Rule Induction on Data Sets with Set-Value Attributes

By

Luke Dodge

Submitted to the Department of Electrical Engineering and Computer Science and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Committee members

Dr. Jerzy Grzymala-Busse, Chairperson

Dr. Bo Luo

Dr. Arvin Agah

Date defended: ____________________

Rule Induction on Data Sets with Set-Value Attributes

Abstract

Data sets may have instances where multiple values are possible which are described as set-value attributes. The established LEM2 algorithm does not handle data sets with set-value attributes. To solve this problem, a parallel approach was used during LEM2's execution to avoid preprocessing data. Changing the creation of characteristic sets and attribute-value blocks to include all values for each case allows LEM2 to induce rules on data sets with set-value attributes. The ability to create a single local covering for set-value data sets increases the variety of data LEM2 can process.

Contents

1 Introduction **1**

2 General Definitions **2**

3 Preparing Data Sets for LEM2 **5**

4 Rule Induction Using LEM2 **12**

5 Data Sets with Set-Value Attributes **16**

6 Preparing Set-Value Data Sets for LEM2 **17**

6.1 Sequential Methods 17

6.2 Parallel Method 20

7 LEM2 Execution on Set-Value Data Sets **27**

7.1 Execution Walkthrough 27

7.2 Method Evaluation 41

8 Implementation **43**

9 Conclusion **46**

List of Tables

2.1 Example data set 3

3.1 Inconsistent data set 5

4.1 Computing rules for [(Flu, Yes)] 14

4.2 Computing rules for [(Flu, No)] 14

5.1 Example data set with set-value attributes 16

6.1 Case-splitting of data set with set-value attributes 18

6.2 Computing rules for split set-value attribute values (Flu, yes) 18

6.3 Attribute variables data set 19

6.4 Computing rules for data set with attribute variables 19

6.5 Inconsistent data set with set-value attributes 23

7.1 Randomly generated data set 28

7.2 Computing rules for the upper approximation of [(Letter, A)] 35

7.3 Computing rules for the upper approximation of [(Letter, B)] 36

7.4 Computing rules for the upper approximation of [(Letter, C)] 37

7.5 Computing rules for the upper approximation of [(Letter, C)] (continued) 38

7.6 Computing rules for the upper approximation of [(Letter, D)] 39

7.7 Description of evaluation data sets 41

7.8 Experimental evaluation results 42

9.1 Split cases of concept (Letter, B) 47

Chapter 1

Introduction

The LEM2 (Learning from Examples Module version 2) algorithm uses predefined concepts to induce rules. Unfortunately, the algorithm's implementation is limited to data sets which contain only single-value attributes. Fortunately, inducing rules from set-value information can be accomplished without modification of the established algorithm. Only the creation of characteristic sets, attribute-value blocks, and approximations will be updated.

Chapter 2 will define terms and concepts used throughout the paper. Chapter 3 will cover the necessary data set preparations required for the LEM2 algorithm. Rule induction using LEM2 will be shown in Chapter 4. Chapter 5 will describe set-value attributes. The process for preparing set-value data for LEM2 will be given in Chapter 6. Chapter 7 will show how to induce rules on set-value data using the LEM2 algorithm and evaluate the different methods. Lastly, in Chapter 8 a walk-through of an implementation of LEM2 capable of handling set-value attributes will be given.

Chapter 2

General Definitions

Rough Set Theory A brief foundation of Rough Set theory is needed to establish a common set of terms. For an in-depth explanation of Rough Set theory see *Rough set theory and its applications* [8]. *Attributes* are independent variables while the *decision* is a variable dependent on the set of attributes. A will represent the set of all attributes and v will represent the corresponding value of the attribute. Cases, represented by x, are numbered rows in the decision table. U will represent the Universe or set of all cases.

Decision Table Data sets can be represented in the form of a *decision table*. A decision table is comprised of a number of cases (rows) which contain values belonging to either the set of attributes or the decision. Table 2.1 is an example data set displayed as a decision table. Each case represents the symptoms of a patient and the decision represents the flu diagnosis. Cases have a value for each attribute (Temperature, Headache, Weakness, Nausea) and a corresponding decision value (Flu).

