Simultaneous Concept Learning of Fuzzy Rules

Jacobus van Zyl and Ian Cloete

School of Information Technology, International University in Germany,
76646 Bruchsal, Germany

Abstract. FuzzyBexa was the first algorithm to use a set covering approach for induction of *fuzzy* classification rules. It followed an iterated concept learning strategy, where rules are induced for each concept in turn. We present a new algorithm to allow also simultaneous concept learning and the induction of ordered fuzzy rule sets. When a proper rule evaluation function is used, simultaneous concept learning performs far better than iterated concept learning with respect to rule set size, rule complexity, search complexity, and classification accuracy. We provide empirical results of five experiments on nine data sets and also show that the algorithm compares favourably to other well known concept learners.

1 Introduction

We recently extended the set covering framework to the fuzzy domain, and presented a new algorithm, FuzzyBexa, that induces fuzzy classification rules [1]. This method compares well with state of the art concept learners. There exists very few fuzzy rule learners that, unlike fuzzy decision trees and fuzzy neural networks, directly induce fuzzy classification rules, and to the best of our knowledge, none that use a fuzzy set covering approach to learning.

The *set covering* approach to concept learning has also been called *sequential learning* [2], because in this approach one rule is learned, the data covered by the rule are removed, and the process iterated, until a disjunctive set of rules for all concepts have been found. Learning multiple concepts generally follow 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. [2], 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 [1, 3–5]. 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 [6, 7].

J.-F. Boulicaut et al. (Eds.): ECML 2004, LNAI 3201, pp. 548–559, 2004.

In this paper we introduce FuzzyBexaII, that uses SCL to induce classification rules from fuzzy data. The induction process produces an ordered rule set, and we show that in many cases this methodology produces superior results to FuzzyBexa, i.e. on average better classification performance, radically smaller rule sets, and also less search complexity. We also introduce the fuzzy accuracy function for rule evaluation in SCL, and demonstrate that this function is much better behaved for SCL learning than, for example, the fuzzy entropy function. In the next section we review fuzzy set covering and FuzzyBexa. In Section 3 we show how to extend FuzzyBexa to use the SCL strategy, and in the following section we show that the rule evaluation function plays a pivotal role in finding good classification rules. In Section 5 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 6 concludes our paper.

2 FuzzyBexa: A Fuzzy Set Covering Framework

Fuzzy instances differ from crisp instances in two fundamental ways. Firstly, fuzzy instances belong to all attributes and attribute values, whereas crisp instances belong to only one attribute value per attribute. Secondly, fuzzy instances belong to attribute values to a certain degree, measured on the scale $[0, 1]$, whereas crisp instances either match an attribute value or not, i.e. their memberships are measured in the set $\{0, 1\}$. The fuzzy instance space is described by a vector of linguistic variables (fuzzy attributes) A_i, $I = \langle A_1, A_2, ..., A_n \rangle$, where each variable is a family of fuzzy sets, $A_i = \langle L_1, L_2, ..., L_m \rangle$. Each L_j is a fuzzy set, i.e. a linguistic term (fuzzy attribute value). Together, all the linguistic terms belonging to the same linguistic variable form the *term set* of that linguistic variable. Linguistic terms and variables are referred to simply as terms and variables. A fuzzy instance belongs to each term to a certain degree, its membership degree. The membership of fuzzy instance i to the term L is given by its membership function to the term, $\mu_L(i)$, and thus a fuzzy instance is of the form, $i = \langle \langle \mu_{11}(i), ... \mu_{1p}(i) \rangle ... \langle \mu_{n1}(i), ... \mu_{nq}(i) \rangle \rangle$, where the first subscript is the variable index and the second the term index of the respective variable. Propositional rules are of the form "IF X THEN Y," where X is called the antecedent and Y the consequent. Contrary to the crisp case, fuzzy instances match antecedents only to a certain degree. Let $\mu_A(i)$ and $\mu_B(i)$ be the membership of a fuzzy instance i, $i \in I$, to the two terms A and B respectively, then the degree to which an instance matches an antecedent is computed using the standard fuzzy operators, $\mu_{A \wedge B}(i) = \mu_{A \cap B}(i)$, $\mu_{A \vee B}(i) = \mu_{A \cup B}(i)$, and $\mu_{\neg A}(i) = \mu_{\bar{A}}(i)$. Instances may belong to antecedents to a very small degree. This may be undesirable, and to avoid such cases we apply an α-cut to the antecedent. The membership of an instance that belongs to an antecedent below the threshold α_a, called the antecedent threshold, is defined to be zero. Only instances that belong to an antecedent with membership α_a or above match the antecedent,

