Fundamenta Informaticae 31 (1997) 27-39
IOS Press

A New Version of the Rule Induction System LERS

Jerzy W. Grzymala-Busse
Department of Electrical Engineering and Computer Science
University of Kansas
Lawrence, KS 66045, USA
jerzy@eecs.ukans.edu

In memory of professor Helena Rasiowa

Abstract. A new version of the rule induction system LERS is described and compared with the old version of LERS. Experiments were done for comparison of performance for both versions of LERS and the two other rule-induction systems: AQ15 and C4.5. The new LERS system performance is fully comparable with performance of the other two systems.

1. Introduction

The rule induction system, called LERS (Learning from Examples using Rough Sets), has been developed at the University of Kansas. The first implementation of LERS was done in Franz Lisp in 1988 [4]. In the first version of LERS the only algorithm of rule induction was LEM1, a global algorithm to induce all rules from the input data. In 1990 the basic algorithm of LERS, LEM2, was implemented [2, 3, 6]. The algorithm LEM2 uses the more sophisticated idea of a local covering.

The first classification system to classify unseen cases using a rule set induced from training cases was used in system ESEP (Expert System for Environmental Protection) [7]. This classification system was much improved in 1994 [8] by using a modified version of the bucket brigade algorithm [1, 10].

The current version of LERS, implemented in C, is a family of knowledge discovery systems. LERS is equipped with the set of discretization schemes to deal with numerical attributes. Discretization is an art rather than a science, so many methods should be tried for a specific data set. Similarly, a variety of methods may help to handle missing attribute values. On the other hand, LERS uses a unique way to work with inconsistent data, following rough set theory. If a data set is inconsistent, LERS will always compute lower and upper approximations for every concept and then will induce certain and possible rules, respectively. Also, LERS is assisted with tools for rule validation: leaving-one-out, ten-fold cross validation, and hold-out.

The contents of this paper was presented at the Minisemester Logic, Algebra and Computer Science: Helena Rasiowa in Memoriam, Stefan Banach International Mathematical Center, Warsaw, Poland, December 2–22, 1996.

2. Fundamentals of LERS

LERS is a rule induction system, inducing rules from training examples (cases). The input examples are presented in a table, called a decision table. A small example of such a table is presented as Table 1. Note that LERS may process much bigger tables, with hundreds of thousands of rows and hundreds of variables.

< a	a	a	d >
[Clock	Cache	Main_Memory	Quality]
slow	no	small	low
slow	yes	medium	low
slow	yes	medium	medium
fast	no	large	medium
fast	yes	large	high

Table 1 Decision Table

Table 1 represents the input data set in the format of LERS [6]. Table 1 contains four variables: *Clock*, *Cache*, *Main_Memory* and *Quality*. The first three variables are called *attributes*, the last one is called *decision*. Table 1 presents five examples (starting from the third row and ending with the seventh row). The first row of the table, starting from "<", declares which variables are attributes (those denoted by *a*s) and which are decisions (denoted by *d*s). This line ends with ">". The first line may contain the letter x (corresponding variable and variable values will be ignored in all subsequent computations). The second line, starting from "[" and ending with "]", contains the names of all variables.

Any subset of the set of all examples, defined by the same value of the decision is called a *concept*. Table 1 consists of three concepts, defined by (*Quality*, *low*), (*Quality*, *medium*), and (*Quality*, *high*). For example, the concept defined by (*Quality*, *low*) consists of the first two examples.

As the first step of LERS processing of the input data file, LERS checks if the input data file is *consistent* (i.e., if the file does not contain conflicting examples). Table 1 is inconsistent because the second and the third examples are conflicting. For these examples, the values of all three attributes are the same (*slow*, *yes*, *medium*), but the decision values are different, *low* for the second example and *medium* for the third example. If the input data file is inconsistent, LERS computes *lower* and *upper approximations* of all concepts. The ideas of lower and upper approximations are fundamental for rough set theory [5, 13, 14]. For other systems of rule induction, based on rough set theory, see [12, 15, 16, 20].