Table 2.1: Example data set

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

Rules The value of the decision given a set of attributes can be determined using *rules*. A rule is a set of conditions (attribute-value pairs) which indicate a corresponding decision-value pair (action). A rule can be expressed as a statement of conditions to an action.

$$if\ (attribute_1,\ value_1)\ and\ (attribute_2,\ value_2)\ and...and\ (attribute_n,\ value_n)\ then\ (decision,\ value)$$

These statements are represented in the following form where d is the decision, $a_1, a_2, ...a_n$ are attributes and $v_1, v_2, ...v_n$ are attribute values:

$$(a_1,\ v_1) \wedge (a_2,\ v_2) \wedge ...(a_n,\ v_n) \longrightarrow (d,\ v).$$

For example, a rule which identifies case 1 of the decision table in Table 2.1 is shown below:

$$(Temperature,\ very\_high) \wedge (Headache,\ yes) \wedge (Weakness,\ yes) \wedge (Nausea,\ no) \longrightarrow (Flu,\ yes).$$

The rule can be interpreted as such: at the time of data collection the patient's temperature was very high, and the patient had a headache and experienced weakness. The diagnosis based on the information was the patient had the flu.

Concept A *concept* is the set of all cases with the same decision value [5]. Each concept is a subset of the universe. For Table 2.1, the concept of all cases affected by Flu is the case set {1, 2, 4, 5}. Each case in this set has a value of *Flu* being *yes*. Similarly, the concept of all cases not affected by Flu is the case set {3, 6, 7}. Each case in this set has a value of *Flu* being *no*.

Rule Set A *rule set* is a set of rules used to identify a decision value. A case is covered by a rule, if and only if, every condition of the rule is satisfied by the case. The set of cases covered by a rule set is the union of all cases covered by the rules in the rule set. Ideally we determine a rule set which covers an entire concept. A rule set completely covers a concept, if and only if, each case in the concept is covered by a rule in the rule set.

Local Covering Using a rule set, a *local covering* describes a concept. Let B be a non-empty concept. Let T be a set of attribute-value pairs. Let $\mathbb{T}$ be a non-empty collection of non-empty sets of attribute-value pairs. $\mathbb{T}$ is a local covering, if and only if, the following conditions are satisfied [3]:

1. Each member T of $\mathbb{T}$ is a minimal complex of B. T is a *minimal complex* of B, if and only if, B depends on T and no proper subset T' exists such that B depends on T' [2],
2. $B = \bigcup_{t \in \mathbb{T}} [T]$ where t is an attribute-value pair (a,v),
3. $\mathbb{T}$ has the smallest possible number of members (minimal).

Chapter 3

Preparing Data Sets for LEM2

Consistency The LEM2 algorithm requires a *consistent* data set as an input. A consistent data set contains no indistinguishable cases which belong to more than one concept. A case is *indistinguishable* from another case if each and every attribute value for the cases are equal. An example of a consistent data set can be seen in Table 2.1. An example of an inconsistent data set can be seen in the Table 3.1 where cases 5, 6 and cases 3, 7 are indistinguishable.

Table 3.1: Inconsistent data set

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

Attribute-value Blocks LEM2 uses *blocks* of attribute-value pairs to create rules. A block of an attribute-value pair is the set of cases which contain a specific attribute value. Let $a \in A$, and $t = (a,\ v)$ be an attribute-value pair. A block of t, denoted by $[t]$, is a set of all cases from U for which attribute a has value v [3].

Attribute-value Blocks Example The attribute-value blocks of the data set shown in Table 2.1 are as follows:

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

Characteristic Sets Definition A *characteristic set* is the intersection of cases which contain a specific attribute value for an attribute corresponding to a particular case. More specifically, given B is a subset of A, the characteristic set $K_B(x)$ is the intersection of blocks of attribute-value pairs $(a,\ v)$ for all attributes a from B for which $a(x)$ is specified and $a(x) = v$ [3]

$$K_B(x) = \bigcap_{a \in B} K_a(x) \ where\ K_a(x) = [(a, v)].$$

Consistency of a data set can be determined by comparing the characteristic set of each case to the concepts of the data set. If any characteristic set is not a subset of a concept, the data set is considered inconsistent and an approximation should to be created.

