Local Attribute Value Grouping for Lazy Rule Induction

Grzegorz Góra and Arkadiusz Wojna

Institute of Informatics, Warsaw University
ul. Banacha 2, 02-097 Warszawa, Poland
{ggora,wojna}@mimuw.edu.pl

Abstract. We present an extension of the lazy rule induction algorithm from [1]. We extended it to deal with real-value attributes and generalised its conditions for symbolic non-ordered attributes. The conditions for symbolic attributes are defined by means of a metric over attribute domain. We show that commonly used rules are a special case of the proposed rules with a specific metric. We also relate the proposed algorithm to the discretisation problem. We illustrate that lazy approach can omit the discretisation time complexity.

1 Introduction

One of the main goals of machine learning, knowledge discovery and data mining is to induce the description of a target concept from its instances. The instances can be represented, e.g., in the form of a decision table. Objects from a decision table are represented by values of some features, also called attributes.

In order to obtain an approximation of a concept with good quality, searching for relevant primitive concepts is required. Feature extraction is a problem of searching for relevant primitive concepts. Two important cases of a feature extraction problem are those of presence of real value attributes and symbolic (nominal) value attributes with large cardinality of domain. In these cases we look for new features like $age \in [30, 50]$ and $color \in \{green, yellow, blue\}$ respectively.

A well known approach in machine learning is the lazy learning (see e.g. [4], [1]). We base our research on the lazy rule induction algorithm presented in [1]. It classifies objects equivalently to the algorithm considering all minimal decision rules, i.e., the most general rules consistent with training examples. We propose extension of this algorithm to deal with real-value attributes and generalisation for symbolic non-ordered attributes. In the latter case we propose descriptors for symbolic attributes grouping the values of the attribute domain. For both kinds of attributes the partition of attribute domains is made locally for a tested object.

Our algorithm is related to the problem of discretisation of the numerical attributes and the methods for grouping attribute values (see e.g. [8], [9]). Our approach does not require a previous discretisation. A similar approach for numerical attributes was presented in [6]. However, in our approach discretisation

J.J. Alpigini et al. (Eds.): RSCTC 2002, LNAI 2475, pp. 405–412, 2002.

is done during classification locally for a test example. Also our approach is parameterised by the choice of a metric on non-ordered attributes.

The paper is organised as follows. Section 2 outlines the basics of rule induction and discretisation. In Section 2.3 the algorithm for lazy rule induction is introduced. Our modification of this algorithm is presented in Section 3. Section 4 concludes the paper with a discussion of possible directions for future research.

2 Preliminaries

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table, where U is a finite set of examples. Each example is described by a finite set of attributes (features) $A \cup \{d\}$, i.e. $a : U \to V_a$ for $a \in A \cup \{d\}$, where $d \notin A$ denotes the decision attribute and V_a is the value domain of an attribute a. The domain of a symbolic (discrete-value) attribute is a finite set, while the domain of a numerical (real-value) attribute is an interval. We denote by $Class(v)$ a subset of training examples with a decision v. We also assume that $V_d = \{1, ..., m\}$, where $m = |V_d|$ is finite.

2.1 Minimal Rule Induction

Rule induction algorithms induce decision rules from a training set. A decision rule consists of a conjunction of attribute conditions and a consequent. The commonly used conditions for symbolic attributes are equations $attribute = value$, while for numerical attributes are specified by interval inclusions, e.g.:

$$IF\ (a_1 = 2 \wedge a_3 \in [3; 7] \wedge a_6 = 5)\ THEN\ (d = 1)$$

A rule is said to cover an example, and vice versa the example is said to match it, if all the conditions in the rule are true for the example. The consequent $(d = v)$ denotes a decision value that is assigned to an object if it matches the rule.

From the knowledge discovery perspective, an important problem is to compute a complete set of consistent and minimal decision rules denoted by *MinRules* (see e.g. [10]), i.e. all rules (matched at least by one training example) that are maximally general and consistent with the training set. In order to discover $MinRules$, rough set methods could be used (see e.g. [10]).

