Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 378 – 387

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

# Comparison of heuristics for inhibitory rule optimization

Fawaz Alsolami[a,b,*], Igor Chikalov[a], Mikhail Moshkov[a]

[a]Computer, Electrical and Mathematical Sciences and Engineering Division
King Abdullah University of Science and Technology,Thuwal 23955-6900, Saudi Arabia
[b]Computer Science Department, King Abdulaziz University, Saudi Arabia

**Abstract**

Knowledge representation and extraction are very important tasks in data mining. In this work, we proposed a variety of rule-based greedy algorithms that able to obtain knowledge contained in a given dataset as a series of inhibitory rules containing an expression "attribute ≠ value" on the right-hand side. The main goal of this paper is to determine based on rule characteristics, rule length and coverage, whether the proposed rule heuristics are statistically significantly different or not; if so, we aim to identify the best performing rule heuristics for minimization of rule length and maximization of rule coverage.

Friedman test with Nemenyi post-hoc are used to compare the greedy algorithms statistically against each other for length and coverage. The experiments are carried out on real datasets from UCI Machine Learning Repository. For leading heuristics, the constructed rules are compared with optimal ones obtained based on dynamic programming approach. The results seem to be promising for the best heuristics: the average relative difference between length (coverage) of constructed and optimal rules is at most 2.27% (7%, respectively). Furthermore, the quality of classifiers based on sets of inhibitory rules constructed by the considered heuristics are compared against each other, and the results show that the three best heuristics from the point of view classification accuracy coincides with the three well-performed heuristics from the point of view of rule length minimization.

Peer-review under responsibility of KES International.
*Keywords:* inhibitory rules; rule heuristics; rule length; rule coverage

## 1. Introduction

Rule sets (systems) are widely used as techniques to solve problems, as classifiers for unseen instances (objects), and as tools to comprehend data. In data mining, rules often allow people to gain insight into data significantly more than other representations (models) such as neural networks, etc.

On the one hand, deterministic (usual) rules with relation "attribute = value" on the right-hand side are well-known tools for knowledge representation and pattern discovery[1,2,3]. On the other hand, inhibitory (nondeterministic) rules have in the consequent part a relation "attribute ≠ value"[4]. For the case of a binary decision attribute, inhibitory rules are equivalent to the deterministic ones; however, the situation is different for $k$-valued decision attribute, $k \geq 3$. It was

* Corresponding author.
*E-mail address:* fawaz.alsolami@kaust.edu.sa


Peer-review under responsibility of KES International.
doi:10.1016/j.procs.2014.08.118

shown in[5,6], for some datasets, deterministic rules cannot describe the whole information contained in the datasets whereas inhibitory rules are able to describe the whole information for every dataset[4]. Moreover, classifiers based on inhibitory rules can often acheive accuracy as good as classifiers based on deterministic rules[4,7,8,9].

Obtaining rules directly from a dataset is usually more easiser than deriving rules from a model. Different approaches have been proposed in the literature to obtain (induce) rules whether directly from datasets, for example, boolean reasoning[10,11], sequential covering[12,13,14,15,16,17], or from models such as decision trees[2,3,18]. Sequential covering approaches are based on learning (inducing) sequentially a set of rules in greedily manner. In short, a sequential covering algorithm searches (learns) for a single rule that covers a large number of positive examples (rows), eliminates the examples (rows) that in compliance with such rule, and learns recursively additional rules to cover the available examples. However, such approach does not guarantee to find as few as possible the number of learned rules, indeed, it is an instance of set covering problem. Our work has connections to test theory, rough set theory, and logical analysis of data theory. The three theories share the same concept of representing data using decision tables. Moreover, the notion of rules are known under various names such as representative tuples in test theory, patterns in logical analysis of data, and decision rules in rough set theory[19,20].