Let U denote the set of all examples of the decision table and let P denote a nonempty subset of the set Q of all variables, i.e., attributes and decisions. Let P be a subset of A. An indiscernibility relation ρ on U is defined for all $x, y \in U$ by $x \rho y$ if and only if for both x and y the values for all variables from P are identical. Equivalence classes of ρ are called *elementary sets* of P. An equivalence class of ρ containing x is denoted $[x]_P$. Any finite union of elementary sets of P is called a *definable set* in P. Let X be a concept. In general, X is not a definable set in P. However, set X may be approximated by two definable sets in P, the first one is called a *lower approximation* of X in P, denoted by $\underline{P}X$ and defined as follows

$$\{x \in U | [x]_P \subseteq X\}.$$

The second set is called an *upper approximation* of X in P, denoted by $\overline{P}X$ and defined as follows

$$\{x \in U | [x]_P \cap X \neq \emptyset\}.$$

The lower approximation of X in A is the greatest definable set in A, contained in X. The upper approximation of X in A is the least definable set in A containing X. A rough set of X is the family of all subsets of U having the same lower and the same upper approximations of X.

In our example, the lower approximation of the concept defined by (*Quality*, *low*) is the set consisting of the first example. The upper approximation of the same concept is the set consisting of the first three examples. Next, LERS creates a pair of new tables for every concept. The first table is created from the lower approximation of the concept, the second table is created from the upper approximation of the concept. Thus, for Table 1 and concept defined by (*Quality*, *low*), Table 2 presents the former table, Table 3 presents the latter table.

< a	a	a	d >
[Clock	Cache	Main_Memory	Quality]
slow	no	small	low
slow	yes	medium	medium
slow	yes	medium	medium
fast	no	large	medium
fast	yes	large	high

Table 2 Decision Table

< a	a	a	d >
[Clock	Cache	Main_Memory	Quality]
slow	no	small	low
slow	yes	medium	low
slow	yes	medium	low
fast	no	large	medium
fast	yes	large	high

Table 3 Decision Table

Rules induced from the lower approximation of the concept *certainly* describe the concept, so they are called *certain*. On the other hand, rules induced from the upper approximation of the concept describe the concept only *possibly* (or *plausibly*), so they are called *possible* [5].

In general, LERS uses two different approaches to rule induction: one is used in machine learning, the other in knowledge acquisition, see Figure 1. In machine learning, or more specifically, in learning from examples, the usual task is to learn *discriminant description* [11], i.e., to learn the smallest set of minimal rules, describing the concept. To accomplish this goal, i.e., to learn discriminant description, LERS uses two algorithms: LEM1 and LEM2 (LEM1 and LEM2 stand for Learning from Examples Module, version 1 and 2, respectively). Both algorithms are described in [6]. The option LEM2 of LERS is most frequently used since—in most cases—it gives best results. Thus we will quote basic definitions to describe only the LEM2 algorithm.

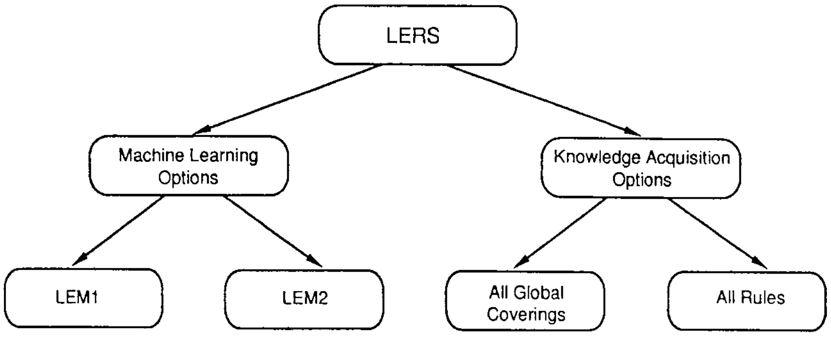

Figure 1. Structure of LERS

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 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. $\bigcap_{t \in \mathcal{T}} [T] = B$, and
3. $\mathcal{T}$ is minimal, i.e., $\mathcal{T}$ has the smallest possible number of members.

The user may select an option of LEM2 with or without taking into account attribute priorities. The procedure LEM2 with attribute priorities is presented below. The other option differs from the one presented below in the selection of a pair $t \in T(G)$ in the inner loop WHILE. When LEM2 is not to take attribute priorities into account, the first criterion is ignored. In our experiments all attribute priorities were equal to each other.

```
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 $t \in T(G)$ with the highest attribute priority,
                if a tie occurs, select a pair $t \in T(G)$ such that $|[t] \cap G|$
                is maximum; if another tie occurs, select a pair $t \in T(G)$
```

```
                    with the smallest cardinality of $[t]$; if a further 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}.
```