Characteristic Sets / Consistent Data Set Example For example, the intersection of blocks for case 1 in Table 2.1 are as follows:

$$[(Temperature,\ High)] \cap [(Headache,\ Yes)] \cap [(Weakness,\ Yes)] \cap [(Nausea,\ No)]$$
$$= \{1\} \cap \{1,\ 2,\ 4\} \cap \{1,\ 4,\ 5,\ 7\} \cap \{1,\ 3,\ 5,\ 6,\ 7\}$$
$$= \{1\}.$$

The characteristic sets of the consistent data set shown in Table 2.1 are as follows:

$$\begin{aligned}
K_A(1) &= \{1\} \cap \{1,\ 2,\ 4\} \cap \{1,\ 4,\ 5,\ 7\} \cap \{1,\ 3,\ 5,\ 6,\ 7\} \\
&= \{1\}, \\
K_A(2) &= \{2,\ 5,\ 6\} \cap \{1,\ 2,\ 4\} \cap \{2,\ 3,\ 6\} \cap \{2,\ 4\} \\
&= \{2\}, \\
K_A(3) &= \{3,\ 4,\ 7\} \cap \{3,\ 5,\ 6,\ 7\} \cap \{2,\ 3,\ 6\} \cap \{1,\ 3,\ 5,\ 6,\ 7\} \\
&= \{3\}, \\
K_A(4) &= \{3,\ 4,7\ \} \cap \{1,\ 2,\ 4\} \cap \{1,\ 4,\ 5,\ 7\} \cap \{2,\ 4\} \\
&= \{4\}, \\
K_A(5) &= \{2,\ 5,\ 6\} \cap \{3,\ 5,\ 6,\ 7\} \cap \{1,\ 4,\ 5,\ 7\} \cap \{1,\ 3,\ 5,\ 6\ ,7\} \\
&= \{5\}, \\
K_A(6) &= \{2,\ 5,\ 6\} \cap \{3,\ 5,\ 6,\ 7\} \cap \{2,\ 3,\ 6\} \cap \{1,\ 3,\ 5,\ 6,\ 7\} \\
&= \{6\}, \\
K_A(7) &= \{3,\ 4,\ 7\} \cap \{3,\ 5,\ 6,\ 7\} \cap \{1,\ 4,\ 5,\ 7\} \cap \{1,\ 3,\ 5,\ 6,\ 7\} \\
&= \{7\}.
\end{aligned}$$

Given a consistent data set, each characteristic set is a subset of one and only one concept.

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

$$K_A(1) = \{1\} \subseteq \{1,\ 2,\ 4,\ 5\},$$
$$K_A(2) = \{2\} \subseteq \{1,\ 2,\ 4,\ 5\},$$
$$K_A(3) = \{3\} \subseteq \{3,\ 6,\ 7\},$$
$$K_A(4) = \{4\} \subseteq \{1,\ 2,\ 4,\ 5\},$$
$$K_A(5) = \{5\} \subseteq \{1,\ 2,\ 4,\ 5\},$$
$$K_A(6) = \{6\} \subseteq \{3,\ 6,\ 7\},$$
$$K_A(7) = \{7\} \subseteq \{3,\ 6,\ 7\}.$$

Inconsistent Data Set Example A data set is inconsistent if any characteristic set is not a subset of a concept. The characteristic sets of the inconsistent data set shown in Table 3.1 are as follows:

$$\begin{aligned}
K_A(1) &= \{1\} \cap \{1,\ 2,\ 4\} \cap \{1,\ 3,\ 5,\ 6,\ 7\}\\
&= \{1\},\\
K_A(2) &= \{2,\ 5,\ 6\} \cap \{1,\ 2,\ 4\} \cap \{2,\ 4\}\\
&= \{2\},\\
K_A(3) &= \{3,\ 4,\ 7\} \cap \{3,\ 5,\ 6,\ 7\} \cap \{1,\ 3,\ 5,\ 6,\ 7\}\\
&= \{3,\ 7\},\\
K_A(4) &= \{3,\ 4,\ 7\} \cap \{1,\ 2,\ 4\} \cap \{2,\ 4\}\\
&= \{4\},\\
K_A(5) &= \{2,\ 5,\ 6\} \cap \{3,\ 5,\ 6,\ 7\} \cap \{1,\ 3,\ 5,\ 6,\ 7\}\\
&= \{5,\ 6\},\\
K_A(6) &= \{2,\ 5,\ 6\} \cap \{3,\ 5,\ 6,\ 7\} \cap \{1,\ 3,\ 5,\ 6,\ 7\}\\
&= \{5,\ 6\},\\
K_A(7) &= \{3,\ 4,\ 7\} \cap \{3,\ 5,\ 6,\ 7\} \cap \{1,\ 3,\ 5,\ 6,\ 7\}\\
&= \{3,\ 7\}.
\end{aligned}$$

