Building Scalable Lazy Rough Classifiers

Piotr Kwiatkowski and Hung Son Nguyen
Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland

Abstract
We present a lazy classification method, which is equivalent to the rough set based method for construction of classifiers, but is scalable and applicable for large data sets. The proposed method is based on our previous papers Nguyen (2002) Nguyen (2006), where the relationship between rough sets methods and association learning methods were discovered. In this method the set of decision rules matching the test object is generated directly from the training data set. To make it scalable, we adopted the idea of the FP-growth algorithm for frequent item-sets Han et al. (2000), Han and Kamber (2000). Except the classification task, this method can be applied for construction of adaptive rule generation system where data are growing up in time.

Keywords: data mining, scalability, rough set, case-based reasoning, lazy learning

1 Introduction
Classification of new unseen objects is the most important task in data mining. There are many classification approaches likes “nearest neighbors”, “naive Bayes”, “decision tree”, “decision rule set”, “neural networks” etc. Every classification method has some advantages and disadvantages, hence the choice of classification methods in practical data mining applications depends on different criteria like: accuracy, description clearness, time and memory complexity etc.

Almost all methods based on rough sets use the rule set classification approach (see e.g., Bazan (1998), Stefanowski (1998), Wroblewski (1998), Ziarko (1998)), which consists of two steps: generalization and specification. In generalization step, some decision rule set is constructed from data as a knowledge base. In specialization step the set of such rules, that match a new object (to be classified) is selected and a conflict resolving mechanism will be employed to make the decision for the new object.

Unfortunately, there are opinions that rough set based methods can be used for small data set only. The main reproach is related to their lack of scalability (more precisely: there is a lack of proof showing that they can be scalable). The biggest troubles stick in the rule induction step. As we know, the potential number of all rules is exponential. All heuristics for rule induction algorithms have at least $O(n^2)$ time complexity, where $n$ is the number of objects in the analyzed data set. Moreover, the existing algorithms require multiple data scanning.
In our previous paper Nguyen (2002), we proposed to adopt the lazy learning idea to make rough set based methods more scalable. The proposed method does not consist of the generalization step. The main effort is shifted into rule matching step. We had shown that the set of such rules, that match an object (to be classified) can be selected by a modification of Apriori algorithm proposed in Agrawal et al. (1996) for sequent item set generation from data bases.

This paper presents another method to this problem. The approach is based on modification of FP-growth algorithms Han et al. (2000), Han and Kamber (2000). The FP-growth algorithm is known as an efficient and scalable method for frequent pattern discovery from transaction data sets. We present the method called FDP, which is in fact a modification of FP-growth, but is applicable for decision tables. We present the experimental results to confirm the advantages of the proposed method.

2 Basic Notions

In this Section, we recall some well known notions related to rough sets and classification systems.

An information system Pawlak (1991) is a pair $A = (U, A)$, where $U$ is a non-empty, finite set of objects and $A = \{a_1, ..., a_k\}$ is a non-empty finite set of attributes (or features), i.e. $a_i : U \rightarrow V_{a_i}$ for $i = 1, ..., k$, where $V_{a_i}$ is called the domain of $a_i$. Let $B = \{a_{i_1}, ..., a_{i_j}\}$, where $1 \leq i_1 < ... < i_j \leq k$, be a subset of $A$, the set

$$INF_B = V_{a_{i_1}} \times V_{a_{i_2}} \times ... \times V_{a_{i_j}}$$

is called information space defined by $B$. Function $inf_B : U \rightarrow INF_B$ defined by $inf_B(u) = \langle a_{i_1}(u), ..., a_{i_j}(u) \rangle$ is called “$B$-information map”. The function $inf_B$ defines a projection of objects from $U$ into information space $INF_B$ (or a view of $U$ on features from $B$).

