Applying Boolean Transformations to Fuzzy Rule Bases

Aljoscha Klose and Andreas Nürnberger
Faculty of Computer Science (FIN-IWS), University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
Tel. +49.391.67-12189, Fax: +49.391.67-12018
E-Mail: aljoscha.klose@cs.uni-magdeburg.de, URL: http://fuzzy.cs.uni-magdeburg.de

ABSTRACT: Neuro-fuzzy classification systems allow to derive fuzzy classifiers by learning from data. The obtained fuzzy rule bases are sometimes hard to interpret, even if the learning method uses constraints to ensure an appropriate fuzzy partitioning of the input domains. This paper describes an approach to build more expressive rules by performing boolean transformations during and after the learning process.

1 Introduction

Fuzzy rules are a well-suited representation of classification knowledge. Their application is rather intuitive and understandable. The abstraction from figures to linguistic variables eases the readability and interpretability of the rule base. Automatic induction of fuzzy rules from data is therefore an interesting topic. One approach to automatic rule generation is based on neuro-fuzzy systems, that use learning algorithms derived from neural network theory to generate fuzzy rules [1, 6, 10]. Unfortunately, the result of automatic rule induction is not always as easy to interpret as a rule base built by human experts. In this paper we discuss some of the problems and approaches to improve the interpretability of neuro-fuzzy classification systems. We present strategies to simplify fuzzy rule bases to improve the interpretability for the user.

For the implementation and analysis of the presented approach we used an implementation of the NEFCLASS (**NE**uro **F**uzzy **CLASS**ification) model [7], since it was designed as an interactive classification tool. One of our goals is to develop algorithms that can learn automatically, but also allow the user to influence the learning and classification process, e.g. by initializing the system, and modifying or extracting knowledge.

The next section gives a brief introduction to fuzzy rules, neuro-fuzzy rule generation and occurring problems. Sect. 2 describes our approach to address some of these problems. In Sect. 3 the impacts of the changes of the rules on the fuzzy classification are considered. Experimental evaluation and conclusions are given in Sect. 4 and 5.

1.1 Inducing Fuzzy Rules

In the following, we consider fuzzy rules in the usual form

> **if** x_1 *is* μ_1 **and** x_2 *is* μ_2 **and** ... **and** x_n *is* μ_n **then** *pattern* $(x_1, x_2, \ldots, x_n)$ *belongs to class* i,

where the $\mu_1, \ldots, \mu_n$ are fuzzy sets. The degree of fulfillment of a rule is calculated from the membership degrees of the antecedents by use of a t-norm, usually $\top_{min}$ or $\top_{prod}$. For each class the maximum or the weighted average of the rule activations is calculated. The output class is determined by a winner-takes-all principle, i.e. the class with the highest accumulated activation is predicted.

Rule induction in systems like NEFCLASS, that start with a fixed number p_d of manually defined or equally spaced fuzzy sets as partitioning for each dimension d, is pretty straight-forward: for each example from the database a rule is constructed by choosing for each dimension the fuzzy set with the highest membership degree as antecedent. The consequent of a rule is determined from the classes of all patterns covered by this rule. In NEFCLASS a measure is used that takes the alignment of pattern and rule center into account. The initial

Acknowledgment: The research presented in this paper is partly funded by DFG contract KR 521/3-2.

fuzzy partitioning of the input dimensions, i.e. the parameters of the fuzzy sets, is usually adapted in a separate learning phase.

As the number of antecedents initially matches the dimensionality of the input space, the number of possible different rules grows exponentially with the number of dimensions. The number of generated rules is bounded by the number of training examples. Thus, commonly observed problems are either a huge number of rules (with low readability and interpretability) or a sparse covering of the input space (sometimes resulting in insufficient generalization). Both make pruning an important step in rule induction.

1.2 Pruning Fuzzy Rule Bases

Pruning of fuzzy rule bases can be done in several ways. Two basic approaches can be separated: direct deletion vs. expanding of the rules. Approaches of the first kind seek to identify by some measure the less important rules and remove them. As we deal with fuzzy rules, the remaining rules can cover more of the free space and eventually grow. The latter approach sees the fuzzy rules as hyper boxes in input space[1]. The approaches seek to expand the (hyper)volume of the boxes by merging adjacent rules. Thus, multiple rules are represented by one merger and the number of rules is decreased.

