Chapter 1

RULE INDUCTION

Jerzy W. Grzymala-Busse
University of Kansas

Abstract This chapter begins with a brief discussion of some problems associated with input data. Then different rule types are defined. Three representative rule induction methods: LEM1, LEM2, and AQ are presented. An idea of a classification system, where rule sets are utilized to classify new cases, is introduced. Methods to evaluate an error rate associated with classification of unseen cases using the rule set are described. Finally, some more advanced methods are listed.

Keywords: Rule induction algorithms LEM1, LEM2, and AQ; LERS data mining system, LERS classification system, rule set types, discriminant rule sets, validation.

1. Introduction

Rule induction is one of the most important techniques of machine learning. Since regularities hidden in data are frequently expressed in terms of rules, rule induction is one of the fundamental tools of data mining at the same time. Usually rules are expressions of the form

$$if\ (attribute-1, value-1)\ and\ (attribute-2, value-2)\ and\ \cdots$$
$$and\ (attribute-n, value-n)\ then\ (decision, value).$$

Some rule induction systems induce more complex rules, in which values of attributes may be expressed by negation of some values or by a value subset of the attribute domain.

Data from which rules are induced are usually presented in a form similar to a table in which *cases* (or *examples*) are *labels* (or *names*) for rows and variables are labeled as *attributes* and a *decision*. We will restrict

our attention to rule induction which belongs to *supervised learning*: all cases are preclassified by an expert. In different words, the decision value is assigned by an expert to each case. Attributes are independent variables and the decision is a dependent variable. A very simple example of such a table is presented as Table 1.1, in which attributes are: *Temperature*, *Headache*, *Weakness*, *Nausea*, and the decision is *Flu*. The set of all cases labeled by the same decision value is called a *concept*. For Table 1.1, case set {1, 2, 4, 5} is a concept of all cases affected by flu (for each case from this set the corresponding value of *Flu* is *yes*).

Table 1.1. An Example of a Dataset.

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	yes	no	no

Note that input data may be affected by errors. An example of such a data set is presented in Table 1.2. The case 7 has value 42.5 for Weakness, an obvious error, since the attribute Weakness is symbolic, with possible values yes and no. Such errors must be corrected before rule induction.

Table 1.2. An Example of an Erroneous Dataset

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	42.5	no	no

Another problem is caused by numerical attributes, for example, Temperature may be represented by real numbers, as in Table 1.3.

Obviously, numerical attributes must be converted into symbolic attributes before or during rule induction. The process of converting numerical attributes into symbolic attributes is called *discretization* (or *quantization*).

Table 1.3. An Example of a Dataset with a Numerical Attribute.

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	41.6	yes	yes	no	yes
2	39.8	yes	no	yes	yes
3	36.8	no	no	no	no
4	37.0	yes	yes	yes	yes
5	38.8	no	yes	no	yes
6	40.2	no	no	no	no
7	36.6	no	yes	no	no

Input data may be incomplete, i.e., some attributes may have missing attribute values, as in Table 1.4, where ? denotes lack of the attribute value (for example, the original value was not recorded or was erased).

Table 1.4. An Example of a Dataset with Missing Attribute Values.

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	?	yes	no	yes	yes
3	normal	no	?	no	no
4	normal	?	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	yes	no	no

Additionally, input data may be inconsistent, i.e., some cases may conflict with each other. Conflicting cases have the same attribute values yet different decision values. An example of an inconsistent data set is presented in Table 1.5. Cases 7 and 8 are conflicting.

In Section 2 a brief discussion of different rule types is presented. In the next section a few representative rule induction algorithms are discussed. Section 4 presents the main application of rule sets, classification systems, which are used to classify new cases on the basis of induced rule sets.

Table 1.5. An Example of an Inconsistent Dataset

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	very_high	yes	yes	no	yes
2	high	yes	no	yes	yes
3	normal	no	no	no	no
4	normal	yes	yes	yes	yes
5	high	no	yes	no	yes
6	high	no	no	no	no
7	normal	no	yes	no	no
8	normal	no	yes	no	yes

2. Types of Rules

A case x is *covered* by a rule r if and only if every condition (attribute-value pair) of r is satisfied by the corresponding attribute value for x. The concept C defined by the right hand side of rule r is *indicated* by r. We say that a concept C is *completely* covered by a rule set R if and only if for every case x from C there exists a rule r from R such that r covers x. A rule set R is *complete* if and only if every concept from the data set is completely covered by R.