From Table 1, LEM2 generates the following rule sets:

certain rule set:

```
2 0.2 0.2 1.0
(Main_Memory, small) -> (Quality, low)
3 0.2 0.2 1.0
(Clock, fast) & (Cache, no) -> (Quality, medium)
3 0.2 0.2 1.0
(Clock, fast) & (Cache, yes) -> (Quality, high)
```

possible rule set:

```
2 0.2 0.6 0.667
(Clock, slow) -> (Quality, low)
2 0.2 0.6 0.5
(Main_Memory, medium) -> (Quality, medium)
3 0.2 0.6 1.0
(Clock, fast) & (Cache, no) -> (Quality, medium)
3 0.2 0.2 1.0
(Clock, fast) & (Cache, yes) -> (Quality, high)
```

Every rule is preceded by four numbers: the total number of attribute-value pairs in the rule, the quality of the lower approximation, the quality of the upper approximation, and the conditional probability that the rule describes the concept defined by the right-hand side of the rule given the conditions on the left-hand side of the rule. A *quality of lower approximation* of X by P is equal to

$$\frac{|\underline{P}X|}{|U|}.$$

Thus, the quality of lower approximation of X by P is the ratio of the number of all certainly classified examples by attributes from P as being in X to the number of all examples of the system. A *quality of upper approximation* of X by P is equal to

$$\frac{|\overline{P}X|}{|U|}.$$

The quality of the upper approximation of X by P is the ratio of the number of all possibly classified examples by attributes from P as being in X to the number of all examples of the system.

On the other hand, if the user wants to induce rules using an approach based on knowledge acquisition, two additional options of LERS are available. The first option computes *global coverings* of required size and then rules from all of these coverings using similar ideas as in LEM1. The other option is similar to LEM2 (both are *local*) [6]. All rules in minimal form that can be derived from the input data are computed. In both options used algorithms are of exponential time complexity, with respect to the number of attributes. From Table 1, LERS using ALL_RULES option generates the following rule sets:

```
certain rule set:

2 0.2 0.2 1.0
(Main_Memory, small) -> (Quality, low)
3 0.2 0.2 1.0
(Clock, slow) & (Cache, no) -> (Quality, low)
3 0.2 0.2 1.0
(Clock, fast) & (Cache, no) -> (Quality, medium)
3 0.2 0.2 1.0
(Cache, no) & (Main_Memory, large) -> (Quality, medium)
3 0.2 0.2 1.0
(Clock, fast) & (Cache, yes) -> (Quality, high)
3 0.2 0.2 1.0
(Cache, yes) & (Main_Memory, large) -> (Quality, high)

possible rule set:

2 0.2 0.6 0.667
(Clock, slow) -> (Quality, low)
2 0.2 0.6 1.0
(Main_Memory, small) -> (Quality, low)
2 0.2 0.6 0.5
(Main_Memory, medium) -> (Quality, low)
3 0.2 0.6 0.5
(Clock, slow) & (Cache, yes) -> (Quality, medium)
3 0.2 0.6 1.0
(Clock, fast) & (Cache, no) -> (Quality, medium)
2 0.2 0.6 0.5
(Main_Memory, medium) -> (Quality, medium)
3 0.2 0.6 1.0
(Cache, no) & (Main_Memory, large) -> (Quality, medium)
3 0.2 0.2 1.0
(Clock, fast) & (Cache, yes) -> (Quality, high)
3 0.2 0.2 1.0
(Cache, yes) & (Main_Memory, large) -> (Quality, high)
```

3. Classification Systems

A classification system uses the rule set to classify new cases (examples). In general, the classification system classifies testing data using the rule set induced from training data.

Sometimes such a system is called an *expert system* since the rule set represents regularities hidden in training data that are known to experts. For example, we may use LERS to induce a rule set from the data describing patients diagnosed previously by an expert and then classify (or diagnose) new patients using this rule set.

Say that a decision table presents two concepts, I and II. A rule induction system induces two rule sets, describing these concepts. The set of all rules describing concept I will be called *description* I, while the set of all rules describing concept II will be called *description* II (see Figure 2). In general, descriptions are not perfect for many reasons (e.g., because the training data set was inconsistent). Thus, description I may uniquely and correctly classify some examples from concept I, but some examples from concept I may be described by rules from two different descriptions: I and II (area A from Figure 2), some examples from concept I may be described (wrongly) only by rules from description II (are B from Figure 2), and some examples from concept I may be not described at all, by any description (area C from Figure 2).