Two objects $x, y \in U$ are said to be indiscernible by attributes from $B$ if $inf_B(x) = inf_B(y)$. It is easy to show that the relation $IND(B) = \{(x, y) : inf_B(x) = inf_B(y)\}$, called indiscernibility relation, is the equivalent relation (see Skowron and Rauszer (1992)). For any $u \in U$, the set $[u]_B = \{x \in U : (x, u) \in IND(B)\}$ is called equivalent class of $u$ relative to $B$. Equivalent classes can be treated as building block to define basic notions of rough set theory.

The main subject of rough set theory is concept description, which is the most important challenge in Data Mining. Any concept can be associated with the set of elements belonging to this concept. Let $X \subseteq U$ be a concept to be describe and let $B \subseteq A$ be a set of accessible attributes. The set $X$ can be described by attributes form $B$ by $(\overline{B}X, \overline{BX})$, where

$$\overline{B}X = \{u \in U : [u]_B \subseteq X\}, \quad \overline{BX} = \{u \in U : [u]_B \cap X \neq \emptyset\}$$

are the $B$-lower approximation of $X$ and the $B$-upper approximation of $X$, respectively.
2.1 Classification Problem

Any information system of the form $A = (U, A \cup \{\text{dec}\})$ with a distinguished attribute $\text{dec}$ is called a decision table. The attribute $\text{dec} \notin A$ is called the decision attribute (or the decision, for short).

The classification problem can be formulated using decision tables. Assume that objects from an universe $X$ are classified into $d$ classes by a decision function $\text{dec}: X \to V_{\text{dec}} = \{1, \ldots, d\}$ which is unknown for learner. Every object from $X$ is characterized by attributes from $A$, but the decision $\text{dec}$ is known for objects from a sample set $U \subset X$ only. The information about function $\text{dec}$ is given by decision table $A = (U, A \cup \{\text{dec}\})$. The problem is to construct from $A$ a function $L_A: \text{INF}_A \to V_{\text{dec}}$ in such a way, that the probability

$$P(\{u \in X : \text{dec}(u) = L_A(\text{inf}_A(u))\})$$

is sufficiently high. The function $L_A$ is called decision algorithm and the methods constructing them from given decision table $A$ are called classification methods.

Many classification methods have been proposed to solve the classification problem. In this paper, we are dealing with the decision rule based approach, which is preferred by many Rough Set based classification methods, e.g., Bazan (1998), Stefanowski (1998), Wroblewski (1998), Ziarko (1998).

Let $A = (U, A \cup \{\text{dec}\})$ be a decision table. Without loss of generality we assume that $V_{\text{dec}} = \{1, \ldots, d\}$. Then the set $\text{DEC}_k = \{x \in U : \text{dec}(x) = k\}$ will be called the $k^{th}$ decision class of $A$ for $1 \leq k \leq d$. Any implication of form

$$(a_{i_1} = v_1) \land \ldots \land (a_{i_m} = v_m) \Rightarrow (\text{dec} = k) \quad (1)$$

where $a_{i_j} \in A$ and $v_j \in V_{a_{i_j}}$, is called decision rule for $k^{th}$ decision class. Let $r$ be an arbitrary decision rule of the form (1), then $r$ can be characterized by following parameters:

- **The length of $r$**: the number of descriptor in the assumption of $r$ (i.e., the left hand side of implication). We denote the length of $r$ by $\text{length}(r)$.
- **The carrier of $r$**: the set of all objects from $U$ satisfying the assumption of $r$. We denote the carrier of $r$ by $[r]$.
- **The support of $r$**: the number of objects satisfying the assumption of $r$:

$$\text{support}(r) = |[r]|,$$

where by $|X|$ we denote the cardinality of the set $X$.
- **The confidence of $r$**: the measure of truth of the decision rule. It can be defined by:

$$\text{confidence}(r) = \frac{|[r] \cap \text{DEC}_k|}{|[r]|}$$

The decision rule $r$ is called consistent with $A$ if $\text{confidence}(r) = 1$.

In data mining philosophy, we are interested in extraction of short and strong decision rules with high confidence. The linguistic features like "short", "strong"
“high confidence” of decision rules can be formulated using of their length, support and confidence. Such rules can be treated as interesting, valuable and useful patterns in data. Unfortunately, the number of such patterns can be exponential with regard to the size of the given decision table Komorowski et al. (1999), Skowron and Rauszer (1992), Bazan (1998), Wroblewski (1998). In practice, we must apply some heuristics to generate a subset of decision rules.

