Learning Complexity-Bounded Rule-Based Classifiers by Combining Association Analysis and Genetic Algorithms

Yu Yi and Eyke Hüllermeier
Institut für Technische und Betriebliche Informationssysteme
Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, Germany

Abstract

We propose a method for learning rule-based classifiers that can be seen as a compromise between a complete enumeration of the hypothesis space and heuristic strategies to search this space in a greedy manner. The method consists of two main parts: In a first step, a sufficiently large number of individual, high-quality classification rules is generated. In a second step, a classifier is assembled from the candidate rules thus obtained. This comes down to selecting a proper subset of these rules, a combinatorial problem that we shall approach by means of genetic algorithms. For the candidate generation step, we suggest using association rule mining. Apart from learning accurate classifiers, a main motivation of our method is the possibility to control the tradeoff between accuracy and transparency (complexity) of a model in a more explicit way.

Keywords: classification, rule induction, association analysis, genetic algorithms

1 Introduction

In machine learning, classification systems induced from empirical data (examples) are first of all rated by their *predictive accuracy.* In practice, however, the *interpretability* or *transparency* of a classifier is often important as well. In this connection, *rule-based classifiers* enjoy great popularity, because rules can easily be understood by human beings. In particular, *fuzzy rules* are appealing in this regard, as they facilitate the representation of a rule in terms of linguistic concepts.

Unfortunately, the potential transparency of rule-based models is often foiled in practice, since accurate classifiers are usually complex at the same time, consisting of many rules with long antecedent parts. In this paper, we propose a novel strategy for learning rule-based classifiers in which the tradeoff between accuracy and interpretability (simplicity) can be controlled in an explicit way. This strategy, which combines association rule mining with genetic search, is detailed in the next section and validated empirically in section 3. Section 4 briefly comments on related work, and section 5 concludes the paper.

2 Learning Rule-Based Classifiers

The process of learning from empirical data can generally be considered as searching a given hypothesis space $\mathcal{H}$ for an optimal hypothesis h^* [12]. Of course, a complete enumeration of $\mathcal{H}$ is impossible if this space is large, as it is the case in most practical applications. In rule induction, $\mathcal{H}$ is indeed huge, since a rule base is in a sense a highly "parameterized" model: Its "parameters" comprise, amongst others, the number of rules, the attributes appearing in the antecedent part of each rule, and the predicates restricting the attributes. The number of potential predicates becomes especially large in the case of numerical attributes, where predicates usually correspond to intervals specifying subsets of the attribute's range (or fuzzy sets in the case of fuzzy rules).

Due to the enormous size of the hypothesis space, methods for rule induction usually employ heuristic strategies in order to search $\mathcal{H}$ in a greedy way. Well-known examples include the divide-and-conquer (recursive partitioning) strategy underlying decision tree induction [14] and the separate-and-conquer (covering) strategy that learns rules in succession, one at a time [6]. On the one hand, approaches of such kind are computationally efficient. On the other hand, due to their heuristic nature, they cannot guarantee optimality or at least near-optimality.

In the following, we shall propose a method that can be seen as a compromise between complete enumeration and greedy search. The key idea is to learn a rule-based classifier in two steps: In the first step, a relatively large number of candidate rules is generated. In the second step, a classifier in the form of a rule base is build by selecting a suitable subset of the candidate rules.

More specifically, we make use of association rule mining in order to generate candidate rules and employ genetic algorithms to construct a classifier from the candidate rules. This strategy can be seen as a compromise between greedy search (adding rules one by one) and complete enumeration of rule bases: The (genetic) search is carried out directly in a space of candidate rule bases, that is, complete classifiers. Yet, since classifiers must be assembled from a limited number of candidate rules, this space is considerably smaller than the space of all potential classifiers.