In this work, we concentrate on problem of minimization of rule length, the number of conditions (constraints) on the left-hand side, since shorter rules are usually more understandable. Similarly, we also focus on problem of maximization of rule coverage, the number of rows (objects) for which a rule applies to, because rule coverage is important for discovering major patterns in a given decision table (dataset). Of course, exact algorithms for the considered problems are computationally expensive. Thus, we propose a variety of heuristics for minimization of length of inhibitory rules and maximization of rule coverage. Statistical tests are applied on these algorithms to draw out the significant differences among them. Specifically, we compare the greedy algorithms for construction of inhibitory rules separately for length and coverage over multiple datasets from UCI Machine Learning Repository[21]. We perform this comparison using a Friedman test with Nemenyi post-hoc test[22,23] based on computing mean ranks of the algorithms. For leading heuristics, we compare the constructed rules with optimal ones obtained by dynamic programming algorithms[24]. For the best heuristics, the average relative difference between length of constructed and optimal rules is at most 2.27%. For coverage, the average relative difference is at most 7%. We compare the considered heuristics from the point of view of accuracy of classification based on constructed sets of inhibitory rules. The results show that three best greedy algorithms are those heuristics that can minimize on average the length of obtained rules compared to the other rule heuristics.

The paper consists of six sections. Section 2 contains definitions of main notions that employ throughout this paper. We present procedures of a variety of greedy algorithms in Section 3 and constructions of systems of rules in Section 4. The experiments setup and the results of the experiments are described in Section 5. Finally, Section 6 concludes the paper.

## 2. Main notions

A *decision table* $T$ is a rectangular table with $n$ columns labeled with conditional attributes $f_1, \ldots, f_n$. Rows of this table contain values of corresponding attributes. Rows of $T$ are pairwise different and each row is labeled with a value of the decision attribute $d$. We denote by $D(T)$ the set of decisions attached to rows of the table $T$.

Let $f_{j_1}, \ldots, f_{j_m} \in \{f_1, \ldots, f_n\}$ and $a_1, \ldots, a_m$ are values of the attributes $f_{j_1}, \ldots, f_{j_m}$ respectively. We denote by $T(f_{j_1}, a_1) \ldots (f_{j_m}, a_m)$ the subtable of $T$ which contains only rows that have $a_1, \ldots, a_m$ at the intersection with columns $f_{j_1}, \ldots, f_{j_m}$ respectively.

The expression

$$f_{j_1} = a_1 \wedge \ldots \wedge f_{j_m} = a_m \rightarrow d \neq q \tag{1}$$

is called an inhibitory rule over $T$ if $f_{j_1}, \ldots, f_{j_m} \in \{f_1, \ldots, f_n\}$, $a_1, \ldots a_m$ are values of $f_{j_1}, \ldots, f_{j_m}$, and $q \in D(T)$; we denote this rule (expression) by $\tau$. Thus, the left-hand side of the rule $\tau$ consists of conditions (constraints) where each constraint takes the form $(f_{j_i} = a_i)$.

Let $r = (v_1, \ldots, v_n)$ be a row of $T$ labeled with a decision $p$. We say that the rule $\tau$ covers the row $r$ if $a_1 = v_{j_1}, \ldots, a_m = v_{j_m}$. We say that the rule $\tau$ is true for $T$ if any row of $T$ covered by $\tau$ is labeled with a decision from $D(T) \setminus \{q\}$. If the rule $\tau$ is true for $T$ and realizable for row $r$, then the rule will be called an inhibitory rule for $T$,

$r$, and $q$. The length of the rule $\tau$ is the number of conditions on the left-hand side, which is denoted by $l(\tau)$. The coverage of the rule $\tau$ is the number of rows of $T$ covered by $\tau$, which is denote by $c(\tau)$.

## 3. Greedy algorithms for rule construction

In this section, we present a description of rule heuristics algorithms for an inhibitory rule construction. In general, all the proposed eight greedy algorithms perform the same search procedure for best conditions (constraints) to be added to the left-hand side based on a given attribute selection strategy. In other words, building an inhibitory rule follows the top-down fashion where in each search step a constraint is chosen according to attribute selection strategy of the considered rule heuristic. Thus, given a decision table $T$ with $n$ conditional attributes $f_1, \ldots, f_n$ each heuristic search algorithm generates for a row (instance) $r \in T$ an inhibitory rule labeled with decision (class label) $q \in D(T) \setminus \{p\}$, where $p$ is the decision of the row $r$.