The rule based classification methods work in three phases:

1. Learning phase: generates a set of decision rules \( RULES(\mathcal{A}) \) (satisfying some predefined conditions) from a given decision table \( \mathcal{A} \).

2. Rule selection phase: selects from \( RULES(\mathcal{A}) \) the set of such rules that can be supported by \( x \). We denote this set by \( MatchRules(\mathcal{A}, x) \).

3. Post-processing phase: makes a decision for \( x \) using some voting algorithm for decision rules from \( MatchRules(\mathcal{A}, x) \).

\[
\text{Figure 1: The Rule base classification system}
\]

2.2 Rough Sets and classification problems

Many decision rule generation methods have been developed by using Rough set theory. One of the most interesting approaches is related to minimal consistent decision rules. Given a decision table \( \mathcal{A} = (U, A \cup \{\text{dec}\}) \), the decision rule:

\[
r = \text{def} (a_{i_1} = v_1) \land ... \land (a_{i_m} = v_m) \Rightarrow (\text{dec} = k),
\]

is called minimal consistent decision rule if it is consistent with \( \mathcal{A} \) and any decision rule \( r' \) created from \( r \) by removing one of descriptors from left hand side of \( r \) is not consistent with \( \mathcal{A} \). The set of all minimal consistent decision rules for a given decision table \( \mathcal{A} \) is denoted by \( \text{MinConsRules}(\mathcal{A}) \).

The set of all minimal consistent decision rules can be found by computing object oriented reducts (or local reducts) Komorowski et al. (1999), Bazan (1998) Wroblewski (1998). Let us recall the method based on boolean reasoning approach for computing of such reducts Skowron and Rauszer (1992), Komorowski et al. (1999), Nguyen (2006). Let \( \text{Var} = \{\alpha_1, ..., \alpha_k\} \) be a set of boolean variables
corresponding to attributes $a_1,\ldots,a_k$ from $A$. Let $u,v \in U$ be objects from $U$. One can define the discernibility function for $u,v$ as follows:

$$disc_{u,v}(\alpha_1,\ldots,\alpha_k) = \bigvee \{\alpha_i : a_i(u) \neq a_i(v)\}.$$ 

Let $u \in U$ be an arbitrary object in decision table $A = (U, A \cup \{\text{dec}\})$. We can define a function $f_u(d_1,\ldots,d_k)$ called discernibility function for $u$ as follows:

$$f_u(\alpha_1,\ldots,\alpha_k) = \bigwedge_{\text{dec}(u) \neq \text{dec}(v)} disc_{u,v}.$$ 

Every prime implicant of $f_u$ corresponds to “local reduct” for the object $u$ and such reducts are associated with a minimal consistent decision rules Skowron and Rauszer (1992), Komorowski et al. (1999), Nguyen (2006). We denote by $\text{MinRules}(u)$ the set of all minimal consistent decision rules created from boolean function $f_u$. One can show that

$$\text{MinConsRules}(A) = \bigcup_{u \in U} \text{MinRules}(u).$$ 

The set $\text{MinConsRules}(A)$ can be used as a knowledge base in classification systems. In practice, instead of $\text{MinConsRules}(A)$, we can use the set of short, strong, and high accuracy decision rules defined by:

$$\text{MinRules}(A, \lambda_{\text{max}}, \sigma_{\text{min}}, \alpha_{\text{min}}) = \left\{ r : r \text{ is minimal } \land \text{length}(r) \leq \lambda_{\text{max}} \land \text{support}(r) \geq \sigma_{\text{min}} \land \text{confidence}(r) \geq \alpha_{\text{min}} \right\}.$$ 

All heuristics for object oriented reducts can be modified to extract decision rules from $\text{MinRules}(A, \lambda_{\text{max}}, \sigma_{\text{min}}, \alpha_{\text{min}})$.