A rule r is *consistent* (with the data set) if and only if for every case x covered by r, x is a member of the concept C indicated by r. A rule set R is *consistent* if and only if every rule from R is consistent with the data set.

For example, case 1 from Table 1.1 is covered by the following rule r:

$$(Headache, yes) \ \rightarrow \ (Flu, yes).$$

The rule r indicates concept {1, 2, 4, 5}. Additionally, the concept {1, 2, 4, 5} is not completely covered by a rule set consisting of r, since r covers only cases 1, 2, and 4, but the rule r is consistent with the data set from Table 1.1.

On the other hand, the single rule

$$(Headache, no) \ \rightarrow \ (Flu, no)$$

completely covers the concept {3, 6, 7} in Table 1.1, though this rule is not consistent. The above rule covers cases 3, 5, 6, and 7.

Any of the following two rules:

$$(Headache, yes) \ \& \ (Weakness, yes) \rightarrow \ (Flu, yes)$$

and

$$(Temperature, high) \ \& \ (Headache, yes) \ \rightarrow \ (Flu, yes)$$

is consistent with the data set from Table 1.1, but the concept {1, 2, 4, 5} is not completely covered by the rule set consisting of the above two rules since case 5 is not covered by any rule. The first rule covers cases 1 and 4, the second rule covers case 2.

The most frequent task of rule induction is to induce a rule set R that is consistent and complete. Such a rule set R is called *discriminant* [Michalski, 1983]. For Table 1.1, the rule set consisting of the following four rules:

$$(Headache, yes) \ \rightarrow \ (Flu, yes),$$

$$(Temperature, high) \ \& \ (Weakness, yes) \ \rightarrow \ (Flu, yes),$$

$$(Temperature, normal) \ \& \ (Headache, no) \ \rightarrow \ (Flu, no),$$

$$(Headache, no) \ \& \ (Weakness, no) \ \rightarrow \ (Flu, no).$$

is discriminant.

There are many other types of rules that are used. For example, some rule induction systems induce rule sets consisting of *strong* rules, i.e., rule sets in which every rule covers many cases. Another task is to induce *associative* rules, in which in both sides of a rule, left and right, involved variables are attributes. For Table 1.1, an example of such an associative rule is

$$(Nausea, yes) \ \rightarrow \ (Headache, yes).$$

3. Rule Induction Algorithms

In this section we will assume that input data sets are free of errors, numerical attributes were already discretized, no missing attribute values are present in the input data sets, and that input data sets are consistent.

In general, rule induction algorithms may be categorized as *global* and *local*. In global rule induction algorithms the search space is the set of all attribute values, while in local rule induction algorithms the search space is the set of attribute-value pairs.

There exist many rule induction algorithms, we will discuss only three representative algorithms, all inducing discriminant rule sets. The first is an example of a global rule induction algorithm called LEM1 (Learning from Examples Module version 1).

3.1 LEM1 Algorithm

The algorithm LEM1, a component of the data mining system LERS (Learning from Examples using Rough Sets), is based on some rough set definitions [Pawlak, 1982], [Pawlak, 1991], [Pawlak *et al.*, 1995]. Let B be a nonempty subset of the set A of all attributes. Let U denote the set of all cases. The indiscernibility relation $IND(B)$ is a relation on U defined for $x, y \in U$ by $(x, y) \in IND(B)$ if and only if for both x and y the values for all attributes from B are identical.

The indiscernibility relation $IND(B)$ is an equivalence relation. Equivalence classes of $IND(B)$ are called *elementary sets* of B. For example, for Table 1.1, and B = {Temperature, Headache}, elementary sets of $IND(B)$ are {1}, {2}, {3, 7}, {4}, {5, 6}.