Despite its higher computational complexity, this approach has several advantages: Firstly, the explicit generation of candidate rules allows one to exclude low-quality rules from the start, which is not as easy in many greedy approaches. Just to give an example, the well-known "small disjuncts" problem [7] can easily be avoided. The latter refers to the problem that, in order to be as accurate as possible, many algorithms that successively split the data have to induce relatively "small", specialized rules that cover only a few examples and, hence, are not very reliable from a statistical point of view.

Secondly, as the search process is more extensive and less restrictive, it should produce classifiers that are more accurate than those produced by simple greedy strategies. A related and perhaps even more important advantage of this approach lies in the flexibility of the objective function that guides the genetic optimization process: By specifying the "fitness" of a classifier in a proper way, it becomes possible to trade off accuracy against other criteria such as interpretability in an explicit and elegant way.

In the remainder of this section, we give a more detailed description of our approach to rule learning, that we shall subsequently refer to as ClAss (Classification with Association Rules).

2.1 Data Preprocessing

As usual, we assume data to be given in the form of a relational table: An example is a tuple (x_i, y_i), where $x_i = (x_{i1} \ldots x_{im}) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_m$ is the feature vector characterizing the i-th instance in terms of m attribute values, and $y_i \in \mathcal{Y}$ is the class label of that instance. Before data of such kind can be submitted as input to a learning algorithm, it must often be preprocessed in various ways. Here, we focus on one particular preprocessing step, namely the discretization of numerical attributes. This step is necessary since association analysis assumes discrete attributes as input. As discretization has an immediate effect of the potential quality of single rules and, hence, complete classifiers, it is also a relatively important step.

Instead of discretizing the data before inducing a model, the discretization can also be integrated into the learning process. Since this latter approach makes less commitments and is more flexible, it often leads to more accurate models. On the other hand, the interpretability of the model usually suffers due to the inability to properly control the partitions that are eventually obtained for the individual attributes. As interpretability is a major issue in our approach, we therefore prefer to pre-partition the data.

In principle, it is thus possible to partition a numerical attribute by hand. In the fuzzy case, for example, a human expert might split the domain into a finite number of partially overlapping fuzzy sets. By defining the membership functions in a proper way, he guarantees that a meaningful linguistic label can be assigned to each fuzzy set. Unfortunately, partitioning by hand can be troublesome and, hence, is not always practicable.

By discretizing the domain $\mathcal{X}_i = [x_i^{\min}, x_i^{\max}] \subseteq \mathfrak{R}$

of a numerical attribute (in a non-fuzzy way) one usually means splitting $\mathcal{X}_i$ into a finite number of intervals $I_1 \ldots I_k$ such that $I_u \cap I_v = \emptyset$ for $u \neq v$ and $I_1 \cup \ldots \cup I_k = \mathcal{X}_i$. A partition of such kind is obviously determined by $k+1$ boundary points $b_0 = x_i^{\min} < b_1 < \ldots < b_{k-1} < b_k = x_i^{\max}$; the j-th interval is then given by $I_j = [b_{j-1}, b_j)$ for $j = 1 \ldots k-1$ and by $I_j = [b_{j-1}, b_j]$ for $j = k$. We note that $x_i^{\min}$ and $x_i^{\max}$ are often unknown in practice and, hence, must be estimated from the data. As one can never be sure that the estimated bounds are valid, the left-most and right-most interval should then be defined by $I_1 = (-\infty, b_1)$ and $I_k = [b_{k-1}, \infty)$, respectively.

In the literature, a large number of discretization methods has been proposed. After having examined many of these methods, we found that the CAIM approach recently proposed in [9] is a favorable choice for our application. CAIM is a *supervised* discretization method, which means that it takes information about the class labels y_j associated with the attribute values x_{ji} into account. In the context of our application, this is of course reasonable, since above all the partition should support the construction of a good classifier.

As another advantage of CAIM let us mention that it determines an appropriate number of intervals in an automatic way, whereas k must be prespecified by the user in alternative approaches such as, e.g., k-means clustering.