### 2.3 Lazy learning

The classification methods based on learning schema presented in Figure 1 are called eager (or laborious) methods. Every eager method tries to extract some generalized theory from given decision table (this process is called generalization) and next it classifies new objects by applying the generalized theory to the new objects (specialization).

In lazy learning methods new objects are classified without generalization step. For example, in kNN (k Nearest Neighbors) method, the decision of new object $x$ can be made by taking a vote between $k$ nearest neighbors of $x$. In lazy decision tree method, we try to reconstruct the path $p(x)$ of the “imaginable decision tree” that can be applied for new object $x$.

The lazy methods need more time complexity for the classification step, i.e., the answer time for the question about decision of a new object is much longer than in eager classification methods. But lazy classification methods are well scalable, i.e. it can be realized for larger decision table using distributed computer system Bondi (2000), Shafer et al. (1996). The scalability property is also very advisable in data mining. Unfortunately, the eager classification methods are weakly scalable. As we recall before, the time and memory complexity of existing algorithms does not
make it possible to apply rule base classification methods for very large decision table\textsuperscript{1}.

\section{3 Lazy learning for rough sets methods}

Decision rules play an important role in KDD and data mining. Rule-based classifiers establish an accurate and interpretable model for data.

As it has been mentioned before, any rule-based classification method consists of three steps: (1) rule generation, (2) rule selection and (3) decision making (e.g., by voting). The general framework for rule based classification methods is presented in Fig. 1. In machine learning, this approach is called \textit{eager (or laborious) learning methodology}.

In lazy learning methods new objects are classified without the generalization step. For example, in \(k\)NN (\(k\) Nearest Neighbors) classifiers, the decision of new object \(x\) can be made by taking a vote between \(k\) nearest neighbors of \(x\). In lazy decision tree method, we try to reconstruct the path \(p(x)\) of the “imaginable decision tree” that can be applied to \(x\).

Lazy learning methods need more time complexity for the classification step, i.e., the answer time for the question about decision of a new object is longer than in eager classification methods. But lazy classification methods are \textit{well scalable}, i.e., they can be realized for larger decision table using distributed computer system Fayyad \textit{et al.} (1996). The scalability property is also very advisable in data mining. Unfortunately, the eager classification methods are weakly scalable. As we recalled before, the time and memory complexity of existing algorithms does not make possible to apply rule base classification methods for very large decision table.

The most often used approach for rough set-based methods is characterized by the lack of scalability. We will show that some classification methods based on rough set theory can be modified using lazy learning algorithms to make them more scalable. The lazy rule-based classification diagram is presented in Fig. 2.

In other words, we will extract the set of decision rules covering the object \(x\) directly from data without explicit rule generation. We show that this diagram can work for the classification method described in previous section using the set of decision rules from \(\text{MinRules}(S, \lambda_{\text{max}}, \sigma_{\text{min}}, \alpha_{\text{min}})\). Formally, the problem is formulated as follows: \textit{given a decision table} \(S = (U, A \cup \{\text{dec}\})\) and a new object \(x\), \textit{find all (or almost all) decision rules from the set} \(\text{MatchRules}(S, x) = \{ r \in \text{MinRules}(S, \lambda_{\text{max}}, \sigma_{\text{min}}, \alpha_{\text{min}}) : x \text{ satisfies } r \}\).

In the case of too large number of such rules, one can find as many rules from \(\text{MatchRules}(S, x)\) as required.

The searching method for \(\text{MatchRules}(S, x)\), based on FP-growth algorithm, consists of the following steps:

\begin{itemize}
  \item Construction of the data structure called \(FDP(x)\) (Frequent Decision Pattern tree). This step requires only two data scanning passes:
\end{itemize}

\footnotesize{\textsuperscript{1}by large decision table we mean such tables containing at least \(10^6\) objects and \(10^2\) attributes.}
Rule selection

\[ \text{Decision table } \mathcal{S} \]

\[ \text{New object } x \rightarrow \text{Rule selection} \]

\[ \text{MatchedRules}(\mathcal{S}, x) \]

\[ \text{Classification} \rightarrow \text{dec}(x) \]

**Figure 2:** The lazy rule-based classification system

- The first scanning pass is required to calculate the frequencies of descriptors from \( \text{inf}_A(x) \). After the first data scan, these descriptors are ordered with respect to their frequencies. The low-frequent descriptors are useless in order to construct strong decision rules and can be removed. Let \( \text{DESC}(x) \) be the resulting list of frequent descriptors.

