tree. The fuzzy decision tree is augmented with several parameters, resulting in a weighted fuzzy decision tree. A weighted fuzzy decision tree contains three sets of parameters in each leaf node, the degree of truth of the classification corresponding to the leaf node (usually called the certainty factor), the degree of importance of each segment on the path from root to leaf node (called local weights), and the degree of importance of the leaf node's contribution to the consequent or classification (called the global weight). A weighted fuzzy decision tree is equivalent to a set of fuzzy production rules with local and global weights, as introduced by Yeung and Tsang [Yeung and Tsang, 1997]. An artificial neural network trained by an adapted backpropagation algorithm was used to adapt the tree weights. The weighted fuzzy decision tree significantly improved its accuracy compared to a normal fuzzy decision tree, while maintaining high comprehensibility.

In reference [Dong and Kothari, 2001] the fuzzy ID3 algorithm as proposed in [Cios and Sztandera, 1992] was extended to include a multi-step look-ahead method based on the smoothness of the class label surface. The smoothness is measured by calculating the cooccurance matrix. The algorithm jointly optimises the node splitting criterion, i.e. the information gain or gain ratio, and the classifiability of instances along each branch of the node. The look-ahead term $L(k)$ requires finding instances within a distance r from a given instance. These values, however, can be computed once beforehand. In reference [Mitra et al., 2002] different types of decision trees are evaluated using a quantative measure called the T-Measure. This measure incorporates both the compactness and performance of the decision tree. Various methods were also proposed for incorporating a fuzzy ID3 algorithm into a neural network [Singal et al., 2001; Mitra et al., 2002].

Guetova *et al* proposed a method for the incremental training of fuzzy decision trees [Guetova et al., 2002]. In reference [Abonyi et al., 2003] a binary decision-tree-based initialisation of fuzzy classifiers was proposed and used to select the relevant features and obtain an effective initial partitioning of the input domains for a fuzzy system. The decision tree initialized fuzzy classifier is reduced in an iterative scheme by means of similarity-driven rule-reduction. A genetic algorithm is used to remove redundancy while maintaining high accuracy. Olaru and Whenkel proposed soft decision trees which combine tree growing and pruning to determine the structure of the tree, and applied refitting and backfitting to improve its generalization performance [Alaru and Wehenkel, 2003]. In reference [Chiang and jen Hsu, 2002] fuzzy classification trees are proposed. Fuzzy classification trees integrate fuzzy classifiers with decision trees. In references [Safavian and Landgrebe, 1991; Murthy, 1998] an overview of decision trees in general is given.

2.5 Similarity Search

We group all fuzzy concept learners that employ some kind of distance or closeness measure under the term ”similarity search.” These include clustering methods, instance-based methods, and some SVM (Support Vector Machine) methods. We discuss each of these sub-groups separately.

There exist many different clustering techniques, and also many different fuzzy clustering techniques, of which the fuzzy c-means algorithm is probably the best known [Bezdek, 1981; Bezdek and Pal, 1992]. Fuzzy clustering has been employed for supervised rule learning in many different forms. Fuzzy clustering algorithms for pattern recognition suggested in the literature include Self-Organizing Maps, Fuzzy Learning Vector Quantization [Karayiannis and Bezdek, 1997], Fuzzy Adaptive Resonance Theory [Carpenter et al., 1991], Growing Neural Gas [Heinke and Hamker, 1998], and Fully Self-Organizing Simplified Adaptive Resonance Theory [Baraldi and Alpaydin, 2002; Baraldi and Blonda, 1999].

The aim of fuzzy clustering algorithms is to find a good prototype for each fuzzy cluster and suitable membership degrees for the data to each cluster. The simplest example is fuzzy c-means, first introduced in reference [Dunn, 1974]. Fuzzy c-means searches for spherical clusters of approximately the same size and uses Euclidean distance as a similarity measure. In [Leski, 2001] an ε-insensitive fuzzy clustering method based on Vapnik's ε-insensitive loss function is introduced. This algorithm is robust with respect to noise and outliers, and the fuzzy c-means clustering algorithm is obtained as special case.

In reference [Klawonn and Keller, 1997] a typical method for extracting Mamdani-type rules is described. The output space, which may be scalar or multidimensional, is partitioned using a clustering technique. Each data point is then assigned to a class based on the partitioning. Using the data from each class, prototype features are extracted for each class. The prototype features are then projected on the different dimensions of the input space. Using the projection, membership functions are extracted. These are then used to form the antecedent for an IF-THEN rule, where the consequent is the output space projection of the feature vector. Unsupervised clustering can also be employed to derive IF-THEN rules by projecting each cluster to the corresponding coordinate spaces. The projection to the i^{th} domain is obtained by taking the i^{th} coordinate of each data point in the cluster and assigning to it the membership degree of the original data point to the cluster. In this way a piece-wise linear membership function on the i^{th} domain is defined [Klawonn and Keller, 1997], and a Mamdani-type fuzzy controller [Sugeno and Yasukawa, 1993] is obtained. If triangular membership functions are used, each feature can be seen as a point in the instance space, where increasing distance from the feature implies increasing vagueness and is “less typical.”

Takagi-Sugeno (TS) type rules can also be inferred directly from data [Setnes, 2000]. The model identification process consists of two steps. In the first step the fuzzy antecedents of the rules are determined. In the second step the antecedents are kept fixed, and a least-squares estimation from data is applied to determine the consequent parameters of the rules ($\mathbf{a}_i^T$ and b_i in Eq (2.2)). The Gustafson-Kessel clustering method [Gustafson and Kessel, 1979] is often used in the identification of TS type rules, and employs an adaptive distance norm to detect clusters of different geometric shapes in the data set [Setnes et al.,

1998]. It defines the linear functions in Eq (2.2) on the basis of the eigenvalues and eigenvectors of the matrix C_i of cluster i, where C_i is a symmetric positive definite matrix obtained from the covariance of the clusters. In reference [Setnes, 2000] it is proposed to use the orthogonal least squares (OLS) method and to remove redundant or less important clusters during the clustering process, thereby extracting fuzzy rules that capture the data set features in a compact and transparent way. The clustering methods by Gustafson and Kessel [Gustafson and Kessel, 1979] and Gath and Geva [Gath and Geva, 1989] search for hyper-ellipsoidal clusters of varying size. The Gustafson Kessel method was modified by Klawonn and Kruse to obtain clusters whose axes are parallel to the coordinate axes [Klawonn and Kruse, 1995]. This technique is more flexible than fuzzy c-means and results in a smaller loss of information as compared to the standard methods in [Gustafson and Kessel, 1979; Gath and Geva, 1989]. It is also more computationally efficient since matrix inversion is avoided. In reference [Roubos et al., 2000] an initial rule base is derived using a modified Gustafson Kessel method, and then refined using a genetic algorithm. Hong and Lee's fuzzy expert system also employs a clustering method to extract rules from training data [Hong and Lee, 1996]. Berthold et al. [2002] proposed an interactive method based on neighborgrams to generate a set of clusters from data. The algorithm first computes neighborgrams for all patterns for a given class, and then computes the optimum depth, i.e. the depth for which a certain minimum purity is guaranteed. The algorithm then iteratively adds new clusters to the set of clusters by selecting the cluster with the highest coverage, where the coverage of a cluster with a given depth r is determined by how many positive patterns fall within its radius. All patterns that belong to the cluster are then removed from the training data, and the process iterated until the sum of all covered patterns exceeds a specified threshold. The authors state that the algorithm can be fuzzified by using fuzzy membership functions to model a degree of membership of a particular pattern to a cluster. In some domains it may also be preferable to fuzzify the class membership and to adapt the purity measurement to the fuzzy case.

Model-based approaches make assumptions about the structure of the system under consideration, and instance-based methods such as nearest neighbour classification rely on some kind of "closeness" or "representativeness" assumption [Dubois et al., 2002]. Dubois *et al* formulate a similarity based reasoning (SBR) hypothesis stating "similar problems have similar solutions." They use possibility rules in order to formalize the SBR hypothesis, "the more similar two situations are, the more possibly the corresponding outcomes are similar," and build a fuzzy case-based reasoning system [Dubois et al., 1998]. The approach is based on similarity guided extrapolation of observed cases, where already encountered cases are taken as evidence for the existence of similar cases. This evidence is expressed in terms of degrees of possibility assigned to the hypothetical cases.

In reference [Yin, 2004] a fuzzy inference system based on characteristic points (CPs) is proposed. Characteristic points are points in the input-output space chosen such that all outputs in the data set can be well approximated by the interpolation of some chosen outputs of the CPs, and are closely related to fuzzy rules. The main difficulty lies in finding suitable CPs for a given system. The method starts by mapping each data point to a fuzzy rule. The number of rules are then minimized using three separate procedures, gradient projection, a Gauss-Jordan based elimination method, and back-propagation fine tuning.

Fuzzy relational rules are learned in reference [Gaweda and Zurada, 2003]. First, the fuzzy c-means algorithm is used to cluster the output space into classes. For each class, an arbitrary α-cut is applied to the corresponding subset of instances, and then initial parameters for membership functions are extracted by calculating the class centre and spread in each dimension. Thereafter the Levenberg-Marquardt method [Levenberg, 1944; Marquardt, 1963; Moré, 1978] is used to optimise the membership functions, and the membership function parameters are then translated into the corresponding linguistic terms. This rule extraction method serves to initialise a relational fuzzy reasoning system that can be applied to function approximation.

Recently, support vector machines (SVMs) have been used for fuzzy modeling. SVMs use a hypothesis space of linear functions in a high dimensional feature space. They are trained with the statistical learning strategy introduced by Vapnik and co-workers in the early 1990's [Cristianini and Shawe-Taylor, 2000]. Recently, a fuzzy SVM for solving two-class classification problems was introduced [Tsang et al., 2003]. Fuzzy membership degrees are assigned to each training instance according to its membership degree to different classes. The fuzzy SVM generalizes the traditional SVM to a fuzzy one, and when all degrees of membership are equal for all training samples it degenerates to the traditional non-fuzzy SVM. The SVM by Chiang and Hoa uses a modified fuzzy basis function as it kernel function [Chiang and Hao, 2004]. The extracted support vectors are then used to build the fuzzy IF-THEN rules. In [Chen and Wang, 2003] fuzzy rules are extracted from the SVM hyperplanes. Fuzzy sets have also been used to build Fuzzy SVMs to reduce the effect of outliers on the SVM [Inoue and Abe, 2001; Abe and Inoue, 2002; Huang and Liu, 2002]. The clustering method of rule extraction was shown to be effective by its use in various application fields [Li and Elbestawl, 1996; Stutz and Runkler, 2002; Gedeon et al., 2002].

2.6 Stochastic Search

Simulated annealing [Kirkpatrick et al., 1983] and genetic algorithms [Goldberg, 1989] are examples of stochastic search methods. Simulated annealing was used to derive a fuzzy rule set by optimising Takagi-Sugeno rules with constant outputs [Guély et al., 1999]. Symmetric triangular membership functions were used and the midpoint and base length of the triangle adapted by using a simulated annealing technique. The adaptation of the triangle base was performed by perturbing the base width by a percentage of its initial width. This reduced the effect of membership functions either being very wide or very thin. The number of membership functions is set by an initial parameter, and the authors note that using too many or too few membership functions reduce the generalization performance. In some respects this method is similar to the clustering method for rule induction—both induce rules that contain all input variables in the antecedent and the formation of membership functions is tightly coupled with the method, i.e. it is not suited for problems where the membership functions are known a-priori. Simulated annealing was used to optimise an existing expert system that was fuzzified by hand [Garibaldi and Ifeachor, 1999].

Genetic algorithms (GAs) is a methodology loosely based on biological evolution [Goldberg,

1989], and have been widely used to evolve fuzzy rule sets [Cordón et al., 2004; Cordón, Herrera, Hoffmann and Magdalena, 2001; Herrera and Lozano, 1998; Hoffmann, 2004]. Much work has been done to combine the use of GAs and fuzzy logic [Alander, 1997]. GAs were used to optimise various aspects of fuzzy rule base systems, including evolving rule sets, optimising parameters in inference systems, and to obtain membership functions [Castro et al., 1993; Herrera et al., 1994; Kang et al., 2000; Wang and Bridges, 2000; Surmann, 2000; Cordón, Herrera and Villar, 2001]. Fuzzy logic techniques have also been used to model different GA components or adapt GA control parameters. The resulting GAs are termed Fuzzy GAs [Herrera and Lozano, 1998]. In reference [Castro et al., 1993] GAs are used to obtain fuzzy rules from examples. Rules are assumed to be of the form:

$$\text{IF } X_1 \text{ is } L_1 \wedge \ldots \wedge X_n \text{ is } L_n \text{ THEN } Y \text{ is } T$$

where T and L_i are linguistic labels. The membership functions were assumed to be triangular and the GA was used to adapt the size of the conjunction and the location membership functions. Real coded GAs were used to encode rules of the form

$$\text{IF } X_1 \text{ is } L_1 \wedge \ldots \wedge X_n \text{ is } L_n \text{ THEN } Y_1 \text{ is } T_1 \wedge \ldots \wedge Y_m \text{ is } T_m$$

where in this case trapezoidal membership functions were assumed [Herrera et al., 1994].

In reference [Ishibuchi et al., 1992] the input space is divided into K^M fuzzy subsets, where it is assumed that each of the M axes is divided into K partitions. Each subset describes one fuzzy IF-THEN rule, and the consequent is chosen such that the rule has maximum compatibility with the data set. In [Ishibuchi et al., 1995] a genetic algorithm is used to optimise the resulting rule set by minimizing the size of the rule set while still maintaining high classification accuracy. A similar approach, but with a different candidate rule generation scheme, was followed in reference [Ishibuchi and Yamamoto, 2004]. Here rule antecedents include a "don't care" term, and rules are screened by calculating their confidence and support before the evolution process. Only a certain number of "best" rules for each class are used to initialise the genetic algorithm. In reference [Nozaki et al., 1996] rules were given greater certainty when they classified data patterns correctly and less certainty when they classified them incorrectly. The certainty adjustment is controlled by the learning constants η_1 and η_2 for correct and incorrect classifications, respectively.

In reference [Luukka et al., 2001] the maximal fuzzy similarity in the generalized Łukasiewicz structure was used to build a classifier. This method requires a weight optimisation which is implemented using a genetic algorithm. Fuzzy CoCo (Cooperative Coevolution) employs a cooperative coevolutionary approach to fuzzy modeling [Peña-Reyes and Sipper, 2001, 2000]. Two cooperating species are defined—a database of membership functions and a conjunctive rule base. The two species are then evolved simultaneously. During fitness evaluation an individual establishes cooperation with one or more representatives of the other species. The fitness value depends on the performance of the fuzzy system obtained by the combination of the cooperating genes. An incremental version of the algorithm was also proposed [Peña-Reyes, 2003]. In references [Gómez et al., 2002; Dasgupta and González, 2001] GAs

have been used to evolve complete expression trees applied to network intrusion detection. The trees can represent arbitrary AND/OR rules.

In reference [Surmann, 2000] membership function shapes were assumed to be approximately Gaussian and used to evolve a fuzzy rule based knowledge representation [Surmann, 2000]. Genetic algorithms were also used to reduce and optimise a Takagi-Sugeno type rule base obtained by a fuzzy c-means clustering method [Roubos and Setnes, 2000]. Here rule based simplification is used together with a real coded GA to optimise a Takagi-Sugeno type rule base. Both rule structure and membership functions were obtained by the method published in reference [Angelov, 2003]. Genes contain both membership functions, in the form of the centre and spread of Gaussian type functions, as well as rule sets, where each possible rule is encoded by a positive integer number. In reference [Hoffmann, 2004] a boosting algorithm was used together with an iterative approach for classification rule learning. In this approach, one classification rule at a time is evolved, and the boosting mechanism reduces the weight of the correctly classified training examples, resulting in more focus on uncovered examples during the induction of the next rule.

2.7 Partitioning Methods

One of the earlier rule learning methods capable of inducing fuzzy rules directly from data was introduced by Wang and Mendel [Wang and Mendel, 1992]. The rule antecedent is a conjunction of input variables, and a output variable forms the consequent. The method requires that the input and output dimensions are partitioned into a set of fuzzy regions, where the partitions need not be of equal length. Triangular membership functions are then placed on the partitions such that the membership at the centre of a partition is unity, zero at the centre of the adjacent partition centres, and non-zero in between. Thus, the state space is effectively divided into a set of fuzzy hyperrectangles, where each hyperrectangle represents a possible rule. A fuzzy rule for each data point is then created such that the rule has maximum membership in all regions. This may result in conflicting rules, i.e. rules with the same antecedent but conflicting consequents. To resolve such conflicts the rule that maximizes D_j is chosen, where D_j is the product of all antecedent membership degrees,

$$D_j = \prod_i \mu_i(x_i) \tag{2.19}$$

where x_i is the i^{th} dimension of data point j that generated the rule. The authors also suggest that an expert can assign a degree of usefulness to each data point, and that this degree can be multiplied with D_j in the presence of noise. Linguistic rules obtained from experts may then be merged into the rule base and if conflicts occur the rule with the highest degree D_j is chosen. In the last step, the centroid defuzzification formula is used to defuzzify the output variable y,

$$y = \frac{\sum_{i=1}^{K} m^i_{O^i} \bar{y}^i}{\sum_{i=1}^{K} m^i_{O^i}} \tag{2.20}$$

where $\bar{y}^i$ is the centre of region O^i and K is the number of fuzzy rules in the rule base. The Cooperative Rules (COR) approach was suggested as an improvement to the general method [Casillas et al., 2000].

Instead of selecting the consequent with the highest importance degree, this method considers the possibility of using a rule that did not have the highest degree, but resulted in the best overall performance of the rule set. COR performs a combinatorial search among the candidate rules to obtain the set of consequents with best accuracy. This search may be brute force or using simulated annealing. In reference [Ma et al., 2000] the authors note that using the Wang Mendel method, at most two rules can be activated for any data point in a given dimension. Using this fact they simplify the defuzzification to a linear expansion of fuzzy rules, and then continue to use linear regression to construct a piecewise linear fuzzy system. The parameters obtained from the regression step is used to partition the domains, and triangular membership functions are used. This method reduces the number of fuzzy rules generated.

Hong and Chen improve on their earlier work [Hong and Lee, 1996] by first identifying relevant attributes and building initial membership functions before deriving decision rules [Hong and Chen, 1999]. Rule derivation is done by using a multidimensional decision table, and conflicts are resolved by choosing the rule with the highest degree. The method also allows for the simplification of the decision table by merging of adjacent table cells. In reference [Hong and Chen, 2000] the method was further developed by simplifying the intervals before the decision table is formed.

In reference [Ishibuchi and Nakashima, 2001] the authors assume that the antecedent linguistic values are given by domain experts for each of n domains. Thus, they assume that a fuzzy partition was made before rule induction, and that changing the membership functions would deteriorate the comprehensibility of the fuzzy IF-THEN rules. Their system forms fuzzy rules of the form

$$\text{IF } x_1 \text{ is } A_{j1} \text{ AND } \ldots \text{ AND } x_n \text{ is } A_{jn} \text{ THEN class } C_j \text{ with } CF_j,\ j = 1, 2, \ldots, N$$

where the certainty grade CF_j of rule j is computed and is usually a number on the unit interval $[0, 1]$. Similar to the method in reference [Ishibuchi et al., 1992, 1995], all possible rules are considered, but here rule conflicts are resolved by selecting the rule with the maximal product of compatibility grade and confidence factor,

$$\max_j \{\mu_j(\mathbf{x}) \cdot CF_j : j = 1, 2, \ldots, N\} \tag{2.21}$$

where $\mu_j(\mathbf{x})$ is the compatibility grade of the rule with pattern $\mathbf{x}$. The authors then continue to show how to estimate CF_j. The certainty grade assumes its maximal value when all compatible instances belong to the same class, and if no class is clearly dominant the certainty grade is small.

In reference [Pomares et al., 2002] a two-stage process is used for Takagi-Sugeno type rule induction. In the first stage parameter identification is performed to obtain good membership functions. The parameter identification method presented in reference [Pomares et al., 2000] functions by optimising the rule consequent using the Cholesky algorithm, and optimising rule antecedents using a steepest descent method. In the second stage structure identification is performed to obtain a good system topology. The process requires (1) the selection of input variables from a set of input candidates that are truly significant for the problem, and (2) an optimum partitioning of the input space, and therefore the minimum number of rules necessary for high accuracy. The authors note that this problem is certainly very complex, and that the only method in the literature to really address point (1) is combinatorial trees. They solve the structure identification problem by finding those input dimensions that with an increased num-

ber of membership functions reduce the error most rapidly. Extra membership functions are then placed in these domains such that the accuracy is maximally increased.

2.8 Hierarchical Fuzzy Systems

Yager proposed a hierarchical-type fuzzy model called a hierarchical prioritised structure (HPS) [Yager, 1998]. The structure allows exceptions to more general rules introduced at a higher level in the hierarchy, which themselves may be either terminal points or again rules with further exceptions. In an HPS, the output from the ith level is obtained by combining the output from the previous level with F_i by using an aggregation operator, where F_i is the result of applying the ith level rule base to input to the system. Yager first demonstrates how to build an HPS from rules obtained from experts, and then continues to induce a three layer HPS directly from data. The automatic construction of an HPS starts by initialising the model with some prior expectation of the system model, which could simply be a default rule such as output is X for all input. An observation is then presented and the output from the model calculated. If the output and expected output are close to each other within a threshold α, the data pair is considered to add no new information and is disregarded. If the data pair is not sufficiently explained by the bottom layer, an exception in the form of a point rule is formed and added to the middle layer, where a point rule is of the form

$$\text{IF } input \text{ is } x \text{ THEN } output \text{ is } y$$

With each rule a value M is associated, which is initialized as the strength of the exception P. The strength of the exception is computed as the inverse of the closeness of the calculated and expected output y^* and y respectively, $P = 1 - Close(y, y^*)$. Next, the M values of all the other rules in the middle layer are updated. Let (x, y) be the data pair from which a new point rule was formed, then rule i is updated as follows,

$$M_i' = M_i + Pe^{-\text{Distance}((x,y)-(x_i,y_i))} \tag{2.22}$$

Thus, the current model is modified by adding to the strengths of all other exceptions a value proportional to the strength of the current exception, modulated by its distance to the current exceptions. Next it is checked if the addition of the new rule caused an accumulation of exceptions that can be gathered into a new exception rule. Let $\hat{M}$ be the strength of the exception with the highest M value occurring at a point $(\hat{x}, \hat{y})$. If $\hat{M} \geq \beta$, a new rule of the following form is added to the top layer of the hierarchy,

$$\text{IF input is } about\ \hat{x} \text{ THEN output is } about\ \hat{y}$$

In the final step, rules in the middle layer that are now accounted for by the formulation of the new rule in the top layer are removed. The rule strength of each point (x, y) in the middle layer is adapted as follows,

$$M_i' = M_i + Pe^{-\text{Distance}((x,y)-(\hat{x},\hat{y}))} \tag{2.23}$$

and all point rules with $M' \leq 1 - \alpha$ are removed. Let $\hat{A}$ and $\hat{B}$ be the fuzzy subsets $about\hat{x}$ and $about\hat{y}$, respectively, then exception i in the middle layer is removed if $\min(\hat{A}(x_i), \hat{B}(x_i)) \geq \gamma$.

Gabriel and Berthold proposed a hierarchical rule system that arranges rules into different levels of precision [Gabriel and Berthold, 2003]. Rules in each level depend on only a few important features. Lower levels describe regions in input space with low evidence in the data, whereas higher levels describe rules with more support from the data. Each layer is built autonomously using the method proposed in reference [Berthold, 2003] and described in Section 2.10. A hierarchy of fuzzy rule sets are built by first inducing rules on a set of training data, and then extracting all rules with low relevance using an outlier-threshold θ_{outlier}. Next instances from the rule set are extracted using the filter parameter θ_{filter}. The procedure is iterated until all rules are above the outlier-threshold and the outlier model remains empty. The outlier models and the rule model of the last iteration form the model hierarchy. To make a classification the outputs of the rule systems at different levels are combined. Each rule model provides an output for a given input. Two inferencing approaches are possible, the fuzzy membership degrees of the different levels are summed, or the first rule that fires when traversing the hierarchy in a bottom-up manner is used. Further approaches to hierarchical fuzzy systems are proposed in [Holve, 1997, 1998; Shieh et al., 1999; Cordón et al., 2002; Moon G. Joo, 2001; Joo and Lee, 2002; Lee and Kim, 2002; Lee et al., 2003].

2.9 Gradient Descent

A variety of neural networks that work with fuzzy rules were studied in the literature [Mitra and Hayashi, 2000]. Fuzzy rules obtained by other means can be encoded into artificial neural networks [Nauck and Kruse, 1993; Kasabov et al., 1997; Frayman et al., 1999]. Typically fuzzy neural networks have an input layer, a conditional or functional membership layer, a rule layer and an output layer. Fuzzy neural networks have also been used as an oracle for querying in fuzzy inference systems [Mitra and Pal, 1995]. Fuzzy neural networks can be trained with an adapted form of back propagation, or by using an evolutionary approach [Kasabov, 2001a]. The methodology of extracting rules from a trained neural network for crisp rules [Cloete, 2000; Craven and Shavlik, 1994] has also been generalized to the fuzzy case [Matthews and Jagielska, 1995; Duch et al., 2000; Faifer et al., 1999]. This allows one to encode, refine, and extract fuzzy knowledge from artificial neural networks.

2.10 Other Methods

There are of course some fuzzy rule induction methods that do not clearly fall in any of the seven classes defined above. In this section we review a selection of interesting examples. Jain and Abraham compared four methods for the induction of fuzzy classification rules for a breast cancer data set [Jain and Abraham, 2003]. The first method generates a single fuzzy rule for each class by computing the mean and standard deviation in each dimension to build fuzzy membership functions. The second method partitions each input domain into twenty triangular membership functions, and then calculates a histogram using 0.5 as a threshold level. The histogram is used to build a rule for each class. The third method partitions the input domains into a grid, and generates a fuzzy rule for each partition. A method similar to that in reference [Ishibuchi et al., 1992] is used to calculate the consequents. The last method

is similar to the grid method, except that the membership functions are partitioned only on overlapping areas. For the Wisconsin Breast Cancer Data [Blake and Merz, 1998] the authors found the simple grid method to perform best, and the first method second best. They do note, however, that on real-world classification problems a single rule per class may not be sufficient.

In reference [Ishibuchi et al., 1997] Ishibuchi *et al* published a fuzzy Q-Learning algorithm, and used it in reference [Ishibuchi et al., 2003] to generate training examples for a market game example [Ishibuchi et al., 1997]. The learning process makes use of the methods in references [Ishibuchi et al., 1992; Nozaki et al., 1996] (also described above) and learns by iterative execution of games.

In [Carmona et al., 2004] a method is proposed to obtain compact rule sets by including exceptions in rules. The proposed method extends on that in [Castro et al., 1999] which learns rules of the form,

$$R^i : \text{IF } X_1 \text{ is } A_1^i \text{ AND } \ldots \text{ AND } X_n \text{ is } A_n^i \text{ THEN } Y \text{ is } LY^i$$

where each A_j^i is a set of labels, associated disjunctively with the jth input variable, and taken from the respective fuzzy domain $DX_j = \{LX_{j,1}, \ldots, LX_{j,p_j}\}$, and similarly LY^i is the label of the output variable taken from its fuzzy domain. Such rules are called *compound rules*. The method proceeds by initially creating a rule for each instance. The input domains are partitioned and triangular membership functions are used. The label of the fuzzy set that has maximal membership to the rule is then assigned to each domain,

$$A_j^i = \max_q \{\mu_q(x_j)\} \tag{2.24}$$

For each initial rule, if the rule is subsumed by any rule of the set of definitive rules, the method continues with the next rule. A rule R^i is *subsumed* by the rule R^k if for each $A_j^i \subseteq A_j^k$ and $LY^i = LY^k$. If the rule is not subsumed by any other rule, then for each label in each input variable, while the amplification of the rule is possible, it is amplified. Finally the new rule is stored in the set of definitive rules. Amplification consists of adding a label to one of the input domains where the label is not yet present. The amplification of rule R^i is possible if there exists no rule R^k such that each $A_j^i \subseteq A_j^k$ and $LY^i \neq LY^k$. Classification proceeds by converting an instance to a set of labels, where each input domain is represented by label of maximal membership. If an instance is subsumed by one or more rules with the same output, the example is classified as an element of the rule consequent class. If an instance is subsumed by more than one rule with different outputs, arbitration is done by choosing the rule with maximum degree of convenience,

$$\text{degree of convenience} = \min\{\phi_i(x_i)\} \tag{2.25}$$

where each ϕ_i is a membership function associated with the input variable X_i, and computed as a function of the labels present in the rule for the i^{th} domain. In [Carmona et al., 2004] the algorithm is extended by removing redundant rules during the initial rule formation, and then ordering the remaining rules in descending order according to their certainty degrees. Thus, arbitration during ambiguous classification is replaced by using the rule with the highest degree of certainty. Let R_i^j be the rule

$$\text{IF } X_1 \text{ is } LX_1 \text{ and } \ldots \text{ and } X_n \text{ is } LX_n \text{ THEN } Y \text{ is } LY_i$$

where each LX_l is a fuzzy set, then the degree of certainty $\omega(R_i^j)$ is calculated as,

$$\omega(R_i^j) = \frac{\beta(R_i^j) - \bar{\beta}(R_i^j)}{\sum_{k=1}^{q} \beta(R_k^j)} \tag{2.26}$$

where q is the number of labels in the output fuzzy domain, and

$$\beta(R_i^j) = \sum_{e \in T} \mu_{LX_1}(x_1) \times \ldots \times \mu_{LX_n}(x_n) \times \mu_{LY_i}(y) \tag{2.27}$$

and

$$\bar{\beta}(R_i^j) = \sum_{k=1, k \neq i}^{q} \frac{\beta(R_k^j)}{q-1} \tag{2.28}$$

where e is a training instance from the training set T with x in the input space and y in the output space. The algorithm described thus far may yield rules that allow different consequents to coexist in some fuzzy regions of the input space. In [Carmona et al., 2004] the authors note that a compound rule is equivalent to a set of "single" rules, i.e. rules with just one label associated with each input variable. For example, the compound rule

$$R^1 : \text{IF } X_1 \text{ is } \{S, L\} \text{ AND } X_2 \text{ is } \{M\} \text{ THEN } Y \text{ is } M$$

is equivalent to the single rules

$$R^2 : \text{IF } X_1 \text{ is } S \text{ AND } X_2 \text{ is } \{M\} \text{ THEN } Y \text{ is } M$$

and

$$R^3 : \text{IF } X_1 \text{ is } L \text{ AND } X_2 \text{ is } \{M\} \text{ THEN } Y \text{ is } M$$

The Fuzzy Rule Induction with Exceptions (FRIwE) algorithm uses this fact to form exceptions to rules in regions of the input space where two or more different consequents coexist. The method selects the best single rule in a compound rule, and adds the remaining conflicting single rules as exceptions to the rule. The new rule set with exceptions can then be further reduced by removing parts of the rule antecedent that are totally excluded due to the subset of rule exceptions. After rule reduction, some rules may now subsume and should be deleted. Thus another check for subsumption is performed. Finally, the rule antecedents and rule exceptions can be merged again. Two rule antecedents can be merged if they differ in only one domain and the rules have the same consequent. Similarly, two rule exceptions can be merged if the rule antecedent and consequents are the same, and the exceptions differ in only one domain. Merging consists of forming the union of the label sets in each domain.

Berthold proposed an algorithm for the formation of mixed fuzzy rules [Berthold, 2003]. If D_i is a dimension in feature space D, mixed fuzzy rules can handle continuous, granulated and nominal domains, i.e. $D_i \subset \mathbb{R}$, $D_i = \{\mu_j | 1 \leq j \leq m_i\}$ and $D_i = \{val_j | 1 \leq j \leq m_i\}$, respectively. With each mixed rule R two vectors are associated, $\vec{c}^{\,supp}$ describes the most general constraint (support region) and $\vec{c}^{\,core}$ describes the most specific constraint (core region). Constraints can also be true, i.e. they do not constrain the domain. An optimistic classification, possibly resulting in a heavy portion of overlap, can thus be made using the support constraint,

$$R(\vec{x}) = \bigwedge_{i=1}^{n} (x_i \in c_i^{supp}) \tag{2.29}$$

and a pessimistic classification, possibly resulting in a large area of the feature space not being covered, can be made using the core constraint,

$$R(\vec{x}) = \bigwedge_{i=1}^{n} (x_i \in c_i^{core}) \tag{2.30}$$

The fuzzy classification solves this problem by computing a degree of match for each rule and input pattern,

$$\mu(R, \vec{x}) = \min_{i=1}^{n} \{\mu_i \{c_i^{supp}, c_i^{core}, x_i\}\} \tag{2.31}$$

where the minimum was used as the t-norm, and the particular form of μ_i depends on the type of the domain D_i. With each rule there is also associated a weight w which counts how many patterns are explained by the rule, and an anchor $\vec{\lambda}$ which stores the original pattern that triggered the formation of the rule. Rule induction proceeds by performing a number of training epochs. During an epoch each pattern in the training set is considered. If there exists a rule that covers pattern $\vec{x}$ correctly, i.e. $\vec{x}$ lies within the support region of the rule, the core region of the rule is increased to cover $\vec{x}$ if it does not cover it yet, and the rule weight w is also increased. On the other hand if no rule correctly covers $\vec{x}$, a new rule is created with its support region covering the entire feature space and the core region covering only $\vec{x}$. The rule weight is set to one, and $\vec{\lambda} = \vec{x}$. In the next step the support regions of all rules that incorrectly cover $\vec{x}$ are reduced. For such a rule, if $\vec{x}$ lies outside the core region, the support region is decreased just enough not to cover $\vec{x}$, resulting in zero membership for $\vec{x}$ while still covering the remainder of its patterns. This is done by finding the component of $\vec{x}$ that does not lie in the rule's core region and results in a minimal loss of volume. If $\vec{x}$ lies inside the core region it is not possible to remove the conflict without influencing the covering of previous patterns. The same procedure as above is used to resolve the conflict. The volume of a rule is calculated as follows,

$$\text{vol}(R) = (\text{vol}(\vec{c}^{\,supp}), \text{vol}(\vec{c}^{\,core})) \tag{2.32}$$

where the volume of a constraint is calculated as

$$\text{vol}(\vec{c}) = \Pi_{i=1}^{n} vol(c_i) \tag{2.33}$$

and $\text{vol}(c_i) = 1$ if $c_i =$`true`, $\text{vol}(c_i) = \frac{c_i.max - c_i.min}{D_i.max - D_i.min}$ if D_i is numeric, and $\text{vol}(c_i) = \frac{|c_i|}{|D_i|}$ if D_i is granulated or nominal. At the end of each epoch, all rules are reset by setting the core region to λ and the weight value to zero. The training is complete if after an epoch no more changes are made to the rule set. Berthold proposes to address the problem of outliers by generating two models, one describing the overall behaviour and one describing patterns that were considered irrelevant or uninformative. The normalized rule weight parameter can be taken as a measure of the rule's relevance. Rules with low relevance are then extracted from the general model and used as a filter for a second training phase, thereby generating a new rule base that has less rules with higher significance. Thus only data from the training set that are not covered by the outlier model are used to construct the general model.

In [Botta and Giordana, 1993; Botta et al., 1993] Botta *et al* describes an improved version of ML-SMART [Bergadano et al., 1988] called SMART+, which makes use of fuzzy sets. SMART+ learns concept descriptions in First Order Logic using a combined deductive and/or abductive strategy. The

knowledge learned defines a structured classification theory, which can be described as a discrimination graph. As output, SMART+ generates a classification theory described in a Horn clause language L extended with functions, negation and numerical quantifiers. Let H_i and F_i be a set of concepts and instances, respectively, and let $H_i \subset H_j \subset H_0$, then a well formed formula in L has the form,

$$H_i \wedge \psi(t_1, \ldots, t_n) \stackrel{w}{\rightarrow} H_j \tag{2.34}$$

where $\psi(t_1, \ldots, t_n)$ is a logical formula stating a condition over terms $t_1, \ldots, t_n$, and w is a weight value. Eq (2.34) states that if an instance f, $f \in F_i$, is an instance of a concept h, $h \in H_i$, and $\psi(t_1, \ldots, t_n)$ is true of f, then $h \in H_j$. The value of the weight w is evaluated as the ratio between the number of correct instances matched and the number of total instances matched in F_0, and represents an estimation of the probability that the classification by ψ is correct. A formula $\psi(t_1, \ldots, t_n)$ contains predicates from a set P consisting of connectives $\wedge$ and $\neg$, and quantifiers "atmost", "atleast", and "exactly". Each predicate $p \in P$ defines a fuzzy set with a triangular membership function, defined by a set of parameters. The user specifies a range for each parameter in which to search for good values, as well as the granularity of the search process. SMART+ uses the *more-specific-than* concept of FOIL [Quinlan, 1990], but employs more specialization operators and more sophisticated strategies. The basic search strategy is combined with a reduction to subproblems technique, producing a structured classification theory. A subproblem consists of the pair (H, F), where H denotes a set of concepts and F a set of instances. The initial problem is given by the pair (H_0, F_0). The learned rules are organized into a *subproblem graph* G, where nodes are subproblems and edges, and labeled by logical formulas ψ_i. This structure is well suited to be applied to, for example, a diagnostic process based on multi-stage refinement. Within each subproblem $SP_i = (H_i, F_i)$, a *specialization tree* is built applying a similar strategy to that of FOIL. Different kinds of search strategies can be applied, including greedy, best-first, and beam-search. Within a specialization tree, nodes correspond to logical formulas and are specialized by appending new literals, resulting in specializations with non-negative information gain. The formulas are then evaluated using an evaluation function that contains both deductive and abductive components, with the user defining the importance of each,

$$s(\psi) = a\theta(\psi) + b\nu_T(\psi) \tag{2.35}$$

where a and b is given by the user, $\theta(\psi)$ measures the quality of the hypothesis ψ, and $\nu_T(\psi)$ is a measure that tries to capture how well ψ is "explained" by a given domain theory T. The function θ consists of two weighted components, the first part being the information gain obtained with respect to the immediate predecessor in the tree, and the second part an evaluation of the completeness and consistency of ψ. The best N nodes are then chosen as branches in the specialization tree, where $N = 1$ for a greedy search. If a formula ψ is found such that $\psi \rightarrow H_j$, with $H_j \subset H_i$, the instances belonging to the extension of ψ, denoted by F', $F' \subseteq F_i$, are declared as "solved." The focus of the search is then moved on to those formulas on the frontier of the specialization tree that still contain instances that are not yet solved, and the search stops when all instances are solved. At this point all instances belonging to the same concept set H_j are grouped in a new set F_j, and a new subproblem (H_j, F_j) is defined. The logical formulas with the same consequent set H_j thus form the disjunctive rule that connects the

subproblems (H_i, F_i) and (H_j, F_j) on the subproblem graph G. The whole process terminates when a subproblem contains only one concept in the concept set. The classification of an unknown instance will proceed along the graph G. Initially it is placed in (H_0, F_0). Then, depending on which logical formula connecting (H_0, F_0) to child nodes the instance satisfies, the instance is assigned to one of the subproblems. This process is iterated until a leaf subproblem node is reached.

In reference [Störr, 2002] a fuzzy generalization of the Naive Bayesian classification algorithm is introduced and applied to the classification of web layout preferences. Bayesian methods are based on the knowledge about the prior probabilities of alternative hypotheses and the probability of observing various data given the hypotheses. Naive Bayesian classifiers assume that attribute values are conditionally independent of the classification of the instance [Mitchell, 1997]. The requirements for the application described in [Störr, 2002] is that the classifier should support fast, incremental learning, learn from few examples, have a compact representation of the internal model, and allow the use of fuzzy attributes. The fuzzy case is described by letting the attributes be fuzzy, i.e. an example does not have exactly one value for each attribute, but has each value to a certain degree. The attribute names a linguistic variable, and each value corresponds to a linguistic term. Examples are also allowed to belong to each class to a certain degree. The learner defaults to the crisp case in the extreme with membership degrees either 1 or 0. In reference [Castro and Zurita, 1997] a fuzzy rule learner based on an assumption-based truth maintenance system (ATMS) [de Kleer, 1986; McAllester, 1990] is proposed. The algorithm induces fuzzy rules by finding the minimal node in the ATMS.

2.11 Summary

In this chapter we reviewed several fuzzy rule learning algorithms. We showed that the set of all fuzzy learning algorithms can be divided into seven major classes, excluding the class that contains all algorithms that do not fit in any of the seven other classes. We found only four greedy incremental rule construction algorithms, making this class the smallest of all. In the remainder of the dissertation we will introduce a new group of fuzzy rule learning algorithms. These algorithms will all follow a set covering approach to learning, and can be seen as a subclass of the class of greedy incremental rule construction algorithms.

CHAPTER 3

The BEXA Covering Framework

3.1 Introduction

The BEXA Covering framework was introduced by Theron and Cloete [Theron and Cloete, 1996; Theron, 1993]. It provides a framework for relating different set covering algorithms. The framework consists of three layers, a top layer implementing the set coving strategy, a middle layer implementing search heuristics, and a bottom layer implementing the specific evolutionary behaviour of the algorithm. Theron and Cloete showed that by adapting the evolutionary behaviour, i.e. by changing the bottom layer, several set covering algorithms, e.g. CN2 [Clark and Niblett, 1989], the AQR family of algorithms (specifically AQ15 [Michalski et al., 1986a]), PRISM [Cendrowska, 1987], GREEDY3 [Pagallo and Hassler, 1990] and Gray's algorithm [Gray, 1990], fit into the framework.

Theron Theron and Cloete [1996] also introduced the idea that the search starts with a most general description, which is then continuously specialized to form better descriptions. BEXA introduced a new method of specialization based on excluding attribute values rather than appending them to concept descriptions, and it was shown that this method performs particularly well [Theron and Cloete, 1996]. BEXA can, for example, find concept descriptions that other methods cannot, and is guaranteed to find the most general consistent concept descriptions. BEXA uses VL_1 (Variable Valued Logic System 1) as description language to express its concept descriptions [Michalski, 1972]. VL_1 is a very rich language, and allows internal disjunction, for example. The complete search space for any but the most trivial problems is therefore very large, and BEXA contains several search restrictions that prevent unnecessary search in uninteresting regions of the search space.

In the next chapter we will propose fuzzy set covering for inductive rule learning. We will also develop a fuzzy set covering algorithm, FUZZYBEXA, applying this methodology. FUZZYBEXA, as indicated by its name, is related to BEXA. It has the same hierarchical structure, and also makes use of the exclusion principle. However, we will show that FUZZYBEXA is far more than simply a "fuzzy version" of BEXA. To establish the background for the development of FUZZYBEXA, this chapter provides an overview of BEXA, and the layout of the remainder of the chapter is as follows. Section 3.2 reviews set covering concepts and Section 3.3 BEXA's description language. The set covering framework is discussed in Section 3.4, while BEXA's exclusion specialization model is treated in more detail in Section 3.5. Finally, Section 3.6 concludes the chapter.

3.2 Set Covering

This section introduces the terminology used for concept learning in the classical case. Let $A_1, \ldots, A_n$ denote attributes (referred to as variables in classification rules) with domains $D_1, \ldots, D_n$. Attributes are either nominal and take a finite set of unordered values (e.g. attribute *outlook* takes the values *sunny*, *cloudy*, *rainy*), or real valued taking values from a linearly ordered range (e.g. *temperature*). The attributes define the instance space $I = \langle D_1, \ldots, D_n \rangle$.

The goal of concept learning, which is a supervised learning method, is to find a description for each concept $c \in concepts$, where $concepts$ denotes the set of all concepts for which descriptions are desired. A concept is defined by a subset of instances (examples), and an instance is denoted by $\langle x, c \rangle$ where $x \in I$ and $c \in Concepts$. A subset P, $P \subseteq T$ of the training set T, $T \subseteq I$ contains the set of positive instances, i.e. all instances of the concept (or class) to be learned, while the subset N, $N = T - P$, holds the negative instances. Since we are using an inductive process to "infer" rules, we assume that the rules obtained from T generalize to unseen instances from $I - T$.

Set covering algorithms induce classification rules of the form, IF X THEN Y, where X is called the antecedent and Y the consequent. Thus for a set covering algorithm, rule antecedents are concept descriptions and rule consequents the concept. The set of all possible forms that the antecedent may assume is called the *description language* of the learner. The antecedent is often formed by the conjunction of several expressions, in which case it is also called a conjunction. We say that an instance i matches a conjunction c when the conjunction is true for this instance; we also say that c covers i. The connection from a description to the instances matched by it is made through its *extension* . Given a description c, the set of all instances from the set of instances S covered by c is called the extension of c in S, and is denoted by $X_S(c)$. Let $c = c_1 \vee c_2 \vee \ldots \vee c_n$ be a concept description given by the disjunction of several conjunctions. The concept description is *consistent* if it covers no negative instances, i.e. if the following holds true,

$$X_N(c_1) \cup X_N(c_2) \cup \ldots \cup X_N(c_n) = \emptyset \tag{3.1}$$

The concept description is *complete* if it covers all positive instances, i.e. if the following holds true,

$$X_P(c_1) \cup X_P(c_2) \cup \ldots \cup X_P(c_n) = P \tag{3.2}$$

The objective of a set covering algorithm is: given a training set T of instances, iteratively induce rules that cover the subset P, $P \subseteq T$, of positive instances, but not the disjoint subset N, $N = T - P$, of negative instances. Mitchell [1997, p. 275, 280] defines set covering (sequential covering) algorithms as follows. A set covering algorithm contains a subroutine for the induction of a single rule that distinguishes between the input sets of positive and negative instances by covering a large subset of positive instances while covering few or no negatives. After the induction of a single rule, the positive instances covered by the rule are removed from the training set, the negative instances retained, and the process iterated. The set covering approach followed by algorithms such as CN2 differ from divide-and-conquer type search followed by decision tree learners such as ID3. "The key difference occurs in the most primitive step in the search. At each step ID3 chooses among alternative *attributes* by comparing the

partitions of the data they generate. In contrast, CN2 chooses among the set of *attribute-value* pairs, by comparing the *subsets* of data they cover" [Mitchell, 1997, p. 280]. Thus, we can define a set covering algorithm as follows.

Definition 3.2.1 *A rule induction algorithm employing the set covering approach to concept learning has the following key characterstics:*

1. *a single rule is induced at each step, and only the positive instances covered by the rule are removed from the training set, and*
2. *the induction of a single rule proceeds by iteratively choosing among alternative attribute-value pairs, and comparing the subsets of data they cover.*

Ideally, a concept description should be maximally accurate, maximally general, and minimally complex. Accuracy refers to the performance of the description as measured by its ability to classify instances correctly, whereas the generality refers to the ability to correctly classify instances not in the training set, i.e. from the set $I - T$. In the case of a covering algorithm, complexity refers to both the complexity of the individual rules as well as to the number of rules in the rule set. In the remainder of this chapter we present a description of BEXA, which applies set covering in the crisp case. For a review of other crisp set covering algorithms we refer the reader to Appendix A.

3.3 BEXA's Description Language

BEXA's description language allows antecedents to be expressed in VL_1 [Michalski, 1972], and induces concept classification rules of the form,

$$\text{IF } antecedent \text{ THEN } consequent$$

The antecedent is a disjunction of conjunctions, and following the VL_1 convention, each conjunction can be *internally disjunctive*. For example, given the data set in Table 3.1, consider the antecedent of the classification rule at the bottom the table, where "THEN" is indicated by the symbol $\rightarrow$. This expression is a disjunction of two *conjunctions*,

$$[outlook = sunny \vee cloudy][temp = 13]$$

and

$$[humidity = normal][temp = 28]$$

Every disjunctive expression can be written as a set of equivalent production rules, e.g.

$$\text{IF } [outlook = sunny \vee cloudy][temp = 13] \text{ THEN } weights$$

$$\text{IF } [humidity = normal][temp = 28] \text{ THEN } weights$$

The conjunction $[outlook = sunny \vee cloudy][temp = 13]$ implicitly assumes $[outlook = sunny \vee cloudy] \wedge [temp = 13]$, but omits the $\wedge$ symbol for brevity. The conjunction $[outlook = sunny \vee$

Table 3.1: A crisp learning problem and an example of a VL_1 concept description.

```
@relation sport

@attribute outlook  {sunny, cloudy, rainy}
@attribute temp     real
@attribute humidity {humid, normal}
@attribute wind     real
@attribute activity {volleyball, swimming, weights}

@data
sunny,  30, humid,  26, swimming    ;1
sunny,  26, normal, 5,  volleyball  ;2
cloudy, 28, normal, 12, swimming    ;3
cloudy, 23, normal, 14, volleyball  ;4
rainy,  28, normal, 20, weights     ;5
cloudy, 13, humid,  24, weights     ;6
rainy,  10, normal, 10, weights     ;7
cloudy, 12, normal, 14, volleyball  ;8
sunny,  33, humid,  22, swimming    ;9
sunny,  13, normal, 33, weights     ;10
sunny,  31, humid,  0,  swimming    ;11
cloudy, 20, normal, 16, volleyball  ;12
sunny,  18, normal, 28, weights     ;13
cloudy, 21, normal, 28, weights     ;14
rainy,  9,  humid,  31, weights     ;15
sunny,  15, normal, 7,  volleyball  ;16
```

An example of a VL_1 concept description:

$$[outlook = sunny \vee cloudy][temp = 13] \vee$$
$$[humidity = normal][temp = 28] \rightarrow weights$$

$cloudy][temp = 13]$ consists of two *conjuncts*, of which $[outlook = sunny \vee cloudy]$ is internally disjunctive. This internally disjunctive expression is interpreted as follows: the value of an instance for attribute *outlook* is an element of the set $\{sunny, cloudy\}$, i.e. $outlook \in \{sunny, cloudy\}$. When the sets of nominal values of attributes are disjoint, we can omit the attribute name and write, for example, $[sunny, cloudy][temp = 13]$. BEXA's syntax also allows the negation of nominal values, e.g. $[not\ rainy]$, or equivalently $[\neg rainy]$, for $[sunny, cloudy]$. The consequent of a rule simply names the concept, and has the syntax *class attribute = nominal value*. Like most other crisp machine learning algorithms, BEXA caters for linearly ordered attributes by learning ranges, e.g. $21 < temp \leq 26$.

BEXA requires the creation of the *most general conjunction* (*mgc*). This conjunction should cover all instances in the instance space. In VL_1, this requirement translates to the conjunction that is formed by the conjunction of the disjunction of all attribute values for each attribute. For the learning problem in Table 3.1 the *mgc* is the conjunction of $[sunny, cloudy, rainy][humid, normal]$ with the disjunction of all temperature ranges and the disjunction of all wind strength ranges that can be formed from the instances in the training set.

Table 3.2: BEXA's set cover procedure.

```
     PROCEDURE Cover-P(T, beamwidth, concepts)
1      ruleset = ∅;
2      FOR EACH concept c_i ∈ concepts DO
3        P = instances in T belonging to concept c_i; N = T − P;
4        REPEAT
5          bestconj =FindBestConjunction(P, N, beamwidth);
6          IF bestconj ≠ NULL THEN
7            Add the rule "IF bestconj THEN concept = c_i" to ruleset;
8            P = P − X_P(bestconj);
6          END IF
9        UNTIL (P = ∅) OR (bestconj = NULL);
10     END FOR
11     RETURN ruleset;
     END PROCEDURE
```

3.4 The Set Cover Framework and BEXA

BEXA is a unifying framework that can be used to relate different crisp set covering algorithms. BEXA consists of three layers. The top layer implements BEXA's set covering behaviour, the middle layer implements BEXA's search heuristics, and the bottom layer implements the specific conjunction specialization behaviour of the algorithm under consideration. This section describes the three layers of the framework.

3.4.1 The Set Covering Layer

Table 3.2 shows BEXA's set cover procedure. The procedure *Cover-P* receives a training set of instances as input, and iteratively learns classification rules for each concept c_i. The training set is split into two disjoint sets, one containing the positives instance, P, and the other the negative instances N. The procedure *FindBestConjunction* is then called repeatedly to learn a description for a subset of the positive instances. If a description is found, a rule with the current concept as consequent and the description as antecedent is formed. The instances covered by the rule are removed from the set P, but the negative set N remains unchanged. The procedure continues until all positive examples are covered (P is empty) or a good description could not be found ($bestconj$= NULL).

The extension of the conjunction $bestconj$ in the set P, $X_P(bestconj)$, denotes the subset of instances of P which are covered (matched) by the description $bestconj$. Note that the set N for learning a particular concept (or class) remains unchanged and only P is split up further, hence the term separate-and-conquer. In decision tree learning, in contrast, the entire training set is split based on the examples matched by the conjunction.

Table 3.3: BEXA's FindBestConjunction procedure.

```
     PROCEDURE FindBestConjunction(P, N, beamwidth)
1      bestconj =NULL;
2      specializations = {the mgc for BEXA, or the constant true
       with X_P = P and X_N = N};
3      WHILE specializations ≠ ∅ DO
4        specializations =GenerateSpecializations(P, N,
         specializations, beamwidth);
5        FOR each conjunction c ∈ specializations DO
6          IF c is significant according to the significance test
7            AND c is better than bestconj according to the evaluation
             function THEN
8            bestconj = c;
9        Remove from specializations all the conjunctions that cover
         no negative instances or
10         that satisfy the additional stop-growth test;
11       Retain in specializations only the beamwidth best conjunctions
12     END WHILE
13     IF the evaluation function value for bestconj is the same or worse
       than that of the complete training set THEN
14       RETURN NULL;
15     ELSE
16       RETURN bestconj;
     END PROCEDURE
```

3.4.2 The Search Heuristics Layer

The procedure *FindBestConjunction* is shown in Table 3.3. Its purpose is to induce a concept description that covers as many instances from P and as little instances from N as possible. The best description found during the search process is maintained in the variable $bestconj$. It starts by creating the *mgc*, which covers all instances in the instance space, and thus also the sets P and N. The representation of this conjunction can be either the constant TRUE, or a concept description in the description language of the relevant algorithm that covers all possible instances. The *mgc* is added to the set $specializations$, which maintains the set of candidate specializations at each step of the induction process.

The bottom layer routine *GenerateSpecializations* is then invoked to obtain the set of specializations of the concept descriptions in $specializations$. Each description in the resultant set is considered in turn. If a description is significant according to a significance test, and its evaluation according to an evaluation method is better than that of the previous best conjunction, then it replaces the current best conjunction. The evaluation method used in BEXA was the Laplace estimate,

$$L(c) = \frac{|X_P(c)| + 1}{|X_P(c)| + |X_N(c)| + \#concepts} \tag{3.3}$$

where $\#concepts$ is fixed to two classes, positive and negative. Descriptions can be tested for significance by comparing their distribution to the distribution of the complete training set. This can be done

by, for example, using the log-likelihood ratio test with respect to the distribution of the concept in the data set [Kalbfleisch, 1979],

$$S(c) = 2(f_P \log \frac{f_P}{e_P} + f_n \log \frac{f_N}{e_N}) \tag{3.4}$$

where f_P and f_N are the observed frequency distributions of the positive and negative instances that match c, respectively, and e_P and e_N are the expected frequency distributions of the positive and negative instances, respectively. The statistic is distributed approximately as χ^2 with one degree of freedom [Clark and Niblett, 1989], and provides a measure of significance—the lower the score, the more likely that the apparent regularity is due to chance.

In the next step, conjunctions that cover no negative instances are removed, since specializing them further conflicts with the objective of finding maximally general concept descriptions. Other pre-pruning steps can be inserted here. One possibility is to stop specializing conjunctions whose distribution is not significantly different from their immediate predecessor. This test can again be implemented by using the log-likelihood ratio test as in Eq (3.4), where now f_P and f_N are the observed frequency distributions of the positive and negative instances that match c, respectively, and e_P and e_N are the frequency distributions of the positive and negative instances that match c', the predecessor of c, respectively.

BEXA performs a beam search by retaining only the $beamwidth$ best conjunctions available for subsequent specialization at each step. Thus, the amount of search is controlled by the parameter $beamwidth$. If the best concept description found by *FindBestConjunction* is no better than simply using the *mgc* as concept description, the result $NULL$ is returned, indicating that no good concept description could be found. Otherwise, the best description found during the search is returned.

3.4.3 The Specialization Model Layer

BEXA's specialization model is implemented by the procedure *GenerateSpecializations*. This routine implements the specific method of evolving concept descriptions, i.e. refining a parent description to form one or more descendent descriptions. CN2's specialization model, for example, forms descendants by adding further conjunctions to a parent conjunction. Theron and Cloete showed that many machine learning algorithms, for example CN2, AQ15, Greedy3, PRISM, and Gray's Algorithm [Clark and Niblett, 1989; Michalski et al., 1986a; Pagallo and Hassler, 1990; Cendrowska, 1987; Gray, 1990], can fit into the framework by implementing the specific rule refinement technique in the specialization model [Theron and Cloete, 1996]. In addition they proposed a specialization method, called specialization by exclusion (hence the name BEXA: **B**asic **EX**clusion **A**lgorithm) that forms descendants by removing (excluding) conditions in the description. We discuss this specialization model and its distinguishing characteristics in the next section.

3.5 Specialization by Exclusion

Assume C denotes the set of all VL_1 conjunctions for a learning problem, and $c_1, c_2 \in C$ are two conjunctions in this description language. Then define $c_1 \preceq c_2$, c_1 is *more specific than or equal to* c_2,

Table 3.4: BEXA's specialization model.

```
    PROCEDURE GenerateSpecializations(P, N, conjunctions,
    beamwidth)
1     specializations = ∅
2     FOR each conjunction c ∈ conjunctions DO
3       // First remove from c.usable all the values that will lead
        to unnecessary specializations
4       FOR each value or interval a_i ∈ c.usable DO
5       IF X_P(c) ⊆ X_P(a_i) // Prevents conjunctions for
        which X_P = ∅
6         OR X_N(c) ∩ X_N(a_i) = ∅ // Ensure more negative
          instances will be uncovered
7         OR {X_N(b_i)|b_i ∈ c.excluded ∪ {a_i}} is a
          redundant partial cover of N THEN
8           c.usable = c.usable − {a_i}
9       // Next generate all useful specializations of the conjunction
10      FOR each value a_i ∈ c.usable DO
11        c' = c specialized by removing a_i from it;
12        X_P(c') = X_P(c) − X_P(a_i);
13        X_N(c') = X_N(c) − X_N(a_i);
14        c'.usable = c.usable − {a_i};
15        c'.excluded = c.excluded ∪ {a_i};
16        specializations = specializations ∪ {c'};
17      END FOR
18    END FOR
19    IF beamwidth > 1 THEN
20      Remove from specializations all duplicate conjunctions;
21    RETURN specializations
    END PROCEDURE
```

if and only if $X_T(c_1) \subseteq X_T(c_2)$. We consider $c_1 = c_2$ whenever $X_T(c_1) = X_T(c_2)$. Conjunction c_1 is *more specific than* c_2, denoted by $c_1 \prec c_2$, when $c_1 \preceq c_2$ and $c_1 \neq c_2$. Thus, the set C is partially ordered under the $\preceq$ relation, and the conjunctions in C and their corresponding extensions form the lattice $\langle C, \preceq \rangle$.

BEXA and AQR have the same description language. It is also clear that VL_1 is more descriptive than the description language employed by CN2 [Clark and Niblett, 1989] and PRISM [Cendrowska, 1987], i.e. the set of legal descriptions in VL_1 is a superset of the set of legal descriptions in CN2 or PRISM. Thus, all of the algorithms that fit in the framework use the same description language, where different algorithms simply apply different search restrictions or heuristics. The algorithms can thus be related to each other by their specialization behaviour within the lattice of VL_1 concept descriptions.

The procedure *GenerateSpecializations* shown in Table 3.4 implements BEXA's exclusion specialization model. Specialization by exclusion performs a general-to-specific search for the best conjunction. The model consists of two phases. In the first phase a set of stop-growth criteria is tested both to prevent

search in regions of the hypothesis space that can have no good solutions, and to prevent overspecialisation. With each conjunction two sets are associated. The set *usable* contains all atoms (attribute values) that may be used to specialize the conjunction further, and the set *excluded* contains all the atoms that were removed from the *mgc* to form the conjunction. An atom here refers to the smallest description possible, e.g. $[outlook = sunny]$.

The first test (Line 5) prevents the creation of conjunctions that cover no positive instances, thus preventing unnecessary search. The second test (Line 6) prevents the creation of conjunctions that cover the same number of negative instances as their predecessor. The third test (Line 7) prevents conjunctions from being over-specialized. Let A be a set of instances and B a set of sets such that $b \in B$ and $b \subseteq A$. The set B is called a set cover of A if $\bigcup_i b_i = A$, and a partial set cover of A if $\bigcup_i b_i \subset A$. If B is a set cover of A, then B is an irredundant set cover of A if the deletion of any element in B results in B forming a partial set cover of A. Theron and Cloete proved that members of the set C_M, the set of most general consistent conjunctions, all have the property that their associated set *excluded* forms an irredundant partial set cover of N, the set of negative instances [Theron and Cloete, 1996].

In the second phase of *GenerateSpecializations*, each conjunction in the set $conjunctions$ is specialized. A conjunction is specialized by excluding (removing) an atom from its description, thereby generating a specialization ("new" conjunction). Only atoms from the set *usable* are excluded from the conjunction. After the creation of a new conjunction, its positive and negative extension is computed. These extensions can be efficiently computed by subtracting the extension of the atom from the extension of the parent conjunction. The extensions of the atoms need only be computed once prior to the concept learning. Theron and Cloete used the irredundancy requirement to prove that contrary to other specialization models, the specialization by exclusion model is guaranteed to find members of C_M [Theron and Cloete, 1996].

Table 3.5 contains a small artificial learning problem as illustration. We follow the convention that instances are numbered, and that these numbers are used to indicate which instances belong to a particular set. Specializing the *mgc*, $[A = a \vee b \vee c][B = x \vee y]$, by removing the attribute value a (excluding $[A = a]$) produces the specialization $[A = b \vee c][B = x \vee y]$. Denote this specialization by s, then its extension is the set $X_T(s) = X_P(s) \cup X_N(s) = \{3, 4, 6\} \cup \{5\}$, where X_P and X_N denote the *positive extension* and *negative extension* of s respectively.

If the beam width is greater than one, all duplicate conjunctions are removed from the set of specializations before it is returned. Duplicate specializations occur when the same two atoms are excluded from the *mgc* in different order. For example, the conjunction $[a, b][x]$ can be formed either by excluding y from $[a, b][x, y]$ or by excluding c from $[a, b, c][x]$.

3.6 Summary

In this chapter we reviewed the BEXA set covering framework. The framework consists of three layers, a set covering layer, a heuristics layer, and a specialization model layer. Many machine learning algo-

Table 3.5: A small artificial learning problem.

```
@relation smallproblem
@attribute A       {a, b, c}
@attribute B       {x, y}
@attribute concept {yes, no}

@data
 a, x, yes ;1 | b, y, yes ;4
 a, y, no  ;2 | c, x, no  ;5
 b, x, yes ;3 | c, y, yes ;6
```

rithms fit into this framework, i.e. they make use of the same description language, and are characterized by their search method within the lattice of VL_1 concept descriptions as implemented by their specific specialization model. Thus, the framework provides a basis for the comparison of different crisp set covering algorithms. We also introduced the specialization by exclusion model which starts with the most general concept description and specializes it by excluding atoms, as opposed to the method of adding more and more atoms to the constant TRUE employed by most other set covering algorithms. Contrary to other specialization models, the specialization by exclusion model is guaranteed (under certain conditions) to find elements of the set of most general consistent conjunctions.

CHAPTER 4

Fuzzy Set Covering and FUZZYBEXA

4.1 Introduction

We have shown in the previous chapters that set covering is a very successful and well established concept learning methodology. We have also established that although fuzzy sets as the generalization of crisp sets are much more powerful, set covering has not been used for the induction of fuzzy classification rules. In this chapter we propose *fuzzy set covering* as a new methodology for fuzzy concept learning. We identify and address all the problems that arise when applying set covering in the fuzzy case, and we propose a novel algorithm, FUZZYBEXA, as the first algorithm that uses set covering and a partial ordering of its description language for the induction of fuzzy classification rules.

There are multiple reasons for generalizing classification rules based on crisp sets to classification rules based on fuzzy sets. Fuzzy sets have increased expressive power. They allow the explicit expression of imprecision, vagueness and ambiguity, whereas crisp sets imply rigid boundaries, and only allow for the concepts true and false. This is not to say that crisp sets are more precise than fuzzy sets. The term fuzzy should not be taken to mean that results obtained from fuzzy rules are imprecise, rough estimations. In fact, it was shown that a fuzzy system can be used to model any real continuous function on a given domain with arbitrary precision, i.e. fuzzy sets are universal approximators [Kosko, 1994]. FUZZYBEXA makes use of linguistic terms as described by fuzzy sets for its concept descriptions, thus it unifies the symbolic and sub-symbolic knowledge representations, bringing the fuzzy and symbolic machine learning community closer together. Fuzzy systems are able to model highly complex systems with poorly understood or non-linear behaviour, and fuzzy rule-based systems usually execute faster than conventional rule-based systems, since fuzzy rule bases usually have fewer rules [Cox, 1998]. A smaller rule base is easier to understand and maintain, leading to fuzzy systems often being more comprehensible. Of course, when a fuzzy rule base becomes very big, these benefits will not be as pronounced, or may disappear entirely.

Instances can belong to a fuzzy concept to any degree of membership in the range $[0, 1]$. FUZZYBEXA uses this information during rule induction, and we will show that FUZZYBEXA is capable of inducing very accurate rule sets. However, we will also show that FUZZYBEXA fulfills the three requirements for inducing highly comprehensible rule sets [Guillaume, 2001]: FUZZYBEXA's rule sets are very small, its rules are incomplete, and its rules use linguistic terms. The potential search space

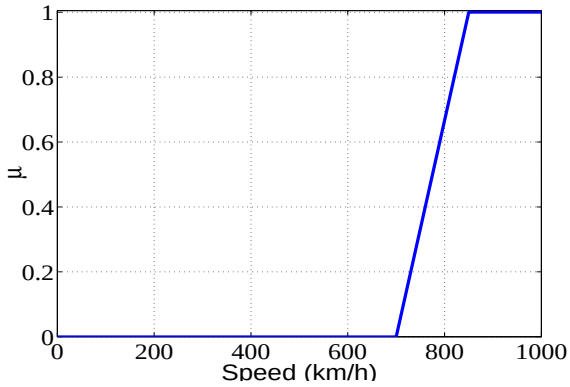
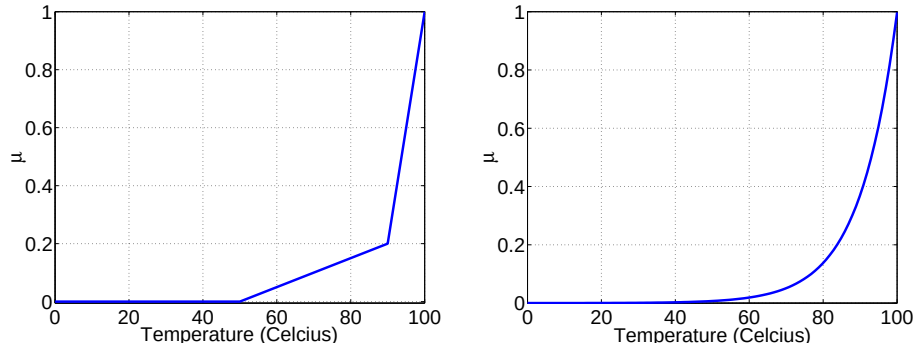


Figure 4.1: An example of a membership function for the fuzzy set *speed.high*.

defined by FUZZYBEXA's description language grows exponentially in the number of linguistic terms defined by the problem domain. Thus, we develop several efficiency and prepruning criteria applicable in the fuzzy case, and we demonstrate that FUZZYBEXA is a practical tool for knowledge discovery, capable of completing in reasonable time even for very large domains.