The rules induced from training examples are then used to classify objects. For a given test object the subset of rules matched by the object is selected. If the object matches only rules with the same decision, then the decision predicted by those rules is assigned to the example. If the test object matches rules corresponding to different decisions, the conflict has to be resolved (see e.g. [2]). A common approach is to use a measure for conflict resolving. Then the decision with the highest measure value is chosen. In this paper we focus on a commonly used measure, i.e.:

$$Strength(tst, v) = \left| \bigcup_{r \in MatchRules(tst,v)} supportSet(r) \right|, \qquad (1)$$

where v denotes the v-th decision ($v = 1, ..., |V_d|$), tst is a test example, *supportSet(r)* is the set of training examples matching the rule r, $MatchRules(tst, v)$ is the subset of minimal rules $MinRules$, such that the premise is satisfied by tst and the consequent is a decision v. For each decision $Strength$ measure counts the number of training examples that are covered by the minimal rules with the decision matching a test example tst.

The minimal rule induction classifier based on the $Strength$ measure predicts the decision that is most frequent in the set of training examples covered by the rules matched by a test example, i.e.:

$$decision_{MinRules}(tst) = \arg\max_{v \in V_d} Strength(tst, v) .$$

Algorithms for computing all minimal rules ($MinRules$) are very time consuming, especially when the number of training objects or attributes is large. This is due to the fact that the size of the $MinRules$ set can be exponential with respect to the size of the training set. There are also other approaches to induce a set of rules, which cover the input examples using e.g. smallest number of rules (see e.g. [5], [2]). However, we focus in this paper on the *MinRules* set.

2.2 Discretisation and Value Partitioning

When data are described with real-value attributes, they must undergo a process called discretisation (or quantisation), which divides the range of attribute values into intervals. Such intervals form new values for the attribute and, in consequence, allow to reduce the size of the attribute value set.

Let a be a real-value attribute. A cut is defined as a pair (a, c), where $c \in V_a$. A set of cuts over attribute a defines a partition on V_a into sub-intervals. Any set of cuts transforms $\mathbf{A} = (U, A \cup d)$ into a new decision table $\mathbf{A^P} = (U, A^P \cup \{d\})$, where $A^P = \{a^P : a \in A\}$ and $a^P(x) = i \Leftrightarrow a(x) \in [c_i^a, c_{i+1}^a)$ for any $x \in U$ and $i \in \{0, ..., k_a\}$, where k_a is the number of cuts over the attribute a. A set of cuts is said to be consistent with $\mathbf{A}$ if and only if the generalised decisions of $\mathbf{A}$ and $\mathbf{A^P}$ are identical. For more details on discretisation the reader is referred to [8]. Our attention in the paper is focused on the following theorem (see e.g. [8]):

Theorem 1. *The problem of searching for a consistent partition with the minimal number of cuts is NP-hard.*

It shows that the problem of discretisation from the global point of view is a complex task. We will show in the Subsection 3.2 that it is in a sense possible to overcome this problem if one focuses on a local area instead of the whole universe. This is the case with the presented lazy rule induction algorithm.

Sometimes it is also desired to partition not only real-value attributes, but also symbolic non-ordered attributes. Formally the partition over an attribute a is any function $P_a : V_a \to \{1, ..., m_a\}$. There is a similar theorem to the presented above (see e.g. [9]):

Theorem 2. *The problem of searching for a consistent family of partitions with the minimal* $\sum_{a \in A} m_a$ *is NP-hard.*

2.3 Lazy Rule Induction

In the previous section we have discussed an approach based on calculating $MinRules$. Another approach can be based on construction of algorithms that do not require calculation of the decision rule set in advance. These are memory based (lazy concept induction) algorithms. An example of such an algorithm is presented in [1]. Below we briefly describe this algorithm.

Definition 1. *For objects tst, trn we denote by* $rule^H_{tst}(trn)$ *the local rule with decision* $d(trn)$ *and the following conditions* t_i *for each symbolic attribute* a_i*:*

$$t_i = \begin{cases} a_i = a_i(trn) & \text{if } a_i(tst) = a_i(trn) \\ a_i = * & \text{if } a_i(tst) \neq a_i(trn) \end{cases}$$

*where * denotes any value (such a condition is always true).*

The conditions are chosen in such a way that both the training and the test example satisfy the rule and the rule is maximally specific. Please note that it is formed differently then minimal rules that are minimally specific. But, the important thing is that if only such local rule is consistent with the training data then it can be extended to a minimal rule. Thus, we have the following relation between $MinRules$ and local rules (see e.g. [1]):

Proposition 1. *Premise of the* $rule^H_{tst}(trn)$ *implies a premise of a rule from the set* $MinRules$ *if and only if* $rule^H_{tst}(trn)$ *is consistent with a training set.*

This proposition shows that instead of computing the support sets for rules contained in $MinRules$ and covering a new test case, it is sufficient to generate the local rules formed by the test case with all the training examples and then check their consistency against the training set. It is done by the lazy rule induction algorithm (RIA^H) presented below. The function $isConsistent(r, verifySet)$ checks if a local rule r is consistent with a $verifySet$.