One efficient way of increasing the size of fuzzy rules is the removal of superfluous antecedents from the rules. This can also be interpreted as setting the according dimension to the linguistic term *don't care*. Thus the corresponding hyper box covers the whole input space in that particular dimension. Setting input dimensions to *don't care* is a special case of merging adjacent rules, i.e. merging the rules that are equal in all but one dimension and use all possible linguistic terms of the fuzzy partition in the remaining dimension. E.g. if one dimension is partitioned by 3 fuzzy sets, the disjunctive term *small OR medium OR large* can be interpreted as a *don't care*.

This search for superfluous antecedents using an approach based on the minimal description length principle (MDL) has been a field of interest of our previous research [3]. We used the (crisp) statistic class distributions that result from the fuzzy classification rules as basis for the MDL calculation.

2 Merging on Logical Level

The approach in this paper can be seen as a third pruning step after rule pruning on dataset and on rule level. However, we simplify the rule base only on a 'logical level', i.e. only the rules are used and the data or their classification are not considered. Concretely, the considered fuzzy rule bases are given in disjunctive normal form (DNF), i.e. each fuzzy rule is a conjunction of its antecedents, and the rule base is a disjunction of its rules. The boolean transformations in our approach allow disjunctions of the fuzzy sets as antecedents of the rules.

Let us, for illustration purposes, assume a rule base consisting of the following two rules

> R_1: *IF x IS small AND y IS large THEN the pattern belongs to class 1* **OR**
>
> R_2: *IF x IS medium AND y IS large THEN the pattern belongs to class 1.*

Then we can merge these rules by using only the rule

> R': *IF x IS small OR medium AND y IS large THEN the pattern belongs to class 1.*

We have to specify how to deal with the disjunction in the antecedent. This is clear in the crisp case, but ambiguous with fuzzy rules. One possibility is to interpret *small OR medium* as a new linguistic term with a new associated membership function. We chose this interpretation, as it allows to represent the new rules in DNF, i.e. we still have the two-layered scheme of conjunctive combination of antecedents and disjunctive combination of rules. Thus, the NEFCLASS inference system needs not to be changed. Apart from this, we can use the simplified rule base only as a more compact output representation and reconstruct the original rules before applying the classifier. As this reconstruction is always possible, the merging can be used as lossless representation. Therefore, we will start with a description of the simplification algorithm in the next section. Considerations on different interpretations of the merged rules are given in Sect. 3.

[1] This holds for the outer borders of single fuzzy rules. The membership degrees increase from 0 at these outer borders to 1 at the cluster center. Therefore multiple overlapping rules can represent other, more complex shapes. For more detailed descriptions and a discussion of sparse vs. complete rule bases see [9].

2.1 The Algorithm of Quine-McCluskey

The simplification of the rule base is basically similar to the simplification of boolean functions. There are various algorithms to simplify VLSI circuits. They can be divided whether they find optimal solutions or apply heuristics. For our problem the basic Quine-McCluskey algorithm is well suited [4], which will be briefly presented in the following. A detailed description of this algorithm as well as of several extended versions can be found in [2]. These algorithms are designed to deal with (possibly incompletely specified) boolean functions, i.e. functions $f : \{0,1\}^n \rightarrow 0, 1, DC$, where DC means *don't care* and marks the unspecified values of f. Each input-output-pair is called a minterm.

The central element of the Quine-McCluskey algorithm is a so-called *prime implicant table*. This is generated from the minterms of the boolean function and allows efficient simplification afterwards.

Starting point of the algorithm are the minterms. In a first step all implicants, i.e. all valid mergers of these minterms, are generated. This is easier for binary input variables, as a merger of binary variables always directly removes an antecedent. In case of fuzzy rules with 3 or more linguistic terms per input range, we can still look for consistent mergers, i.e. pairs of rules, that are equal in all but one dimension and adjacent or overlapping in the differing dimension. In a first phase all implicants are generated for the initial rules. An implicant can be directly interpreted as a merged rule and vice versa.