FUZZYBEXA's structure is based on the crisp set covering algorithm BEXA, discussed in the previous chapter. As pointed out in Section 3.3, BEXA deals differently with numerical and nominal attributes. BEXA's method of handling numerical attributes requires the algorithm to create $2N$ new attribute values, where N is the number of unique measurements. Fuzzy sets provide a natural way to deal with numerical attributes for reasoning, providing a framework for a non-rigid interface between classes represented by symbolic labels and numerical values [Dubois and Prade, 2003]. Consider for example the attribute $speed$, as measured for an aircraft. Crisp algorithms learn sharp thresholds for ranges, and outside of these ranges the condition is 100% false. If there is a condition $[speed > 850]$, this condition will be false for the observation 849. A fuzzy set describing high speed may have the membership function shown in Figure 4.1. An instance with a speed measurement of 849 will still highly belong to the fuzzy set $speed.high$, whereas a measurement of 750 will belong only somewhat, and a measurement of 700 will definitely not belong to the fuzzy set anymore. Fuzzy sets thus provide a more natural way to deal with numerical attributes, allowing the definition of concepts such as high speed, rather than employing sharp thresholds. The fuzzy set describing speed can also be adapted for different situations, as depicted in Figure 4.2. FUZZYBEXA thus unifies the treatment of linearly ordered and unordered (nominal) attributes—FUZZYBEXA makes no distinction between different kinds of attributes, as they can all be described by the general case of a fuzzy set.

The layout of the rest of the chapter is as follows. Section 4.2 presents the theoretical background for developing the basic fuzzy set covering approach, and Section 4.3 introduces our fuzzy generalization of BEXA. In Section 4.4 we prove that FUZZYBEXA's description language induces a lattice of concept descriptions. The following two sections describe FUZZYBEXA's top and bottom layers in detail, and their functionality is demonstrated on a small data set in Section 4.7. Section 4.8 discusses FUZZYBEXA's inductive bias, and Section 4.9 describes the fuzzy inference system used for classifying instances. In Section 4.10 we investigate more theoretical aspects of the algorithm such as the size of the hypothesis



(a) Scenario 1 (b) Scenario 2

Figure 4.2: An example of two different membership functions for the fuzzy set *temp.high* in different scenarios. In figure (a) there is a slow linear increase in membership from 50 degrees until 90 degrees, where after there is a sharp increase in membership. Such a membership function may be applicable in a scenario where temperatures above 50 degrees are undesirable but tolerable, and system failure may occur above 95 degrees. In figure (b) the idea of high temperature is formed slowly. This may be applicable in a scenario where temperatures above 100 degrees are definitely unwanted, and temperatures between 50 and 100 degrees are increasingly undesirable, with more emphases on higher temperatures.

space and the kind of learning problems that FUZZYBEXA is most suitable for. Section 4.11 concludes the chapter.

4.2 Basic Fuzzy Set Theory

We repeat some elements of fuzzy set theory to be used during the development of the FUZZYBEXA algorithm. Let U be a given universal set, or universe of discourse. Traditionally, a set A, $A \subseteq U$, is defined using one of three methods: listing each element in the set, e.g. $A = \{a, b, c\}$, using a proposition to describe a property that must be satisfied by all the members of the set, e.g. $A = \{x | x \in \mathbb{Z}, 0 < x < 10\}$, or using a function, usually called the characteristic function, that declares which elements are members of the set,

$$\mu_A(u) = \begin{cases} 1, & \text{for } u \in A \\ 0, & \text{for } u \notin A \end{cases} \tag{4.1}$$

where $u \in U$.

Fuzzy sets are a generalization of crisp sets, and are defined using the functional method, where the characteristic function is now defined as

$$\mu_A(u) : U \rightarrow [0, 1] \tag{4.2}$$

Note, the universe of discourse U is still a crisp set of elements, and A is a fuzzy subset of U. The degree to which an element u, $u \in U$, belongs to the fuzzy set A is now described in terms of the *membership function* $\mu_A(u)$. This degree of membership expresses the certainty or ambiguity that u belongs to A, with $\mu_A(u) = 1$ meaning absolute certainty that $u \in A$, and $\mu_A(u) = 0$ absolute certainty that $u \notin A$.

Crisp sets are special cases of fuzzy sets, since for a crisp set, $\mu_A(u) : U \rightarrow \{0, 1\}$, i.e. the membership function is either 1 or 0, and elements can either belong to a set or not with absolute certainty. The fuzzy set operations corresponding to the crisp set operations union, intersection, and negation are defined by the membership functions of the respective operations,

$$\mu_{A\cup B}(u) = \max(\mu_A(u), \mu_B(u)), \qquad \forall u \in U \tag{4.3}$$

$$\mu_{A\cap B}(u) = \min(\mu_A(u), \mu_B(u)), \qquad \forall u \in U \tag{4.4}$$

$$\mu_{\bar{A}}(u) = 1 - \mu_A(u), \qquad \forall u \in U \tag{4.5}$$

where A and B are both fuzzy sets. Contrary to Boolean logic, in fuzzy set theory the *law of the excluded middle* and the *law of contradiction* are broken, and therefore the following may be true:

$$A \cup \bar{A} \neq U \tag{4.6}$$

$$A \cap \bar{A} \neq \emptyset \tag{4.7}$$

The fuzzy instance space I is described by the product of one or more linguistic variables A_i,

$$I = \langle A_1 \times A_2 \times \ldots \times A_n \rangle \tag{4.8}$$

Each linguistic variable is described by a product of one or more fuzzy sets L_j, called linguistic terms,

$$A_i = \langle L_1 \times L_2 \times \ldots \times L_m \rangle \tag{4.9}$$

Together the linguistic terms form the *term set* of the linguistic variable. Each linguistic term is a fuzzy set, and a linguistic variable is thus a family of fuzzy sets, and a fuzzy set itself. A fuzzy instance[1] i, $i \in I$, is thus defined by its membership degrees to the linguistic terms (fuzzy sets) of the various linguistic variables in the problem space,

$$i = \langle \langle \mu_{A_1,L_1} \times \ldots \times \mu_{A_1,L_p} \rangle \times \ldots \times \langle \mu_{A_m,L_1} \times \ldots \times \mu_{A_m,L_q} \rangle \rangle \tag{4.10}$$

Note that the instance space I includes the crisp instance space described in Section 3.2 as a special case. Contrary to the case for crisp instances, in the case of fuzzy instances a fuzzy instance can belong to the entire term set of a linguistic variable to a certain degree, or even with degree one. Thus, in the fuzzy case the condition $[height = tall] \wedge [height = short]$ can be true.

Table 4.1 shows a learning problem akin to that of Table 3.1. We call this data format Fuzzy Attribute Relation File Format, or FARFF. A more detailed description of FARFF is given in Appendix B. The learning problem defines five linguistic variables: *outlook*, *temp*, *humidity*, *wind* and *activity*. Each linguistic variable declaration is followed by its respective term set declaration. For example, *outlook* has the term set $\{sunny, cloudy, rainy\}$.

The membership degrees shown in Table 4.1 represent the degrees of truth (or equivalently certainty, ambiguity or vagueness) to which instances belong to linguistic terms, and should not be confused with probabilities. The probability of an event describes the certainty or likelihood of the outcome of the

[1] The term soft instance has also been used [Wang et al., 2003]

Table 4.1: A fuzzy learning problem analogous to the learning problem in Table 3.1.

```
@relation sport

@attribute outlook  {sunny, cloudy, rainy}
@attribute temp     {hot, mild, cold}
@attribute humidity {humid, normal}
@attribute wind     {windy, calm}
@attribute activity {volleyball, swimming, weights}

@data
(.9 .1 .0), (1.  .0 .0), (.8 .2), (.4 .6), (.0 .8 .2) ;1
(.8 .2 .0), (.6 .4 .0), (.0 1.), (.4 .6), (1.  .7 .2) ;2
(.0 .7 .3), (.8 .2 .0), (.1 .9), (.2 .8), (.3 .6 .1) ;3
(.2 .7 .1), (.3 .7 .0), (.2 .8), (.3 .7), (.9 .1 .0) ;4
(.0 .1 .9), (.7 .3 .0), (.5 .5), (.5 .5), (.0 .0 1.)  ;5
(.0 .7 .3), (.0 .3 .7), (.7 .3), (.4 .6), (.2 .0 .8) ;6
(.0 .3 .7), (.0 .0 1.), (.0 1.), (.1 .9), (.0 .0 1.)  ;7
(.0 1.  .0), (.0 .2 .8), (.2 .8), (.0 1.), (.7 .0 .3) ;8
(1.  .0 .0), (1.  .0 .0), (.6 .4), (.7 .3), (.2 .8 .0) ;9
(.9 .1 .0), (.0 .3 .7), (.0 1.), (.9 .1), (.0 .3 .7) ;10
(.7 .3 .0), (1.  .0 .0), (1.  .0), (.2 .8), (.4 .7 .0) ;11
(.2 .6 .2), (.0 1.  .0), (.3 .7), (.3 .7), (.7 .2 .1) ;12
(.9 .1 .0), (.2 .8 .0), (.1 .9), (1.  .0), (.0 .0 1.)  ;13
(.0 .9 .1), (.0 .9 .1), (.1 .9), (.7 .3), (.0 .0 1.)  ;14
(.0 .0 1.), (.0 .0 1.), (1.  .0), (.8 .2), (.0 .0 1.)  ;15
(1.  .0 .0), (.5 .5 .0), (.0 1.), (.0 1.), (.8 .6 .0) ;16
```

event, whereas fuzzy membership describes the ambiguity or certainty to which the event occurs. Both kinds of uncertainty are measured on the scale $[0, 1]$. However, a zero probability implies an event cannot occur, and probability one implies an event is certain to occur. A zero or one membership degree means a complete lack of ambiguity in the description of an event or element—a zero membership implies an element is definitely not part of a fuzzy set, whereas membership one implies an element is definitely part of a fuzzy set. Furthermore, the sum of membership degrees for a specific term set does not need to be one. For example, the membership degrees for the variable *activity* of instances 11 and 16 sum to 1.1 and 1.4, respectively.

Many real world processes have linearly ordered attributes, both continuous and discrete. For linearly ordered attributes, the fuzzy membership function maps the linear domain to membership degrees on the scale $[0, 1]$. Figure 4.3 shows how temperature values are mapped onto membership degrees for the term set of *temp*, defining the membership functions μ_{cold}, μ_{mild} and μ_{hot}.

Linguistic variables with an unordered input domain, for example *outlook* in Table 4.1, have no associated mapping from a linear domain to membership degrees. In this case the membership function just describes the ambiguity that an instance belongs to a certain term. The semantic interpretation of the term set for *outlook* of instance 1 of Table 4.1, for example, would be that the day was almost certainly sunny and cloudless, and that there was definitely no rain. Note, after the membership degrees for linearly ordered variables have been inferred from their respective membership functions, there is no

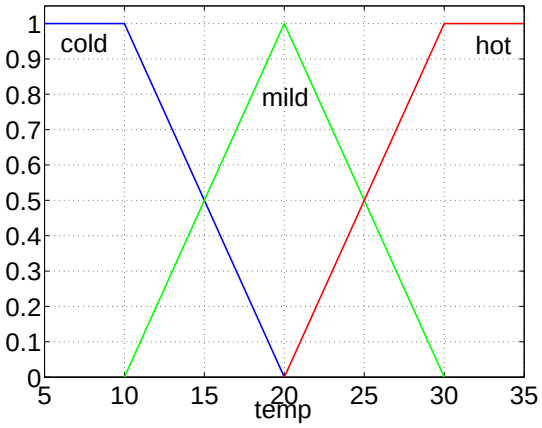


Figure 4.3: The term set and membership functions of the fuzzy attribute *temp* containing the terms *cold*, *mild*, and *hot*.

difference between membership degrees for linearly ordered and nominal attributes. For example, given only Table 4.1, it is not possible to say which variables stem from originally linearly ordered attributes.

4.3 FUZZYBEXA

We now introduce FUZZYBEXA, which extends the definitions used in BEXA to the fuzzy realm. BEXA served as a framework for comparing different crisp set covering algorithms. While FUZZYBEXA can also serve this purpose for a set of fuzzy set covering algorithms, some fuzzy set covering algorithms are not accommodated. We discuss this issue in Chapter 9, and introduce the General Fuzzy Set Covering Framework, FCF (for Fuzzy Covering Framework). However, FCF will borrow extensively from the concepts introduced for FUZZYBEXA.

There were several premises that held true for BEXA in the crisp case. We will refer to these premises by their numbers in the subsequent discussion.

1. It is possible to construct descriptions that cover the whole instance space, and thus will match all possible instances, whether they have been observed (i.e. are in the training set) or not.

2. A conjunct such as $[outlook = sunny]$, will match *all* instances that has attribute value $sunny$ for attribute $outlook$.

3. An instance contains exactly one value for each attribute. Missing values in the data can be catered for, as is typically done, by substituting the most common value for nominal attributes or by the average for numerical values.

4. If an attribute value is excluded (removed) from an internally disjunctive condition that contains all possible values for that attribute, then the description will only match those instances that match the remaining attribute values. This is equivalent to the negation of the excluded values in the condition, and thus exclusion performs the set difference operation with respect to the extension

of the excluded values. This was illustrated in Section 3.5 using the example of Table 3.5, where the attribute value a was excluded from the condition $[A = a \vee b \vee c]$.

5. This process of exclusion can be continued until only one attribute value remains (see Section 3.5). Removing all attribute values and thus leaving only the empty set, is semantically equivalent to stating that the condition will always be false. In practice, of course, such rules are useless.

6. If an attribute value is excluded from an internally disjunctive condition, all the instances covered (matched) by the attribute value are no longer matched by the condition, i.e. they are "uncovered."

7. If a rule antecedent contains an attribute that takes all its possible values, i.e. no value was excluded, the condition matches all instances and can be removed from a description since it is irrelevant.

A fuzzy algorithm will have to reevaluate these premises, as not all of them carry over directly to the fuzzy case. For example, each fuzzy instance is a member of the fuzzy instance space as defined by Eq (4.8) and (4.9), and thus the premise stated in point 3 above for the crisp case does not hold in the fuzzy case—a fuzzy instance can belong to *all* terms to a non-zero degree. This has important implications for a fuzzy set covering algorithm.

As discussed in Chapter 2, much work has already been done on the development of algorithms that can extract fuzzy rules from data. The implementation of a fuzzy rule based system consists of two stages, parameter and structure identification. Parameter identification entails the acquisition of various parameters used during the induction process, for example obtaining membership functions. Much less work has been devoted to the second stage, structure identification, likely since it is a very complex process [Pomares et al., 2002]. Structure identification entails the development of optimal rule structures. Some algorithms perform no or very limited structure identification, inducing huge rule sets consisting of complete rules, and at most delete irrelevant rules [Wang and Mendel, 1992; Ishibuchi et al., 1995]. One of the primary reasons for developing fuzzy rule based systems, however, is the high comprehensibility of fuzzy rules.

After surveying the design of fuzzy learners from an interpretability point of view, Guillaume [2001] stipulated three main requirements for the high comprehensibility of rule sets,

a. The fuzzy sets must be interpretable as linguistic terms (labels). The linguistic terms must be meaningful within the problem domain such that they are understandable to domain experts. This allows the rules to be comparable with one another, leading to knowledge discovery.

b. The set of rules must be as small as possible. Smaller rule sets perform worse on training sets, but often obtain better generalization performance and are easier to read and thus comprehend.

c. The rules must be incomplete. If a rule premise involves all linguistic variables, there is a loss of interpretability without an increase in performance when the rule context could be restricted to the relevant subset of variables only. The systematic presence of all variables in the antecedent

can be seen as a drawback of most automatic rule induction systems. This is due to the induction technique, and an intrinsic characteristic of the problem domain.

Guillaume's first requirement for rule set comprehensibility is that the rules must make use of linguistic terms, and that the rule induction method should not blindly optimise membership functions purely in pursuit of better classification accuracy performance. Cox also notes that the discovery of suitable membership functions in non-control type problems is often more straightforward than one might think—often their specification is already fixed by external processes [Cox, 1998, p. 512]. Fuzzy systems are also not as brittle as crisp rule based system, allow for a certain amount of noise in the membership functions, and can be refined quickly to bring the model prototype into alignment with reality [Cox, 1998, p. 22]. To satisfy the first requirement we assume that membership functions were obtained either from the experts themselves, or during an automatic external parameter identification phase. Thus, the process is not an intricate part of the induction process. This is also done for fuzzy decision tree induction [Yuan and Shaw, 1995] and algorithms such FRIwE [Carmona et al., 2004].

The second requirement is that rule sets should be as small as possible. Therefore, the development of our fuzzy algorithms will, for now, focus solely on structure identification. The methods described in the remainder of this work are able to use any membership function specification (regardless of how it was obtained) to find as small a structure as possible that explains the given data set by inducing rules for it. FUZZYBEXA performs a general-to-specific search and also prefers more general rules over more specific rules. Thus, FUZZYBEXA's rules are biased to cover as many instances as possible, and as a direct consequence the rule sets are very small compared to most other induction methodologies. We demonstrate this empirically in the next chapters.

Fuzzy set covering also satisfies the last requirement—the induction of incomplete rules. In fact, the whole fuzzy set covering methodology is perfectly positioned to fully satisfy all three requirements. It will become clear in this and the next chapters that algorithms implementing the fuzzy set covering methodology are capable of inducing highly comprehensible rule sets. However, we will also show that the methodology allows the induction of not only interpretable rule sets, but also very accurate rules. The rest of this section describes the various aspects of FUZZYBEXA.

4.3.1 FUZZYBEXA's Description Language

Instead of using crisp attributes and attribute values, FUZZYBEXA's concept descriptions use linguistic variables and terms. FUZZYBEXA allows the formation of conjunctions of *conjuncts*, where a conjunct is a disjunctive expression of linguistic terms from a single linguistic variable. Thus, BEXA and FUZZYBEXA's description languages are of similar form, and both allow internal disjunction (see Section 3.3 for BEXA's description language). We call FUZZYBEXA's description language FuzzyAL, for fuzzy attributional logic.

In the case of BEXA, an instance matches a conjunct if the instance has one of the attribute values present in the conjunct. However, in the fuzzy case instances do not only match or not match a description, and we have to specify what it means for an instance to match an antecedent, and thus also, a rule with this

antecedent. We also have to specify how the membership degree of the instance to the antecedent is computed. The standard fuzzy set operators are defined as follows,

$$\text{standard fuzzy complement :} \qquad \bar{A}(i) = 1 - A(i) \tag{4.11}$$

$$\text{standard fuzzy intersection :} \qquad A(i) \cap B(i) = \min(A(i), B(i)) \tag{4.12}$$

$$\text{standard fuzzy union :} \qquad A(i) \cup B(i) = \max(A(i), B(i)) \tag{4.13}$$

where $A(i)$ and $B(i)$ are fuzzy sets [Ruspini et al., pp. B2.7:8-9, 1998].

The truth value of a description d (i.e. the antecedent in a classification rule) for an instance i can be computed from the membership functions for the fuzzy sets determined by the expression as follows. Let A and B be any two linguistic terms, and $\mu_A(i)$ and $\mu_B(i)$ the membership degree of instance i to the respective linguistic terms, then

$$\mu_{A \vee B}(i) = \mu_{A \cup B}(i) \tag{4.14}$$

$$\mu_{A \wedge B}(i) = \mu_{A \cap B}(i) \tag{4.15}$$

$$\mu_{\neg A}(i) = \mu_{\bar{A}}(i) \tag{4.16}$$

where on the right hand side the subscript of μ indicates the expression used for evaluation. The conventional precedence rules apply for more than two terms. When the membership degree of an instance to the antecedent is not zero, the condition is considered to be true, and the instance matches the antecedent. Our implementation uses the standard fuzzy operators. However, this can effortlessly be replaced by any appropriate t-norm, t-conorm and fuzzy complement operation [Klir and Yuan, 1995]. We also add the additional operator **!**. The meaning of this operator is related but not equivalent to *not*. Let V be a linguistic variable with term set $\{a, b, c, d, e\}$. Then the conjunction $[!a]$ is equivalent to $[b, c, d, e]$, thus the disjunction of all the remaining terms in the term set. We can also write $[!\ d, e]$, which is equivalent to $[a, b, c]$. It should be easy to see that the operator describes exclusion. If the term a is excluded from $[a, b, c, d, e]$, we obtain $[!a]$ (i.e. $[b, c, d, e]$). Thus, FuzzyAL contains the following language constructs,

1. linguistic terms (labels) defined by the problem domain, grouped by linguistic variables,
2. the grouping symbols [and],
3. the operators $\wedge$, $\vee$, and !,
4. and the constants TRUE and FALSE.

We can make the relationship between linguistic variable and an associated term explicit by the notation *variable.term*, e.g. *temperature.hot*. We can write an expression in explicit notation such as [*temperature* is *hot* $\vee$ *mild*], or in short hand form as [*temperature.hot,temperture.mild*], or when there is no confusion between terms simply as [*hot,mild*].

4.3.2 The Extension of a Conjunction

In the crisp case $X_S(c)$, the extension of the conjunction c in the set of instances S, is defined as the set of instances in S that match the conjunction. We say that c *covers* the set $X_S(c)$ in S, and there is no

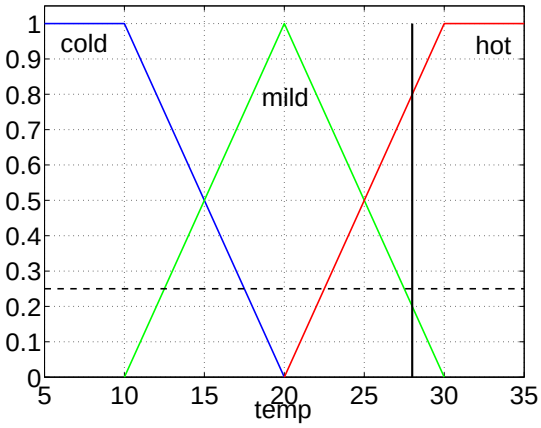


Figure 4.4: The term set for the linguistic term *temp*, and an instance with obervation $temperature = 28$. The alpha cut at $\alpha_a = 0.25$ is shown by a dashed line.

ambiguity of whether an instance matches a conjunction or not. In the fuzzy case, however, instances match a conjunction to a degree in the range $[0, 1]$. Thus, an instance can match a conjunction to a degree 0.001, for example. This may be undesirable, and to prevent such instances from being covered, an α-cut can be applied to the instance memberships (also called alpha leveling [Cox, 1998]), i.e. all instance memberships to linguistic terms that lie below a certain threshold α_a is set to zero. Instances can therefore either match a given conjunction to a degree α_a or above, or not match the conjunction. Thus, in the fuzzy case we define the extension of a conjunction c in the set of instances S, $S \subseteq I$, as follows,

$$X_S(c) = \{s \in S | \mu_c(s) \geq \alpha_a\} \tag{4.17}$$

where we call α_a the *antecedent threshold*. Note, $X_S(c)$ is a subset of the universe of discourse, $X_S(c) \subseteq U$, and is a crisp set of instances. For example, consider the conjunction $[cold][humid]$ for the learning problem in Table 4.1, and let $\alpha_a = 0.7$. The extension in the training set T will be, $X_T([cold][humid]) = \{6, 15\}$, where we have enumerated the instances by their numbers as before. For the standard fuzzy operators, we can also see the alpha leveling as an α-cut being applied to the antecedent membership after the matching process, thus leaving the membership degrees intact, i.e. for both methods the same instances will be covered by a given conjunction c and antecedent threshold α_a. If no α-cut is applied, we define X_S as follows,

$$X_S(c) = \{s \in S | \mu_c(s) > 0\} \tag{4.18}$$

In reference [Wang et al., 1999] the term α-cover is used for the concept of applying an α-cut to the antecedent membership. We will simply use the term cover, since we do not necessarily apply an α-cut. As in reference [Wang et al., 1999], the value of α_a is user-defined, but in our case it is used to prevent instances from matching rules with small memberships. The alpha leveling should not be confused with defining a threshold where membership above implies true and below false - as should be clear from the fact that the alpha leveling is not a necessary requirement for the algorithm.

The premise that if a description contains an attribute value, then it will match all instances that has this attribute value, as stated in point 2 in the list of premises above, does not hold in the fuzzy case.

Instances that belong to a term (with non-zero membership), but with membership below α_a, will *not necessarily* be covered by a conjunction that contains this term. They may still be covered, however, since premises 3 and 6 also do not hold in the fuzzy case—an instance may still belong to another linguistic term from the same term set that is still present in the conjunction with membership above α_a, and therefore may still be covered. Consider for example the conjunct $[hot, mild]$ and instance 3 from Table 4.1, i.e. $\mu_{hot}(i_3) = .8$ and $\mu_{mild}(i_2) = 0.2$. Let $\alpha_a = 0.25$, then although instance 3 belongs to $mild$ with $\mu_{mild} > 0$, the conjunct $[mild]$ does not cover it. However, the instance also belongs to other fuzzy sets from the same term set, and since its membership $\mu_{hot} \geq 0.25$ the conjunct $[hot, mild]$ indeed covers the instance. This concept is illustrated in Figure 4.4. Note, instance 16 has membership 0.5 to both hot and $mild$. The instance matches both terms hot and $mild$ with membership greater than α_a. The exclusion of hot or $mild$ from $[hot, mild, cold]$ results in the conjunctions $[!hot]$ or $[!mild]$, respectively. However, both *still* covers the instance, which was not the case for crisp sets and BEXA. We address this point again in Section 4.6 when we discuss specialization by exclusion in the fuzzy case.

4.3.3 The Most General Conjunction

The *mgc* (most general conjunction) of BEXA contains all the attribute values from all the attributes, and its extension covers the whole instance space I, as stated in point 1 in the list of premises. Let $L_{(i,j)}$ denote a term from the i^{th} variable A_i, and suppose there are m variables. Now consider the interpretation of the *mgc* when its elements no longer denote nominal values in propositional logic, but linguistic terms from linguistic variables, i.e. applied to the fuzzy case. Then the *mgc* is the following description,

$$mgc_{crisp} = [L_{(1,1)}, \ldots, L_{(1,p)}] \ldots [L_{(m,1)}, \ldots, L_{(m,q)}] \tag{4.19}$$

where we use the subscript $crisp$ to be able to refer back to this first version of the fuzzy *mgc*. The same notation is used as in Section 3.3, i.e., square brackets delimit internally disjunctive expressions (conjuncts), each of which form a part of the conjunction.

The fundamental premise of our set covering approach is that it is possible to construct descriptions that determine a family of sets, the union of which covers the whole instance space, and thus will match all possible instances, whether they have been observed (i.e. are in the training set) or not (premise 1). This premise does not carry over directly to the fuzzy case, since it can happen that an instance does not belong to any term of a variable to a sufficient degree (due to the antecedent alpha-cut), and thus cannot be matched by mgc_{crisp} as defined in Eq (4.19). When applying Eq (4.19) and (4.17) to a fuzzy data set, it may then happen that some instances are not covered, and therefore not used in the search process, and they cannot be excluded (see premises 4 to 6). These instances will be exactly those that belong to all terms of a particular term set with membership degrees less than α_a. If we set $\alpha_a = 0.7$ for the data set in Table 4.1 then mgc_{crisp} will, for example, not cover instance 1, because in this case $\mu_{mgc_{crisp}}(\text{instance } 1) = 0.6$ due to the membership degrees to terms of the $wind$ variable.

We address this issue by adding a new term to each variable's term set, the *alpha complement* for that variable, and denote it with $\bar{\alpha}$. The alpha complement has the property that if the membership degrees of

instance i to all the terms in a particular variable's term set are less than α_a, $\mu_{\bar{\alpha}}(i)$ will have membership degree of at least α_a, i.e. exactly in the case when an instance does not belong to the disjunction of the term sets of that variable. We define the degree to which an instance belongs to $A.\bar{\alpha}$ belongs to a variable A as follows.

Definition 4.3.1 *Let $i \in I$ be an instance, A be a variable with the term set $\{L_1, \ldots, L_n\}$, and $m = \sigma(\mu_{L_1}(i), \ldots, \mu_{L_n}(i))$, where σ is any s-norm (i.e. a t-conorm), then,*

$$\mu_{A.\bar{\alpha}}(i) \equiv \begin{cases} 1, & \text{for } m < \alpha_a \\ 0, & \text{for } m \geq \alpha_a \end{cases}$$

We could also use another function with similar behaviour, e.g. a sigmoid function. However, inverse step function is used here since it is also easy to implement. FUZZYBEXA's *mgc* is thus defined as:

$$mgc = [L_{(1,1)}, \ldots, L_{(1,p)}, \bar{\alpha}_1] \ldots [L_{(m,1)}, \ldots, L_{(m,q)}, \bar{\alpha}_m] \tag{4.20}$$

and the membership of an instance i to the *mgc* is given by

$$\begin{aligned} \mu_{mgc}(i) = \min(&\max(\mu_{L_{(1,1)}}(i), \ldots, \mu_{L_{(1,p)}}(i), \mu_{\bar{\alpha}_1}(i)), \\ &\ldots, \max(\mu_{L_{(m,1)}}(i), \ldots, \mu_{L_{(m,p)}}(i), \mu_{\bar{\alpha}_m}(i))) \end{aligned} \tag{4.21}$$

assuming the standard fuzzy operations. The extension of the *mgc* in the instance space I, $X_I(mgc)$, now includes the whole instance space, since for any instance for which the membership degrees to all terms from the same term set are less than α_a, the membership to $\bar{\alpha}$ is above α_a. For example, if we set $\alpha_a = 0.7$ for the data set in Table 4.1, then $\mu_{mgc}(\text{instance } 1) = 0.8$; the alpha complement of the *wind* variable has membership degree 1, and the lowest membership is due to the membership degree of *humidity*—therefore instance 1 is now a member of $X_I(mgc)$. Note, without the addition of the alpha complement, premise 7 also does not carry over to the fuzzy domain. Since mgc_{crisp} does not cover the whole instance space, none of its variables can be seen as irrelevant, even though they all contain their complete term sets, and thus none of the variables could be ignored as in the crisp case.

4.3.4 The Positive and Negative Extension

Set covering algorithms require that for a given concept the training set of instances T, $T \subseteq I$, can be split into two sets, a set of positive instances P, $P \subseteq T$, that contains the desired concept, and a set of instances N, $N \subset T$, that does not contain the desired concept. In the fuzzy case, all instances contain all concepts (that can be described) to a certain degree, since the concept is now specified as a fuzzy set. We therefore define a threshold, or α-cut value α_c that defines which instances belong to the concept and which not.

Definition 4.3.2 *An instance i is positive when $\mu_{concept}(i) \geq \alpha_c$, and negative when $\mu_{concept}(i) < \alpha_c$, where $concept$ denotes the desired concept, and α_c is called the concept threshold.*

Now we can form the set of positive instances P,

$$P = \{i \in T | \mu_{concept}(i) \geq \alpha_c\} \tag{4.22}$$

If, for example, we want to learn the concept $activity.volleyball$, and we set $\alpha_c = 0.8$, then the set P will contain instances 2, 4, and 16 in Table 4.1. Similarly we form the set of negative instances N as follows,

$$N = \{i \in T | \mu_{concept}(i) < \alpha_c\} \tag{4.23}$$

According to the definitions of P and N, it is clear that the sets P and N are disjoint. Thus, the set N can also be obtained by,

$$N = T - P \tag{4.24}$$

The set difference operation is performed on crisp sets, and thus we do not need to be concerned with the implications of Eq (4.6) and (4.7). The positive extension of the conjunction c, $X_P(c)$, is the extension of the conjunction in P, and dually the negative extension, $X_N(c)$ is the extension of the conjunction in N. Consider for example the concept $activity.volleyball$ and the conjunction $c = [sunny][normal]$. Let $\alpha_c = \alpha_a = 0.8$, then $X_T(c) = \{2, 10, 13, 16\}$, $P = \{2, 4, 16\}$, $X_P(c) = \{2, 16\}$, $N = T - P$, and $X_N(c) = X_T(c) - X_P(c) = \{10, 13\}$, where we list the instances by their instance numbers in the table.

Note that we are still working with fuzzy classes. Just as applying an alpha leveling to the antecedent does not revert fuzzy instances to crisp instances, applying an alpha leveling to the concept also does not make the consequent crisp. We simply require that class memberships must lie above a certain level. If one does not want to specify this, one can also define P as,

$$P = \{i \in T | \mu_{concept}(i) > 0\} \tag{4.25}$$

and the definition of N remains as before.

4.3.5 FUZZYBEXA's Rule Semantics

FUZZYBEXA induces rules of the form "IF *antecedent* THEN *consequent*, where the antecedent is a conjunction in FuzzyAL, and the concept is a linguistic term from the term set of the class variables. $X_T(c)$, the extension of the conjunction c in the set T, $T \subseteq I$, defines the set of instances from T for which a rule with c as its antecedent will fire. According to Eq (4.17), a rule will only fire if the antecedent membership is at least α_a. For example, consider the rule

$$\text{IF } [sunny, cloudy][mild]@0.7 \text{ THEN } weights@0.8$$

The number following the antecedent is the value of α_a, the antecedent threshold, and the number following the consequent is the value of α_c, the concept threshold. Thus, this rule will only fire for instances with antecedent membership of 0.7 or above. The explicit indication of α_a and α_c may be omitted when their values are specified for the whole training set. The semantic interpretation of this rule is: if the membership of the antecedent is 0.7 or above, then the membership of the concept is 0.8

or above. A value of 0.7 for α_a is relatively high, and may result in little or no overlap between terms from the same term set for a linear attribute [Cox, 1998, p. 95]. This is often undesirable in fuzzy logic, and thus α_a is usually a smaller value. As already stated before, it is also possible not to apply any alpha leveling to either the antecedent or the consequent. The semantics associated with a fuzzy rule in this case will be that if an instance belongs to the rule antecedent with non-zero membership, it also belongs to the consequent with non-zero membership. One may of course still apply an alpha-leveling to the antecedent after rule induction. Note however, at this point no statement about the actual membership to the consequent is made, except that it will be above α_c if the membership to the antecedent is above α_a. Future research can address methods to predict the concept membership for rules that fire.

The aforementioned rule will fire for instances 4, 13, and 14 in Table 4.1. Instances 13 and 14 have $\mu_{activity.weights} \geq 0.8$, that is, the positive extension contains instances 13 and 14, and the negative extension contains instance 4. Thus the classification accuracy of this rule is $(\frac{|X_P|}{|X_P+X_N|})100\% = 66\%$.

4.4 The Lattice of Fuzzy Concept Descriptions

In Section 3.5 we showed that BEXA's concept descriptions can be arranged in a lattice using a *more specific than* relation. If the extension of a description c is a subset of the extension of a description c', i.e. $X_S(c) \subset X_S(c')$, then c is more specific than c'. The question arises, what happens to the description lattice under fuzzy conditions—does the partial ordering still hold? In BEXA the partial order is defined under set inclusion of description extensions. However, premises 3 and 6 do not hold in the fuzzy case any more (see Section 4.3.2). By defining the partial order for the fuzzy case in a different manner, we show that the description language (FuzzyAL) forms a lattice also in the fuzzy case.

The term set of a linguistic variable defines a number of fuzzy sets for this variable. The description language uses these terms as "labels" in its conjunction description. Let C denote the set of all FuzzyAL conjunctions for a learning problem, and $D(c)$ the set of linguistic terms used in a conjunction c, $c \in C$. The set $D(c)$ is called the *description set* of the conjunction, and $D(c) \subseteq D(mgc)$. Each linguistic term in a description set is assumed to be unique, e.g. the linguistic term $high$ of a linguistic variable "cost" is different from the linguistic term $high$ of a linguistic variable "inflation." We can relabel these linguistic terms as $cost.high$ and $inflation.high$ to maintain uniqueness of names if necessary. Thus, there is a one to one mapping between descriptions and description sets. We now define the following relations,

Definition 4.4.1 *Let C denote the set of all possible FuzzyAL conjunctions in the description language for a specific learning problem, and $c_1, c_2 \in C$, then $c_1 \preceq c_2$, c_1 is* more specific than or equal to *c_2, if and only if $D(c_1) \subseteq D(c_2)$. We consider $c_1 = c_2$ whenever $D(c_1) = D(c_2)$. Conjunction c_1 is* more specific than *c_2, denoted by $c_1 \prec c_2$, when $c_1 \preceq c_2$ and $c_1 \neq c_2$.*

Thus, the set C is partially ordered under the $\preceq$ relation.

The set of description sets D is formed from the power set of all linguistic terms, i.e. $D = \mathcal{P}(L)$, where L is the set of all linguistic terms in the problem domain. Thus, D is closed under arbitrary unions

and intersections, and forms the **complete lattice** $\langle D; \cup; \cap \rangle$ [Davey and Priestly, 2002, p. 36]. For each conjunction c there is a unique associated description set $D(c)$. Next we show that the conjunctions in C form a lattice.

Theorem 4.4.1 *For a given problem domain, the set of FuzzyAL conjunctions C forms the lattice $\langle C; \vee; \wedge; \rangle$, where the meet and join operations are respectively $\wedge$ and $\vee$.*

Proof of Theorem 4.4.1

> For each description $d \in D$ there is an associated conjunction $D^{-1}(d) = c$ formed by the conjunction of conjuncts, where each conjunct is a disjunction of all linguistic terms in d that are from the same term set. An empty conjunct is equivalent to FALSE, as is a conjunction that contains an empty conjunct. Group all conjunctions $c \in D^{-1}(d)$ that are semantically equivalent to FALSE in one node, called FALSE. Thus, except for FALSE, all conjunctions $c \in C$ have a unique corresponding description set $D(c)$. Let $x = D^{-1}(d_1)$ and $y = D^{-1}(d_2)$ be conjunctions such that d_1 and d_2 contain an empty conjunct, then $x \wedge y =$ FALSE and $x \vee y =$ FALSE exist in C. Now let $x = D^{-1}(d_1)$ and $y = D^{-1}(d_2)$ be conjunctions such that d_1 contains an empty conjunct and d_2 does not, then $x \wedge y =$ FALSE and $x \vee y = y$ exist in C. Since $x \vee y$ and $x \wedge y$ exist for all $x, y \in C$, $\langle C; \vee; \wedge \rangle$ is a lattice [Davey and Priestly, 2002, p. 34].

The operator $D(c)$ maps each conjunction $c \in C$ to a unique description set, and the operator $D^{-1}(d)$ maps each description $d \in D$ that does not contain an empty conjunct to a unique conjunction, and all descriptions that contain an empty conjunct to FALSE. Thus, since $\langle D; \subseteq \rangle$ is a complete lattice, $\langle C; \preceq \rangle$ is also a complete lattice.

For each element $c \in C$, it follows that $c \preceq$ *mgc*, and *mgc* is called a **top**. Also, for each element $c \in C$, it is the case that FALSE $\preceq c$, and FALSE is called a **bottom** [Davey and Priestly, 2002, p. 15]. For all elements $c \in C$, if follows that $c = c \wedge$ *mgc*, and *mgc* is called a **one**, and for all elements $c \in C$, it is the case that $c = c \vee$ FALSE, and FALSE is called a **zero** [Davey and Priestly, 2002, p. 41]. Since $\langle C; \vee; \wedge \rangle$ has both a one and a zero, it is a **bounded lattice** [Davey and Priestly, 2002, p. 41]. Let x be a description set and $^-$ be the following operator $\bar{x} = (D(mgc) - x)$. Then, for each element $x \in D$, $x \cap \bar{x} = x \cap (D(mgc) - x) = \emptyset$ and $x \cup \bar{x} = x \cup (D(mgc) - x) = D(mgc)$. Thus, $\langle D; \cup; \cap; \emptyset; D(mgc); ^- \rangle$ is a **complemented lattice**. For each element $x \in \langle D; \cup; \cap \rangle$ its **complement** $\bar{x}$ exists, and therefore $\langle D; \cup; \cap \rangle$ is a lattice with complements. Due to the mapping from D to C, the lattice $\langle C; \preceq \rangle$ has similar characteristics.

4.5 FUZZYBEXA's Top Layers

With all the definitions in place, we can now describe FUZZYBEXA's two top layers. FUZZYBEXA's top layer routine, *CoverConcepts*, implements the fuzzy set covering approach to rule induction, and is

Table 4.2: FUZZYBEXA's fuzzy set covering layer.

```
     PROCEDURE CoverConcepts(T, concepts)
1      ruleset = ∅
2      FOR EACH concept C_i ∈ concepts DO
3        P = {i ∈ T | μ_concept(i) ≥ α_c}
4        N = T − P
5        REPEAT
6          bestconj = FindBestConjunction(P, N)
7          IF bestconj ≠ NULL THEN
8            Add the rule "IF bestconj@α_a THEN concept = C_i@α_c" to ruleset
9            P = P − X_P(bestconj)
10         END IF
11       UNTIL (P = ∅) OR (bestconj = NULL)
12     END FOR
13     RETURN ruleset
     END PROCEDURE
```

shown in Table 4.2. It receives a training set T of fuzzy instances and a list of concepts for which to induce classification rules. For each concept C_i the training set is split into a set of positive instances P and a set of negative instances N. To obtain P, we make use of either Eq (4.22) or Eq (4.25). The set N is formed by subtracting P from T, $N = T - P$. Next, FUZZYBEXA invokes its middle layer routine to obtain the conjunction that best describes the current concept. It then adds the rule with this conjunction as antecedent and the current concept as consequent to its rule set. All the positive instances covered by this rule are then removed from the set of positive instances, while the set N remains unchanged. FUZZYBEXA iteratively induces more rules until either all the positive instances are covered, or no "useful" conjunction could be found, indicated by a NULL value for $bestconj$. It then continues with the next concept until classification rules for all the concepts are induced. Since $|P|$ is reduced during each iteration, the algorithm is guaranteed to terminate.

FUZZYBEXA's middle layer is called *FindBestConjunction*, and is shown in Table 4.3. It implements a set of search heuristics to guide the search. The routine maintains a set of conjunctions, called *specializations*, that are iteratively specialized by invoking the bottom layer routine. This set is initialized by the *mgc*, which is formed as in Eq (4.20). *FindBestConjunction* also keeps track of the best conjunction found during the search by storing this conjunction in the variable $bestconj$, which is initialized to `NULL`.

The set of specializations obtained from specializing the conjunctions in *specializations* replaces *specializations*. Each specialization is then evaluated according to an evaluation function. FUZZYBEXA can use any evaluation function that assigns better scores to conjunctions that cover the positive set better than the negative set, where the exact definition of better is defined by the evaluation function itself. One example of such an evaluation function is the Laplace estimate. The evaluation function plays a pivotal role during rule induction. It is thus very important to use a suitable evaluation function for the problem to solve. The effect of the evaluation function will be the subject of Chapter 6. If a conjunction is found

Table 4.3: FUZZYBEXA's FindBestConjunction procedure.

```
     PROCEDURE FindBestConjunction(P, N)
1      bestconj =NULL;
2      specializations = {mgc }
3      WHILE specializations ≠ ∅ DO
4        specializations =generateSpecializations(P, N, specializations);
5        FOR each conjunction c ∈ specializations DO
6            IF c is better than bestconj according to the evaluation THEN
7              bestconj = c;
8            ELSEIF c and bestconj have the same evaluation
9              AND |X_P(c)| > |X_P(bestconj)| THEN
10             bestconj = c;
11           ELSEIF c and bestconj have the same evaluation
12             AND c is less complex than bestconj THEN
13             bestconj = c;
14           ENDIF
15         ENDIF
16       ENDFOR
17       Remove from specializations all the conjunctions that cover
         no negative instances
18       Remove from specializations all the conjunctions whose optimistic
           evaluation is worse than bestconj's evaluation
19       Retain in specializations only the beamwidth best conjunctions
20     END WHILE
21     IF the evaluation function value for bestconj is the same or worse
         than that of the mgc THEN
22       RETURN NULL;
23     ELSE
24       RETURN bestconj;
     END PROCEDURE
```

with a better evaluation it replaces the current best conjunction. Lines 8 to 18 implement further search heuristics discussed in the next sections.

After searching the current set of specializations for an improvement on the current best conjunction, a set of stop-growth criteria is used to prune the search. FUZZYBEXA prunes conjunctions that are consistent (conjunctions that cover no negative instances) from the search process since these cannot be improved by further specialization. Then only the remaining *beamwidth* best conjunctions are retained in *specializations* for further specialization. FUZZYBEXA performs a type of best-first search if its beam width is set to one. Best first search is a hill-climbing strategy that expands the current state and evaluates its children. The best child is selected and the parent and siblings are ignored. The search halts when it reaches a state that is better than any of its children [Luger and Stubblefield, 1998, p. 127]. FUZZYBEXA searches top-down, and keeps track of the best conjunction found during the whole search. Therefore the *first* (and most general) "best conjunction" found will be returned, not the last. By setting the beam width parameter to a value greater than one, FUZZYBEXA performs a local beam search of

the hypothesis space. That is, in each layer of the lattice of conjunctions, up to a *beamwidth* number of conjunctions may be specialized further. If the best conjunction found during a search performs no better than the *mgc*, the result "no useful conjunction found" (i.e. NULL) is returned, otherwise, *bestconj* is returned. The middle layer also employs other search heuristics to improve efficiency and performance. These are discussed next.

4.5.1 Choosing Bigger Positive Extensions

Lines 8 to 10 in Table 4.3 are used to compensate for some evaluation functions that may assign the same evaluation to two conjunctions, but one conjunction covers more positive instances. Consider for example the Laplace evaluation,

$$L(c) = \frac{|X_P(c)| + 1}{|X_P(c)| + |X_N(c)| + \#concepts} \tag{4.26}$$

Assume $|X_P(c_1)| = |X_N(c_1)| = 10$, and $|X_P(c_2)| = |X_N(c_2)| = 100$. Then evaluations of c_1 and c_2 are $L(c_1) = 11/22 = 0.5$ and $L(c_2) = 101/202 = 0.5$. In this case we prefer c_2 over c_1, as c_2 is more general (i.e. cover more instances) and also has the potential to become better than c_1 by further specialization. Note, this test is not neccesary if $c_1 \prec c_2$, since c_2 would have been evaluated first, and thus kept as best conjunction—$L(c_1)$ was not *better* than $L(c_2)$. However, it may be that c_1 and c_2 are found in the same layer of the lattice and that c_2 is evaluated after c_1—in this case we wish c_2 to replace c_1, even though they obtained equal evaluations.

4.5.2 Preferring Less Complex Conjunctions

If two conjunctions are equivalent according to the evaluation function, and the size of their positive extensions are the same, we prefer the least complex conjunction. Complexity can be measured either in terms of the number of conjuncts in a conjunction, or in terms of the number of linguistic terms in the description set of the conjunction. We chose to used the first approach, since this approach favours rules with more conditions on one linguistic variable to rules using more linguistic variables. The reasoning is that if one linguistic variable can be used instead of two, this identifies the variable as significant and justifies its use in the rule. The cost of measuring one variable is also less than measuring two variables, assuming equal cost for all variables. The complexity test is implemented by Lines 11 to 13 in Table 4.3. If a conjunction obtains the same evluation as the best conjunction, we retain the best conjunction only if it is less complex, otherwise it is replaced.

4.5.3 Optimistic Evaluation

Line 17 cause all conjunctions with empty negative extensions to be removed from the search. Once the negative extension of a conjunction becomes empty, no further amount of specialization will cause the conjunction to improve, as it already covers the maximum number of positive instances (assuming a general-to-specific search). However, it may happen that a conjunction can still be improved, i.e. its

negative extension is not empty, but no amount of specialization will make it better than the current best conjunction. We test for this condition by evaluating the conjunction optimistically. This means, we assume that the conjunction can be specialized such that its positive extension remains intact, but its negative extension becomes empty. If such an optimistic evaluation is still worse than the normal evaluation of the current best conjunction, we remove the conjunction from the search process. Consider for example the conjunction c_1 with $|X_P(c_1)| = 10$ and $|X_N(c_1)| = 10$, then $L(c_1) = 0.5$. The optimistic evaluation of this conjunction is $L_{\text{optimistic}}(c_1) = \frac{10+1}{10+2} = 91.166$. If the best conjunction had an evaluation greater than 91.166, we remove the conjunction from the search, since no specialization can receive a better evaluation and thus replace the best conjunction. The test is implemented by Line 18 in Table 4.3.

4.6 FUZZYBEXA's Bottom Layer

FUZZYBEXA's specialization model, shown in Table 4.4, follows the same specialization by exclusion principle as BEXA and functions as follows. With each conjunction c two sets are associated, the sets $c.usable$ and $c.excluded$. The set $c.usuable$ contains all the terms that may still be used to specialize a conjunction. The set $c.excluded$ contains all the terms that were used to specialize the conjunction. Accordingly, the *mgc* will have $mgc.excluded = \emptyset$ and $mgc.usable = D(mgc)$, the set of all possible terms. When a conjunction c is specialized, one term from $c.usable$ is excluded from c to form $cnew$. The *usable* and *excluded* sets for each conjunction provides for efficient specialization by preventing blind repetition of specialization effort. Like BEXA, FUZZYBEXA employs several criteria to improve efficiency and prevent the generation of useless specializations (referred to as stop-growth of a conjunction) and pre-prune conjunctions. These are discussed next.

4.6.1 Recalculating the Positive and Negative Extensions

In the crisp case BEXA computed $X_P(cnew)$ and $X_N(cnew)$ efficiently as follows. Let $X_P(A_{(i,j)})$ and $X_N(A_{(i,j)})$ be the positive and negative extensions of the conjunction containing only the attribute value $A_{(i,j)}$ where i and j are the attribute and attribute value indices respectively, then

$$X_P(cnew) = X_P(c) - X_P(A_{(i,j)}) \tag{4.27}$$

$$X_N(cnew) = X_N(c) - X_N(A_{(i,j)}) \tag{4.28}$$

The extensions $X_P(A_{(i,j)})$ and $X_N(A_{(i,j)})$ need only be computed once prior to rule induction. However, this computation is invalid in the fuzzy case; it depends upon premise 6 (see Section 4.3). Let $L_{(i,j)}$ denote the j^{th} linguistic term from the i^{th} linguistic variable. In the fuzzy case, the exclusion of $L_{(i,j)}$ from an internally disjunctive conjunct does not necessarily uncover all the instances covered by the term—an instance may still be a member of $X_T(cnew)$, even though this instance is an element of $X_T(L_{(i,j)})$. This will happen when an instance belongs to more than one term in the remainder of the same term set with membership greater than or equal to α_a, and these terms are still in $c.usable$. Thus, unlike in the crisp case, in the fuzzy case it is not true in general that $X_T([!L_{(i,j)}]) \cap X_T(L_{(i,j)}) = \emptyset$.

Table 4.4: FUZZYBEXA's specialization model.

PROCEDURE GenerateSpecializations($P, N, conjunctions$, $beamwidth$)

1 $specializations = \emptyset$
2 // First remove from $c.usable$ all the values that will lead to unnecessary specializations
3 FOR each term $L_i \in c.usable$ DO
4 IF $X_N(c) \cap X_N(L_i) = \emptyset$ THEN // Ensure more negative instances will be uncovered
5 $c.usable = c.usable - \{L_i\}$
6 // Next generate all useful specializations of the conjunction
7 FOR each conjunction $c \in conjunctions$ DO
8 FOR each value $L_i \in c.usable$ DO
9 $cnew = c$ specialized by removing L_i from it;
10 IF $cnew \in specializations$ THEN CONTINUE;
11 $X_P(cnew) = (X_P(c) - X_P(L_i)) \cup X_P(L_1) \cup \ldots \cup X_P(L_j) \cup \ldots \cup X_P(L_n), \quad L_j \notin c.excluded,$ L_j is in the same term set as L_i, and $L_j \neq L_i$
12 IF $X_P(cnew) = \emptyset$ THEN CONTINUE
13 $X_N(cnew) = (X_N(c) - X_N(L_i)) \cup X_N(L_1) \cup \ldots \cup X_N(L_j) \cup \ldots \cup X_N(L_n), \quad L_j \notin c.excluded,$ L_j is in the same term set as L_i, and $L_j \neq L_i$
14 $cnew.usable = c.usable - \{L_i\}$;
15 $cnew.excludedvalues = c.excludedvalues \cup \{L_i\}$;
16 $specializations = specializations \cup \{cnew\}$;
17 END FOR
18 END FOR
19 RETURN $specializations$
END PROCEDURE

We can, of course, simply recalculate $X_P(cnew)$ and $X_N(cnew)$ by computing the membership of each instance with the new conjunction and determine if it is α_a or above. However, $X_P(cnew)$ and $X_N(cnew)$ can be calculated more efficiently in the following way. Let $X_P(L_{(i,j)})$ and $X_N(L_{(i,j)})$ be the positive and negative extensions of the conjunct containing only the term $L_{(i,j)}$. Let $cnew$ be the conjunction formed by excluding the term $L_{(i,j)}$ from c, then,

$$X_P(cnew) = (X_P(c) - X_P(L_{(i,j)})) \cup \bigcup_{k,k \neq j} X_P(L_{i,k}) \tag{4.29}$$

$$X_N(cnew) = (X_N(c) - X_N(L_{(i,j)})) \cup \bigcup_{k,k \neq j} X_N(L_{i,k}) \tag{4.30}$$

where $L_{(i,k)} \notin c.excluded$. The extensions $X_P(L_{(i,j)})$ and $X_N(L_{(i,j)})$ are only computed once prior to rule induction. The above equations require only set difference and union, and can be efficiently implemented by a set data structure backed by a hash table.

Consider the complexity of calculating the extension of a conjunction without this efficiency measure. Here we assume an efficient approach where the calculation is halted as soon as it can be determined

that an instance either matches a conjunction or not, i.e. we do not follow the simple approach of computing the membership of an instance to the conjunction and then testing whether this is below α_a or not. Assuming the standard fuzzy operations, matching requires testing instance membership degrees to the terms in $D(c)$ until the expression can be declared true or false. As soon as the membership of the instance to a *conjunct* is α_a or above, it is known that the instances matches the conjunct. To decide that an instance does not match a conjunct all the terms must be tested and their membership degrees must all be below α_a. The complexity of the matching operation of course grows as the number of terms per term set increases, although not in a linear manner. Let p be the probability that the membership of an instance to a term is α_a or above, then the probability that the membership of the n^{th} term is α_a or above and the membership of the first $n-1$ terms are below α_a follows a geometric distribution, given by

$$P_{\text{geometric}} = (1-p)^{n-1}p \tag{4.31}$$

The mean of the geometric distribution for an infinite series of terms is p^{-1}, i.e. on average p^{-1} terms must be examined before it can be decided that an instance matches a given conjunct. For example, if $p = 0.2$ then on average 5 terms will be tested per conjunct for instances in $X_T(c)$. Thus, for q variables, the number of tests to perform for terms in $X_T(c)$ on average is given by,

$$qp^{-1} \tag{4.32}$$

Let r be the probability that an instance membership to any conjunct in c is α_a or above. Assuming an infinite number of conjuncts in c, for instances in $T - X_T(c)$ on average r^{-1} conjuncts must be tested until one is found that is not matched. Thus, for instances in $T - X_T(c)$ the number of tests performed on average is given by,

$$kr^{-1} \tag{4.33}$$

where k is the average number of linguistic terms per conjunct in c. Consider the sport data in Table 4.1 with $q = 4$ variables and $\alpha_a = 0.5$. Of the 160 membership degrees, 67 are 0.5 or greater. Thus for the *mgc*, $p = \frac{67}{160} = 0.419$, and the average number of term membership tests per instance is $qp^{-1} = 9.55$. We can estimate r as,

$$\hat{r} = \frac{1}{\#\text{conjuncts}} \sum_{i=1}^{\#\text{conjuncts}} \frac{\#\text{ matched instances for conjunct}_i}{\#\text{instances}} \tag{4.34}$$

For the conjunct $[!sunny][!hot][!humid][!windy]$, $r = \frac{1}{4}(\frac{9}{16} + \frac{10}{16} + \frac{10}{16} + \frac{11}{16}) = 0.625$, $k = \frac{6}{4}$, and thus on average $kr^{-1} = 2.4$ membership tests must performed for instances in $T - X_T(c)$. Of course, in reality the number of linguistic terms per conjunct and the number of conjuncts are far from infinite, and the calculations in Eq (4.32) and Eq (4.33) are only approximations. However, the efficiency measures of Eq (4.29) and (4.30) removes the need for matching altogether, and instances in $X_T(L_i)$ can be removed $X_T(cnew)$, provided they are not in $\bigcup X_T(L_j)$, where $L_j \notin cnew.excluded$.

4.6.2 Empty Positive Extension

BEXA used a subset test to prevent the generation of new conjunctions with $X_P = \emptyset$. These are shown in Lines 5 and 8 of Table 3.4,

$$5 \quad \text{IF } X_P(c) \subseteq X_P(a_i)$$
$$8 \quad\quad c.usable = c.usable - \{a_i\}$$

If the positive extension of a conjunction is solely due to a certain attribute value, then excluding this attribute value is senseless.

This test, however, depends upon premise 6, which does not hold in the fuzzy case, as fuzzy instances may have more than one term membership from the same term set in c of at least α_a. Therefore, even though the positive extension of one term may subsume the entire positive extension of the conjunction, this does not mean that if it is excluded the positive extension will become empty – another term's membership may be at least α_a with a non-empty positive extension. However, it is still useless to generate conjunctions with an empty positive extension, as further specialization cannot find better conjunctions. Thus, we remove the subset test as in Line 5 of Table 3.4, and add an alternative test in Line 12 of Table 4.4, after the new positive extension is computed.

$$12 \quad \text{IF } X_P(cnew) = \emptyset \text{ THEN CONTINUE}$$

If $X_P(cnew) = \emptyset$, we do not add $cnew$ to $specializations$ because it does not cover any positive instances. Note, premise 5 is still valid in the fuzzy case—excluding all terms from the same term set means that no term remains to be matched, and is equivalent to the condition FALSE that covers no instances.

4.6.3 Uncover New Negatives

FUZZYBEXA does not specialize conjunctions by excluding terms if the specialization covers as many negative instances as its parent (Line 4 of Table 4.4). When the negative extension of a term has no members in common with the negative extension of a conjunction, specializing the conjunction by excluding this term will not uncover any new negatives. However, the specialization may now not cover some members from the positive extension of the conjunction, making it less general. For this reason, and since we want the most general consistent rule, we do not specialize the conjunction c by excluding the term L if the negative extensions of c and L have no common members, i.e. when $X_N(c) \cap X_N(a_i) = \emptyset$.

4.6.4 Irredundant Set Cover

As discussed in Section 3.5, Theron and Cloete [1996] showed that using the irredundant set cover test leads to the discovery of members of the set C_M, the set of most general consistent conjunctions. We implemented the irredundancy test as an optional extra stop growth measure in the fuzzy case, and we

Table 4.5: A small fuzzy learning problem.

```
@relation smallproblem
@attribute A       {a, b}
@attribute B       {x, y}
@attribute concept {yes, no}

@data

(0.7 0.3), (0.8 0.2), (0.3 0.7) ;1
(0.8 0.2), (0.2 0.8), (0.4 0.6) ;2
(0.7 0.3), (0.1 0.4), (0.7 0.3) ;3
(0.2 0.8), (0.1 0.4), (0.3 0.7) ;4
(0.3 0.7), (0.3 0.7), (0.8 0.2) ;5
(0.5 0.5), (0.3 0.7), (0.6 0.4) ;6
```

examine its use empirically in the next chapter. Since fuzzy instances belong to more than one fuzzy set at the same time, we expect the irredundancy test to have a smaller impact in the fuzzy case, i.e. we expect that the test will not be satisfied often enough to make a significant difference.

4.6.5 Remove Duplicate Specializations

To prevent unnecessary work in the rest of the algorithm, BEXA removed duplicate specializations from the set of specializations after the specialization process (See Table 3.4, Lines 19 and 20). Duplicate specializations are created if two conjunctions differing in only one term are specialized such that the remaining terms are the same, e.g. by excluding hot from $[hot, mild][windy]$ and excluding $cold$ from $[mild, cold][windy]$, the specializations of two different conjunctions are the same, i.e. the conjunction $[mild][windy]$. The number of duplicate conjunctions will increase with the beam width and the sizes of the term sets. Since the calculation of the positive and negative extensions of the specializations is one of the operations with high computational cost (especially for large training sets), FUZZYBEXA optimises this test by moving the duplication test to Line 10 in Table 4.4. This removes the calculation of $X_P(cnew)$ and $X_N(cnew)$ for duplications, which results in a substantial reduction in specialization cost.

4.7 A Small Practical Example

In this section we consider the data set shown in Table 4.5. We induce classification rules for $concept.yes$ using the concept threshold $\alpha_c = 0.4$ and the antecedent threshold $\alpha_a = 0.4$, and we do not consider any stop growth tests. Figure 4.5 shows the complete lattice of conjunctions defined by FUZZYBEXA's description language, subject to the usual constraint that there remains at least one term per variable, i.e. we do not consider the antecedent FALSE that covers no instances. Each node contains a conjunction description, and indicates the sizes of the positive and negative extensions ($X_P(c)$ and $X_N(c)$) of the conjunction in parentheses. In the figure, the alpha complement of a linguistic variable is denoted

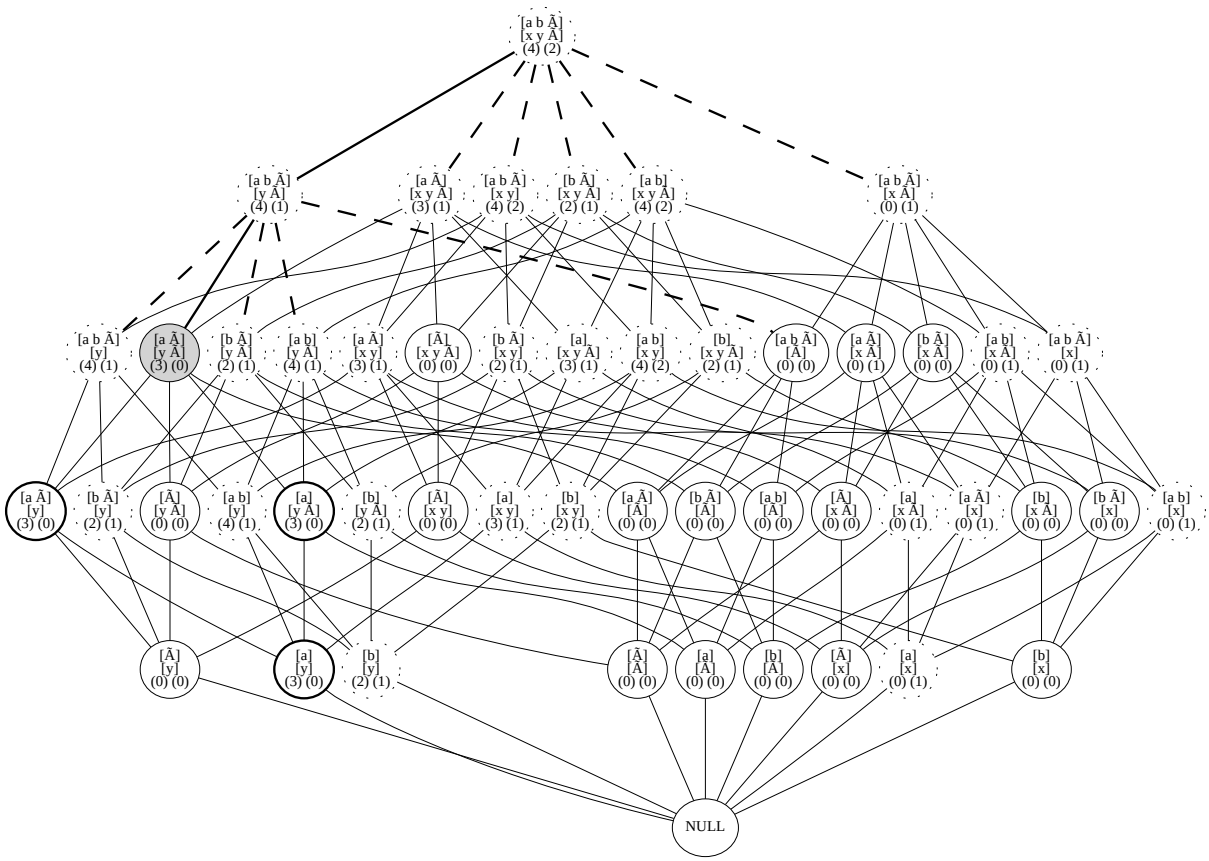


Figure 4.5: The lattice of the learning problem of Table 4.5. The symbol Ã is used to denote the alpha complement.

by the symbol Ã. Dotted nodes indicate inconsistent conjunctions and solid nodes indicate consistent conjunctions.

The procedure *CoverConcepts* computes the sets P and N according to Eq (4.22) and (4.23). The value $\alpha_c = 0.4$ defines the sets $P = \{2, 3, 5, 6\}$ and $N = \{1, 4\}$. Thereafter the procedure *FindBestConjunction* constructs the *mgc* and adds it to the set $specializations$. The *mgc* is shown as the top node of Figure 4.5. By design, the most general conjunction covers the complete training set, and therefore $X_P(mgc) = P$ and $X_N(mgc) = N$. The procedure *GenerateSpecializations* is now called. It constructs all the specializations of the conjunctions in $specializations$, which initially contains only the *mgc*. The six nodes shown in the second layer will be generated by FUZZYBEXA during the first iteration, and are all the specializations possible by excluding one term from the *mgc*. Since there are six terms altogether in the *mgc*, the second layer contains six conjunctions. The positive and negative extensions of each specialization is then computed according to Eq (4.29) and (4.30). The first invocation of *GenerateSpecializations* cannot create any duplicate conjunctions.

The set of specializations generated by *GenerateSpecializations* is then returned to *FindBestConjunction*, and each specialization is then evaluated according to the evaluation function. For this example we use the Laplace function in Eq (4.26). *FindBestConjunction* keeps track of the best conjunction found thus far using the variable $bestconj$. After the evaluation process, only the $beamwidth$ best specializa-

tions are retained in $specializations$, implementing FUZZYBEXA's beam search behaviour. The best conjunction in the second layer is $[a, b, \bar{\alpha}][y, \bar{\alpha}]$, i.e. the *mgc* with x excluded. By excluding x, a negative instance (instance 1) is uncovered. The membership degree of instance 1 to the term y is 0.2, and 0 to the alpha complement of variable B. The membership to the remaining term sets are all above 0.2, and thus the membership to the whole conjunction is 0.2, that is, below α_a, and the instance is uncovered. The dashed bold edges in Figure 4.5 show the paths that FUZZYBEXA will examine with a beam width of one, while the solid bold edges indicate the path that FUZZYBEXA actually follows using the Laplace estimate to rank conjunctions.