Algorithm 3 $RIA^H(tst)$

for *each decision* $v \in V_d$
  $supportSet(v) = \emptyset$
  for *each* $trn \in U$ *with* $d(trn) = v$
    if $isConsistent(rule^H_{tst}(trn), U)$ ***then***
      $supportSet(v) = supportSet(v) \cup \{trn\}$
$RIA^H = \arg\max_{v \in V_d} |supportSet(v)|$

From Proposition 1 it can be concluded that the algorithm RIA^H computes the measure $Strength$ and therefore the results of the mentioned algorithm are equivalent to the results of the algorithm based on calculating $MinRules$ and using the $Strength$ measure as a strategy for conflict resolving (see [1]).

Corollary 1. *For any test object tst,* $RIA^H(tst) = decision_{MinRules}(tst)$.

The time complexity of the RIA^H algorithm for a single test object is $O(n^2)$, where n is the number of objects in training data. For more details related to this algorithm the reader is referred to [1].

3 Lazy Rule Induction with Attribute Value Grouping

Here we present the extension of the algorithm presented in the previous section. The idea is that we want to use more specific conditions forming a local rule instead of the "star" condition in case when attribute values of the examples differ. In a sense star represents the group of all values from the domain of an attribute. Our idea bases on the observation that it is possible to find smaller groups of attribute values that can be more relevant for the classification.

We divide attributes into two groups according to whether domains of the attributes are linearly ordered or not. In the first group there are numerical attributes and some of linearly ordered symbolic attributes. For such attributes we form the condition requiring the attribute to lay between the values of the examples forming a local rule.

Non-ordered attributes are treated differently. For each such attribute we require a metric to be defined (see example in Subsection 3.1). Such a metric should measure the distance between two values belonging to the domain of the attribute. Then we consider the group of values which are described as balls $B_a(c, R) = \{v \in V_a : \delta_a(c, v) \leq R\}$, where $a \in A$ is an attribute, δ_a is a metric related to this attribute, $c \in V_a$ is a center of the ball and R is a radius of the ball. Then one can measure the distance between values of examples and create condition allowing only these attribute values that are close to a test example in terms of the measured distance. Hence, we propose the following generalisation of Definition 1.

Definition 2. *For objects tst, trn we denote by* $rule^{\delta}_{tst}(trn)$ *the local rule with decision* $d(trn)$ *and the following conditions* t_i *for each attribute* a_i*:*

$$t_i = \begin{cases} min \leq a_i \leq max & \text{when } a_i \text{ is linearly ordered} \\ a_i \in B\left(a_i(tst), R_{a_i}\right) & \text{otherwise} \end{cases}$$

where $min = min(a_i(tst), a_i(trn))$, $max = max(a_i(tst), a_i(trn))$,
$R_{a_i} = \delta_{a_i}(a_i(tst), a_i(trn))$ *and* δ_{a_i} *is a measure of attribute value similarity.*

For both kinds of attributes the conditions are chosen in such a way that both the training and the test example satisfy a rule and the conditions are maximally specific. Using this definition one can use Algorithm 3 with local rules $rule^{\delta}_{tst}(trn)$ from Definition 2 instead of $rule_{tst}(trn)$ from Definition 1. This algorithm groups attribute values during the classification and we denote it by RIA^{δ}.

The advantage of the algorithm for lazy rule induction is that it does not require generating rules in advance. Here we have another advantage, i.e. the algorithm deals with numerical attributes without need of discretisation and it groups symbolic attributes without prior searching for global partition.