Inconsistencies identified in the data set shown in Table 3.1 are as follows:

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

$$K_A(5) = \{5,\ 6\} \nsubseteq \{1,\ 2,\ 4,\ 5\},$$
$$K_A(5) = \{5,\ 6\} \nsubseteq \{3,\ 6,\ 7\},$$
$$K_A(6) = \{5,\ 6\} \nsubseteq \{1,\ 2,\ 4,\ 5\},$$
$$K_A(6) = \{5,\ 6\} \nsubseteq \{3,\ 6\ ,7\}.$$

Cases 5 and 6 belong to more than one concept, meaning the data set is inconsistent.

Approximations Definition An *approximation* of a consistent data set can be created from an inconsistent data set. Approximations are a way to represent a concept in a consistent manner. The upper approximation is the union of characteristic sets and the lower approximation is the largest definable set containing the concept [5]. Rules induced from lower approximations are described as *certain* and rules induced from upper approximations are described as *possible*. Let $\overline{B}X$ be the upper approximation and $\underline{B}X$ be the lower approximation.

$$\underline{B}X = \cup\{K_B(x)|x \in X, K_B(x) \subseteq X\},$$
$$\overline{B}X = \cup\{K_B(x)|x \in X\}.$$

Approximations Example The approximations for the data set shown in Table 3.1 are as follows:

$$\begin{aligned}
\overline{A}\{3,\ 6,\ 7\} &= K_A(3) \cup K_A(5) \cup K_A(6) \cup K_A(7) \\
&= \{3,\ 7\} \cup \{5,\ 6\} \cup \{5,\ 6\} \cup \{3,\ 7\} \\
&= \{3,\ 5,\ 6,\ 7\}, \\
\overline{A}\{1,\ 2,\ 4,\ 5\} &= K_A(1) \cup K_A(2) \cup K_A(4) \cup K_A(5) \cup K_A(6) \\
&= \{1\} \cup \{2\} \cup \{4\} \cup \{5,\ 6\} \cup \{5,\ 6\} \\
&= \{1,\ 2,\ 4,\ 5,\ 6\},
\end{aligned}$$

$$\begin{aligned}
\underline{A}\{3,\ 6,\ 7\} &= K_A(3) \cup K_A(7) \\
&= \{3,\ 7\} \cup \{3,\ 7\} \\
&= \{3,\ 7\}, \\
\underline{A}\{1,\ 2,\ 4,\ 5\} &= K_A(1) \cup K_A(2) \cup K_A(4) \\
&= \{1\} \cup \{2\} \cup \{4\} \\
&= \{1,\ 2,\ 4\}.
\end{aligned}$$

Chapter 4

Rule Induction Using LEM2

Because the LEM2 algorithm will not be modified to enable processing of set-value attributes, only a brief overview will be provided. For an in-depth explanation of the LEM2 algorithm see *Rough Set Theory with Applications to Data Mining* [3].

LEM2 Given a decision table, the LEM2 algorithm will compute a single local covering in the form of a rule set for each concept [1]. For a set X, $|X|$ denotes the cardinality of X.

```
Procedure LEM2
(input: a set $B$,
output: single local covering $\mathbb{T}$ of set $B$);
begin
    $G := B$;
    $\mathbb{T} := \emptyset$;
    while $G \neq \emptyset$
        begin
            $T := \emptyset$;
            $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$;
            while $T = \emptyset$ or $[T] \nsubseteq B$
```

```
                **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** $T := T - \{t\}$;
            $\mathbb{T} := \mathbb{T} \cup \{T\}$;
            $G := B - \bigcup_{T \in \mathbb{T}} [T]$;
        **end** {while};
        **for** each $T \in \mathbb{T}$ **do**
            **if** $\bigcup_{S \in \mathbb{T} - \{T\}} [S] = B$ **then** $\mathbb{T} := \mathbb{T} - \{T\}$;
**end** {procedure}
```

Start with a consistent data set (or it's approximation) and create attribute-value blocks. A goal equal to the set of cases for a selected concept is created. Loop through the blocks, selecting the best block for the goal based on LEM2 heuristic criteria. Each loop will select an attribute which may be used in a rule.