There are six nodes in the second layer, one node for each term that was excluded from the term sets of the *mgc*. In the next layer, however, there are not 30 $(5 * 6)$ nodes, since some conjunctions are created by excluding the same two terms in different order from the *mgc*. FUZZYBEXA tests for this condition in Line 10 in Table 4.4. FUZZYBEXA will search in an ordered top-down general-to-specific manner through the lattice, until it finds a member of the set C_M, the set of most general consistent conjunctions. In the third layer the conjunction $[a, \bar{\alpha}][y, \bar{\alpha}]$ has an empty negative extension. It is therefore consistent, and since no conjunction more general is also consistent, it is a member of C_M, and is indicated as a filled node in the figure. Specializing this conjunction further can lead to $[a, \bar{\alpha}][y]$, $[a][y, \bar{\alpha}]$, and $[a][y]$, which all have the same value for the evaluation function, but which are not members of C_M. Nodes with the same Laplace evaluation as that of the filled node are indicated with bold outlines. However, they occur lower down the lattice, and are therefore less general than the filled node. FUZZYBEXA will not specialize further than $[a, \bar{\alpha}][y, \bar{\alpha}]$ due to the stop growth rule in Line 17 of Table 4.3 (i.e. this conjunction does not cover any negative examples).

The first rule has a 100% classification accuracy, since it covers no negative instances, and uncovered three positive instances. These instances are removed from the training set. Since $P \neq \emptyset$, further iterations of the REPEAT loop in *CoverConcepts* are required until all the positive instances are covered, or no more useful rules can be found. In this particular learning problem and data set, the extensions of the alpha complements of both A and B are empty, and thus one could argue that they can be ignored because they can actually not be used to exclude any instances. This yields less general descriptions, but with the same classification performance over the observed instances, i.e.

$$\text{IF } [a][y]@0.4 \text{ THEN } concept.yes@0.4$$

would also be a valid description—and is exactly one of the less general, but equally good descriptions discussed above.

One could argue that any conjunction in the lattice below c, $c \in C_M$, with the same extension is as good as c. However, if we stand by the principle of preferring more general descriptions, then it must be kept in mind that since membership degrees are real numbers from a continuous domain, any training set T is a proper subset of the whole instance space I, $T \subset I$, and it is true in general that if conjunction c_1 is more specific that conjunction c_2, $c_1 \prec c_2$, all instances from I that match c_1 will also match c_2, but a subset of instances B, $B \subset I$, can always be found such that an instance b, $b \in B$, matches c_2 but does not match c_1. That is, even though c_1 and c_2 can have the same extension in T, there always exists a set of unobserved instances B such that $X_B(c_1) \subset X_B(c_2)$.

4.8 FUZZYBEXA's Inductive Bias

We first prove the following theorems and then use them to form FUZZYBEXA's inductive bias when we assume the use of the standard fuzzy operators. The inductive bias will of course change for different fuzzy operators.

Theorem 4.8.1 *Let c be a conjunction, and $c' \prec c$. The membership of an instance $i \in I$ to c' will be less than or equal to its membership to c, that is, $\forall i \in I : \mu_{c'}(i) \leq \mu_c(i)$.*

Proof of Theorem 4.8.1

The membership of an instance $i \in I$ to c will be the minimum membership value of i to any conjunct in c. According to Def 4.4.1, the description set of c' is a proper subset of the description set of c, $D(c) \subset D(c')$. Thus, c contains all linguistic terms of c', but for at least one conjunct c_i in c, there is a linguistic term that is not in the correspoding conjunct c'_i. Let $L \in D(c)$ be a linguistic term such that $D(c') = D(c) - \{L\}$. If $\mu_L(i)$ was not the maximum of the membership degrees to the linguistic terms remaining in the conjunct of L's term set, then $\mu_c(i) = \mu_{c'}(i)$. If it was the maximum, then the membership of i to the conjunct will decrease since the maximum membership decreases. If the decreased membership is lower than the smallest membership to any conjunct, the membership to the conjunction will decrease. Since the membership to more specific conjunctions can decrease, but never increase, the theorem is proven.

Corollary 4.8.1 *Let c be a conjunction, and $c' \preceq c$. The extension of c' will be a subset of the extension of c: $X_I(c') \subseteq X_I(c)$.*

Proof of Corollary 4.8.1

Let $c' \prec c$. Only instances in $X_I(c)$ need to be considered, since all other instances do not match c, they are known to have membership degrees less than α_a. According to Theorem 4.8.1, the membership of an instance $i \in I$ to a conjunction will either decrease or remain the same as it is specialized. Thus, $\forall i \in I(\mu_{c'}(i) \leq \mu_c(i))$, and therefore only the same or fewer instances than in $X_I(c)$ will have membership degrees greater or equal to α_a, and according to Eq (4.17) only the same or a subset of instances can be included in $X_I(c')$.

Starting with the *mgc* defined in Eq (4.20), FUZZYBEXA performs a greedy, general-to-specific, separate-and-conquer beam search of the space of all *conjunctions* in its description language to induce a concept description. FUZZYBEXA's specialization model specializes a conjunction by removing, i.e. excluding, one term from one of the term sets in the conjunction. Next we show that FUZZYBEXA also searches the *instance sets* in the instance space in a general-to-specific order.

Let c_1 and c_2 be elements of the set C of all FuzzyAL conjunctions in the description language of a learning problem, $c_1, c_2 \in C$, and assume c_1 is a specialization of c_2. Thus, $D(c_1)$, the description set of specialization c_1, was formed by removing one or more terms from $D(c_2)$, the description set of conjunction c_2, i.e. $D(c_1) \subset D(c_2)$. The extension operator maps conjunctions in C to instance sets in $\mathcal{P}(S)$, where $\mathcal{P}(S)$ is the power set of a set of instances S. Thus, X is the map,

$$X_S : C \to \mathcal{P}(S) \tag{4.35}$$

The set $\mathcal{P}(S)$ is partially ordered under the set inclusion operation, $\subseteq$ [Davey and Priestly, 2002, p. 4]. The conjunction c_1 is more restrictive than c_2 because less terms are present in the internal disjunction of at least one variable. Thus, c_1 cannot cover instances that are not covered by c_2, but c_1 may *not cover* some of the instances covered by c_2. According to Corollary 4.8.1, the set of instances covered by c_1 is a subset of those covered by c_2, $X_S(c_1) \subseteq X_S(c_2)$, when $c_1 \preceq c_2$. The map X_S is said to be order-preserving or monotone [Davey and Priestly, 2002, p. 26].

FUZZYBEXA only replaces the current best conjunction when a competing conjunction has a better evaluation. The lattice of descriptions is searched in a general-to-specific manner, and thus more specific conjunctions are generated later during the search, and only replace more general conjunctions if they have a higher evaluation. Since the extension operator is order-preserving, the instance sets covered by the conjunctions are also searched in a general-to-specific order. Furthermore, whenever a conjunction becomes consistent, it is removed and not specialized further. Thus, we can formulate FUZZYBEXA's **inductive bias** as follows:

> Conjunctions with good evaluations are preferred over conjunctions with bad evaluations, and conjunctions that cover more instances are preferred over conjunctions that cover fewer instances.

Less formally we can say that FUZZYBEXA prefers good descriptions higher up in the lattice.

4.9 The Inference System

FUZZYBEXA is only concerned with the induction of a good rule set. For this purpose it makes use of a training set of instances. FUZZYBEXA is thus not used to classify instances—it is the task of the fuzzy inference system to use the induced rule set for classification of a set of arbitrary instances (but still from the problem domain). Although each rule is induced in isolation, FUZZYBEXA returns a set of rules. The performance of the inference system is dependent on both the performance of single rules, as well as the degree to which they cooperate. The method of rule cooperation, or in the case of conflicts, the method of rule arbitration, is an important task of the inference system. The inference system is a component largely independent from the specific induction method, in our case FUZZYBEXA. However, since all results (i.e. both for the training and test sets) are directly influenced by the specific implementation of the inference system, we discuss its characteristics in more detail. We first pay attention to the difference between the truth (or membership) of an instance to a rule antecedent and the concept predicted by the rule. Then we discuss instance classification and the default rule.

4.9.1 Conjunction Truth And Concept Truth

Rules that from part of a fuzzy control system are often required to predict a numerical value. Thus, often the final step in an inference system is to convert a generated (or predicted) fuzzy set to a single (crisp) value [Klir and Yuan, 1995, p. 332]. This process is called *defuzzification*. The two most popular methods for defuzzification in such systems are the centre of gravity (or centroid) method and the maximum of the output membership function method [Cox, 1998, p. 31]. The centroid method calculates the weighted mean of the fuzzy region,

$$R = \frac{\sum_{i=0}^{n} d_i \mu_A(d_i)}{\sum_{i=0}^{n} \mu_A(d_i)} \tag{4.36}$$

where d_i is the i'th domain value, and $\mu_A(d_i)$ is the truth membership value of rule A for that domain point. Other methods include the average of maximum values, the average of the support set, the far and near edges of the support set, and the centre of maximum methods [Cox, 1998, p. 314].

FUZZYBEXA' rules, however, are not designed to predict numerical values, but a concept. In this case the membership degree of an instance to the antecedent (conjunction truth) indicates the *certainty* or *confidence* that the rule fires. The certainty or degree to which a rule fires does not predict the membership of the instance to the rule consequent (concept truth), but specifies that the instance membership to the rule consequent lies within a certain range, i.e. the range $[\alpha_c, 1]$. If α_c was not set, then the instance membership to the concept is simply predicted to be greater than zero. For example, let $\alpha_c = 0.8$ and let there be a rule "IF $X = x_1 \vee x_2$ THEN Y." If an instance i matches the antecedent to degree 0.75, we can say that we are rather certain that the rule fires—we are certain to degree 0.75 that the instance belongs to the concept with membership in the range $[0.8, 1]$. Future research can address the possibility to predict a membership degree to a class as well.

4.9.2 Instance Classification and the Default Rule

The process of rule induction does not necessarily produce a set of rules that can classify (i.e. match) all instances in the training set, nor can one guarantee that the induced rule set will assign a classification to each instance in the instance space. In fact, only the *mgc* covers all instances in the instance space. This problem is handled by adding a default rule to the rule set. The default rule matches all instances that are not matched by any rule of the induced rule set. The antecedent of the default rule is therefore the *mgc* or the constant `TRUE` that matches all instances. Although the default rule is always matched, it is only used for classification when no other rules fired.

The consequent of the default rule depends on whether the learning problem requires single class learning or multi-class learning. Single class learning is the case where one class must be distinguished from all other classes, and is dealt with by learning only a set of rules for the instances denoted as positive by the user (i.e. the set *Concepts* in Table 4.2 has only one member). The other instances (negative) are classified as `FALSE`, or `NOT CONCEPT` by the default rule. The user can choose to designate the majority class as the consequent (the class of the negative instances) of the default rule, and only induce rules for the class with the least number of instances (the minority class). This is not automatically

done, since the purpose of rule induction may be knowledge discovery, and not only best classification performance, and it may be the case that the user wants an explanation for the majority class and not the minority class. The consequent of the default rule for multi-class learning (i.e. $|Concepts| > 1$ in Table 4.2) is the majority class, since this rule has the highest probability to be correct.

Another issue to consider when classifying unseen data is that more than one rule may match an instance. In this situation many different approaches can be taken. If all matched rules have the same consequent, the classification is unambiguous. If the matched rules have different consequents, an arbitration method can decide what classification to assign to the instance. One method is to assign the most frequent consequent among the matched rules. Other possibilities are to use the first rule that matched, to assign the classification of the rule that covered the most positive instances, to assign the classification of the rule with the highest evaluation, to assign the classification of that consequent among the consequents of the matched rules that occur most frequently in the training set, or simply to use the default rule.

In the fuzzy case a further possibility exists. Fuzzy instances can belong to more than one consequent with membership above α_c. Thus, an approach only possible in the fuzzy case is to assign more than one classification to an instance when it matches multiple rules with different consequents. However, we did not follow this approach. Although we implemented various different strategies, for the purposes of this dissertation our inferencing system resolves rule conflicts by selecting the rule with the highest set coverage.

4.10 Further Theoretical Aspects

4.10.1 Subsequent Versus Previously Found Rules

One interesting question is, can a subsequently found rule be more general than a previously found rule? That is, can we specialize a rule r_2 found subsequently to form a rule r_1 found previously? If this is the case, we could remove r_1 from the rule set, since r_2 fires for instances that matches r_1, but r_2 is less complex. The answer, however, is typically this does not happen. To subsume a previous rule, r_2 must be higher in the lattice on a chain from r_1 to the *mgc*. However, the extension of r_1 is a subset of the extension of any conjunction above r_1 in the lattice. Since the positive extension of r_1 was removed after the rule was added to the rule set, the evaluation of all conjunctions more general than r_1 will decrease. Thus, although it is not impossible for a subsequent rule to subsume a previous rule, it is more likely that conjunctions that are not in the principal filter of r_1 [Davey and Priestly, 2002, p. 45] will be preferred during the induction of subsequent rules.

A second question is, can subsequent rules be better than previous rules, as measured by the evaluation function? During the induction of earlier rules, more instances are available for the estimation of rule quality by the evaluation function. This estimate should be more accurate, and good evaluation functions should lead to better rules during the early stages of the search, with subsequent rules performing increasingly worse. Furthermore, since there are more positive instances earlier in the search, earlier rules are also more likely to obtain higher evaluations—$|P|$ goes to zero during the induction process while

$|N|$ remains constant. If a finite beam search is performed it cannot be guaranteed that the best rules are found first. In general, however, in our experience even with a very small beam width subsequent rules are always worse with respect to their coverage of P. Thus, the order of rule induction can in most cases be used as an arbitration method to resolve rule conflicts during inference.

4.10.2 Size of the Hypothesis Space

In this section we address the question, how does the size of the lattice of conjunctions grow as a function of the number of linguistic terms. It is to be expected that the induction of a rule within a very large lattice will take longer. The size of the lattice $\langle C; \preceq \rangle$ can be calculated as,

$$|\langle C; \preceq \rangle| = 1 + \prod_i (2^{n_i+1} - 1) \tag{4.37}$$

where i indexes linguistic variables and n_i is the number of linguistic terms in the i^{th} term set. For the i^{th} linguistic variable 2^{n_i} disjunctive expressions can be formed. However, the expression with no linguistic terms is equivalent to false for all conjuncts, and is thus subtracted inside the product and added once outside. The number of linguistic terms per variable is increased by one to include the alpha complement. Since the lattice is searched for the induction of each rule, the size of the complete hypothesis space (i.e. the number of possible rule sets) is,

$$|\text{hypothesis space}| = 2^{|\langle C; \preceq \rangle|} = 2^{1+\prod_i (2^{n_i+1}-1)} \tag{4.38}$$

A small problem may have 5 linguistic variables, each with three linguistic terms, resulting in a lattice of conjunctions of size $1 + (2^{3+1} - 1)^5 = 1 + (15)^5 = 759376$. This space can still be searched exhaustively. However, since the size of the hypothesis space grows exponentially in the number of linguistic terms, larger problems cannot be explored exhaustively. Due to FUZZYBEXA's efficient search heuristics, however, only a very small subset of the entire hypothesis space is examined in reality, and larger problems are still tractable. We will provide empirical experiments substantiating this statement in Chapter 5.

4.10.3 The Importance of Alpha Leveling

Alpha leveling (applying an α-cut) is an important part of a fuzzy inference system [Cox, 1998]. FUZZYBEXA applies alpha leveling at two stages, during training and during inference. Since the need for alpha leveling in a fuzzy learner may be questioned, and we briefly discuss this aspect first for inference and then for training.

During rule matching, the inference engine applies alpha leveling. Thus, instances must match rules at least with a minimum membership degree before they fire. If no alpha leveling is applied during inference, conjunctions would have to be very specific not to also cover many negative instances (although to

small degrees). This would influence the classification accuracy and especially the rule set comprehensibility negatively. Alternatively, the rule conflict resolution method could use the rule that was matched to the largest degree, which can be seen as an implicit form of alpha leveling.

Alpha leveling is also applied during the search process. Since the evaluation functions are fuzzy, alpha leveling is again not mandatory. In some cases exactly the same results are obtained with and without applying an α-cut. Here, applying alpha leveling during inference and not during search gives the same results, but the search is less efficient—instances that do not match rules during inference are kept in the training set longer, and are thus included in calculations during search more often. Not applying alpha leveling during search can influence the search negatively. The cumulative effect of many weakly covered instances can easily obscure the contribution of fewer, but more strongly covered instances, thus leading to the induction of worse rules.

Unfortunately, there is no algorithm for choosing α_a during search or for inference. In the next chapter we present experimental results on the influence of α_a for both search and inference. We will show that although α_a can influence the performance of the algorithm in some cases, many data sets are not very sensitive to it.

4.10.4 For What Kind of Learning Problems is FUZZYBEXA Suitable?

FUZZYBEXA's description language, FuzzyAL, is very powerful and able to form a large variety of concept descriptions—just look at the size of the hypothesis space. Thus, FUZZYBEXA is a general learning algorithm, applicable in most cases. However, there are some problems that FUZZYBEXA was not designed to solve, and for which it will not induce natural (comprehensible) rule sets. FUZZYBEXA (as well as BEXA and most other covering algorithms) cannot count conditions, that is, it cannot describe a concept such as "any four of the nine conditions must be true." A typical type of description that solves this problem is M of N rules. FUZZYBEXA's behaviour for such problems will be awkward. It will induce many rules with antecedents made of different four-conjunct combinations. The classification performance may still be high, but the rule set will be overly complex, and probably not lead to knowledge discovery.

FUZZYBEXA also cannot describe (in a natural way) problems that require the description of relations between different attributes. Thus, FUZZYBEXA cannot describe a concept such as "inflation rate is larger than growth rate" or "green vote is more than red vote." Both situations may be addressed by the addition of extra linguistic variables, making these relations explicit. In the first problem we can add a new variable that describes the number of true conditions. In the second problem we can add the variable describing (inflation rate - growth rate) or (red vote - green vote). However, in most cases this does not provide a satisfactory solution, since we assume that one purpose of rule induction is knowledge discovery, and the addition of these new variables would require too much prior knowledge. Future research can address the extension of the description language to enable fuzzy covering algorithms to also deal with these types of problem domains.

4.11 Summary

In this chapter we proposed set covering as a new methodology for the induction of fuzzy classification rules. We first developed the basic theory for fuzzy set covering, and showed that crisp covering algorithms are based on a set of premises which do not all hold true in the fuzzy case. We then proposed a novel algorithm, FUZZYBEXA, which makes use of the fuzzy set covering approach. FUZZYBEXA uses the same hierarchical structure as the crisp covering algorithm, BEXA. FUZZYBEXA clearly complies with the basic criteria for set covering as stated in Def 3.2.1, it induces a single rule at a time by choosing among several attribute-value pairs, and it removes the positive instances covered after the induction of a rule. We proposed the description language FuzzyAL for describing FUZZYBEXA's rule antecedents, and proved that the descriptions in FuzzyAL form a lattice. Furthermore, we also proved that the fuzzy extension operator is an order-preserving mapping from conjunction space to instance space. Thus, FUZZYBEXA performs a top-down, general-to-specific beam search of the space of instance sets, using a fuzzy evaluation measure to guide its search.

FUZZYBEXA employs a host of efficiency measures and prepruning criteria, improving the speed of rule induction as well as the quality of the induced rule set. Some of the efficiency criteria were adapted from the crisp case, but several novel measures are only valid in the fuzzy case. To demonstrate FUZZYBEXA's search behaviour we traced its search through the lattice of conjunctions of a small toy problem. We also discussed the importance of rule arbitration and the effect of the default rule in a fuzzy inference system. Finally we discussed theoretical aspects of the algorithm, such as the likelihood of rule subsumption, the size of the hypothesis space, and the kind of learning problems that FUZZYBEXA is suited for. The next chapter provides an empirical evaluation of FUZZYBEXA's various parameters.

CHAPTER 5

Empirical Evaluation

5.1 Introduction

The previous chapter introduced FUZZYBEXA, a novel rule induction algorithm employing our proposed fuzzy set covering methodology. In this chapter we investigate FUZZYBEXA's performance on benchmark data sets from the UCI (University of California, Irvine) machine learning repository [Blake and Merz, 1998]. FUZZYBEXA has several learning parameters, for example the beam width and antecedent threshold. A second purpose of this chapter is to determine empirically the effect that each of these parameters have on the performance of the algorithm. FUZZYBEXA also incorporates various efficiency and stop-growth criteria, and we investigate the effect of each of these criteria.

The layout of the chapter is as follows. Section 5.2 discusses the experimental methodology followed during the experiments. The following section provides an evaluation of FUZZYBEXA on benchmark data sets. Section 5.4 investigates the influence of the beam width, followed by an investigation of FUZZYBEXA's sensitivity to noise in Section 5.5, and an investigation of FUZZYBEXA's sensitivity to the antecedent threshold in Section 5.6. In Section 5.7 we evaluate the effect of various stop-growth criteria, and we conclude the chapter with a summary of the main results in Section 5.8.

5.2 Experimental Methodology

There are different ways to evaluate the performance of rule induction algorithms. The classification accuracy, rule set complexity, and computational complexity of algorithm can be computed in several ways. The classification accuracy is a measure of the value or confidence we place in the class predictions made by the rule set. In the absence of a domain expert, rule set complexity often serves as an estimation of rule set comprehensibility. An algorithm that can obtain good classification accuracy as well as rule set comprehensibility in theory, but does not complete in reasonable time is not of practical use, and as such computational complexity is also an importance measurement.

The classification accuracy is perhaps the most important statistic of a classifier. We measure the classification accuracy of our classifiers by counting the number of correct classifications. This calculation follows the procedure as detailed in Table 5.1. In Table 5.1, the *positive coverage* is the number of positive instances in the training set covered by the rule. The different interpretations of the default

$numcorrect = 0$
FOR each instance i
  Find the set of rules R that matches i
  Let r, $r \in R$, be the rule with the highest positive coverage
  IF $R \neq \emptyset$ AND r classifies i correctly THEN
    $numcorrect = numcorrect + 1$
  END IF
  IF $R = \emptyset$ AND the default rule classifies i correctly THEN
    $numcorrect = numcorrect + 1$
  END IF
END FOR

Table 5.1: The procedure for calculating the number of correct classifications on a data set.

Table 5.2: Values for computing ROC ratios.

a	number of correct positive classifications
b	number of positive instances
c	number of correct negative classifications
d	number of negative instances
e	number of incorrect positive classifications
f	number of positive classifications
g	number of incorrect negative classifications
h	number of negative classifications

rule as used in Table 5.1 were discussed in Section 4.9.2. FUZZYBEXA allows the user to specify the consequent of the default rule, and in our case we always choose the class with the highest frequency (majority class). In some experiments we report the Receiver Operator Curve (ROC) measurements, true positives, false positives, true negatives, and false negatives. These ratios are defined as follows,

Definition 5.2.1 *Let a to h be as in Table 5.2, then we define the following ratios, True Positives = a/b, True Negatives = c/d, False Positives = e/f, and False Negatives = g/h.*

When we compute the ratios as defined in Def 5.2.1 the default classification rule is not used. If no rule fires, we count the classification as negative. All experiments are performed using 10-fold cross validation. For a single experiment with a given parameter the input data is kept the same, i.e. we do not use different random folds for the 10-fold cross validation for different values of the investigated parameter, but perform the folding only once, prior to the experiment. Where appropriate, we report the mean and standard deviation of each performance measurement.

We measure the complexity of a rule by counting the number of conjuncts in the rule antecedent. If no linguistic terms from a given linguistic variable were excluded from a conjunction, we do not count the conjunct associated with the linguistic variable (the conjunct is considered as equivalent to TRUE). The complexity of the rule set is the sum of the complexities of its rules. Since the default rule contains no conjuncts in its antecedent (its antecedent is equivalent to TRUE), it does not contribute to the complexity

Table 5.3: The databases used for experiments. [(1)]The FuzzSport data was obtained from [Yuan and Shaw, 1995]. [(2)]The Generated data were randomly generated and labeled using an *a priori* rule set.

Database	Short Description	# Instances	# Linear Att.	#Nominal Att.	#Classes
Anneal	Annealing data	798	6	32	6
Autos	Car import data	205	15	11	4
BreastCr	Reoccurence of breast cancer	286	4	5	2
Colic	Horse colic database	368	7	15	2
Credit-A	Credit approval	690	6	9	2
Digit	LED digits	500	0	8	2
FuzzSport	Sport selection based on weather[1]	16	5	0	10
Generated	Generated fuzzy data[2]	300	0	4	2
Hepatitis	Hepatitis domain	155	6	13	2
Iris	Iris plants database	300	4	0	3
Labor	Final settlements in labor negotitions	57	8	0	2
Lymph	Lymphography domain	148	3	15	4

of a rule set.

We measure the complexity of the search by the total number of conjunctions examined (generated) to induce a rule set. As discussed in Chapter 4, the same conjunction may be reached via different paths through the lattice, which allows duplicate conjunctions to be formed. However, since our algorithm detects this and does not perform any unnecessary work, we do not count a specialization twice during the induction of a single rule.

An instance is matched by a rule if its membership to the rule antecedent is above α_a. For each data set we use a fixed value for α_a throughout the chapter, except where explicitly stated otherwise. During the investigation of a single parameter, all other parameters are kept constant as to not bias the experiment. The data sets were obtained from the UCI Machine Learning Repository [Blake and Merz, 1998], and their characteristics are shown in Table 5.3. Since none of the data sets were originally fuzzy, a fuzzification preprocessing step was performed. The fuzzified data sets were stored and used for all experiments. The fuzzification process is described in Appendix C.

5.3 Evaluation on Bench Mark Data Sets

Table 5.4 shows FUZZYBEXA's performance on the different data sets listed in Table 5.3. The benchmark results serve as reference point for experimental results with different parameters later in the chapter. Table 5.5 shows the ROC breakdown of the classification performance on each data set. We make some observations. The percentage false positives is usually much larger than the percentage false negatives. FUZZYBEXA is biased to induce more general rules, and thus in the absence of negative instances, FUZZYBEXA's rule sets rather cover more than fewer instances, resulting in a relatively larger percentage false positives.

For some data sets the percentage false positives and negatives are very small (e.g. Anneal, Iris), and the classification accuracy is mainly the result of the percentage true positives and negatives. However, as a result of the false positives and negatives, for most data sets the classification accuracy is slightly

Table 5.4: FUZZYBEXA's performance on the benchmark data sets in Table 5.3.

Database	Accuracy		# Conjuncts		Search Effort	
	Mean	StdDev	Mean	StdDev	Mean	StdDev
Anneal	99.00	1.23	102.3	4.6	5003.3	260.0
Autos	74.13	9.80	171.0	14.1	15311.8	1280.8
BreastCr	73.02	4.77	135.0	19.5	4428.0	482.3
Colic	85.60	4.30	169.6	17.7	11208.3	882.3
Credit-A	85.80	6.22	279.7	16.5	14198.1	856.5
Digit	72.72	2.75	194.7	20.6	2429.7	132.5
FuzzSport	62.50	58.93	11.0	1.4	121.0	14.8
Generated	95.30	1.83	49.0	4.5	2412.7	200.2
Hepatitis	81.29	8.71	71.7	5.5	2648.2	148.3
Iris	97.14	4.99	7.0	1.6	283.6	34.3
Labor	91.23	13.84	16.3	1.7	516.3	60.0
Lymph	83.78	12.42	89.9	7.1	2973.6	254.4

Table 5.5: ROC measurement breakdown on the benchmark data sets in Table 5.3.

Database	TP		TN		FP		FN	
	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
Anneal	99.11	0.88	99.67	0.26	1.66	1.27	0.18	0.18
Autos	78.57	9.30	83.72	4.62	39.13	8.60	7.62	3.34
BreastCr	87.67	5.35	40.70	7.35	40.35	3.50	23.25	9.09
Colic	89.67	5.51	65.49	6.88	27.79	4.16	13.62	6.14
Credit-A	93.33	3.29	64.35	10.52	27.64	6.53	9.39	4.45
Digit	75.43	2.58	96.47	0.33	29.65	2.49	2.75	0.29
FuzzSport	75.00	41.76	68.75	33.41	45.45	35.06	15.38	20.19
Generated	92.90	2.47	92.50	2.99	7.47	2.66	7.13	2.24
Hepatitis	82.58	9.54	74.19	7.23	23.81	5.63	19.01	9.15
Iris	94.29	5.63	96.79	3.13	6.38	5.67	2.87	2.73
Labor	92.98	9.54	80.70	16.54	17.19	12.64	8.00	10.73
Lymph	84.46	10.41	88.51	7.45	28.98	12.23	5.53	3.82

worse than the percentage true positives. It may seem strange that for some data sets (e.g. Generated, Labor) the classification accuracy is *better* than both the percentage true positives and the percentage true negatives. This is a result of the method of measuring the ROC values—the measurement does not take into account the default rule. Furthermore, one may also expect the relatively high false positive and negative percentages to impair the classification performance more than it does, e.g. for Hepatitis the percentage true positives and true negatives are respectively 82.6% and 74.2%, and the percentage false positives and false negatives are respectively 23.8% and 19.0%—how can the overall classification accuracy then be 81.3%? The answer is again in the measurement process. The ROC values are strictly measured, with no default rule and no rule arbitration. For each instance and each class, if one or more rules fire and a single rule predicts the class it is counted as a positive (true if the instance belongs to the class), and if no rule fires it is counted as a negative (true if the instance does not belong to the class). However, the inference system makes use of a default rule and a rule arbitration method (see Section 4.9 and Table 5.1). FUZZYBEXA's good overall performance compared to the individual ROC measurements demonstrates the effectiveness of FUZZYBEXA's inference system.

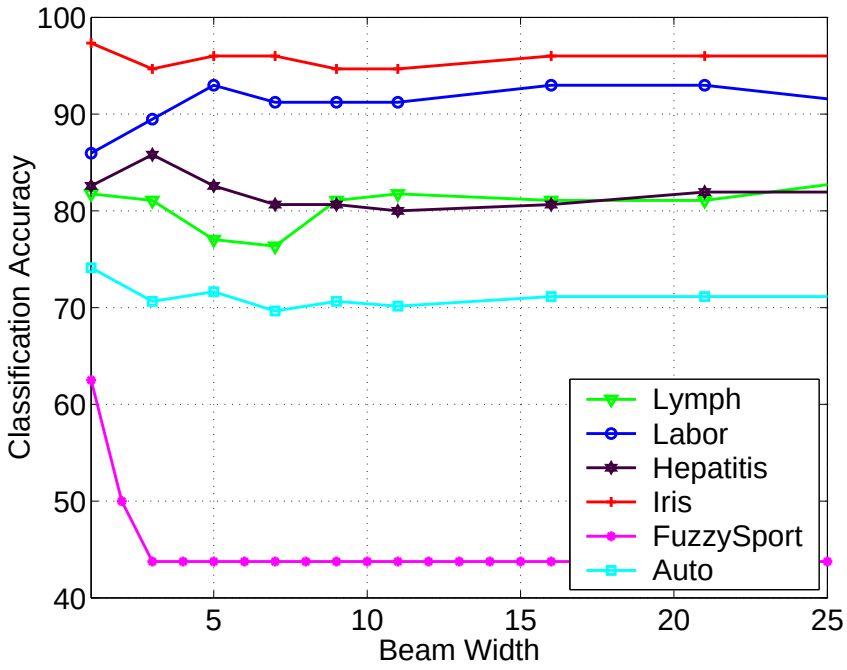


Figure 5.1: The classification accuracy on different data sets as the beam width is increased.

5.4 The Effect of the Beam Width

The beam width is an important factor for selecting the amount of search desired. During each iteration of the middle layer, the $beamwidth$ best conjunctions are retained in the set of conjunctions to specialize further. For an infinite beam width an exhaustive search is performed. However, parts of the search space may be pruned (also for an infinite beam width) due to the stop growth criteria. We evaluate the effect of the beam width for six data sets, and for each value of the beam width a 10-fold cross validation is performed. We discuss the effect of the beam width on the classification accuracy, rule set complexity, and search effort next.

5.4.1 Classification Accuracy

Figures 5.1 shows the classification accuracy as the beam width was increased from 1 to 25. Although, at first, one may expect an increase in beam width to almost always have a beneficial effect, increasing the beam width often do not benefit the accuracy of the rule set. Quinlan and Cameron-Jones obtained similar results when increasing the beam width for a crisp inductive learner [Quinlan and Cameron-Jones, 1995b]. They often found good behaviour for a small beam width, and thereafter only slight improvements or deteriorating performance. They ascribe this behaviour to the learner encountering "fluke" descriptions that overfit the training data when big parts of the search space are searched. The Iris, Auto and Fuzzy Sport data exhibited this kind of behaviour, where the best classification accuracy was obtained using no beam search. Increasing the beam width had a dramatic effect on the Fuzzy Sport data. This data set has only 16 instances, but 3 classes. Further increases allowed FUZZYBEXA to discover so-called flukes, and degraded its accuracy.

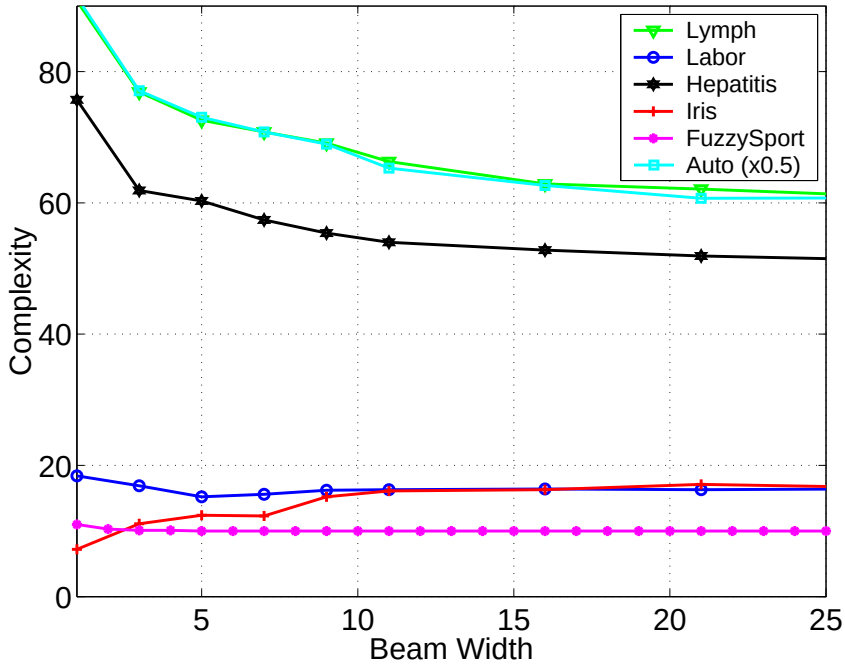


Figure 5.2: The rule set complexity on different data sets as the beam width is increased.

The Hepatitis and Labor data benefited from a small increase in beam width, with the classification accuracy on the Labor data increasing almost eight percent. Further increases degraded performance again. Best classification accuracy is obtained at beam widths three and five for the Hepatitis and Labor data, respectively. Beam widths larger than three degraded performance below that of no beam search for the Hepatitis data, while larger beam widths were always beneficial for the Labor data. The Lymph data demonstrated a high sensitivity to the beam width. Initially, an increase in the beam width degrades performance. Even further increases then improve performance again, after which the performance stays relatively constant.

5.4.2 Rule Set Complexity

Figure 5.2 shows the rule set complexity for different beam widths. The curve for the Auto data set is shown at half scale. For all of the data sets, with the exception of the Iris data set, an increase in beam width resulted in a decrease in rule set complexity. The larger search increases the probability that the heuristics implemented in FUZZYBEXA's middle layer are activated. Thus, conjunctions with higher positive coverage and with less conjuncts are preferred. If conjunctions with higher positive coverage are found, fewer rules need be induced to cover the set of positive training instances, resulting in decreased complexity. However, as discussed above, the higher positive coverage did not always benefit the classification accuracy.

For the Iris data set, the rule set complexity strangely increased with increasing beam width, while the accuracy stayed relatively constant. This behaviour is only observed for the Iris data—the rule set complexity decreased with increasing beam width for all other data sets. One explanation for this strange

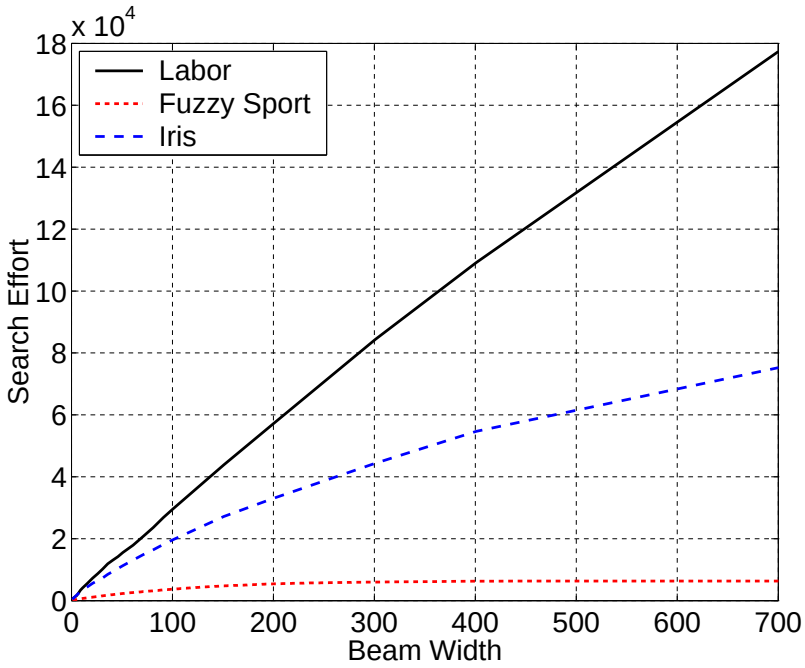


Figure 5.3: The number of conjunctions generated during the search process as a function of the beam width for the Fuzzy Sport, Labor, and Iris data sets.

behaviour may be that FUZZYBEXA may follow different paths through the lattice for different beam widths, and (only) in the case of the Iris data set did the greedy search (i.e. beam width one) based on classification accuracy not favour the rule set complexity.

5.4.3 Search Effort

Figures 5.3 shows the search effort for the Labor, Fuzzy Sport and Iris data for beam width up to 700. Figure 5.4 the search effort for the Hepatitis and Lymph data for beam width up to 50. As expected, the size of the explored search space increased with an increasing beam width. However, the search effort does not increase exponentially, but may rather be modelled by an equation such as,

$$y = \gamma(1 - e^{-\lambda x}) \tag{5.1}$$

This function initially increases almost linearly and then flattens off, eventually reaching a plateau. As the number of linguistic terms increase, both γ and λ increase.

The input domain of the fuzzy sport data set is described by two linguistic variables with two linguistic terms, and two linguistic variables with three linguistic terms. The size of the FuzzyAL search space is computed according to Eq (4.37), i.e. $1 + (2^{2+1} - 1)^2 \times (2^{3+1} - 1)^2 = 11026$ conjunctions in the lattice. The lattice will be searched for each rule to be extracted, and the size of the complete hypothesis space is thus 2^{11026}. However, the size of the space actually examined for the Fuzzy Sport data, for example, is only a very small fraction of the size of the lattice, and insignificant compared to the size of the hypothesis space. This is due to FUZZYBEXA's various stop growth restrictions.

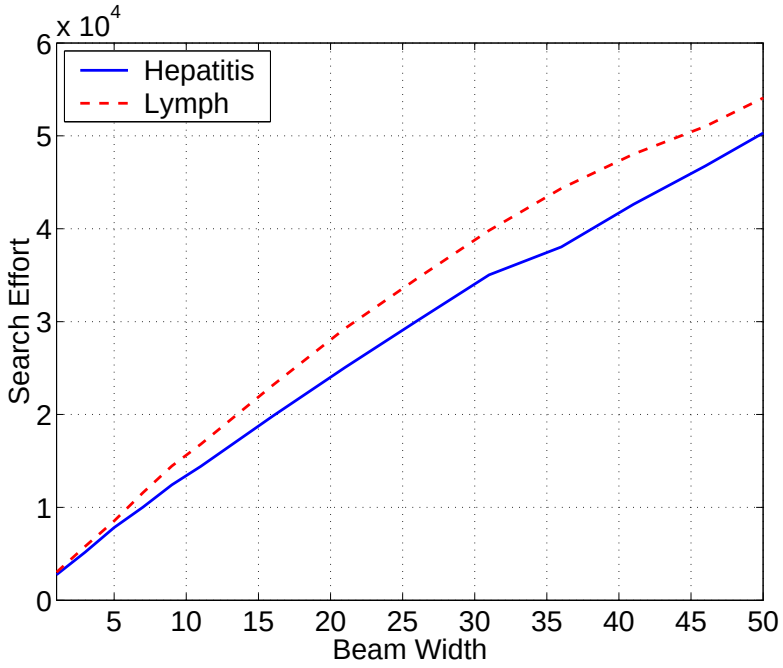


Figure 5.4: The number of conjunctions generated during the search process as a function of the beam width for the Hepatitis and Lymph data sets.

The Iris data exhibited a similar trend to that of the Fuzzy Sport data. The Iris data set also has three concepts, but has 15 linguistic terms compared to 10 of the sport data. Its minimal potential search space is roughly 48 times more than for the Fuzzy Sport data. As is evident from Figure 5.3, the search effort for Iris is far less than 48 times more than the search effort for Fuzzy Sport, again demonstrating FUZZYBEXA's ability to find good rules with comparatively little search. The Hepatitis data set is the second most complex data set used in this experiment. The comparatively small number of conjunctions examined for beam width 50 is further proof of the effectiveness of the stop growth restrictions. The complexity of the Auto data set increased so fast that we do not show it here, and it suffices to say that the increase in search complexity was linear for the examined beam widths.

5.4.4 Discussion

The best accuracy for the Iris and Fuzzy Sport data sets were obtained without using beam search, and the associated rule sets were also the least complex. Increasing the beam width resulted in both lower classification accuracy and higher rule set complexity. There seem to be a general relationship between accuracy and complexity. In most cases a sharp increase in accuracy resulted in a sharp decrease in rule set complexity. Decreasing accuracy was typically also correlated either with an increase in rule set complexity or with a slower decrease than before (smaller gradient).

The Hepatitis and Labor data sets benefited both in terms of rule set complexity and classification accuracy for small increases in the beam width, and the Lymph data obtained best classification accuracy and also much decreased rule set complexity for a relatively large beam width. This demonstrates that bigger

beam widths are not always a bad choice and may yield improvements in some cases. However, care should be taken when using beam search, and it is definitely not true in general that larger beam widths always increase (or even maintain) classification accuracy. Finally, the search complexity experiments showed that although the size of the search space grows exponentially in the number of linguistic terms, the search effort grows at most linearly with the beam width, and in the limit follows a trend more like Eq (5.1). We can ascribe this largely to FUZZYBEXA's effective stop-growth criteria (we will discuss these in detail in Section 5.7).

5.5 Sensitivity to Noise

An important feature of a concept learner is its generalization performance. Another important aspect is its ability (or lack thereof) to maintain good generalization performance in the presence of noise. If a data set contains noise, the learner runs the danger of overfitting on the noise. Some learners, like the Candidate Elimination Algorithm for example [Mitchell, 1997], may completely fail if noise is present. To investigate this issue we generated a synthetic data set and labeled it using a predefined rule set. The data contained five linguistic variables labeled A to E with term set sizes 2, 5, 4, 2, and 3, respectively. The generated data set consisted of 1000 instances with fuzzy membership degrees to the linguistic terms uniformly distributed in the range $[0, 1]$. The predefined rule set was randomly generated, and when those conjuncts that were equivalent to true (i.e. a disjunction of the entire term set) were removed, 40 conjuncts remained in the rule set. We then added noise to the membership degrees from a zero mean Gaussian random variable, and increased the standard deviation in steps of 0.1 from zero (no noise) to one (extreme noise). For each noise level a 10-fold cross validation was performed and the results graphed.

The accuracy in the presence of increasing noise is shown in the top graph of Figure 5.5. FUZZYBEXA was able to classify 94% of the instances correct with no noise added. With the addition of noise, FUZZYBEXA exhibited graceful degradation until noise with standard deviation of 0.5 was added. At this point the classification performance became as good as guessing. At this noise level perturbations of the observed data of size up to 0.5 occur with probability 0.6. For noise levels above 0.5 FUZZYBEXA started to fit the noise distribution, as is demonstrated by the erratic classification accuracy behaviour.

The rule set complexity measured in number of conjuncts per rule set is shown in the middle graph of Figure 5.5. The rule set induced under noiseless conditions was much smaller than the predefined rule set. FUZZYBEXA was designed to induce general rules, covering more instances rather than fewer. FUZZYBEXA was thus able to find a substantially smaller rule set than the predefined randomly generated rule set, while still obtaining high classification accuracy. In accordance with the general decrease in classification accuracy with increasing noise, the rule set complexity showed a general increasing trend, which seems almost linear. As the noise level was increased, more rules and also more complex rules were induced to fit the increasingly unpredictable behaviour, thus increasing the overall rule set complexity.

The bottom graph in Figure 5.5 shows the increasing search effort associated with increasing noise

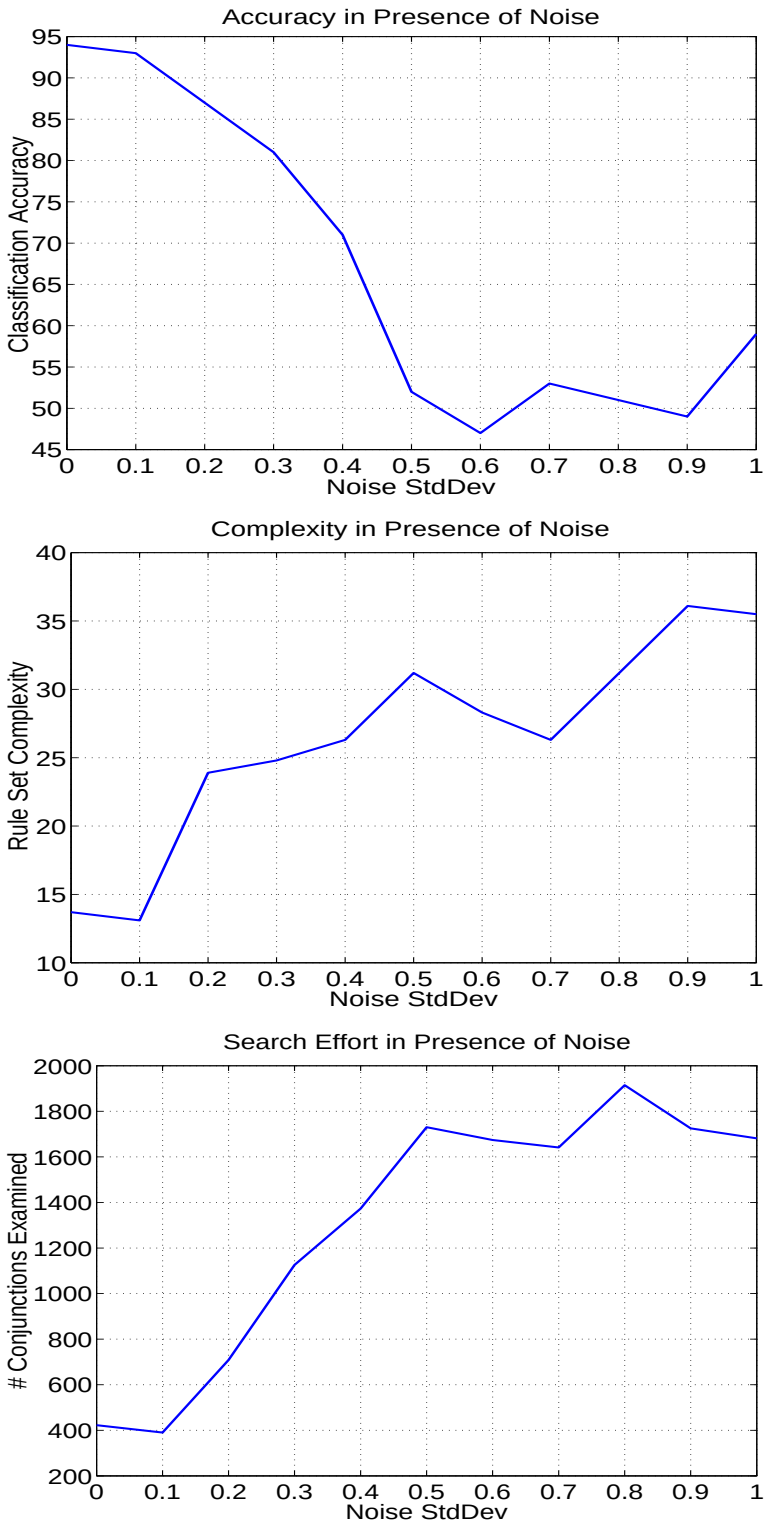


Figure 5.5: FUZZYBEXA's rule set accuracy, complexity and search effort for increasing noise levels.

levels. The size of the lattice of conjunctions for this problem is (see Eq (4.37)),

$$|\langle C; \preceq \rangle| = 1 + (2^3 - 1)(2^6 - 1)(2^5 - 1)(2^3 - 1)(2^4 - 1) = 1435456 \tag{5.2}$$

and the size of the hypothesis search space is thus $2^{1435456}$. However, for no noise, only 400 conjunctions were examined to induce the rule set, demonstrating FUZZYBEXA's high search efficiency. The number of conjunctions examined to compute the rule sets is correlated with the rule set complexity because the search space is examined again for each extra rule induced. FUZZYBEXA's search effort initially increased almost linearly and leveled off at standard deviation 0.5. The experiment thus shows that FUZZYBEXA copes well with noise and demonstrates graceful degradation behaviour with increasing noise levels. With additive normal noise with standard deviation 0.1 FUZZYBEXA's classification performance decreased with only 1%, while its search effort and rule set complexity even improved slightly.

5.6 Sensitivity to α_a

One of FUZZYBEXA's user-determined input parameters is the antecedent threshold α_a. It is certainly not uncommon to apply alpha leveling in fuzzy decision systems and the specific threshold values are highly system dependent [Cox, 1998]. It is therefore interesting to measure FUZZYBEXA's sensitivity to the antecedent threshold. In this section we consider two questions (a) how sensitive are FUZZYBEXA's induced rule sets to a changes in α_a during inference (i.e. after training) and (b) what influence does the training value of α_a have.

5.6.1 Post-Training Sensitivity to α_a

It is of course possible to obtain a set of rules using one value for α_a during training, and then use a different value for α_a during inference on unseen instances. To distinguish between the training and inference values for α_a we denote the training value as α_{aT} and the inference value as α_{aI}. The experimental method for this experiment was as follows. We divided each data set into ten distinct train-test data set pairs as for normal 10-fold cross validation. We then induced ten classifiers using a fixed (specified) value for α_{aT}. For each classifier and test set we obtained classification results as α_{aI} was varied from 0.01 to 0.99 in steps of 0.01. The test set results were then averaged across the different folds. Contrary to all other experiments, the membership functions for this experiment had triangular shapes with adjacent functions crossing at $\mu = 0.5$. Triangular membership functions with this crossing point were chosen since we are interested in the behaviour for larger and smaller overlap between membership functions—when $\alpha_{aI} > 0.5$ there is no overlap and the most overlap occur at $\alpha_{aI} = 0$. As an example of the general form of triangular membership functions, Figure 5.8 shows the membership functions extracted for the linguistic variable "Pulse" of the Colic data. Only classification accuracy is reported since the rule set complexity and search effort are of course constant for each individual value of α_{aT}.

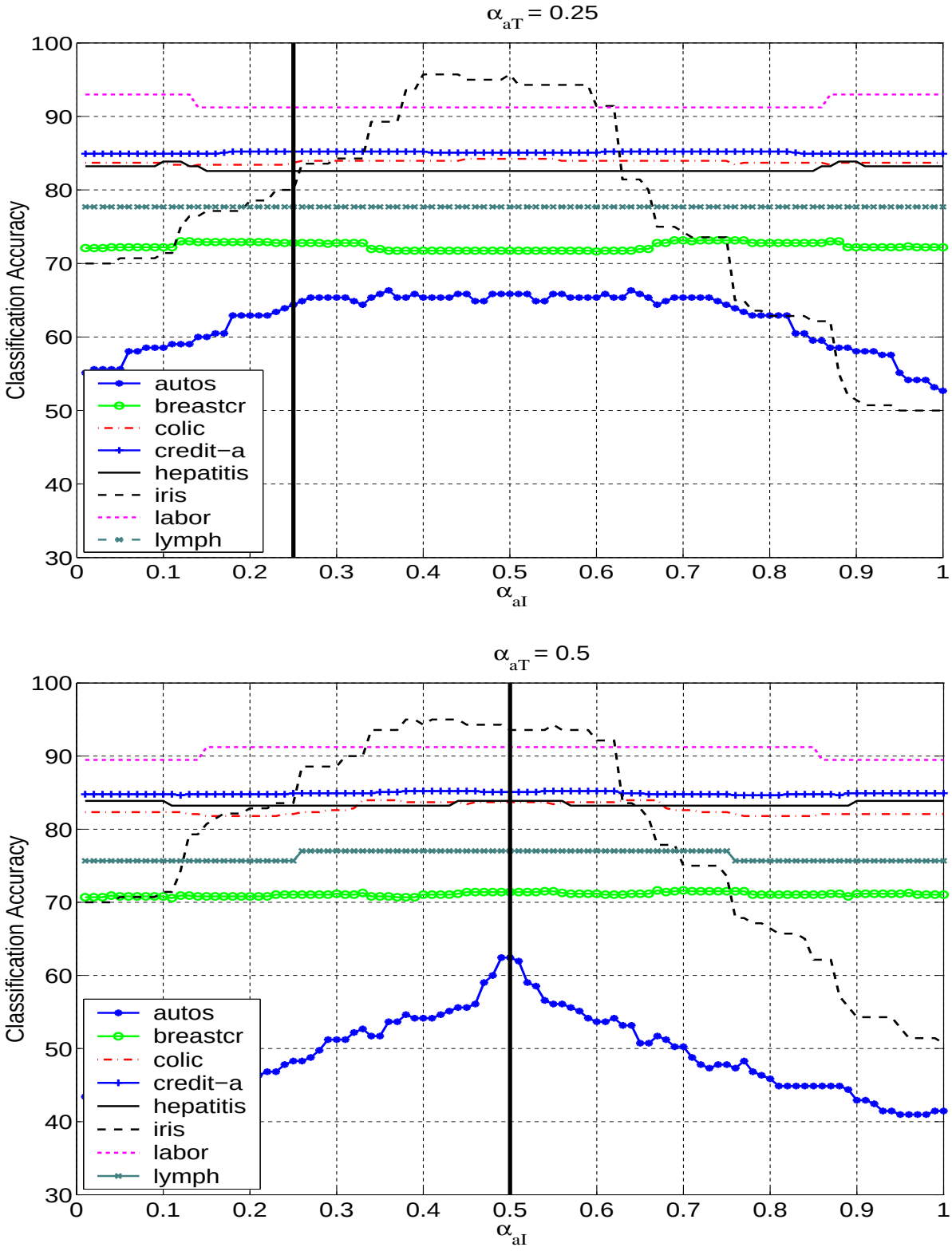


Figure 5.6: Classification accuracies with $\alpha_{aT} = 0.25$ (top figure) and $\alpha_{aT} = 0.5$ (bottom figure).

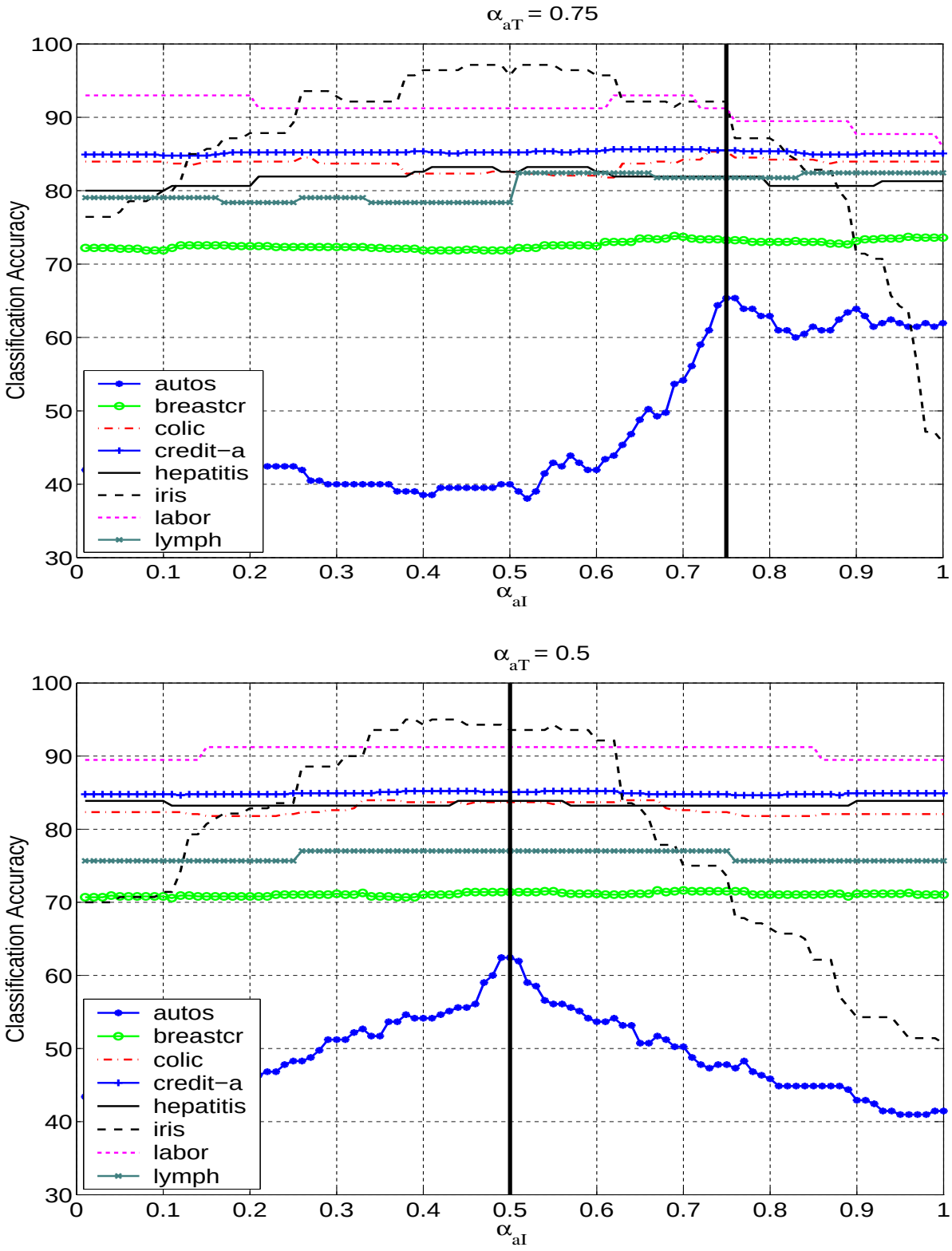


Figure 5.7: Classification accuracies with $\alpha_{aT} = 0.75$ (top figure) and $\alpha_{aT} = 0.5$ (bottom figure).

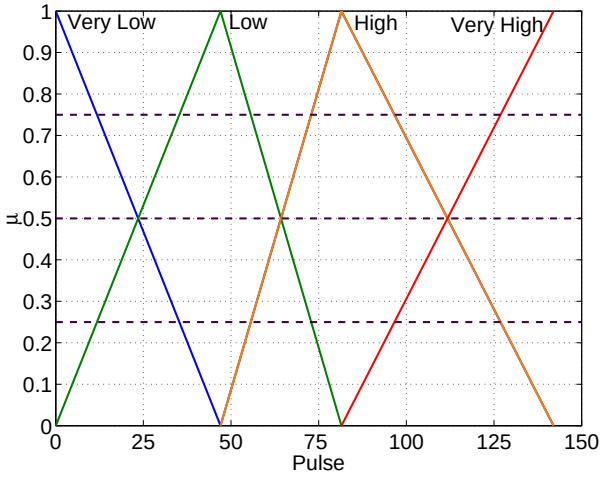


Figure 5.8: Triangular membership functions extracted for the linguistic variable Pulse of the Colic data set. The dashed lines indicate α-cut values 0.25, 0.5 and 0.75.

The experiment was performed using $\alpha_{aT} = 0.25$, $\alpha_{aT} = 0.5$ and $\alpha_{aT} = 0.75$, and the results are shown in Figures 5.6 and 5.7. Our first observation is that the value of α_{aT} influenced the shape of the resultant curves dramatically for some data sets (e.g. Iris, Credit-A), while having very little influence on others (e.g. Lymph, Colic). The sensitivity of the rule set's performance to α_{aT} will be the subject of the next section. Here we just note that different values for α_{aT} result in different performance curves for the different values of α_{aI}. The influence of α_{aT} on the classification performance for different values of α_{aI} will be weakened as the number of linguistic variables with fuzzy linguistic terms (used in the rule set) decrease. Linguistic variables originating from crisp nominal attributes have (a) no overlapping membership, and (b) membership of either 0 or 1. Thus, varying α_{aI} has no influence on these variables. For $\alpha_{aI} = 0$ the most instances in neighbouring (overlapping) terms are covered. As α_{aI} is raised, fewer and fewer instances also covered by neighbouring terms are covered, and for $\alpha_{aI>0.5}$ no instances are covered by more than one term. The number of linguistic variables and their type for the different data sets are shown in Table 5.3.

The next important observation is that best test set results are not necessarily obtained using when $\alpha_{aT} = \alpha_{aI}$, i.e. using the same value for rule induction and inference. For the Iris classifier with $\alpha_{aT} = 0.25$, optimal performance is obtained if $0.4 \lesssim \alpha_{aI} \lesssim 0.6$. This range is slightly wider for induction with $\alpha_{aT} = 0.5$ and much wider for induction with $\alpha_{aT} = 0.75$. The Credit-A and Hepatitis classifiers, on the other hand, exhibited almost complete insensitivity to post-induction variation of α_{aI} for all the training values α_{aT}. Both these data sets had six linguistic variables with fuzzy sets obtained from numerical data, and thus the potential existed for sensitivity to α_{aI}. In fact, the classification accuracy of these data sets *was* sensitive to the training value α_{aT}. The relative insensitivity to post-induction variation in α_a could be due to few fuzzy sets used in the induced rule sets. A further explanation is that the induced rules made use of fuzzy sets that require very little membership to be valid for use in the rule—their mere presence (non-zero membership) is already a clear indication that the rule can fire given that the remainder of the conjuncts are matched.

The Colic and Lymph data sets showed little post-induction sensitivity to α_{aI} for $\alpha_{aT} = 0.25$, while showing higher sensitivity for $\alpha_{aT} = 0.75$. The BreastCr data had the inverse behaviour. Using different induction values α_{aT} can result in different rule sets that make use of different fuzzy sets. The different fuzzy sets may be more or less sensitive to variation of α_{aI} in serving as class predictors, resulting in different sensitivities to α_{aI} of the rule sets employing these fuzzy sets.

The Auto data set demonstrated widely varying results dependent on the induction value for α_{aT}. A very high sensitivity is obtained for $\alpha_{aT} = 0.5$, with even slight post-induction changes in α_{aI} leading to a large decrease in classification accuracy. The inverse behaviour is demonstrated for $\alpha_{aT} = 0.25$. The rule set obtained for $\alpha_{aT} = 0.75$ demonstrated still different behaviour. Setting $\alpha_{aI} < 0.75$ resulted in very bad performance, while the performance was influenced less for $\alpha_{aI} > 0.75$. This behaviour may be explained by considering the effect that overlapping membership functions have on the induction process. For $\alpha_{aT} = 0.25$ instances can belong to more than one fuzzy set with non-zero membership. For $\alpha_{aT} = 0.75$ no instance belongs to more than one fuzzy set. The induced rule sets for the Auto data vary widely depending on the choice of α_{aT}. For $\alpha_{aT} = 0.75$ the rule set is specialized to use fuzzy sets whose domain cover only a certain region, while excluding all others. A decrease in α_{aI} then increases the domain of such fuzzy sets, allowing them to cover more negatives, and results in a decrease in performance. For $\alpha_{aT} = 0.25$ there is much overlap between fuzzy sets during rule induction. Fuzzy sets covering negative instances are excluded until these regions of the domain are not covered anymore. If α_{aI} is raised, no extra negative instances (from neighbouring terms) are covered, while most positive instances are still covered, resulting in relative insensitivity to α_a. The results for $\alpha_{aT} = 0.5$ is a mixture of both behaviours, resulting in sensitivity to either an increase or decrease in the post-induction value of α_a. The Labor data also showed some sensitivity to post-induction variation of α_{aI}, with best results obtained for training with $\alpha_{aT} = 0.75$ and inference with $\alpha_{aI} < 0.2$.

The main conclusion from this experiment is that there is no universal behaviour exhibited by all data sets. For some data sets best results are obtained for $\alpha_{aI} = \alpha_{aT}$, while for other data sets either a bigger or smaller value of α_{aI} results in better classification performance. Since rule inference is not very expensive (in contrast to rule induction), some experimentation with different values of α_{aI} is recommendable, as at least in some cases improved performance may be obtained. The experiment was performed with high frequency variation in α_{aI} (steps of 0.01). However, for most data sets there is not a very fast variation in the classification performance, and for most data sets ten experiments (varying α_{aI} in steps of 0.1) are sufficient to discover the best value or range of values of α_{aI}.

5.6.2 Training Sensitivity to α_a

In the previous section we examined the sensitivity of the rule set to α_{aI}. In this section we investigate the influence of the choice of α_{aT} on the induction process with respect to classification performance, rule set complexity, and search effort. As in the rest of this dissertation except where explicitly stated otherwise, we set $\alpha_{aI} = \alpha_{aT}$. Figure 5.9 shows the classification accuracy, rule set complexity, and the search effort for ten different data sets as α_{aT} is varied from 0.05 to 0.95 in steps of 0.05.