3.1 Metrics for Attribute Value Grouping and Example

In this section we discuss the variety of the proposed local rules while changing a metric. First, let us consider the case when all attributes are symbolic and

when we use Kronecker delta as a metric for all attributes ($\delta_a^H(v_1, v_2) = 1$ if $v_1 \neq v_2$ and 0 otherwise). This case relates to the Hamming distance between attribute vector values (counting the number of attributes for which examples differ). Please note that in case when $a_i(trn) \neq a_i(tst)$ then $B(a_i(tst), R_{a_i}) = V_{a_i}$ and in case when $a_i(trn) = a_i(tst)$, we get $B(a_i(tst), R_{a_i}) = \{a_i(tst)\}$. Thus, the conditions from Definition 2 are equivalent to the conditions from the Definition 1 when Kronecker metric is used.

But the proposed generalisation of local rules opens a variety of possibilities for grouping the attributes. Let us now present more informative alternative of a metric than Hamming distance, i.e. Simple Value Difference Metric (SVDM):

$$\delta_a^{SVDM}(v_1, v_2) = \sum_{v \in V_d} |P(Class(v)|a = v_1) - P(Class(v)|a = v_2)|^q ,$$

where $v_1, v_1 \in V_a$, $a \in A$ and q is a natural-value parameter ($q = 1, 2, 3, ...$). SVDM considers two symbolic values to be similar if they have similar decision distribution, i.e. if they correlate similarly with the decision. Different variants of this metric have been successfully used previously (see e.g. [3]).

As an example let us consider the following training set and the test example:

Object	Age	Weight	Gender	BloodGroup	Diagn	Object	Age	Weight	Gender	BloodGroup	Diagn
trn_1	35	90	M	A	Sick	trn_5	45	75	M	B	Sick
trn_2	40	65	F	AB	Sick	trn_6	35	70	F	B	Healthy
trn_3	45	68	F	AB	Healthy	trn_7	45	70	M	O	Healthy
trn_4	40	70	M	AB	Healthy	tst	50	72	F	A	?

Age and Weight are numerical while Gender and BloodGroup (BG) are symbolic non-ordered attributes. Let us take SVDM metric for attributes BG and $Gender$. We have $\delta_{BG}^{SVDM}(A, AB) = \left|1 - \frac{1}{3}\right| + \left|0 - \frac{2}{3}\right| = \frac{4}{3}$, $\delta_{BG}^{SVDM}(A, B) = 1$, $\delta_{BG}^{SVDM}(A, O) = 2$. Let us consider $rule_{tst}^{SVDM}(trn_1)$ and $rule_{tst}^{SVDM}(trn_2)$:

$$\text{if } (A \in [35; 50] \wedge W \in [72; 90] \wedge BG \in \{A\}) \text{ then } Diagn = Sick$$

$$\text{if } (A \in [40; 50] \wedge W \in [65; 72] \wedge Gen = F \wedge BG \in \{A, AB, B\}) \text{then } Diagn = Sick$$

The former rule is consistent just because no other object from the training set satisfy the premise of this rule. The latter rule is inconsistent because the object trn_3 satisfies the premise of the rule and has a different decision.

3.2 Relation to Discretisation

In this section we are going to relate the proposed algorithm to the local discretisation (in the area of a tested object). First, we will introduce definitions analogous to the presented in Section 2. Let us consider a decision table with all real-value attributes. We will say that a set of cuts is locally consistent with $\mathbf{A}$ for a case u if $\mathbf{A^P}$ preserves the generalised decision of an object u. The set of cuts is locally irreducible if any proper subset of cuts is not locally consistent.

For each training object we consider all possible consistent and irreducible sets of local cuts. Please note that there are many possible local cuts for a single example and different sets of local cuts are possible for different examples. Every set of local cuts defines a rule. The set of all such rules over the training objects is denoted by $MinRules_{LC}$. This is analogous to the construction of $MinRules$, where each rule is created locally.

We have the following relation between $MinRules_{LC}$ and local rules:

Proposition 2. *For decision tables with real-value attributes the premise of the* $rule^{\delta}_{tst}(trn)$ *implies premise of a rule from the set* $MinRules_{LC}$ *if and only if* $rule^{\delta}_{tst}(trn)$ *is consistent with a training set.*

Proof. If $rule^{\delta}_{tst}(trn)$ is inconsistent then no rule from $MinRules_{LC}$ could be implied by this rule. In other case the local cuts would not preserve the consistency. If $rule^{\delta}_{tst}(trn)$ is consistent then let us maximally lengthen each interval so that consistency is preserved. Such extended rule is contained in the set $MinRules_{LC}$ (from the definition of $MinRules_{LC}$).