**Algorithm 1** $Alg^1_{\mathcal{H}}, \mathcal{H} \in \left\{\mathcal{H}_1 = \frac{b}{a+1}, \mathcal{H}_2 = \frac{b}{\log_2(a+2)}\right\}$

**Require:** Decision table $T$ with conditional attributes $f_1, \ldots, f_n$, row $r = (v_1, \ldots, v_n)$ of $T$ labeled with a decision $p$, and a decision $q \in D(T) \setminus \{p\}$
**Ensure:** Inhibitory rule for $T$, $r$ and $q$
1: $Q \leftarrow \emptyset$
2: $T' \leftarrow T$
3: **while** $T'$ contains rows labeled with decision $q$ **do**
4: select $f_j \in \{f_1 \ldots f_n\}$ such that maximizes $\mathcal{H}$
5: $T' \leftarrow T'\left(f_j, v_j\right)$
6: $Q \leftarrow Q \cup \left\{f_j\right\}$
7: **end while**
8: $\bigwedge_{f_j \in Q}\left(f_j = v_j\right) \rightarrow d \neq q$

First, we describe two algorithms: an algorithm $Alg^1_{\mathcal{H}_1}$, denoted by *max_b_div_a*, and an algorithm $Alg^1_{\mathcal{H}_2}$, denoted by *max_b_div_*log *_a*, (see Algorithm 1). We will explain the work of the algorithms $Alg^1_{\mathcal{H}_1}$ and $Alg^1_{\mathcal{H}_2}$ for generating an inhibitory rule for the row $r = (v_1, \ldots, v_n) \in T$ that labeled with a decision $p$ for a fixed decision $q \in D(T) \setminus \{p\}$ in the consequent part. Algorithms $Alg^1_{\mathcal{H}_1}$ and $Alg^1_{\mathcal{H}_2}$ start with the initialization step in line 2 where they assign the given decision table $T$ to the subtable $T'$. The latter gets updated inside the `while-loop` to be a subtable formed by $T'(f_j, v_j)$. In line 4, both algorithms $Alg^1_{\mathcal{H}_1}$ and $Alg^1_{\mathcal{H}_2}$ select the best attribute value $(f_j, v_j)$ that optimizes the considered attribute selection strategy $\mathcal{H}$. Then the chosen attribute value is added to the body of the rule being constructed. Specifically, the algorithm $Alg^1_{\mathcal{H}_1}$ aims to find the attribute value that optimizes the ratio $\mathcal{H}_1 = \frac{b}{a+1}$ whereas $Alg^1_{\mathcal{H}_2}$ optimizes this ratio $\mathcal{H}_2 = \frac{b}{\log_2(a+2)}$, where $b$ is difference between the number of improperly classified rows or cases by the current rule before and after adding the chosen constraint $(f_j, v_j)$. Similarly, $a$ is difference between the number of properly classified rows or cases by the current rule before and after adding the constraint $(f_j, v_j)$.

Thus, the algorithms $Alg^1_{\mathcal{H}_1}$ and $Alg^1_{\mathcal{H}_2}$ choose the best attribute value that separates from the row $r$ a maximum number of rows labeled with decision $q$ and a minimum number of rows with decision other than $q$ in every search step. In addition, both rule heuristics stop adding new constraints to the rule being constructed when there is no any row labeled with the decision $q$.

Analogously, the work of the algorithm $Alg^2$, denoted by *min_a*, is similar to the previous rule heuristics algorithms and shown in Algorithm 2. However, the algorithm $Alg^2$ considers only a single criterion for the attribute selection strategy. Specifically, $Alg^2$ chooses in every iteration of the `while-loop` the attribute value $(f_j, v_j)$ that decreases the number of improperly classified rows or cases after adding the constraint $(f_j, v_j)$ by at least one and removes the minimum number of properly classified rows or cases. Where improperly classified rows have the same decisions as in the right part of the rule and properly classified rows have different decisions.

**Algorithm 2** $Alg^2$

**Require:** Decision table $T$ with conditional attributes $f_1, \ldots, f_n$, row $r = (v_1, \ldots, v_n)$ of $T$ labeled with a decision $p$, and a decision $q \in D(T) \setminus \{p\}$
**Ensure:** Inhibitory rule for $T$, $r$ and $q$
1: $Q \leftarrow \emptyset$
2: $T' \leftarrow T$
3: **while** $T'$ contains rows labeled with decision $q$ **do**
4: select $f_j \in \{f_1 \ldots f_n\}$ such that $b \geq 1$ and $a$ is minimum
5: $T' \leftarrow T'\left(f_j, v_j\right)$
6: $Q \leftarrow Q \cup \left\{f_j\right\}$
7: **end while**
8: $\bigwedge_{f_j \in Q}\left(f_j = v_j\right) \rightarrow d \neq q$