The family of all B-elementary sets will be denoted B^*, for example, in Table 1.1,

$$\{Temperature, Headache\}^* = \{\{1\}, \{2\}, \{3, 7\}, \{4\}, \{5, 6\}\}.$$

For a decision d we say that $\{d\}$ depends on B if and only if $B^* \leq \{d\}^*$. A *global covering* (or *relative reduct*) of $\{d\}$ is a subset B of A such that $\{d\}$ depends on B and B is minimal in A. Thus, global coverings of $\{d\}$ are computed by comparing partitions B^* with $\{d\}^*$. The algorithm to compute a single global covering is presented below.

Algorithm to compute a single global covering
(**input**: the set $A$of all attributes, partition $\{d\}^*$ on U;
output: a single global covering R);
begin
compute partition A^*;
$P := A$;
$R := \emptyset$;
  if $A^* \leq \{d\}^*$
    then
      begin
      for each attribute a in A **do**
        begin
        $Q := P - \{a\}$;
        compute partition Q^*;
        if $Q^* \leq \{d\}^*$ **then** $P := Q$
      end {for}
      $R := P$
    end {then}
end {algorithm}.

On the basis of a global covering rules are computed using the *dropping conditions* technique [Michalski, 1983]. For a rule of the form

$$C_1 \ \& \ C_2 \ \& \ ... \ \& \ C_n \ \rightarrow \ D$$

dropping conditions means scanning the list of all conditions, from the left to the right, with an attempt to drop any condition, checking against the decision table where the simplified rule does not violate consistency of the discriminant description.

For Table 1.1,

$$\{Temperature,\ Headache,\ Weakness,\ Nausea\}^* = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\}\},$$

$$\{Flu\}^* = \{\{1,2,4,5\},\{3,6,7\}\},$$

and

$$\{Temperature,\ Headache,\ Weakness,\ Nausea\}^* \leq \{Flu\}^*.$$

Next we need to check whether

$$\{Headache,\ Weakness,\ Nausea\}^* \leq \{Flu\}^*.$$

This condition is false since

$$\{Headache,\ Weakness,\ Nausea\}^* = \{\{1\},\{2\},\{3,6\},\{4\},\{5,7\}\}.$$

Then we compute

$$\{Temperature,\ Weakness,\ Nausea\}^* = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\}\},$$

We observe that

$$\{Temperature,\ Weakness,\ Nausea\}^* \leq \{Flu\}^*.$$

The next partition to compute is

$$\{Temperature,\ Nausea\}^*,$$

equal to

$$\{\{1\}, \{2\}, \{3, 7\}, \{4\}, \{5, 6\}\},$$

and

$$\{Temperature,\ Nausea\}^* \not\leq \{Flu\}^*.$$

The last step is to compute

$$\{Temperature,\ Weakness\}^*,$$

equal to

$$\{\{1\}, \{2, 6\}, \{3\}, \{4, 7\}, \{5\}\},$$

since

$$\{Temperature,\ Weakness\}^* \not\leq \{Flu\}^*,$$

the total covering is

$$\{Temperature,\ Weakness,\ Nausea\}.$$

The first case from Table 1.1 implies the following preliminary rule

$$(Temperature, very\_high)\ \&\ (Weakness, yes)\ \&\ (Nausea, no) \rightarrow\ (Flu, yes)$$

The above rule covers only the first case. The first condition,

$$(Temperature, very\_high),$$

cannot be dropped since the rule

$$(Weakness, yes)\ \&\ (Nausea, no)\ \rightarrow\ (Flu, yes)$$

covers cases 1 and 7 from different concepts. However, an attempt to drop the next condition, $(Weakness, yes)$ is successful since the rule

$$(Temperature, very\_high)\ \&\ (Nausea, no)\ \rightarrow\ (Flu, yes)$$

covers only case 1. The next possibility, to drop the last condition $(Weakness, yes)$ is successful as well, since the resulting rule

$$(Temperature, very\_high)\ \rightarrow\ (Flu, yes)$$

covers only case 1.