Candidate rules will be built by adding attribute-value blocks together until the candidate rule forms a subset of the selected concept. The cases covered by the candidate rule are removed from the goal and the loop continues until the entire goal has been met. Once a candidate rule covers a subset of the selected concept, the rule is finished and the goal is reduced based on the cases covered by the candidate rule. This process continues until the entire concept has been covered. All rules are reduced to their simplest form and redundant rules are removed.

Table 4.1 and Table 4.2 are examples of determining rules for the data set shown in Table 2.1. The • symbol indicates the attribute value was selected as a condition for the candidate rule.

Table 4.1: Computing rules for [(Flu, Yes)]

$(a,v) = t$	$[(a,v)]$	{1,2,4,5}	{5}	{5}	{5}
[(Temperature, very_high)]	{1}	{1}	∅	-	-
[(Temperature, high)]	{2,5,6}	{2,5}	{5} •	-	-
[(Temperature, normal)]	{3,4,7}	{4}	∅	-	-
[(Headache, yes)]	{1,2,4}	{1,2,4} •	∅	∅	-
[(Headache, no)]	{3,5,6,7}	{5}	{5}	{5} •	-
[(Weakness, yes)]	{1,4,5,7}	{1,4,5}	{5}	{5}	{5} •
[(Weakness, no)]	{2,3,6}	{2}	∅	∅	∅
[(Nausea, yes)]	{2,4}	{2,4}	∅	∅	∅
[(Nausea, no)]	{1,3,5,6,7}	{1,5}	{5}	{5}	{5}
	T	{1,2,4}	{2,5,6}	{5,6}	{5}
	$\mathbb{T}$	{{1,2,4}}	{{1,2,4}}	{{1,2,4}}	{{1,2,4},{5}}

Table 4.2: Computing rules for [(Flu, No)]

(a, v) = t	[(a, v)]	{3,6,7}	{3,6,7}	{3,6,7}	{6}	{6}	{6}
[(Temp, very_high)]	{1}	∅	∅	∅	∅	-	-
[(Temp, high)]	{2,5,6}	{6}	{6}	{6}	{6} •	-	-
[(Temp, normal)]	{3,4,7}	{3,7}	{3,7}	{3,7} •	∅	-	-
[(Headache, yes)]	{1,2,4}	∅	-	-	∅	∅	∅
[(Headache, no)]	{3,5,6,7}	{3,6,7} •	-	-	{6}	{6}	{6} •
[(Weakness, yes)]	{1,4,5,7}	{7}	{7}	{7}	∅	∅	-
[(Weakness, no)]	{2,3,6}	{3,6}	{3,6}	{3,6}	{6}	{6} •	-
[(Nausea, yes)]	{2,4}	∅	∅	-	∅	∅	∅
[(Nausea, no)]	{1,3,5,6,7}	{3,6,7}	{3,6,7} •	-	{6}	{6}	{6}
	T	{3,5,6,7}	{3,5,6,7}	{3,7}	{2,5,6}	{2,6}	{6}
	$\mathbb{T}$	∅	∅	{{3,7}}	{{3,7}}	{{3,7}}	{{3,7},{6}}

Rules

1. $(Headache,\ yes) \longrightarrow (Flu,\ yes)$,

2. $(Temperature,\ high) \wedge (Headache,\ no) \wedge (Weakness,\ yes) \longrightarrow (Flu,\ yes)$,

3. $(Headache,\ no) \wedge (Nausea,\ no) \wedge (Temperature,\ normal) \longrightarrow (Flu,\ no)$,

4. $(Temperature,\ high) \wedge (Weakness,\ no) \wedge (Headache,\ no) \longrightarrow (Flu,\ no)$.

These rules can be simplified by dropping $(Headache,\ no)$ from Rule 2, $(Nausea,\ no)$ from Rule 3, and $(Temperature,\ high)$ from Rule 4. LEM2 will produce a simplified rule set (see below) for the data set shown in Table 2.1.

1. $(Headache,\ yes) \longrightarrow (Flu,\ yes)$,

2. $(Temperature,\ high) \wedge (Weakness,\ yes) \longrightarrow (Flu,\ yes)$,

3. $(Nausea,\ no) \wedge (Temperature,\ normal) \longrightarrow (Flu,\ no)$,

4. $(Weakness,\ no) \wedge (Headache,\ no) \longrightarrow (Flu,\ no)$.