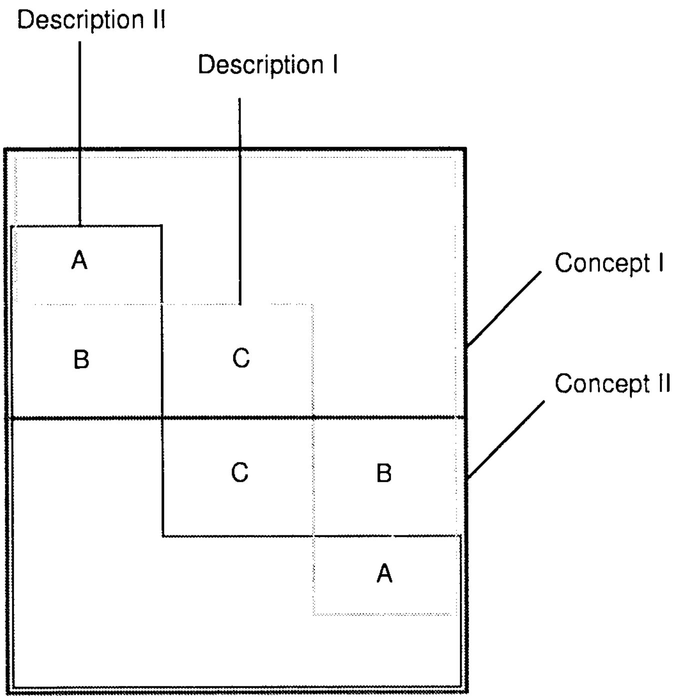

Figure 2. Concepts and descriptions

An original classification system of LERS was used in ESEP [7]. ESEP may run *manually* (one case at a time) or *automatically* (an entire file containing cases). In the manual mode for a new case all rules matching the case were used to produce the decision. The user was presented with all solutions together with their probabilities. For example, say that a new case is (*Cache*, *no*), (*Clock*, *fast*), (*Main_Memory*, *medium*). We would like to classify this case using certain and possible rules induced by LEM2. Two rules match the case:

3 0.2 0.2 1.0
(Clock, fast) & (Cache, no) -> (Quality, medium)

and

2 0.2 0.6 0.5
(Main_Memory, medium) -> (Quality, medium),

both indicate the same concept, so the conclusion reached is the case is classified as of medium quality.

Another case: (*Cache, no*), (*Clock, slow*), (*Main_Memory, medium*) is matched by the following two rules:

2 0.2 0.6 0.667
(Clock, slow) -> (Quality, low)

and

2 0.2 0.6 0.5
(Main_Memory, medium) -> (Quality, medium).

This time rules indicate different concepts: (*Quality, low*) and (*Quality, medium*). The conditional probability of the first rule (0.667) is larger than the conditional probability of the second rule (0.5), so it is more probable that the correct outcome is (*Quality, low*).

In the automatic mode, if more than one concept was indicated by rules, the classification of the case was counted as an error. Likewise, if a case was not completely classified by any of the rules, it was considered an error. This classification scheme is said to be a *naive* LERS classification scheme.

If an example is not completely matched by any rule, some classification systems use *partial matching.* System AQ15, during partial matching, uses the probabilistic sum of all measures of fit for rules [11]. Another approach to partial matching is presented in [16, 17]. Holland *et al.* [10] do not consider partial matching as a viable alternative of complete matching and rely on a default hierarchy instead. In ESEP, the user could run the system with rule matching *less one fact*, *less two facts*, or *less three facts.* If the option *less one fact* was selected, ESEP tries to match rules partially, with all conditions matching except one. Similarly for the remaining two options. These options increase the number of rules that may be used for classification.

The new classification system of LERS is a modification of the *bucket brigade algorithm* [1, 10]. The decision to which concept an example belongs is made on the basis of three factors: strength, specificity, and support. They are defined as follows: *Strength* is the total number of examples 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 scores of all matching rules from the concept. The concept C for which the support, i.e., the value of the following expression

$$\sum_{\textit{matching rules R describing C}} Strength(R) * Specificity\_factor(R)$$

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

In the new 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 an example.