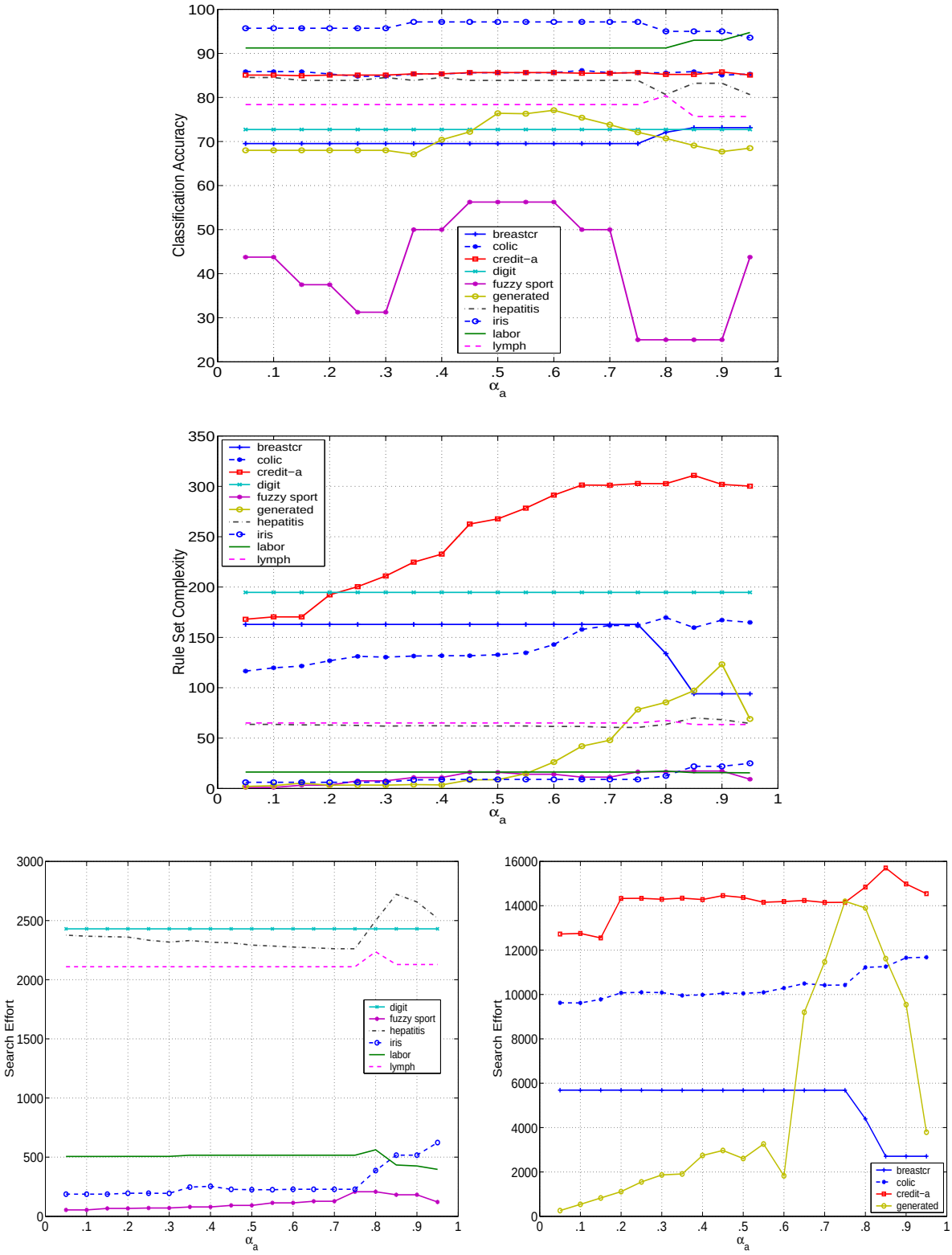


Figure 5.9: Rule set accuracy, complexity, and search effort for different values of α_{aT}.

The Digit data set has no linguistic variables with fuzzy sets (i.e. obtained from numerical data), and is thus completely independent of the choice of α_{aT}. The Fuzzy Sport data set contains only fuzzy data, and the membership degrees were not obtained from membership functions. This data set accordingly showed the largest classification accuracy sensitivity to α_{aT}. This sensitivity is also partly due to the very small size of the data set (16 instances), since the coverage of a single instance has a relatively large influence on overall performance. Contrasting with the high sensitivity in classification accuracy, the rule set complexity and search effort sensitivity of the Fuzzy Sport data were much less sensitive to the choice of α_{aT}.

Except for the Generated data, the remaining data sets did not show large variation in classification accuracy as α_{aT} was varied. However, even small increases in classification performance can be significant (and difficult to achieve), and the absolute variation is thus not the only consideration—a good choice for α_{aT} can benefit the classification performance. Another observation is that for no value of α_{aT} did the classification performance complete deteriorate.

The rule set complexity for some data sets increased with an increase in α_{aT}, for some data sets it remained relatively constant, while for yet others it decreased. An increase in rule set complexity was often correlated with a decrease in classification accuracy. As the learner finds it harder to find good rules (rules with high positive coverage and low negative coverage), more rules are induced, increasing both rule set complexity and search effort. In general these rules are also less accurate—typically, less complex rule sets perform better. However, this is not necessarily true in general. The Credit-A data set for example maintained the same classification accuracy level, while showing a linear *increase* with α_{aT} in rule set complexity. Furthermore, even though the increase in complexity was linear, large parts of the search effort curve were flat. Increasingly more complex rule sets were induced for increasing α_{aT}, requiring generally the same search effort and obtaining the same accuracy over the whole spectrum.

The Generated data set demonstrated interesting behaviour that can be traced to the method of creating the data (see Section 5.5). The classification rules used to label the synthetic data used $\alpha_a = 0.6$. Thus, the best results are obtained for rule sets induced with $\alpha_{aT} = 0.6$, with a dramatic increase observed in both rule set complexity and search effort for $\alpha_{aT} > 0.6$. The important result here is that there is no easy procedure for choosing an optimal value for α_{aT}, as there was also not a clear indication of a generally good value for α_{aI}. A domain expert may well be in the position to choose these values, and further research on the issue may provide more automated procedures.

From the experiment we can make the following general conclusions. The classification performance of the induced rule sets are not as sensitive to α_{aT} as one might expect. However, a good choice of α_{aT} may in some cases provide a small but significant benefit. Furthermore, FUZZYBEXA does not perform extremely bad (compared to best classification performance) for any choice of α_{aT}. FUZZYBEXA was thus able to use the given information to induce good rule sets even under sub-optimal conditions. Thus, if α_{aT} happens to be a very bad value for one linguistic term, FUZZYBEXA does not use this term but rather use other terms to induce as good a rule set as is possible for the given of α_{aT}. However, this has an influence on the rule complexity and search effort—good choices for α_{aT} typically reduce rule set complexity as well as search effort.

5.6.3 $\alpha_{aT} - \alpha_{aI}$ Sensitivity Surface

Sections 5.6.2 and 5.6.1 investigated FUZZYBEXA's sensitivity to α_a during induction and post-induction, respectively. This section shows the result of varying both α_{aT} for rule induction and α_{aI} for post-induction inference in the form of a surface plot. We performed the experiment on the Iris, Lymph, Fuzzy Sport and Generated data sets. As expected, the results are very dependent on the data set, and we only provide brief remarks for each data set.

Iris

Figure 5.10 shows the relative insensitivity of the classifier to α_{aT} and α_{aI} for large parts of the surface. However, if α_{aT} is set higher than 0.8, the accuracy drastically decreases for all post-induction values of α_{aI}.

Lymph

The surface plot for the Lymph data set is similar to that of the Iris data, and shows relative insensitivity to both α_{aI} and α_{aT} for large parts of the surface. Induction values of α_{aT} larger than 0.8 again degrades performance. Raising the post-induction value of α_{aI} above 0.8 resulted in a strong decrease in performance of all induction values of α_{aT}.

Fuzzy Sport

The Fuzzy Sport data set is very sparse; it has only 16 instances, but 3 classes. As already observed in the previous sections, the choice for α_{aT} is important for good performance, and the data set is also very sensitive for post-induction varying of α_{aI}. The best choice of α_{aT} for induction lies at 0.6, with graceful degradation as α_{aI} is lowered.

Generated

This data set was obtained by generating random data and then labeling it using a given set of rules. The value of α_a was taken as 0.6. The surface plot shows that best performance is obtained for $\alpha_{aT} = \alpha_{aI} = 0.6$. Varying α_{aI} or α_{aT} in any direction decreased performance. FUZZYBEXA demonstrated a graceful degradation in performance as α_{aI} or α_{aT} was varied above and below 0.6, as may be expected for a truly fuzzy data set with adequate training instances.

5.7 The Effect of Stop Growth Measures

The purpose of the experiment described in this section was to explore the impact of FUZZYBEXA's various prepruning and efficiency criteria. The experiment compares the base performance to performance

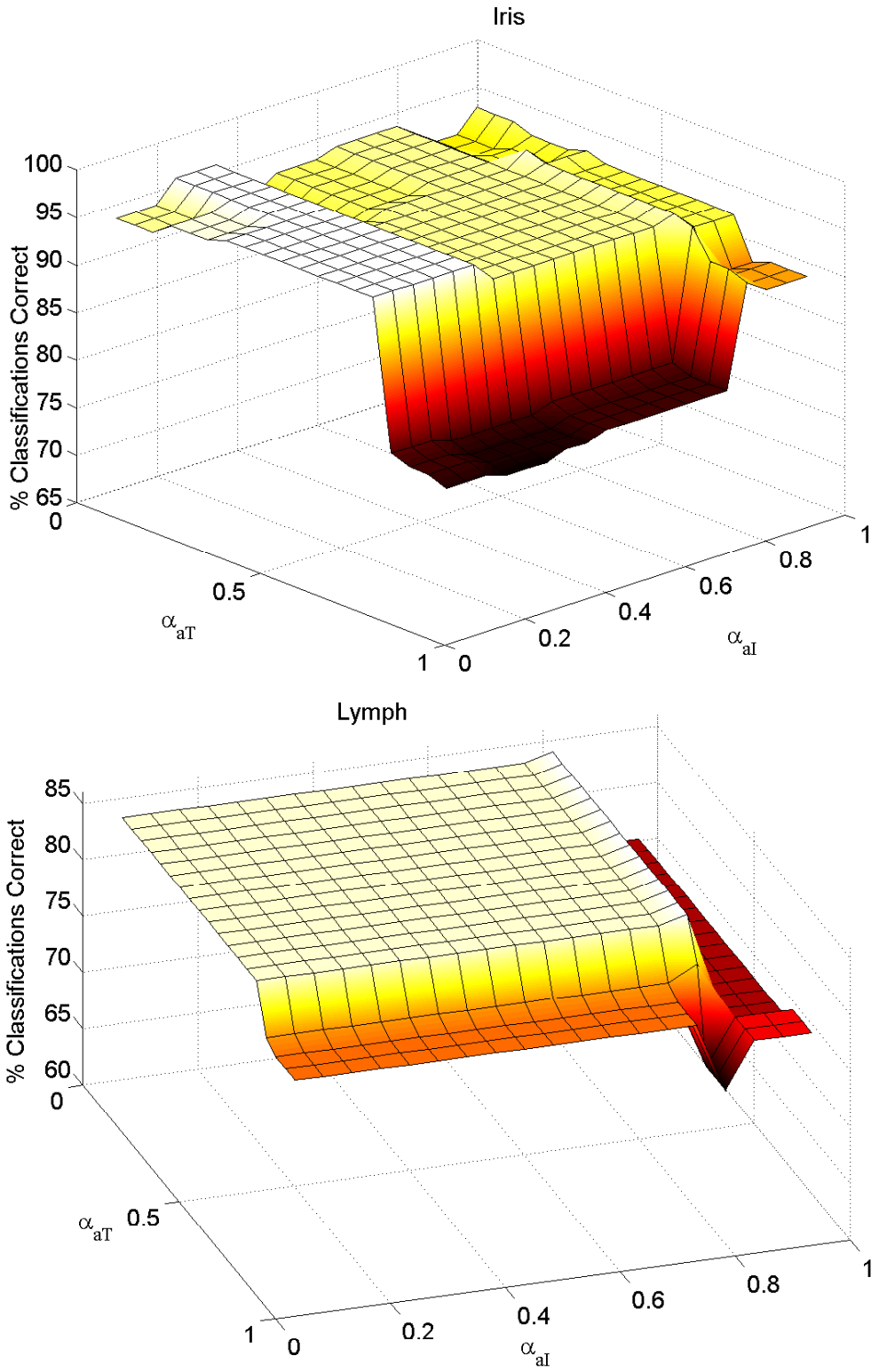


Figure 5.10: Classification accuracy surface for the Iris (top) and Lymph (bottom) data sets.

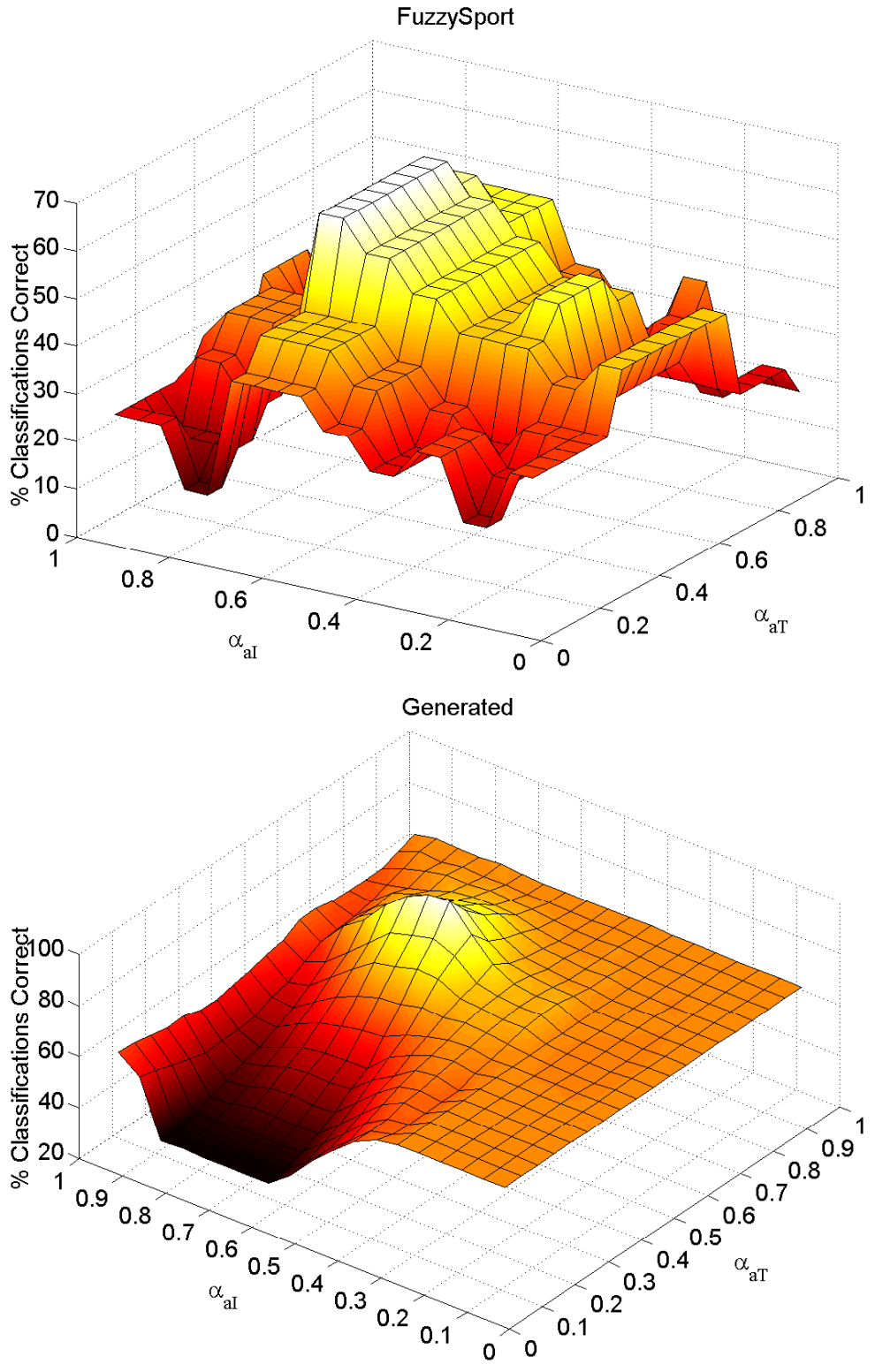


Figure 5.11: Classification accuracy surface for the Fuzzy Sport (top) and Generated (bottom) data sets.

obtained when adding the different search improvements. We performed the experiment on seven data sets using a beam search of width two, thus allowing duplicate conjunctions to be formed (see Section 4.6.5). The base result for each data set is obtained by applying no pruning or efficiency tests, except to stop the search process when either the positive or negative extension of a conjunction becomes empty. To obtain each result we only apply a single extra measure. Finally, we show the result when all measures are combined.

Table 5.6 shows the result of the experiment. The gray column shows the results of 10-fold cross validation for the classification accuracy, rule set complexity, and search effort on the different data sets if no tests are applied. The columns to the right of the base result show the percentage increase of the respective test on the base result. This is computed as,

$$\Delta_{\text{test}} = 100\frac{r_{\text{test}} - r_{\text{base}}}{r_{\text{base}}} \tag{5.3}$$

where r_{base} is the base result and r_{test} is the result for the respective test and data set.

The second column contains the result when using the optimistic evaluation efficiency measure. This test prunes conjunctions from the search when the conjunction cannot be improved to such a degree that its evaluation will be higher than that of the current best conjunction (see Section 4.5.3). Thus, we expect in most cases that there will be no change to the rule set (i.e. both accuracy and complexity). The test significantly influenced the search effort for four of the data sets, while having a moderate effect on the remaining three. The search effort for the Fuzzy Sport data set, for example, was reduced by 20%, and for the Lymph data set the reduction was 15.7%. It may seem strange at first glance that there are any changes in the rule set at all. The small changes in the rule sets of two data sets are due to the specific implementation of FUZZYBEXA. The exclusion of certain parts from the search can change the order in which specializations are generated within a single execution of the bottom layer, and this order can play a role in determining the rule set (especially in the absence of the "improve rule" test discussed below). If two conjunctions have the same evaluation, the conjunction generated first is chosen, and when this happens with the best conjunction, the search process is influenced.

The third column shows the effect of the "improve rule" test (see Sections 4.5.1 and 4.5.2). This test is activated when two antecedents are equivalent based on the evaluation function. In this case, the test prefers the antecedent that covers more positive instances. If both cover the same number of positive instances, the antecedent that is least complex is preferred. This test positively influenced the classification accuracy of three data sets, while having a slight negative influence on only one data set. The test resulted in a reduction in rule set complexity of all data sets. The rule set complexity of the Fuzzy Sport, Hepatitis, and Iris data sets were significantly improved—reductions of 13.4%, 10.2%, and 12.4%, respectively, were obtained. In most cases the test did not influence the search effort. Slight reductions in the search effort for the BreastCr and Digit data were obtained.

We implemented BEXA's irredundancy test to examine its effect in the fuzzy case (see Section 4.6.4. Except for Iris data set, the irredundancy test had no influence on any data set. For the Iris data it resulted in a reduction of 61.4% in rule set complexity and reduction of 51% in search effort. It seems that the rule sets induced for the Iris data without any tests was overly complex. The Iris data can be classified

Table 5.6: Comparison of different prepruning and efficiency criteria. The gray column shows the absolute performances without any stop-growth criteria, while the other columns shows percentage change relative to the absolute performance. Increased classification accuracy and decreased rule set complexity and search effort are desirable.

	Without Any Tests	Optimistic Evaluation	Improve Rule	Irredundancy	Uncover Negatives	All Tests
Classification Accuracy						
breastcr	69.53	0.0	0.7	0.0	-0.2	0.0
digit	73.31	0.0	-0.2	0.0	0.0	-0.2
fuzzy sport	50.00	0.0	0.0	0.0	0.0	0.0
hepatitis	83.87	0.0	2.3	0.0	0.0	2.3
iris	93.57	0.0	0.0	0.8	0.0	0.8
labor	91.23	0.0	0.0	0.0	0.0	0.0
lymph	79.73	-0.8	0.8	0.0	0.0	0.0
Rule Set Complexity						
breastcr	260.6	0.0	-8.1	0.0	-0.6	-8.4
digit	232.8	0.0	-2.6	0.0	0.0	-2.6
fuzzy sport	11.9	0.0	-13.4	0.0	0.0	-13.4
hepatitis	68.5	0.4	-10.2	0.0	-0.9	-11.2
iris	23.3	0.0	-12.4	-61.4	-0.9	-63.9
labor	13.0	0.0	-0.8	0.0	0.0	-0.8
lymph	85.9	-0.3	-6.3	0.0	0.5	-6.1
Search Effort						
breastcr	125873	-1.6	-3.1	0.0	-89.6	-90.1
digit	18277	-0.1	-1.5	0.0	-73.6	-74.1
fuzzy sport	569	-20.0	0.0	0.0	-68.7	-68.8
hepatitis	65998	-14.3	0.0	0.0	-94.3	-94.2
iris	3159	-2.6	0.0	-51.0	-70.4	-83.1
labor	13365	-15.7	0.0	0.0	-95.3	-95.4
lymph	68168	-7.5	0.0	0.0	-93.5	-93.5

with a fairly simple rule set compared to the other rule sets. Without the irredundancy test too much specialization occurred, resulting in worse overall performance. The improve rule test, for example, reduced the rule set complexity of the Iris data more than for the other rule sets, except for the sparse FuzzySport data. For most data sets the irredundancy test did not change the search effort, and we can deduce that the test is not easily satisfied in the fuzzy case. Thus, in some cases the irredundancy test does improve performance, but its usefulness is limited compared to the crisp case.

The uncover new negatives test prevents overspecialisation by requiring that specializations cover less negatives than their ancestors. This test had a very small negative influence on the BreastCr classification accuracy, and also in general had very little impact on the rule set complexity. However, the test had a huge impact on the search effort. Reduction in search effort of 90% and more were obtained for four data sets, while the smallest reduction was obtained for the Fuzzy Sport data set (68.7%). While the test resulted in great improvement in search efficiency, its computational cost is relatively inexpensive.

The last column shows the result when all the tests are applied together. Large improvements in search efficiency is obtained for all the data sets, while the rule set complexity for most data sets were also significantly improved. Only the rule set complexity of the Labor and Digit data sets were only slightly improved. The overall impact on the classification accuracy was not as pronounced, with the best re-

sult obtained for the Hepatitis and Iris data sets. The classification accuracy of the Digit data was slightly reduced. In general, the empirical results speak largely in favour of FUZZYBEXA's stop growth criteria—classification accuracy is either maintained or increased, rule set complexity is often significantly improved, and the search effort is dramatically reduced.

5.8 Summary

In this chapter we provided an experimental evaluation of FUZZYBEXA. We investigated its different parameters and stop growth criteria. The experiments showed that the specific characteristics of each data set have more influence than any given parameter. Some data sets are sensitive to the value of the antecedent threshold used during induction (we denoted it as α_{aT}), while others show little sensitivity. However, it was more often than not the case that a wide range of good induction values of α_{aT} exist, and that FUZZYBEXA is thus not overly sensitive to α_{aT}. Similarly, some data sets are sensitive to post-induction variation of α_a (we denoted it as α_{aI}), while others demonstrated relative insensitivity. The experiments also showed that FUZZYBEXA's search complexity increase at worst linearly with increasing beam width. However, a small beam width is typically sufficient or even indicated for good performance. We also showed that FUZZYBEXA is capable of dealing with noise, demonstrating graceful degradation with increased additive noise. Finally, the results of experiments with FUZZYBEXA's various stop growth criteria confirmed their usefulness—the use of the criteria resulted in either increased or maintained classification performance, while rule set complexity and search effort were always improved.

CHAPTER 6

The Influence of the Evaluation Function

6.1 Introduction

Machine learning algorithms typically use an evaluation function to score the performance of hypotheses on a training set during the learning process, and to select the set of best candidates for further exploration. Thus, the performance and characteristics of the evaluation function as a search heuristic are very important, because they have a large impact on the performance of the learning algorithm on a particular data set. The evaluation function is an important determinant of rule quality because it selects the next best specialization at each step, and thus guides the search through the space of all possible conjunctions. It is therefore important to investigate its influence and to compare the behaviour of different evaluation functions. In this chapter we present three results: (1) the effect of novel evaluation functions adapted to the fuzzy set domain, (2) the search paths followed in description lattice, and (3) benchmark results for each evaluation function on different data sets. The layout of the chapter is as follows. Section 6.2 introduces several fuzzy rule evaluation functions, and Section 6.3 contrasts their behaviour by investigating the subset of the hypothesis space explored for each choice of specialization function. Section 6.4 provides an empirical comparison of the different measures, and Section 6.5 concludes the chapter.

6.2 Evaluation Functions

FUZZYBEXA's *FindBestConjunction* procedure was discussed in detail in Chapter 4. FUZZYBEXA searches for conjunctions by starting with the *mgc* and specializing it by excluding one term at a time in all possible ways. The new candidate conjunctions (specializations) generated in this way (specialization by exclusion) are ranked according to an *evaluation function*, and the best $beamwidth$ conjunctions are selected for further specialization. Thus, a general-to-specific search through a description lattice is performed. The evaluation function is clearly a very important factor in determining the success of the algorithm-conjunctions are either pruned or retained in the search based on their score by the evaluation function. Different evaluation functions are based on different heuristic ideas. Many different functions for evaluating and assigning a score to crisp rules have been proposed in the literature [Fürnkranz, 1999]. For some of these, fuzzy variations were designed and used to score fuzzy rules [Fertig et al., 1999; Yuan and Shaw, 1995]. We review these and propose three additional fuzzy evaluation functions. In the

Table 6.1: Notation for evaluation function definitions, where r is the rule IF A THEN B.

$\|P\|$	$\|N\|$	p	n
# Positive Instances	# Negative Instances	# Positive Instances Covered	# Negative Instances Covered
r	A	B	c
Candidate Rule	Fuzzy Antecedent	Fuzzy Consequent	Conjunction Describing A

Table 6.2: Summary of evaluation functions.

Name	**Function**	**Range**
Laplace Estimate	$L(r) = \frac{p+1}{p+n+\#classes}$	$[0, 1)$
Fuzzy Laplace	$F(r) = \frac{\sum_{i \in X_T(c)} \mu_{A \cap B}(i) - \frac{1}{2}}{M(T,c)}$	$[0, 1)$
Fuzzy ls-Content	$LSC(r) = \frac{\frac{p+1}{P+2}}{\frac{n+1}{N+2}} \cong \frac{p+1}{n+1}$	$[0, \infty)$
Fuzzy Accuracy Function	$A(r) = M(P, c) - M(N, c)$	$(-\infty, \infty)$
Fuzzy Purity	$P(r) = \frac{\sum_{u \in U} \min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)}$	$[0, 1]$
Fuzzy Information Content	$IC(r) = \log \frac{M(P,c)}{M(T,c)} = \log M(P, c) - \log M(N, c)$	$(-\infty, 0]$
Fuzzy Entropy	$E(r) = \frac{n}{p} \left(\frac{M(P,c)}{M(T,c)} \log \frac{M(P,c)}{M(T,c)} + \frac{M(N,c)}{M(T,c)} \log \frac{M(N,c)}{M(T,c)} \right)$	$(-\infty, 0]$

following sections we will use the notation shown in Table 6.1. Here, p, n, P, and N are integer numbers, where an instance is either covered with membership α_a or above, or not covered (membership 0).

Recall that $X_S(c)$ denotes the extension of the conjunction c in the set of instances S, i.e. all instances in the set S that match c with membership α_a or above (see Eq (4.17)). In the fuzzy case we describe the "number" of instances matched by the rule using the cardinality operator (also called the *sigma count*) as follows,

$$M(S, c) = \sum_{i \in X_S(c)} \mu_c(i) \tag{6.1}$$

We will use the sigma count to use the heuristic ideas already employed by some well-known "crisp" evaluation functions and derive rule evaluation functions that can be used to score fuzzy rules. Table 6.2 provides an overview of the functions.

6.2.1 The Entropy Function

The Entropy function stems from the physics domain of thermodynamics, where it is a measure of the order in a system. It is also used in Shannon's information theory as the measure of information in a random variable. It was originally used by the ID3 decision tree induction algorithm [Quinlan, 1986] to

choose the decision attribute at each node. In the crisp case the Entropy function for a rule r is given by,

$$E(r) = \frac{p}{p+n} \log \frac{p}{p+n} + \frac{n}{n+p} \log \frac{n}{n+p} \tag{6.2}$$

where p and n denote the numbers of positive and negative instances covered by the rule, respectively. For the induction of fuzzy decision trees many fuzzy versions of the entropy are proposed in the literature [Dong and Kothari, 2001; Guetova et al., 2002; Marsala, 1998; Boyen and Wehenkel, 1999; Mitra et al., 2002; Yuan and Shaw, 1995].

The Entropy function assigns higher scores to conjunctions with high class separation—a valid heuristic for choosing a linguistic variable for a decision node in a decision tree. However, the Entropy function alone is not appropriate for use with FUZZYBEXA since it does not take into account whether the majority of instances are positive or not. To assign higher scores to conjunctions with higher class separability and also higher positive coverage, we adapt the crisp version of the entropy to the fuzzy domain as follows,

$$E(r) = \frac{n}{p} \left(\frac{M(P,c)}{M(T,c)} \log \frac{M(P,c)}{M(T,c)} + \frac{M(N,c)}{M(T,c)} \log \frac{M(N,c)}{M(T,c)} \right) \tag{6.3}$$

This equation gives evaluations in the range $(-\infty, 0]$, with higher evaluations to better conjunctions. A conjunction that covers only positive instances will get a score of 0 and a conjunction that covers only negative instances will get a score of $-\infty$.

6.2.2 The Information Content Function

The Information Content function measures the amount of information contained in the classification of the covered instances [Fürnkranz, 1999]. Information Content was also introduced in Shannon's information theory and is closely related to the Entropy function—the entropy is a weighted average of the information content of the classes. The Information Content can also be thought of as a measure of the purity of a partition. It was originally used in the PRISM inductive learner [Cendrowska, 1987], and is given by

$$IC(r) = \log \frac{p}{p+n} \tag{6.4}$$

in the crisp case. Eq (6.4) is in fact the negative of the Information Content, so that better evaluations obtain larger scores.

The fuzzy version of the Information Content is expressed as [Guetova et al., 2002]

$$IC(r) = \log \frac{M(P,c)}{M(T,c)} = \log M(P,c) - \log M(N,c) \tag{6.5}$$

The range of the Information Content function is the same as that of the Entropy function above, $(-\infty, 0]$, with conjunctions covering only positives obtaining a maximum score of 0 and conjunctions covering only negatives obtaining a score of $-\infty$.

6.2.3 The Accuracy Function

In the crisp case the accuracy of a rule is evaluated as [Fürnkranz, 1999]

$$A(r) = \frac{p + (N - n)}{P + N} \cong p - n \tag{6.6}$$

We fuzzify the Accuracy function by using the sigma count operator as follows,

$$A(r) = M(P, c) - M(N, c) \tag{6.7}$$

This function favours high coverage by using the difference in positive and negative instances covered, regardless of the absolute magnitude of the positive and negative sets. It scores rules in the range $(-\infty, \infty)$, with higher scores given to better evaluations. The approximation made in the crisp case remains valid in the fuzzy case, since the sigma counts of the positive and the negative instances are constant for any given training set.

6.2.4 The Laplace Estimate

The Laplace estimate [Fürnkranz, 1999] is given by,

$$L(r) = \frac{p + 1}{p + n + \#classes} \tag{6.8}$$

In its current form, FUZZYBEXA learns multi-class concepts by learning one class at a time. Thus, $\#classes$ is always 2. The Laplace estimate assigns higher scores to conjunctions with higher coverage of the positive instances. Conjunctions with low coverage are penalized—if the ratio of positive to negative instances covered is the same, conjunctions with lower absolute coverage will have a lower score.

6.2.5 The ls-Content Function

The ls-Content evaluation function was used in its general form in the algorithm HYDRA [Ali and Pazzani, 1993]. It is given by

$$LSC(r) = \frac{\frac{p+1}{P+2}}{\frac{n+1}{N+2}} \cong \frac{p + 1}{n + 1} \tag{6.9}$$

The function divides the Laplace estimate of the conjunction with the Laplace estimate of the training set. The denominators $P + 2$ and $N + 2$ stay constant in each iteration of the *FindBestConjunction* procedure, and can thus be ignored without altering the behaviour of the algorithm. We fuzzify the ls-Content function by using the sigma count operator,

$$LSC(r) \cong \frac{M(P, c) + 1}{M(N, c) + 1} \tag{6.10}$$

The function scores rules in the range $[0, \infty)$, giving higher scores to better rules.

6.2.6 The Purity Function

The purity of a rule is the percentage of positive instances among the instances covered [Fürnkranz, 1999],

$$P(r) = \frac{p}{p+n} \tag{6.11}$$

The function assigns a value of 1 to rules that cover no negative instances. However, low coverage is not penalized. A fuzzy variant of this function was introduced for fuzzy decision tree induction [Yuan and Shaw, 1995]. The fuzzy rules induced by FUZZYBEXA are propositions of the form $A \rightarrow B$. Although fuzzy implication can be implemented in different ways, the subsethood operator $S(A, B)$ is often used. Thus, the implication $A \rightarrow B$ holds true with degree $S(A, B)$ [Yuan and Shaw, 1995],

$$S(A, B) = \frac{M(A \cap B)}{M(A)} = \frac{\sum_{u \in U} \min(\mu_A(u), \mu_B(u))}{\sum_{u \in U} \mu_A(u)} \tag{6.12}$$

where $M(A)$ is the cardinality of the fuzzy set A, and $\mu_A(u)$ is the membership to the fuzzy set A of u, an element of the universe of discourse U, $u \in U$. For rule r with A the rule antecedent and B the rule consequent, $M(A \cap B)$ is the fuzzification of the number of positive instances covered p, and $M(A)$ is the fuzzification of the number of instances covered by the rule $n + p$, and we can define the Purity function $P(r)$ in the fuzzy case simply as $P(r) = S(A, B)$.

6.2.7 The Fuzzy Laplace Estimate

A fuzzy evaluation function related to the Laplace estimate was used in reference [Fertig et al., 1999],

$$F(r) = \frac{\sum_{i \in X_T(c)} \mu_{A \cap B}(i) - \frac{1}{2}}{M(T, c)} \tag{6.13}$$

This function favours rules with higher coverage. Consider for example the two cases $M(A) = M(A \cap B) = 100$ and $M(A) = M(A \cap B) = 1$. In the former case a score of 0.995 will be assigned, and in the latter a score of 0.5. The Purity function on the other hand would assign a score of 1 in both cases.

6.3 Paths Through the Lattice

Table 6.3 contains a small toy problem with linguistic variables A and B with linguistic term sets $\{f, g\}$ and $\{x, y\}$, respectively. The membership values of instances for each term are listed in the two columns in the table. For clarity sake, the membership to the concept for this problem is crisp. The data were obtained by randomly generating membership values, and then assigning only instances for which the following two rules fire for *class.yes*,

$$[f][x]@0.5 \rightarrow yes$$
$$[g][y]@0.5 \rightarrow yes$$

where we used $\alpha_a = 0.5$.

Table 6.3: A fuzzy learning problem.

```
@relation smallproblem
@attribute A     {f, g}
@attribute B     {x, y}
@attribute class {pos, neg}
@DATA
```

(.37 .49), (.84 .99), (0 1)	(.39 .46), (.49 .11), (0 1)
(.19 .82), (.41 .31), (0 1)	(.12 .82), (.73 .03), (0 1)
(.75 .63), (.86 .70), (1 0)	(.61 .45), (.37 .98), (0 1)
(.09 .67), (.88 .97), (1 0)	(.30 .25), (.34 .16), (0 1)
(.19 .26), (.83 .07), (0 1)	(.36 .73), (.58 .64), (1 0)
(.71 .85), (.69 .75), (1 0)	(.48 .17), (.83 .83), (0 1)
(.69 .06), (.70 .85), (1 0)	(.13 .30), (.08 .90), (0 1)
(.61 .70), (.77 .26), (1 0)	(.46 .26), (.49 .48), (0 1)
(.10 .49), (.77 .32), (0 1)	(.64 .54), (.10 .08), (0 1)
(.10 .21), (.23 .45), (0 1)	(.19 .53), (.79 .87), (1 0)

The search process, using the seven different evaluation functions, is illustrated in Figure 6.1. The part of FUZZYBEXA's lattice of conjunctions generated for an infinite beam width is shown. Some conjunctions were pruned from the search by FUZZYBEXA's efficiency measures, e.g. no conjunctions with empty positive extensions are shown. We have also omitted the most specific element for brevity. Each node in the graph shows the two conjuncts of a conjunction in the first two lines. The third line and fourth lines contain the scores assigned to a conjunction by the evaluation functions Fuzzy Laplace $F(r)$, Purity $P(r)$, Accuracy $A(r)$, Information Content $IC(r)$, ls-Content $LSC(r)$, Entropy $E(r)$, and Laplace Estimate $L(r)$, in this order. The last line contains the number of positive and negative instances covered by the conjunction. For the purposes of the discussion here we will denote conjuncts by listing their linguistic terms, e.g. $[f, g]$ is meant to mean $[Atextis f \vee g]$. Since all the linguistic terms have unique names there can be no confusion which linguistic variable is implied.

From the figure we observe that during the first few iterations of *FindBestConjunction* the different evaluation functions all have the same behaviour. They all prefer conjunctions that cover many positive instances and few negative instances. At the bottom layers fewer instances are covered and the different heuristics of the evaluation functions play a bigger role. The circled nodes and bold edges in Figure 6.1 show the path followed by FUZZYBEXA for the different evaluation functions and beam width one. We discuss the behaviour by individual evaluation functions next.

6.3.1 The Laplace Estimate

The exclusion of $A.\bar{\alpha}$ in the first iteration gave the highest score, and since there is no beam search, only this conjunction is specialized further. Two more specializations are made to obtain the conjunction $[f, g][x]$. The Laplace evaluation is shown as the last number in the fourth node line, and $[f, g][x]$ obtains a score of 0.8. It will be specialized further since $X_N(c) \neq \emptyset$. By excluding g from $[f, g][x]$ the final conjunction $[f][x]$, which has an empty negative extension, is obtained. Thus, the best rule returned

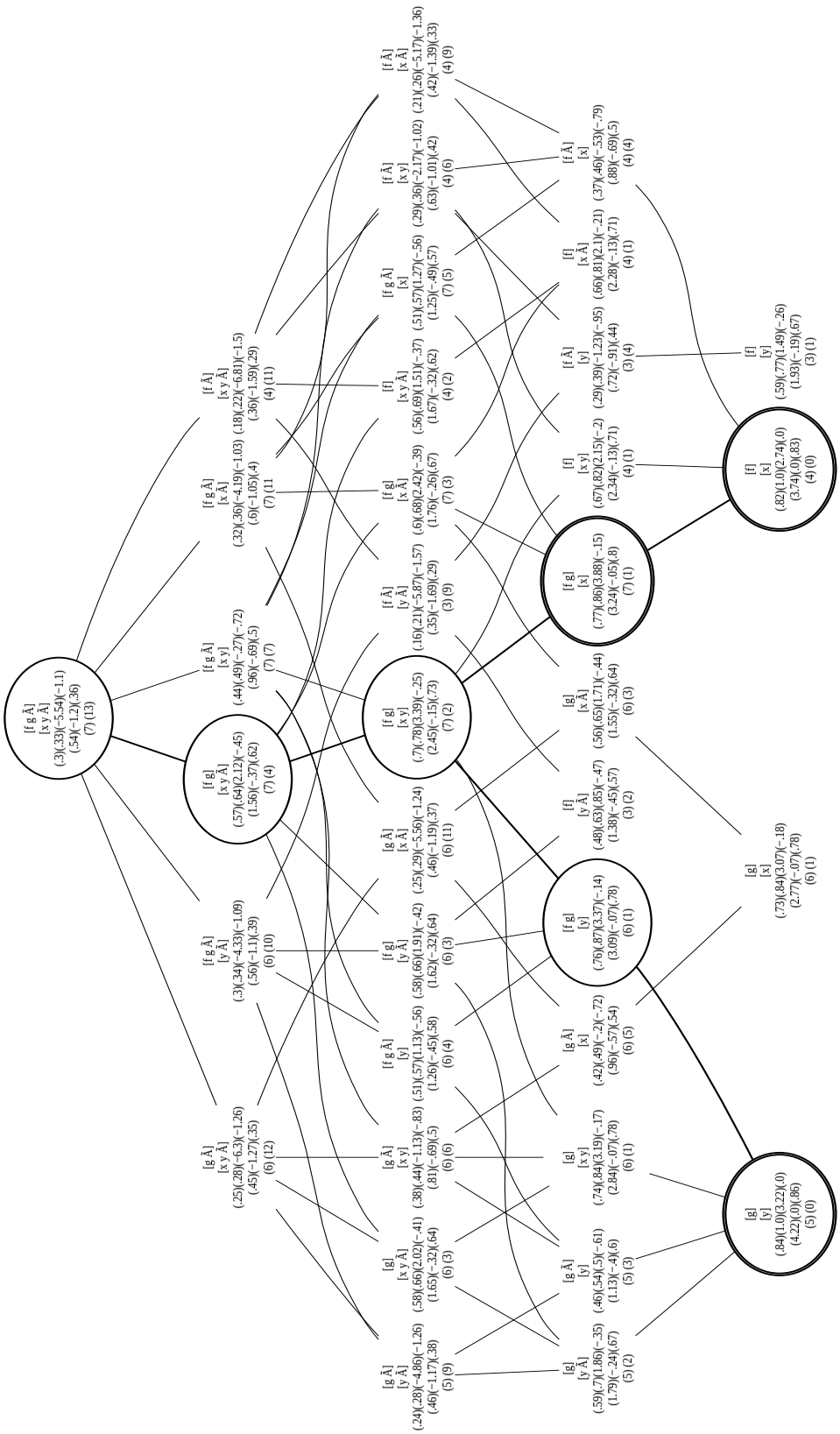


Figure 6.1: The lattice of the learning problem of Table 6.3.

when using the Laplace estimate will be

$$[f][x]@0.5 \rightarrow pos$$

Solution nodes (i.e. the node of the conjunction returned by *FindBestConjunction*) are indicated by bold circles on the figure. It is interesting to note that with a beam width of 2, the conjunction $[g][y]$ would have been returned, since it has a higher score. This conjunction, however, was excluded from the search (with beam width 1) by the exclusion of the linguistic term y higher up in the lattice.

6.3.2 The Purity Function

The score of the Purity function is given by the second number of the third line of each node. The Purity function dictates the same path as the Laplace function for the first three layers. If only the positive and negative coverage counts p and n are considered, the conjunction $[f, g][y]$ in the fourth layer chosen by the Purity function is worse than $[f, g][x]$, chosen by the Laplace estimate—one less positive instance and the same number of negative instances are covered. Interestingly enough then, in the bottom layer the solution conjunction covers more positive instances and also zero negative instances, which is better than that obtained by the Laplace estimate. Thus, the Purity function returns the rule,

$$[g][y]@0.5 \rightarrow pos$$

6.3.3 The Fuzzy Laplace Estimate

The score of the Fuzzy Laplace function $F(r)$ is shown as the first number of the third line of each node. The exact same path as that of the normal Laplace function is followed. The subtraction of the half in Eq (6.13) resulted in a score of 0.77 and 0.76 for the conjunctions $[f, g][x]$ and $[f, g][y]$, respectively. Just the opposite behaviour was observed for the Purity function where 0.86 and 0.87 were observed. Note that the evaluation assigned by the Fuzzy Laplace function may be negative for some conjunctions. This happens when $M(P, c) < \frac{1}{2}$. A negative value cannot occur for crisp sets.

6.3.4 The Information Content Function

The Information Content evaluation is shown as the last number in the third line of each node. Like the other functions, it follows the same path for the top three layers of the lattice. In the fourth layer it prefers $[f, g][y]$. The fuzzy behaviour of the evaluation functions become apparent when one observes that the fuzzy evaluation functions assign different scores to the conjunctions $[g][x, y]$, $[f, g][y]$, and $[g][x]$ (in the bottom two layers) even though they all cover one negative and six positive instances. A crisp evaluation function cannot distinguish between these conjunctions—the Laplace function for example assigned the same score to all of them. The Information Content function assigns a higher score to $[f, g][y]$ than to $[f, g][x]$, even though both cover one negative instance, but $[f, g][x]$ covers one more positive than $[f, g][y]$. Finally, the Information Content function finds $[g][y]$, and since it covers no negative instances, the Information Content gave it the highest score (zero).

6.3.5 The ls-Content Function

The ls-Content evaluation function is shown as the first value of the fourth line of each node. In the fourth layer of the graph the ls-Content function prefers $[f, g][x]$, and then in the bottom layer it prefers $[f][x]$. Thus, the same behaviour as for the Laplace estimate is observed. Similarly also, the conjunction $[g][y]$ would be returned if a beam width of two was used, since it performs better than $[f][x]$.

6.3.6 The Entropy Function

The Entropy function (shown as the second number in the fourth line of each node) preferred $[f, g][x]$ in the fourth layer of the lattice, and also found $[f][x]$. Note that if there was another conjunction in the bottom layer that covered one positive and no negative instances, the entropy and Information Content functions would assign zero scores to both, and could therefore not intelligently choose between its current choice and this conjunction. The ls-Content function on the other hand would still prefer $[f][x]$.

6.3.7 The Accuracy Function

The Accuracy function $A(r)$ (shown in the third position of the third line of each node) prefers the same conjunctions as the other evaluation functions in the first three layers. In fact, all the methods followed the same path in the first three layers of the lattice. This is indicative thereof that the different evaluation functions prefers the same macro features, but specialize differently as the number of instances become small—as happens lower down in the lattice.

Of the four conjunctions formed by specializing $[f, g][x, y]$, the conjunction $[f, g][x]$ obtains the highest score of 3.88. From this conjunction, either f or g can be excluded to form $[f][x]$ and $[g][x]$ with scores 2.74 and 3.07, respectively. Neither of these conjunctions score higher than $[f, g][x]$, and since they cannot be specialized further, the best rule found by using the Accuracy function is

$$[f, g][x]@0.5 \rightarrow pos$$

This rule still covers one negative instance, but it also covers all the positive instances, whereas the rules returned by the other functions do not. This happens because the Accuracy function places equal importance on the positive and negative instances, whereas the other functions emphasizes positive coverage. The Accuracy function kept $[f, g][x]$ since $[g][x]$ covered the same negative instances, but one less positive instance, and $[f][x]$ uncovered one negative instance at the cost of uncovering three positive instances. In this sense one can say that the Accuracy function has some form of stop growth functionality built in.

Whether this type of stop growth functionality is beneficial depends on the data. The Accuracy function could not make a perfect cover of the positive sets. Using the other functions, further iterations of the algorithm will induce rules that cover the remaining positive instances, possibly without covering any negative instances. The behaviour of the Accuracy function will be beneficial when a data set contains noise, since the Accuracy function has a built in preference for more general but still good conjunctions

Table 6.4: Accuracy results for different evaluation methods on real world domains.

		Anneal	Autos	BrCancer	Colic	Credit-A	Hepatitis	Iris	Labor	Lymph	Average
L(r)	Mean	**99.22**	**74.13**	70.00	84.51	83.91	82.58	**97.14**	89.47	**81.76**	84.75
	StdDev	*1.23*	*11.60*	*6.69*	*6.20*	*4.71*	*9.02*	*4.99*	*12.94*	*11.24*	*7.62*
F(r)	Mean	98.89	70.15	70.58	84.24	85.22	**83.87**	96.43	89.47	**81.76**	84.51
	StdDev	*1.37*	*8.28*	*3.88*	*6.28*	*5.54*	*9.62*	*5.05*	*12.94*	*14.40*	*7.48*
LSC(r)	Mean	98.89	70.65	69.19	84.51	85.22	83.23	96.43	91.23	**81.76**	84.57
	StdDev	*1.37*	*10.95*	*5.43*	*5.20*	*4.62*	*10.29*	*5.05*	*13.84*	*12.52*	*7.70*
A(r)	Mean	94.32	73.63	**73.14**	**85.60**	**85.65**	80.65	95.71	**92.98**	81.08	84.75
	StdDev	*3.06*	*10.70*	*4.92*	*4.30*	*6.28*	*12.26*	*6.02*	*14.44*	*11.48*	*8.16*
P(r)	Mean	98.11	67.16	59.77	79.89	82.17	78.06	96.43	91.23	79.73	81.39
	StdDev	*1.53*	*11.06*	*4.18*	*7.81*	*6.77*	*10.73*	*5.05*	*13.84*	*15.21*	*8.46*
IC(r)	Mean	98.11	67.66	59.53	79.89	82.17	78.06	96.43	91.23	79.73	81.42
	StdDev	*1.53*	*10.28*	*3.95*	*7.81*	*6.77*	*10.73*	*5.05*	*13.84*	*15.21*	*8.35*
E(r)	Mean	98.00	65.17	60.81	81.79	82.90	79.35	96.43	91.23	75.00	81.19
	StdDev	*1.96*	*10.34*	*5.29*	*7.63*	*4.52*	*8.63*	*5.05*	*13.84*	*16.69*	*8.22*

over more specific conjunctions that cover far less positives but also a small percentage negatives. The ls-Content function may be better suited for domains that contain concepts that can only be described by many small but significant disjuncts [Holte et al., 1989]. A final observation is that it is of course very possible that $A(r) < 0$, as is, for example, the case for the most general conjunction. To summarize, the Fuzzy Laplace, ls-Content, Laplace and Entropy functions returned $[f][x]$, the Purity and Information Content functions returned $[g][y]$, and the Accuracy returned $[f, g][x]$ as the best conjunction. However, in the top layers of the lattice all functions returned the same best conjunction.

6.4 Empirical evaluation

In this section we compare the different evaluation functions based on three different criteria, rule set classification accuracy, rule set complexity, and the number of conjunctions examined to obtain the rule set. We show results obtained on nine data sets obtained from the UCI Machine Learning Repository [Blake and Merz, 1998]. All results are averages on test set results of 10-fold cross validation.

6.4.1 Classification Accuracy

Table 6.4 shows the classification accuracy of the different evaluation functions. Bold numbers indicate the best performance for a specific data set among the methods, and italic numbers show the standard deviation. The last column contains the average performance over all data sets.

From the discussion in Section 6.3 we may expect that the Laplace, ls-Content, and Fuzzy Laplace functions would perform similarly and that the Information Content and Purity functions would be similar. Although the Entropy function found the same conjunction as members of the Laplace group in Section 6.3, mathematically speaking it is more closely related to the Information Content group, and we may expect them to perform similarly. The Accuracy function seemed to be in a group of its own.

The classification accuracy results shown in Table 6.4 confirm some of our expectations. The "Information Content Group" obtained very similar results, and our expectation that the Entropy function actually

belongs to this group is also validated. The members of the "Laplace Group" also obtained similar results. The mean of the Accuracy function was closer to the mean of the Laplace group, although its standard deviation shows that it does not fit in this group as well as the other members. This suspicion is confirmed by inspecting the individual data set results—while the other members perform very similarly for all data sets, the Accuracy function had significantly different performance on the Anneal, Autos, BrCancer, and Hepatitis data sets.

It is also interesting to note that the simple Accuracy function often performed very well. For four of the nine data sets it had the best performance, and had only notably worse performance on the Anneal data. This indicates that its preference for general descriptions is often an effective heuristic. For the Annealing data the Accuracy function obtained 94.32% classification accuracy, while all of the other methods obtained very high classification accuracy—close to 100%. This means that the rules induced by the other methods to classify the training data also classified the test data very accurately. The Accuracy function's preference for more general conjunctions proved detrimental in this case, as conjunctions were often not specialized until they became consistent, and still covered some negative instances in the training set and thus also in the test set.

The Laplace function had the best performance for four data sets, and the Fuzzy Laplace only for one. The average performance of the Laplace and Accuracy functions were the same, but the Laplace function had a slightly better standard deviation. The mean performance of the ls-Content and Fuzzy Laplace lies within 0.2% of the mean performance of the Accuracy and Laplace functions, and it is difficult to say which one is the best on average. No member of the Information Content group obtained the best classification accuracy for any data set, and their performance is on average 3% worse than that of the Laplace group. However, the Information Content group outperformed the Laplace and Fuzzy Laplace functions for the Labor data, and the Accuracy function performed considerably worse than the Information Content group on the Anneal data. This shows that there is not a one-fit all solution to concept learning. The best result is obtained by using the evaluation function employing heuristics best suited for the kind of input data and concept to learn. In general the heuristics employed by the Laplace group seem to give better classification accuracy results than the Information Content group.

6.4.2 Rule Set Complexity

Complex rules are more difficult to understand and are frequently an indication of overfitting. We measure rule set complexity as the number of *conjuncts* in the rule set. The complexities for the different evaluation functions are shown in Table 6.5. Bold numbers indicate the smallest number of conjuncts per rule set for a particular data set. As may be expected from the preceding discussions, the Accuracy evaluation function produced the smallest rule sets on average (115 conjuncts versus 125 of the next best, the Fuzzy Laplace function). It also led to the shortest rule sets for five of the nine data sets.

It is interesting to note that although the Accuracy and Laplace functions obtained very similar classification accuracy results, it is clear from their respective rule set complexities that the rule sets differed considerably. Furthermore, the Laplace and Fuzzy Laplace functions also had very similar average clas-

Table 6.5: Complexity of the rule set for different evaluation methods on real world domains.

		Anneal	Autos	BrCancer	Colic	Credit-A	Hepatitis	Iris	Labor	Lymph	Average
L(r)	Mean	102.8	182.7	252.7	213.6	447.4	74.2	7.0	13.8	91.0	153.9
	StdDev	*5.5*	*8.4*	*20.8*	*15.3*	*35.9*	*5.1*	*1.6*	*2.0*	*7.6*	*11.4*
F(r)	Mean	**87.7**	**138.9**	198.2	175.4	369.2	**63.1**	7.0	14.7	76.5	125.6
	StdDev	*5.9*	*8.2*	*11.6*	*14.3*	*13.7*	*4.0*	*1.6*	*1.8*	*5.3*	*7.4*
LSC(r)	Mean	100.3	182.1	257.6	212.9	436.1	74.3	7.0	**13.6**	89.4	152.6
	StdDev	*8.3*	*11.2*	*17.5*	*14.5*	*23.4*	*7.4*	*1.6*	*2.1*	*8.4*	*10.5*
A(r)	Mean	123.7	171.6	**136.8**	**169.2**	**280.6**	63.6	**6.0**	16.3	**67.9**	115.1
	StdDev	*10.0*	*11.6*	*17.3*	*17.4*	*16.0*	*4.0*	*0.0*	*1.7*	*7.2*	*9.5*
P(r)	Mean	104.5	262.5	292.3	236.7	512.4	76.2	7.0	16.5	96.2	178.3
	StdDev	*7.6*	*17.0*	*23.8*	*21.1*	*22.8*	*7.0*	*1.6*	*3.1*	*7.3*	*12.4*
IC(r)	Mean	104.5	199.9	291.9	236.7	508.5	76.2	7.0	16.5	96.2	170.8
	StdDev	*7.6*	*11.2*	*23.8*	*21.1*	*26.3*	*7.0*	*1.6*	*3.1*	*7.3*	*12.1*
E(r)	Mean	103.0	197.6	287.8	233.6	507.2	77.1	7.1	15.9	92.8	169.1
	StdDev	*8.4*	*13.5*	*19.0*	*29.7*	*25.2*	*8.9*	*1.6*	*2.5*	*5.1*	*12.7*

Table 6.6: The number of conjunctions searched.

		Anneal	Autos	BrCancer	Colic	Credit-A	Hepatitis	Iris	Labor	Lymph	Average
L(r)	Mean	4993	17432	8039	14045	22174	2732	410	417	3018	8140
	StdDev	*284*	*1494*	*793*	*820*	*1560*	*145*	*58*	*53*	*266*	*608*
F(r)	Mean	**4443**	**14900**	7603	12094	20073	**2429**	369	433	2719	7229
	StdDev	*261*	*1379*	*603*	*1126*	*1045*	*199*	*48*	*59*	*168*	*543*
LSC(r)	Mean	4959	16827	7774	13944	21422	2716	289	**413**	2946	7921
	StdDev	*389*	*1412*	*566*	*819*	*1363*	*258*	*48*	*61*	*270*	*576*
A(r)	Mean	6132	15381	**4499**	**11176**	**14185**	2498	**196**	542	**2243**	6317
	StdDev	*384*	*1056*	*457*	*873*	*788*	*184*	*13*	*64*	*222*	*449*
P(r)	Mean	5152	26002	9931	16589	26853	2885	421	490	3286	10179
	StdDev	*348*	*1753*	*880*	*1421*	*1143*	*248*	*87*	*88*	*285*	*695*
IC(r)	Mean	5152	19653	9510	16589	26119	2885	286	490	3286	9330
	StdDev	*348*	*1514*	*817*	*1421*	*1216*	*248*	*48*	*88*	*285*	*665*
E(r)	Mean	5062	19376	9285	16205	25560	2975	301	477	3205	9161
	StdDev	*380*	*1733*	*716*	*2014*	*1461*	*283*	*37*	*73*	*200*	*766*

sification accuracy, but significantly different rule set complexities. The Fuzzy Laplace function resulted in an average reduction in rule set complexity of 18.3% compared to the (crisp) Laplace function. Thus, the fuzzification of the evaluation method had a beneficial effect on rule set complexity. Finally we note that the Laplace and ls-Content had similar rule set complexities.

The members of the Information Content group again obtained similar performances, with the Entropy method obtaining the best result. Another observation is that for the Anneal data, the accuracy method obtained the most complex rule set, and also had the worst accuracy performance. For this data set the Fuzzy Laplace obtained both the best classification accuracy and rule set complexity, while the remaining evaluation methods all had very similar rule set complexity results. We have shown emprically in Chapter 5 that rule set complexity is often correlated with rule set accuracy (cf. Section 5.5 and Section 5.6.2). This is further emphasized here by the results for the Accuracy function on the BreastCancer, Colic and Credit-A data sets.

6.4.3 Search Space Explored

The amount of search effort is quantified by counting the number of candidate conjunctions that were investigated during rule induction. The evaluation function effectively prunes unpromising conjunctions

from the search, and ranks the conjunctions according to its perceived classification performance. Table 6.6 shows the size of the search space explored to obtain the rules. The bold numbers indicate the smallest search effort per data set.

Again the Accuracy evaluation function was the most effective, generating on average approximately 6300 conjunctions, versus approximately 7200 for the Fuzzy Laplace, 7900 for the ls-Content, and over 8100 for Laplace evaluation function. It also caused the least search for five of the nine data sets. The reason is that the induction of each rule requires a new search of the lattice of rule descriptions, and the Accuracy function induced far fewer rules. Thus, it also comes as no surprise that the Fuzzy Laplace required the second least search.

In the Information Content group, Entropy and Information Content had similar results (~ 9200 conjunctions), while the Purity function required significantly more search (over 10100 conjunctions). Thus, the Entropy and Information Content functions obtained very similar results for classification accuracy, rule set complexity, and also search requirement, indicating that they basically employ the same heuristic. When this heuristic is required, the Information Content should be used since it less complex and therefore faster to compute. If we compare the average results for the Laplace and ls-Content functions we may come to a similar conclusion. However, comparing the classification accuracy results for the Autos data and the search requirement for the Iris data, for example, we see that the average results may sometimes hide some information, and that while the two methods are related, they employ different heuristics.

6.5 Summary

In this chapter we investigated the effect of the evaluation function on the induction process. We also provided an empirical evaluation of the different functions by comparing their performance with respect to classification accuracy, rule set complexity, and search requirements on nine real world data sets. Of the several evaluation functions discussed in this chapter, four were obtained from the literature, while we proposed the remaining functions [van Zyl and Cloete, 2004c; Cloete and van Zyl, 2004a].

By using a small example we demonstrated that the different evaluation methods all follow the same path through the lattice in the first few layers. This means that macro features are equally well distinguished by the different methods. Lower down in the lattice conjunctions cover fewer instances, and the individual characteristics of the different methods had a bigger influence. Of the different methods, only the Accuracy function often prefers conjunctions higher up in the description lattice. This happens because higher up in the lattice more instances are covered, and $M(X_P(c)) - M(X_N(c))$ is more likely to be big than lower down in the lattice where fewer instances are covered. The other methods all prefer more consistent conjunctions. This property of the Accuracy function often helps the algorithm to prevent overfitting, but in a few cases, e.g. for the Anneal data set, this property may also deteriorate performance.

The different evaluation functions could be divided into roughly three groups, with the Accuracy func-

tion in its own group. The Information Content group's overall performance seemed worse than that of the other groups. Thus, functions from this group should only be used in special circumstances when it is clear that such a function should perform well. As a rule of thumb, the Accuracy function seems to be a good choice for an evaluation function. It had the best overall performance, and resulted in the least complex rule sets while also requiring the least search effort (on average). The similar classification performance of members within a group could to some extend be attributed to FUZZYBEXA's search heuristics other than the evaluation function. Without these heuristics, the evaluation function will have a greater impact, and the results are likely to be more diverse.

CHAPTER 7

Comparison Between FUZZYBEXA and Other Fuzzy Rule Learners

7.1 Introduction

In the previous chapters we introduced FUZZYBEXA, an algorithm for the induction of fuzzy classification rules. We provided an empirical evaluation of the algorithm with respect to various parameters and characteristics. We also introduced several fuzzy rule evaluation functions, and showed the importance of the evaluation function. In Chapter 2 we reviewed several fuzzy concept learners and categorised them in seven classes of algorithms. We now ask the question, what are the differences and similarities of these algorithms to FUZZYBEXA. We make this comparison with respect to the different characteristics of FUZZYBEXA. Since space prohibits us from comparing FUZZYBEXA to all possible algorithms, we compare FUZZYBEXA with those main classes of fuzzy rule learners that are somewhat related: (a) inductive learners (b) decision trees (c) similarity search, (d) partitioning methods, and (e) stochastic search.

We do not pay much attention to gradient descent (neural network) methods, since these do not directly induce a fuzzy rule set, but a fuzzy rule set must be extracted using one of a variety of methods. The number of possible combinations of neural architecture, training methods, and rule extraction algorithms makes it very difficult to derive general characteristics for gradient descent methods. It suffices to say that they induce subsymbolic (connectionist) results, and have very little in common with FUZZYBEXA or even the other classes of rule induction methods. Note, gradient descent here refers to the direct induction of rules using a gradient descent method, and not the optimisation of certain parameters using an artificial neural network within a scheme that employs another method, e.g. the optimisation of defuzzification parameters via an artificial neural network within a fuzzy decision tree induction algorithm.

The layout of the chapter is as follows. In Section 7.2 we discuss the inductive bias of each of the different groups of concept learners. In the following sections we discuss the differences and similarities between FUZZYBEXA and other concept learners with respect to the description language, parameters and structure identification, evaluation function, beam search, lattice and partial ordering, rule set ordering, and stop growth measures. Section 7.4 concludes the chapter with a discussion.

7.2 Inductive Principle Comparison

We grouped the different fuzzy rule induction algorithms into classes based on the induction method employed. The basic search strategy is the first and most important distinguishing characteristic of a learning algorithm, and in this section we summarize the principles or heuristics on which the induction process of each class is based.

7.2.1 Fuzzy Inductive Learners

Inductive learners induce rules by identifying features that empirically distinguish positive from negative training examples [Mitchell, 1997]. We already formulated FUZZYBEXA's inductive bias in Section 4.8, and we restate it here briefly. FUZZYBEXA performs a separate-and-conquer search of the hypothesis space. It prefers conjunctions with good evaluations over conjunctions with bad evaluations, where "good" and "bad" are defined by an evaluation function. FUZZYBEXA's description language, FuzzyAL allows internal disjunction, and forms a lattice of antecedent descriptions, and a top-down, general-to-specific beam search of this description lattice is performed. Thus, FUZZYBEXA employs both a *search bias* (characterized as greedy, general-to-specific with pre-pruning) and a *language bias* (its description language).

7.2.2 Divide-and-Conquer Strategies

Fuzzy decision trees are the fuzzy generalization of classical decision trees, and employ linguistic variables at decision nodes and linguistic terms at branches. Typically a fuzzy form of the information theoretic measure entropy is used to obtain the fuzzy information gain. The inductive bias of fuzzy decision tree induction is therefore strongly related to the inductive bias of the classical algorithm. Mitchell describe this inductive bias as [Mitchell, 1997], "Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not." If we exchange attribute with linguistic variable and information gain with fuzzy information gain, we obtain the inductive bias for the fuzzy ID3 algorithm [Cios and Sztandera, 1992; Dong and Kothari, 2001]. Although this definition was primarily derived for ID3, with minor alteration it is also applicable to many other fuzzy decision tree induction algorithms.

7.2.3 Similarity Search

Similarity search relies on some kind of *closeness* or *representativeness* assumption, and the concept of *distance* or *similarity* plays an important role [Dubois et al., 2002]. After clustering the input or sometimes the output domains, the cluster centres are typically used to represent a rule, and the different individual instances are discarded. Since a cluster centre is a single point in the input-output space, these methods typically only induce *purely conjunctive* rules, and the description language is not specified beforehand, but induced from the clustering process. There is no evaluation function employed during

the search (other than the distance measure), and no general-to-specific ordering. We can thus say that similarity search methods are biased towards solutions where instances are grouped such that they lie close to other instances of the same group. From these groups rules and/or membership functions are deduced.

7.2.4 Stochastic Search

The inductive principle on which evolutionary fuzzy rule learners are based is a combination of the rule encoding method as well as in the general inductive bias of genetic algorithms. In some systems the rule encoding method restricts the description language (language bias) to allow only certain representations, e.g. the conjunction of all input domains where each input domain is divided into three evolving fuzzy sets. Other methods place no restriction on the representation language and allow arbitrary rules. The population evolution over time within a genetic algorithm has been described using Holland's schema theorem [Holland, 1975], which roughly interpreted states that better schemas (gene sections) tend to grow in importance over time [Mitchell, 1997]. Genetic algorithms in general may employ a language bias as well as a learning bias [Whigham, 1995].

7.2.5 Partitioning Methods

Most partitioning methods make very few assumptions overall, and as a result usually produce very large rule sets. They typically proceed by dividing each input dimension into partitions, resulting in the division of the input space into fuzzy hyperrectangles, where the centre of the hyperrectangle is most characteristic of the hyperrectangle. Each hyperrectangle forms the antecedent of a *purely conjunctive* rule. The available instances are assigned to their respective hyperrectangles, and the rule consequent of the respective hyperrectangles are then determined by the output domain of the instances. The main task of the algorithm is to resolve conflicting rules. The bias of the algorithm is then determined by the method of resolving the conflicts. A typical example is to prefer the rule that matches the instances to the highest degree [Hong and Chen, 1999; Wang and Mendel, 1992].

7.3 Characteristic Comparison

FUZZYBEXA is characterized by a variety of features. In this section we compare FUZZYBEXA to algorithms with respect to these features. Some features are unique to FUZZYBEXA, while others are also found in other learners.

7.3.1 Description Language

FUZZYBEXA employs FuzzyAL as its description language. Descriptions in FuzzyAL are formed by the conjunction of several conjuncts, $C_1 \wedge C_2 \ldots \wedge C_n$. Each conjunct uses linguistic terms from the same

linguistic variable, and may be *internally disjunctive*, $C_i = c_1 \vee c_2 \ldots c_m$. Except for the method by Castro *et al* [Castro et al., 1999] (extended by Carmona *et al* [Carmona et al., 2004]), all other methods surveyed *do not* allow internal disjunction, and form purely conjunctive rules.

7.3.2 Parameter and Structure Identification

Parameter identification refers to the identification of system parameters like membership functions. Structure identification refers to the identification of the structure of the classifier, i.e. the rule structure. FUZZYBEXA does not perform parameter identification, since many methods already deal with that. FUZZYBEXA is purely concerned with structure identification, and as such is capable of inducing *incomplete* rules , i.e. rules that do not use all variables in the antecedent description.

Similarity search techniques typically do not implement any structure identification, and is only concerned with parameter identification. They induce complete rules, i.e. rules that contain all linguistic variables in the antecedent. Most partitioning techniques are also solely concerned with parameter identification, inducing complete rules. There are some exceptions, where optimisation methods such as genetic algorithms are used to identify important and unimportant fuzzy sets in the rule base [Ishibuchi et al., 1995]. Genetic algorithms have in some cases been used to evolve membership functions only, assuming purely conjunctive complete rules [Wang and Bridges, 2000], and in other cases to evolve both the rule set and rule base [Peña-Reyes and Sipper, 2001]. Fuzzy decision trees may be converted into an equivalent set of fuzzy rules. Similar to FUZZYBEXA, the main task of fuzzy decision trees is to perform structure identification, and thus they can induce incomplete rules.