In a similar way the remaining rules are induced. The final rule set, induced by LEM1, is

$(Temperature, very\_high) \rightarrow (Flu, yes)$
$(Nausea, yes) \rightarrow (Flu, yes)$
$(Temperature, high)$ & $(Weakness, yes) \rightarrow (Flu, yes)$
$(Weakness, no)$ & $(Nausea, no) \rightarrow (Flu, no)$
$(Temperature, normal)$ & $(Nausea, no) \rightarrow (Flu, no)$

For Table 1.1, the second global covering is

$$\{Temperature,\ Headache,\ Weakness\}.$$

3.2 LEM2

An idea of blocks of attribute-value pairs is used in the rule induction algorithm LEM2 (Learning from Examples Module, version 2), another component of LERS. The option LEM2 of LERS is most frequently used since—in most cases—it gives better results. LEM2 explores the search space of attribute-value pairs. Its input data file is a lower or upper approximation of a concept (for definitions of lower and upper approximations of a concept see, e.g., [Grzymala-Busse, 1997]), so its input data file is always consistent. In general, LEM2 computes a local covering and then converts it into a rule set. We will quote a few definitions to describe the LEM2 algorithm [Chan and Grzymala-Busse, 1991], [Grzymala-Busse, 1992].

For an attribute-value pair $(a,\ v) = t$, a *block* of t, denoted by $[t]$, is a set of all cases from U such that for attribute a have value v. Let B be a nonempty lower or upper approximation of a concept represented by a decision-value pair (d, w). Set B *depends* on a set T of attribute-value pairs $t = (a, v)$ if and only if

$$\emptyset \neq [T] = \bigcap_{t \in T} [t] \subseteq B.$$

Set T is a *minimal complex* of B if and only if B depends on T and no proper subset T' of T exists such that B depends on T'. Let $\mathcal{T}$ be a nonempty collection of nonempty sets of attribute-value pairs. Then $\mathcal{T}$ is a *local covering* of B if and only if the following conditions are satisfied:

(1) each member T of $\mathcal{T}$ is a minimal complex of B,

(2) $\bigcup_{t \in \mathcal{T}} [T] = B$, and

$\mathcal{T}$ is minimal, i.e., $\mathcal{T}$ has the smallest possible number of members.

The procedure LEM2 is presented below.

Procedure LEM2
(**input**: a set B,
output: a single local covering $\mathcal{T}$ of set B);
begin
$G := B$;
$\mathcal{T} := \emptyset$;
while $G \neq \emptyset$
begin
$T := \emptyset$;
$T(G) := \{t | [t] \cap G \neq \emptyset\}$;
while $T = \emptyset$ **or** $[T] \not\subseteq B$
begin
select a pair $t \in T(G)$ such that $|[t] \cap G|$ is maximum; if a tie occurs, select a pair $t \in T(G)$ with the smallest cardinality of $[t]$;
if another tie occurs, select first pair;
$T := T \cup \{t\}$;
$G := [t] \cap G$;
$T(G) := \{t | [t] \cap G \neq \emptyset\}$;
$T(G) := T(G) - T$;
end {while}
for each $t \in T$ **do**
if $[T - \{t\}] \subseteq$ B **then** $T := T - \{t\}$;
$\mathcal{T} := \mathcal{T} \cup \{T\}$;
$G := B - \cup_{T \in \mathcal{T}} [T]$;
end {while};
for each $T \in \mathcal{T}$ **do**
if $\bigcup_{S \in \mathcal{T} - \{T\}} [S] = B$ **then** $\mathcal{T} := \mathcal{T} - \{T\}$;
end {procedure}.

For a set X, $|X|$ denotes the cardinality of X.
The first step of the algorithm LEM2 is to compute all attribute-value pair blocks. For Table 1.1. these blocks are

$[(Temperature, very\_high)] = \{1\}$,
$[(Temperature, high)] = \{2, 5, 6\}$,
$[(Temperature, normal)] = \{3, 4, 7\}$,
$[(Headache, yes)] = \{1, 2, 4\}$,
$[(Headache, no)] = \{3, 5, 6, 7\}$,
$[(Weakness, yes)] = \{1, 4, 5, 7\}$,

$[(Weakness, no)] = \{2, 3, 6\}$,
$[(Nausea, no)] = \{1, 3, 5, 6, 7\}$,
$[(Nausea, yes)] = \{2, 4\}$.