and the antecedent is said to cover the instances. The set of all instances within a set S that is covered by an antecedent c is called the extension of the antecedent in S, and is denoted as $X_S(c)$,

$$X_S(c) = \{i \in S | \mu_c(i) \geq \alpha_a\}. \tag{1}$$

As stated above, fuzzy instances belong to terms only with certain membership, and this is also true for the concept. To prevent instances from belonging to a concept to only a small degree, we define instances to belong to a concept only when their membership to the concept is above a_c, the concept threshold.

FuzzyBexa's description language is a fuzzy version of VL_1, called Fuzzy VL_1 [1]. Antecedents in Fuzzy VL_1 are conjunctions of internally disjunctive expressions, also called conjuncts. Each conjunct may contain terms from only one term set. Consider for example the variables *outlook* and *wind* with term sets $\{sunny, cloudy, rainy\}$ and $\{windy, calm\}$, respectively. An example of a Fuzzy VL_1 antecedent is, $[outlook = sunny \vee cloudy] \wedge [wind = calm]$, where the conjuncts are delimited with square brackets. We can also write this expression in short form as $[sunny, cloudy][calm]$, where the conjunction symbol is dropped and the disjunction is replaced by a list of its elements.

FuzzyBexa performs a top-down general-to-specific search of the hypothesis space starting with the most general conjunction (*mgc*). The most general conjunction must have the property that it covers all instances. Consider the conjunction $[sunny, cloudy, rainy][windy, calm]$. This conjunction cannot in general cover all instances, since there may exist a subset B, $B \subset I$, such that $B = \{i \in I | \mu_{sunny \vee cloudy \vee rainy}(i) < \alpha_a\}$, and B is therefore not covered. To form the *mgc* we add to each term set a new term, called its *alpha complement*. The membership of an instance to this term is defined to be zero when the instance belongs to any other term in the term set with membership α_a or greater, and one when the instance does not belong to any other term in the term set with membership α_a or above. Thus, the *mgc* for the example above is $[sunny, cloudy, rainy, \bar{\alpha}_{outlook}][windy, cloudy, \bar{\alpha}_{wind}]$. This conjunction covers all instances in the instance space, and thus is most general.

FuzzyBexa consists of three layers. The top layer implements the fuzzy set covering approach to rule induction and is called *CoverConcepts*. It receives a training set T of instances and a list of concepts for which to induce classification rules. For each concept con_i the training set is split into a set of positive instances P and a set of negative instances N. To obtain P, we use Eq (1), $P = X_T(con_i)$, and for N use $N = T - P$. Next, FuzzyBexa invokes its middle layer 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. Then all the positive instances covered by this rule are 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. It then continues with the next concept until all the concepts are covered.

FuzzyBexa's middle layer, *FindBestConjunction*, implements a set of search heuristics to guide the search. It first forms the *mgc* as described above, and

then invokes the bottom layer routine to obtain a set of specializations of this conjunction. Each of the conjunctions are 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. Let $M(S,c) = \sum_{i \in X_S(c)} \mu_c(i)$, where S is a set of instances and c a conjunction. In this paper we investigate two functions, a fuzzy evaluation function related to the Laplace estimate, $L(c) = \frac{M(P,c)+.5}{M(T,c)}$, and the fuzzy accuracy function, $A(c) = M(P,c) - M(N,c)$. FuzzyBexa can perform a beam search of the hypothesis space by choosing the *beamwidth* best conjunctions to specialize further. Conjunctions that are consistent, i.e. that cover no negative instances, are removed from the search process. FuzzyBexa keeps the best conjunction found thus far in a variable *bestconj*. If the best conjunction found during the search performs no better than the *mgc*, the result "no useful conjunction found" is returned, otherwise, *bestconj* is returned. The middle layer also employs other search restrictions for efficiency and performance measures not discussed here. For more details about FuzzyBexa, see [1].