7.3.3 Evaluation Function

In Chapter 6 we introduced several fuzzy evaluation functions that can be used to rank fuzzy rules. The rule (antecedent) evaluation function is an integral part of FUZZYBEXA, and FUZZYBEXA's performance can in some cases be greatly influenced by the choice of the evaluation function.

In genetic algorithm optimisation of the fuzzy rule base, the objective function may be seen as a kind of evaluation function. This objective function, however, typically operates on the whole rule set, and does not have the same function as in FUZZYBEXA. Decision tree induction usually employs an information theoretic method to decide which linguistic term to use at each decision node. This function also does not operate on a single rule. Similarity search techniques, partitioning methods, and gradient descent methods do not use an evaluation function. In fact, only algorithms in the class of fuzzy inductive learning employ an evaluation function for guiding search in the sense that FUZZYBEXA does.

7.3.4 Beam Search

FUZZYBEXA can perform a beam search of the hypothesis space. One may consider the population size used in genetic algorithms as a kind of beam width. Other than references [Fertig et al., 1999;

Wang et al., 2003] we did not find any methods employing a beam search.

7.3.5 Lattice and Partial Order

The descriptions in FUZZYBEXA's description language FuzzyAL is partially ordered and forms a lattice. This is exploited by FUZZYBEXA to perform a general-to-specific search, and to employ a variety of efficiency and pruning methods. We are not aware of any other work on fuzzy rule learning where this partial order is explicitly exploited.

7.3.6 Rule Ordering: Iterated and Simultaneous Concept Learning

We will introduce FUZZYBEXAII in Chapter 10, but for completeness of the comparison we mention it here. FUZZYBEXAII allows the induction of ordered rule sets through the use of simultaneous concept learning [van Zyl and Cloete, 2004f]. Simultaneous concept learning does not iterate through the list of concepts for which descriptions are desired, but assigns the concept during the induction process. This kind of induction process has not been used for fuzzy rule induction before. Although most of the other fuzzy learning techniques induce rules for the different concepts at the same time, these rules are unordered, and in many cases it would make no difference to the resultant rule set if the rules had been induced by iterating through the concepts.

7.3.7 Stop Growth and Efficiency Measures

FUZZYBEXA employs many different efficiency measures. These measures typically exploit the partial ordering of descriptions. For example, if the positive extension of a description is empty, it is clear that no further specialization will result in descriptions that have a non-empty positive extension. Another example is early stopping based on an optimistic evaluation. Here we assume that the negative extension of a description can be made empty, and the positive extension be kept unchanged through specialization. If this optimistic evaluation is still worse than that of the current best description, we remove the description (and thus all it's descendants) from the search. We have not encountered any fuzzy learning methods that employ this kind of stop growth and efficiency measures—i.e. making use of the partial ordering. In the literature several (early) stopping criteria were suggested for neural network training [Hayken, 1999]. However, these techniques cannot be seen as efficiency measures, but rather an attempt to prevent overfitting.

7.4 Discussion

Table 7.1 provides a comparative overview of representative examples of the different groups of fuzzy concept learning algorithms discussed in this chapter. The different comparison criteria only include those aspects where some other method also had this characteristic, i.e. we do not show "stop growth,"

Table 7.1: A table of comparison between FUZZYBEXA and a selection of other classification rule learning techniques.

	Structure Identification	Incomplete rules	Purely Conjunctive Rules	Internal Disjunction	General to Specific	Evaluation Function	Use Partial Order	Beam Search
FUZZYBEXA	✓	✓	✓	✓	✓	✓	✓	✓
Greedy Incremental Rule Construction								
Wang *et al* [1999]	✓	✓	✓	×	✓	✓	×	×
Fertig *et al* [1999]	✓	✓	✓	×	✓	✓	×	✓
Wang *et al* [2003]	✓	✓	✓	×	✓	✓	×	✓
Fuzzy Decision Trees								
Yuan *et al* [1995]	✓	✓	✓	×	✓	×	×	×
Chi *et al* [1996]	✓	✓	✓	×	✓	×	×	×
Dong *et al* [2001]	✓	✓	✓	×	✓	×	×	∗[2]
Similarity Search								
Klawonn *et al* [1997]	×	×	✓	×	×	×	×	×
Sugeno *et al* [1993]	✓	✓	✓	×	×	×	×	×
Setnes [2000]	✓	×	✓	×	×	×	×	×
Dubois *et al* [2002]	×	×	✓	×	×	×	×	×
Yin [2004]	×	×	✓	×	×	×	×	×
Hong *et al* [1996]	×	×	✓	×	×	×	×	×
Partitioning Methods								
Wang *et al* [1992]	×	×	✓	×	×	×	×	×
Pomares *et al* [2002]	✓	×	✓	×	×	×	×	×
Berthold [2003]	×	✓	✓	×	×	×	×	×
Nozaki *et al* [1996]	×	×	✓	×	×	×	×	×
Carmona *et al* [2004]	✓	✓	×	✓	×	×	×	×
Casillas *et al* [2000]	×	×	✓	×	×	×	×	×
Genetic Algorithms								
Reyes *et al* [2001]	✓	✓	✓	×	×	×	×	×
Herrera *et al* [1994]	✓	✓	✓	×	×	×	×	×
Ishibuchi *et al* [1995]	✓	×	✓	×	×	×	×	×
Ishibuchi *et al* [2004]	✓	✓	✓	×	×	×[1]	×	×

[1] The authors apply a prescreening method that examine short candidate antecedents scored using the subsethood function. An evaluation function, however, is not used to pick candidates for specialization.
[2] The authors employ a look-ahead strategy, which may be interpreted to some extent as a beam search.

for example, since no other method employs it in the manner FUZZYBEXA does. From the table it is clear that no other method implements all of FUZZYBEXA's features. Methods within the same group tend to have similar characteristics. The methods most closely related to FUZZYBEXA are those published in references [Fertig et al., 1999] and [Wang et al., 2003]. We discussed both these methods in detail in Chapter 2. In Chapter 9 we propose a general fuzzy set covering framework, as well as several specialization models, some which are based in part on these two algorithms.

CHAPTER 8

Inducing Fuzzy Conjunctive Rules: FUZZCONRI

8.1 Introduction

Up to now we have introduced one example of a fuzzy set covering algorithm, FUZZYBEXA. We have also proposed various additions and improvements to the basic covering algorithm, and have also investigated the influence of different rule evaluation methods. We now present another algorithm implementing the fuzzy set covering approach. This algorithm's description language is different compared to that of FUZZYBEXA—it induces conjunctive rules, and thus we call the algorithm FUZZCONRI, for **F**uzzy **Con**junctive **R**ule **I**nducer. Since FUZZCONRI is a fuzzy set covering algorithm, it implements most of the ideas presented in Chapter 4. The inspiration for FUZZCONRI comes from the crisp rule induction CN2 by Clark and Boswell [1991], and FUZZCONRI behaves exactly like CN2 in the special case when crisp data are used. Thus, FUZZCONRI can be seen as the fuzzy generalization and extension of CN2.

The layout of the remainder of the chapter is as follows. In Section 8.2 we propose a new description language, FuzzyCAL. Then we present FUZZCONRI, a fuzzy rule induction algorithm employing FuzzyCAL, in Section 8.3. Section 8.4 demonstrates FUZZCONRI's rule induction behaviour on a toy data set, and Section 8.5 concludes the chapter.

8.2 FuzzyCAL

FuzzyCAL (Fuzzy Conjunctive Attributional Logic) is a new description language that has no counterpart in the crisp case. Valid descriptions in FuzzyCAL may be formed by the conjunction of *any* set of linguistic terms in the problem space. Consider again the Fuzzy Sport problem in Table 4.1, examples of valid expressions are $[outlook.cloudy \wedge outlook.rainy]$ and $[outlook.sunny] \wedge [temp.mild \wedge temp.cold]$. Thus, FuzzyCAL does not have internal disjunction, but instead allows the conjunction of *arbitrary* linguistic terms. To make the descriptions easy to read, we group expressions of linguistic terms from the same linguistic variable together, and indicate this with square brackets. We also write a description in FuzzyCAL in short hand, for example the previous two expressions can be written as

$[cloudy, rainy]$ and $[sunny][mild, cold]$. Although the shorthand form of these expression look similar to shorthand expressions in FuzzyAL, their semantic interpretation in FuzzyCAL is different from that in FuzzyAL. The descriptions in FuzzyCAL form a lattice as follows.

Definition 8.2.1 Let c_1 and c_2 be two conjunctions in C, then $c_1 \succeq c_2$, c_1 *is more general than or equal to* c_2 if $D(c_1) \subseteq D(c_2)$, c_1 and c_2 are considered equal when $D(c_1) = D(c_2)$, and $c_1 \succ c_2$, c_1 *is strictly more general than* c_2 if $c_1 \succeq c_2$ and $c_1 \neq c_2$.

Thus, set C is partially ordered under the $\succeq$ relation and forms the lattice $\langle C; \succeq \rangle$. The top element of the lattice contains no elements in its description set, i.e. $D(mgc) = \emptyset$, and is defined to be semantically equivalent to TRUE. Note, the conjunct $[\,]$ is thus also equivalent to TRUE.

The alpha complement was added to each linguistic variable for descriptions in FuzzyAL. The reason for adding the alpha complement was that without its addition the *mgc* would not cover the entire instance space. However, since the *mgc* in FuzzyCAL is the conjunction TRUE, the alpha complement is not required for FuzzyCAL. It may still be added to describe sections of the domain of a linguistic variable where all membership functions are below α_a, but its addition is not a necessary requirement to form a valid *mgc*.

As is clear from Def 4.4.1 and Def 8.2.1, FuzzyAL and FuzzyCAL are syntactically related. Except for the alpha complement, the FuzzyAL and FuzzyCAL would form mirror lattices of each other for the same learning problem. That is, the description sets in the top layers of the FuzzyAL lattice would be the same as the description sets of the mirror bottom layers of the FuzzyCAL lattice. However, the expressions associated with the description sets differ between the description languages. We delay a more in depth comparison of FuzzyAL and FuzzyCAL to Section 9.7, after the introduction of the general fuzzy set covering framework.

8.3 FUZZCONRI

FUZZCONRI consists of two layers, an upper layer implementing the set covering approach to fuzzy rule induction, and a lower layer for inducing a single rule. The algorithm is shown in Table 8.1. The upper layer has the same functionality as FUZZYBEXA's top layer (for a discussion of the top layer refer to Section 4.5).

As discussed above, the FuzzyCAL rules induced by FUZZCONRI are of the form:

$$\text{IF } [temp \text{ is } hot \wedge mild] \wedge [wind \text{ is } calm] \text{ THEN } plan \text{ is } swimming$$

where $temp$, $wind$ and $plan$ are linguistic variables, and $mild$, hot, $calm$ and $swimming$ are linguistic terms representing fuzzy sets. Here the difference between fuzzy and crisp rule induction becomes clear. The above rule will *not cover any instances* in the crisp case, since no instance can be both $mild$ and hot at the same time. This, however, is perfectly possible in the fuzzy case.

Table 8.1: The FUZZCONRI algorithm.

FuzzConRI
Input: Set of training instances T,
Set of concepts to learn $Concepts$
Output: A rule set describing the concepts
Set the rule set to empty
FOR EACH $concept \in Concepts$
$P = \{i \in T | \mu_{concept}(i) \geq \alpha_c\}$
$N = T - P$
WHILE P is not empty, and more rules can be found DO
$antecedent$=FindBestAntecedent(P, N)
If a suitable antecedent is found, augment the rule
set with "IF $antecedent$ THEN $concept$"
Remove the positive instances covered by the added rule
Return the rule set

FindBestAntecedent
Input: Set of positive instances, Set of negative instances
Output: Antecedent that covers the positive instances best
Let $STAR$ contain the antecedent $TRUE$
Let $BESTANT$ be nil
Let $TERMS$ contain all possible terms
While $STAR$ is not empty
$NEWSTAR = \{x \wedge y | x \in STAR, y \in TERMS\}$
$NEWSTAR = NEWSTAR - STAR$
For each antecedent A in $NEWSTAR$
If A is better than $BESTANT$ according to an evaluation function, then
Replace the current best conjunction with the new one
Remove all antecedents that cover only positive instances
Retain a user defined number of best elements in $NEWSTAR$
$STAR = NEWSTAR$
Return $BESTANT$

FUZZCONRI's bottom layer receives a set of positive and a set of negative instances, and returns the antecedent that best covers the positive instances while attempting not to cover any negative instances. It starts by initialising the set $STAR$ with the (FuzzyCAL) *mgc*, i.e. the conjunction TRUE. The best conjunction found during the search is stored in the variable $BESTANT$. The set $TERMS$ contains all linguistic terms of the given problem. A new $STAR$ is obtained by forming the conjunction of single linguistic term descriptions in $TERMS$ with each conjunction in $STAR$. In the next step, $STAR$ is subtracted from $NEWSTAR$. The conjunction of a description in $STAR$ with a description in $TERMS$ that was already present in the original description will of course bring no change, e.g. the conjunction of $[hot][calm]$ with the description $[hot]$ does not change the conjunction. By subtracting $STAR$ from $NEWSTAR$, conjunctions that were not changed are removed from $NEWSTAR$.

After the new conjunctions are formed, they are evaluated according to the evaluation function and com-

pared to the best conjunction found thus far. If a conjunction obtains a higher score than the current best conjunction, it replaces the current best conjunction. Finally, the conjunctions that cover only positive instances are removed from the search since further specialization cannot improve the performance. $STAR$ is then replaced by $NEWSTAR$ and the process iterated until $STAR$ becomes empty. $STAR$ will be empty when no new descriptions different from those in the original $STAR$ can be formed. FUZZCONRI implements a beam search by retaining more than one conjunction for further specialization in the new $STAR$. Finally, the best conjunction found during the search process is returned. There is scope for improving this version of FUZZCONRI, but since we will show in the next chapter that FUZZCONRI fits within a more general framework, we postpone that discussion until then.

8.4 Small Example

Consider the sport data set, as shown in Table 4.1. Let the concept to learn be *plan.swimming*, and let $\alpha_c = 0.6$ and $\alpha_a = 0.5$. The concept threshold $\alpha_c = 0.6$ defines the set of positive instances to be 1, 2, 3, 9, 11 and 16, using the instance indices in the table.

Figure 8.1 demonstrates FuzzConRI's specialization process for this learning problem, and shows (only) the conjunctions generated in the lattice of FuzzyCAL descriptions. Each node represents an antecedent, where the conjuncts of the antecedent are shown below each other, and the most general conjuncts are not shown. The two numbers in parentheses are the number of positive and negative instances covered by the antecedent, respectively. The specialization process starts with the antecedent $TRUE$, and appends all possible terms to it, forming all one-term antecedents in the second layer. Since there are ten terms, the second layer contains ten nodes. For this example we allow only the best antecedent to remain in $STAR$ for each iteration in *FindBestAntecedent*, and we use the Laplace estimate $L(A)$ as an antecedent evaluation function,

$$L(A) = \frac{p+1}{p+n+2} \tag{8.1}$$

where p and n are the number of positive and negative instances covered by the antecedent A, respectively. The evaluation of a conjunction is shown as the third value in parentheses of each node. In the second layer, the antecedent with the highest score is $[hot]$, and $[sunny]$ is the second best. If a beam search with width two was selected, both antecedents would be expanded. Since no beam search is specified, only $[hot]$ is expanded by appending all possible terms to the conjunction. The third layer contains nine conjunctions, since $[sunny, sunny]$ is removed.

The best conjunction in the third layer is $[sunny][hot]$, which covers five positive instances and no negative instances. This is also the best conjunction found during the whole search process. Since it is consistent, it is removed from $STAR$ and not expanded further. For a beam search of width two this antecedent could be generated both by appending $[sunny]$ to $[hot]$ or vice versa. Note the conjunction $[hot, mild]$ (i.e. $temp.hot \wedge temp.mild$) covers one positive and no negative instances, and ties the third best score. For more complicated problem domains it may well be that such conjunctions can be useful concept descriptions.

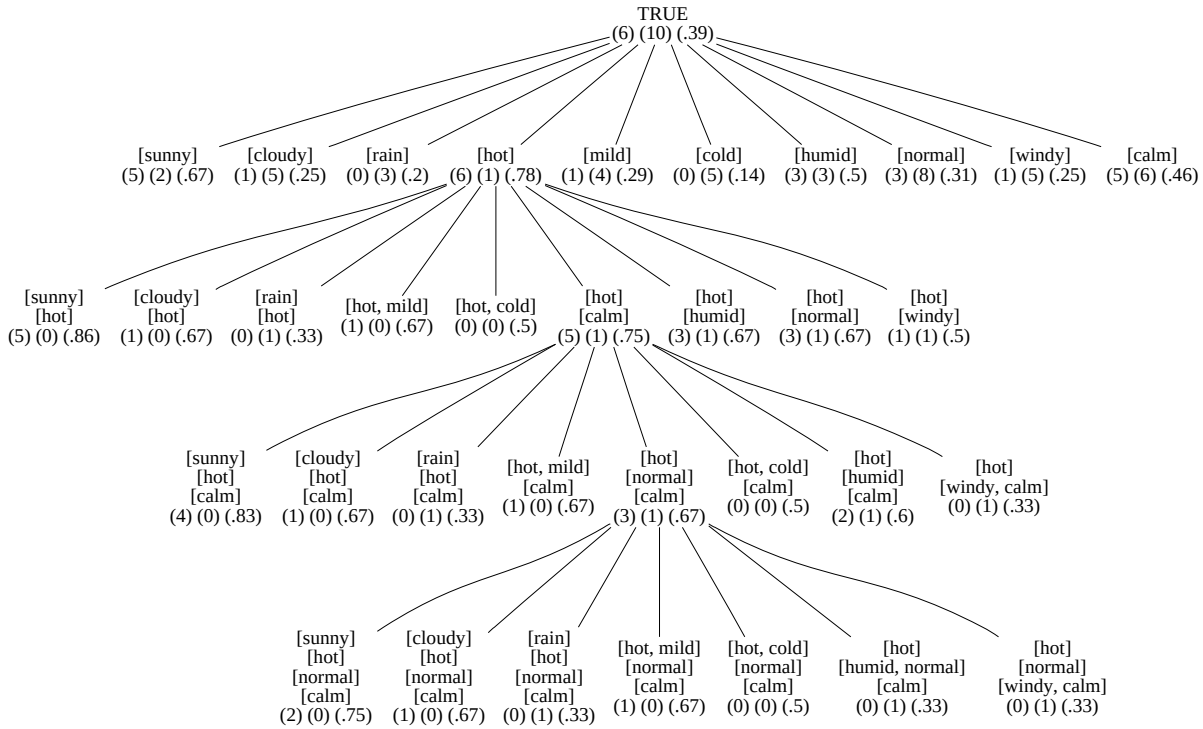


Figure 8.1: The specialization paths followed using FUZZCONRI.

Since $[sunny][hot]$ is removed from the search, the next best conjunction, $[hot][calm]$ is expanded further, forming the fourth layer with eight conjunctions. The best conjunction in this layer is the conjunction $[sunny][hot][calm]$. This conjunction is also consistent and thus not expanded further. No conjunction in the fourth layer covers five positive instances (like $[sunny][hot]$ does). Thus, further search cannot find a conjunction that outperforms $[sunny][hot]$, and this conjunction is returned as the best conjunction found. This lattice property was exploited by FUZZYBEXA to improve its search, and we show in the next chapter that it can also benefit FUZZCONRI.

The five instances covered by the rule

$$\text{IF } outlook.sunnny \wedge temp.hot \text{ THEN } plan.swimming$$

are instances 1, 2, 9, 11 and 16. These are removed from the set P, and since $P \neq \emptyset$ the process is repeated and another rule is induced.

8.5 Summary

In this chapter we introduced FUZZCONRI, the fuzzy generalization of the CN2 rule induction algorithm. FUZZCONRI's description language is FuzzyCAL, which can form conjunctions containing any set of terms in the problem space. This description language has no equivalent in the crisp case. We showed that like for FuzzyAL, FuzzyCAL descriptions also form a lattice. We demonstrated the behaviour of the algorithm by tracing its specialization process for the Fuzzy Sport problem. Although

they use different description languages, there are clearly many similarities between FUZZYBEXA and FUZZCONRI—they are both fuzzy set covering algorithms. In the next chapter we introduce a general fuzzy set covering framework, and show that FUZZYBEXA and FUZZCONRI, among others, fit within this framework.

CHAPTER 9

Fuzzy Specialization Models for a General Fuzzy Set Covering Framework

9.1 Introduction

Thus far we have introduced two algorithms that apply fuzzy set covering, FUZZYBEXA and FUZZCONRI. We now propose a general fuzzy set covering framework, which we call FCF (for Fuzzy Covering Framework). The framework is general in the sense that it does not dictate the use of a specific description language, or how the hypothesis (description) space should be searched. The framework consists of three layers - a top layer implementing fuzzy set covering, a middle layer implementing several learning heuristics, and a bottom layer, called the *specialization model*. A specialization model receives a set of concept descriptions, and returns a set of refined concept descriptions to be evaluated by the higher layers of the framework. Thus, the specialization model implements the specific search strategy within the defined description space.

The separation and modularization of the different functions of the framework has several benefits. It presents a unifying model for fuzzy set covering algorithms—of which crisp set covering algorithms are special cases. It allows the characterization of different algorithms based on their description languages, specialization operators, and search strategies. The modular design allows the incremental development of new learning algorithms that follow the fuzzy set covering approach to rule learning. Furthermore, the hierarchical architecture of the framework means that improvements to the top layers automatically benefit all algorithms that fit into the framework.

We also propose four different specialization models for FCF. Each algorithm applies a different search strategy, encompassing a mixture of different description languages and search heuristics. The framework allows for the fair comparison of these algorithms—the top layers are kept constant and only the specialization model is exchanged. The performance of the specialization models on data sets can be compared, as well as the individual characteristics of the specialization models, such as search heuristics, specialization operators, or search paths in the hypothesis space.

The layout of the chapter is as follows. In the following section we describe FCF, particularly the top two layers of the framework. We then continue in the next four sections to describe several specialization models that fit within the general framework, respectively the specialization models FEM

Table 9.1: FCF's top layer.

```
    PROCEDURE CoverConcepts(T, Concepts)
1     ruleset = ∅;
2     FOR each concept c_i ∈ Concepts DO
3       P = X_T(c_i); N = T − P;
4       REPEAT
5         bestconj =FindBestConjunction(P, N);
6         IF |X_P(bestconj)| > θ_p AND isBetter(bestconj, getMGC(DL)) THEN
7           Add rule "IF bestconj@α_a THEN concept is c_i@α_c" to ruleset;
8         END IF
9         P = P − X_P(bestconj);
10      UNTIL (|P| ≤ θ_p);
11    END FOR
12    RETURN ruleset;
    END PROCEDURE
```

(Fuzzy Exclusion Model), FUZZYSEEDSEARCH, FUZZCONRI, and FUZZYPRISM. We introduce the different characteristics of each specialization model as we introduce the specialization model itself, and then provide an overview and comparison of their different characteristics in Section 9.7. Section 9.8 presents an empirical comparison of the specialization models on data sets retrieved from the UCI Machine Learning Repository. In Section 9.9 we discuss partial covering, where we look at the implications of fuzzifying the set of positive instances P. We present final remarks and conclusions in Section 9.10.

9.2 FCF, A General Fuzzy Set Covering Framework

Thus far we introduced fuzzy set covering as a methodology for the induction of fuzzy rules, and we proposed two algorithms, FUZZYBEXA and FUZZCONRI as examples of fuzzy set covering algorithms. In this section we propose a general framework for algorithms following the fuzzy set covering approach, which we call FCF. The framework will make the relationship between different covering algorithms (both fuzzy and crisp) explicit, and will allow different covering algorithms to be characterized in terms of their differences and similarities within this framework. FCF provides the functionality common to all fuzzy set covering algorithms, and leaves the specific details, such as conjunction specialization and related search heuristics, to the lowest layer. This allows each algorithm to only implement this bottom layer, relying on the remainder of the framework to provide an implementation of the common components.

Table 9.1 shows the top layer of the framework. It implements the general set covering approach, as detailed in Chapter 4. It starts by initialising the rule set to empty. Then, for each concept, it divides the training set into the sets of positive and negative instances. For each pair, rules are induced until the number of positive instances remaining in P is equal to or less than the positive coverage threshold θ_p. The parameter θ_p may be zero, and is user-defined. The induction loop makes use of the second layer

Table 9.2: FCF's middle layer.

```
    PROCEDURE FindBestConjunction(P, N, beamwidth)
1     bestconj =NULL;
2     specializations = {getMGC(DL)};
3     WHILE specializations ≠ ∅ DO
4       specializations = GenerateSpecializations(P, N, specializations);
5       FOR each conjunction c ∈ specializations DO
6         IF isBetter(c,bestconj) THEN
7           bestconj = c;
8       FOR each conjunction c ∈ specializations DO
9         IF stopGrowth(c) THEN
10          specializations = specializations − c;
11      Retain in specializations only the beamwidth best conjunctions
12    END WHILE
13    RETURN bestconj;
    END PROCEDURE
```

routine, *FindBestConjunction*, to induce a single good rule antecedent. If the induced antecedent passes a set of criteria, a rule with this antecedent and the current concept as consequent is formed. Rules with too low positive coverage (i.e. $|X_P(c)| \leq \theta_p$) are not considered to be significant, thus the induced antecedent must cover more than θ_p positive instances before it can be used to form a rule. The function *isBetter*(c_1, c_2) is a user-defined conjunction evaluation function that compares the two conjunctions c_1 and c_2, and returns true only if c_1 is better than c_2. The evaluation function plays an important role in guiding the search process—conjunctions are retained or pruned from the search based on their ranking by the evaluation function. Since different domains may be suited to different evaluation functions, FCF does not dictate the implementation of this function, and various possible conjunction evaluation functions were discussed in Chapter 6.

The function *getMGC*(DL) returns the most general conjunction (*mgc*) in the description language used by the current induction algorithm. In both the description languages FuzzyAL and FuzzyCAL (see Sections 4.3.1 and 8.2) this conjunction is equivalent to TRUE, and thus covers the whole instance space. If the evaluation of the candidate antecedent is not better than that of the *mgc*, it is not used to form a rule. After the induction of each rule antecedent, the positive instances covered by it are removed from the set of positive training instances P. Since P always gets smaller, the algorithm is guaranteed to terminate.

Table 9.2 shows FCF's middle layer. This layer employs several heuristics to improve performance. It implements a local beam search by maintaining the current best $beamwidth$ conjunctions in the set $specializations$, as well as the current best overall conjunction in the variable $bestconj$. The set $specializations$ is initialized with the *mgc* of the current description language. Then, while the set of specializations is not empty, it is refined using the bottom layer routine *GenerateSpecializations*. The bottom layer routine implements the specialization model specific to the induction algorithm using the framework. Each specialization obtained by the refinement process is then compared to the current best

Table 9.3: One implementation of the compare function.

```
    PROCEDURE isBetter(c1, c2)
1     IF eval(c1)>eval(c2) THEN RETURN TRUE;
2     IF eval(c1)<eval(c2) THEN RETURN FALSE;
3     IF |X_P(c1)| > |X_P(c2)| THEN RETURN TRUE;
4     IF complexity(c1)<complexity(c2) THEN RETURN TRUE;
5     RETURN FALSE;
    END PROCEDURE
```

conjunction, and replaces it only if it performs better. It is thus clear that the conjunction evaluation and comparison process is an important factor in the search process. One example of such a comparison routine is given in Table 9.3, where *eval*(c) and *complexity*(c) are two user-defined functions for evaluating the classification performance and the complexity of the conjunction c, respectively. After the best conjunction was selected from the current set of specializations, conjunctions satisfying the stop growth prepruning criteria are removed. One implementation of the prepruning criteria is shown in Table 9.4. It prunes conjunctions with empty negative extensions. It also prunes conjunctions that even when evaluated optimistically still do not perform better than the current best conjunction. Optimistic evaluation was described in Section 4.5.3. Note, both functions *isBetter* and *stopGrowth* can be changed and expanded by the user. The exact functionality of these functions can be controlled by a selection of control parameters, and any additions or improvements to these functions will benefit all algorithms that fit in the framework. When no more conjunctions remain to be specialized, the function returns the best conjunction found during the search.

In the following section we present four different specialization models for the FCF framework. In order to relate the different specialization models to one another, it is useful to compare them keeping the following points in mind. Different specialization models (1) employ different description languages, (2) specializes conjunctions in different ways (we will associate a *specialization operator* with each method to formalize its specialization method), (3) apply different search heuristics, (4) select different specialization paths through their description lattices, and (5) search this lattice to different degrees of thoroughness. The functionality of the specialization models need not be limited to specialization, and one may also use the term "refinement model" instead of "specialization model," where refinement would also include in the general case a bottom-up search strategy. However, since all the algorithms introduced in this chapter perform a top-down search, we use "specialization model" instead of "refinement model."

For the purpose of comparing the specialization models we will use the toy fuzzy learning problem in Table 9.5. Three linguistic variables A, B and $class$ are defined, and their respective term sets are $\{a, b\}$, $\{x, y, z\}$, and $\{pos, neg\}$. Six fuzzy instances are given, with the instances' linguistic term memberships sorted into term sets, and given in the order of declaration.

Table 9.4: One implementation of the stop growth function.

```
    PROCEDURE stopGrowth(c, bestconj)
1     IF X_N(c) = ∅ THEN RETURN TRUE;
2     IF evalOptimistic(c) < eval(bestconj) THEN RETURN TRUE;
3     RETURN FALSE;
    END PROCEDURE
```

Table 9.5: A small fuzzy learning problem.

```
@relation smallfuzzyproblem
@attribute A     {a, b}
@attribute B     {x, y, z}
@attribute class {pos, neg}
@data

(0.7 0.3), (0.6 0.2 0.4), (0.3 0.7)  ;1
(0.8 0.2), (0.6 0.4 0.1), (0.4 0.6)  ;2
(0.7 0.5), (0.1 0.4 0.6), (0.9 0.1)  ;3
(0.2 0.7), (0.1 0.1 0.9), (0.1 0.9)  ;4
(0.3 0.7), (0.3 0.7 0.6), (0.8 0.2)  ;5
(0.6 0.5), (0.3 0.7 0.2), (0.6 0.4)  ;6
```

9.3 Fuzzy Exclusion Model

The Fuzzy Exclusion Model (FEM) is the specialization model employed by FUZZYBEXA [Cloete and van Zyl, 2006]. It uses FuzzyAL as description language, and thus allows internally disjunctive expressions. Conjunction specialization entails the exclusion of a single linguistic term from the conjunction. Thus, the specialization operator for FEM is *exclude*. Table 9.6 shows the FEM algorithm. FUZZYBEXA's specialization model was discussed in detail in Section 4.6, including the implementation of "remove uninteresting terms" and the efficient computation of the positive and negative extensions of specializations.

For the sake of comparison with the other specialization models, we demonstrate FEM's specialization behaviour by considering the specializations generated for the toy problem in Table 9.5. We set the beam width to two, $\alpha_c = 0.7$, and $\alpha_a = 0.5$. The set of positive instances is thus $P = \{3, 5, 6\}$, and the set of negative instances is $N = T - P = \{1, 2, 4\}$, where we denote instances by their indices. The maximum memberships of all instances i in the training set T to both linguistic variables A and B are greater than 0.5, and thus the alpha complement can be ignored for this example, since $\forall(i \in T)(\mu_{A.\bar{\alpha}}(i) = 0$ and $\mu_{B.\bar{\alpha}}(i) = 0)$. Figure 9.1 shows the lattice of FuzzyAL conjunctions. The sizes of the positive and negative extensions of a conjunction is shown as the first and second value in parentheses in each node in the figure, respectively. The third value in parentheses is the evaluation obtained by the respective conjunction. The conjunctions were evaluated according to the accuracy evaluation function in Eq (6.7).

Table 9.6: The Fuzzy Exclusion Model.

```
INPUT: sets of positive instances P, negative instances N,
       and conjunctions to specialize C
OUTPUT: set of specializations S
S = ∅
FOR each conjunction c ∈ C DO
  remove uninteresting terms from c.usable
  FOR each term L ∈ c.usable DO
    cnew = specialize (c)
    IF cnew ∈ S THEN CONTINUE
    cnew.X_P = compute X_P(cnew)
    IF cnew.X_P = ∅ THEN CONTINUE
    cnew.X_N = compute X_N(cnew)
    S = S ∪ {cnew}
  END FOR
END FOR
RETURN S
```

During the first iteration, *GenerateSpecializations* receives only the *mgc*, i.e. the top element in the lattice, to specialize. The circled nodes indicate the conjunctions generated by FEM, whereas the bold nodes show the $beamwidth$ best conjunctions in each layer, i.e. those chosen for further specialization. The paths followed during the search are indicated by bold edges. During the first iteration, it is possible to exclude any of the five linguistic terms, and thus five specializations are formed, as shown in the second layer of the graph. The best two conjunctions in the second layer are $[b][x, y, z]$ and $[a, b][x, y]$. By either excluding z from the first conjunction, or a from the latter conjunction, the specialization $[b][x, y]$ is obtained. This conjunction covers all positive instances, and none of the negative instances, and is the best conjunction found by FEM. It is a member of C_M, the set of most-general consistent conjunctions. The middle layer will halt the search at this time, since no other candidate conjunction or their specializations can obtain a better evaluation than this conjunction.

The search process is initialized with the *mgc* of the respective description languages, in this case FuzzyAL. Starting with the *mgc*, FEM specializes by excluding single linguistic terms at a time, generating *all* (useful) specializations. Thus, FEM performs a systematic, general-to-specific search of the lattice of FuzzyAL descriptions. Besides the search efficiency measures which can only speed up the search but not influence its outcome, FEM employs no heuristics for selecting a set of terms to use for specialization while ignoring the remainder of useable terms. Thus it performs the widest possible search for the given description language and beam width.

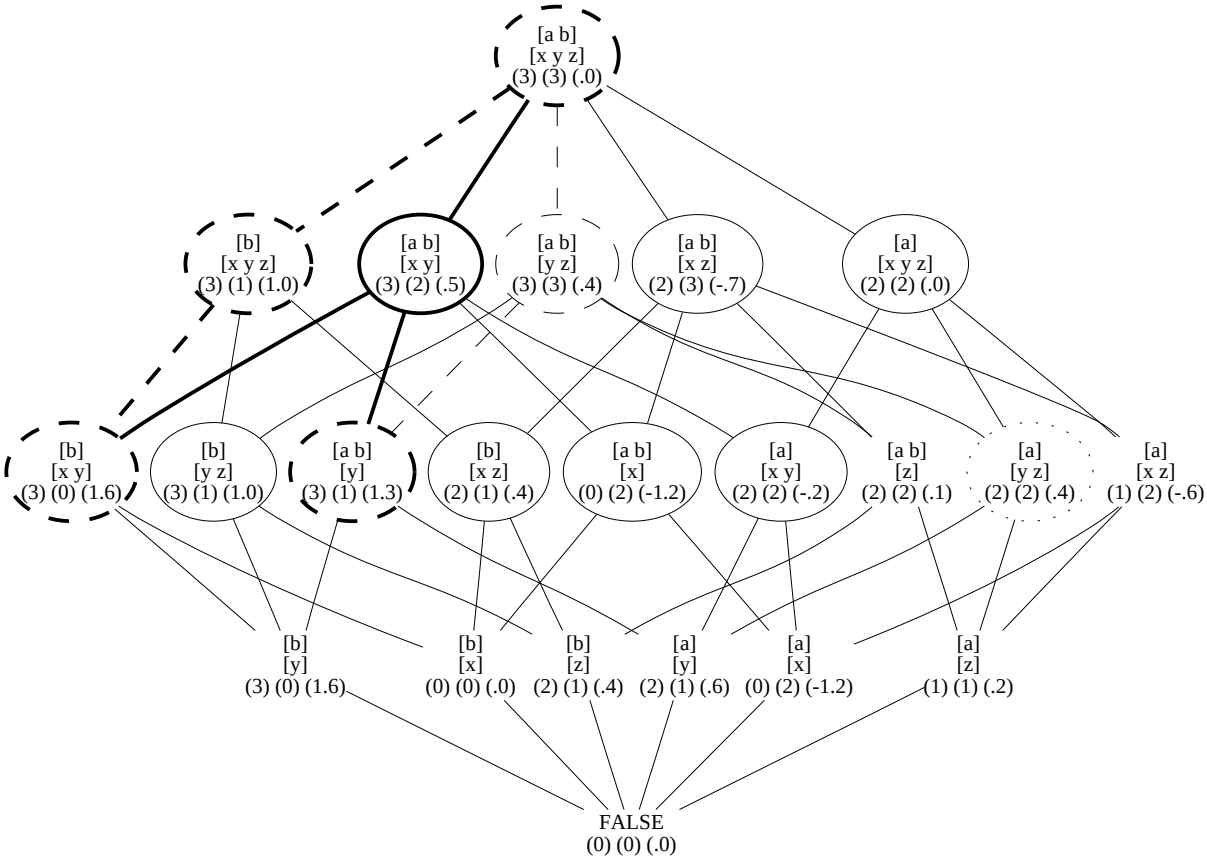


Figure 9.1: The lattice of FuzzyAL descriptions for the toy problem in Table 9.5.

9.4 FUZZYSEEDSEARCH

The next specialization model borrows its creative inspiration from the AQ family of algorithms [Michalski, 1969; Michalski et al., 1986a]. Like FEM, FUZZYSEEDSEARCH also uses FuzzyAL as description language, and therefore its specialization operator is also *exclude*. If the description set for a particular learning problem is very large, creating every possible specialization of a conjunction may have large computational overhead. Thus, instead of specializing conjunctions by excluding all possible linguistic terms, like FEM does, FUZZYSEEDSEARCH only excludes terms in TE, the set of terms to exclude. This set is computed by a subroutine call. FUZZYSEEDSEARCH's specialization procedure is shown in Table 9.7.

The subroutine *SelectTermsToExclude* functions as follows. First, a positive seed instance s_p and a negative seed instance s_n covered by a parent conjunction $c \in Conjuncts$ are selected, that is, $s_p \in X_P(c)$ and $s_n \in X_N(c)$. Note, if the seeds were not covered by any conjunction, then the specialization process will not generate any new conjunctions due to the efficiency criteria. The seed selection process is implemented by the routines *SelectPositiveSeed* and *SelectNegativeSeed*, and can be user-defined. We will discuss different seed selection strategies in the following section. Each linguistic term in the description set of the *mgc* is tested to determine if its positive extension *does not* contain the positive seed and its negative extension *does* contain the negative seed. If this is the case the linguistic term is

Table 9.7: The FUZZYSEEDSEARCH specialization model.

INPUT: sets of positive instances P, negative instances N,
and conjunctions to specialize C
OUTPUT: set of specializations S
$S = \emptyset$
$TE =$ SelectTermsToExclude ($Conjunctions, P, N$)
FOR each conjunction $c \in C$ DO
  remove uninteresting terms from $c.usable$
  FOR each term $L \in TE$ DO
    $cnew$ = specialize (c)
    IF $cnew \in S$ THEN CONTINUE
    $cnew.X_P$ = compute $X_P(cnew)$
    IF $cnew.X_P = \emptyset$ THEN CONTINUE
    $cnew.X_N$ = compute $X_N(cnew)$
    $S = S \cup \{cnew\}$
  END FOR
END FOR
RETURN S

SelectTermsToExclude($Conjunctions, P, N$):
$TE = \emptyset$
$s_p =$ SelectPositiveSeed($\bigcup X_P(c_i)$); // $c_i \in Conjunctions$
$s_n =$ SelectNegativeSeed($\bigcup X_N(c_i)$); // $c_i \in Conjunctions$
FOR each term $L \in D(mgc)$ DO
  IF $(s_n \in X_N(L))$ AND $(s_p \notin X_P(L))$ THEN
    $TE = TE \cup \{L\}$
RETURN TE

added to the set TE of linguistic terms to exclude from the search. The reasoning is that if this term is excluded from a concept description, the negative seed will be removed from its negative extension, while the positive seed will remain in its positive extension. Thus, all terms that cover the negative seed but not the positive seed are found.

The terms from TE are now used to specialize the conjunctions in C. Each conjunction c is specialized to form all descendants $cnew_i$ such that $D(cnew_i) = D(c) - a$, where $a \in TE$. Thus, we exclude a term from c that before only contributed to the covering of the negative seed. This allows the new conjunction to possibly obtain a better evaluation. If the positive and negative seeds were representative of distributions within the training set, many other negatives may now be uncovered. The remainder of the algorithm is exactly the same as FEM, including all efficiency measures.

Figure 9.1 shows an example run for a beam width of two for the toy problem in Table 9.5. In the first iteration, the conjunction to specialize is the *mgc*, which contains all linguistic terms. Let s_p be instance 5 and s_n be instance 1 during this iteration. Then the set of linguistic terms differentiating the two instances is *TE*$= \{a, b, x, y, z\}$. Only the positive extensions of the terms a and x do not contain s_p and contain s_n. Thus, only two specializations of the *mgc* are possible. The two specializations and

the paths from the *mgc* to them are indicated by dashed nodes and edges in Figure 9.1. Since the beam width is two, both conjunctions are specialized further in the next iteration. Suppose next that instances 6 and 4 were selected as the positive and negative seeds, respectively. The terms that differentiate the seeds are $\{a, b, y, z\}$. Of these, the positive extensions of $\{b, z\}$ do not contain s_p, and their negative extensions do contain s_n. Thus, the conjunctions $[b][x, y, z]$ and $[a, b][y, z]$ are specialized by excluding b and z. Note, the dotted node $[a][y, z]$ is generated by FUZZYSEEDSEARCH, but not by FEM. Finally, FUZZYSEEDSEARCH finds the same conjunction as FEM.

9.4.1 Seed Selection Methods

The positive and negative seeds clearly play an important role in guiding FUZZYSEEDSEARCH's refinement process. FUZZYSEEDSEARCH is designed in such a way that the seed selection method can easily be altered. The method of selecting the positive and negative seeds can be changed by changing the implementation of *SelectPositiveSeed* and *SelectNegativeSeed*, respectively. One implementation is to simply select a random instance from the set.

Instead of selecting a random instance, FUZZYSEEDSEARCH can also select the instance with the highest membership to any conjunction in $Conjunctions$. However, it is often the case that a data set also contains linguistic variables that have linguistic terms with crisp membership values, e.g. the sex of a patient. Thus, if the instance with the highest membership is always picked, the seed selection will be biased towards selecting instances from which no crisp terms were excluded since the crisp terms contribute most to the membership degree. Another consideration is that the positive and negative seeds with the highest confidence may not differ much, and therefore the set *TermsToExclude* could be empty (i.e. no term is such that it covers the negative seed and not the positive seed). This may be solved by reverting to the random seed selection if *TermsToExclude*$= \emptyset$ for the current iteration of *GenerateSpecializations*.

The algorithm can also be adapted not to use a positive seed, and to simply exclude those terms that contain the negative seed in their negative extensions. The argument for using the positive seed is that we want to cover as many positive instances as possible. Thus, if we can uncover the negative instance in some other way (i.e. by excluding another term) and still cover the positive instance, this route is preferred. Another approach is to select those seed instances that differ in most terms. However, this approach may have too much computational overhead, and the main goal of FUZZYSEEDSEARCH is to reduce this overhead.

9.4.2 FUZZYSEEDSEARCH and FAQR

Wang et al. [2003] proposed a fuzzy learning algorithm, FAQR, which is also based on the AQR crisp inductive learning strategy. Since FUZZYSEEDSEARCH and FAQR both take ideas from AQR, we briefly review the FAQR algorithm (for a detailed review of FAQR refer to Section 2.3) and then discuss the similarities and differences between FAQR and FUZZYSEEDSEARCH.

FAQR consists of two layers. The top layer starts with an empty rule R, and a set of positive and negative fuzzy instances, P and N respectively. The following steps are then iterated until the rule covers all positive instances. The instance that has the highest concept membership and is not α-covered by R is selected as positive seed. An instance i is α-covered if $\mu_R(i) \geq \alpha$. The function *GenComplex* is used to generate a set of candidate conjunctions that α-cover the positive seed and no negative instances. The best conjunction C_{best} is then added to the rule by forming $R = R \vee C_{best}$. The best conjunction has the highest evaluation $E(c)$,

$$E(c) = \mu_{include}(c)\ \rho\ \mu_{exclude}(c) \tag{9.1}$$

$$\mu_{include}(c) = \frac{\sum_{i \in P}(\mu_P(i)\ \tau\ \mu_c(i))}{\sum_{i \in P}\mu_P(i)} \tag{9.2}$$

$$\mu_{exclude}(c) = \frac{\sum_{i \in N}(\mu_N(i)\ \tau\ (1 - \mu_N(i))}{\sum_{i \in N}\mu_N(i)} \tag{9.3}$$

where ρ is an addition or a union operator, such as maximum, and τ is a t-norm operator, such as minimum. When R covers all positive instances it is returned as result.

The procedure *GenComplex* generates a set of candidate conjunctions. It initialises a set C_{set} of candidate solutions with the set of all single term descriptions, such as $[Temperature = hot]$ that α-cover the positive seed. It then repeats the following procedure while any description in C_{set} α-covers a negative instance. The conjunction $c \in C_{set}$ with the smallest value $\mu_{exclude}(c)$ is selected and then the negative instance in $X_N(c)$ that has the highest membership to N is chosen as the negative seed. All conjunctions in C_{set} are then refined not to α-cover the negative seed. This refinement process is performed as follows. Let S be the set of terms that α-cover the positive seed and not the negative seed. Then the new C_{set} is obtained by forming conjunctions of descriptions in the old C_{set} with terms in S. All descriptions in the new C_{set} that are subsumed by other descriptions are removed, and then the set C_{set} is pruned by removing the worst complexes until its size is less than a specified threshold. Once all complexes in C_{set} cover no negative instances, C_{set} is returned as result.

Clearly FUZZYSEEDSEARCH and FAQR have many similarities. Both are inductive learning strategies, and both induce incomplete fuzzy rules. An incomplete fuzzy rule is a rule of which the antecedent does not necessarily contain all linguistic terms. FAQR and FUZZYSEEDSEARCH share the heuristic to use positive and negative seeds to guide the search for rule descriptions. Both methods employ a generate-and-test strategy of generating a set of candidate rule descriptions and then selecting the best based on an evaluation function. Both methods also employ a beam search. However, there are also many fundamental differences between the two algorithms' search strategies, which we enumerate shortly.

(1) FAQR *does not* implement the fuzzy set covering methodology. Set covering algorithms iteratively induce rules that cover the set of positive instances, but not the set of negative instances, and after the induction of each individual rule, the positive instances covered by the rule are removed from the training set while the negative instances are retained (see Section 3.2 and Def. 3.2.1, Point 1). FAQR does not remove covered instances from the training set, but keeps adding more rules until all positive instances are covered. However, at each step FAQR will cover at least one positive instance that was not covered before. FUZZYSEEDSEARCH, on the other hand, fits with in the FCF framework, and thus implements

the fuzzy set covering methodology. At each rule induction step the positive instances covered by the induced rule are removed. Thus, FUZZYSEEDSEARCH follows a separate-and-conquer strategy, and each rule is biased to cover as many positive instances in the set of "still not covered" positive instances. FAQR does not have this bias. FAQR will thus prefer rule one over rule two if rule one covers more positive instances than rule two, regardless of whether the positive instances are already covered by other rules. For example, if rule one covers 20 positive instances of which one is the positive seed and the others are already covered, and rule two covers 15 positive instances of which one is the positive seed and the others were not covered before, FAQR will prefer rule one. This can result in much more complex rule sets, containing many rules that differ in only a few terms and covers overlapping sets of instances.

(2) FAQR's description language only allows conjunctions of linguistic terms, i.e. it does not allow internal disjunction. In contrast, FUZZYSEEDSEARCH induces a rule set where the antecedent of each rule is a description in FuzzyAL.

(3) FUZZYSEEDSEARCH uses *exclusion* as specialization operator whereas FAQR uses *append* as specialization operator. This has important implications. FUZZYSEEDSEARCH performs a systematic search from top to bottom in its description lattice. The top layer contains the *mgc* that covers *all* instances. By excluding terms from descriptions FUZZYSEEDSEARCH restricts the descriptions more and more such that they cover fewer and fewer instances. The search is then guided to cover progressively fewer negative instances while still covering as many positive instances as possible. FAQR starts its search with a set of conjunctions, each consisting of only one term, such that the conjunctions cover at least one positive instance. The conjunctions are then restricted to cover fewer and fewer instances by adding more and more terms to them. The search is guided to result in a set of conjunctions that cover none of the negative instances while still covering the positive seeds. Thus, during the search process FUZZYSEEDSEARCH is biased toward high positive coverage and low negative coverage, whereas FAQR is guided only toward low negative coverage.

(4) Due to the difference in FUZZYSEEDSEARCH and FAQR's specialization operators they choose seeds in different ways. The terms chosen for exclusion by FUZZYSEEDSEARCH should cover the negative seed and not the positive seed. After the exclusion of a term, the specialization will still cover the positive seed and may now not cover the negative seed any longer. It will (not) cover the negative seed if (no more) terms that have the negative seed in their extensions remain in the internal disjunction. The terms chosen by FAQR to add to the current conjunctions should cover the positive seed and not the negative seed. Thus, the specializations still cover the positive seed, but will definitely not cover the negative seed.

(5) FUZZYSEEDSEARCH as specialization model fits in the FCF framework. As such, FUZZYSEEDSEARCH inherits all the beneficial characteristics of FCF's top layers. Some of the more important inherited characteristics are search efficiency measures, early stopping and rule pre-pruning criteria, beam search, and an easily interchangeable description evaluation function. We showed that the evaluation function is very important for the learning process, and it is also very important to note that different evaluation functions are suited to different learning problems. The Accuracy evaluation function, for ex-

ample, is well suited to deal with noisy data and incomplete domain knowledge, but is not the optimal choice when dealing with the problem of small disjuncts. The Fuzzy Laplace and ls-Content evaluation functions, on the other hand, are better positioned to deal with the problem of small disjuncts. However, they do not perform as well as the Accuracy function in the presence of noise or incomplete domain knowledge, especially with respect to the size of the rule sets induced.

(6) FAQR and FUZZYSEEDSEARCH both perform a beam search. However, FUZZYSEEDSEARCH performs a systematic top-to-bottom general-to-specific search of the description lattice, moving down one layer at a time. During each step the next layer of the description lattice is considered, and up to a user adjustable number of conjunctions in this layer are generated and tested. The best conjunction found during the whole search is stored in the parameter $bestconj$. FAQR maintains a set of current conjunctions which may be specialized. This set may grow up to a user adjustable size—if it grows bigger than this size, the worst conjunctions are removed. If more specialized conjunctions are worse than more general conjunctions these are removed from the set, and are in fact again specialized during the next iteration of the search process. Thus, FAQR does not in general perform a systematic top-down search, and may again jump back up to more general conjunctions. The beam search is also not done in a systematic way, and may include conjunctions at different levels of generality.

(7) FAQR has two pruning steps. Conjunctions that are subsumed by other conjunctions and all conjunctions worse than the user defined number of best conjunctions are pruned from the search. FUZZYSEEDSEARCH also employs a fixed beam width, but also has further pruning criteria. Since a general-to-specific search is performed, FUZZYBEXA can determine whether further specialization can improve a conjunction to such a degree that it can replace the current best conjunction. If this is not the case, the conjunction is removed from the search. For example, if the best conjunction found thus far covers 20 positive and no negative instances, all conjunctions that cover less than 20 positive instances can be removed from the search process. Conjunctions are also not overspecialised. During specialization a conjunction is not simply specialized by excluding from it all the terms that cover the negative seed and not the positive seed. If excluding a term has no benefit, it is not excluded, resulting in FUZZYSEEDSEARCH's bias towards maximal generality and maximal classification accuracy. Duplicate specializations are removed from the search for efficiency reasons.

(8) The final difference we discuss here is not about the search method but about the semantic interpretation of a rule. The interpretation of Wang *et al* states that the membership degree of an instance to the rule consequent can be set equal to the membership degree of the instance to the rule antecedent. Thus, the degree to which an instance matches the antecedent can be used to predict the class membership of the instance. Our interpretation differs in that we take the membership degree of an instance to a rule as the *certainty* or *confidence* that the rule fires. The certainty or degree to which a rule fires does not predict the membership of the instance to the rule consequent, but specifies that the instance membership to the rule consequent lies within a certain range—the range $[\alpha_c, 1]$. If α_c was not set, then the instance membership to the concept is simply greater than zero. For example, if $\alpha_c = 0.8$ and we have the rule IF $X = x_1 \vee x_2$ THEN Y, and an instance i matches the antecedent to degree 0.75, our interpretation is that we are rather certain that the rule fires—we are certain to degree 0.75 that the instance belongs to the

Table 9.8: The FUZZCONRI specialization model.

```
INPUT: sets of positive instances P, negative instances N,
       and conjunctions to specialize C
OUTPUT: set of specializations S
S = ∅
FOR each conjunction c ∈ C DO
  remove uninteresting terms from c.usable
  FOR each term L ∈ c.usable DO
    cnew = specialize (c)
    IF cnew ∈ S THEN CONTINUE
    cnew.X_P = compute X_P(cnew)
    IF cnew.X_P = ∅ THEN CONTINUE
    cnew.X_N = compute X_N(cnew)
    S = S ∪ {cnew}
  END FOR
END FOR
RETURN S
```

concept with membership in the range $[0.8, 1]$. We do not believe that the membership of an instance to the antecedent can be used to predict the membership of the instance to the concept with an acceptable degree of accuracy without any form of (likely non-linear) transformation of the input domain to the output domain, e.g. like that performed by a neural network.

9.5 FUZZCONRI as Specialization Model

FUZZCONRI is an acronym for Fuzzy Conjunctive Rule Inducer, and was the main subject of Chapter 8. Two basic factors distinguish the algorithms FUZZCONRI and FUZZYBEXA, their respective description languages and specialization operators. FUZZCONRI employs FuzzyCAL as description language and performs a general-to-specific search by using *append* as it specialization operator. In the remainder of this section the term FUZZCONRI is used to refer to the specialization model implementing FUZZCONRI's conjunction specialization strategy.

FUZZCONRI follows the same strategy as FEM, and thus the same algorithm as in Table 9.6 can be used, but with different operations associated with the different subroutines. FUZZCONRI also associates a set $usable$ with each conjunction. This set contains the set of usable linguistic terms, i.e. the set of terms that may still be used by the specialization operator. The $usable$ set of the *mgc* contains all terms—the same as for FEM. Note however, for FuzzyAL $D(mgc)$ contains all terms and $mgc.usable = D(mgc)$, while $D(mgc) = \emptyset$ in FuzzyCAL.

The measure removing "uninteresting" terms from each conjunction's usable set differs between FEM and FUZZCONRI. FUZZCONRI's measure consists of two tests, an efficiency measure and a pre-pruning step. The efficiency measure tests whether the intersection of the positive extension of the conjunction c

with the extension of a usable linguistic term L is empty, thus if $X_P(c) \cap X_P(L) = \emptyset$. If the intersection is empty, appending this linguistic term will create a new conjunction with an empty positive extension, which creates unnecessary work. By removing these linguistic terms from the conjunction's usable set, such conjunctions are never created. The second test prevents over-specialisation. If the union of the negative extensions of the conjunction c and the term to append L is equal to the negative extension of the term, i.e. if $X_N(c) \cup X_N(L) = X_N(L)$, then appending the term will not improve the performance of the conjunction. Since the append operator specializes conjunctions, no new (previously not covered) instances can be covered, i.e. positive instances can only become uncovered. However, all the negative instances covered by the conjunction are also covered by the linguistic term, and therefore no negatives will become uncovered. Since this overspecialisation is undesirable, we do not create such conjunctions.

Specialization proceeds via the specialization operator *append*, i.e. $cnew = c \wedge L$ where $L \in c.usable$. The computation of the positive and negative extension of a specialization can of course be implemented by considering the training set and matching each instance with the new conjunction. The instances that contain the concept belong to the positive extension of the specialization and those that do not belong to its negative extension. The calculation of the positive and negative extension can be done much more efficiently by making use of the information of the parent's extensions. The positive extension of the specialization can be efficiently computed by intersection of the positive extensions of L and the parent conjunction, i.e. $X_P(cnew) = X_P(c) \cap X_P(L)$. This computation does not require any matching, which is the computationally expensive part of this simplistic method. The negative extension follows dually. The test for an empty positive extension of $cnew$ as in FEM is not required in FUZZCONRI, since the efficiency test prevents the creation of such conjunctions.

Figure 9.2 shows the FuzzyCAL description lattice for the problem in Table 9.5, where we have set $\alpha_c = 0.6$ and $\alpha_a = 0.5$. The nodes contain the same information as the nodes in Figure 9.1. The top element is the *mgc*, and can be expanded into five second layer nodes. The candidate antecedents generated by FUZZCONRI for beam search with beam width two are marked by circled nodes, and the bold circled nodes are the best candidate antecedents at each specialization step. In the second layer, the conjunctions $[y]$ and $[b]$ obtained the highest (and equal) evaluation scores. At this point any one of the two conjunctions could be picked at random if no beam search was performed. The solid bold edges below $[y]$ and $[b]$ show the paths that are picked for $beamwidth = 2$. The best conjunction returned is $[y, z]$, i.e. the expression $B = y \wedge z$. This conjunction covers all positive instances and no negatives. The middle layer halts the search upon finding this conjunction, since it is certain that there exists no other conjunction that can outperform it. For example, the conjunction $[b][y, z]$, which is a specialization of $[y, z]$ and has the same evaluation as $[y, z]$, is never generated.

9.6 FUZZYPRISM

PRISM, a classical set covering algorithm for learning modular rules, was first introduced by Cendrowska [Cendrowska, 1987]. Wang et al. [1999] proposed a fuzzy version of this algorithm based on the fuzzy information gain, and the algorithm is shown in Table 9.9. It proceeds by forming an initially empty

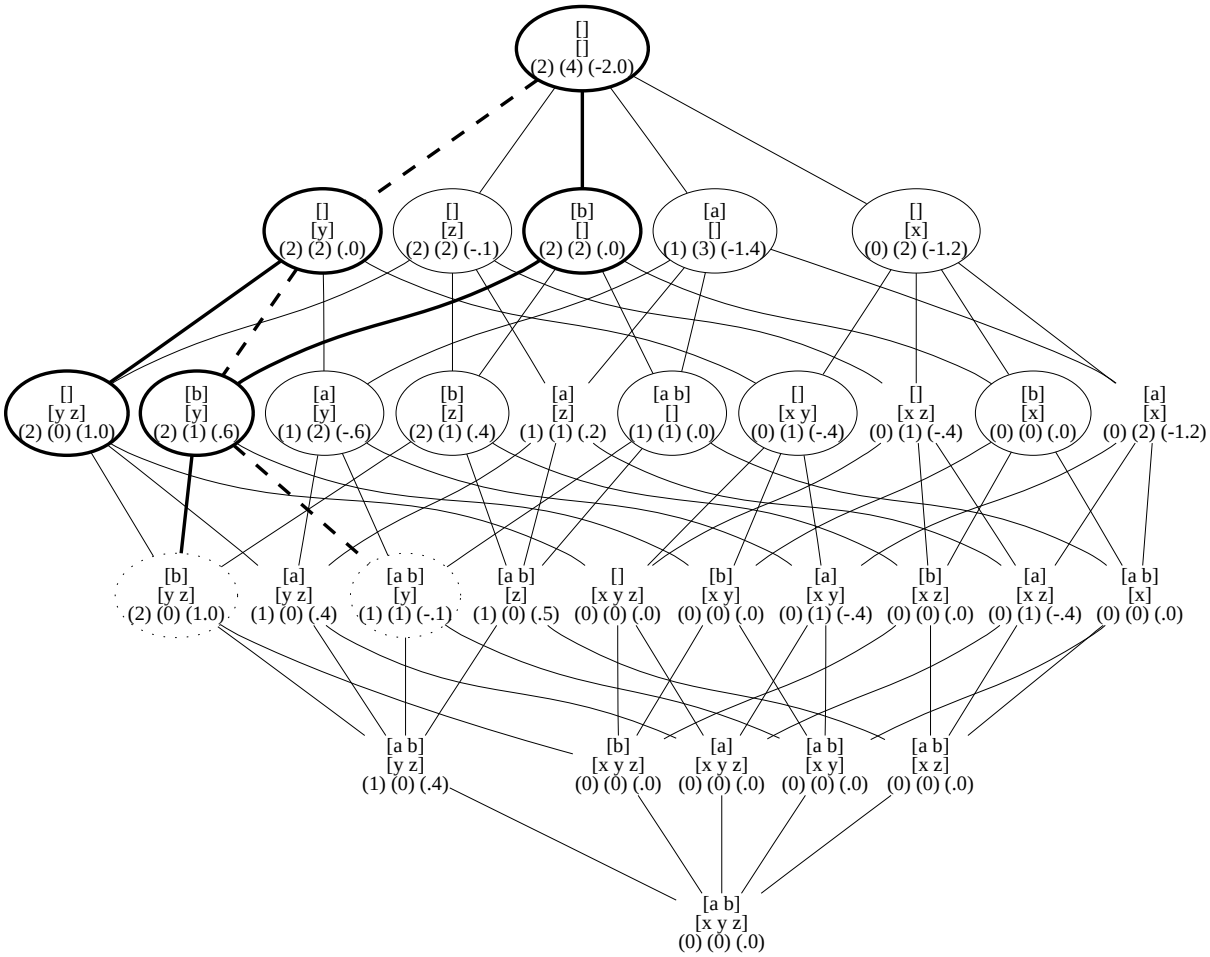


Figure 9.2: The lattice of FuzzyCAL descriptions for the toy problem in Table 9.5.

conjunction, which is semantically equivalent to TRUE. It then computes the fuzzy information gain $I(\delta_k|s_i)$ for the concept δ_k and each linguistic term s_i. The fuzzy information gain is calculated as,

$$I(\delta_k|s_i) = \log_2(\frac{H(\delta_k|s_i)}{H(\delta_k)}) \tag{9.4}$$

where $H(\delta_k)$ and $H(\delta_k|s_i)$ are the antecedent and consequent fuzzy information, respectively, and are defined as,

$$H(\delta_k|s_i) = \frac{\sum_{j=1}^{n} \mu_{\delta_k}(e_j) \tau \mu_{s_i}(e_j)}{\sum_{j=1}^{n} \mu_{s_i}(e_j)} \tag{9.5}$$

and

$$H(\delta_k) = \frac{1}{n} \sum_{j=1}^{n} \mu_{\delta_k}(e_j) \tag{9.6}$$

where τ is a t-norm operator (like minimum), n the size of the training set, and $\mu_{\delta_k}(e_j)$ the membership degree to the concept δ_k of e_j, the j^{th} instance in the training set. After calculating the information gain for all linguistic terms, the term leading to maximum information gain is appended to the current conjunction C. The fuzzy strength of the rule is then computed using the fuzzy Bayes function,

$$B(\delta_k|C) = \frac{\sum_{j=1}^{n} \mu_C(e_j) \tau \mu_{\delta_k}(e_j)}{\sum_{j=1}^{n} \mu_C(e_j)} \tag{9.7}$$

Table 9.9: The fuzzy inductive algorithm for learning modular rules.

1	Initiate a null complex C
2	Measure the fuzzy information gain, $I(\delta_k\|s_i)$, of the classification δ_k for each possible linguistic term s_i
3	Choose the linguistic term s_i for which $I(\delta_k\|s_i)$ is maximum
4	Add s_i to C, $C = C \wedge s_i$, and calculate $B(\delta_k\|C)$
5	If $B(\delta_k\|C) \geq \beta$, go to step 6, otherwise create a new training set in which each instance is covered to degree α by the term s_i, and go to step 2.
6	Form the rule "IF C THEN δ_k"
7	Remove the instances covered to degree α by the rule from the original training set.
8	Repeat steps 1 to 7 until all instances belonging to the class δ_k in the original training set have been removed.

If the rule evaluation is larger than a predefined threshold β, a new rule is formed, and all instances covered by the rule are removed from the original training set. If uncovered instances of class δ_k remain in the training set, the procedure is repeated to induce the next rule. If the rule evaluation is below the threshold β, the instances in the current training set covered by the linguistic term is used to form a new training set, and the algorithm continues to add more linguistic terms to the conjunction.

We now propose a novel specialization model for FCF, called FUZZYPRISM. FUZZYPRISM makes use of the same information theoretic heuristic as the algorithm by Wang *et al* (Table 9.9). However, FUZZYPRISM is a specialization model within FCF, and thus employs the fuzzy set covering rule induction methodology. FUZZYPRISM's description language is FuzzyCAL, its specialization operator is thus *append*. Contrary to Wang's algorithm, FCF decouples conjunction refinement from conjunction evaluation and other learning heuristics.

Table 9.10 shows the FUZZYPRISM specialization model. Clearly, FUZZYPRISM and FUZZCONRI are strongly related. However, instead of generating all possible (useful) specializations, FUZZYPRISM uses the fuzzy information as a heuristic to select the set TE of linguistic terms to append. The term selection routine, *SelectTermsToExclude*, functions as follows. For each remaining linguistic term L in the set $c.usable$ the fuzzy information gain is computed. To allow a beam search of the hypothesis space, FUZZYPRISM allows up to a $beamwidth$ number of specializations in TE. The remainder of the algorithm is the same as FUZZCONRI.

The early stopping criterion proposed by Wang *et al* amounts to search until the evaluation function reaches a predefined threshold. It may be difficult to judge what threshold to use in general, and the authors gave no method to determine its value. FCF can employ several stopping criteria, of which search until reaching some threshold may serve as one example (we discussed several others). More importantly, Wang's algorithm *does not* apply the fuzzy set covering methodology, since *all* instances covered by a rule are removed after its addition to the rule set, while the set covering approach dictates that after the induction of each individual rule, the positive instances covered by the rule are removed from the training set, but the negative instances retained (see Section 3.2 and Def 3.2.1). In contrast, FCF benefits from the separate-and-conquer search of the fuzzy set covering methodology, since the retention of all negative instances in the training set means that more instances are available as counter examples

Table 9.10: The FUZZYPRISM specialization model.

INPUT: sets of positive instances P, negative instances N,
and conjunctions to specialize C
OUTPUT: set of specializations S
$S = \emptyset$
FOR each conjunction $c \in C$ DO
  remove uninteresting terms from $c.usable$
  $TE =$ SelectTermsToExclude (c)
  FOR each term $L \in TE$ DO
    $cnew$ = specialize (c)
    IF $cnew \in S$ THEN CONTINUE
    $cnew.X_P$ = compute $X_P(cnew)$
    $cnew.X_N$ = compute $X_N(cnew)$
    $S = S \cup \{cnew\}$
  END FOR
END FOR
RETURN S

SelectTermsToExclude(c):
FOR each term $L \in c.usable$ DO
  Compute the information gain $I(concept|L)$
RETURN set containing $beamwidth$ best terms

during the induction of subsequent rules, thereby improving the accuracy of these rules. However, of all the algorithms surveyed, Wang's algorithm is the most related to our work.