- In the second scanning pass, each training object \( u \) is converted into a list \( D(u) \) of frequent descriptors from \( \text{DESC}(x) \) that occur in \( \text{inf}_A(u) \), and then we insert the list \( D(u) \) into the data structure \( \text{FDP}(x) \).

- Generation of the set of frequent decision rules from \( \text{FDP}(x) \) by a recursive procedure. This step does not guarantee the minimality of the obtained rules (some rules are still reducible)

- Insert the obtained rules into a data structure called the **minimal rule tree** – denoted by \( \text{MRT}(x) \) – to get the set of irreducible decision rules. This data structure can be used to perform different voting strategy.

As we can see, the key concept in this method is the FDP tree structure. In fact, similarly to the original FP-tree, FDP is the prefix tree for the collection of ordered list of descriptors. But, unlike FP-tree, each node in FDP tree consists of four fields: \textit{descriptor name}, \textit{support}, \textit{class distribution} and \textit{node link}, where \textit{descriptor name} is the name of descriptor, \textit{support} is the number of training objects that contain all descriptors on the path from the root to the current node, \textit{class distribution} is the detail support for each decision class and \textit{node link} are used to create list of nodes of the same descriptor.

The detailed definitions and algorithms for this method were described in Kwiatkowski (2008). Because of the space limitation, we will illustrate the proposed method by the following example.

### 3.1 Example

Let us illustrate our concept for the \textit{whether} decision table presented in Figure 3 (left).
The test object induces four descriptors: $d_1 : a_1 = \text{sunny}$, $d_2 : a_2 = \text{mild}$, $d_3 : a_3 = \text{high}$ and $d_4 : a_4 = \text{TRUE}$. After the first data scan we have:

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
Descriptor: & (outlook=sunny) & (temp.=mild) & (hum.=high) & (windy=true) \\
\hline
Notation:   & d1              & d2              & d3              & d4              \\
Frequency:  & 5               & 6               & 7               & 6               \\
\hline
\end{tabular}
\caption{The occurrences of descriptors in the training data}
\end{table}

Thus we can fix the order of descriptors as follow: $\text{DESC}(x) = [d_3, d_2, d_4, d_1]$, and the training objects can be rewritten as follow as presented in Figure 3 (right).

The corresponding FDP tree for this collection of frequent descriptor lists is shown in the following figure:

In order to generate decision rules from the FDP tree, one can apply the FDP-growth algorithm which is the modification of FP-growth algorithm Han et al. (2000), Han and Kamber (2000). The readers can read more about the detail of this algorithm in Kwiatkowski (2008). In this example, one can obtain the following set of decision rules:

\begin{align*}
1 \quad & (\text{outlook} = \text{sunny}) \land (\text{hum.} = \text{high}) \Rightarrow \text{play} = \text{no} \\
2 \quad & (\text{outlet} = \text{sunny}) \land (\text{temp.} = \text{mild}) \land (\text{windy} = \text{TRUE}) \Rightarrow \text{play} = \text{yes} \\
3 \quad & (\text{outlet} = \text{sunny}) \land (\text{temp.} = \text{mild}) \land (\text{hum.} = \text{high}) \Rightarrow \text{play} = \text{no} \\
4 \quad & (\text{outlet} = \text{sunny}) \land (\text{hum.} = \text{high}) \land (\text{windy} = \text{TRUE}) \Rightarrow \text{play} = \text{no}
\end{align*}

One can see that this is not the set of irreducible decision rules, because, the rules number 3 and 4 are the extensions of rule nr 1. To reduce the set of rules one can use the additional data structure called MRT (minimal rule tree). In fact, MRT is the modification of FPMAX tree, the data structure for extraction
of maximal frequent patterns, presented in Grahne and Zhu (2003). The following figure illustrates the resulting MRT tree after inserting all decision rules.