FuzzyBexa's bottom layer, called *GenerateSpecializations*, receives a set of conjunctions and returns the set of refinements of these conjunctions. FuzzyBexa can use different specialization models, but for the purposes of this paper we will only consider the exclusion specialization model. The set of terms used to describe conjunction c is called its description set, denoted by $D(c)$. In this set, all the terms are uniquely labeled, e.g. the term *high* of the variable *humidity* and the term *high* of the variable *cost* are different. If necessary, we can rename the terms to *humidity.high* and *cost.high*. Thus, there exists a one-to-one mapping between c and $D(c)$. In the exclusion specialization model, a conjunction is specialized by excluding (removing) one of the terms from its description set, i.e. c_2 is specialized to form c_1 by removing a term from $D(c_2)$, e.g. $[a,b][z]$ specialized forms $[a][z]$ or $[b][z]$. Let c_1 and c_2 be two conjunctions from the set of Fuzzy VL_1 conjunctions for a particular problem. Then $c_1 \preceq c_2$, c_1 *is more specific or equal to* c_2 if $D(c_1) \subseteq D(c_2)$, and $c_1 = c_2$ if $D(c_1) = D(c_2)$. We say $c_1 \prec c_2$, c_1 *is more specific than* c_2 if $c_1 \preceq c_2$ and $c_1 \neq c_2$. Thus, the set C, is partially ordered under the relation $\preceq$, and forms a lattice. Furthermore, if $c_1 \prec c_2$, $D(c_1)$ is formed by excluding one or more terms from $D(c_2)$, and c_1 is therefore a more restrictive description, able to match fewer instances than c_2. In fact, c_1 can cover only a subset of the instances covered by c_2, and $X_I(c_1) \subset X_I(c_2)$. Thus, the extension operator is an order-preserving map from conjunctions to instance sets. FuzzyBexa's exclusion specialization model therefore results in a general-to-specific, top-down search of the lattice of Fuzzy VL_1 descriptions.

3 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 1 shows FuzzyBexaII's *Cover-Concepts* routine. Compared to that of FuzzyBexa, the SCL top layer routine

Table 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

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 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 done by dividing the instances covered by the conjunction into groups according to their class,

$$G_i(ant) = \{d \in X_T(ant) |\ \mu_{concept_i}(d) \geq \alpha_c\}. \tag{2}$$

The sigma count or scalar cardinality of each group is then computed,

$$M(G_i(ant)) = \sum_{d \in G_i(ant)} \mu_{ant}(d) \tag{3}$$

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 stop growth 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

Table 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

Table 3. FuzzyBexaII's specialization model

GenerateSpecializations
Input: Set of instances T, set of conjunctions C
Output: Set of specializations of the conjunctions in C
Initialise the set of *spec* to be empty
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*

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 that it. This test is an adaptation of an approach by Quinlan and Cameron-Jones for the crisp iterated concept rule learner by Webb [8, 3]. 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.

FuzzyBexaII's bottom layer routine *GenerateSpecializations*, shown in Table 3, implements the specialization model. The function of this routine is similar to that of FuzzyBexa, i.e. to obtain a set of refinements of the input set of conjunctions. The routine starts by initialising the set of specializations *spec* to be

empty. Then follows two loops. 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

$$X_T(L) \cap X_T(c) = \emptyset \tag{4}$$

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.

4 The Rule Evaluation Function

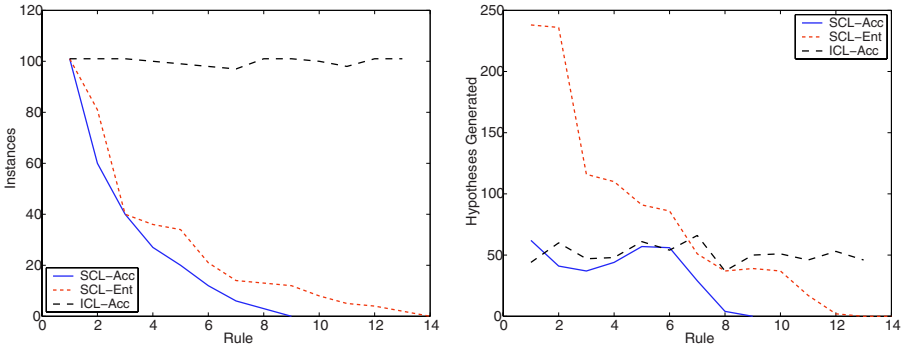
The entropy evaluation function is often used for SCL learning, including decision tree and fuzzy decision tree learning [9, 10]. Let r be a rule with a as antecedent and $\{c_1, ..., c_N\}$ the possible consequents of r, then the normalized fuzzy entropy is given by