In Figure 9.2, the path followed by FUZZYPRISM for greedy search is indicated by dashed lines. In the first iteration the linguistic term y had the highest and b the second highest information gain. For greedy search, FUZZYPRISM picked y to specialize further, which was a good choice. For the conjunction $[y]$, the linguistic term b had the highest and z the second highest information gain. The evaluation of conjunction $[][y, z]$ is worse than that of $[b][y]$, and thus with greedy search FUZZYPRISM only selects the second best conjunction in the third layer of the lattice. In the next step the heuristic seemed to have failed. FUZZYPRISM selected to append a to $[b][y]$ as opposed to appending z. Since $[b][y]$ has a higher evaluation than $[a, b][y]$ the best conjunction returned for greedy search will be $[b][y]$. This conjunction covers all positive instances, but also a negative instance. However, if a beam search of width two is performed, FUZZYPRISM follows the same paths as FUZZCONRI, i.e. it generates all the bold nodes. It therefore also finds $[y, z]$, the best conjunction in the entire lattice. For this search ($beamwidth = 2$) FUZZYPRISM thus only generated five conjunctions as opposed to the thirteen conjunctions that FUZZCONRI generated. Since at most a $beamwidth$ number of conjunctions are specialized in at most $beamwidth$ number of ways, FUZZYPRISM generates at most $beamwidth^2$ specializations per layer of the lattice.

9.7 Discussion

In the previous sections we presented four specialization models. Each specialization model has its own strengths and weaknesses, and the problem domain will dictate which specialization model is best suited. The first and most important distinguishing characteristic of the specialization models is their respective description languages. FUZZYSEEDSEARCH and FEM use FuzzyAL as description language, and FUZZYPRISM and FUZZCONRI use FuzzyCAL. The intersection of semantically equivalent descriptions in FuzzyAL and FuzzyCAL is not empty, but at the same time relatively small compared to the size of the lattice. There are many antecedents (and thus rules) that can be formed in FuzzyAL but cannot be formed in FuzzyCAL, and vice versa. FuzzyAL descriptions can contain conjuncts such as $temp = mild \vee hot$, whereas FuzzyCAL can contain conjuncts such as $temp = mild \wedge hot$. For example, for the toy problem with $\alpha_c = 0.6$ there does not exist a conjunction in FuzzyAL that covers all positive and no negative instances, but the inverse is also true for $\alpha_c = 0.7$. Thus, there are some problems that are better described by FuzzyCAL than FuzzyAL and vice versa.

A most general *conjunct* in FuzzyAL is a conjunct from which no linguistic terms were excluded, and a most general conjunct in FuzzyCAL is a conjunct to which no terms were appended (i.e. $[\,]$). Most general conjuncts in both FuzzyCAL and FuzzyAL cover the entire instance space. A most specific conjunct in FuzzyAL is a conjunct from which the complete term set was excluded, and is equivalent to FALSE. A most specific conjunction in FuzzyCAL is a conjunct to which the complete term set was appended, and is *not necessarily* equivalent to FALSE. Consider for example instance five in Table 9.5 and the conjunct $[B = x \wedge B = y \wedge B = z]$. If $\alpha_a < 0.3$ the conjunct, although being most specific, covers the instance. Such conjunctions may therefore still be generated. If a general to specific search is desired, the specialization operator is dependent on the description language, e.g. FuzzyCAL descriptions can only be specialized using *append*, while FuzzyAL descriptions can only be specialized using *exclude*. The specialization model may, however, use the specialization operator in different ways, e.g. a specialization model may choose to exclude all but one term from a conjunct in a single specialization step.

FuzzyAL describes inherently more general conditions than FuzzyCAL. The second layer of the FuzzyCAL description lattice contains all descriptions with a single term appended. The semantically equivalent descriptions in FuzzyAL is located in the layer just above the most specific conjunction FALSE. Some conjunctions in the third and lower layers in FuzzyCAL describe conditions that are more specific than any conjunction in FuzzyAL (except for FALSE), since these conjunctions require that the membership functions overlap. The rate at which instances are excluded during search is also much higher for FuzzyCAL than for FuzzyAL. By moving from the *mgc* to the next layer, FuzzyCAL requires that an instance belongs to a specific linguistic term, while FuzzyAL allows the instance to belong to any linguistic term remaining in the conjunct. Clearly, in general many more instances will be matched by conjunctions in the second layer of the FuzzyAL lattice than in the FuzzyCAL lattice. The exclusion rate is accelerated even more as more than one term from the same variable is appended to FuzzyCAL conjunctions. In this case, only instances that fall in the region of overlapping membership functions remain covered. Thus, although more descriptions from the FuzzyCAL lattice could potentially be generated,

Table 9.11: Specialization model properties.

	Complete Search	Heuristic Search
FuzzyAL	FEM	FUZZYSEEDSEARCH
FuzzyCAL	FUZZCONRI	FUZZYPRISM

in practice less descriptions in FuzzyCAL are generated. The conjunctive expressions quickly restrict the extensions such that only a few linguistic terms from a single term set are typically appended. This of course will depend entirely on the membership functions. If there is no overlap between membership functions, only a single linguistic term per term set will be appended to conjunctions. Thus, one indication of which description language to use is the amount of overlap between membership functions. If there is no overlap (or very little) between membership functions, conjunctions with non-empty extensions will have semantically equivalent expressions in FuzzyAL, and FuzzyAL should be preferred to enlarge the hypothesis space. In some cases however interesting regions may be exactly those where membership functions overlap. These conditions can *only* be described with FuzzyCAL, and an algorithm using FuzzyCAL should thus be used in such cases.

The description lattice provides a visual method of comparison between different specialization models. By marking the specialization paths followed through the description lattice the specific behaviour of the specialization model becomes clear. FEM, for example, clearly performs the most comprehensive search of the lattice of FuzzyAL conjunction descriptions. Depending on the size of the beam width, it expands all conjunctions for which the possibility to improve exists. It is therefore more likely to find good descriptions than an algorithm which expands only a selection of conjunctions. FEM is also guaranteed to find the most general consistent FuzzyAL conjunctions. FUZZYSEEDSEARCH and FEM are clearly related. However, whereas FEM specializes using all terms in $c.usable$, FUZZYSEEDSEARCH specializes using only a subset of the terms, employing a heuristic to guide its search. The heuristic entails comparing seed positive and negative instances and selecting terms that differentiate them. The quality of the outcome is dependent on the validity of the heuristic. While FEM may seem to require much search, FCF makes use of the partial order to identify conjunctions that cannot be refined to form specializations that can outperform the best conjunction, and therefore performs no unnecessary search. Even with a moderate beam width FEM's search requirement is seldom prohibitive.

FUZZCONRI and FUZZYPRISM both use FuzzyCAL as description language. FUZZCONRI performs a comprehensive search of the hypothesis space, and generates all useful specializations. FUZZYPRISM on the other hand use a heuristic to decide which specializations to generate. Using the fuzzy information gain, it ranks linguistic terms and uses the beam width best linguistic terms to specialize a conjunction. FUZZCONRI appends a single linguistic term per specialization step to all conjunctions, and is therefore guaranteed to find most general consistent conjunctions. FUZZYPRISM only appends linguistic terms to selected conjunctions, and is therefore not guaranteed to find most general consistent solutions. In the example it was shown that with a beam width of one, FUZZCONRI generated many more conjunctions, but found the best result, while FUZZYPRISM did not find an optimum result. With a beam width of two FUZZYPRISM still generated very few conjunctions, and also found the optimum result.

It is clear that FEM and FUZZCONRI are counterparts of each other, performing a comprehensive systematic search of the hypothesis space, while FUZZYSEEDSEARCH and FUZZYPRISM both use heuristics to guide their search. FUZZCONRI and FUZZYPRISM employ the same language bias, while FUZZCONRI and FEM employ the same search bias, and FEM and FUZZYSEEDSEARCH employ the same language bias, while FUZZYSEEDSEARCH and FUZZYPRISM both perform a heuristic search. If search time is not a limitation, and an infinite beam search is feasible, FEM and FUZZCONRI are both guaranteed to find the optimum solution (according to the evaluation function) at each rule induction step, while FUZZYSEEDSEARCH and FUZZYPRISM are not. In practice, a large beam search, for example beam width fifty, is a good approximation of an infinite beam width. While FUZZYSEEDSEARCH and FUZZYPRISM are not guaranteed to find most general consistent conjunctions, they are more likely to do so with a larger beam width. FUZZYSEEDSEARCH uses a heuristic that may involve a random element, the picking of seed instances, depending on the implementation of the seed picking mechanism. Thus, while the other specialization models will always return the same result for the same parameters, FUZZYSEEDSEARCH will not—even with an infinite beam width. For the same beam width, FUZZYPRISM performs far less search than the other specialization models, since it will only append a limited number ($beamwidth^2$) of terms. Thus, a larger beam width should be used with FUZZYPRISM to ensure a reasonable number of hypotheses are explored. The other specialization models perform more search as the size of the lattice increases (i.e. with more linguistic terms in the problem domain).

Depending on the ability of the conjunction evaluation function to indicate good paths in the lattice, FEM and FUZZCONRI will always find the most accurate and most general conjunctions. The price paid for this ability is that a larger part of the search space is investigated. Both FUZZYPRISM and FUZZYSEEDSEARCH try to limit this search by restricting the search in some manner. In fact, any other specialization model using FuzzyAL or FuzzyCAL will have to restrict the search in some way to differentiate them from FEM and FUZZCONRI, respectively. The inductive bias of all specialization models described here include a general-to-specific beam search element. FUZZYPRISM adds the assumption that good conjunctions will contain linguistic terms with high information gain. FUZZYSEEDSEARCH adds the assumption that good conjunctions contain more linguistic terms that match positive instances but not negative instances. Although the specialization models differ with respect to search and language bias they all fit within the general set covering framework. FCF thus provides a unifying framework for fuzzy set covering algorithms, regardless of their description language or specific conjunction refinement mechanism.

9.8 Empirical Comparison

Table 9.12 shows the classification accuracy obtained by the different specialization models on six data sets retrieved from the UCI repository [Blake and Merz, 1998]. The data sets were fuzzified by using a clustering technique to obtain centres for bell-shaped membership functions. The same membership functions were used for all experiments. The results were obtained from 10-fold cross validation runs. FCF using FEM obtained the best classification accuracy on average, with FUZZCONRI being sec-

Table 9.12: The classification accuracy obtained using different specialization models.

	FuzzyBexa		FuzzySeedSearch		FuzzConRI		FuzzyPRISM	
Database	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
BreastCancer	73.02	4.77	64.88	5.74	72.33	4.83	70.58	5.20
Colic	85.60	4.30	84.78	5.25	85.87	5.31	75.82	6.83
Credit-A	85.80	6.22	85.22	6.44	85.65	6.35	85.51	6.07
Hepatitis	81.29	8.71	81.94	7.48	84.52	7.10	78.71	8.49
Iris	97.14	4.99	92.14	5.27	93.57	6.25	95.71	4.99
Lymph	83.78	12.42	80.41	13.31	76.35	12.76	76.35	13.15
Average	84.44	6.90	81.56	7.25	83.05	7.10	80.45	7.46

Table 9.13: The search effort required by different specialization models.

	FuzzyBexa		FuzzySeedSearch		FuzzConRI		FuzzyPRISM	
Database	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
BreastCancer	4428.0	482.3	565.6	85.4	1324.4	246.6	25.6	9.2
Colic	11208.3	882.3	2353.8	248.9	10365.5	527.7	263.1	12.7
Credit-A	14198.1	856.5	2582.1	381.1	9070.1	641.9	1128.9	72.0
Hepatitis	2648.2	148.3	700.7	65.8	6411.3	308.2	108.7	16.6
Iris	283.6	34.3	119.9	30.0	283.9	26.2	47.4	8.9
Lymph	2973.6	254.4	790.3	73.0	6580.7	377.2	244.1	38.8
Average	5956.63	443.04	1185.40	147.37	5672.65	354.63	302.97	26.38

ond best. In a few domains, for example for the hepatitis data set, FUZZCONRI outperformed FEM. FUZZYSEEDSEARCH and FUZZYPRISM both obtained slightly worse classification accuracy results than FUZZYBEXA and FUZZCONRI, respectively.

Table 9.13 shows the search effort measured by the number of candidate hypotheses generated for the different specialization models. The good classification accuracy obtained by FEM clearly comes at the cost of increased search effort. FUZZYSEEDSEARCH, for example, required roughly six times less search. FUZZCONRI required slightly less search than FEM. The respective search effort results are another confirmation that some domains are more suited to FuzzyCAL than to FuzzyAL, and vice versa. For the BreastCancer domain, for example, FUZZCONRI required roughly four times less search than FEM, while for the Lymph Data FEM required roughly half the search effort compared to FUZZCONRI. Finally, FUZZYPRISM required extremely little search—on average about twenty times fewer hypotheses were generated by FUZZYPRISM than by FEM. This casts FUZZYPRISM's relatively bad classification accuracy in a new light. In very high dimensional domains FUZZYPRISM may be a sensible choice as specialization model.

9.9 Partial Covering

One may ask why are the sets P and N kept crisp—should a fuzzy set covering not use fuzzy sets for everything, including its internal representations such as P and N? (We could call such a covering process "partial covering" or "weighted covering".) The answer is not simply yes or no, and it is also

not obvious how to fuzzify P and N. First, we only need to investigate P as a fuzzy set, since the set covering approach does not remove elements from N. One approach is to assign to each instance a membership (or weight) and that is initialize to one. After a rule antecedent was found, the instances covered by it is not removed from the training set entirely, but their memberships to P are decreased. The question is, by how much? We briefly investigate three approaches. The first approach is to decrease the membership by the degree with which the instance matched the antecedent c, that is,

$$\mu_P(i)^t = \mu_P(i)^{t-1} - \mu_c(i) \tag{9.8}$$

where $\mu_P(i)^t$ is the membership of i to P at time t, and $\mu_P(i)^0 = 1$. It is interesting to look at alpha leveling in this case—should alpha-leveling also be applied to P, i.e. should instance memberships to P be set to zero if they fall below α_a? If we do apply alpha leveling, an α_a value above 0.5 would mean that covered instances will always be removed from P entirely, and similarly, the larger α_a is, the higher the probability to remove instances entirely. The next issue arising is how much credit the evaluation function should assign to a subsequent antecedent if it matches instances that were matched before. One option is to use the instance membership to P, $\mu_P(i)$, as a weight, and multiply the evaluation with the instance membership. The second partial covering approach is to remove i from P by a fraction determined by $\mu_c(i)$, that is,

$$\mu_P(i)^t = (1 - \mu_c(i))\mu_P(i)^{t-1} \tag{9.9}$$

The third approach is simply to decrease $\mu_P(i)$ by a constant factor each time i is covered. In this case,

$$\mu_P(i)^t = \gamma\mu_P(i)^{t-1} \tag{9.10}$$

where $\gamma \in (0, 1)$. This approach has the complication that instances are never fully removed from P, since their membership to P never reaches zero. Thus, in this case we are forced to apply alpha leveling.

It is clear that fuzzifying P is not a simple extension. Thus, it is important to ask "what do we expect to gain from partial covering?" The two goals of a rule learning algorithm are to induce accurate rule sets, and to keep these simple and comprehensible. Partial covering leaves positive instances that were already covered in the training set, while giving them less emphasis by reducing their contribution to the evaluation of a conjunction. Thus, partial covering may easily lead to the induction of many, largely overlapping rules. If the positive instances were all removed from the training set, the learning algorithm would be more likely to explore other disjoint sections of the hypothesis space. Thus, partial covering may yield slightly higher accuracy in some cases, but very often lead to the induction of slightly more accurate, but much larger rule sets.

We implemented the different strategies described above. The third strategy of decreasing $\mu_P(i)$ by a constant factor led to the induction of unacceptably large rule sets, where the same rule was often induced multiple times. The second strategy of reducing $\mu_P(i)$ by $\mu_c(i)$ led to smaller rule sets than the first method. However, the induced rule sets were still larger than using a crisp P. The first method described by Eq (9.10) yielded very similar results than the second method. Tables 9.14 and 9.15 compare respectively classification accuracy and rule set size results on different data sets for the normal (crisp P) implementation of FUZZYBEXA and the second and third approaches to partial covering described above. In this experiment we used the Accuracy evaluation function, $\alpha_a = 0.2$, *beamwidth* $= 1$,

Table 9.14: The classification accuracy for normal and partial covering strategies.

	Method 2		Method 1		Normal	
Database	Mean	StdDev	Mean	StdDev	Mean	StdDev
anneal	93.99	3.07	93.99	3.07	93.99	3.07
colic	85.05	5.13	85.05	5.13	84.51	4.33
credit-a	85.22	6.58	85.22	6.58	84.64	6.34
diabetes	72.92	3.41	72.79	3.43	72.14	4.02
hepatitis	83.87	12.17	83.87	12.17	84.52	9.83
iris	95.00	5.88	95.00	5.88	95.00	5.88
labor	87.72	16.54	87.72	16.54	87.72	16.54
lymph	82.43	12.25	82.43	12.25	78.38	12.82
	85.77	8.13	85.76	8.13	85.11	7.86

Table 9.15: The size of the rule sets for normal and partial covering strategies.

	Method 2		Method 1		Normal	
Database	Mean	StdDev	Mean	StdDev	Mean	StdDev
anneal	31.70	1.89	31.70	1.89	19.20	0.79
colic	13.50	1.51	13.50	1.51	5.10	0.99
credit-a	8.90	1.10	8.90	1.10	5.70	1.34
diabetes	7.80	3.03	11.00	5.96	4.80	1.48
hepatitis	8.50	1.18	8.60	0.97	4.70	0.67
iris	5.00	0.00	5.00	0.00	4.00	0.00
labor	6.80	1.32	6.80	1.32	3.90	0.57
lymph	13.30	1.34	13.30	1.34	5.90	1.20
	11.94	1.42	12.35	1.76	6.66	0.88

and $\theta_p = 2$ for all data sets. As expected, partial covering obtained slightly higher classification accuracy compared to the normal method, with the two partial covering approaches performing on average very similarly with respect to classification accuracy. However, the normal method induced substantially smaller rule sets, with the partial methods obtaining on average twice as many rules. Method 2 induced on average slightly smaller rule sets than method 1.

It is also important to take into account the impact on the search effort of partial covering. Table 9.16 compares the search effort in terms of the number of conjunctions explored. Method 2 required on average in double the search effort of the normal method, and method 1 resulted in a tripling of the normal search effort. In addition the computation of each step for partial covering is more expensive, since it requires more floating point operations compared to the normal method which removes instances without extra computation.

Our primary goal in this work is to investigate the induction of *comprehensible* fuzzy rule sets. For this goal, partial covering is not an attractive proposition. However, when high accuracy is more important than comprehensibility (and rule sets as a description language is desired), partial covering should be considered.

Table 9.16: The search effort for normal and partial covering strategies.

	Method 2		Method 1		Normal	
Database	Mean	StdDev	Mean	StdDev	Mean	StdDev
anneal	26520	1436	28700	2193	14471	746
colic	23617	1399	25903	1528	12501	845
credit-a	31771	3051	49658	6882	18317	977
diabetes	36426	1838	79972	4739	8582	573
hepatitis	5530	570	5899	743	3164	184
iris	353	21	498	64	234	19
labor	1174	200	1174	200	707	100
lymph	6214	685	6214	685	2970	275
	16451	1150	24752	2129	7618	465

9.10 Conclusion

FUZZYBEXA is the first algorithm that made use of the set covering methodology for the induction of fuzzy classification rules. Set covering has proven to be perfectly suitable for the induction of highly accurate and interpretable fuzzy rule sets. In this chapter we proposed the general fuzzy set covering framework, FCF. FCF consists of two top layers and a bottom layer implementing a specialization model. The top layers implement the fuzzy set covering methodology, and also apply various search heuristics for improving the performance of the framework. FCF is designed to allow the implementation of various specialization models with different description languages and specialization behaviour. Since different covering algorithms (both fuzzy and crisp) all fit within the same framework, they can easily be characterized and compared. In the remainder of the dissertation we use the term FCF to refer to the collection of fuzzy set covering algorithms that fit within the framework.

We also proposed four specializations models for the framework, thereby bringing the total number of fuzzy set covering algorithms proposed in this dissertation to four. FUZZYSEEDSEARCH and FEM both use FuzzyAL as description language, and we have shown that FEM is a more general algorithm performing a more thorough search of its description space, while FUZZYSEEDSEARCH incorporates seed instances to guide its search. FUZZYPRISM and FUZZCONRI's description language is FuzzyCDL. FUZZCONRI performs a more thorough search while FUZZYPRISM employs the fuzzy information gain to decide how to specialize. We compared the different specialization models by tracing the conjunctions generated during specialization in the lattice of concept descriptions of the respective description languages. Finally, we presented a comparison of the different specialization models with respect to classification accuracy and search effort which substantiated the expectation that the more general and thorough algorithms would obtain better classification accuracy, require more search.

CHAPTER 10

Simultaneous Concept Learning

10.1 Introduction

Learning multiple concepts generally follows one of two strategies. (1) For a concept (or class) in the data set, a set of disjunctive rules are induced by repeating the learning procedure for each concept in turn. (2) Multiple concepts are learned by finding a good classification rule for any one of the concepts, and assigning this class as consequent of the rule. The literature, e.g. [Mitchell, 1997], offers no preference for one strategy over the other. We call the two strategies for this process *iterated concept learning* (learning one class at a time, iterated over all classes) and *simultaneous concept learning* (simultaneously considering all classes by learning one rule at a time for any class, repeated until all data are covered), and abbreviate them as ICL and SCL, respectively. Examples of algorithms following the ICL strategy are FUZZYBEXA, BEXA, and Webb's rule learner, whereas C4.5, CN2, and Neural Networks all follow the SCL strategy [Cloete and van Zyl, 2006; Webb, 1993; Quinlan, 1996b; Clark and Niblett, 1989]. Fuzzy classification rules can be extracted from fuzzy decision trees and fuzzy neural networks, and although learning is done using SCL, unordered rule sets are obtained [Yuan and Shaw, 1995; Kasabov, 2001b].

FUZZYBEXA is the first algorithm to use set covering for the induction of fuzzy classification rules. The rule sets induced by FUZZYBEXA are unordered, and rules can be evaluated in any order. To the best of our knowledge, no work has been done on the induction of ordered fuzzy rule sets (also called decision lists), using any induction method. Here we mean that the induction method explicitly uses the order of rule induction, and not the ordering or prioritising of an unordered rule set after rule induction. The semantics of an ordered rule set is thus different from that of an unordered rule set. In an ordered rule set, as opposed to an unordered rule set, an instance is only matched against a rule (and the rule can thus only fire) if all previous rules did not fire. Thus, a single rule rule cannot be seen in isolation, and the antecedents of previous rules must also be considered.

In this chapter we introduce FUZZYBEXAII, the first fuzzy rule induction algorithm that induces *fuzzy decision lists*. FUZZYBEXAII makes use of SCL for its induction process. This induction process produces an ordered rule set, and we show that in many cases this methodology produces superior results compared to ICL, i.e. on average better classification performance, radically smaller rule sets, and also significantly less search effort. We also introduce the fuzzy Accuracy function for rule evaluation in

Table 10.1: FUZZYBEXAII's CoverConcepts procedure.

CoverConcepts
Input: Set of training instances T, Set of concepts to learn C
Output: A rule set describing the concepts
Set the current rule set to empty
While T contains instances
$best$ = FindBestRule(T, C)
Add $best$ to the rule set
Remove the instances covered by $best$
Return the rule set

SCL, and demonstrate that this function is much better behaved for SCL learning than, for example, the fuzzy Entropy function used during SCL in fuzzy decision trees.

The layout of the chapter is as follows. In Section 10.2 we show how to extend FUZZYBEXA to use the SCL strategy. In the next section we show that the rule evaluation function plays a pivotal role in finding good classification rules, and we introduction the Accuracy function for SCL. In Section 10.4 we provide the results of five different experiments on nine data sets for FUZZYBEXA with ICL and SCL using several different evaluation functions, as well as an empirical comparison between FUZZYBEXAII and other concept learners. The following section contains a discussion of the experimental data, and Section 10.5 concludes the chapter.

10.2 FUZZYBEXAII: Induction of Ordered Fuzzy Rules

In this section we introduce FUZZYBEXAII, a novel SCL approach that induces ordered fuzzy rules from a fuzzy data set. Table 10.1 shows FUZZYBEXAII's *CoverConcepts* routine. Compared to that of FUZZYBEXA, the SCL top layer routine of FUZZYBEXAII is less complex. It starts by initialising the rule set to empty. Then, it iteratively finds the best *rule* for the current set of training examples using the middle layer routine *FindBestRule*–in ICL the middle layer returned the *antecedent* that best covered the concept it was forced to use. For SCL the training set is not split into positive and negative parts, but passed as a whole to the middle layer. The rule found by the middle layer is then added to the rule set, and all instances covered by the rule are removed from the training set. This also differs from ICL, where only the positive instances covered by the rule are removed. We will discuss the implications of this decision later.

FUZZYBEXAII's middle layer, see Table 10.2, implements several heuristics for guiding the search in the hypothesis space. It uses the set $spec$ to maintain the set of current conjunctions to consider as rule antecedents. This set is initialized with the *mgc*. The routine functions as follows. A set of specializations of the conjunctions in $spec$ is obtained by invoking FUZZYBEXAII's bottom layer routine. Then, for each specialization ant in $spec$, the concept best described by the conjunction is selected. This is

Table 10.2: FUZZYBEXAII's FindBestRule procedure.

FindBestRule
Input: Set of instances, Set of concepts C
Output: The best rule found during this search
Set the current best rule to empty
Add the *mgc* to the set of current conjunctions, $spec$
While $spec$ contains conjunctions
$spec$ = GenerateSpecializations(T, $spec$)
For each conjunction ant in $spec$
Let $consequent$ be the concept from C best covered by the conjunction ant
If eval(ant, $consequent$) is better than that of the best rule,
Replace the current best rule with "IF ant THEN $consequent$"
If ant can never be better than the best rule, remove it from $spec$
Retain only the $beamwidth$ best conjunctions in $spec$
Return the best rule found

done by dividing the instances covered by the conjunction into groups G_i according to their class,

$$G_i(ant) = \{d \in X_T(ant) |\ \mu_{concept_i}(d) \geq \alpha_c\} \tag{10.1}$$

The sigma count or scalar cardinality of each group is then computed,

$$M(G_i(ant), ant) = \sum_{d \in G_i(ant)} \mu_{ant}(d) \tag{10.2}$$

and the concept of the group with the highest cardinality is chosen as the best rule consequent. The potential rule is then evaluated according to an evaluation function. This function is fundamental in guiding the search through the hypothesis space, and we will investigate its influence on the search process and overall performance in more detail later. If the potential rule outperforms the current best rule, it replaces the current best rule.

The next step implements an efficiency measure. This measure is very important to prevent unnecessary exploration of parts of the hypothesis space that cannot yield rules better than the current best rule. Let j be the index of the concept chosen as rule consequent. Assume that in the idealistic case all groups G_i, $i \neq j$, are empty. If even in this case the performance of the potential rule is worse than the best rule, it is futile to continue further exploration of this part of the hypothesis space. This is true since we are specializing antecedents, moving from top to bottom in the lattice of antecedents, and thus subsequent rules can never cover more instances, and therefore cannot increase their cardinality and performance above that of the best rule. Note, this test includes the consistency test as a special case—when an antecedent is consistent no subsequent antecedent can perform better than it. This test is an adaptation of an approach by Quinlan and Cameron-Jones for the crisp iterated concept rule learner by Webb [Quinlan and Cameron-Jones, 1995b; Webb, 1993]. After all conjunctions were considered, a beam search is implemented by retaining only the $beamwidth$ best conjunctions in the set $spec$. The process is iterated until $spec$ becomes empty and the best rule is returned.

Table 10.3: FUZZYBEXAII's specialization model.

GenerateSpecializations
Input: Set of instances T, set of conjunctions C
Output: Set of specializations of the conjunctions in C
$spec = \emptyset$
For each conjunction c and associated usable term L,
  If $X_T(L)$ and $X_T(c)$ have no instances in common,
    Mark this term as unusable in this conjunction
For each conjunction c and associated usable term L,
  Create a specialization by excluding L from c
  Add the specialization to $spec$
Remove all duplicate conjunctions from $spec$
Return $spec$

10.2.1 The Fuzzy Exclusion Model Using SCL

Table 10.3 shows FUZZYBEXAII's bottom layer routine, *GenerateSpecializations*. Here the fuzzy exclusion model is implemented. The function of this routine is similar to that of FUZZYBEXA, i.e. to obtain a set of specializations of the input set of conjunctions. The routine starts by initialising the set of specializations $spec$ to be empty. Then two loops follow . The first implements an efficiency measure, and the second performs the specialization. With each conjunction we associate a set of "usable" terms that may be used to specialize the conjunction, and we initialise the *mgc* to contain all terms in its usable set. The first loop compares the extension of the conjunction and the extension of terms from its usable set. Any term where the two extensions have no members in common, i.e. any term L and conjunction c where

$$X_T(L) \cap X_T(c) = \emptyset \tag{10.3}$$

is removed from the set of usable terms for this conjunction. Excluding such a term will not change the extension of the conjunction, and therefore make it overly specific. The next loop generates specializations by excluding from each conjunction the terms from its associated usable set in turn. Duplicates may occur if two conjunctions were specialized by excluding the same terms in different order, and are removed. The resulting specializations are returned.

10.2.2 Other Specialization Models Using SCL

FUZZYBEXAII's specialization behaviour can easily be adjusted by exchanging the specialization model implementation. Table 10.4 shows the FUZZYBEXAII using FUZZCONRI as specialization model. Each conjunction c is specialized by appending one of the remaining usable terms to it. The new extension in the training set can be efficiently computed using $X_T(cnew) = X_T(c) \cap X_T(L)$, since the conjunction with L restricts the extension to only allow instances that match L. If the extension becomes empty the conjunction is not useful and is discarded. Finally, if the conjunction was not already present in $spec$ it

Table 10.4: FUZZYBEXAII using FUZZCONRI as specialization model.

GenerateSpecializations
Input: Set of instances T, set of conjunctions C
Output: Set of specializations of the conjunctions in C
$spec = \emptyset$
FOR each conjunction c and associated usable term L DO
$cnew = c \wedge L$
$cnew.usable = cnew.usable - L$
$X_T(cnew) = X_T(c) \cap X_T(L)$
IF $X_T(cnew) = \emptyset$ THEN
CONTINUE
IF $cnew \notin spec$ THEN
$spec = spec \cup \{cnew\}$
Remove all duplicate conjunctions from $spec$
Return $spec$

is added to it. FUZZYSEEDSEARCH may be adapted to implement the SCL strategy in a similar way. In this case care must be taken that the seeds are from different classes.

10.3 The Rule Evaluation Function

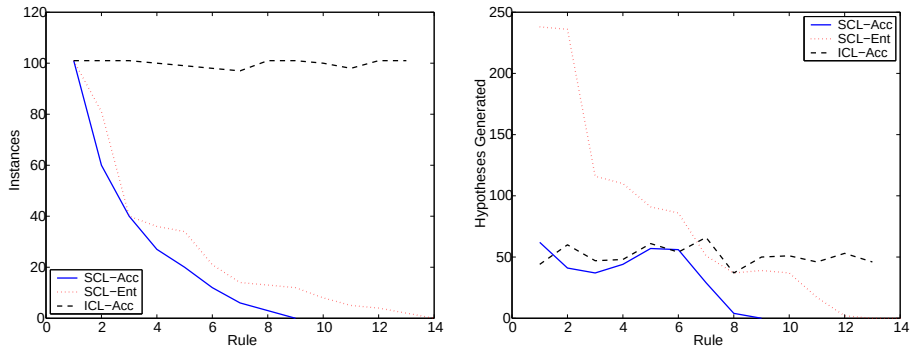
The entropy evaluation function is often used for SCL learning, including decision tree and fuzzy decision tree learning [Cios and Sztandera, 1992; Dong and Kothari, 2001]. Let r be a rule with a as antecedent and $\{c_1, ..., c_N\}$ the possible consequents of r, then the normalized fuzzy entropy is given by

$$E(r) = \frac{1}{\log N} \sum_{i=1}^{N} \frac{M(T, a \wedge c_i)}{M(T, a)} \log \frac{M(T, a \wedge c_i)}{M(T, a)} \tag{10.4}$$

where $M(T, x)$ is the sigma count of the expression x in the set of instances T. Since we want an evaluation function that assigns higher scores to better conjunctions, we use the evaluation function $1 - E(r)$. This function has a maximum value of one for rules that cover only one class, and a minimum value of zero for rules that cover each class in the same proportion. However, the Entropy function does not favour high coverage, e.g. a rule that covers five instances of one class and none of other classes and a second rule that covers a thousand instances from one class and none of other classes will both have a score of one. The Laplace estimate was suggested as an improvement to the CN2 algorithm that also used the Entropy function [Clark and Boswell, 1991]. In Chapter 6 we suggested the fuzzy Accuracy function for ICL,

$$A_{\text{ICL}}(r) = \sum_{i \in X_P(a)} \mu_a(i) - \sum_{i \in X_N(a)} \mu_a(i) \tag{10.5}$$

where P is a subset of T containing all instances that belong to the concept, and $N = T - P$. We adapt the Accuracy function for use in SCL by considering each concept in turn, and regard instances belonging to other concepts as members of N. We assign the rule consequent as the concept that results



(a) Uncovered instances remaining to be used during rule induction.

(b) The number of candidate rules generated during rule induction.

Figure 10.1: Results for ICL and SCL with different evaluation functions on the Zoo data.

in the highest evaluation, and also assign this evaluation value to the rule, that is,

$$A_{\text{SCL}}(r) = M(G_j(a), a) - M(X_T(a) - G_j(a), a) \tag{10.6}$$

and

$$j = \underset{i}{\text{argmax}}(M(G_i(a), a)) \tag{10.7}$$

where $G_i(a)$ is defined by Eq (10.1). This evaluation function will prefer rules that cover a large number of instances from one concept and few instances from the other concepts.

10.4 Experiments

In this section we show experimental results on six real world domains obtained from the UCI machine learning repository. We fuzzified data by assigning membership values from $\{0, 1\}$ to nominal attributes, and by using a clustering method to place bell shaped membership functions on the continuous domains of linearly ordered attributes. We will discuss results obtained for FUZZYBEXA (ICL) with the accuracy and Laplace evaluation functions, and also for FUZZYBEXAII (SCL) with the entropy and accuracy evaluation functions. We denote FUZZYBEXAII with the entropy and accuracy evaluation functions as SCL-Ent and SCL-Acc respectively, and FUZZYBEXA with the accuracy and Laplace evaluation functions as ICL-Acc and ICL-Lap, respectively.

10.4.1 Fuzzy Rule Induction With ICL and SCL

Figure 10.1 shows results obtained by SCL and ICL on the training set of the Zoo data, where we ignored the variable "animal," and learned the concept "type of animal," e.g. mammal, bird, fish, etc. The different methodologies of SCL and ICL are clearly discernable from Figure 10.1(a). For most of the rules, ICL considered all the instances during the induction process. This happens since ICL removes only the positive instances covered from the training set for each class, and *reinserts* these into the training set when the next concept is considered. SCL, however, *never reinserts* covered instances.

The SCL graph of remaining uncovered instances is thus monotonely decreasing. From Figure 10.1(a) one can also see the number of instances covered by each consecutive rule. This is indicated by the difference on the y-axis of two consecutive points. When there is no difference for ICL, it implies that the rule covered all the positive instances. The last seven rules induced by SCL-Ent covered very few instances each. Figure 10.1(b) shows the number of candidate hypotheses generated for each rule during the search. SCL-Ent started out with a very high number, and then, as there were successively fewer instances available, generated successively fewer candidates rules. For the first six rules SCL-Acc and ICL-Acc had similar behaviour. However, for the last two rules of SCL-Acc there were less than 10 instances, and consequently it searched only a few hypotheses before covering them. The use of the Accuracy function also resulted in a much smaller rule set for SCL. SCL-Acc had 9 rules and SCL-Ent 14 rules.

Table 10.5 shows results for five experiments. All results quoted are on test set results from a 10-fold cross validation. For each data set the mean and standard deviation were computed, and the average of the means of all data sets are shown in the last column. The best performance on each data set is set in bold face. The first experiment investigated the accuracy of the induced rule sets. SCL-Ent had the worst overall performance, and did significantly worse on the Colic, Hepatitis and Lymph data sets. It had the best performance on the Zoo data set. SCL-Acc, in contrast, performed very well, and obtained better overall results than any of the other methods. ICL-Acc and ICL-Lap had very similar results, and was overall about 2% worse than SCL-Acc, but 3.5% better than SCL-Ent. The second experiment compared the size of the rule set induced by each method. Here, SCL-Acc was the clear winner. On average its rule sets contained about three times fewer rules than ICL-Acc and ICL-Ent. It also became clear that SCL-Ent is not a good method to use, as it induced 12 times more rules than SCL-Acc, and also had worse classification accuracy performance. This result is most likely due to the entropy evaluation function not favouring conjunctions with higher coverage. Thus, a large number of consistent conjunctions covering only small sets of instances are induced.

One obvious observation is that SCL-Acc is able to induce extremely compact rule sets. This behaviour cannot be attributed only to the Accuracy function, as ICL-Acc did not perform as well. One big difference between SCL and ICL is that the rules induced by SCL are ordered and that by ICL unordered. Table 10.6 shows the rule set induced by SCL for the Zoo data. The first rule correctly classifies all mammals. Thus, after the first iteration, all mammals are removed from the data set. Similarly, the second rule removes all birds from the data set. Now consider the third rule, it states that animals with fins are fish. On its own, this rule would incorrectly classify whales and dolphins as fish. However, since the rules are evaluated in order, the first rule would fire for a whale, correctly classifying it as a mammal, and further rules would not be considered.

We believe the aforementioned characteristic is present in many data sets, and is the reason why SCL outperforms ICL on many data sets. After the first few rules took care of macro features that are easily identified, rules found later need not concern themselves with these features, and can distinguish between the special cases. An ordered rule set is a representation of a more complex unordered rule set, and also does not require the arbitration process of unordered rule sets when multiple rules fire. When ICL has

Table 10.5: Various test results for SCL with the Entropy and Accuracy evaluation functions and ICL with the Accuracy and Laplace Evaluation functions.

	Colic		Diabetes		Hepatitis		Iris		Lymph		Zoo		Ave
	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean
	Accuracy of the Rule Set												
SCL, Ent	78.0	6.0	68.2	2.7	76.8	11.2	93.6	6.3	73.6	12.1	**96.0**	**8.2**	81.0
SCL, Acc	84.5	3.3	**74.0**	**4.8**	**86.5**	**6.9**	**96.4**	**5.1**	81.1	12.3	**96.0**	**10.5**	**86.4**
ICL, Acc	**85.3**	**4.7**	71.2	4.1	83.9	10.1	95.7	6.0	81.1	11.5	91.1	11.2	84.7
ICL, Lap	83.4	5.7	71.1	2.8	80.6	7.6	**96.4**	**5.1**	**81.8**	**14.4**	93.1	13.4	84.4
	Number of Rules in the Rule Set												
SCL, Ent	88.0	7.8	183.2	16.5	34.0	2.1	9.9	0.3	41.8	3.5	12.7	0.7	61.6
SCL, Acc	**5.1**	**0.3**	4.4	1.7	**3.1**	**0.3**	**3.9**	**0.3**	**6.5**	**0.7**	**7.9**	**0.3**	**5.2**
ICL, Acc	34.4	2.7	**2.4**	**0.5**	19.3	1.2	4.0	0.0	19.6	1.5	12.3	1.1	15.3
ICL, Lap	34.5	1.6	8.6	1.1	19.0	1.1	4.3	0.5	21.8	1.3	10.6	0.7	16.5
	Complexity of the Rule Set Measured in Terms												
SCL, Ent	184.4	20.8	759.2	79.1	56.0	4.3	13.5	1.0	70.0	8.6	12.5	1.1	182.6
SCL, Acc	**14.2**	**2.8**	**5.6**	**2.9**	**4.5**	**1.4**	**3.5**	**0.5**	**12.7**	**1.7**	**10.3**	**1.3**	**8.5**
ICL, Acc	128.0	9.6	6.2	1.9	62.9	5.0	6.0	0.0	67.9	7.2	35.4	3.9	51.1
ICL, Lap	166.7	12.1	36.6	6.0	68.3	5.9	7.0	1.6	76.5	5.3	27.9	2.2	63.8
	Number of Conjunctions Generated During Rule Set Induction												
SCL, Ent	53800	5632	24420	2349	11300	744	334	24	8008	722	955	134	16470
SCL, Acc	**2409**	**192**	**355**	**110**	**632**	**74**	**123**	**15**	**907**	**55**	**315**	**23**	**790**
ICL, Acc	10137	637	6780	436	2360	122	196	13	2243	222	601	61	3719
ICL, Lap	12373	1068	5851	306	2592	251	263	27	2719	168	493	38	4048
	Average Number of Hypotheses Generated per Rule												
SCL, Ent	610.9	32.4	132.6	2.5	332.0	13.9	33.4	3.1	191.3	15.0	74.5	7.4	229.1
SCL, Acc	472.1	28.1	83.4	11.1	203.7	12.0	31.3	4.0	139.9	11.4	39.5	3.0	**161.7**
ICL, Acc	294.6	11.4	2924.8	579.9	121.8	4.5	48.6	3.4	113.8	5.2	48.3	2.1	592.0
ICL, Lap	357.6	19.5	689.8	101.5	135.9	10.5	61.1	7.4	124.7	8.8	46.0	0.9	235.9

to induce a rule for fish, it will have to find a more complex antecedent, e.g. [milk.false][fins.true], i.e. the rule must not fire on any of the macro features, but still differentiate the special cases. Consequently, ordered rule sets can be much smaller than unordered rule sets, while still obtaining high accuracy. ICL often induces many more rules to prevent the covering of macro features while still covering some of the micro features. The small number of instances available for induction of the last rules in SCL implies that less search is necessary for these rules. This is different for ICL and clearly visible in Figure 10.1(b)—the number of hypotheses examined per rule remains relatively constant for ICL but is very small for the last few rules for SCL. The overall result is that SCL-Acc requires less search for rule set induction. The rule sets induced by SCL are also not unnatural, as humans also represent concepts such as animal type using an ordered rule set, i.e. reasoning by working with exceptions. The last rule induced by SCL often has the antecedent TRUE. This happens when after the exclusion of instances covered by previous rules, only instances of one class remain. This must not be confused with the default rule used in unordered rule sets. In unordered sets, the default rule fires when no other rule fires, and usually has the majority class as consequent. SCL could also employ such a default rule when the last rule does not have TRUE as antecedent.

SCL in combination with the Entropy function did not perform well. This is because entropy does not

Table 10.6: SCL-Acc induced rule set for the Zoo data.

[milk.true] $\rightarrow$ type.mammal	[eggs.true][backbone.false][legs.$\neg\bar{\alpha}$] $\rightarrow$ type.insect
[feathers.true] $\rightarrow$ type.bird	[backbone.true][tail.true] $\rightarrow$ type.reptile
[fins.true] $\rightarrow$ type.fish	[aquatic.false] $\rightarrow$ type.invertebrate
[eggs.true][breathes.false] $\rightarrow$ type.invertebrate	TRUE $\rightarrow$ type.amphibia

guide the search in the direction of high coverage. The first rule induced by SCL-Ent, for example, had "bird" as consequent. However, there are 20 bird and 41 mammal instances. Thus, SCL-Acc induced a rule for the class with the most instances since this rule has the highest coverage. On the Colic data SCL-Acc alternated between the classes such that the most instances are covered by each consecutive rule. Subsequent rules should in general cover fewer instances than previous rules, thus rules with stronger support are placed higher up in the rule hierarchy. This can be clearly seen in the shape of the graph for SCL-ACC in Figure 10.1(a). SCL-Ent in the same figure, however, had subsequent slopes higher than previous slopes, demonstrating its unbiasedness towards high coverage.

The third experiment in Table 10.5 measured the complexity of rules as the number of terms in the rule set. Here, the good performance of the Accuracy function for both SCL and ICL is evident. Again SCL-Acc had the best performance, requiring six times fewer terms than ICL-Acc and seven times fewer than ICL-Lap. The rule sets found by ICL-Acc were about 15% less complex than that found by ICL-Lap. The rule set complexity found by SCL-Acc was on average about 5% of that of SCL-Ent. The fourth experiment shows that, interestingly, SCL-Acc needed to investigate only a very small part of the search space to obtain its results. ICL-Acc was second, but generated 4.6 times more candidate rules, whereas ICL-Lap generated 5.2 times more candidates. SCL-Ent's struggle to obtain good rule sets becomes clear; it generated 16470 hypotheses versus ICL-Acc's 790. The last experiment compares the number of hypotheses generated per induced rule. SCL-Acc again needed the least number of hypotheses. Interestingly though, SCL-Ent generated the second least. However, since the induced rules cover so few instances, many rules were needed making the total search very large. ICL-Acc generated the most hypotheses. However, if we remove the outlier of the Diabetes data, the ICL-Acc would have 125.4, ICL-Lap 145.1, and SCL-Acc 177.3, resulting in ICL-Acc with the smallest search per rule. ICL-Acc induced the smallest and second most accurate rule set for the diabetes data. However, it required 20 times more search than SCL-Acc, and therefore a very large number of hypotheses were generated per induced rule.

10.4.2 Comparison With Other Concept Learners

We also compared our FUZZYBEXAII algorithm (using the Accuracy function) with three other concept learners. The results quoted for C4.5, Layered Search and Exhaustive Search were obtained from the literature [Quinlan and Cameron-Jones, 1995b; Quinlan, 1996a,b]. The first column shows the average error on the test sets. FUZZYBEXAII had similar classification results as the other methods for the

Table 10.7: Results of FUZZYBEXAII, C4.5, Layered Search and Exhaustive Search on three data sets. Theory size for C4.5 is measured in tree nodes, in number of test conditions for layered and exhaustive search, and in number of terms for FUZZYBEXAII.

	Error			Theory Size		
	Diabetes	Hepatitis	Lymph	Diabetes	Hepatitis	Lymph
C4.5	25.4	20.4	21.7	44.0	17.8	N/A
Layered Search	26.9	19.1	18.9	207.4	27.0	30.1
Exhaustive Search	27.2	20.0	19.0	208.7	27.9	30.1
FuzzyBexaII	23.0	13.6	18.9	5.6	4.5	12.7

Lymph data, better results on the Diabetes data, and significantly better results on the Hepatitis data. It's theory size (complexity) is also significantly smaller in all cases.

10.5 Summary

This chapter presented FUZZYBEXAII, an algorithm for learning ordered fuzzy rule sets for classification. We also enhanced the method with early stopping efficiency measures, without which the search would be prohibitively large. We further presented five empirical experiments on six data sets, and demonstrated that if the correct kind of evaluation function were used, i.e. functions that give preference to rules with high coverage, ordered rule sets are much less complex than unordered rule sets, while at the same time being very accurate. As an example of an appropriate evaluation function we showed how to adapt the fuzzy Accuracy function for SCL. We discussed the various reasons for SCL's good performance, and also showed with further experiments that FUZZYBEXAII can outperform other learning systems with respect to rule set size and accuracy.

CHAPTER 11

Arguments in Favour of Fuzzy Set Covering

11.1 Introduction

In the previous chapters we developed fuzzy set covering as a new methodology for fuzzy rule induction. In this chapter we address the validity of this approach as a concept learner. We assume a rule representation is desired, and thus we do not focus on the more general case of symbolic versus numeric concept learning. We proceed to demonstrate two main points, (a) fuzzy rules, as a generalization of crisp rules, are more powerful than crisp rules for several reasons, and (b) set covering as a methodology for fuzzy rule induction performs very well compared to other fuzzy rule learners with respect to classification accuracy, and especially with respect to comprehensibility.

The layout of the chapter is as follows. Section 11.2 addresses point (a) by presenting a set of theoretical arguments in favour of fuzzy set covering. Section 11.3 demonstrates FUZZYBEXA's performance compared to the well-known decision tree learner C4.5 and one of the most powerful rule induction algorithms, RIPPER[1]. In Section 11.4 we provide empirical results in support of point (b), and Section 11.5 further emphasizes this point by providing results where FCF significantly outperforms previous methods on two real world applications (the classification of SPAM and the prediction of mortality in septic shock patients). We conclude the chapter with a summary in Section 11.6.

11.2 Fuzzy Versus Crisp Rule Learning

An element i belongs to a fuzzy set A to a certain degree, typically in the range $[0, 1]$, as defined by the membership function $\mu_A(i)$ associated with the fuzzy set. A crisp set is a special case of the more general fuzzy set, where in the crisp case the membership function (characteristic function) assigns membership degrees from the *set* $\{0, 1\}$. Thus, it is not surprising that rules based on fuzzy sets would be more powerful with respect to representation power and modelling capability. In fact, it has been proven that fuzzy sets can be used as universal approximators [Kosko, 1994].

In addition to their improved modelling capability, fuzzy sets provide a natural mechanism for dealing with linearly ordered domains. Crisp rule learners either do not allow such attributes, or resort to defin-

[1] RIPPER is an acronym of Repeated Incremental Pruning to Reduce Error Reduction

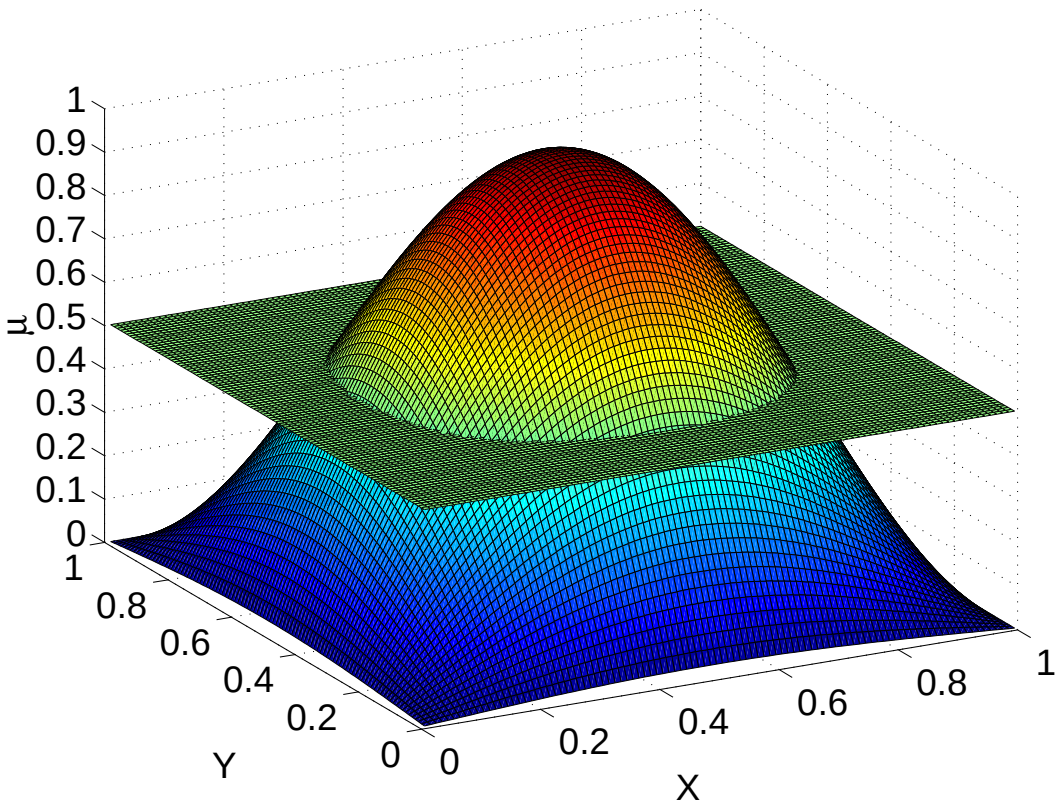


Figure 11.1: Membership functions and an α-cut plane at 0.5 for the linguistic terms X and Y.

ing ranges on the domain. These ranges, however, cannot represent the real world concept of gradual transition from membership to non-membership, as encountered in most natural domains. For example, the transition from the concept *hot* to the concept *cold* is not sharp, and cannot be pinpointed at a certain temperature. Fuzzy rules model numeric domains with linguistic terms. Each linguistic term is defined by a fuzzy set, which deals with such transitions in a natural and comprehensible manner. Furthermore, fuzzy rules do not make the unnatural distinction between nominal and numerical domains, but treat all the same.

A crisp instance can only have a single attribute value for a given attribute. Thus, there is no scope for uncertainty, ambiguity, or vagueness. The use of fuzzy sets, however, allow these real world concepts to be modelled in a mathematically sound way. A fuzzy instance can belong to several linguistic terms from the same variable simultaneously, e.g. an instance may be *hot* to degree 0.7 and *cold* to degree 0.4.

A fuzzy instance matches fuzzy conditions (i.e. the antecedent of a fuzzy rule) to a certain degree, whereas a crisp instance can either match a rule or not. Thus, in the fuzzy case additional information is available to the inference system during classification of unlabeled instances. For example, the degree to which unseen instances match a conjunction can be compared to those observed for instances from the training set. This information can be used, for example, for rule conflict resolution. Large deviations from the observed mean may indicate that the rule should not be used, or that the instance is a novelty, i.e. it lies outside of the observed distribution for this rule.

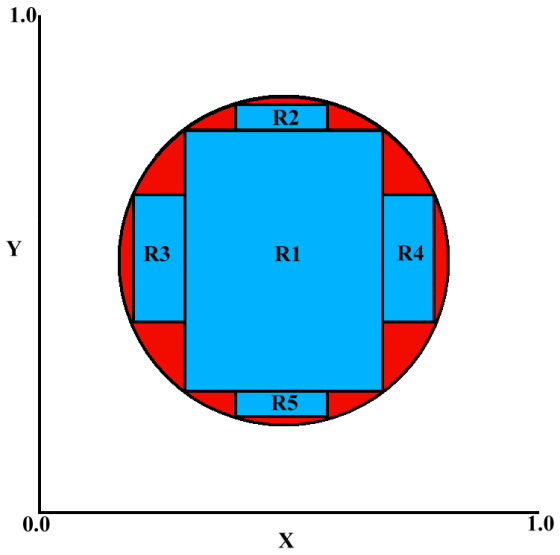


Figure 11.2: Crisp rules used to classify points inside and outside of the circle. Each blue region is correctly covered by a rule, while the red regions should have been covered, but are not.

Another fundamental difference between crisp and fuzzy rules is that in the fuzzy case decision boundaries need not be axis-parallel. This is illustrated in Figure 11.1 for the rule condition $X \wedge Y$, and product as the t-norm operator. Both linguistic terms X and Y have the same domain, the range $[0, 1]$, and their respective membership functions to an instance i are $\mu_X(i) = sin(\pi x)$ and $\mu_Y(i) = sin(\pi y)$. For the classification task of identifying points inside and outside the circle with origin at $(0.5, 0.5)$ and radius $\frac{1}{3}$, the single rule,

$$\text{IF } [X][Y]@0.5 \text{ THEN } inside \tag{11.1}$$

provides a perfect classification. Crisp learners are forced to approximate the decision boundary, and use more and more rules to increase their accuracy. Figure 11.2 shows the crisp approximation where five rules were used. A perfect classification in the crisp case can only be achieved with infinitely many rules.

11.3 Empirical Comparison with State of the Art Concept Learners

The previous section provided theoretical arguments why fuzzy rule learners are more powerful than crisp rule learning algorithms. In this section we provide empirical evidence that fuzzy set covering is a powerful rule induction methodology, and capable of competing with state of the art concept learners. As stated before, we do not compare ourselves to numerical methods such as support vector machines or neural networks, but to methods that provide an explanation for their prediction. The two major concept representations that explain their classification are decision trees and rule sets. C4.5, the successor of ID3, is probably the best-known decision tree learner, and has proven very successful over time [Quinlan, 1993a]. RIPPER [Cohen, 1995] is arguably the most powerful rule induction algorithm available today [Fürnkranz and Flach, 2004]. Both C4.5 and RIPPER employ a post-pruning phase, and RIPPER also

Table 11.1: Classification accuracy results for state of the art concept learners.

	FCF					RIPPER		C4.5		OneR
Database	ICL	ICL, Acc, θp=2	SCL, Acc, θp=0	SCL, Acc, θp=2	SCL, Acc, θp=5	Unpruned	Pruned	Unpruned	Pruned	
Anneal	99.00	93.76	93.65	93.43	93.32	98.44	98.33	98.44	98.44	83.63
BreastCancer	73.02	73.14	71.16	71.74	71.86	70.98	77.27	68.2	70.28	69.93
Colic	85.60	85.33	85.33	85.33	85.87	78.80	85.05	82.34	84.51	81.52
Credit-A	85.80	85.07	85.94	85.94	85.94	85.07	86.23	81.88	83.33	85.51
Digit	72.72	69.47	73.91	72.58	71.72	72.40	74.60	74.20	72.60	21.40
Hepatitis	81.29	82.58	81.94	82.58	82.58	81.29	78.06	80.65	83.87	81.29
Iris	97.14	95.00	96.43	96.43	96.43	92.00	94.67	96.00	96.00	94.00
Labor	91.23	87.72	89.47	89.47	89.47	84.21	77.19	78.95	73.68	75.44
Lymph	83.78	79.05	81.08	79.05	78.38	79.73	79.05	75.68	77.03	74.32
Average	85.51	83.46	84.32	84.06	83.95	82.55	83.39	81.81	82.19	74.12

includes incremental pruning and multiple rule set optimisation phases (for a description of RIPPER see Appendix A).

We compare FCF in several configurations with three concept learners, C4.5, RIPPER, and 1R, which builds a single rule per class [Holte, 1993], on several benchmark data sets. Table 11.1 shows the accuracy of the various learners on the data using 10-fold cross-validation. The results for C4.5, RIPPER, and 1R were obtained using the WEKA[2] package [Witten and Frank, 2000]. RIPPER and C4.5 were configured first not to use post-pruning, and then to include post pruning, in which case RIPPER was allowed to optimise the rule set twice (thus leading to RIPPER2). We configured FCF in several ways. The first column used ICL and evaluation functions suited to the domain. In the remaining columns we used the Accuracy evaluation function for all data sets, which clearly had a detrimental effect on the Anneal data, for example. We also set θ_p to different values ranging from 0 to 5, and we applied SCL where indicated.

The bottom row of the table shows the average classification performance over all data sets. On average 1R is outperformed by all methods, although on some data sets 1R obtained similar performance. Elomaa [1994] discusses the results of 1R versus other concept learners, and provides arguments why the small improvement in accuracy obtained by C4.5 is still significant. FCF with ICL (first column of the table) obtains almost 3% higher classification accuracy than RIPPER unpruned, and still over 2% higher than RIPPER2. FCF compares even better with C4.5, and outperform the unpruned and pruned versions with 3.7% and 3.2%, respectively. It is also clear that the post-pruning phases in RIPPER and C4.5 improved their performance—something FCF does not have the benefit of. We already did preliminary work on the post-pruning of fuzzy rules, but future research can improve upon this further [Robbel, van Zyl and Cloete, 2004].

Table 11.2 shows the average number of rules per rule set for the different learners and data sets. We do not show the number of rules for 1R, since this is constant. Here the beneficial effect of pruning on the rule set size is clearly demonstrated for RIPPER and C4.5. RIPPER is able to reduce the average number of rules from 13.33 to 5.22, while C4.5 was able to reduce the rule set, as measured by the number of leaf nodes in the tree, from 45.33 to 17.78. The high accuracy obtained by FCF configured with ICL and suitable evaluation functions resulted in a comparatively large rule set, although still

[2]WEKA is an acronym of Waikato Environment for Knowledge Analysis

Table 11.2: Average number of rules per rule set for state of the art concept learners.

	FCF					RIPPER		C4.5	
Database	ICL	ICL, Acc, θp=2	SCL, Acc	SCL, Acc, θp=2	SCL, Acc, θp=5	Unpruned	Pruned	Unpruned	Pruned
Anneal	37.7	23.0	9.0	9.0	7.0	12.0	7.0	53.0	35.0
BreastCancer	32.9	10.0	5.0	2.0	2.0	12.0	2.0	59.0	24.0
Colic	44.8	3.0	5.0	4.0	4.0	14.0	3.0	95.0	17.0
Credit-A	55.1	6.0	4.0	3.0	2.0	18.0	4.0	101.0	30.0
Digit	42.5	29.0	23.0	15.0	11.0	34.0	13.0	32.0	16.0
Hepatitis	22.6	5.0	3.0	3.0	2.0	9.0	4.0	16.0	11.0
Iris	5.3	3.0	4.0	3.0	3.0	5.0	4.0	5.0	5.0
Labor	8.0	3.0	3.0	2.0	2.0	4.0	4.0	13.0	3.0
Lymph	27.5	4.0	6.0	5.0	4.0	12.0	6.0	34.0	19.0
Average	30.71	9.56	6.89	5.11	4.11	13.33	5.22	45.33	17.78

much smaller than the unpruned decision trees. The Accuracy evaluation function clearly results in a reduction of the rule set size, while classification performance is not severely influenced. For example, a run of FCF with SCL, the Accuracy evaluation function, and $\theta_p = 5$ obtained only 1.56% worse classification accuracy than the best classification accuracy configuration (first column), but the average rule set size was reduced from 30.71 to 4.11 rules. This is radically smaller than the pruned decision tree, and also smaller than RIPPER2. At the same time this configuration obtained 1.76% and 0.56% better classification accuracy than C4.5 pruned and RIPPER2, respectively. Thus, we can conclude that FCF compares well with RIPPER and outperforms C4.5 on the benchmark data sets with respect to both classification accuracy and rule set interpretability (as approximated by rule set complexity).

11.4 FCF versus Other Fuzzy Rule Learners

Set covering is one of the most successful machine learning methodologies in the crisp case, and many different algorithms were proposed for this methodology [Fürnkranz, 1999]. The fuzzy set covering methodology proposed in this dissertation extends these principles to the fuzzy realm, keeping the crisp case as a special case of the more general fuzzy case. It will thus not be very surprising that the fuzzy set covering methodology is also capable of competing with the fuzzy generalization of other crisp symbolic methodologies, such as decision trees or simple beam search. In this section we will seek to provide some empirical proof of this expectation. Note, the term FCF is used to describe any fuzzy set covering algorithm that fits with the general fuzzy set covering framework, as discussed in Chapter 9.

We will be interested primarily in the quality of the rule sets induced, as measured by the rule set size and accuracy. The fuzzy systems investigated in this section all satisfy two of the three conditions for interpretable fuzzy rule sets, they all produce incomplete rules, and they all make use of defined fuzzy sets as linguistic terms [Guillaume, 2001]. Guillaume's third requirement is that the rule set should be as small as possible. When the size of the rule set becomes very large, the inherit assumption that fuzzy rules are more comprehensible disappears. In this case the question arises, what is gained by fuzzy rules as opposed to powerful numeric methods such as support vector machines or neural networks?

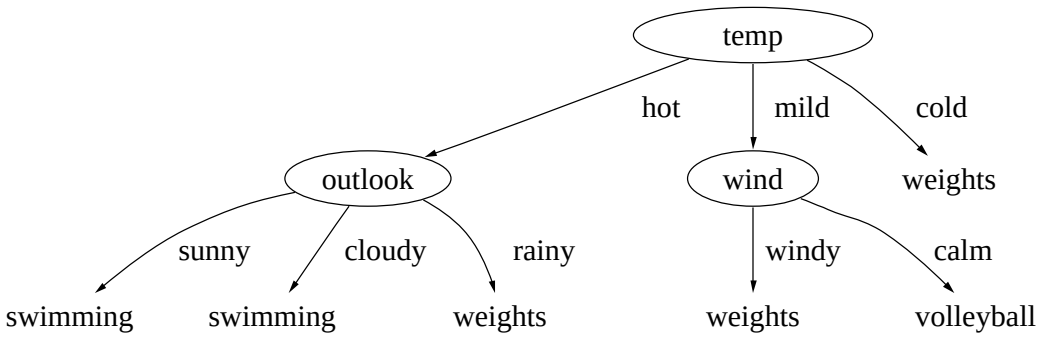


Figure 11.3: Fuzzy Decision tree induced by Yuan *et al* for the Fuzzy Sport data.

11.4.1 The Sport Problem

The results of Yuan *et al*'s fuzzy decision tree induction algorithm [Yuan and Shaw, 1995] on their sport data set (see Table 4.1), are reproduced here. Their algorithm requires two parameters, the truth level and the evidence level thresholds. The reported values for this experiment were 0.7 and 0.5, respectively. The decision tree shown in Figure 11.3 was induced, and from the tree the following rule set was extracted:

1 IF $[sunny][hot]$ THEN $swimming$

2 IF $[cloudy][hot]$ THEN $swimming$

3 IF $[rainy][hot]$ THEN $weights$

4 IF $[mild][windy]$ THEN $weights$

5 IF $[mild][calm]$ THEN $volleyball$

6 IF $[cool]$ THEN $weights$

This rule set classifies 81% of the data set correctly. Setting FCF's corresponding parameters α_a and α_c to 0.5 and 0.7 respectively, with $\theta_p = 0$ and $beamwidth = 1$, and using FEM (Fuzzy Exclusion Model, see Section 9.3), the following rules were induced:

1 IF $[sunny, cloudy][mild, cold]$ THEN $volleyball$

2 IF $[sunny][humid]$ THEN $swimming$

3 IF $[mild, cold][windy]$ THEN $weights$

4 IF $[rainy]$ THEN $weights$

where for brevity we do not show α_a and α_c for each rule. The classification accuracy of these rules for the data set is 94%. FCF found four rules, compared to six, with higher classification accuracy.

11.4.2 Comparison with Fuzzy Classifiers on Real World Data

For the final experimental evaluation, we compared FCF to the fuzzy classier learners FID [Janikow and Fajfer, 1999] and FBS (Fuzzy Beam Search) [Fertig et al., 1999]. FID uses the divide-and-conquer strategy to induce fuzzy decision trees (as discussed in Section 2.4), and FBS performs a

Table 11.3: Accuracy and complexity results for different fuzzy classifiers on seven real world domains.

Data Set	credit-a		labor		lymph		iris		diabetes		hepatitis		colic		average	
classification accuracy																
FB	85.1	5.2	91.2	13.8	80.4	11.2	96.4	5.1	75.5	2.4	82.6	11.3	85.6	5.5	85.3	7.8
FID	82.2	7.3	89.5	9.3	73.0	10.4	92.9	3.4	72.1	4.7	83.5	11.5	85.9	4.4	82.7	7.3
FBS	69.4	6.2	91.2	13.8	60.1	15.1	91.2	13.8	69.9	3.5	83.9	8.6	74.2	7.9	77.1	9.8
number of rules																
FB	7.1	1.0	4.9	0.7	8.0	1.3	5.0	0.0	8.8	1.3	5.0	0.7	5.6	1.4	6.3	0.9
FID	69.0	51.6	11.9	3.8	8.6	1.8	7.0	2.5	20.4	21.6	12.7	6.4	24.3	7.5	22.0	13.6
FBS	50.0	-	10.0	-	50.0	-	10.0	-	50.0	-	50.0	-	50.0	-	38.6	-

beam search of its hypothesis space (see Section 2.3 for a detailed discussion of FBS). Table 11.3 shows the comparative results for seven data sets from the UCI repository. We report the classification accuracy and the size of the induced rule sets. The table shows the mean and standard deviation of the test set results of a 10-fold cross validation, except for Diabetes where a 5-fold cross validation was performed.