Since we are eventually interested in learning a *fuzzy* classifier, our point of departure should actually be a fuzzy partition for each numerical attribute. Therefore, we "soften" each partition obtained by CAIM in the following way: Each crisp boundary point b_j is replaced by a "boundary region" $[\beta_j^l, \beta_j^u]$, where $\beta_j^l \in [(b_{j-1} + b_j)/2, b_j]$ and $\beta_j^u \in [b_j, (b_j + b_{j+1})/2]$. The interval between β_j^l and β_j^u determines the right resp. left boundary of the $(j-1)$-th and the j-th fuzzy set, the membership functions of which are defined as trapezes.

In order to determine the points β_j^l and β_j^u in an optimal way, we took advantage of the fact that CAIM is based on the maximization of an objective function that can be "fuzzified" in a straightforward way. To this end, one simply replaces cardinalities (number of data points in an interval) by fuzzy cardinalities (sum of membership degrees of points in a fuzzy interval). The bounds β_j^l, β_j^u were then chosen so as to maximize this extended optimization criterion. So far, this strategy is realized in an ad hoc manner, namely by quantizing the ranges of β_j^l, β_j^u and trying out each among the possible combinations. It is worth mentioning that this kind of optimization can be done independently for each boundary region.

2.2 Generating Candidate Rules

Recall that we proceed from a relational scheme consisting of $m+1$ attributes $X_1 \ldots X_m, Y$ with domains $D_1 \ldots D_m, D_{m+1}$, respectively; thanks to the pre-processing step, these domains can now assumed to be discrete. The union of these domains, $\mathcal{D}$, can be considered as a set of fuzzy features, where each feature corresponds to the value of some attribute. For an instance (example) x and a feature $F \in \mathcal{D}$, we denote by $F(x) \in [0,1]$ the degree to which F applies to x. The latter is determined by the fuzzy set associated with the feature (attribute value). For example, if `HEIGHT` is an attribute that can assume the value `tall`, which is modeled by an associated fuzzy set $\mu_{\texttt{tall}}$, then $\mu_{\texttt{tall}}(x)$ is the degree to which an instance (person) x is tall. Note that the features are not "independent" in the sense that two (or more) features may refer to the same attribute (belong to the same domain D_i).

An association rule is a directed association between subsets of features. Formally, it is of the form $\mathcal{A} \rightharpoonup \mathcal{B}$, where $\mathcal{A}, \mathcal{B} \subseteq \mathcal{D}$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$. Intuitively, it is meant to suggest that whenever an instance has all the features in $\mathcal{A}$, it does also have all the features in $\mathcal{B}$. We determine the quality of an association in terms of two measures: The *support* is given by

$$\mathsf{supp}(\mathcal{A} \rightharpoonup \mathcal{B}) \overset{\text{df}}{=} \sum_{i=1}^{n} \mathcal{A}(x_i) \otimes \mathcal{B}(x_i),$$

where $\mathcal{A}(x_i) \overset{\text{df}}{=} \bigotimes_{F \in \mathcal{A}} F(x_i)$ is the degree to which the example x_i does have all of the features in $\mathcal{A}$; here, $\otimes$ is a t-norm that serves as a generalized conjunction. Roughly speaking, the support thus defined corresponds to the number of examples that satisfy both, the antecedent part

$\mathcal{A}$ and the consequent part $\mathcal{B}$ of the association. Hence, it is quite comparable to measures of *coverage* that are often used in rule induction in order to express the generality (statistical support) of a rule. By focusing on sufficiently supported rules, the aformentioned "small disjuncts" problem can be avoided. The second measure is the *confidence* of an association, defined by

$$\mathsf{conf}(\mathcal{A} \rightharpoonup \mathcal{B}) \overset{\mathrm{df}}{=} \frac{\mathsf{supp}(\mathcal{A} \rightharpoonup \mathcal{B})}{\mathsf{supp}(\mathcal{A})}.$$

It can be seen as an estimation of the conditional probability that an example satisfies the conclusion $\mathcal{B}$, given that it satisfies the condition $\mathcal{A}$. A rule $\mathcal{A} \rightharpoonup \mathcal{B}$ is considered "interesting" if it has high support and high confidence.