This is the proposition analogous to Proposition 1. It shows that instead of computing the support sets for rules contained in $MinRules_{LC}$ and covering a new test case, it is sufficient to generate the local rules $rule^{\delta}$ for all training examples and then check their consistency against the training set. We have also the analogy to the Corollary 1:

Corollary 2. *For any test object tst,* $RIA^{\delta}(tst) = decision_{MinRules_{LC}}(tst)$.

Again it can be concluded that the results of the algorithm RIA^{δ} are equivalent to the results of the algorithm based on calculating $MinRules_{LC}$ and using the *Strength* measure as a strategy for conflict resolving. It shows that in a sense one can overcome the complexity of discretisation problem by using lazy discretisation with lazy rule induction. Finally, let us note that the presented results hold true for a decision table with mixed real-value and symbolic non-ordered attributes with Kronecker delta as a measure of attribute value similarity. In such case the proof of Proposition 2 would be analogous to the presented one.

4 Conclusions and Further Research

We considered lazy rule induction algorithm. We presented local rules that can deal with all types of attributes without prediscretisation of numerical attributes. Moreover, the presented rules group values of symbolic attributes. The kind of grouping depends on the metric used. For the special kind of a metric, i.e. Kronecker delta metric, the proposed local rules coincide with the commonly used rules.

The value grouping of the attributes is made locally, i.e. for each test example different grouping is possible. It is parameterised by a metric used, thus it opens many possibilities of forming rules and opens the field for a range of experiments. Practical verification of the possible classifiers needs further research.

As a good starting step we propose SVDM metric, which gave good results in many applications.

We also showed interpretation of the lazy rule induction algorithm for real-value attributes. We showed analogous proposition known for symbolic attributes.

Further research requires explanation whether a specific distribution of training examples in domain space and the related position of a classified object may influence performance of the created classifier. Also it is interesting how noise in data may influence final results.

Acknowledgements. The authors are grateful to prof. Andrzej Skowron and dr Marcin Szczuka for their useful remarks. This work was supported by grants 8 T11C 009 19 and 8 T11C 025 19 from the Polish National Committee for Scientific Research.

References

1. Bazan, J.G. (1998). *Discovery of decision rules by matching new objects against data tables.* In: L. Polkowski, A. Skowron (eds.), Proceedings of the First International Conference on Rough Sets and Current Trends in Computing (RSCTC-98), Warsaw, Poland, pp. 521–528.
2. Bazan, J.G., Szczuka, M. (2000). *RSES and RSESlib - A Collection of Tools for Rough Set Computations.* In: W. Ziarko, Y. Yao (eds.), Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing (RSCTC-2000), Banf, Canada, pp. 106–113.
3. Biberman, Y. (1994). *A context similarity measure.* Proceedings of the Ninth European Conference on Machine Learning, pp. 49–63, Catania, Italy: Springer-Verlag.
4. Friedman, J.H., Kohavi, R., Yun, Y. (1996). *Lazy Decision Trees.* Proceedings of the Thirteenth National Conference on Artificial Intelligence, Cambridge, pp. 717–724, MA: MIT Press.
5. Grzymala-Busse, J.W. (1992). *LERS – A system for learning from examples based on rough sets.* In: R. Slowinski (Ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory. Kluwer Academic Publishers, Dordrecht, Boston, London, pp. 3–18.
6. Grzymala-Busse, J.W., Stefanowski, J. (1997). *Discretization of numerical attributes by direct use of the LEM2 induction algorithm with interval extension.* Proceedings of the VI Int. Symp. on Intelligent Information Systems, Zakopane, IPI PAN Press, pp. 149–158.
7. Michalski, R.S. (1983). *A theory and methodology of inductive learning.* Artificial Intelligence, 20, pp. 111–161.
8. Nguyen, H. Son and Nguyen, S. Hoa. (1998). *Discretization Methods in Data Mining* In: Polkowski, L., Skowron A. (eds.) Rough sets in knowledge discovery 1 - methodology and applications, Physica-Verlag, Heidelberg, pp. 451–482.
9. Nguyen, S. Hoa (1999). *Regularity analysis and its applications in data mining* Ph.D Dissertation, Warsaw University.
10. Skowron, A. and Rauszer, C. (1992). *The Discernibility Matrices and Functions in Information Systems.* R. Słowiński (ed.), Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory, pp. 331–362, Dordrecht: Kluwer.