The results for FID were obtained using a freely available implementation[3]. We used the default parameters as supplied with the software, with the exception that where possible we allowed the software to define its own membership functions for numerical attributes. The reported number of rules was calculated by counting the number of leaf nodes in the tree. For the data sets Credit-A and Hepatitis the software failed to return an answer for three of the ten folds, and for each the results are the mean and standard deviation of the remaining seven folds. The results for FBS were obtained from our own implementation of the algorithm as specified in reference [Fertig et al., 1999], where we set the maximum search depth to 15. We performed experiments with beam widths 10, 20, and 50 for each data set, and report the results for the beam width that resulted in the best test set classification accuracy. By definition, the algorithm returns a rule set that contains a *beamwidth* number of rules, and thus the standard deviation of the number of rules is not reported for FBS. FCF's results were obtained using FEM with a beam width of one for all data sets. We set $\theta_p = 1$ for Lymph and Labor, and $\theta_p = 2$ for the remaining data sets. The accuracy evaluation function was used for all data sets, except for Iris, where the Laplace function was used. The value for α_c was set to 0.5 for all data sets, and the value for α_a was set to 0.5, except for Iris (0.2), and for Hepatitis and Colic (both 0.8). The same α values were used for FCF and FBS. The experiments were also performed using the same instances for the different folds between methods.

Table 11.3 shows that in general FCF outperforms the other fuzzy learning methods with respect to classification accuracy. FID outperforms FCF by 0.3% and 0.9% on the colic and Hepatitis data sets, respectively, but FCF outperforms FID on all the remaining data sets, for example by 7.4% on the Lymph data and 2.9% on the Credit-A data. FBS often failed to return good rule sets, and FCF outperforms FBS by 15.7% and 20.3% on the Credit-A and Lymph data sets, respectively. FBS obtained the best classification accuracy for the Hepatitis data, outperforming FID by 0.4%, and tied for the best performance with FCF on the Labor data. Overall, FCF obtained the best classification accuracy results for five of the seven data sets. Averaged over all the data sets, FCF, FID, and FBS obtained classification accuracies of 85.3%, 82.7%, and 77.1%, respectively.

[3]We used FID3.3 obtainable at http://www.cs.umsl.edu/∼/janikow/fid/fid32/

Table 11.4: Rule set induced by FCF for a single fold of Diabetes. The operator ! means the disjunction of all terms in the respective term set, except for the indicated term (see Section 4.3.1). The default rule fires only if no other rule fires, and classifies instances as negative.

IF [!plas.low][!mass.low][!age.young] THEN positive
IF [plas.high, plas.$\bar{\alpha}$][!insu.med][pedi.high, pedi.$\bar{\alpha}$] THEN positive
IF [!preg.many][plas.high, plas.$\bar{\alpha}$][!insu.high][!pedi.low][!age.middleage] THEN positive

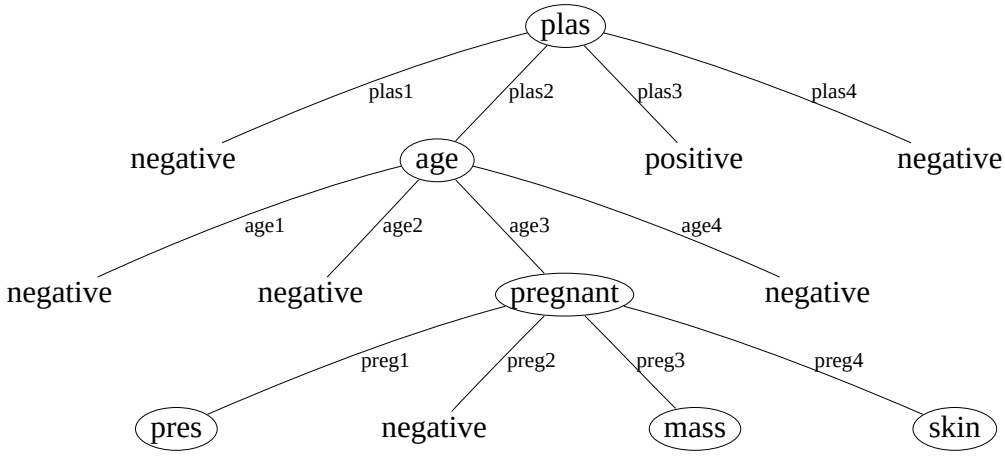


Figure 11.4: Fuzzy decision tree induced by FID. The lower part of the tree (13 more nodes) is not shown.

Overfitting occurs when classification performance on training data improves while deteriorating on an independent test set [Mitchell, 1997]. FCF's inductive bias prefers more general over more specific hypotheses, as explained in Section 4.8. The intention of its pre-pruning criteria is to prevent too specific rules and overfitting in general, while its user-defined parameters (especially θ_p) can be further adjusted to cater to the characteristics of a particular data set and in this way inhibit overfitting (or fitting of noise) and the corresponding inferior performance on an independent test set. Table 11.4 shows the rule set induced by FCF for a single fold of the Diabetes data set, with parameters $\alpha_a = 0.5$, $\alpha_c = 0.5$, $\theta_p = 5$, $beamwidth = 1$, and using the accuracy evaluation function (Eq (6.7)). Figure 11.4 shows the decision tree induced by FID for the same fold. We used the same parameters for FID as were used to obtain the results in Table 11.3. For this single fold the classification accuracies for FCF and FID on the same test set were 80.4% and 76.4%, respectively. (Recall that the classifiers used the identical independent training set too.) The size of the induced trees on all five folds (measured by the number of leaf nodes) varied considerably between folds, and very small or very large trees performed much worse than trees closer to the average size. The first rule in Table 11.4 was induced in each fold, and for one of the folds this rule comprised *the entire* rule set. FCF prevents the learning of rules with too low coverage of positive instances, as controlled by the θ_p parameter. Rules that cover very few positive instances often cover some or even comparatively many negatives, and could be fitting noise in the data. For the given

fold's training set, 68 rules, covering 90 positive and 340 negative instances in total, were not added to the rule set. Most classifiers benefit from post-pruning of their classification rules, but this was not done for the classifiers and rule sets induced here. With these statements we do not say that FCF's rules do not need post-pruning at all, but simply that its unpruned rules avoid overfitting to a large extent already.

A comparison of the rule set sizes obtained by the different methods demonstrates that FCF's induction strategy produces shorter and more comprehensible rule sets. FCF obtained the smallest rule sets on *all* data sets. For the Credit-A data FCF induced 7.1 rules versus 69 and 50 rules by FID and FBS, respectively, and for the colic data, FCF, FID, and FBS induced 5.6, 24.3, and 50 rules on average. Averaged over all the data sets, FCF, FID, and FBS induced rule sets of sizes 6.3, 22.0, and 38.6, respectively. Thus, the combination of FCF's fuzzy set covering methodology, fuzzy evaluation function, and pruning criteria performed better than the other related fuzzy rule learners with respect to classification accuracy, and performed much better with respect to rule set size. The significantly smaller sizes of the fuzzy rule sets induced by FCF enhance their comprehensibility, and FCF does this while even increasing classification accuracy. The results empirically demonstrate the validity of set covering as a new methodology for learning fuzzy classification rules.

11.5 Application of FCF to Two Relevant Real World Problems

11.5.1 SPAM Classification

The Ling-Spam Corpus

With the increase use of email above regular mail, the opportunity of advertisement via email have increased dramatically. When such mail advertisement is unsolicited, it is commonly referred to as SPAM. Since the cost of advertisement through media such as television, newspaper or magazines is much more expensive, the popularity of SPAM increased dramatically in recent times. In fact, the number of SPAM emails is starting to overwhelm the number of legitimate emails—to such a degree that it is feared that SPAM may cause the demise of the use of email, as users find it too cumbersome to sort out legitimate messages from SPAM. Most users find SPAM at least annoying, if not blatantly offensive, especially since a large proportion SPAM contains (graphic) advertisement for pornographic sites.

Different strategies to combat this threat exist. On the one hand there is legislature, e.g. the "Controlling the Assault of Non-Solicited Pornography and Marketing Act of 2003," as passed on December 16, 2003, by the United States Congress. In some cases the law can be applied effectively, e.g. an Internet service provider, CIS Internet Servers, won a lawsuit against SPAM senders who were sending up to 10 million SPAM messages per day to their server. The law dictates that SPAM senders be fined $10 per message, and the total damages amounted to one billion dollars [CNET News.com, 20 December 2004].

However, with the email protocols in use today the enforcement of such laws is often undermined, as it is difficult or often impossible to identify where the SPAM was sent from. Thus, another ap-

Table 11.5: Performance comparison of different classifiers on the LingSpam corpus.

Classification Method	Classification Accuracy	SPAM Recall	SPAM Precision
FCF	98.17	92.95	95.42
Naive Bayesian	96.93	82.35	99.02
TiMBL(1)	96.89	85.27	95.92
TiMBL(2)	96.75	83.19	97.10
Outlook Patterns	90.98	53.01	87.93
TiMBL(10)	89.08	34.54	99.64
No Filter	83.37	0	∞

proach is the proposal of new email standards that removes the anonymity of email. A third approach is identifying SPAM and automatically deleting it. One common approach to distinguish between different classes of text documents is the use of Bayesian classifiers such as Naive Bayesian [Mitchell, 1997]. This method also proved relatively successful to separate SPAM from HAM (a term for legitimate email) [Sahami et al., 1998; Schneider, 2003], outperforming advanced rule learners such RIPPER [Pantel and Lin, 1998].

The Ling-Spam corpus is a publicly available corpus of SPAM and legitimate messages[4]. The corpus contains 2893 messages sent via the mailing list Linguist. Linguist is a moderated mailing about the science of linguistics[5]. Approximately 16% of the messages in the corpus is SPAM, and the labelling was done by hand to minimize noise. Although the corpus covers mainly the domain of linguistics, legitimate messages also include, for example, job postings and software announcements.

Experiments

To induce a fuzzy rule set capable of distinguishing between SPAM and HAM, the different text documents are first preprocessed into feature vectors. We used the freely available software FeatureFinder[6] for feature extraction. FeatureFinder uses mutual information to select a user-defined number of features. The feature types that can be created include TF (term frequency) and TF-IDF (Term Frequency / Inverse Document Frequency). Let the i^{th} feature have the i^{th} greatest mutual information, let TF_i be the number of occurrences of the i^{th} feature in a given document, and let $|D|$ be the total number of documents, then f_i, the TF-IDF of the i^{th} feature for a given document is calculated as,

$$f_i = \log \frac{|D|}{TF_i} \tag{11.2}$$

To create a fuzzy training set for our experiments we first extracted 500 TF-IDF type features for each document. We then extracted membership functions from each feature using the approach described in Appendix C, where we allowed up to four linguistic terms per linguistic variable. However, in general the extraction process suggested the use of three membership functions.

Sakkis *et al* compared the performance of an adapted k-nearest neighbour classifier called TiMBL

[4]The Ling-Spam corpus can be downloaded at http://www.dcs.ex.ac.uk/corpora/

[5]An archive of the Linguist mailing list is available at http://listserv.linguistlist.org/archives/linguist.html

[6]Retrieved from http://www.cs.iastate.edu/∼andymenz/573Project.html on 1 December 2004

[Daelemans et al., 2000] for values of $k = 1, 2, 10$ with that of Naive Bayesian and MicroSoft Outlook patterns on the Ling-Spam corpus [Sakkis et al., 2003; Androutsopoulos et al., 2000]. We repeat their results along with that obtained using FCF in Table 11.5. FCF was configured as follows, $\alpha_c = 0.5$, $\alpha_a = 0.2$, $beamwidth = 1$, $\theta_p = 0$, simultaneous concept learning, Laplace evaluation, and FUZZ-CONRI as specialization model. SPAM recall is a measurement of the percentage of SPAM documents correctly identified with respect to all SPAM documents, and is thus equivalent to the True Positive measurement. SPAM precision is a measurement of the accuracy of a prediction, and is computed by the percentage of correctly identified SPAM documents with respect to all documents identified as SPAM, and is thus equivalent to $(1 - \text{False Postive Ratio}) \times 100\%$. Classification accuracy measures the number of correctly classified documents, where the classification is either SPAM or HAM. The No-Filter classifier classifies all documents as legitimate, and accordingly has zero recall. It's classification accuracy is 83.19%.

FCF outperformed all the other methods with respect to classification accuracy. It obtained a classification accuracy of 98.17%, while the second best classifier, Naive Bayesian, obtained a classification accuracy of 96.93%. FCF also significantly outperformed all other classifiers on SPAM recall. It obtained 92.95% recall, while the three next best classifiers, TiMBL(1), TiMBL(2), and Naive Bayesian, obtained recall percentages 85.27%, 83.19, and 82.35%, respectively. FCF was thus much more successful at identifying SPAM messages than any of the other learners. FCF's SPAM precision was slightly worse, but still comparable to that of the other classifiers. In the order of best recall, the precision of FCF, TiMBL(1), TiMBL(2), and Naive Bayesian, were 95.42%, 95.92%, 97.10%, and 99.02%. We provide the rule set induced by FCF during one fold of the 10-fold cross validation in Appendix E.

11.5.2 Septic Shock

Septic Shock is defined as "*...a serious, abnormal condition that occurs when an overwhelming infection leads to low blood pressure and low blood flow. Vital organs, such as the brain, heart, kidneys, and liver may not function properly or may fail. Decreased urine output from kidney failure may be one symptom.*" [MedlinePlus Medical Encyclopedia, 2004]. Septic shock is associated with a mortality rate of around 50%, and is still an important research subject for medical experts and data analysts [Paetz, 2003]. During the Deutsche Forschungsgemeinschaft (DFG) sponsored project MEDAN[7], medical experts and data analysts cooperated to gather data of septic shock patients. The H16 data set contains the sixteen most measured physiological parameters of 138 septic shock patients. Of the 138 patients, 68 patients survived.

Paetz [2002] used a Fuzzy Rectangular Basis Function Dynamic Decay Adjustment Neural Network (Fuzzy-RecBF-DDA-NN) [Berthold and Huber, 1995; Huber and Berthold, 1995] to learn rules for predicting whether a patient will survive or not. The results reported were obtained from 5-fold cross validation. The rule sets attained classification accuracy with mean and standard deviation 84.02% and 4.44%, respectively. The average rule set size was 16 rules. We obtained the same test and training sets

[7] See http://www.medan.de/

Table 11.6: Performance of different configurations of FCF on the MEDAN data.

Config	Accuracy		Rule Set Size	
	Mean	Std Dev	Mean	Std Dev
RecBF	84.02	4.44	16.0	N/A
HI	87.93	3.70	40.8	7.00
GO	87.00	3.49	26.2	3.03
SR	84.48	1.66	3.2	0.84

Table 11.7: Different configurations of FCF on use with the MEDAN data.

Config	Beam Width	Spec. Mod.	Inf. Thres.	Weighted	SCL	Eval. Meth.
HI	1	FuzzConRI	0.1	T	F	lscontent
GO	4	FuzzConRI	0.1	F	F	lscontent
SR	3	FuzzyBexa	0.4	F	T	accuracy

as used in the experiments from the author of reference [Paetz, 2002] for direct comparison with FCF. Table 11.6 reports the results for the three different configurations of FCF shown in Table 11.7, as well as the results from reference [Paetz, 2002].

The goal of configurations HI and GO was high classification accuracy. The best classification accuracy obtained by FCF was 87.93%, which is 3.91% better than the previous result obtained by the NN (neural network). The weighted cover resulted in the induction of many overlapping rules, and the resulting rule set has relatively many rules (40.8). If no weighted cover is used, as in configuration GO, a good overall result is obtained, with 87.0% classification accuracy and 26.2 rules on average. For configuration SR we used the simultaneous concept learning induction strategy. The rule set classification accuracy for these rules were only slightly better than that of the NN, however, the rule set sizes were dramatically smaller—on average 3.2 rules per rule set. An example of one such rule set is the following:

IF [!*BlutdruckSystolisch.mf0*][!*Temperatur.mf4*][!*Thrombozyten.mf0*][!*Urinmenge.mf0*]
THEN *class.ueberlebt*
ELSE IF [!*BlutdruckDiastolisch.mf4*] THEN *class.verstorben*
ELSE *class.ueberlebt*

FCF was thus able to improve significantly on the previous results, both with respect to rule set comprehensibility and rule set classification accuracy[8].

11.6 Summary

In this chapter we provided arguments why set covering is a good methodology for the induction of fuzzy rules. Crisp covering algorithms are a special case of fuzzy covering algorithms, and as such fuzzy covering algorithms are at least as powerful as crisp covering algorithms. We also provided theoretical

[8] In English, "BlutdruckSystolisch" means systolic blood pressure, "Temperature" means temperature, "Thrombozyten" means platelets, "Urinmenge" means urine quantity, "ueberlebt" means survived, "BlutdruckDiastolisch" means diastolic blood pressure, and "verstorben" means deceased.

arguments why fuzzy covering algorithms are more powerful, for example that, unlike for crisp rules, the decision boundary in the fuzzy case need not be axis-parallel. We also proposed a series of experiments to demonstrate cases where crisp rule induction fail, but fuzzy rules provide good results. We provided an empirical evaluation of different fuzzy methods on benchmark data sets to substantiate our claim that fuzzy set covering often perform better than other fuzzy learning methods, such as fuzzy decision trees or beam search, for example. FCF was able to convincingly outperform the other fuzzy classifiers with respect to classification performance. In addition, FCF obtained significantly less complex rule sets. Finally, we provided two applications where FCF improved upon the performance of previously used methods.

CHAPTER 12

Conclusions and Directions for Future Research

The objective of this dissertation was to prove that set covering can be applied successfully for the induction of fuzzy classification rules from training data. Set covering has proven to be a very successful concept learning methodology in the crisp case, and many different algorithms applying this approach have been proposed. Fuzzy sets are a generalization of crisp sets, and a crisp set is a special case of a fuzzy set. As such, many different methods for the induction of fuzzy rules have been proposed. Some of the more successful induction methodologies are fuzzy decision trees, genetic algorithms, and partitioning methods. One drawback of most of these methods is that the induced rule sets are often not very comprehensible due to their rather large number of rules. There are also comparatively few methods that allow both the induction of incomplete rules. Furthermore, most methods concentrate on extracting fuzzy set membership functions, and thus forgo the use of fuzzy sets as linguistic labels with meaning to domain experts. However, according to Guillaume [2001], the use of linguistic terms, small rule sets, and the induction of incomplete rules are exactly the criteria for obtaining comprehensible fuzzy rule sets.

By developing the fuzzy set covering rule induction methodology, this dissertation addressed the problem of inducing accurate, but also comprehensible fuzzy classification rules. Thus, we have extended the different classes of rule induction methods, and added fuzzy set covering to it. We have also developed four new fuzzy rule induction algorithms implementing this new methodology. The first algorithm, FUZZYBEXA, inherits its structure from its crisp ancestor BEXA. FUZZYBEXA induces a single rule through a conjunction specialization process based on excluding linguistic terms. It starts with the most general conjunction in its description language, and expand this allowing a local beam search until certain stopping criteria are met. We have also proved various characteristics of the algorithm, for example that its description language induces a lattice, and that the fuzzy extension operator is an order-preserving mapping from descriptions to associated instance sets.

We also presented several experiments with FUZZYBEXA. An experimental evaluation with benchmark data sets investigated its different learning parameters. We measured the effect of the beam width, FUZZYBEXA's sensitivity to noise, it's pre- and post-training sensitivity to the value of the α-cut, and the effect of its various stop growth tests. The principle results are that FUZZYBEXA's search effort

grows at most linearly with increasing beam width, that it behaves well in the presence of noise, that it is not overly sensitive to the antecedent threshold, and that the use of the stop growth criteria significantly improves the search in terms of rule set complexity and search effort. The experiments also show that although FUZZYBEXA's hypothesis space for most problems can be very large, the algorithm easily copes with normal size data sets, and that even very large data sets can be successfully searched.

The conjunction evaluation measure plays an important role to guide the search for single rules. As such, we proposed a range of conjunction evaluation functions specially adapted to the fuzzy case. We also conducted experiments to investigate their performance for different data sets. The results showed that the evaluation function should be matched to the data set's characteristics, and that no single evaluation function always performs best. However, our proposed Accuracy evaluation function performed very well in most circumstances, especially as measured by the size of the rule set.

We also presented a survey of different algorithms for the induction of fuzzy rules. These algorithms can be grouped into seven classes, depending on their induction strategy: greedy incremental rule learners, divide-and-conquer, similarity, stochastic, partitioning, hierarchical, and gradient descent. Of course there also some algorithms that do not fit neatly into one of these classes. We provided a comparison between the different classes and FUZZYBEXA, as an example of a fuzzy set covering algorithm. None of the algorithms have all of FUZZYBEXA's characteristics, in fact, most have very little in common with FUZZYBEXA.

Since one new algorithm is not enough to establish a paradigm, we developed more fuzzy algorithms applying the set covering approach, FUZZYSEEDSEARCH, FUZZCONRI, and FUZZYPRISM. FUZZCONRI and FUZZYPRISM use FuzzyCAL as description language, and employ *append* as specialization operator. FUZZYBEXA and FUZZYSEEDSEARCH use FuzzyAL as description language, and employ *exclude* as specialization operator.

FCF was introduced as a general framework for set covering algorithms, both crisp and fuzzy. The top layers of the framework encapsulate everything that is similar between different set covering algorithms. This include the fuzzy set covering approach, and search heuristics such as conjunction evaluation, beam search, prepruning, and efficiency improvements. Any improvement to the top layers, or the addition of new or more advanced heuristics, will automatically benefit all algorithms that fit in the framework. FCF allows different covering algorithms to be characterized and compared. Thus, FCF allows the rapid development of new covering algorithms, since the designer need to concentrate only on what differentiates his algorithm from the rest. To demonstrate the applicability of the framework we showed that all four proposed covering algorithms fit within the framework. We also characterised each algorithm and described its various properties.

To the best of our knowledge, there existed no algorithm for the induction *ordered* fuzzy rule sets, or fuzzy decision lists. FUZZYBEXAII is a novel fuzzy rule induction algorithm following the simultaneous concept learning approach, and is capable of inducing decision lists (ordered rule sets). We showed that decision lists can compare favourably to unordered rule sets under the right conditions. If an appropriate conjunction evaluation function is used, the induced rule set can be very descriptive and highly accurate, while being extremely compact.

To motivate the use of fuzzy set covering we have provided arguments for the use of fuzzy set covering as opposed to crisp set covering of other fuzzy rule learning methods. Fuzzy set covering as a generalization of crisp set covering is far more powerful, and includes crisp set covering as a special case. We have also compared fuzzy set covering to other algorithms that are also capable of inducing incomplete rules and use fuzzy sets as linguistic labels. On average, FCF outperforms methods such as decision trees (e.g. FID) or beam search (e.g. FBS) in terms of classification accuracy. However, at the same time FCF significantly outperforms these methods in terms of rule set comprehensibility. Finally, we provided results on two real world applications where FCF improved upon the state of the art. In the next section we list the major scientific contributions made by this dissertation. We then provide some directions for future research in Section 12.2, and Section 12.3 concludes the dissertation.

12.1 Scientific Contributions

We list the major scientific contributions made by this dissertation:

1. Establishing a new paradigm for the induction of fuzzy classification rules
   ("Fuzzy rule induction in a set covering framework", [Cloete and van Zyl, 2006]);

2. Narrowing the gap between the symbolic and sub-symbolic machine learning communities
   ("A machine learning framework for fuzzy set covering algorithms", [Cloete and van Zyl, 2004c]);

3. The first ever algorithm for the induction of fuzzy decision lists
   ("Simultaneous concept learning of fuzzy rules", [van Zyl and Cloete, 2004f]);

4. A general fuzzy set covering framework
   ("Specialization models for a general fuzzy set covering framework", [van Zyl and Cloete, 2006]);

5. Novel fuzzy rule evaluation functions, and their importance during rule induction
   ("Heuristic functions for learning fuzzy conjunctive rules", [van Zyl and Cloete, 2004c], "Evaluation function guided search for fuzzy set covering", [Cloete and van Zyl, 2004a]);

6. The algorithm FUZZYBEXA based on exclusion
   ("Fuzzy set covering with FuzzyBexa", [Cloete and van Zyl, 2004b]);

7. The algorithm FUZZCONRI that induce rules in FuzzyCAL
   ("An inductive algorithm for learning conjunctive fuzzy rules", [van Zyl and Cloete, 2004d], "FuzzConRI - a fuzzy conjunctive rule inducer", [van Zyl and Cloete, 2004a]);

8. The algorithm FUZZYPRISM that uses fuzzy information gain
   ("FuzzyPRISM: a specialization model for the FuzzyBexa framework", [van Zyl and Cloete, 2004b]);

9. Encoding FuzzyAL rules as prior knowledge in a neural network
   ("Prior knowledge for fuzzy knowledge-based artificial neural networks from fuzzy set covering", [van Zyl and Cloete, 2004e]).

12.2 Directions for Future Research

We have proposed fuzzy set covering as a new paradigm for fuzzy classification rule learning. However, the field is still wide open for further research. Some ideas include further extensions and improvements to FCF, the development of more algorithms, and the extension of fuzzy set covering in general. In the remainder of the chapter we give a brief overview of some of the open problems and also propose some possible strategies.

12.2.1 Neural Network Encoding of Fuzzy Rules

The encoding of an extracted fuzzy rule set in a neural network will provide a link between the symbolic and sub-symbolic connectionist approaches to concept learning. In this area we have already taken preliminary steps, and developed a method for encoding FuzzyAL rules [van Zyl and Cloete, 2004e]. The network can represent a fuzzy rule set with internal disjunction accurately under the right conditions. The knowledge encoding strength (bias) should be large enough, and the slope parameter of the sigmoidal activation functions should also be sufficiently large. We also showed empirically that the network is capable of correcting incorrectly encoded knowledge, and of improving given further training data. However, the final step of taking the trained neural network and again extracting FuzzyAL rules has still not been taken. Rule extraction would allow the seamless migration between both knowledge representations. Since the encoding method is related, but not central to the theme of this dissertation, we provide a summary of the method in Appendix D.

12.2.2 Extending the Description Language

FuzzyAL and FuzzyCAL are both powerful description languages - as can be seen from the good performance of algorithms using them. However, as discussed in Section 4.10.4, these description languages do not allow for the description of relations between different attributes. One possible way for extending the description language is to add more operators, such as relational operators. A further extension is the addition of fuzzy hedges, such as “very,” “little,” “at most,” etc.

12.2.3 Predicting Concept Membership

We discussed the semantic interpretation of the rules induced by FCF in Section 4.3.5. The membership of an instance to a rule antecedent is no prediction of the membership of the instance to the rule consequent. In some cases it may be desirable to know the membership to the concept. One approach may be to learn a non-linear mapping between instances’ membership to a rule’s antecedent and its consequent for all instances matched by the rule. Another approach may to adapt the induction process to learn a hierarchy of rules, such that rules on higher levels have higher membership strengths.

12.2.4 Rule Post-Pruning

FCF includes many efficiency criteria, and also includes the prepruning of rules. However, in this dissertation we did not address the question of rule post-pruning—i.e. pruning after the induction of the complete rule set. In the crisp case, rule post-pruning often increases the generalization performance of the rule set [Mitchell, 1997, p. 71]. We have already undertaken some preliminary steps to address rule post-pruning [Robbel, van Zyl and Cloete, 2004], but much more remains to be done.

12.2.5 Computing the Complete Most General Consistent Rule Set

FUZZYBEXA searches the lattice of conjunctions from top to bottom in a consistent manner, and it is guaranteed to find members of C_M, the set of most general consistent conjunctions, during each iteration when using an infinite beam width. In fact, using an infinite beam width, FUZZYBEXA will find *all* members of C_M during the *first* iteration of *FindBestConjunction*. However, presently *FindBestConjunction* returns only a single conjunction. A further possible extension to FCF is to keep track of the set of "best conjunctions." This can be implemented by maintaining the set *best_conjunctions* in *FindBestConjunction*. This set is cleared each time the best conjunction is replaced by a conjunction that has a better evaluation. Each time a conjunction is found with the same evaluation as the best conjunction, this conjunction is added to *best_conjunctions*. The set of best conjunctions is then returned. To prevent the addition of many similar rules, *CoverConcepts* could only add rules from *best_conjunctions* that have no instances in common with other rules from *best_conjunctions*. Using this method, the set of all disjoint but equally good rules is found during each iteration of *FindBestConjunction*, which could be renamed to *FindBestConjunctions*. A larger beam width may prove helpful in this case.

12.2.6 Automatic Selection of α_a

FCF requires the antecedent threshold α_a to be specified by the user. Often the user (domain expert) may have a good feeling for a suitable value of α_a, but this may also not be the case. Another extension to the framework is thus to allow the framework to select α_a automatically, and even select different values of α_a for different rules. One concern, however, is that too many individually tuned values for α_a may reduce the comprehensibility of the rule set.

12.2.7 Evaluation Function Sensitivity to α_a

We have not investigated the sensitivity of each rule evaluation method to α_a. This may be an interesting experiment, and we expect different evaluation functions to have different levels of sensitivity to α_a. We expect the Laplace function to be very sensitive, but the Accuracy function to be relatively insensitive to α_a. Depending on the problem domain, one may opt to use a more insensitive function if α_a cannot be determined externally.

12.2.8 Using Genetic Algorithms for Adapting Membership Functions

We have spent very little time exploring the influence of different membership functions on the induction process. The rationale was that the membership functions are determined externally to the induction process, and the induction algorithm should make do with what it has. However, the induction process is certainly influenced by the membership functions, and better membership functions should allow the induction of more accurate and also more comprehensible rule sets. FCF would allow the genetic optimisation of membership functions, by providing an objective function in the form of a rule set. The process may functions roughly as follows. FCF is used to bootstrap the process by the induction of a rule set. The rule set is then used as objective function for membership function optimisation. After optimisation, the rule set is discarded, but the membership functions are kept for the next iteration of rule induction. The process can then be iterated until the classification performance of successive iterations do not improve anymore.

12.2.9 Incremental Learning and Prior Knowledge

It may be desirable to keep an old tried-and-tested rule set even when new information (training data) becomes available. In this case an incremental learning approach exploiting the prior information may be used. Prior information may also be presented in the form of knowledge extracted from domain experts. A first approach is to add the prior knowledge in the form of rules to the rule set prior to rule induction, and to continue rule induction as usual. Rule antecedents may also be pruned using the extra training data. Another approach may be to adapt rules that classify the new data incorrectly either by specializing or generalizing them.

12.2.10 Information from Knowledge Discovery

The last aspect which we address is the application of FCF to real world domains. FCF presents a new methodology for knowledge discovery which may prove very useful in many different domains. We have showed some preliminary results for two such applications, and FCF performed very well. However, we did not customize or adapt the data in any way. We expect a custom solution involving FCF and data adapted to fit the algorithm to yield very satisfactory results.

12.3 Conclusion

This dissertation advanced the state of the art in fuzzy classification rule induction by establishing fuzzy set covering as a new fuzzy rule induction paradigm. Fuzzy set covering algorithms are capable of inducing very comprehensible but also highly accurate rule sets. Thus, we hope the work presented in this dissertation make the use of fuzzy classification rules more acceptable to both the crisp rule set and numerical concept learning communities.

APPENDIX A

Classical Set Covering Algorithms

In this dissertation we established set covering as a methodology for rule induction in the fuzzy case. Thus, it is appropriate to provide a brief review of classical (crisp) set covering algorithms. For a definition of set covering please see Section 3.2. Section A.1 reviews the AQR family of algorithms, Section A.2 reviews PRISM, Section A.3 reviews CN2, and Section A.4 reviews RIPPER.

A.1 AQR

The AQR family of inductive learning algorithms, of which AQ15 is an example, generates rules from training instances by following the principles first introduced by Michalski in 1969 [Michalski et al., 1986b; Michalski, 1969]. AQR builds decision rules that accounts for all positive and no negative instances by following a heuristic search of the a space of legal logical expressions. AQR rules are represented in VL_1, which is a multiple-valued logic propositional calculus with typed variables [Michalski, 1974a].

As an example of AQR, Table A.1 shows the basic AQ15 algorithm [Michalski et al., 1986a,b]. The algorithm is initialized with a partial cover of the positive examples. This initial partial cover may simply have the value true, or be a user defined hypothesis, providing AQ15 with an incremental learning facility. The procedure $getStar$ obtains all maximally general complexes, or hypotheses, that cover a positive seed and not a negative seed. These are obtained by generating all maximally general complexes covering the positive seed, and removing those that also cover the negative seed. The maximally general complexes are then intersected with the current partial cover. This results in a new partial cover that still covers the positive seed, while not covering the negative seed.

This process is iterated and the results combined until no negative examples are covered. The best complex from the result of $getStar$ is then added to the current rule set. AQ15 iterates the whole process until all positive examples are covered. The most recent incarnation of the AQ algorithm is AQ20 [Cervone et al., 2001]. Important new features include an object oriented implementation, handling continuous variables without prior discretization, and selecting multiple rules from $star$.

Table A.1: The AQ15 algorithm.

```
    PROCEDURE AQ15(partialcover)
1     WHILE partialcover does not cover all positive examples
2       seed = any uncovered positive example
3       star = getStar(seed)
4       best = best complex from star according to evaluation function
5       partialcover = patialcover ∨ best
6     RETURN partialcover
    END PROCEDURE


    PROCEDURE getStar(positiveseed)
1     partialstar = TRUE
2     WHILE partialstar covers some negative examples
3       negativeseed = any negative example covered by partialstar
4       negativestar = {c|c is maximally general, c covers positiveseed,
        and c does not cover negativeseed}
5       partialstar = patialstar ∩ negativestar
6       retain maxstar best disjoint complexes in partialstar
7     RETURN partialstar
    END PROCEDURE
```

A.2 PRISM

PRISM is an induction algorithm that borrows some ideas from ID3 to implement an inductive rule learner [Cendrowska, 1987]. Rules are iteratively induced for each class following the algorithm shown in Table A.2. In the first step the probability of occurrence $p(\delta_n|a_x)$ of the classification δ_n for each attribute value pair a_x is calculated. In the second step the pair for which $p(\delta_n|a_x)$ is maximum is selected, and a subset of the training set comprising all the instances which contain the selected a_x is created. This subset is then considered as the new training set, and the previous two steps are repeated until all instances in the training set belong to class δ_n. An IF-THEN rule is then formed by taking the conjunction of all a_x chosen as antecedent and δ_n as consequent. The original training set is then restored, but all instances covered by the new rule are removed. This procedure is iterated until all instances of class δ_n are covered. At this point the initial training set is restored and the induction of rules covering the next class begins. The induced rule set is clearly unordered. If two attribute values have the same class probability, PRISM selects the attribute value that has the highest probability that the class occurs within the subset considered, thereby opting to induce the most general rule first.

A.3 CN2

The CN2 induction algorithm is based partially on ID3 [Quinlan, 1986] and partially on the AQR family described in Section A.1. CN2 removes the need for seed examples during the search process, and employs a beam search and stop growth tests [Clark and Niblett, 1989]. The CN2 algorithm learns an ordered list of rules, with a default rule predicting the most frequently occurring class as the last rule.

Table A.2: The PRISM algorithm.

PROCEDURE PRISM(T)
1 Calculate the probability of occurrence $p(\delta_n|a_x)$ of classification δ_n for each attribute-value pair a_x.
2 Select the a_x such that $p(\delta_n|a_x)$ is a maximum, and create a subset F, of the training set T such that $F = \{e|e$ contains a_x and e belongs to $\delta_n\}$.
3 Repeat steps 1 and 2 until $\forall e \in F(e$ belongs to $\delta_n)$. Form the rule r with δ_n as consequent and as antecedent $a_1 \wedge a_2 \wedge \ldots \wedge a_n$, where the a_i was chosen in step 2.
4 Remove all instances covered by the rule from the training set.
5 Repeat steps 1 to 4 until all instances of class δ_n have been removed.
END PROCEDURE

To classify an unlabeled instance, rules are considered in order until the first one fires. This rule is then used to predict the class. Ordered rule learning uses entropy as an example evaluation function. The relationship between ID3, CN2 and the AQ family is well studied in the literature [Clark and Niblett, 1989; Theron and Cloete, 1996].

The CN2 algorithm is given in Table A.3. The algorithm receives a training set E of instances. It then iteratively searches for a complex (description) covering a large number of instances from class C and while covering few instances of other classes. If a rule is found, the examples covered by it are removed from the training set and the rule is added to the rule list. This process repeats until the training set becomes empty, or no new suitable rules are found.

New complexes are generated in a pruned general to specific search. A size-limited set of the best complexes found thus far is maintained. Complexes from this set are specialized by adding new conjunctive terms. Specializations that cover no instances, or that was generated in the previous iteration, are removed. For its rule evaluation function CN2 uses the entropy measure

$$H = -\sum_i p_i \log_2(p_i) \tag{A.1}$$

where p_i is the probability that an instance covered by the rule belongs to class i. CN2 prunes rules by testing whether they are significant. The likelihood ratio statistic is used to compare the observed distribution to the expected distribution of the training set. Only significant rules are added to the rule set. In an improved version of CN2, the induction of an unordered rule set was proposed by using the Laplace estimate to evaluate conjunctions [Clark and Boswell, 1991].

A.4 RIPPER

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [Cohen, 1995] is an improved version of the algorithm IREP (Incremental Reduced Error Pruning) by Fürnkranz and Widmer [1994], and is arguably the most powerful rule learning algorithm today [Fürnkranz and Flach, 2004]. The basic algorithm is given in Table A.4. *GrowRule* starts with an empty conjunction of conditions, and iteratively appends conditions that maximizes FOIL's information gain criterion [Quinlan, 1990;

Table A.3: The CN2 algorithm.

```
    PROCEDURE CN2(E)
1     rulelist = ∅
2     REPEAT
3       bestcomplex = findBestComplex(E)
4       IF bestcomplex ≠ NULL
5         E′ = examples covered by bestcomplex
6         E = E − E′
7         C = most common class in E′
8         rulelist = rulelist ∪ 'IF bestcomplex THEN C'
9       UNTIL bestcomplex = NULL OR E = ∅
10    RETURN rulelist
    END PROCEDURE


    PROCEDURE findBestComplex(E)
1     star = TRUE, bestcomplex = NULL, selectors = all possible selectors
2     WHILE star ≠ ∅
3       newstar = {x ∧ y|x ∈ star, y ∈ selectors}
4       newstar = newstar − star ∪ {c|c ∈ star, c = NULL}
5       FOR c_i ∈ newstar
6         IF c_i is statistically significant when tested on E AND
7           c_i is better than bestcomplex according to the evaluation function THEN
8             bestcomplex = c_i
9       retain beamwidth best complexes in newstar
10      star = newstar
11    RETURN bestcomplex
    END PROCEDURE
```

Quinlan and Cameron-Jones, 1993] until no negatives are covered. *PruneRule* considers deleting any final sequence of conditions from the rule, choosing the deletion that maximizes the function

$$v(rule, pruneP, pruneN) \equiv \frac{p-n}{p+n} \tag{A.2}$$

where P and N are the numbers of positive and negative instances in the sets $pruneP$ and $pruneN$, respectively, and p and n are the number of positive and negative instances covered by $rule$, respectively. The deletion process continues until no deletion improves the value v. For multi-class problems, RIPPER orders classes in sequence of increasing prevalence. Rules are then induced using the first class as positive and the remaining classes as negative instances. The instances of the first class are then removed from the training set, and the process iterated until only the majority class remains, which is classified by the default rule. RIPPER's stopping criterion works as follows. The total description length of rule set and instances are computed. Induction terminates when this description length is more than d bits larger than the smallest description length obtained thus far, or when no positive instances remain. For a discussion on computing the description length of a rule please see references [Cohen, 1995] and [Quinlan and Cameron-Jones, 1995a]. The rule set obtained from IREP* is further optimised by a post-pruning phase. The whole process can be iterated by adding additional rules induced by IREP* and optimising again. This algorithm is called RIPPERk for k optimisation steps. By design RIPPER induces extremely compact rule sets. An accuracy based comparison on 37 data sets between RIPPER2

Table A.4: The RIPPER algorithm.

```
    PROCEDURE RIPPER (T, Concepts)
1     order Concepts in order of increasing prevalence
2     ruleset = ∅
3     REPEAT k TIMES
4       FOR EACH concept c ∈ Concepts DO
5         Let (P, N) be the positive and negative instances in T
6         ruleset = ruleset ∪ IREP*(P, N)
7         remove instances belonging to concept c from T
8       END FOR
9       PostPrune(ruleset)
10    END REPEAT
11    RETURN ruleset
    END PROCEDURE

    PROCEDURE IREP* (P, N)
1     ruleset = ∅, minmdl = ∞
2     WHILE P ≠ ∅
3       split (P, N) into (GrowP, GrowN) and (PruneP, PruneN)
4       rule = GrowRule (GrowP, GrowN)
5       rule = PruneRule (rule, PruneP, PruneN)
6       IF minmdl > MDL(rule) THEN
7         minmdl =MDL(rule)
8       IF MDL(rule)> minmdl + d THEN
9         RETURN ruleset
10      ruleset = ruleset ∪ rule
11      remove instances covered by rule from training data
12    END WHILE
    END PROCEDURE
```

and C4.5 [Quinlan, 1993b] resulted in one draw, 21 wins and 15 losses for RIPPER2 [Cohen, 1995].

APPENDIX B

Fuzzy Attribute Relation File Format

The Fuzzy Attribute Relation File Format (FARFF) is an extension of the Attribute Relation File Format (ARFF) used by the WEKA data mining package [Witten and Frank, 2000]. ARFF was designed for crisp data sets, and does not allow the definition of membership functions. An example of an FARFF data file is shown in Table B.1. The first line defines the name of the relation using the keyword `@relation`. The definition of the linguistic variables is followed by the definition of the fuzzy instances and are separated by the keyword `@data`.

FARFF allows the definition of three different fuzzy attribute types for dealing with linguistic variables. The simplest type allows the definition of a linguistic variable with a single linguistic term. Such attributes are indicated by the keyword `@attribute`, followed by the name of the linguistic variable and then the keyword `@mfvalue`. The linguistic variable *skill* is an example of such a definition. The membership degree to the fuzzy set for each instance is given in the data section of the file. Each attribute definition is delimited by parentheses and separated by commas. The membership for instances 1, 2 and 3 to *skill* is 0.4, 0.6, and 0.9, respectively. Linguistic variables with fuzzy sets for which the instance membership degrees are known, but not the membership functions, are defined by the keyword `@attribute`, followed by the name of the linguistic variable, followed by the definition of its term set. The term set is delimited by curly braces, and the individual linguistic terms are comma separated. The instance membership degrees to the individual linguistic terms are given in the same order as the definition of the linguistic terms. For example the membership of instance 1 to the linguistic terms *Surrealism*, *AbstractExpressionism*, and *PopArt* are 0.4, 0.7, and 0.0, respectively. Note, crisp nominal observations are a special case of this type of attribute, like for example the linguistic variable *stillAlive*. In this case the membership degree to a single linguistic term is one while the membership degrees to remaining terms are zero.

FARFF also allows the definition of the membership functions of the individual linguistic terms. Such attributes are indicated by the keyword `@mf`, followed by the name of the linguistic variable and the definition of the membership functions of its term set, delimited by curly braces. Each linguistic term is defined by specifying its name, followed by a colon and the specification of the membership function. The membership function can be any piece-wise linear function, specified by giving the coordinates of the function. The membership degree specification of the lowest and highest points on the domain defines the membership degrees for points lower and higher than these, respectively. For example, the linguistic variable *age* has four linguistic terms, *child*, *young*, *midlife*, and *old*. Instances contain the

Table B.1: An example of a FARFF data file.

```
@relation FARFFExample
@attribute skill      mfvalue
@attribute stillAlive {yes, no}
@mf age {
       child:    (15 , 1.0) - (20 , 0.0)
       young:    (10 , 0.0) - (20 , 1.0) - (30, 1.0) - (35 , 0.0)
       midlife:  (30 , 0.0) - (35 , 1.0) - (50, 1.0) - (65 , 0.0)
       old:      (60 , 0.0) - (70 , 1.0)
}
@attribute style      {Surrealism, AbstractExpressionism, PopArt}

@data
(0.4), (1 0), (71.0), (0.4, 0.7, 0.0)
(0.6), (1 0), (55.0), (0.0, 0.2, 0.8)
(0.9), (0 1), (57.0), (0.9, 0.4, 0.0)
```

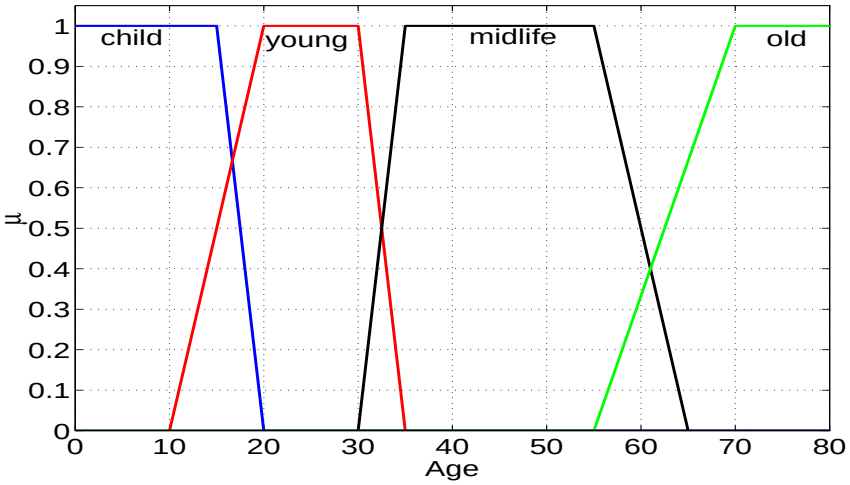


Figure B.1: Membership functions of the linguistic terms for the linguistic variable *age*.

observation on the domain of the linguistic variable, for example instances 1, 2, and 3 specify *age* as 71, 55, and 57, respectively. The membership of an instance with age 15 or less to the linguistic term *child* will be one, and similarly the membership of an instance with age 20 or more will be zero. The membership degree of an instance to all the specified linguistic terms can thus be computed from their respective definitions.

APPENDIX C

Membership Function Extraction

The fuzzy set covering algorithms introduced by this work assume that a fuzzy training set is available for the induction of fuzzy classification rules. Some approaches do not make this assumption, and assume numerical observations as input [Wang and Mendel, 1992; Kasabov, 2001b]. One of their main functions is thus also to induce a mapping from numerical observation to membership degrees. Other methods, most notable fuzzy decision tree induction methods [Yuan and Shaw, 1995; Cios and Sztandera, 1992], assume instances are defined by their membership degrees to linguistic terms. FCF allows both approaches, either the membership degrees, or the membership function and numerical observations can be specified. Different methods for automatically determining membership functions have been proposed in the literature. These include using self-organizing maps, clustering methods, neural networks, and genetic algorithms (for an overview refer to Chapter 2). In the following discussion we will introduce the method used to extract membership functions as used for all the experiments in this work.

C.1 Fuzzifying Training Data

If the training data is specified by the membership degrees of instances to linguistic terms, no further fuzzification is required. However, it is more common that the data contain numerical observations of system variables. In this case a membership function mapping the numerical domain to fuzzy membership degrees is used. The definition of such membership functions may be clear from the application, or given by an expert. If this is not the case, an automatic process can be used to extract suitable membership functions directly from the data.

C.1.1 Membership Function Shapes

Several membership function shapes have been proposed and used in literature. The most common functions used are the triangular, trapezoidal, and Gaussian functions. Except where stated otherwise, we utilized rough piecewise-linear bell shaped functions. This kind of function fits many real world problems [Surmann, 2000]. The bell shape is approximated using five lines, as shown in Figure C.1. A six line approximation of the Gaussian function was used for evolutionary optimisation of a fuzzy rule based system [Surmann, 2000]. Using an approximation instead of the real Gaussian function

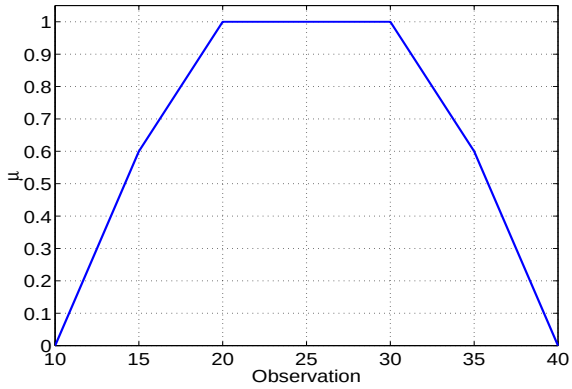


Figure C.1: A piecewise-linear bell shape function approximation.

speeds up computation, and allows for easier tuning of the membership functions. It further allows the membership function to be adapted later to any other shape, in which case the bell shape only serves as an initial starting point.

Let $\hat{\mu}_L(u)$ be any piecewise linear membership function, and let Q be an ordered set of n points in the Cartesian plain,

$$Q = \{q_i|, q_i < q_j \text{ for } i < j\} \tag{C.1}$$

where q_i is a point in the Cartesian plain. We define the less-than operation in terms of the x-coordinates, that is, q_i is less than q_j, $q_i < q_j$, if the x-coordinate of q_i, $q_i.x$, is less than the x-coordinate of q_j.

Definition C.1.1 $q_i < q_j \leftrightarrow q_i.x < q_j.x$

We then define $\hat{\mu}_L(u) = f(Q, u)$, where $f(Q, u)$ is defined as follows,

Definition C.1.2 $$f(Q,u) = \begin{cases} q_1 & \textit{for } u \leq q_1.x \\ q_n & \textit{for } u > q_n.x \\ \frac{q_{i+1}.y - q_i.y}{q_{i+1}.x - q_i.x} \cdot u - \frac{q_{i+1}.y - q_i.y}{q_{i+1}.x - q_i.x} \cdot q_i.x + q_i.y & \textit{for } q_i.x < u \leq q_{i+1}.x \end{cases}$$

Thus, $f(Q, u)$ defines the membership degree of any instance u to the linguistic term L for the ordered set of points Q.

C.1.2 Membership Function Extraction

Using definition C.1.2 we extract membership functions from a data set for all the continuous attributes using the algorithm shown in Table C.1. The algorithm computes the elements of Q as defined in Eq (C.1). The algorithm functions roughly as follows. A continuous attribute is picked, and a clustering is performed on the instances of the data set. Each point in the continuous domain can be assigned to a closest cluster centre, and the section of the continuous domain belonging to a given cluster defines the cluster interval. Along each interval a membership function of the shape depicted in Figure C.1 is placed. If an interval is completely subsumed by another it is deleted.

Table C.1: An algorithm for extracting bell shape membership functions from a data set, where the constants 0.25, 0.8 and 0.125 can be changed by the user.

FOR a continuous attribute A_i
  Cluster the instance data for A_i into a list of clusters C
  FOR each cluster $C_j \in C$ DO
    Set $min_j = \min(C_j)$, $max_j = \max(C_j)$, and $range_j = max_j - min_j$
  END FOR
  Sort C such that $C_j < C_{j+1}$ when $min_j < min_{j+1}$
  FOR each cluster $C_j \in C$ DO
    IF max_{j+1} exists AND $max_{j+1} < max_j$ THEN
      Delete C_{j+1}
    END IF
  END FOR
  FOR each cluster C_j DO
    IF min_{j+1} exists AND $min_{j+1} < max_j$ THEN
      $intersect = 0.5 \cdot (min_{j+1} - max_j)$;
      Set $max_j = max_j - intersect$; Set $min_{j+1} = min_{j+1} + intersect$
    END IF
    Set $Q_1 = (min_j - 0.25 \cdot range_j, 0)$;
    Set $Q_2 = (min_j, 0.8)$;
    Set $Q_3 = (min_j + 0.125 \cdot range_j, 1)$;
    Set $Q_4 = (max_j - 0.125 \cdot range_j, 1)$
    Set $Q_5 = (max_j, 0.8)$;
    Set $Q_6 = (max_j + 0.25 \cdot range_j, 0)$
    Set $MF_j(u) = f(Q, u)$;
  END FOR
END FOR

A variation of this algorithm would be to use the complete input domain when computing the cluster centres, i.e. to use all the continuous attributes together when clustering. This will mean that the cluster centres in one dimension are not independent from the other dimensions. This may have the effect that a classification algorithm will need less attributes to classify correctly. However, it may also be seen as smoothing of the training data, and may degrade the performance of a good classifier when smaller clusters in one dimension are grouped as a result of data grouping in other dimensions.

C.2 Influence of the Number of Clusters

We also conducted experiments to determine the influence of the number of membership functions (linguistic terms) extracted on the classification accuracy and the rule set comprehensibility of our algorithm. For this experiment we induced ordered rule sets using no beam search, the Accuracy evaluation function and $\theta_p = 2$. The value of α_a depended on the data set, and was kept constant for each respective data set for all experiments. For eight data sets we extracted fuzzy data sets with three, five, and seven membership functions per variable. We then performed 10-fold cross validation experiments for

Table C.2: Classification accuracy results for different configurations.

# clusters	3	5	7	Ripper2
anneal	99.3	97.8	93.1	98.3
breastcancer	70.8	71.6	70.1	77.3
colic	82.9	82.6	83.4	85.1
credit-a	84.9	85.2	85.4	86.2
hepatitis	81.3	81.9	81.3	78.1
iris	96.4	92.9	95.7	94.7
labor	89.5	89.5	87.7	77.2
lymph	78.4	77.7	78.4	79.1
average	**85.44**	**84.90**	**84.39**	**84.48**

Table C.3: Number of rules per extracted rule set for different configurations.

# clusters	3	5	7	Ripper2
anneal	7.0	9.0	7.0	7.0
breastcancer	5.0	5.0	5.0	2.0
colic	5.0	5.0	5.0	3.0
credit-a	4.0	3.0	2.0	4.0
hepatitis	4.0	3.0	3.0	4.0
iris	3.0	3.0	3.0	4.0
labor	2.0	2.0	2.0	4.0
lymph	5.0	5.0	5.0	6.0
average	**4.38**	**4.38**	**4.00**	**4.25**

the different fuzzy data sets and report the averages obtained.

Table C.2 shows the classification accuracies obtained by FUZZYBEXA for the different data sets, and a base line performance of RIPPER2. On average, FUZZYBEXA's classification performance was comparable or better than RIPPER2's for all configurations. FUZZYBEXA's average of the best result for each dataset amounts to 85.75%, compared to its average of 85.44% for three terms, and RIPPER's 84.48%. On average, the classification accuracy declined for an increasing number of terms. However, this is not necessarily true for each individual data set (e.g. credit-a). The anneal data set is the only one that exhibited a clear decline in classification accuracy.

Table C.3 shows the corresponding information for the number of rules per rule set. Although FUZZYBEXA does not post-prune it's rule sets whereas RIPPER2 does, FUZZYBEXA's rule set complexity is very similar to that of RIPPER2. FUZZYBEXA's smallest rule set had fewer rules than RIPPER on five data sets, on one they had the same, and on two data sets RIPPER had fewer rules. The number of rules per data set also decreased on average.

The main observation from the current experiment is that the number of clusters, and thus the number of membership functions (linguistic terms) extracted per variable, does not have a dramatic influence on either the classification accuracy performance or the rule set comprehensibility, as measured by rule set complexity, for these data sets. In general, however, increasing the number of terms of a linguistic variable, will require that more terms are needed to cover the same region as before since the domain is divided into smaller fuzzy sets. This in turn will increase rule complexity and undermine rule comprehensibility, in line with the arguments of Guillaume [2001].

Increasing the number of terms per variable will also negatively affect the search for the best rule. The search effort as measured by the average number of conjunctions examined per rule set for 3, 5 and 7 terms were 987, 1125 and 1402, respectively. This is a consistent increase with increasing number of membership functions. This should clearly be the case, since the size of the hypothesis space increases. However, the increase in hypothesis space is exponential, whereas we only observe an almost linear increase in search effort. This is further testimony to the effectiveness of FUZZYBEXA's search and over-fitting avoidance biases. We conjecture, however, that when a large number of terms per variable is extracted, that it will adversely affect classification accuracy as well. Due to the top-down search and bias toward generality of rules, the search will be led astray by small incremental improvements dictated by the many terms to choose from.

APPENDIX D

Neural Network Encoding of FuzzyAL Rules

Knowledge Based Neurocomputing (KBN) concerns the encoding, extraction, and refinement of knowledge in a neurocomputing paradigm [Cloete and Zurada, 2000]. Prior knowledge in a symbolic form, i.e. a domain theory, can serve to initialise an artificial neural network (ANN) so that this knowledge can be refined using all the techniques available for neural networks [Cloete, 1996; Cloete and Zurada, 2000]. These include further learning, analysis (such as sensitivity analysis) and rule refinement.

We provide preliminary results on addressing the problem of encoding prior knowledge in the form of FuzzyAL classification rules in ANN. A neural network encoding method provides a bridge from the symbolic to the connectionist knowledge representation. Although there exist methods for encoding purely conjunctive rules [Kasabov, 2001b], these methods cannot encode the internally disjunctive FuzzyAL rules extracted by FUZZYBEXA, do not contain the alpha complement $\bar{\alpha}$, and typically do not function directly with membership degree data.

The layout of the appendix is as follows. Section D.1 gives an overview of knowledge-based neural networks, Section D.2 introduces our rule encoding method, and Section D.3 demonstrates the encoding method by encoding rules extracted from the Fuzzy Sport data. Section D.4 demonstrates empirically that the encoding method provides a one-to-one mapping between the symbolic and the connectionist knowledge representations, and Section D.5 provides a summery.

D.1 Knowledge-Based Neural Networks

There exist several methods for encoding classification rules in neural networks. VL_1ANN [Cloete, 2000] encodes propositional rules in VL_1 syntax [Michalski, 1974b], and a knowledge based artificial neural network (KBANN) encodes Horn clauses [Towell and Shavlik, 1994]. Abraham provides an overview of Mamdani and Takagi Sugeno neuro-fuzzy systems, where neural networks are used to infer the membership functions and parameters for fuzzy inference systems [Abraham, 2001]. In our case we assume that the inputs to the neural network are membership values for each of the possible terms of a linguistic variable.

In our encoding method we use KBANN conjunctive and disjunctive neurons [Towell and Shavlik, 1994]. All the weights of a KBANN disjunctive neuron are programmed to a predefined value H,

and the bias input is set to -0.5. Thus, the weighted sum of inputs for a disjunctive neuron is

$$g(x) = \sum_i x_i H - 0.5H = H(\sum_i x_i - 0.5) \tag{D.1}$$

The input applied to KBANN neurons is binary, i.e. either 0 or 1 values. Thus, $g(x)$ will be at least $0.5H$ if any input is on, and if no input is on, $g(x) = -0.5H$. A big enough value for H will drive a sigmoidal neuron into saturation, causing it's output to be either on or off (i.e. approximately zero or one), depending on its input. KBANN conjunctive neurons are constructed similarly, but the bias is set to $P - 0.5$, where P is the number of programmed input weights (with weight $w_{ij} = H$) to the neuron. Thus, if all the inputs are on $g(x) = 0.5H$, and if at least one input is not on $g(x) \leq -0.5H$, effectively implementing the desired conjunctive property.

D.2 Encoding Extracted Rules

In this section we present the mapping from the symbolic to the connectionist domain. We will assume FuzzyCAL descriptions imply the standard fuzzy operations. The neural network we propose consists of six layers, the input, alpha complement, amplification, variable, rule, and class layers. FUZZYBEXA induces rules from fuzzy data where each instance potentially has a non-zero membership degree to each linguistic term. The neural network must function with the same input data, and therefore the input data determines the number of input neurons—one input neuron per linguistic term.

D.2.1 Amplifying Neurons

We encode the fuzzy rule set using KBANN conjunctive and disjunctive neurons [Towell and Shavlik, 1994]. KBANNs were originally designed for crisp rules. Fuzzy data may cause the KBANN neurons to malfunction under certain circumstances. To remedy this problem and provide the KBANN layers with suitable input, we add an amplification layer between the input and KBANN layers. The amplification layer is simply a layer of neurons with standard sigmoidal activation functions, where the activation function has a slope greater than 1,

$$f(g) = \frac{1}{e^{-\lambda g} + 1} \tag{D.2}$$

where g is the weighted input to the neuron, and the slope λ is set greater than 1. As the slope approaches infinity, this function approximates the step function. Each neuron in the input layer is connected to a corresponding neuron in the amplification layer with fixed weight of 1.

The bias weight of the amplifying neuron shifts the function to the left or right. We simulate the matching procedure by setting the bias weight to 1 and the bias input to $-\alpha_a$, or equivalently, setting the bias input to -1 and the weight to α_a. This will have the effect that fuzzy input values with membership degrees less than α_a will activate the neuron only weakly, whereas input values with membership degrees more than α_a will strongly activate the neuron. Thus, the function $g(x)$ for the amplifying neurons is calculated as,

$$g(x) = x + b = x - \alpha_a \tag{D.3}$$

D.2.2 Alpha Complement Neurons

The input data do not contain the linguistic term $\bar{\alpha}$ for linguistic variables. We simulate the $\bar{\alpha}$-term by adding a new type of neuron, the *alpha complement neuron* (denoted as $\bar{\alpha}$-neuron), to the network. An $\bar{\alpha}$-neuron is constructed for each linguistic variable. This neuron is connected to input neurons of the term set of its linguistic variable with weight value 1. It does not receive input from any other neurons. The output is connected to one amplifying neuron. Like the other amplifying neurons, its bias weight is set to 1 and the bias input to $-\alpha_a$. Each input neuron is therefore connected to two other neurons, an amplifying neuron and an $\bar{\alpha}$-neuron.

The activation function of the $\bar{\alpha}$-neuron implements the functionality of the $\bar{\alpha}$ linguistic term. The activation function adds the bias to the maximum input value to the neuron,

$$g(X) = \max(X) + b \tag{D.4}$$

where $X =< x_1, x_2, ..., x_n >$ is an input vector, b is the bias, and we set all weights to 1. It then inverts the result and applies it to the sigmoid function,

$$f(g) = \frac{1}{e^{\lambda g} + 1} \tag{D.5}$$

where λ is the slope of the activation function. The maximum membership will be equal or greater than α_a if the membership to any linguistic term is equal to or greater than α_a. If the bias α_a is subtracted from this value, the result will be positive, and the neuron will not fire. The maximum membership will be less than α_a only if the memberships to all linguistic terms are less than α_a. Since $-g(X)$ is applied to the sigmoid in this case the neuron will fire. This effectively implements the functionality of the $\bar{\alpha}$ linguistic term defined in Def (4.3.1).

D.2.3 The Variable, Rule and Class Layers

The rule section of the neural network can represent any conjunction in FuzzyAL, and is implemented using KBANN conjunctive and disjunctive neurons. The network has the following structure. A conjunctive *rule neuron* is created for each rule in the rule set. The activation of this neuron is analogous to the firing of its corresponding rule. For each class a disjunctive *class neuron* is created. The class neuron is connected to all the rule neurons with rules that have this class as the consequent. Thus, if any rule neuron of the corresponding class fires, the class neuron also fires. The rule neurons receive input from a group of disjunctive *variable neurons*. Each rule has its own set of variable neurons—one variable neuron for each linguistic variable for each rule. The variable neuron is connected to those amplification layer neurons which represent the linguistic terms in the term set of the corresponding linguistic variable.

The weights of a class neuron are all programmed to H, and its bias set to -0.5. The rule neuron weights connected to variable neurons of linguistic variables present in the rule are programmed to H, and the remainder are set to small random values. The bias of the rule neuron is set to $(0.5 - P)$, where P is the number of linguistic variables occurring in the rule. Thus, the rule neuron requires all its variable

neurons to fire before it will fire. The variable neurons are programmed according to the linguistic terms required in the corresponding rule. Only the weights connecting a variable neuron with amplifying neurons representing the linguistic terms present in the linguistic variable of the rule will be set to H, and the remainder will be set to small random values. The bias of variable neurons are all set to 0.5, and the bias weight of any KBANN neuron that contains programmed weights is set to H. All weights not set to H are initialized with small random values.

The "all except one" representation (e.g. $[!x]$) can also be programmed by setting the corresponding linguistic term weight to $-H$, and the bias input of the variable neuron to $+0.5$ instead of -0.5. This representation is only used when a single term is excluded. The input to the activation function of the variable neuron is now

$$g(x) = -Hx + bH = H(0.5 - x) \tag{D.6}$$

Thus, when the linguistic term neuron fires, the variable neuron does not fire, and vice versa.

D.2.4 Training the Network

Many fuzzy neural networks are trained with a genetic algorithm because the neuron activation functions are not differentiable [Abraham, 2001]. This methodology can find good results, but is often time consuming and slow in convergence. Our network can be trained using normal back propagation, or any other typical training method, as the activation functions of all neurons are differentiable. The weights of the amplifying and $\bar{\alpha}$-neurons should be kept fixed, since the only implication of changing these weights is a linear scaling of the input data. This is the case because each input neuron is only connected to an $\bar{\alpha}$-neuron and an amplifying neuron.

D.3 A Practical Example

We demonstrate the process of creating a neural network encoding of an extracted rule set by a practical example using the sport data set. The sport data set is repeated in Table D.1, and has four linguistic variables: **temperature**, **wind**, **outlook**, and **humidity**. The term set of temperature is $\{hot, mild, cold\}$. For wind it is $\{windy, calm\}$, for outlook $\{sunny, cloudy, rainy\}$, and for humidity $\{humid, normal\}$.

For each of the linguistic terms above an input neuron is created. Figure D.1 shows this structure and enumerates the input neurons from 0 to 9. For each of the four linguistic variables an $\bar{\alpha}$-neuron is created. These are numbered 10 to 13. The $\bar{\alpha}$-neuron for temperature, neuron 11, is connected to the input neurons for *hot*, *mild*, and *cold*. The bias values for the amplifying and $\bar{\alpha}$-neurons are all set to -0.6, since 0.6 was the α_a value used during rule induction. All weights between neurons enumerated 1 to 27 are set to 1, and are not updated during training.