In recent years, a lot of effort has been devoted to the development of efficient algorithms for association rule mining. The latter usually refers to the problem of finding all associations $\mathcal{A} \rightharpoonup \mathcal{B}$ reaching user-defined minimum support and confidence thresholds. Several of the algorithms proposed in this connection have also been extended to the fuzzy case, i.e., association rule mining involving fuzzy attributes [2].

Our approach is based on a rather straightforward extension of the well-known APRIORI algorithm [1]. In addition, however, two types of constraints on association rules must be taken into consideration. Firstly, we are only interested in rules $\mathcal{A} \rightharpoonup \mathcal{B}$ the consequent part of which is defined by a single class label, i.e., rules of the form $\mathcal{A} \rightharpoonup \{Y\}$ with $Y \in D_{k+1} = \mathcal{Y}$. Secondly, the condition $\mathcal{A} = \{F_1 \ldots F_p\}$ of an association corresponds to a conjunction of antecedents of the form $x_{ij} \in F_k$, where the feature F_k is associated with a (fuzzy) value of the j-th attribute. Therefore, all the features $F_1 \ldots F_p$ should come from *different* attributes.

Besides, we modified the standard algorithm as follows: First, we mine association rules only up to a certain level. That is, we upper-bounded the length of the antecedent part of a rule in order to avoid overly complex rules that can hardly be interpreted. Second, in order to find a fixed number K of candidate rules for each class, we perform a kind of "iterative deepening" APRIORI search, using a decreasing sequence of support thresholds. If a least K rules have been generated, the K most confident ones among them are chosen as candidates. This procedure avoids the problem that APRIORI often becomes inefficient and returns an extremely large number of association rules if the support threshold is too small. Finally, the candidate rules thus obtained are filtered in various ways, e.g. by eliminating redundant and conflicting rules. We omit further details due to reasons of space.

2.3 Building a Classifier

The point of departure in this part of our approach is a list of candidate rules $\mathcal{R}$, namely those rules that have been obtained via association analysis as outlined above. The problem is to build a classifier by determining a suitable subset of these rules. We approach this problem by means of a genetic algorithm that searches the space of potential classifiers in a systematic way. In this connection, a classifier can be encoded in a natural way as a bitstring of length $|\mathcal{R}|$: The entry at position i is 1 if the i-th rule is part of the classifier and 0 if not. Using this encoding of solutions, standard operators for selection, mutation, and recombination can be used.

In order to classify an instance x, a rule base $C \subseteq \mathcal{R}$ is evaluated as follows: A rule $r \in \mathcal{R}$ with antecedent part $\mathcal{A}$ and consequent Y is considered as a vote in favor of the class label Y. The strength of the vote is given by $\mathcal{A}(x) \cdot \mathsf{conf}(r)$, i.e., by the product of the confidence of the rule and the degree to which x satisfies the condition $\mathcal{A}$. The votes in favor of a class label are added up, and the class with the maximal sum is output as a prediction. Ties are broken in favor of prevalent classes; in particular, this means predicting the most frequent class if no rule applies. This inference method is a particular instance of a generic inference scheme presented in [4].

As mentioned before, one of the main advantages of ClAss is its flexibility with regard to the evaluation criteria, i.e., the specification of the objective function that guides the genetic search process. In its current version, ClAss takes two criteria into consideration, namely *accuracy* and *transparency*: The accuracy of a classifier is estimated by the relative frequency of correctly classified examples in the test set. The second criterion, transparency, is of course a rather vague one that can be formalized in various ways. Here, we were especially interested in the simplicity of a classifier, i.e., a classifier should consist of an as small as possible number of rules and, moreover, the rules themselves should have short antecedent parts. Note that the latter aspect is already taken care of in the rule generation step, where candidate rules are mined only up to a certain length.