After all steps, one can obtain two minimal decision rules:

1. \((\text{outlook} = \text{sunny}) \land (\text{hum.} = \text{high}) \Rightarrow \text{play} = \text{no}\)
2. \((\text{outlook} = \text{sunny}) \land (\text{temp.} = \text{mild}) \land (\text{windy} = \text{TRUE}) \Rightarrow \text{play} = \text{yes}\)

4 Experimental results

The FDP-growth algorithm was implemented and tested on data sets from UCI Machine Learning Repository. We compared the accuracy of FDP-growth algorithm with other lazy classifiers: IBk (nearest neighbors classifier) and LBR (Naive Bayes classifier) which are available in WEKA Witten and Frank (2005). All ex-
Table 2: The accuracy of three lazy classifiers over Poker-hand data.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>FDP_ growth</th>
<th>lBk</th>
<th>LBR</th>
</tr>
</thead>
<tbody>
<tr>
<td>train. size</td>
<td>Acc.</td>
<td>CPU time</td>
<td>Acc.</td>
</tr>
<tr>
<td>10k</td>
<td>0.631</td>
<td>98s</td>
<td>0.528</td>
</tr>
<tr>
<td>20k</td>
<td>0.695</td>
<td>187s</td>
<td>0.551</td>
</tr>
<tr>
<td>50k</td>
<td>0.786</td>
<td>460s</td>
<td>0.589</td>
</tr>
<tr>
<td>100k</td>
<td>0.876</td>
<td>924s</td>
<td>0.608</td>
</tr>
<tr>
<td>200k</td>
<td>0.861</td>
<td>1728s</td>
<td>0.649</td>
</tr>
<tr>
<td>500k</td>
<td>0.915</td>
<td>4431s</td>
<td>0.689</td>
</tr>
<tr>
<td>1000k</td>
<td>0.924</td>
<td>7906s</td>
<td>0.723</td>
</tr>
</tbody>
</table>

The experiments were done on PC with dual Processor Athlon X2 4000+ (2 x 2.1GHz) and 4GB RAM.

The experiment was performed on the poker-hand data set. This data set consists of 10 conditional attributes and 9 decision classes. The training data set consists of 25010 instances, while the test data contain 1000000 instances. In order to verify the scalability of the proposed solution, we switched the role of this data sets. The experiments were performed on training data sets of different sizes: 10000, 20000, 50000, 100000, 200000, 500000 and 1000000. The accuracy of classifiers were estimated on the sample of 1000 instances from the smaller data set and the detailed results are presented in the following table:

Figure 7 presents the plot of computation time for different training data sizes. One can see the scalability of the proposed solution.

5 Conclusions

We have presented a scalable lazy classifier which is is a rough set based classifier. We have modified the FP-growth algorithm to calculate the set of minimal decision rules for test objects. In fact, this algorithm can be used to calculate the object oriented reducts in decision table. Hence the proposed method can be applied also for eager learning. This method can be used for incremental learning.

Acknowledgement:

The research has been partially supported by the Ministry of Scientific Research and Higher Education of the Republic of Poland, research grant No. N N516 368334.

References


Building Scalable Lazy Rough Classifiers

**Figure 6**: Comparing accuracy of three lazy classifiers for poker-hand data.

**Figure 7**: Comparing the scalability of three lazy classifiers for poker-hand data.


Jiawei Han and Micheline Kamber (2000), Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems), Morgan Kaufmann.

Jiawei Han, Jian Pei, and Yiwen Yin (2000), Mining frequent patterns without candidate generation, in Weidong Chen, Jeffrey Naughton, and Philip A. Bernstein, editors, 2000 ACM SIGMOD Intl. Conference on Management of Data, pp. 1–12, ACM Press.