**Algorithm 3** $Alg^3_{\mathcal{U}}, \mathcal{U} \in \{ent, gini, me, rme, R\}$

**Require:** Decision table $T$ with conditional attributes $f_1, \ldots, f_n$, row $r = (v_1, \ldots, v_n)$ of $T$ labeled with a decision $p$, and a decision $q \in D(T) \setminus \{p\}$
**Ensure:** Inhibitory rule for $T$, $r$ and $q$
1: $Q \leftarrow \emptyset$
2: $T' \leftarrow T$
3: **while** $T'$ contains rows labeled with decision $q$ **do**
4: select $f_j \in \{f_1 \ldots f_n\}$ such that $\mathcal{U}\left(T'\left(f_j, v_j\right), q\right)$ is minimum
5: $T' \leftarrow T'\left(f_j, v_j\right)$
6: $Q \leftarrow Q \cup \left\{f_j\right\}$
7: **end while**
8: $\bigwedge_{f_j \in Q}\left(f_j = v_j\right) \rightarrow d \neq q$

Last family of attribute selection strategies depend on minimizing given uncertainty measures. In particular, the constraint $\left(f_j, v_j\right)$ that minimizes a given uncertainty measure $\mathcal{U}$ is added to the left part of the rule being constructed for the row $r = (v_1, \ldots, v_n)$, during each iteration of the `while-loop` as shown in Algorithm 3.

Let $T'$ be a subtable that contains all rows from a given decision table $T$ satisfying all conditions in the left part of current rule. For each $t \in D(T)$, we denote by $N_t$ the number of rows in $T'$ labeled with the decision $t$. Let $p_t = \frac{N_t}{N}$ where $N$ is the total number of rows in the subtable $T'$. We describe five uncertainty measures where two of them depend both on the subtable $T'$ and the fixed decision of the right part of the rule $q \in D(T)$. The other three depend only on the subtable $T'$. In Algorithm 3, we consider general case where $\mathcal{U}$ depends on both parameters.

- Entropy: $ent(T') = -\sum_{t \in D(T)} p_t \log_2 p_t$. We denote $Alg^3_{ent}$ algorithm by *min_ent*.
- Gini Index : $gini(T') = 1 - \sum_{t \in D(T)} (p_t)^2$. We denote $Alg^3_{gini}$ algorithm by *min_gini*.
- $R(T') = \frac{N^2 gini(T')}{2}$ (the number of unordered pairs of rows with different decisions in $T'$). We denote $Alg^3_R$ algorithm by *min_R*.
- Misclassification error: $me(T', q) = N_q$. We denote $Alg^3_{me}$ algorithm by *min_me*.
- Relative misclassification error: $rme(T', q) = \frac{N_q}{N}$. We denote $Alg^3_{rme}$ algorithm by *min_rme*.

## 4. Construction of systems of rules

In this paper, two problems of optimization of the model cost are considered: length and coverage of inhibitory rules. Let $\mathcal{A}$ be one of the eight algorithms described in the previous section. Then, the algorithm $\mathcal{A}$ constructs a set of inhibitory rules for each row $r$ of $T$ in the following way. If $r$ is labeled with a decision $p$, then the algorithm $\mathcal{A}$ generates an inhibitory rule for each decision $q \in D(T) \setminus \{p\}$. Finally, among the set of rules generated for the row $r$, the rule with the minimum length is added to the system of minimum rule length $l_{\mathcal{A}}$ for $T$. Similarly, the rule with the maximum coverage from the set of generated rules to the row $r$ is added to the system of maximum rule coverage $c_{\mathcal{A}}$ for $T$.

Let $T$ contain $N$ rows $\{r_1, \ldots, r_N\}$, and $S$ be a system of $N$ rules either $l_{\mathcal{A}}$ or $c_{\mathcal{A}}$. We consider two performance scores for rules from $S$ : average length $L_{avg}(S)$ and average coverage $C_{avg}(S)$ where

$$L_{avg}(S) = \frac{\sum_{\tau \in S} l(\tau)}{N} \text{ and } C_{avg}(S) = \frac{\sum_{\tau \in S} c(\tau)}{N}. \tag{2}$$