In a second step, all non-prime implicants are removed, i.e. implicants that cover a subset of another implicant. The remaining implicants are stored in the prime implicant table, which has a column for every minterm that has to be covered, and a row for every prime implicant. A field is marked, if the corresponding implicant (row) covers the corresponding minterm (column). The task of the simplification is now to find a minimal subset of prime implicants, that cover all minterms. This simplification is done in a loop of deleting superfluous rows and/or columns and choosing terms that need to be used in a solution:

If an implicant (row) i covers a subset of minterms that an implicant j covers, i is said to be *dominated* by j and thus can be removed. If a minterm (column) i is covered by a subset of implicants that cover a minterm j, j is said to *dominate* i and can be deleted, as no matter which implicant will be chosen to cover minterm i minterm j is also covered. An prime implicant is said to be *essential*, if it is the only possible cover for a certain minterm, i.e. if there is only one marker in that column. All essential prime implicants are immediately added to the solution, the corresponding rows and columns are removed from the prime implicant table.

The prime implicant table is iteratively searched for rows and columns that can be deleted, or rows that are needed to represent the complete rule base. There may be several alternative implicants that can be included in the solution. Such situations are resolved by backtracking, i.e. each possible implicant is recursively tried and the best solution is chosen. The backtracking is bounded by the best solution found so far, where the best solution is that with the least implicants.

According to the original algorithm, which simplifies *ON-set* and *OFF-set* (i.e. the subset of minterms with '1', resp. '0' as consequent) separately, it has to be executed once for each class. The execution time is exponential in the number of minterms. However, due to the direct necessary simplifications in the prime implicant table and the bounded branching, the algorithm can be applied to reasonable numbers of dimensions. An initial reduction of the input dimensionality may thus be necessary and can for example be done by means of the input pruning methods implemented in NEFCLASS [3, 8].

2.2 Filling the Gaps

As fuzzy rules are only generated for the patterns given in the learning dataset, contiguous clusters in input space may be described by non-contiguous rules, i.e. there are gaps between the rules. This becomes especially apparent with higher dimensional domains or less learning patterns. These gaps may cause lower generalization ability and may hinder the logical merging.

On the other hand, fuzzy sets of a partitioning are usually required to be overlapping, and thus adjacent rules fire for patterns lying in between them. That means, that a fuzzy rule also makes predictions for regions adjacent to its 'core'. Thus, gaps between non-adjacent rules are closed by the corresponding fuzzy sets and generalization ability is not necessarily affected.

Still, these gaps may cause unnecessarily complex rules after logical merging. Therefore an explicit filling of the gaps can be applied, where it does not lead to contradictions. Analogous to VLSI these additional rules are marked as DC, i.e. they are used to build implicants, but they do not belong to the set of minterms to be

covered. Thus, they are only used where they lead to simpler mergers.

As criteria, whether a new rule can be added as a DC-rule, its minimal distance to the rules of each class is determined. The distance of two rules is defined as the number of dimensions in which the antecedents are different, but adjacent linguistic terms, or ∞, if they are not adjacent in at least one dimension. A rule may be added as DC-rule for a class c_i, if its minimal distance to a rule of the same class is smaller than to the rules of any class $c_j, j \neq i$, and if this minimal distance does not exceed a threshold given by the user. This threshold allows the user to control, how liberally DC-rules are generated. Sect. 4 empirically shows the influence of the DC-rules.

3 Impacts of the Transformations

When we investigate which impacts the logical rule merging has on the classifier, two aspects can be considered: how the user understands the rule base and how the classification behavior is affected.

Interpreting the Rule Base

As stated before, the given transformations may be done as pure output processing step, i.e. the simplified rule base is presented to the user as result, but the original, unsimplified rule base is (eventually reconstructed from the pruned one) used for classification. One may ask how the pruning of the rule base affects the way the user interprets it: does the new representation help him or her to better understand the data or get a better intuition? The decreased number of merged rules will certainly improve the readability of the rule base. Furthermore the search for rule mergers is quite natural and has already been done manually to present the rules.

If DC-rules are allowed, there may be less and simpler rules after pruning. However, the use of DC-rules affects the reconstruction of the original rule base and thus potentially the classification behavior. On the other hand these changes can be useful in terms of generalization ability. The additional filled input space is limited by the convex hull of the existing rules. As most natural cluster shapes are convex, this may represent the true shape of the data more appropriate.

One thing that needs to be avoided in the previous merging steps [3] are multiple coverings of the original rules by merged rules, as this disturbs the class distributions that are the basis of the simplification algorithm. Such multiple coverings of the minterms are allowed for the logical merging and do not affect the reconstruction of the original rule base.

Changes in Classification Behavior

However, if we do not reconstruct the original rules, but use the merged rules directly for classification, these multiple coverings may have an effect on the classification. However, as in the case of the DC-rules, these effects may be desirable.