Let us induce rules for the concept {1, 2, 4, 5}. Hence, $B = G = \{1, 2, 4, 5\}$. The set $T(G)$ of all relevant attribute-value pairs is

$$\{(Temperature,\ very\_high),\ (Temperature,\ high),$$
$$(Temperature,\ normal), (Headache,\ yes),$$
$$(Headache,\ no), (Weakness,\ yes),$$
$$(Weakness,\ no), (Nausea, no),\ (Nausea,\ yes)\}.$$

The next step is to identify attribute-value pairs (a, v) with the largest $|[(a, v)] \cap G|$. For two attribute-value pairs from $T(G)$, $(Headache,\ yes)$ and (*Weakness, yes*), the cardinality of the set $|[(a, v)] \cap G|$ is equal to three. The next criterion is the size of the attribute-value pair block, this size is smaller for $(Headache,\ yes)$ than for $(Weakness,\ yes)$, so we select $(Headache,\ yes)$. Besides, $[(Headache,\ yes)] \subseteq B$, so $(Headache,\ high)$ is the first minimal complex of G.

The new set G is equal to $B - [(Headache,\ yes)] = \{1, 2, 4, 5\} - \{1, 2, 4\} = \{5\}$. A new set $T(G)$ is equal to

$$\{(Temperature,\ high), (Headache,\ no), (Weakness,\ yes), (Nausea, no)\}.$$

This time the first criterion, the largest $|[(a, v)] \cap G|$, identifies all four attribute-value pairs. The second criterion, the size of the attribute-value block, selects $(Temperature,\ high)$. However,

$$[(Temperature,\ high)] = \{2, 5, 6\} \not\subseteq B,$$

so we have to go through an additional iteration of the internal loop. The next candidates are $(Headache,\ no)$ and $(Weakness,\ yes)$, since for both of these attribute-value pairs the sizes of their blocks are equal to four. On the basis of heuristics, we will select $(Headache,\ no)$. But

$$[(Temperature,\ high)] \cap [(Headache, no)] = \{5, 6\} \not\subseteq B = \{1, 2, 4, 5\},$$

so we have to add $(Weakness, yes)]$ as well. This time

$$[(Temperature,\ high)] \cap [(Headache, no)] \cap [(Weakness, yes)] =$$
$$\{5\} \subseteq B = \{1, 2, 4, 5\},$$

so our candidate for a minimal complex is the set

$$\{(Temperature,\ high), (Headache,\ no), (Weakness, yes)\}.$$

We have to run the following part of the LEM2 algorithm:

for each $t \in T$ **do**
  if $[T - \{t\}] \subseteq$ B **then** $T := T - \{t\}$;

As a result, the second minimal complex is identified:

$$\{(Temperature,\ high), (Weakness, yes)\}.$$

Eventually, the local covering of $B = \{1, 2, 4, 5\}$ is the set

$$\{\{(Headache,\ yes)\}, \{(Temperature,\ high), (Weakness, yes)\}\}.$$

The complete rule set, induced by LEM2, is

$(Headache, yes) \rightarrow (Flu, yes)$
$(Temperature, high)$ & $(Weakness, yes) \rightarrow (Flu, yes)$
$(Temperature, normal)$ & $(Headache, no) \rightarrow (Flu, no)$
$(Headache, no)$ & $(Weakness, no) \rightarrow (Flu, no)$

Obviously, in general, rule sets induced by LEM1 differ from rule sets induced by LEM2 from the same data sets.

3.3 AQ

Another rule induction algorithm, developed by R. S. Michalski and his collaborators in the early seventies, is an algorithm called AQ. Many versions of the algorithm have been developed, under different names [Michalski *et al.*, 1986A], [Michalski *et al.*, 1986B].

Let us start by quoting some definitions from [Michalski *et al.*, 1986A], [Michalski *et al.*, 1986B]. Let A be the set of all attributes, $A = \{A_1, A_2, ..., A_k\}$. A *seed* is a member of the concept, i.e., a positive case. A *selector* is an expression that associates a variable (attribute or decision) to a value of the variable, e.g., a negation of value, a disjunction of values, etc. A *complex* is a conjunction of selectors. A *partial star* $G(e|e_1)$ is a set of all complexes describing the seed $e = (x_1, x_2, ..., x_k)$ and not describing a negative case $e_1 = (y_1, y_2, ..., y_k)$. Thus, the complexes of $G(e|e1)$ are conjunctions of selectors of the form $(A_i, \neg y_i)$, for all i such that $x_i \neq y_i$. A *star* $G(e|F)$ is constructed from all partial stars $G(e|e_i)$, for all $e_i \in F$, and by conjuncting these partial stars by each other, using absorption law to eliminate redundancy. For a given concept C, a *cover*

is a disjunction of complexes describing all positive cases from C and not describing any negative cases from $F = U - C$.