To solve exactly the two problems of optimization of the model cost are considered: length and coverage of inhibitory rules, we can use dynamic programming approach[24]. However, such algorithms are computationally complex and they cannot work on large decision tables. For $i = 1, \ldots, N$, let $l(r_i)$ be the minimum length of an inhibitory rule which is true for $T$ and covers $r_i$, and $c(r_i)$ be the maximum coverage of an inhibitory rule which is true for $T$ and covers $r_i$. Then

$$L^{opt}_{avg}(T) = \frac{1}{N}\sum_{i=1}^{N} l(r_i) \text{ and } C^{opt}_{avg}(T) = \frac{1}{N}\sum_{i=1}^{N} c(r_i). \tag{3}$$

## 5. Experimental results

In this section, the experimental setups for this comparative study are described in terms of data sets (decision tables) and statistical tests. Additionally, the results connected with real decision tables and the accuracy of prediction are presented. For analysis purpose, each algorithm is named by the template ⟨$l$⟩_⟨name of the considered algorithm⟩ when the performance score is the average length of rules. In a similar way, each algorithm is named by the template ⟨$c$⟩_⟨name of the considered algorithm⟩ when the performance score is the average coverage of rules.

### *5.1. Comparison of rule heuristics: statistical tests*

To compare the algorithms statistically, we use Friedman test with Nemenyi post-hoc test as suggested in[22]. Let we have $k$ algorithms $A_1, \ldots, A_k$ for building systems of rules and $M$ decision tables $T_1, \ldots, T_M$. For each decision table $T_i$, $i = 1, \ldots, M$, we rank the algorithms $A_1, \ldots, A_k$ on $T_i$ based on their performance scores (from the point of view either average length or average coverage), where we assign the best performing algorithm the rank of 1, the second best rank 2, and so on. We break ties by computing the average of ranks. Let $r_i^j$ be the rank of the $j$-th of $k$ algorithms on the decision table $T_i$. For $j = 1, \ldots, k$, we correspond to the algorithm $A_j$ the average rank

$$R_j = \frac{1}{M}\sum_{i=1}^{M} r_i^j. \tag{4}$$

For a fixed significance level $\alpha$ (in our work $\alpha = 0.10$ mainly), the performance of two algorithms is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6M}} \tag{5}$$

where $q_\alpha$ is a critical value for the two-tailed Nemenyi test depending on $\alpha$ and $k$ (see[22]).

We can also compare performance scores of algorithms $A_1, \ldots, A_k$ with optimal results obtained by dynamic programming[24]. For $j = 1, \ldots, k$ and $i = 1, \ldots, M$, we denote, respectively, by $l_{ij}$ and $c_{ij}$ average length and average

Table 1: Decision tables from UCI Machine Learning Repository for study of length and coverage

| | Decision table | # Rows | #Attributes | # Decisions |
|---|---|---|---|---|
| 1 | adult-stretch | 16 | 4 | 2 |
| 2 | agaricus-lepiota | 8124 | 22 | 2 |
| 3 | balance-scale | 625 | 4 | 3 |
| 4 | breast-cancer | 266 | 9 | 2 |
| 5 | cars | 1727 | 6 | 4 |
| 6 | flags | 193 | 26 | 6 |
| 7 | hayes-roth | 69 | 4 | 3 |
| 8 | kr-vs-kp | 3196 | 36 | 2 |
| 9 | lymphography | 148 | 18 | 4 |
| 10 | nursery | 12960 | 8 | 5 |
| 11 | shuttle-landing | 15 | 6 | 2 |
| 12 | soybean-small | 47 | 35 | 4 |
| 13 | spect-test | 169 | 22 | 2 |
| 14 | tic-tac-toe | 958 | 9 | 2 |
| 15 | zoo-data | 59 | 16 | 7 |

coverage of the system of rules constructed by the algorithm $A_j$ on the decision table $T_i$. For $i = 1, \ldots, M$, we denote by $l_i^{opt}$ the value of $L_{avg}^{opt}(T_i)$ which is the minimum possible average length of a system of rules for $T_i$, and $c_i^{opt}$ the value of $C_{avg}^{opt}(T_i)$ which is the maximum possible average coverage of a system of rules for $T_i$. Thus, we can compute the average relative difference (for length and coverage) as follows:

$$ARD_j^l = \frac{1}{M}\sum_{i=1}^{M}\frac{l_{ij} - l_i^{opt}}{l_i^{opt}}\ ,\ ARD_j^c = \frac{1}{M}\sum_{i=1}^{M}\frac{c_i^{opt} - c_{ij}}{c_i^{opt}}. \tag{6}$$

The average relative difference shows how close on average a performance (length or coverage) of an approximate solution to the optimal solution.