In order to allow for controlling the complexity of a classifier in an explicit way, we have used a fuzzy upper bound on the number of rules. The latter is specified in terms of the following membership function:

$$B(t) = \begin{cases} 1 & \text{if} \quad t \leq T \\ \max(1 - 4(t/T - 1), 0) & \text{if} \quad t > T \end{cases},$$

where $t = |C|$ is the size of the rule base, and T a (non-fuzzy) upper bound. The fuzzy set B thus defined models the constraint "not much more than T": The membership $B(t)$ starts to decrease as t becomes larger than T and becomes 0 as soon as t exceeds T by more than 25%.

The fitness of a classifier is then defined by

$$\text{ACC}(C) \cdot B(|C|)$$

where $\text{ACC}(C)$ denotes the accuracy of the classifier C as defined above. According to this fitness function, the goal of the genetic algorithm is to find the maximally accurate classifier among those with "not much more than" T rules.

3 Experimental Validation

In order to get a first impression of the practical performance of our method, ClAss, we have performed a number of experiments using benchmark data sets from the UCI repository.[1] For comparison purpose, we used the WEKA[2] implementation of Ripper, a state of the art rule learner [3], and C4.5rules in its standard setting. To guarantee a fair comparison, the same discretized data was used as input for all learners.

For each class label, ClAss first derives $4T$ candidate rules as outlined in section 2.2 (T is the complexity-bound of the classifier). The length of the antecedent parts was bounded by 4. For the genetic search algorithm, the following settings have been used: Ranking selection of parents, elitist selection for environmental selection, 1-point crossover (probability 0.8), and single point random mutation (probability 0.5) [11]. The population size was set to 50 and the maximal number of generations to 1000.

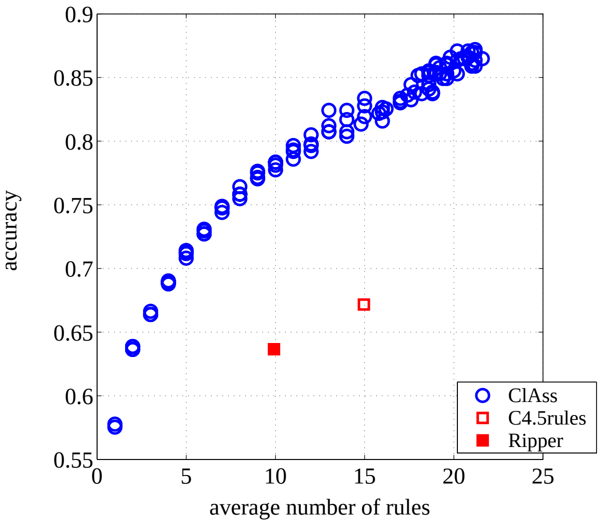

Figure 1: Tradeoff between complexity and accuracy for the CPU data set.

data	Ripper	C4.5rules	ClAss
CPU	.636 ± .052(9.9)	.671 ± .066(14.9)	.781 ± .005(9.9)
Glass	.686 ± .063(7.5)	.672 ± .054(10.9)	.726 ± .008(8.0)
Iris	.941 ± .036(3.2)	.944 ± .034(4.0)	.961 ± .002(3.0)
Liver	.689 ± .043(3.3)	.703 ± .039(3.8)	.701 ± .001(3.0)
thyroid	.940 ± .038(4.6)	.939 ± .039(6.0)	.968 ± .001(5.0)
Pima	.749 ± .033(4.0)	.741 ± .028(11.7)	.751 ± .001(3.5)
Wine	.948 ± .037(4.6)	.918 ± .040(6.2)	.984 ± .002(4.8)

Table 1: Summary of experimental results: average accuracy±std(average number of rules).