The main idea of the AQ algorithm is to generate a cover for each concept by computing stars and selecting from them single complexes to the cover.

For the example from Table 1.1, and concept $C = \{1, 2, 4, 5\}$ described by (Flu, yes), set F of negative cases is equal to 3, 6, 7. A seed is any member of C, say that it is case 1. Then the partial star $G(1|3)$ is equal to

$$\{(Temperature, \neg normal), (Headache, \neg no), (Weakness, \neg no)\}.$$

Obviously, partial star $G(1|3)$ describes negative cases 6 and 7. The partial star $G(1|6)$ equals

$$\{(Temperature, \neg high), (Headache, \neg no), (Weakness, \neg no)\}$$

The conjunct of $G(1|3)$ and $G(1|6)$ is equal to

$$\begin{gathered}\{(Temperature, very\_high),\\ (Temperature, \neg normal)\ \&\ (Headache, \neg no),\\ (Temperature, \neg normal)\ \&\ (Weakness, \neg no),\\ (Temperature, \neg high)\ \&\ (Headache, \neg no),\\ (Headache, \neg no),\\ (Headache, \neg no)\ \&\ (Weakness, \neg no),\\ (Temperature, \neg high)\ \&\ (Weakness, \neg no),\\ (Headache, \neg no)\ \&\ Weakness, \neg no),\\ (Weakness, \neg no)\},\end{gathered}$$

after using the absorption law, this set is reduced to the following set $G(1|\{3, 6\})$:

$$\{(Temperature, very\_high), (Headache \neg no), (Weakness, \neg no)\}.$$

The preceding set describes negative case 7. The partial star $G(1|7)$ is equal to

$$\{(Temperature, \neg normal), Headache, \neg no)\}.$$

The conjunct of $G(1|\{3, 6\})$ and $G(1|7)$ is

$$\{(Temperature, very\_high),$$
$$(Temperature, very\_high)\ \&\ (Headache, \neg no),$$
$$(Temperature, \neg normal)\ \&\ Headache, \neg no),$$
$$(Headache, \neg no),$$
$$(Temperature, \neg normal)\ \&\ (Weakness, \neg no),$$
$$(Headache, \neg no)\ \&\ (Weakness, \neg no)\}.$$

The above set, after using the absorption law, is already a star $G(1|F)$

$$\{(Temperature, very\_high),$$
$$(Headache, \neg no),$$
$$(Temperature, \neg normal)\ \&\ (Weakness, \neg no)\}.$$

The first complex describes only one positive case 1, while the second complex describes three positive cases: 1, 2, and 4. The third complex describes two positive cases: 1 and 5. Therefore, the complex

$$(Headache, \neg no)$$

should be selected to be a member of the star of C. The corresponding rule is

$$(Headache, \neg no)\ \rightarrow\ (Flu, yes).$$

If rules without negation are preferred, the preceding rule may be replaced by the following rule

$$(Headache, yes) \rightarrow (Flu, yes).$$

The next seed is case 5, and the partial star $G(5|3)$ is the following set

$$\{(Temperature, \neg normal), (Weakness, \neg no)\}.$$

The partial star $G(5|3)$ covers cases 6 and 7. Therefore, we compute $G(5|6)$, equal to

$$\{(Weakness, \neg no)\}$$

A conjunct of $G(5|3)$ and $G(5|6)$ is the following set

$$\{(Temperature, \neg normal) \ \& \ (Weakness, \neg no), (Weakness, \neg no)\}$$

After simplification, the set $G(5|\{3, 6\})$ equals

$$\{Weakness, \neg no)\}.$$

The above set covers case 7. The set $G(5|7)$ is equal to

$$\{(Temperature, \neg normal)\}$$

Finally, the partial star $G(5|\{3, 6, 7\})$ is equal to

$$\{(Temperature, \neg normal) \ \& \ (Weakness, \neg no)\},$$

so the second rule describing concept {1, 2, 4, 5} is

$$(Temperature, \neg normal) \ \& \ (Weakness, \neg no) \ \rightarrow \ (Flu, yes).$$