$$E(r) = \frac{1}{\log N} \sum_{i=1}^{N} \frac{M(a \wedge c_i)}{M(a)} \log \frac{M(a \wedge c_i)}{M(a)}, \tag{5}$$

where $M(x)$ is the sigma count. 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 of the CN2 algorithm that also used the entropy function [5]. In [11] we suggested the fuzzy accuracy function for ICL,

$$A(r) = \sum_{i \in X_P(a)} \mu_a(i) - \sum_{i \in X_N(a)} \mu_a(i), \tag{6}$$

where P is a subset of T containing all instances that belongs 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 in the highest evaluation, and also assign this evaluation value to the rule. This evaluation function will prefer rules that cover a large number of instances from one concept and few of the other.



(a) Uncovered instances remaining to be used during rule induction

(b) The number of candidate rules generated during rule induction

Fig. 1. Results for ICL and SCL with different evaluation functions on the Zoo data

5 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.

Figure 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 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 reinsert these into the training set when the next concept is considered. SCL, however, never reinserts covered instances, and its graphs are monotonously decreasing. From Figure 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 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 needed only a few hypotheses to cover 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 4 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 are 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 investigates 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, performs 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 compares the size of the rule set induced by each method. Here, SCL-Acc is the clear winner. On average its rule sets contained about three times fewer rules than ICL-Acc and ICL-Ent. It also becomes 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 5 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 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

Table 4. 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

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 1(b), and 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 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

Table 5. SCL-Acc induced rule set for the Zoo data

[milk = true] → type=mammal	[eggs = true][backbone = false][legs = $\neg\bar{\alpha}$]
[feathers = true] → type=bird	→ type=insect
[fins = true] → type=fish	[backbone = true][tail = true] → type=reptile
[eggs = true][breathes = false]	[aquatic = false] → type=invertebrate
→ type=invertebrate	TRUE → type=amphibia

Table 6. 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

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 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 4 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.

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 [8, 12, 4]. The first column shows the average error on the test sets. FuzzyBexaII had similar classification results as the other methods for the 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.

6 Conclusion

This paper 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 big. We further presented five empirical experiments on six data sets, and demonstrated that if the correct kind of evaluation function 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.

References

1. Cloete, I., van Zyl, J.: Fuzzy rule induction in a set covering framework. (2004) (Submitted).
2. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
3. Webb, G.: Systematic search for categorical attribute-value data-driven machine learning. In: Proceedings Sixth Australian Joint conference of Artificial Intelligence, Melbourne, Singapore: World Scientific (1993) 342–347
4. Quinlan, J.R.: Improved use of continuous attributes in C4.5. Journal of Artificial Intelligence Research **4** (1996) 77–90
5. Clark, P., Boswell, R.: Rule induction with CN2: Some recent improvements. In: Proceedings of the Sixth European Working Session on Learning. (1991) 151–163
6. Yuan, Y., Shaw, M.J.: Induction of fuzzy decision trees. Fuzzy Sets and Systems **69** (1995) 125–139
7. Kasabov, N.K.: On-line learning, reasoning, rule extraction and aggregation in locally optimized evolving fuzzy neural networks. Neurocomputing (2001) 25–45
8. Quinlan, J.R., Cameron-Jones, R.M.: Oversearching and layered search in empirical learning. In: IJCAI. (1995) 1019–1024
9. Cios, K.J., Sztandera, L.M.: Continuous id3 algorithm with fuzzy entropy measures. In: Proc. IEEE Int. Conf. Fuzzy Syst. (1992) 469–476
10. Dong, M., Kothari, R.: Look-ahead based fuzzy decision tree induction. IEEE-FS **9** (2001) 461–468
11. Cloete, I., van Zyl, J.: Evaluation function guided search for fuzzy set covering. In: IEEE International Conference on Fuzzy Systems, Budapest, Hungary (2004)
12. Quinlan, J.R.: Bagging, boosting and C4.5. In: Thirteenth National Conference on Artificial Intelligence, AAAI/MIT Press (1996)