For any partially matching rule R, the additional factor, called *matching factor* (R), is computed. The matching factor (R) is equal to the ratio of the number of matched attribute-value pairs of the rule R to the total number of attribute-value pairs of the rule R. In partial matching, the concept C for which the value of the following expression

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

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

A new version of LERS induces rules in a slightly different format. Rules are the same in both versions of LERS, old and new, only numbers associated with each rule are different. For Table 1, the rule set induced by the new version of LERS (LEM2 option) is:

certain rule set:

```
1, 1, 1
(Main_Memory, small) -> (Quality, low)
2, 1, 1
(Clock, fast) & (Cache, no) -> (Quality, medium)
2, 1, 1
(Clock, fast) & (Cache, yes) -> (Quality, high)
```

and possible rule set:

```
1, 2, 3
(Clock, slow) -> (Quality, low)
1, 1, 2
(Main_Memory, medium) -> (Quality, medium)
2, 1, 1
(Clock, fast) & (Cache, no) -> (Quality, medium)
2, 1, 1
(Clock, fast) & (Cache, yes) -> (Quality, high)
```

Every rule is preceded by three numbers: specificity, strength, and the total number of training cases matching the left-hand side of the rule.

For example, the case: (*Cache, no*), (*Clock, slow*), (*Main_Memory, medium*) is matched by the following two possible rules:

```
1, 2, 3
(Clock, slow) -> (Quality, low)
```

and

```
1, 1, 2
(Main_Memory, medium) -> (Quality, medium)
```

The support for the first concept, (*Quality, low*), is equal to 2. The support for the second concept, (*Quality, medium*), is equal to 1. Thus the system will classify the case as of low quality.

4. Validation

The most important performance criterion of knowledge discovery methods is the error rate. A complete discussion on how to evaluate the error rate from a data set is contained in

[9, 10]. 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 involves 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 example is removed from the data set, a rule set is induced by LEM2 from the remaining cases, and the classification of the omitted example by rules produced is recorded. The error rate is computed as

$$\frac{total\ number\ of\ misclassifications}{number\ of\ examples}$$

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 examples is divided into ten mutually disjoint subsets of approximately equal size. For each subset, all remaining examples 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.

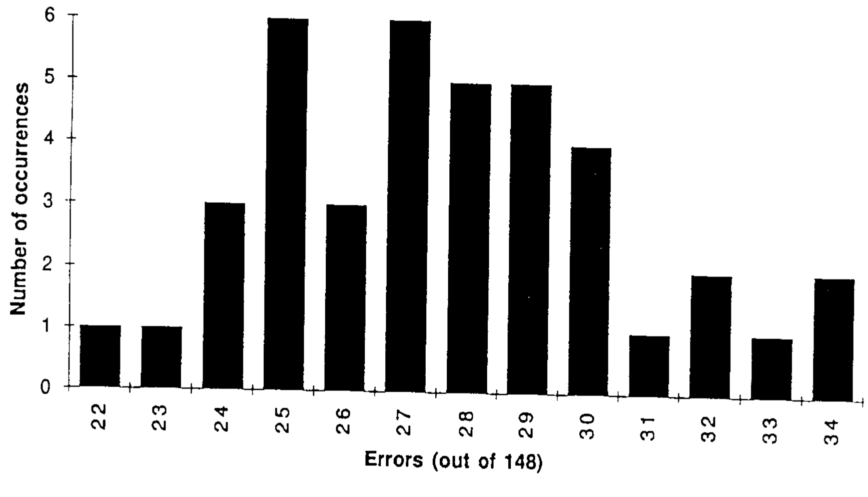

Figure 3. Lymphography data set ten-fold cross validation

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. We made an experiment of running LEM2 with ten-fold cross validation for the well-known data set *lymphography*. Results are presented on Figure 3. Error rates are dispersed between 22 (15%) and 34 (23%), with standard deviation 2.71 (1.83%). For *lymphography* data set and systems AQ15 [11] and C4.5 [16] error rates are cited in Table 5. Comparing LEM2 with AQ15 and C4.5, we may claim that LEM2 is by far superior (assuming 15% error rate) or that LEM2 is by far inferior (assuming 23% error rate). It should be clear that ten-fold cross validation is not quite a reliable tool to estimate the quality of rule induction systems.

For big 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* [19]. Two thirds of cases should be used for training, one third for testing.

5. Experiments

Both naive and new LERS classification schemes were tested on three well- known data sets: *lymphography*, *breast cancer*, and *primary tumor*. All three were obtained from the University Medical Center, Institute of Oncology, Ljubljana, Slovenia. Table 4 shows basic information about the three data sets.