### 5.2. Real data sets

We performed the experiments on 15 decision tables from UCI Machine Learning Repository[21] as shown in Table 1. Sampling was used 100 times to select randomly 70% of rows of each table. We remove from the obtained set of tables the decision tables in which all rows were labeled with the same decision, and make experiments with the remaining tables.

First, we compare the performance scores of rule heuristics algorithms: *l_max_b_div_a*, *l_max_b_div_*log *_a*, *l_min_a*, *l_min_ent*, *l_min_R*, *l_min_me*, *l_min_rme*, and *l_min_gini* according to average length of obtained rules. Figure 1 shows the critical difference diagram containing average (mean) rank for each heuristic on the $x$-axis for significance level at $\alpha = 0.10$. When Nemenyi test is not enough to identify any significant difference between some algorithms, the algorithms are clustered (connected). One can see that the algorithm *l_min_a* leads to generate on average rules with large number of constraints on left part. The best three algorithms are *l_min_me*, *l_min_rme* (these two algorithms are in the same cluster), and *l_max_b_div_*log *_a*. Table 2 contains the average relative difference $ARD^l$ for these algorithms. Thus, it is clear that the best three algorithms can produce on average close to the optimal.

Similarly, we compare average coverage scores of rule heuristics algorithms *c_max_b_div_a*, *c_max_b_div_*log *_a*, *c_min_a*, *c_min_ent*, *c_min_gini*, *c_min_me*, *c_min_rme*, and *c_min_R* as shown in Figure 2. Note that it is convenient for us to rank as the first the algorithm which constructs system of rules with minimum average coverage, and so on. In other words, it means that the best performing algorithm corresponds to the maximum average rank. The average relative difference $ARD^c$ for the three best algorithms *c_max_b_div_a*, *c_max_b_div_*log *_a*, and *c_min_rme* can

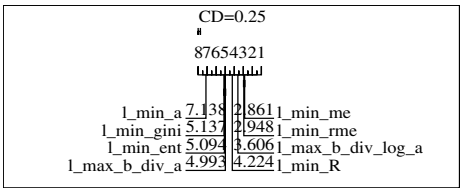


Fig. 1: Comparison of length minimization (tables from [21])

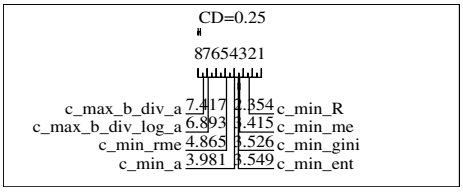


Fig. 2: Comparison of coverage maximization (tables from [21])

Table 2: Leading algorithms for minimizing length (tables from [21])

| Algorithm | Average relative difference |
|---|---|
| *l_min_me* | 2.27% |
| *l_min_rme* | 3.8% |
| *l_max_b_div_*log *_a* | 6.1% |

Table 3: Leading algorithms for maximizing coverage (tables from [21])

| Algorithm | Average relative difference |
|---|---|
| *c_max_b_div_a* | 7% |
| *c_max_b_div_*log *_a* | 11% |
| *c_min_rme* | 28% |

be found in Table 3. Thus, it is clear that the best three algorithms can produce rules with coverage relatively close to optimal.

### *5.3. Classifiers based on sets of inhibitory rules*

In this section, we compare the prediction power of classifiers based on sets of inhibitory rules constructed by the presented algorithms. Let $\mathcal{A}$ be one of the considered algorithms and $T$ be a decision table. For each row $r$ of $T$ labeled with a decision $p$ and a decision $q \in D(T) \setminus \{p\}$ we add to the set of rules an inhibitory rule for $T, r$ and $q$ constructed by $\mathcal{A}$. This set of rules can be considered as a classifier that uses a voting scheme procedure where each rule votes against the decision in its right-hand side.