It is not difficult to see that the following rules describe the second concept from Table 1.1:

$$Temperature, \neg high) \ \& \ (Headache, \neg yes) \rightarrow (Flu, no),$$

$$(Headache, \neg yes) \ \& \ (Weakness, \neg yes) \rightarrow (Flu, no).$$

Note that the AQ algorithm demands computing conjuncts of partial stars. In the worst case, time complexity of this computation is $O(n^m)$, where n is the number of attributes and m is the number of cases. The authors of AQ suggest using the parameter MAXSTAR as a method of reducing the computational complexity. According to this suggestion, any set, computed by conjunction of partial stars, is reduced in size if the number of its members is greater than MAXSTAR. Obviously, the quality of the output of the algorithm is reduced as well.

4. Classification Systems

Rule sets, induced from data sets, are used mostly to classify new, unseen cases. Such rule sets may be used in rule-based expert systems.

There is a few existing classification systems, e.g., associated with rule induction systems LERS or AQ. A classification system used in LERS is a modification of the well-known bucket brigade algorithm [Booker *et al.*, 1990], [Holland *et al.*, 1986], [Stefanowski, 2001]. In the rule induction system AQ, the classification system is based on a rule estimate of probability [Michalski *et al.*, 1986A], [Michalski *et al.*, 1986B]. Some

classification systems use a decision list, in which rules are ordered, the first rule that matches the case classifies it [Rivest, 1987]. In this section we will concentrate on a classification system associated with LERS.

The decision to which concept a case belongs is made on the basis of three factors: *strength*, *specificity*, and *support*. These factors are defined as follows: *strength* is the total number of cases correctly classified by the rule during training. *Specificity* is the total number of attribute-value pairs on the left-hand side of the rule. The matching rules with a larger number of attribute-value pairs are considered more specific. The third factor, *support*, is defined as the sum of products of strength and specificity for all matching rules indicating the same concept. The concept C for which the support, i.e., the following expression

$$\sum_{matching\ rules\ r\ describing\ C} Strength(r) * Specificity(r)$$

is the largest is the winner and the case is classified as being a member of C.

In the classification system of LERS, if complete matching is impossible, all partially matching rules are identified. These are rules with at least one attribute-value pair matching the corresponding attribute-value pair of a case. For any partially matching rule r, the additional factor, called *Matching_factor* (r), is computed. Matching_factor (r) is defined as the ratio of the number of matched attribute-value pairs of r with a case to the total number of attribute-value pairs of r. In partial matching, the concept C for which the following expression is the largest

$$\sum_{\substack{partially\ matching \\ rules\ r\ describing\ C}} Matching\_factor(r) * Strength(r) * Specificity(r)$$

is the winner and the case is classified as being a member of C.

5. Validation

The most important performance criterion of rule induction methods is the error rate. A complete discussion on how to evaluate the error rate from a data set is contained in [Weiss and Kulikowski, 1991]. If the number of cases is less than 100, the *leaving-one-out* method is used to estimate the error rate of the rule set. In leaving-one-out, the number of learn-and-test experiments is equal to the number of cases in the data set. During the i-th experiment, the i-th case is removed from the data set, a rule set is induced by the rule induction system from the remaining cases, and the classification of the omitted case by rules produced is recorded. The error rate is computed as

$$\frac{total\ number\ of\ misclassifications}{number\ of\ cases}.$$

On the other hand, if the number of cases in the data set is greater than or equal to 100, the *ten-fold cross-validation* will be used. This technique is similar to leaving-one-out in that it follows the learn-and-test paradigm. In this case, however, all cases are randomly re-ordered, and then a set of all cases is divided into ten mutually disjoint subsets of approximately equal size. For each subset, all remaining cases

are used for training, i.e., for rule induction, while the subset is used for testing. This method is used primarily to save time at the negligible expense of accuracy.

Ten-fold cross validation is commonly accepted as a standard way of validating rule sets. However, using this method twice, with different preliminary random re-ordering of all cases yields—in general—two different estimates for the error rate [Grzymala-Busse, 1997].