Name	Number of cases	Number of attributes	Number of concepts
Lymphography	148	18	4
Breast cancer	286	9	2
Primary tumor	339	18	4

Table 4 Data Sets

Ten-fold cross validation was performed many times by using LEM2 and both classification schemes of LERS on these three data sets. The average error rate is given in Table 5. For comparison, in Table 5 error rates for two other well-known rule induction systems: AQ15 and C4.5 are also given.

Name	Naive LERS	New LERS	AQ15	C4.5
Lymphography	38%	19%	18–20%	23%
Breast cancer	49%	30%	32–34%	33%
Primary tumor	79%	67%	59–71%	60%

Table 5 Error Rates

For AQ15 error rates were computed using random subsampling with four iterations, where 70% of all cases were selected for training and 30% for testing [11]. The error rate for C4.5 was computed using the corresponding C4.5 feature for ten-fold cross validation. C4.5 reports always the same error rate since the re-ordering and sampling are the same every time this feature is invoked. Results for new classification scheme of LERS, AQ15 and C4.5 were reported in [8].

6. Conclusions

In all experiments rule sets were induced by the same LEM2 algorithm. Differences in error rates, reported in Table 5, are caused exclusively by different classification: naive and new. The difference in performance between these two classification schemes is dramatic. Clearly, the naive classification scheme performs very poorly. On the other hand, performance of the new LERS system, using the new classification scheme, is fully comparable with performance of AQ15 and C4.5.

The error rates for AQ15 and C4.5 from Table 5 were obtained using additional techniques, rule truncating and tree pruning, respectively. All error rates of LERS are original, without any additional techniques, improving error rate, such as elimination of the weakest rules (rules with the smallest values of strength). With these additional techniques the error rate of LERS may be further reduced, making LERS more competitive.

References

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

[2] 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, 1991.

[3] Chan, C.-C. and Grzymala-Busse, J. W.: "On the two local inductive algorithms: PRISM and LEM2". *Foundations of Computing and Decision Sciences*, **19**, 1994, 185 203.

[4] Dean, J. S. and Grzymala-Busse, J. W.: "An overview of the learning from examples module LEM1". Department of Computer Science, University of Kansas, TR-88-2, 1988.

[5] Grzymala-Busse, J. W.: *Managing Uncertainty in Expert Systems*, Boston, MA: Kluwer Academic Publishers, 1991.

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

[7] Grzymala-Busse, J. W.: "ESEP: An expert system for environmental protection", *Proc. of the RSKD-93, International Workshop on Rough Sets and Knowledge Discovery*, Banff, Alberta, Canada, October 12–15, 1993, 499–508.

[8] Grzymala-Busse, J. W.: "Managing uncertainty in machine learning from examples", *Proc. of the Third Intelligent Information Systems Workshop*, Wigry, Poland, June 6–11, 1994, 70–84.

[9] Hamburg, M.: *Statistical Analysis for Decision Making*, pp. 546 561, 716, 721, New York, NY: Harcourt Brace Jovanovich, Inc., third edition, 1983.

[10] Holland, J. H., Holyoak K. J., and Nisbett, R. E.: *Induction. Processes of Inference, Learning, and Discovery*, Cambridge, MA: The MIT Press, 1986.

[11] Michalski, R. S., Mozetic, I., Hong, J., and 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*, 1986, 1041–1045.

[12] Ohrn, A., and Komorowski, J.: "ROSETTA A rough set tool kit for analysis of data", *Proc. of the Fifth International Workshop on Rough Sets and Soft Computing (RSSC'97) at the Third Joint Conference on Information Sciences (JCIS'97)*, Research Triangle Park, NC, March 2–5, 1997, 403–407.

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

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

[15] Skowron, A.: "Extracting laws from decision tables: A rough set approach", Institute of Computer Science, Warsaw Institute of Technology, Res. Rep. 45/93, 1993.

[16] Slowinski, R. and Stefanowski, J.: "Handling various types of uncertainty in the rough set approach", *Proc of the RKSD-93, International Workshop on Rough Sets and Knowledge Discovery*, 1993, 395–397.

[17] Stefanowski, J.: "Classification and supporting based on rough set theory", *Found. Computing Decision Sci.*, **18**, 1993, 371–380.

[18] Quinlan, J. R.: *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[19] 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*, 17–49, San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1991.

[20] Ziarko, W.: "Systems: DataQuest, DataLogic and KDDR", *Proc. of the Fourth Int. Workshop on Rough Sets, Fuzzy Sets and Machine Discovery RSFD'96*, Tokyo, Japan, November 6–8, 1996, 441–442.