To determine the activation of a merged rule for a given pattern we need to define membership functions for the merged linguistic terms. From the boolean transformations we get disjunctions of the basic linguistic terms. This suggests the use of a *t-conorm*, e.g. $\perp_{max}$ or $\perp_{bounded\ sum}$. As always adjacent linguistic terms are merged, it is not intuitive to have non-convex fuzzy sets. Let us for example assume the merged fuzzy set *'small or medium'*, with two underlying triangular fuzzy sets with their peaks at x_{small} and x_{medium}. The membership degree of the mean $x_{mean} = 0.5(x_{small} + x_{medium})$ is definitely less than 1 for $\perp_{max}$ and also for $\perp_{bounded\ sum}$, if the fuzzy sets sum up to less than 1. However, the intuition suggests, that x_{mean} belongs at least as much to the merged fuzzy set as x_{small} or x_{medium}. Thus, it is justified to interpret *'small or medium'* as *'small* **to** *medium'* and associate a membership function which is 1 for $x_{small} \leq x \leq x_{medium}$, otherwise $max\{\mu_{small}(x), \mu_{medium}(x)\}$. Thus, triangular fuzzy sets are merged to trapezoidal ones. The membership functions for other mergers are defined accordingly.

This obviously increases the degrees of membership for patterns lying in between the peaks of merged rules and thus affects classification behavior. However, if a merged rule describes a single cluster in the data, these higher degrees of membership may be beneficial.

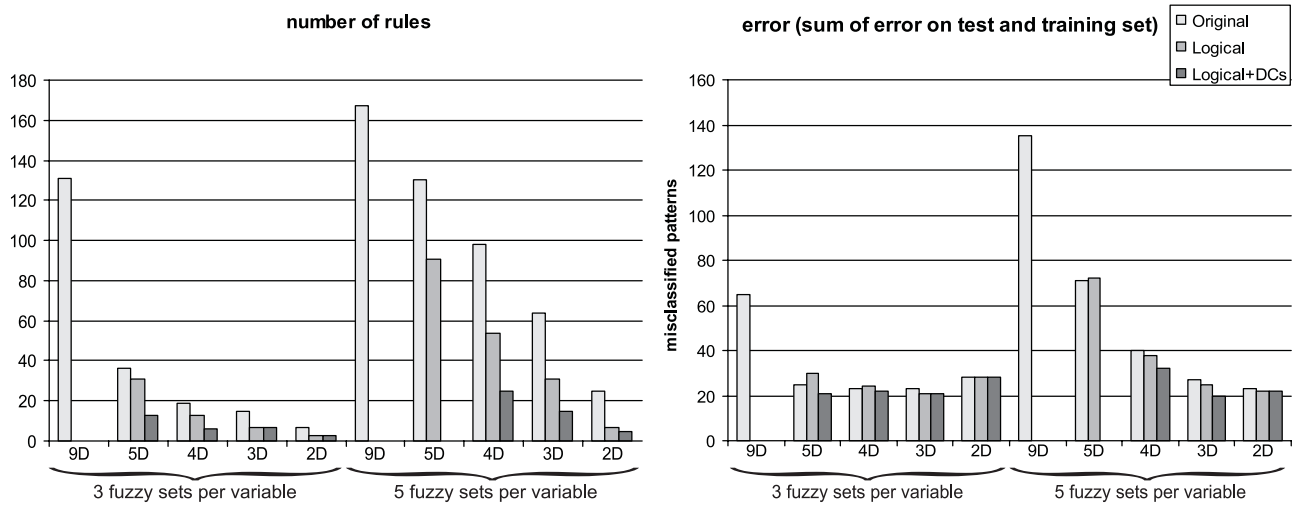

Figure 1: Number of rules and number of misclassifications for different pruning scenarios

4 Empirical Results

To show the effectiveness of our approach we present the results of the application to a dataset from the UCI machine learning repository [5], namely the Wisconsin breast cancer data. This data have 9 input variables. There are 683 patterns (plus 16 patterns with missing values), that are assigned to 2 classes *benign* and *malign*. The data was split into learning (342 patterns) and test data (341 patterns). We initialized NEFCLASS once with 3 and once with 5 fuzzy sets per input variable, resulting in 131 and 167 rules, respectively. On the learning data, the NEFCLASS system with 3 fuzzy sets made 1 misclassification, that with 5 fuzzy sets made no errors at all. Unfortunately, on the test data the results are much worse: 64 and 135 patterns were misclassified – an obvious case of overfitting. Thus, in a first step we pruned the input *on dataset level* [3]. The 'original' bars in Fig. 1 show, how many rules remained and what total error occurred on learn and test set with the number of inputs successively reduced from 9 to 2. The error on the learning set slightly increased to a maximum of 7 misclassifications (3 fuzzy sets, 2 dimensions). However, the total error constantly decreased.