We evaluate the accuracy of considered classifiers for decision tables described in Table 4 using two-fold cross validation approach where a given decision table is randomly partitioned into two folds of equal size. In all eight rule heuristics algorithms, 30% of samples of the first fold are reserved for validation. Train and validation subtables

Table 4: Data sets from UCI Machine Learning Repository for study of classifiers

| | Decision table | # Rows | #Attributes | # Decisions |
|---|---|---|---|---|
| 1 | balance scale | 625 | 4 | 3 |
| 2 | DNA | 3186 | 60 | 3 |
| 3 | flags | 193 | 26 | 6 |
| 4 | hayes-roth | 69 | 4 | 3 |
| 5 | kr-vs-kp | 3196 | 36 | 2 |
| 6 | German | 1334 | 20 | 2 |
| 7 | mammographic | 961 | 5 | 2 |
| 8 | vehicle | 846 | 18 | 4 |

(samples) are passed to the considered rule heuristic algorithm that builds a model as rulesets. The model is applied to predict decisions for all samples from the second fold. Then the folds are swapped: the model is constructed from the samples of the second fold and makes prediction for all samples from the first fold. Finally, misclassification error is estimated as the average of misclassification error of the first and the second fold. In order to reduce variation of the estimate, 50 experiments for each model and each data set were performed.

A post-pruning technique is applied to a set of rules $\mathcal{R}$ generated by one of the present algorithms. For each rule in $\mathcal{R}$ that has uncertainty less than the predefined threshold $t$, we keep removing constraints from the end of the rule until the uncertainty of the resulted rule is greater than or equal the threshold $t$. The threshold that shows the minimum misclassification error on the validation subtables (samples) is selected.

Figure 3 shows the critical difference diagrams at level $\alpha = 0.1$ for the considered classifiers where a classifier with a small average rank has low error rate. One can see that a system of inhibitory rules built by the algorithms *min_rme* and *max_b_div_*log *_a* have significantly more accuracy than others.

It is interesting to note that the three well-performed heuristics are heuristics mentioned in Table 2 as the best heuristics for rule length minimization.

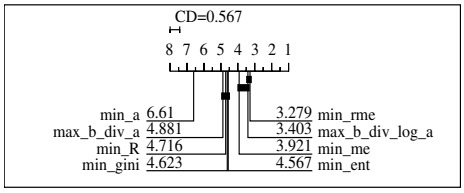


Fig. 3: Comparison of greedy algorithms for construction of classifiers

Table 5: Comparison of C4.5 algorithm with $Alg_{\mathcal{H}_2}$ on some benchmark data sets

| Decision table | Testing Mode | #Examples | C4.5 | $Alg^1_{\mathcal{H}_2}$ |
|---|---|---|---|---|
| DNA | Train & Test | 3186 | 92.4 | 94.3 |
| Vehicle | Cross-validation | 846 | 73.4 | 70 |
| sleep | Train & Test | 105908 | 72.3 | 73.09 |

Finally, we compare the algorithm $Alg_{\mathcal{H}_2}$, also called *max_b_div_log*, with C4.5 algorithm on some benchmark datasets[1 25] and results of comparison are shown on Table 5. One can see that the algorithm $Alg_{\mathcal{H}_2}$ can achieve accuracy as good as the C4.5 classifier.

[1] http://www.sgi.com/tech/mlc/db/

The Nemenyi test shows that the rule heuristic algorithm $Alg^1_{\mathcal{H}_2}$ is better than all the other proposed heuristics since it can produce on average rules with relatively short length, good coverage and accuracy on a par with the C4.5 classifier.

## 6. Conclusions

In this paper, we compare statistically a variety of greedy algorithms for construction inhibitory rules for decision tables from UCI Machine Learning Repository. Statistical tests allow us to identify well performed algorithms and recognize significant difference between the algorithms. Presented results show some of the greedy algorithms can obtain inhibitory rules with average length or coverage close to the optimal obtained by dynamic programming algorithms. Additionally, we compare classifiers based on sets of inhibitory rules constructed by the considered heuristics, and found that the three best heuristics from the point of view classification accuracy coincides with the three well-performed heuristics from the point of view of rule length minimization. The Nemenyi test shows that the rule heuristic algorithm $Alg^1_{\mathcal{H}_2}$ is better than all the other proposed heuristics since it can produce on average rules with relatively short length, good coverage and accuracy on a par with the C4.5 classifier.

## Acknowledgement

Research reported in this publication was supported by the King Abdullah University of Science and Technology (KAUST).