We induced rules for the concepts $class.volleyball$ and $class.swimming$. Accordingly, there are two class neurons, one representing the volleyball class and the other the swimming class. For the volleyball

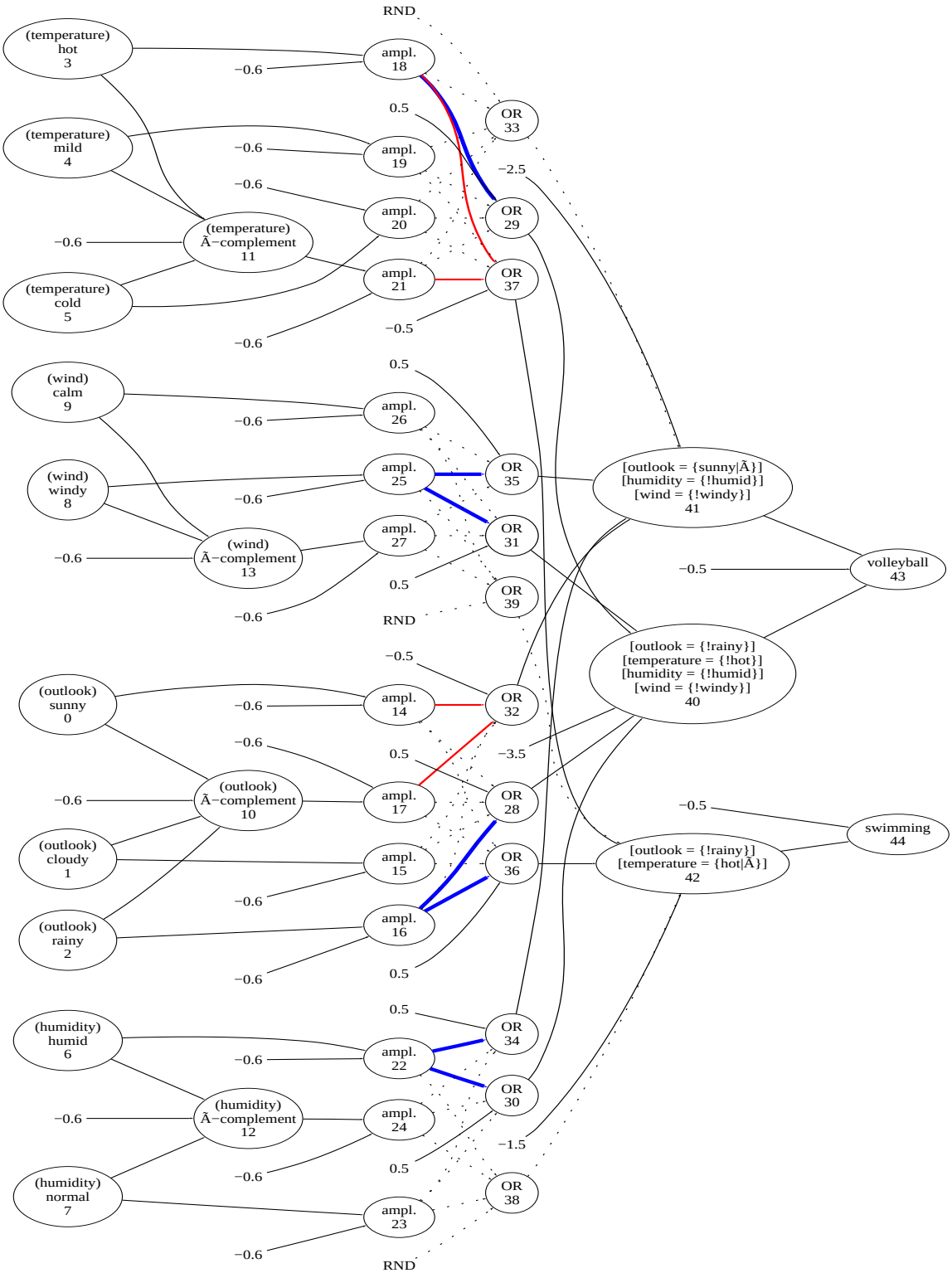


Figure D.1: The network generated for the FuzzySport data set for the classes *volleyball* and *swimming*. The alpha complement is indicated by the symbol Ã.

Table D.1: A fuzzy learning problem.

```
@attribute outlook  {sunny, cloudy, rainy}
@attribute temp     {hot, mild, cold}
@attribute humidity {humid, normal}
@attribute wind     {windy, calm}
@attribute activity {volleyball, swimming, weights}
@data
(.9 .1 .0), (1.  .0 .0), (.8 .2), (.4 .6), (.0 .8 .2) ;1
(.8 .2 .0), (.6 .4 .0), (.0 1.), (.4 .6), (1.  .7 .2) ;2
(.0 .7 .3), (.8 .2 .0), (.1 .9), (.2 .8), (.3 .6 .1) ;3
(.2 .7 .1), (.3 .7 .0), (.2 .8), (.3 .7), (.9 .1 .0) ;4
(.0 .1 .9), (.7 .3 .0), (.5 .5), (.5 .5), (.0 .0 1.)  ;5
(.0 .7 .3), (.0 .3 .7), (.7 .3), (.4 .6), (.2 .0 .8) ;6
(.0 .3 .7), (.0 .0 1.), (.0 1.), (.1 .9), (.0 .0 1.)  ;7
(.0 1.  .0), (.0 .2 .8), (.2 .8), (.0 1.), (.7 .0 .3) ;8
(1.  .0 .0), (1.  .0 .0), (.6 .4), (.7 .3), (.2 .8 .0) ;9
(.9 .1 .0), (.0 .3 .7), (.0 1.), (.9 .1), (.0 .3 .7) ;10
(.7 .3 .0), (1.  .0 .0), (1.  .0), (.2 .8), (.4 .7 .0) ;11
(.2 .6 .2), (.0 1.  .0), (.3 .7), (.3 .7), (.7 .2 .1) ;12
(.9 .1 .0), (.2 .8 .0), (.1 .9), (1.  .0), (.0 .0 1.)  ;13
(.0 .9 .1), (.0 .9 .1), (.1 .9), (.7 .3), (.0 .0 1.)  ;14
(.0 .0 1.), (.0 .0 1.), (1.  .0), (.8 .2), (.0 .0 1.)  ;15
(1.  .0 .0), (.5 .5 .0), (.0 1.), (.0 1.), (.8 .6 .0) ;16
```

class, two rules were found, and for the swimming class one rule was induced that perfectly covered the training data. Class neurons are disjunctive, and therefore their biases are set to -0.5 and the bias weight programmed to H. All programmed weights are indicated with solid lines in Figure D.1.

Neurons 18 to 21 represent the input to the rule section of the network for the linguistic variable temperature. There were 3 rules extracted, and therefore there are three variable neurons for the temperature linguistic variable—neurons 33, 29 and 37. Each of these three variable neurons are connected to the linguistic term amplifying neurons for the temperature linguistic variable, that is, neurons 18 to 21.

Neuron 41 represents the rule

$$[\text{outlook is } \{sunny \vee \bar{\alpha}\}][\text{humidity is } \{!humid\}][\text{wind is } \{!windy\}]$$
$$\rightarrow \text{volleyball}$$

The linguistic variable temperature does not occur in this rule, and therefore the connection between the rule neuron and the temperature variable neuron associated with this rule, neuron 33, is not programmed. All unprogrammed weights are set to small random values, and are indicated by dotted lines in Figure D.1. The connections between the variable neuron 33 and its linguistic term neurons 18 to 21 are also not programmed. Neuron 41 has three programmed variable neuron connections, and therefore its bias is set to -2.5 ($P - 0.5$). All bias weights (except for the 'excluded one' variable neurons) for programmed neurons are set to H, indicated by a thin solid line on the figure.

Neuron 40 represents the rule

$$[\text{outlook is } \{!rainy\}][\text{temperature is } \{!hot\}][\text{humidity is } \{!humid\}][\text{wind is } \{!windy\}]$$

$\rightarrow$ volleyball

This rule requires that [temperature is $!hot$]. Neuron 18 represents the *hot* linguistic term. Neuron 29 is the temperature variable neuron for this rule. Therefore the connection between them is programmed to $-H$, indicated by a thick line between them. The bias of neuron 29 is programmed to -0.5. Since the remaining linguistic terms are not present in the rule, the weights from neuron 29 to the remainder of the linguistic term neurons, neurons 19 to 21, are left unprogrammed. Neuron 40 has all 4 linguistic variables in its rule, and accordingly the bias is set to -3.5.

Neuron 42 represents the rule

[outlook is $\{!rainy\}$][temperature is $\{hot \vee \bar{\alpha}\}$]
$\rightarrow$ swimming

This rule requires the temperature linguistic variable to be either *hot* or have no term membership α_a or above. Neuron 37 is the temperature variable neuron for this rule. Neuron 18 represents the *hot* linguistic term, and neuron 21 the temperature $\bar{\alpha}$-neuron. Thus, the weights connecting neuron 37 to neurons 18 and 21 are programmed to H, indicated by a medium thick line on the figure. The bias is programmed to -0.5, and as stated above, its weight is also programmed to H. The rule represented by neuron 42 makes use of two linguistic variables, and accordingly its bias is set to -1.5.

D.4 Experimental Results

D.4.1 Sensitivity to the Threshold Slope and to H

The network structure discussed in the above paragraphs forms a one to one mapping with the extracted rules under the right conditions. The strength of the mapping will depend on the value of H used to program the network, and the value of λ used to program the slope of the amplifying neurons. If H is too small, the information from the rules will not be encoded strongly enough, and the cumulative effect of the small random weights will in some cases dominate the behaviour of the network. If a too small value of λ is used, the input to the KBANN section of the network will be too weak to ensure a one to one mapping without any training.

The best values of these two parameters will of course depend on the data set. For example, if a boolean data set is used, the value of λ will be less important. When most of the weights between the amplifying and variable neurons are programmed, the value of H will be less dominating. We tested our network by encoding rules induced by our FUZZYBEXA algorithm, and then plotting the error surface against H and λ for *untrained* networks.

Figure D.2 shows the classification error for different values of λ and H for an untrained network that encodes fuzzy rules induced for the data set in Table D.1. As expected, the error is large for small values of either λ or H. If one of the two parameters is programmed too weakly, the network does not properly encode the prior knowledge, e.g. for any value $\lambda < 3$ or $H < 3$ the error is greater than 25%. For $\lambda = 5.0$ and $H = 6.0$, no training is necessary to obtain zero error. Of course, this is the error on the

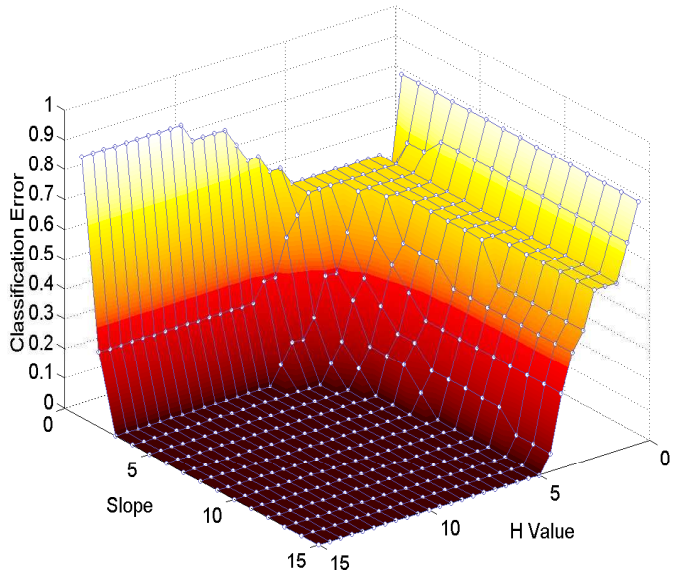


Figure D.2: The classification error as a function of H and λ for the FuzzySport data set.

training data, because for the purposes here we are only interested in how well the network encodes the prior knowledge.

Figure D.3 shows the results for the Iris data set. For the Iris data, a strong knowledge encoding using a large H value, e.g. $H = 10$, can force the error down even for small values of λ. Since the purpose of encoding the rules in the network is to allow it to be trained further, huge values of H may have a detrimental effect if the rules they encode are not exactly correct, as the strong encoding of rules will not be changed easily. Setting the slope even to a moderate value allows H to be much smaller while still maintaining acceptable error. Note that the inverse of the argument is not true—big values for λ never give good performance for small values of H. This is reasonable since very small H values correspond to little prior information.

The best values for further training lie at smaller values for H with moderate values of λ. Unfortunately it is not easy to say which values of H and λ will give good training performance. Snyders *et al* suggested a method for determining good values for H in KBANNs [Snyders and Omlin, 2000]. This method requires the calculation of $\frac{dE}{dH}$, with E the error. Using a similar methodology, it may also be possible to determine good values of λ by calculating $\frac{dE}{d\lambda}$.

D.4.2 Incremental Training

We have shown empirically in Section D.4.1 that the encoding method proposed in Section D.2 provides a one to one mapping between a rule set and neural network under the right conditions. The question remains whether this network can be trained to further refine its encoded knowledge.

To test this methodology we first generated a synthetic fuzzy data set, and then used three *a priori* rules to classify the data into two classes. We then used FUZZYBEXA to extract rules from this data, and encoded them into a neural network. Then an additional data set was created by synthesising more data

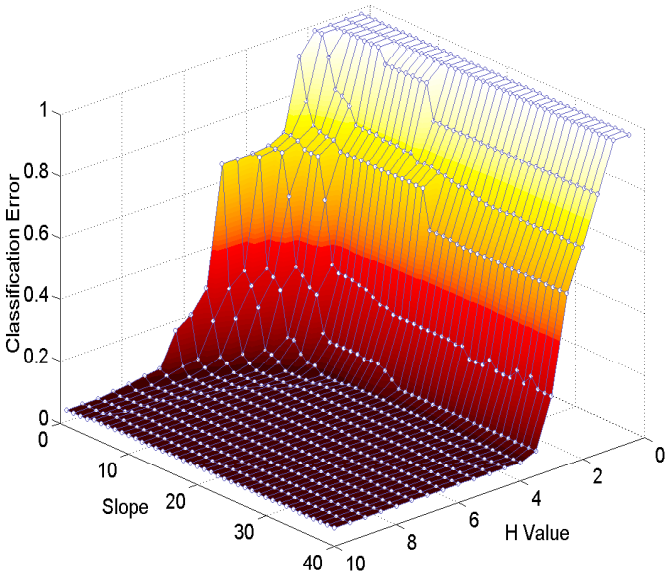


Figure D.3: The classification error as a function of H and λ for the Iris data set.

and classifying this data using the original rules plus an additional rule. This new data set was split into a training and test set, and used to train and evaluate the encoded neural network. The additional rule has the effect that some of the original instances were wrongly classified. The result is that some of the information encoded in the neural network will be good and some bad. The network will have to "unlearn" some information while keeping the correct information.

We encoded the rule information using $H = 4.0$, $\lambda = 10.0$, and trained the network using backpropagation with a learning rate of 0.1 and momentum of 0.1. To count a pattern as correctly classified, all output neurons must be correct within a specified range. The training classification error range was set to 0.25 and the test classification range to 0.5. Note that this method of error evaluation is not exactly the same as that used by the rule extractors. It is more strict, since only the rule neurons of the correct class may fire if the pattern is to be counted as correctly classified. Rule inference systems typically use a rule conflict resolution scheme, as discussed in Section 4.9.

Figure D.4 shows the root mean squared error on the training and test sets. The network starts with an initial error of 0.07, and is able to reduce the error on the test set to 0.009 after a thousand epochs. The training error decreased even more, but started to overfit the data after about 600 epochs. Figure D.5 shows the classification accuracy for the test and training sets. During the first 400 epochs the classification error was rapidly reduced after which overfitting seemed to set in. The neural network is clearly able to refine and change the encoded information.

We also used FUZZYBEXA [Cloete and van Zyl, 2006] to extract rules for the training set with a 10-fold cross validation. FUZZYBEXA obtained 96.7% accuracy on the training set and 96.7% on the test set. After training the neural network for 400 epochs a maximum accuracy of 98.0% on the test set is obtained. A maximum accuracy on the training set of 98.8% is obtained after 900 epochs. With no training the network contained only partial knowledge, and classified 65% of the patterns correctly.

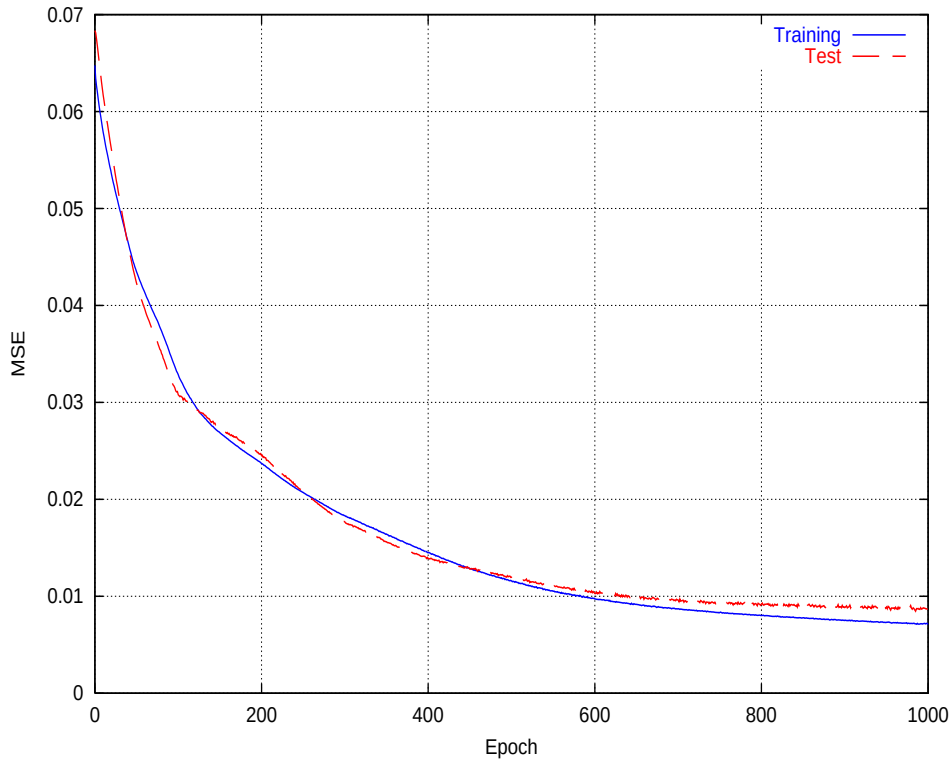


Figure D.4: The root mean squared error at each epoch of training on the training and test sets.

D.5 Summary

In this appendix we presented an encoding method for encoding prior knowledge in the form of FuzzyAL rules (e.g. like those induced by FUZZYBEXA) into a neural network. Two parameters, the slope λ and the weight encoding strength H, are used to specify how strongly the prior information is encoded. For suitably big values of λ and H the encoding method provides a one-to-one mapping between the symbolic and connectionist knowledge representations. Since we only make use of differentiable activation functions, the neural network can be trained using gradient descent. We investigated the behaviour of the error for different values of λ and H by plotting the error surface for different data sets. Moderate values of λ allow the knowledge encoding strength H to be sufficiently small to allow further knowledge refinement. We also showed empirically that the network is able to correct wrongly encoded information, retain correct information, and further refine its knowledge when provided with new training data.

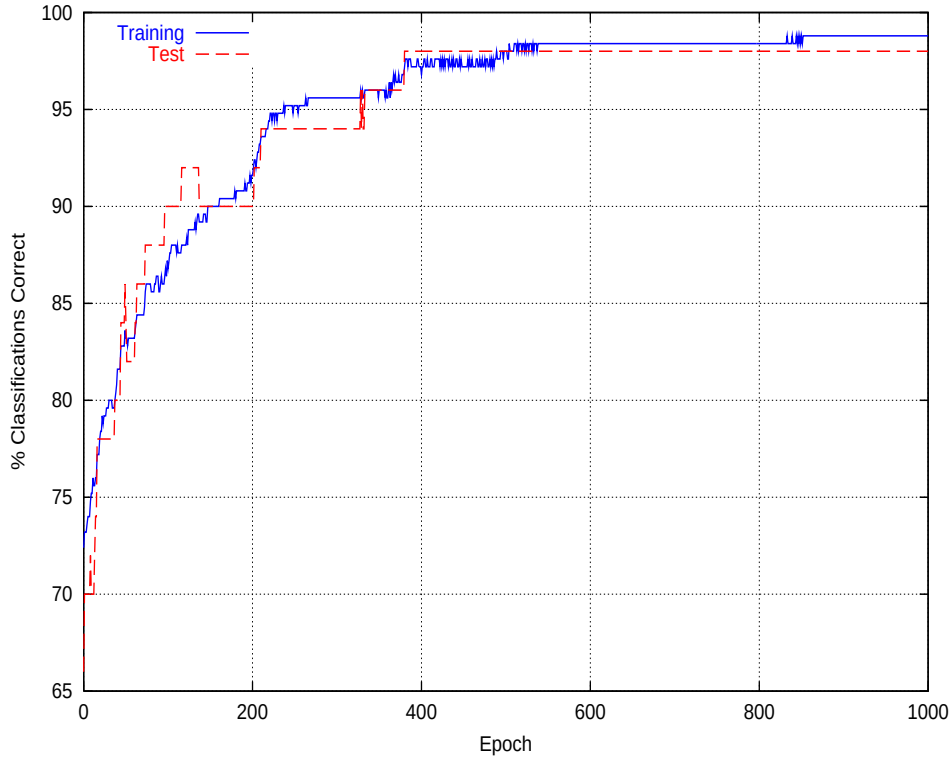


Figure D.5: The classification accuracy of the training and test sets at each epoch of training.

APPENDIX E

Ling-Spam Rule Set

IF *linguistic.med* THEN *class.ham*
ELSE IF *linguistics.med* THEN *class.ham*
ELSE IF *languages.med* THEN *class.ham*
ELSE IF *!.med* THEN *class.spam*
ELSE IF *language.med* THEN *class.ham*
ELSE IF *english.med* THEN *class.ham*
ELSE IF *deadline.med* THEN *class.ham*
ELSE IF *summary.low* THEN *class.ham*
ELSE IF *click.med* THEN *class.spam*
ELSE IF *linguist.low* THEN *class.ham*
ELSE IF *remove.low* THEN *class.spam*
ELSE IF *edu.high* THEN *class.ham*
ELSE IF *speech.med* THEN *class.ham*
ELSE IF *your.med* THEN *class.spam*
ELSE IF *between.med* THEN *class.ham*
ELSE IF *programme.low* THEN *class.ham*
ELSE IF *click.low* THEN *class.spam*
ELSE IF *in.med* THEN *class.ham*
ELSE IF *references.low* THEN *class.ham*
ELSE IF *1995.med* THEN *class.ham*
ELSE IF *today.low* THEN *class.spam*
ELSE IF *john.low* THEN *class.ham*
ELSE IF *).med* THEN *class.ham*
ELSE IF *ever.low* THEN *class.spam*
ELSE IF *words.low* THEN *class.ham*
ELSE IF *grammar.low* THEN *class.ham*
ELSE IF *100.low* THEN *class.spam*
ELSE IF *know.low* THEN *class.ham*
ELSE IF *our.med* THEN *class.spam*
ELSE IF *out.med* THEN *class.ham*
ELSE IF *conference.med* THEN *class.ham*
ELSE IF *de.high* THEN *class.ham*
ELSE IF *j.low* THEN *class.ham*
ELSE IF *go.low* THEN *class.spam*
ELSE IF *list.low* THEN *class.ham*
ELSE IF *french.med* THEN *class.ham*
ELSE IF *always.low* THEN *class.spam*
ELSE IF *com.*$\bar{\alpha}$ THEN *class.ham*
ELSE IF *over.low* THEN *class.ham*
ELSE IF *free.low* THEN *class.spam*
ELSE IF *have.med* THEN *class.ham*
ELSE IF *big.low* THEN *class.spam*
ELSE IF *programs.low* THEN *class.ham*
ELSE IF *modern.*$\bar{\alpha}$ THEN *class.spam*
ELSE *class.ham*

APPENDIX F

Decision Tree for Non-Overlapping Rule Set

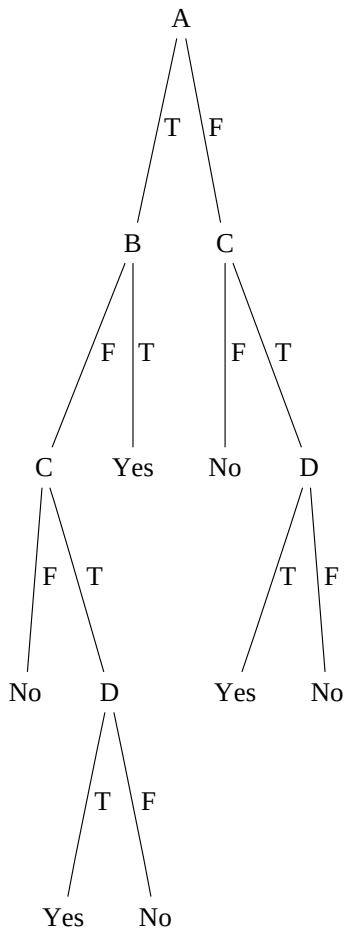


Figure F.1: The decision tree equivalent to the propositional logic rule set $A \wedge B \rightarrow Yes$ and $C \wedge D \rightarrow No$, where the replicated subtree at node C is clearly visible.

APPENDIX G

Publications Resulting from this Dissertation

1. Cloete, I. and van Zyl, J.: 2006, Fuzzy rule induction in a set covering framework, *IEEE Transactions of Fuzzy Systems* **14**(1), 93–110.

2. van Zyl, J. and Cloete, I.: 2006, Specialization models for the general fuzzy set covering framework, *Fuzzy Sets and Systems* **157**(21), 2787–2808.

3. van Zyl, J. and Cloete, I.: 2004, Simultaneous concept learning of fuzzy rules, *Proceedings of the 15th European Conference on Machine Learning*, Vol. 3201 of *Lecture Notes in Artificial Intelligence*, Pisa, Italy, pp. 548–559.

4. van Zyl, J. and Cloete, I.: 2004, FuzzyPRISM: a specialization model for the FuzzyBexa framework, *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom.

5. van Zyl, J. and Cloete, I.: 2004, Heuristic functions for learning fuzzy conjunctive rules, *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands.

6. van Zyl, J. and Cloete, I.: 2004, An inductive algorithm for learning conjunctive fuzzy rules, *International Conference on Machine Learning and Cybernetics*, Shanghai, China, pp. 4181–4188.

7. van Zyl, J. and Cloete, I.: 2004, Prior knowledge for fuzzy knowledge-based artificial neural networks from fuzzy set covering, *International Joint Conference on Neural Networks*, Budapest, Hungary.

8. van Zyl, J. and Cloete, I.: 2004, FuzzConRI - a fuzzy conjunctive rule inducer, *Proceedings of the Workshop on Advances in Inductive Rule Learning, ECML*, Pisa, Italy, pp. 194–203.

9. Cloete, I. and van Zyl, J.: 2004, Evaluation function guided search for fuzzy set covering, *IEEE International Conference on Fuzzy Systems*, Budapest, Hungary.

10. Cloete, I. and van Zyl, J.: 2004, Fuzzy set covering with FuzzyBexa, *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom.

11. Cloete, I. and van Zyl, J.: 2004, A machine learning framework for fuzzy set covering algorithms, *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands.

12. Robbel, P., van Zyl, J. and Cloete, I.: 2004, Rule pruning strategies for fuzzy classifiers, *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom.

Glossary

$\prec$	more specific than
$\preceq$	more specific than or equal to
$\succ$	more general than
$\succeq$	more general than or equal to
α_a	antecedent threshold
α_c	consequent threshold
α_{aT}	antecedent alpha-cut value used during rule induction
α_{aI}	antecedent alpha-cut value used during inference
$\bar{\alpha}$	alpha complement
I	instance space
N	set of negative instances
P	set of positive instances
T	set of training instances
θ_p	positive coverage threshold
$M(S, c)$	sigma count of description c in the set of instances S
$X_S(d)$	extension of the description d in the set of instances S
AQR	A^q family of algorithms
ANN	Artificial Neural Network
BEXA	Basic Exclusion Algorithm
CN2	Clark and Niblett's algorithm 2
ICL	Iterated Concept Learning
KBANN	knowledge-based artificial neural network
mgc	most general conjunction
FARFF	Fuzzy Attribute Ralation File Format

FBS	Fuzzy Beam Search algorithm
FCF	Fuzzy Covering Framework
FEM	Fuzzy Exclusion Model
FOIL	First-Order Inductive Learner
FRIwE	Fuzzy Rule Identification with Exceptions
FuzzyAL	Fuzzy Attributional Logic
FuzzyBEXA	Fuzzy Basic Exclusion Algorithm
FuzzConRi	Fuzzy Conjunctive Rule Inducer
FuzzyCAL	Fuzzy Conjunctive Attributional Logic
ID3	Iterative Dichotomiser 3
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
SCL	Simultaneous Concept Learning

INDEX

BIBLIOGRAPHY

Abe, S. and Inoue, T.: 2002, Fuzzy support vector machines for multiclass problems, *Proc. ESANN*, Bruges, Belgium, pp. 113–118. 18

Abonyi, J., Roubos, J. A. and Szeifert, F.: 2003, Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization, *International Journal of Approximate Reasoning* **32**(1), 1–21. 15

Abraham, A.: 2001, Neuro fuzzy systems: State-of-the-art modeling techniques, *in* J. Mira and A. Pierto (eds), *IWANN 2001*, Vol. 2084 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 269–276. 189, 192

Alander, J. T.: 1997, An indexed bibliography of genetic algorithms with fuzzy logic, *in* W. Pedrycz (ed.), *Fuzzy Evolutionary Computation*, Kluwer Academic, Boston, pp. 299–318. 19

Alaru, C. and Wehenkel, L.: 2003, A complete fuzzy decision tree technique, *Fuzzy Sets and Systems* **138**, 221–254. 15

Ali, K. M. and Pazzani, M. J.: 1993, Hydra: A noise-tolerant relational concept learning algorithm, *in* R. Bajcsy (ed.), *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1064–1071. 98

Andrews, R., Diederich, J. and Tickle, A. B.: 1995, A survey and critique of techniques for extracting rules from trained artificial neural networks, *Knowledge-Based Systems* **8**, 373–389. 1

Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Sakkis, G., Spyropoulos, C. and Stamatopoulos, P.: 2000, Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach, *Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*. 165

Angelov, P. P.: 2003, An evolutionary approach to fuzzy rule-based models synthesis using indices for rules, *Fuzzy Sets and Systems* **137**, 325–338. 20

Baraldi, A. and Alpaydin, E.: 2002, Constructive feedforward art clustering networks, *IEEE Transactions on Neural Networks* **13**(3), 645–661. 16

Baraldi, A. and Blonda, P.: 1999, A survey of fuzzy clustering algorithms for pattern recognition. II, *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* **29**, 786–801. 16

Bergadano, F., Giordana, A. and Saitta, L.: 1988, Automated concept acquisition in noisy environments, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**(4), 555–578. 26

Berthold, M. and Huber, K.-P.: 1995, From radial to rectangular basis functions: A new approach for rule learning from large datasets, *Technical Report 15-95*, University of Karlsruhe. 165

Berthold, M. R.: 2003, Mixed fuzzy rule formation, *International Journal of Approximate Reasoning* **32**, 67–84. 23, 25, 114

Berthold, M. R., Wiswedel, B. and Patterson, D. E.: 2002, Neighborgram clustering interactive exploration of cluster neighborhoods, *IEEE International Conference on Data Mining*, Maebashi City, Japan, pp. 581–584. 17

Bezdek, J.: 1981, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum, New York. 16

Bezdek, J. and Pal, S. (eds): 1992, *Fuzzy Models for Pattern Recognition*, IEEE Press. 16

Blake, C. L. and Merz, C. J.: 1998, UCI repository of machine learning databases. URL: http://www.ics.uci.edu/∼mlearn/. 24, 71, 73, 104, 140

Botta, M. and Giordana, A.: 1993, SMART+: A multi-strategy learning tool, *Proc. of the IJCAI*, pp. 937–944. 26

Botta, M., Giordana, A. and Saitta, L.: 1993, Learning fuzzy concept definitions, *Proc. of the IEEE Conference on Fuzzy Systems*, pp. 18–23. 26

Bouchon-Meunier, B., Rifqi, M. and Bothorel, S.: 1996, Towards general measures of comparison of objects, *Fuzzy Sets and Systems* **84**(2), 143–153. 14

Boyen, X. and Wehenkel, L.: 1999, Automatic induction of fuzzy decision trees and its application to power system security assessment, *Fuzzy Sets and Systems* **102**(1), 3–19. 14, 97

Carmona, P., Castro, J. L. and Zurita, J. M.: 2004, FRIwE: Fuzzy rule identification with exceptions, *IEEE Transactions on Fuzzy Systems* **12**(1), 140–151. 24, 25, 46, 112, 114

Carpenter, G., Grossberg, S., and Rosen, D.: 1991, Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, *Neural Networks* **4**, 759–771. 16

Casillas, J., Cordón, O. and Herrera, F.: 2000, Improving the wang and mendel's fuzzy rule learning method by inducing cooperation among rules, *Proceedings of IPMU*, Vol. 3, Madrid, Spain, pp. 1682–1688. 2, 20, 114

Castro, J. L., Castro-Schez, J. J. and Zurita, J. M.: 1999, Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems, *Fuzzy Sets and Systems* **101**, 331 – 342. 24, 112

Castro, J. L., Delgado, M. and Herrera, F.: 1993, A learning method of fuzzy reasoning by genetic algorithms, *First European Congress on Fuzzy and Intelligent Technologies*, Aachen. 19

Castro, J. L. and Zurita, J. M.: 1997, An inductive learning algorithm in fuzzy systems, *Fuzzy Sets and Systems* **89**, 193 – 203. 28

Cendrowska, J.: 1987, PRISM: An algorithm for inducing modular rules, *International Journal of Man-Machines Studies* **27**, 349–370. 1, 3, 9, 29, 35, 36, 97, 134, 176

Cervone, G., Panait, L. A. and Michalski, R. S.: 2001, The development of the AQ20 learning system and initial experiments, *Tenth International Symposium on Intelligent Information Systems*, Poland. 175

Chang, R. L. P. and Pavlidis, T.: 1977, Fuzzy decision tree algorithms, *IEEE Trans. on Systems, Man, and Cybernetics* **7**(1), 28–35. 13

Chen, Y. and Wang, J. Z.: 2003, Support vector learning for fuzzy rule-based classification systems, *IEEE Transactions on Fuzzy Systems* **11**(6), 716–728. 18

Chiang, I.-J. and jen Hsu, J. Y.: 2002, Fuzzy classification trees for data analysis, *Fuzzy Sets and Systems* **130**, 87–99. 15

Chiang, J.-H. and Hao, P.-Y.: 2004, Support vector learning mechanism for fuzzy rule-based modeling: A new approach, *IEEE Transactions on Fuzzy Systems* **12**(1), 1–12. 18

Cios, K. J. and Sztandera, L. M.: 1992, Continuous ID3 algorithm with fuzzy entropy measures, *Proc. IEEE Int. Conf. Fuzzy Syst.*, pp. 469–476. 3, 13, 15, 110, 149, 183

Clark, P. and Boswell, R.: 1991, Rule induction with CN2: Some recent improvements, *Proceedings of the Sixth European Working Session on Learning*, pp. 151–163. 115, 149, 177

Clark, P. and Niblett, T.: 1989, The CN2 induction algorithm, *Machine Learning* **3**, 261–283. 1, 3, 7, 10, 29, 35, 36, 145, 176, 177

Cloete, I.: 1996, An algorithm for fusion of rules and artificial neural networks, *Workshop on "Foundations of Information/Decision Fusion: Applications to Engineering Problems," Washington D.C.*, pp. 40–45. 189

Cloete, I.: 2000, VL_1ANN: Transformation of rules to artificial neural networks, *in* I. Cloete and J. Zurada (eds), *Knowledge-Based Neurocomputing*, MIT Press, chapter 6, pp. 207–216. 23, 189

Cloete, I. and van Zyl, J.: 2004a, Evaluation function guided search for fuzzy set covering, *IEEE International Conference on Fuzzy Systems*, Budapest, Hungary. 5, 107, 171

Cloete, I. and van Zyl, J.: 2004b, Fuzzy set covering with FuzzyBexa, *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom. 5, 171

Cloete, I. and van Zyl, J.: 2004c, A machine learning framework for fuzzy set covering algorithms, *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands. 5, 171

Cloete, I. and van Zyl, J.: 2006, Fuzzy rule induction in a set covering framework, *IEEE Transactions of Fuzzy Systems* **14**(1), 93–110. 5, 6, 125, 145, 171, 197

Cloete, I. and Zurada, J. M. (eds): 2000, *Knowledge-Based Neurocomputing*, MIT Press, Mass. 189

CNET News.com: Retrieved from the WWW on 21 December 2004, Isp nets $1bn in spam case. http://www.silicon.com/research/specialreports/thespamreport/0,39025001,39126638,00.htm. 163

Cohen, W. W.: 1995, Fast effective rule induction., *Proceedings of ICML 95*, pp. 115–123. 157, 177, 178, 179

Cordón, O., Gomide, F., Herrera, F., Hoffmann, F. and Magdalena, L.: 2004, Ten years of genetic fuzzy systems: current framework and new trends, *Fuzzy Sets and Systems* **141**, 5–31. 2, 19

Cordón, O., Herrera, F., Hoffmann, F. and Magdalena, L.: 2001, *Genetic Fuzzy Systems - Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, World Scientific, Singapore. 19

Cordón, O., Herrera, F. and Villar, P.: 2001, Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base, *IEEE Transactions of Fuzzy Systems* **9**, 667–674. 19

Cordón, O., Herrera, F. and Zwir, I.: 2002, Linguistic modeling by hierarchical systems of linguistic rules, *IEEE Transactions of Fuzzy Systems* **10**(1), 2–20. 23

Cox, E.: 1998, *The Fuzzy Systems Handbook*, 2 edn, Academic Press, London. 39, 46, 48, 52, 66, 68, 81

Craven, M. and Shavlik, J. W.: 1994, Using sampling and queries to extract rules from trained neural networks, *International Conference on Machine Learning*, pp. 37–45. 23

Cristianini, N. and Shawe-Taylor, J.: 2000, *An Introduction to Support Vector Machines*, Cambridge University Press. 18

Daelemans, W., Zavrel, J., van der Sloot, K. and van den Bosch, A.: 2000, TiMBL: Tilburg memory-based learner - version 3.0, reference guide, *Technical report*, ILK, Computational Linguistics, Tilburg University. 165

Dasgupta, D. and González, F. A.: 2001, Evolving Complex Fuzzy Classifier Rules Using a Linear Tree Genetic Representation, *in* L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon and E. Burke (eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, Morgan Kaufmann Publishers, San Francisco, California, pp. 299–305. 19

Davey, B. A. and Priestly, H. A.: 2002, *Introduction to Lattices and Order*, 2 edn, Cambridge University Press. 53, 65, 67

de Kleer, J.: 1986, An assumption-based TMS, *Artificial Intelligence* **28**, 127–162. 28

Dong, M. and Kothari, R.: 2001, Look-ahead based fuzzy decision tree induction, *IEEE-FS* **9**, 461–468. 15, 97, 110, 114, 149

Drobics, M., Bodenhofer, U. and Klement, E. P.: 2003, FS-FOIL: An inductive learning method for extracting interpretable fuzzy descriptions, *International Journal of Approximate Reasoning* **32**, 131–152. 11

Dubois, D., Esteva, F., Carcia, P., Godo, L., Mantaras, R. L. and Prade, H.: 1998, Fuzzy set modeling in case-based reasoning, *International Journal of Intelligent Systems* **13**, 345–373. 17

Dubois, D., Hüllermeier, E. and Prade, H.: 2002, Fuzzy set-based methods in instance-based reasoning, *IEEE Transactions on Fuzzy Systems* **10**(3), 322–332. 8, 17, 110, 114

Dubois, D. and Prade, H.: 2003, Fuzzy set and possibility theory-based methods in artificial intelligence, *Artificial Intelligence* **148**, 1–9. 40

Duch, W., Adamczak, R. and Grabczewski, K.: 2000, A new methodology of extraction, optimization and application of crisp and fuzzy logical rules, *IEEE Transactions on Neural Networks* **11**(2). 1, 23

Dunn, J. C.: 1974, A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters, *Cybernet* **3**(3), 32–57. 16

Elomaa, T.: 1994, In defense of c4.5: Notes on learning one-level decision trees, *in* W. Cohen and H. Hirsch (eds), *Proceedings 11th Intl. Conf. Machine Learning*, Morgan Kaufmann, pp. 62–69. 158

Faifer, M., Janikow, C. and Krawiec, K.: 1999, Extracting fuzzy symbolic representation from artificial neural networks, *Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society*, New York, pp. 600–604. 23

Fertig, C. S., Freitas, A. A., Arruda, L. V. R. and Kaestner, C.: 1999, A Fuzzy Beam-Search Rule Induction Algorithm, *in* J. Zytkow and J. Rauch (eds), *Principles of Data Mining and Knowledge Discovery (Proc 3rd European Conf - PKDD-99)*, Vol. 1704 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, pp. 341–347. 10, 95, 99, 112, 114, 160, 161

Frayman, Y., Ting, K. and Wang, L.: 1999, A fuzzy neural network for data mining: dealing with the problem of small disjuncts, *International Joint Conference on Neural Networks*, Washington, DC. 23

Fürnkranz, J.: 1999, Separate-and-conquer rule learning, *Artificial Intelligence Review* **13**(1), 3–54. 3, 95, 97, 98, 99, 159

Fürnkranz, J. and Flach, P.: 2004, An analysis of stopping and filtering criteria for rule learning, *Proceedings of the 15th European Conference on Machine Learning (ECML-04)*, Springer-Verlag, Pisa, Italy, pp. 123–133. 157, 177

Fürnkranz, J. and Widmer, G.: 1994, Incremental reduced error pruning, *Proceedings of the 11th annual conference on machine learning*, Morgan Kaufmann, New Brunswick, New Jersey. 177

Gabriel, T. R. and Berthold, M. R.: 2003, Constructing hierarchical rule systems, *Proceedings of the 5th International Symposium on Intelligent Data Analysis*, Berlin, Germany. 23

Garibaldi, J. M. and Ifeachor, E. C.: 1999, Application of Simulated Annealing fuzzy model tuning to Umbilical Cord Acid-Base interpretation, *IEEE-FS* **7**(1), 72–84. 18

Gath, I. and Geva, A. B.: 1989, Unsupervised optimal fuzzy clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(7), 773–781. 17

Gaweda, A. E. and Zurada, J. M.: 2003, Data-driven linguistic modeling using relational fuzzy rules, *IEEE Transactions on Fuzzy Systems* **11**(1), 121–134. 18

Gedeon, T. D., Kuo, H. and Wong, P. M.: 2002, Rule extraction using fuzzy clustering for a sedimentary rock data set, *International Journal of Fuzzy Systems* **4**(1), 600–605. 18

Goldberg, D. E.: 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley. 18

Gómez, J., Dasgupta, D., Nasraoui, O. and González, F.: 2002, Complete expression trees for evolving fuzzy classifiers systems with genetic algorithms and application to network intrusion detection, *NAFIPS-FLINT joint conference*, New Orleans, pp. 469–474. 19

Gray, N. A. B.: 1990, Capturing knowledge through top-down induction of decision trees, *IEEE Expert* **5**(3), 41–50. 29, 35

Guély, F., La, R. and Siarry, P.: 1999, Fuzzy rule base learning through simulated annealing, *Fuzzy Sets and Systems* **105**, 353–363. 18

Guetova, M., Hölldobler, S. and Störr, H.: 2002, Incremental fuzzy decision trees, *in* M. Jarke, J. Koehler and G. Lakemeyer (eds), *25th Annual German Conference on AI*, Vol. 2479 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 67–81. 3, 15, 97

Guillaume, S.: 2001, Designing fuzzy inference systems from data: an interpretability-oriented review, *IEEE Transactions on Fuzzy Systems* **9**, 426–443. 2, 4, 39, 45, 159, 169, 186

Gustafson, D. E. and Kessel, W. C.: 1979, Fuzzy clustering with a fuzzy covariance matrix, *Proc. IEEE CDC*, San Diego, CA, pp. 761–766. 16, 17

Hayken, S.: 1999, *Neural Networks, A Comprehensive Foundation*, Prentice-Hall, Upper Saddle River, NJ. 113

Heinke, D. and Hamker, F. H.: 1998, Comparing neural networks: a benchmark on growing neural gas, growing cell structures, and fuzzy ARTMAP, *IEEE Transactions on Neural Networks* **9**(6), 1279–1291. 16

Herrera, F. and Lozano, M.: 1998, Fuzzy genetic algorithms: Issues and models, *Technical Report DECSAI-98116*, Dept. of Computer Science and A.I., University of Granada, Spain. 19

Herrera, F., Lozano, M. and Verdegay, J. L.: 1994, Generating fuzzy rules from examples using genetic algorithms, *Proc. IPMU'94 (5th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems)*, pp. 675–680. 19, 114

Hoffmann, F.: 2004, Combining boosting and evolutionary algorithms for learning of fuzzy classification rules, *Fuzzy Sets and Systems* **141**, 47–58. 19, 20

Holland, J. H.: 1975, *Adaptation in Natural and Artificial Systems*, University of Michigan Press (reprinted in 1992 by MIT Press, Cambridge, MA). 111

Holte, R. C.: 1993, Very simple classification rules perform well on most commonly used datasets, *Machine Learning* **1**, 63–91. 158

Holte, R. C., Acker, L. E. and Porter, B. W.: 1989, Concept learning and the problem of small disjuncts, *Proc. of the 11th International Joint Conference on Artificial Intelligence*, pp. 813–818. 104

Holve, R.: 1997, Rule generation for hierarchical fuzzy systems, *Proc-NAFIPS*, Syracuse, NY, pp. 444–449. 23

Holve, R.: 1998, Investigation of automatic rule generation for hierarchical fuzzy systems, *Proc. of the IEEE Conference on Fuzzy Systems*, Anchorage, Alaska, pp. 973–978. 23

Hong, T. and Chen, J.: 1999, Finding relevant attributes and membership functions, *Fuzzy Sets and Systems* **103**, 389–404. 21, 111

Hong, T. and Chen, J.: 2000, Processing individual fuzzy attributes for fuzzy rule induction, *Fuzzy Sets and Systems* **112**, 127–140. 2, 21

Hong, T.-P. and Lee, C.-Y.: 1996, Induction of fuzzy rules and membership functions from training examples, *Fuzzy Sets and Systems* **84**(1), 33–47. 17, 21, 114

Huang, H.-P. and Liu, Y.-H.: 2002, Fuzzy support vector machines for pattern recognition and data mining, *International Journal of Fuzzy Systems* **3**(4), 826–835. 18

Huber, K.-P. and Berthold, M.: 1995, Building precise classifiers with automatic rule extraction, *Proc. of the IEEE Int. Conf. on Neural Networks (ICNN)*, Perth, Australia, pp. 1263–1268. 165

Inoue, T. and Abe, S.: 2001, Fuzzy support vector machines for pattern classification, *Proc. International Joint Conference on Neural Networks*, Washington, D.C., pp. 1449–1454. 18

Ishibuchi, H. and Nakashima, T.: 2001, Effect of rule weights in fuzzy rule-based classification systems, *IEEE Trans. on Fuzzy Systems* **9**(4), 506–515. 21

Ishibuchi, H., Nakashima, T., Mijamoto, H. and Oh, C. H.: 1997, Fuzzy Q-learning for a multiplayer non-cooperative repeated game, *Proc. 6th IEEE International Conference on Fuzzy Systems*, Barcelona, pp. 1573–1579. 24

Ishibuchi, H., Nozaki, K. and Tanaka, H.: 1992, Distributed representation of fuzzy rules and its application to pattern classification, *Fuzzy Sets and Systems* **52**, 21–32. 19, 21, 23, 24

Ishibuchi, H., Nozaki, K., Yamamoto, N. and Tanaka, H.: 1995, Selecting fuzzy if-then rules for classification problems using genetic algorithms, *IEEE Trans. on Fuzzy Systems* **4**(3), 260–270. 19, 21, 45, 112, 114

Ishibuchi, H., Sakamoto, R. and Nakashima, T.: 2003, Learning fuzzy rules from iterative execution of games, *Fuzzy Sets and Systems* **135**, 213–240. 24

Ishibuchi, H. and Yamamoto, T.: 2004, Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining, *Fuzzy Sets and Systems* **141**, 59–88. 19, 114

Jain, R. and Abraham, A.: 2003, A comparative study of fuzzy classifiers on breast cancer data, *Proc. of IWANN*, LNCS 2687, Springer, Maó, Menorca, Spain. 23

Janikow, C. Z.: 1996, Exemplar learning in fuzzy decision trees, *FUZZIEEE*, New Orleans, USA, pp. 1500–1505. 14

Janikow, C. Z.: 1998, Fuzzy decision trees: Issues and methods, *IEEE Trans. on Systems, Man, and Cybernetics, Part B* **28**(1), 1–14. 14

Janikow, C. Z. and Fajfer, M.: 1999, Fuzzy partitioning with FID3.1, *Proceedings of the 18th International Conference of the North American Fuzzy Information Society*, pp. 467–471. 14, 160

Joo, M. G. and Lee, J. S.: 2002, Universal approximation by hierarchical fuzzy system with constraints on the fuzzy rule, *Fuzzy Sets and Systems* **130**, 175–188. 23

Kalbfleisch, J.: 1979, *Probability and Statistical Inference*, Springer Verlag, New York. 35

Kang, S., Woo, C., Hwang, H. and Woo, K. B.: 2000, Evolutionary design of fuzzy rule base for nonlinear system modeling and control, *IEEE Trans. on Fuzzy Systems* **8**(1), 37–45. 19

Karayiannis, N. B. and Bezdek, J. C.: 1997, An integrated approach to fuzzy learning vector quantization and fuzzy c-means clustering, *IEEE Transactions on Fuzzy Systems* **5**(4), 622–628. 16

Kasabov, N.: 2001a, Evolving fuzzy neural network for supervised/unsupervised on-line, knowledge-based learning, *IEEE Trans. on Man, Machine and Cybernetics, Part B: Cybernetics* **31**(6), 902–918. 23

Kasabov, N. K.: 2001b, On-line learning, reasoning, rule extraction and aggregation in locally optimized evolving fuzzy neural networks, *Neurocomputing* pp. 25–45. 145, 183, 189

Kasabov, N., Kim, J., Watts, M. and Gray, A.: 1997, Funn - a fuzzy neural network architecture for adaptive learning and knowledge acquisition in multi-modular distributed environments, *Information Sciences - Applications* **101(3-4)**, 155–175. 2, 23

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P.: 1983, Optimization by simulated annealing, *Science* **220**, 671–680. 18

Klawonn, F. and Keller, A.: 1997, Fuzzy clustering and fuzzy rules, *Proc. 7th International Fuzzy Systems Association World Congress*, Vol. I, Prague, pp. 193–198. 16, 114

Klawonn, F. and Kruse, R.: 1995, Derivation of fuzzy classification rules from multidimensional data, *in* G. E. Lasker and X. Liu (eds), *Advances in Intelligent Data Analysis*, The International Institute for Advanced Studies in Systems Research and Cybernetics, Windsor, Ontario, pp. 90–94. 17

Klir, G. J. and Yuan, B.: 1995, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, NJ. 47, 66

Kosko, B.: 1994, Fuzzy systems as universal approximators, *IEEE Trans. on Computers* **11**(43), 1329–1333. 39, 155

Lee, D. and Kim, M. H.: 2002, Discovering descriptive rules using fuzzy concept hierarchies, *International Journal of Fuzzy Systems* **3**(4), 776–783. 23

Lee, M.-L., Chung, H.-Y. and Yu, F.-M.: 2003, Modeling of hierarchical fuzzy systems, *Fuzzy Sets and Systems* **138**, 343–361. 23

Leski, J. M.: 2001, An ε-insensitive fuzzy c-means clustering, *Applied Mathematics and Computer Science* **11**(4), 993–1007. 16

Levenberg, K.: 1944, A method for the solution of certain problems in least squares, *Quart. Appl. Math.* **2**, 164–168. 18

Li, S. and Elbestawl, M. A.: 1996, Fuzzy clustering for automated tool condition monitoring in machinery, *Mechanical Systems and Signal Processing* **10**(5), 533–550. 18

Luger, G. F. and Stubblefield, W. A.: 1998, *Artificial intelligence: structures and strategies for complex problem-solving*, 3rd edn, Addison Wesley. 55

Luukka, P., Saastamoinen, K. and Könönen, V.: 2001, A classifier based on the maximal fuzzy similarity in the generalized Łukasiewicz-structure, *Proceedings of the FUZZ-IEEE*. 19

Ma, M., Zhang, Y., Langholz, G. and Kandel, A.: 2000, On direct construction of fuzzy systems, *Fuzzy Sets and Systems* **122**, 165–171. 21

Marquardt, D.: 1963, An algorithm for least-squares estimation of nonlinear parameters, *SIAM J. Appl. Math.* **11**, 431–441. 18

Marsala, C.: 1998, Application of fuzzy rule induction to data mining, *Proc. FQAS-98*, Roskilde, Denmark, pp. 260–271. 14, 97

Marsala, C.: 2000, Fuzzy decision trees to help flexible querying, *Kybernetica* **36**(6), 689–705. 14

Marsala, C. and Bouchon-Meunier, B.: 1996, Fuzzy partitioning using mathematical morphology in a learning scheme, *FUZZIEEE*, New Orleans, USA, pp. 1512–1517. 14

Matthews, C. and Jagielska, I.: 1995, Fuzzy rule extraction from a trained multilayer neural network, *IEEE International Conference on Neural Networks*. 23

McAllester, D.: 1990, Truth maintenance, *in* R. Smith and T. Mitchell (eds), *The Eighth National Conference on Artificial Intelligence*, Vol. 2, AAAI Press, Menlo Park, California, pp. 1109–1116. 28

MedlinePlus Medical Encyclopedia: Retrieved from the WWW on 22 December 2004, Septic shock. http://www.nlm.nih.gov/medlineplus/ency/article/000668.htm. 165

Michalski, R., Mozetic, I., Hong, J. and Lavrac, N.: 1986a, The AQ15 inductive learning system: an overview and experiments, *IMAL*, Université de Paris-Sud, Orsay, France. 1, 11, 29, 35, 127, 175

Michalski, R. S.: 1969, On the quasi-minimal solution of the general covering problem, *Proceedings of the V International Symposium on Information Processing*, Vol. A3, Yugoslavia, pp. 125–128. 3, 11, 127, 175

Michalski, R. S.: 1972, A variable-valued logic system as applied to picture description and recognition, *in* F. Nake and A. Rosenfeld (eds), *Graphic Languages*, North Holland. 7, 29, 31

Michalski, R. S.: 1974a, Variable-valued logic: System VL_1, *Proceedings of the 1974 International Symposium on Multiple-Valued Logic*, West Virginia, pp. 323–346. 175

Michalski, R. S.: 1974b, Variable-valued logic: System VL_1, *Proceedings of the 1974 International Symposium on Multiple-Valued Logic*, West Virginia, pp. 323–346. 189

Michalski, R. S., Mozetic, I., Hong, J. and Lavrac, N.: 1986b, The multi-purpose incremental learning system AQ15 and its testing application to three medical domains, *National Conference on Artificial Intelligence, AAAI*, Philadelphia, pp. 11–15. 3, 7, 175

Mitchell, T. M.: 1997, *Machine Learning*, McGraw-Hill. 1, 9, 13, 28, 30, 31, 79, 110, 111, 145, 162, 164, 173

Mitra, S. and Hayashi, Y.: 2000, Neuro-fuzzy rule generation: Survey in soft computing framework, *IEEE Transactions on Neural Networks* **11**(3), 748–768. 23

Mitra, S., Konwar, K. M. and Pal, S. K.: 2002, Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: Generation and evaluation, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **32**, 328–339. 15, 97

Mitra, S. and Pal, S.: 1995, Fuzzy multi-layer perceptron, inferencing and rule generation, *IEEE Transactions on Neural Networks* **6**(1), 51–63. 23

Moon G. Joo, J. S. L.: 2001, Function approximation by hierarchical fuzzy system with constraints on the fuzzy rule, *FUZZIEEE*, pp. 1112–1115. 23

Moré, J.: 1978, The Levenberg-Marquandt algorithm: Implementation and theory, *in* G. Watson (ed.), *Proceedings of the Dundee Conference on Numerical Analysis*, Springer-Verlag. 18

Murthy, S. K.: 1998, Automatic construction of decision trees from data: A multi-disciplinary survey, *Data Mining and Knowledge Discovery* **2**(4), 345–389. 15

Nauck, D. and Kruse, R.: 1993, A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation, *Proc. IEEE Int. Conf. on Neural Networks*, San Francisco, pp. 1022–1027. 23

Nozaki, K., Ishibuchi, H. and Tanaka, H.: 1996, Adaptive fuzzy rule-based classification systems, *IEEE Trans. on Fuzzy Systems* **3**(4), 238–250. 19, 24, 114

Paetz, J.: 2002, *Adaptive Regelgenerierung und ihre Verwendung zur Diagnose des septischen Schocks*, PhD thesis, Institut für Informatik, J.W.G.-Universität, Frankfurt. 165, 166

Paetz, J.: 2003, Knowledge based approach to septic shock patient data using a neural network with trapezoidal activation functions, *Artificial Intelligence in Medicine, Elsevier, Special Issue: Knowledge-Based Neurocomputing in Medicine* **2**(28), 207–230. 165

Pagallo, G. and Hassler, D.: 1990, Boolean feature discovery in empirical learning, *Machine Learning* **5**. 3, 29, 35

Pantel, P. and Lin, D.: 1998, SpamCop: A spam classification & organization program, *Learning for Text Categorization: Papers from the 1998 Workshop*, AAAI Technical Report WS-98-05, Madison, Wisconsin. 164

Peña-Reyes, C. A.: 2003, Incremental fuzzy CoCo: Incremental coevolution of fuzzy systems, *Proceedings of the Third European Symposium on Intelligent Technologies*, pp. 353–362. 19

Peña-Reyes, C. A. and Sipper, M.: 2000, Applying fuzzy CoCo to breast cancer diagnosis, *Proceedings of the 2000 Congress on Evolutionary Computation*, Vol. 2, IEEE Press, Piscataway, NJ., pp. 1168–1175. 19

Peña-Reyes, C. and Sipper, M.: 2001, Fuzzy CoCo: A cooperative-coevolutionary approach to fuzzy modeling, *IEEE Transactions on Fuzzy Systems* **9**(5), 727–737. 2, 19, 112, 114

Pomares, H., Rojas, I., González, J. and Prieto, A.: 2002, Structure identification in complete rule-based fuzzy systems, *IEEE Trans. on Fuzzy Systems* **10**(4), 349–359. 2, 21, 45, 114

Pomares, H., Rojas, I., Ortega, J., González, J. and Prieto, A.: 2000, A systematic approach to a self-generating fuzzy rule-table for function approximation, *IEEE Trans. on Systems, Man, and Cybernetics* **20**, 431–447. 21

Quinlan, J. R.: 1986, Induction of decision trees, *Machine Learning* **1**(1), 81–106. 1, 7, 13, 96, 176

Quinlan, J. R.: 1990, Learning logical definitions from relations, *Machine Learning* **5**, 239–266. 11, 13, 27, 177

Quinlan, J. R.: 1993a, *C4.5: Programs for Machine Learning*, Morgan Kaufmann. 3, 13, 157

Quinlan, J. R.: 1993b, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc. 179

Quinlan, J. R.: 1996a, Bagging, boosting and C4.5, *Thirteenth National Conference on Artificial Intelligence*, AAAI/MIT Press. 153

Quinlan, J. R.: 1996b, Improved use of continuous attributes in C4.5, *Journal of Artificial Intelligence Research* **4**, 77–90. 145, 153

Quinlan, J. R. and Cameron-Jones, R. M.: 1993, FOIL: A midterm report, *Machine Learning: ECML-93, European Conference on Machine Learning, Proceedings*, Vol. 667, Springer-Verlag, pp. 3–20. 11, 13, 178

Quinlan, J. R. and Cameron-Jones, R. M.: 1995a, MDL and categorical theories (continued), *Machine Learning: Proceedings of the Twelfth International Conference*, Morgan Kaufmann Publishers, pp. 464–470. 178

Quinlan, J. R. and Cameron-Jones, R. M.: 1995b, Oversearching and layered search in empirical learning, *IJCAI*, pp. 1019–1024. 75, 147, 153

Rivest, R. L.: 1987, Learning decision lists, *Machine Learning* **2**(3), 229–246. 3

Robbel, P., van Zyl, J. and Cloete, I.: 2004, Rule pruning strategies for fuzzy classifiers, *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom. 158, 173

Roubos, H. and Setnes, M.: 2000, Compact fuzzy models through complexity reduction and evolutionary optimization, *IEEE International Conference on Fuzzy Systems*, Vol. 2, San Antonio, Texas, pp. 762–767. 20

Roubos, J., Setnes, M. and Abonyi, J.: 2000, Learning fuzzy classification rules from data, *RASC Conference*, Leichester, UK. 17

Ruspini, E. H., Bonissone, P. P. and Pedrycz, W. (eds): 1998, *Handbook of Fuzzy Computation*, Institute of Physics Publishing, Bristol. 47

Safavian, S. R. and Landgrebe, D.: 1991, A survey of decision tree classifier methodology, *IEEE Trans. on Systems, Man and Cybernetics* **21**(3), 660–674. 15

Sahami, M., Dumais, S., Heckerman, D. and Horvitz, E.: 1998, A bayesian approach to filtering junk E-mail, *Learning for Text Categorization: Papers from the 1998 Workshop*, AAAI Technical Report WS-98-05, Madison, Wisconsin. 164

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. and Stamatopoulos, P.: 2003, A memory-based approach to anti-spam filtering for mailing lists, *Information Retrieval* **6**, 49–73. 165

Schneider, K.-M.: 2003, A comparison of event models for naive bayes anti-spam e-mail filtering, *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*. 164

Setnes, M.: 2000, Supervised fuzzy clustering for rule extraction, *IEEE Trans. on Fuzzy Systems* **8**(4), 416–424. 8, 16, 17, 114

Setnes, M., Babuska, R. and Verbruggen, H. B.: 1998, Rule-based modeling: precision and transparency, *IEEE Trans. on Systems, Man and Cybernetics, Part C* **28**(1), 165–169. 16

Shieh, J. S., Linkens, D. A. and Peacock, J. E.: 1999, Hierarchical rule-based and self-organizing fuzzy logic control of anaesthesia, *IEEE Trans. on Systems, Man and Cybernetics, Part C* **29**(1), 98–109. 23

Singal, P. K., Mitra, S. and Pal, S. K.: 2001, Incorporation of fuzziness in ID3 and generation of network architecture, *Neural Computing and Applications* **10**, 155–164. 15

Snyders, S. and Omlin, C. W.: 2000, What inductive bias gives good neural network training performance?, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, Vol. 3, pp. 445–450. 196

Störr, H. P.: 2002, A compact fuzzy extension of the naive bayesian classification algorithm, *International Conference on Intelligent Technologies and Third Vietnam-Japan Symposium on Fuzzy Systems and Applications*, pp. 172–177. 28

Stutz, C. and Runkler, T. A.: 2002, Classification and prediction of road traffic using application-specific fuzzy clustering, *IEEE Trans. on Fuzzy Systems* **10**(3), 297–308. 18

Sugeno, M. and Yasukawa, T.: 1993, A fuzzy logic-based approach to qualitative modeling, *IEEE Trans. on Fuzzy Systems* **1**(1), 7–32. 16, 114

Surmann, H.: 2000, Learning a fuzzy rule based knowledge representation, *Proc. of 2. ICSC Symp. on Neural Computation*, Berlin, pp. 349–355. 19, 20, 183

Takagi, T. and Sugeno, M.: 1985, Fuzzy identification of systems and its application to modeling and control, *IEEE Transactions on Systems, Man and Cybernetics* . 8

Theron, H.: 1993, *Specialization by exclusion: an approach to concept learning*, PhD thesis, University of Stellenbosch. 29

Theron, H. and Cloete, I.: 1996, BEXA: A covering algorithm for learning propositional concept descriptions, *Machine Learning* **24**(1), 5–40. 1, 3, 7, 29, 35, 37, 60, 177

Towell, G. and Shavlik, J.: 1994, Knowledge-based artificial neural networks, *Artificial Intelligence* **70**(1-2), 119–160. 189, 190

Tsang, E. C. C., Wang, X. Z. and Yeung, D. S.: 2000, Improving learning accuracy of fuzzy decision trees by hybrid neural networks, *IEEE Trans. on Fuzzy Systems* **8**(5), 601–614. 15

Tsang, E. C. C., Yeung, D. S. and .Chan, P. P. K.: 2003, Fuzzy support vector machines for solving two-class problems, *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, Xi'an, pp. 1080–1083. 18

van Zyl, J. and Cloete, I.: 2004a, FuzzConRI - a fuzzy conjunctive rule inducer, *Proceedings of the Workshop on Advances in Inductive Rule Learning, ECML*, Pisa, Italy, pp. 194–203. 6, 171

van Zyl, J. and Cloete, I.: 2004b, FuzzyPRISM: a specialization model for the FuzzyBexa framework, *Proceedings of the 5th International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom. 6, 171

van Zyl, J. and Cloete, I.: 2004c, Heuristic functions for learning fuzzy conjunctive rules, *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands. 5, 107, 171

van Zyl, J. and Cloete, I.: 2004d, An inductive algorithm for learning conjunctive fuzzy rules, *International Conference on Machine Learning and Cybernetics*, Shanghai, China, pp. 4181–4188. 6, 171

van Zyl, J. and Cloete, I.: 2004e, Prior knowledge for fuzzy knowledge-based artificial neural networks from fuzzy set covering, *International Joint Conference on Neural Networks*, Budapest, Hungary. 5, 171, 172

van Zyl, J. and Cloete, I.: 2004f, Simultaneous concept learning of fuzzy rules, *Proceedings of the 15th European Conference on Machine Learning*, Vol. 3201 of *Lecture Notes in Artificial Intelligence*, Pisa, Italy, pp. 548–559. 5, 113, 171

van Zyl, J. and Cloete, I.: 2006, Specialization models for a general fuzzy set covering framework, *Fuzzy Sets and Systems* **157**(21), 2787–2808. 6, 171

Wang, C.-H., Liu, J.-F., Hong, T.-P. and Tseng, S.-S.: 1999, A fuzzy inductive learning strategy for modular rules, *Fuzzy Sets and Systems* **103**, 91–105. 9, 48, 114, 134

Wang, C.-H., Liu, J.-F., Hong, T.-P. and Tseng, S.-S.: 2003, Fuzzy inductive learning strategies, *Applied Intelligence* **18**, 179–193. 11, 42, 113, 114, 129

Wang, L. and Mendel, J. M.: 1992, Generating fuzzy rules by learning from examples, *IEEE Trans. on Systems, Man and Cybernetics* **22**(6), 1414–1427. 20, 45, 111, 114, 183

Wang, W. and Bridges, S. M.: 2000, Genetic algorithm optimization of membership functions for mining fuzzy association rules, *Proceedings of the 7th International Conference on Fuzzy Theory and Technology*, Atlantic City, pp. 131–134. 19, 112

Webb, G.: 1993, Systematic search for categorical attribute-value data-driven machine learning, *Proceedings Sixth Australian Joint conference of Artificial Intelligence*, Singapore: World Scientific, Melbourne, pp. 342–347. 145, 147

Whigham, P. A.: 1995, Inductive bias and genetic programming, *in* A. M. S. Zalzala (ed.), *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, GALESIA*, Vol. 414, IEE, Sheffield, UK, pp. 461–466. 111

Witten, I. H. and Frank, E.: 2000, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers Inc., San Francisco, CA. 158, 181

Yager, R. R.: 1998, On the construction of hierarchical fuzzy systems models, *IEEE Trans. on Fuzzy Systems* **28**(1), 55–66. 22

Yeung, D. S. and Tsang, E. C. C.: 1997, Weighted fuzzy production rules, *Fuzzy Sets and Systems* **88**, 299–313. 15

Yin, T.: 2004, A characteristic-point-based fuzzy inference system aimed to minimize the number of fuzzy rules, *IEEE Trans. on Fuzzy Systems* **12**(2), 250–273. 17, 114

Yuan, Y. and Shaw, M. J.: 1995, Induction of fuzzy decision trees, *Fuzzy Sets and Systems* **69**, 125–139. 3, 9, 13, 46, 73, 95, 97, 99, 114, 145, 160, 183

Zadeh, L. A.: 1968, Probability measures of fuzzy events, *Journal of Mathematical Analysis and Applications* **23**(2), 421–427. 14

Zeidler, J. and Schlosser, M.: 1995, Fuzzy handling of continuous-valued attributes in decision trees, *Proc. ECML-95*, Heraklion, Crete, Greece, pp. 41–46. 13

Zheru, C. and Hong, Y.: 1996, ID3-derived fuzzy rules and optimized defuzzification for handwritten numeral recognition, *IEEE Trans. on Fuzzy Systems* **4**(1), 24–31. 13, 114