The middle bars ('logical') show the corresponding results after *rule merging on logical level* as proposed in this paper. The pruning runs underlying the right bars ('logical+DCs') additionally allowed the use of *don't care cubes* in the implicant generation. The most obvious result is the reduction of the number of rules in comparison to the 'original' bars. The use of *don't cares* allows a further reduction of the number of rules. Before determining the classification quality, NEFCLASS has been relearned after rule merging. The number of errors is only slightly affected by the rule merging. The classification quality tends to increase on unseen data, especially when *don't cares* are used. This confirms the assumption, that the merged rules with trapezoidal fuzzy sets can better represent the actual structure in the data.

Tab. 1 shows the 5 resulting rules for the NEFCLASS system with 3 inputs and 3 fuzzy sets after logical rule merging. The linguistic term *'small TO large'*, whose membership function is always 1 and which has therefore no influence on the result, has been omitted for better readability.

5 Conclusions

The presented approaches are the consequent continuation of the pruning methods presented in [3], which leads to further improvements in interpretability and readability. As pure output processing step, NEFCLASS can do automatically, what had to be done manually before. Furthermore, when applying the merged rule base, generalization ability can be improved.

The original NEFCLASS software for UNIX (written in C++, with a user interface in TCL/TK) that was used to obtain the results described in this paper can be freely obtained from the World Wide Web (http://fuzzy.cs.uni-magdeburg.de).

IF	input 1	IS small AND
	input 2	IS small AND
	input 6	IS (small OR medium)
THEN	class IS benign	
IF	input 2	IS small AND
	input 6	IS small
THEN	class IS benign	
IF	input 6	IS large
THEN	class IS malign	
IF	input 6	IS (medium OR large) AND
	input 1	IS (medium OR large)
THEN	class IS malign	
IF	input 6	IS small AND
	input 2	IS (medium OR large)
THEN	class IS malign	

Table 1: Resulting rules for breast cancer data set

References

[1] Hamid R. Berenji and Pratap Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Trans. Neural Networks*, 3:724–740, September 1992.

[2] Srinivas Devadas, Abhijit Ghosh, and Kurt Kreutzer. *Logic synthesis*. Series on computer engineering. McGraw Hill, New York, 1994.

[3] Aljoscha Klose, Andreas Nürnberger, and Detlef Nauck. Some approaches to improve the interpretability of neuro-fuzzy classifiers. In *Proc. EUFIT'98*, pages 629–633, 1998.

[4] E. J. McCluskey. Minimization of boolean functions. *Bell Systems Technical Journal*, 35(6):1417–1444, 1956.

[5] C. J. Merz and P. M. Murphy. UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. URL: http://www.ics.uci.edu/~mlearn/-MLRepository.html.

[6] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. *Foundations of Neuro-Fuzzy Systems*. Wiley, Chichester, 1997.

[7] Detlef Nauck and Rudolf Kruse. NEFCLASS – a neuro-fuzzy approach for the classification of data. In K. M. George, Janice H. Carrol, Ed Deaton, Dave Oppenheim, and Jim Hightower, editors, *Applied Computing 1995. Proc. 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28*, pages 461–465. ACM Press, New York, February 1995.

[8] Detlef Nauck and Rudolf Kruse. New learning strategies for NEFCLASS. In *Proc. Seventh International Fuzzy Systems Association World Congress IFSA'97*, volume IV, pages 50–55, Prague, 1997.

[9] Andreas Nürnberger, Aljoscha Klose, and Rudolf Kruse. Discussing cluster shapes of fuzzy classifiers. In *Proc. NAFIPS'99*, 1999.

[10] Nadine Tschichold-Gürman. Generation and improvement of fuzzy classifiers with incremental learning using fuzzy rulenet. In K. M. George, Janice H. Carrol, Ed Deaton, Dave Oppenheim, and Jim Hightower, editors, *Applied Computing 1995. Proc. 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28*, pages 466–470. ACM Press, New York, February 1995.