For large data sets (at least 1000 cases) a single application of the train-and-test paradigm may be used. This technique is also known as *holdout* [Weiss and Kulikowski, 1991]. Two thirds of cases should be used for training, one third for testing.

6. Advanced Methodology

Some more advanced methods of machine learning in general and rule induction in particular were discussed in [Dietterich, 1997]. Such methods include combining a few rule sets with associated classification systems, created independently, using different algorithms, to classify a new case by taking into account all individual decisions and using some mechanisms to resolve conflicts, e.g., voting. Another important problem is scaling up rule induction algorithms. Yet another important problem is learning from imbalanced data sets [Japkowicz, 2000], where some concepts are extremely small.

References

[Booker *et al.*, 1990] Booker L.B., Goldberg D.E., and Holland J.F. Classifier systems and genetic algorithms. In *Machine Learning. Paradigms and Methods*, Carbonell, J. G. (ed.), The MIT Press, Boston, MA, 1990, 235–282.

[Chan and Grzymala-Busse, 1991] Chan C.C. and Grzymala-Busse J.W. On the attribute redundancy and the learning programs ID3, PRISM, and LEM2. Department of Computer Science, University of Kansas, TR-91-14, December 1991, 20 pp.

[Dietterich, 1997] Dietterich T.G. Machine-learning research. *AI Magazine* 1997: 97–136.

[Grzymala-Busse, 1988] Grzymala-Busse J.W. Knowledge acquisition under uncertainty—A rough set approach. *Journal of Intelligent & Robotic Systems* 1988; **1**: 3–16.

[Grzymala-Busse, 1992] Grzymala-Busse J.W. LERS—A system for learning from examples based on rough sets. In *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory*, ed. by R. Slowinski, Kluwer Academic Publishers, Dordrecht, Boston, London, 1992, 3–18.

[Grzymala-Busse, 1997] Grzymala-Busse J.W. A new version of the rule induction system LERS, *Fundamenta Informaticae* 1997; **31**: 27–39.

[Holland *et al.*, 1986] Holland J.H., Holyoak K.J., and Nisbett R.E. *Induction. Processes of Inference, Learning, and Discovery*, MIT Press, Boston, MA, 1986.

[Japkowicz, 2000] Japkowicz N. Learning from imbalanced data sets: a comparison of various strategies. Learning from Imbalanced Data Sets, AAAI Workshop at the 17th Conference on AI, AAAI-2000, Austin, TX, July 30–31, 2000, 10–17.

[Michalski, 1983] Michalski R.S. A Theory and Methodology of Inductive Learning. In *Machine Learning. An Artificial Intelligence Approach*, Michalski, R. S., J. G. Carbonell and T. M. Mitchell (eds.), Morgan Kauffman, San Mateo, CA, 1983, 83–134.

[Michalski *et al.*, 1986A] Michalski R.S., Mozetic I., Hong J., Lavrac N. The AQ15 inductive learning system: An overview and experiments, Report 1260, Department of Computer Science, University of Illinois at Urbana-Champaign, 1986A.

[Michalski *et al.*, 1986B] Michalski R.S., Mozetic I., Hong J., Lavrac N. The multi-purpose incremental learning system AQ 15 and its testing application to three medical domains. Proc. of the 5th Nat. Conf. on AI, 1986B, 1041–1045.

[Pawlak, 1982] Pawlak Z.: Rough Sets. *International Journal of Computer and Information Sciences* 1982; **11**: 341–356.

[Pawlak, 1991] Pawlak Z. *Rough Sets. Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht, Boston, London, 1991.

[Pawlak *et al.*, 1995] Pawlak Z., Grzymala-Busse J.W., Slowinski R. and Ziarko, W. Rough sets. *Communications of the ACM* 1995; **38**: 88–95.

[Rivest, 1987] Rivest R.L. Learning decision lists. *Machine Learning* 1987; **2**: 229–246.

[Stefanowski, 2001] Stefanowski J. *Algorithms of Decision Rule Induction in Data Mining*. Poznan University of Technology Press, Poznan, Poland, 2001.

[Weiss and Kulikowski, 1991] Weiss S. and Kulikowski C.A. *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*, chapter *How to Estimate the True Performance of a Learning System*, pp. 17–49, San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1991.