We have applied ClAss to each data set with different thresholds T. Each experiment, carried out as a 5-fold cross-validation, yields a classifier with a certain accuracy $\text{ACC}(C)$ and complexity $t = |C|$ (possibly $\neq T$). The set of tuples $(\text{ACC}(C), |C|)$ thus obtained can be seen as an approximation to a corresponding Pareto front. The latter is shown for the CPU data set in Fig. 1, together with the result for Ripper.

The results for a number of other data sets are shown in Table 1. This table compares the accuracies of the three learners, with the complexity

[1] http://www.ics.uci.edu/~mlearn

[2] http://www.cs.waikato.ac.nz/~ml/weka/

bound of ClAss set to the number of rules generated by Ripper. As can be seen, ClAss significantly outperforms Ripper and C4.5rules on most data sets.

4 Related Work

Even though space restrictions prevent from a thorough review of related work, let us briefly mention two approaches that are closely related to our method. In fact, the idea of learning a classifier in two steps, as done by ClAss, is not completely new but has already been suggested in [10]. However, in that approach, called CBA, a classifier is assembled in a quite different way, using a simple greedy strategy. Moreover, no special attention is payed to the model complexity.

Another approach closely related to ClAss is the MOGA system that has recently been introduced in [8]. This approach also makes use of a genetic algorithm in order to assemble a classifier. However, as the authors employ a multi-objective variant in order to approximate the Pareto front in the accuracy–complexity space, it is not possible to control the complexity in an explicit way as in ClAss. Moreover, in the rule generation phase, all potential classification rules are considered, which is of course inefficient and becomes impractical in the case of a large number of attributes.

Finally, our approach is of course related to more traditional genetic fuzzy systems [5]. For example, since a single chromosome in ClAss represents an entire rule base, it is to some extent comparable with the well-known Pittsburgh approach to learning fuzzy rule-based systems [13].

5 Concluding Remarks

An important motivation underlying our approach, ClAss, is the possibility to trade off accuracy against complexity and, in particular, to control the complexity of the induced model in an explicit way. In this regard, our first empirical results are rather promising. Still, there is of course scope for further development. For example, both phases of the algorithm, association rule mining and genetic search, can be optimized in various ways. Moreover, we currently think about replacing association analysis by alternative methods for generating candidate rules such as, e.g., modified versions of standard rule induction algorithms. Another important topic of future work is a more thorough elaboration of the pros and cons of ClAss in comparison with the methods briefly touched on in section 4.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc.* VLDB, pages 487–499, Santiago, Chile, 1994.

[2] G. Chen, Q. Wei, E. Kerre, and G. Wets. Overview of fuzzy associations mining. *Proc. ISIS–2003*. Jeju, Korea, September 2003.

[3] W.W. Cohen. Fast effective rule induction. *Proc. ICML*, pages 115–123, Tahoe City, CA, 1995.

[4] O. Cordon, MJ. del Jesus, and F. Herrera. Analyzing the reasoning mechanisms in fuzzy rule based classification systems. *Mathware & Soft Computing*, 5:321–332, 1998.

[5] O. Cordon, F. Gomide, F. Herrera, and F. Hoffmann anf L. Magdalena. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Syst.*, 141(1):5–31, 2004.

[6] J Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[7] RC. Holte, L. Acker, and B. Porter. Concept learning and the problem with small disjuncts. In *Proceedings IJCAI–89*, pages 813–818, 1989.

[8] H. Ishibuchi and T. Yamamoto. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets and Systems*, 141(1):59–88, 2004.

[9] LA. Kurgan and J. Cios. CAIM Discretization Algorithm. *IEEE Trans. on Data and Knowledge Engin.*, 16(2), 145–153, 2004.

[10] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings KDD–98*, pages 80–86, New York, 1998.

[11] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 1998.

[12] TM. Mitchell. Generalization as search. *Journal of Artificial Intelligence*, 18:203–226, 1982.

[13] SF. Smith. A learning system based on genetic adaptive algorithms. PhD Thesis, Dept. of Comp. Science. Univ. of Pittsburgh, 1980.

[14] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.