## References

1. Błaszczyński, J., Słowiński, R., Szelag, M.. Sequential covering rule induction algorithm for variable consistency rough set approaches. *Information Sciences* 2011;**181**(5):987–1002.
2. Michalski, S., Pietrzykowski, J.. iAQ: A program that discovers rules. AAAI-07 AI Video Competition; 2007.
3. Moshkov, M., Zielosko, B.. *Combinatorial Machine Learning*; vol. 360 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg; 2011.
4. Delimata, P., Moshkov, M., Skowron, A., Suraj, Z.. *Inhibitory Rules in Data Analysis: A Rough Set Approach*; vol. 163 of *Studies in Computational Intelligence*. Springer; 2009.
5. Skowron, A., Suraj, Z.. Rough sets and concurrency. *Bulletin of the Polish Academy of Sciences* 1993;**41(3)**:237–254.
6. Suraj, Z.. Some remarks on extensions and restrictions of information systems. In: *Rough Sets and Current Trends in Computing*; vol. 2005 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2001, p. 204–211.
7. Delimata, P., Moshkov, M., Skowron, A., Suraj, Z.. Two families of classification algorithms. In: *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*; vol. 4482 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2007, p. 297–304.
8. Delimata, P., Moshkov, M., Skowron, A., Suraj, Z.. Lazy classification algorithms based on deterministic and inhibitory decision rules. In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. 2008, p. 1773–1778.
9. Delimata, P., Moshkov, M., Skowron, A., Suraj, Z.. Comparison of lazy classification algorithms based on deterministic and inhibitory decision rules. In: *Rough Sets and Knowledge Technology*; vol. 5009 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2008, p. 55–62.
10. Nguyen, H.. Approximate boolean reasoning: Foundations and applications in data mining. In: *Transactions on Rough Sets V*; vol. 4100 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2006, p. 334–506.
11. Pawlak, Z., Skowron, A.. Rough sets and boolean reasoning. *Information Sciences* 2007;**177**(1):41–73.
12. Dembczyński, K., Kotłowski, W., Słowiński, R.. Ender: a statistical framework for boosting decision rules. *Data Mining and Knowledge Discovery* 2010;**21**(1):52–90.
13. Fürnkranz, J.. Separate-and-conquer rule learning. *Artificial Intelligence Review* 1999;**13**:3–54.
14. Lavrač, N., Fürnkranz, J., Gamberger, D.. Explicit feature construction and manipulation for covering rule learning algorithms. In: *Advances in Machine Learning I*; vol. 262 of *Studies in Computational Intelligence*. Springer; 2010, p. 121–146.
15. Cohen, W.W., Singer, Y.. A simple, fast, and effective rule learner. In: *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*; AAAI '99/IAAI '99. American Association for Artificial Intelligence; 1999, p. 335–342.
16. Clark, P., Niblett, T.. The cn2 induction algorithm. *Machine Learning* 1989;**3**:261–283.
17. Fürnkranz, J., Gamberger, D., Lavrac, N.. *Foundations of Rule Learning*. Cognitive Technologies. Springer; 2012.
18. Quinlan, J.R.. *C4.5: Programs for Machine Learning*. Morgan Kaufmann; 1993.
19. Chikalov, I., Lozin, V., Lozina, I., Moshkov, M., Nguyen, H.S., Skowron, A., et al. *Three Approaches to Data Analysis*; vol. 41 of *Intelligent Systems Reference Library*. Springer-Verlag Berlin Heidelberg; 2013.
20. Crama, Y., Hammer, P.L., Ibaraki, T.. Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research* 1988;**16**:299–325.

21. Asuncion, A., Newman, D.J.. UCI Machine Learning Repository. 2014. `http://archive.ics.uci.edu/ml/`.
22. Demšar, J.. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 2006;**7**:1–30.
23. Peter, N.. *Distribution-free multiple comparisons*. Doctoral thesis; Princeton University; 1963.
24. Alsolami, F., Chikalov, I., Moshkov, M., Zielosko, B.. Length and coverage of inhibitory decision rules. In: *Computational Collective Intelligence. Technologies and Applications*; vol. 7654 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg; 2012, p. 325–334.
25. Kohavi, R., Sommerfield, D., Dougherty, J.. Data mining using mlc++: A machine learning library in c++. *INTERNATIONAL JOURNAL ON ARTIFICIAL INTELLIGENCE TOOLS* 1997;**6**(4).