Chapter 5

Data Sets with Set-Value Attributes

Set-Value Attribute So far we have dealt only with data sets where each attribute contains only one value per case. Even ranges are represented as symbolic values instead of a set of numeric values. For example, the numeric value 3.2 can be represented by the symbolic value 3..4 (3 to 4) through the process of discretization. For an in-depth explanation of the Modified LEM2 algorithm which handles discretization see *A Local Version of the MLEM2 Algorithm for Rule Induction* [6]. When attributes contain more than one value per case, these types of values can be defined as set-value attributes. Table 5.1 shows a data set with set-value attributes.

Table 5.1: Example data set with set-value attributes

Case	Attributes			Decision
	Temperature	Headache	Cough	Flu
1	high, very_high	yes	no	yes
2	high	no	yes	yes
3	very_high	no	no	no
4	normal, high	yes	yes	maybe

A set-value for an attribute means all values in the set are valid for the case. For example, the *Temperature* value for case 1 in the data set shown in Table 5.1 was either *high* or *very high*, but not both, during data collection. These values can be interpreted as being connected with an Exclusive Or clause.

Chapter 6

Preparing Set-Value Data Sets for LEM2

There are two approaches for creating a consistent set of rules from a data set with set-value attributes. A rule set is consistent, if and only if, every rule in the rule set is consistent with the data set [4]. The first approach is sequential (preprocessing the data) and will not modify LEM2, but instead modify the data set to contain only single-value attributes which is compatible with the algorithm. The second approach is parallel (processing set values during induction) and will modify LEM2's definitions and interpretations to handle set values.

6.1 Sequential Methods

Sequential methods to handle set-value attributes convert the original data set, with set-value attributes, into a data with only single-value attributes prior to rule induction.

Split Set-Value Attributes To represent multiple values occurring using single-value attributes requires cases be split into multiple sub-cases. Each combination of attribute values for a set will be represented as a new sub-case. The increase in size of the data set will be considerable, potentially by many orders of magnitude. If even the simple data set shown in Table 5.1 were split in this manner, a fifty percent increase in the number of cases would result (see Table 6.1).

Table 6.1: Case-splitting of data set with set-value attributes

Case	Attributes			Decision
	Temperature	Headache	Cough	Flu
1_1	high	yes	no	yes
1_2	very_high	yes	no	yes
2	high	no	yes	yes
3	very_high	no	no	no
4_1	normal	yes	yes	maybe
4_2	high	yes	yes	maybe

Table 6.2: Computing rules for split set-value attribute values (Flu, yes)

(a, v) = t	[(a,v)]	$\{1_1,1_2,2\}$	$\{1_1,2\}$	$\{1_1,1_2\}$	$\{1_1,1_2\}$
[(Temperature, high)]	$\{1_1,2,4_2\}$	$\{1_1,2\}$ •	-	$\{1_1\}$	$\{1_1\}$
[(Temperature, very_high)]	$\{1_2,3\}$	$\{1_2\}$	-	$\{1_2\}$	$\{1_2\}$
[(Temperature, normal)]	$\{4_1\}$	$\emptyset$	-	$\emptyset$	$\emptyset$
[(Headache, yes)]	$\{1_1,1_2,4_1,4_2\}$	$\{1_1,1_2\}$	$\{1_1\}$	$\{1_1,1_2\}$	$\{1_1,1_2\}$ •
[(Headache, no)]	$\{2,3\}$	$\{2\}$	$\{2\}$ •	$\emptyset$	$\emptyset$
[(Cough, no)]	$\{1_1,1_2,3\}$	$\{1_1,1_2\}$	$\{1_1\}$	$\{1_1,1_2\}$ •	-
[(Cough, yes)]	$\{2,4_1,4_2\}$	$\{2\}$	$\{2\}$	$\emptyset$	-
	T	$\{1_1,2,4_2\}$	$\{2\}$	$\{1_1,1_2,3\}$	$\{1_1,1_2\}$
	$\mathbb{T}$	$\emptyset$	$\{\{2\}\}$	$\{\{2\}\}$	$\{\{2\},\{1_1,1_2\}\}$

Rules

1. $(Temperature,\ high) \wedge (Headache, no) \longrightarrow (Flu,\ yes)$,

2. $(Cough,\ no) \wedge (Headache, yes) \longrightarrow (Flu,\ yes)$,

3. $(Cough,\ no) \wedge (Temperature, very\_high) \longrightarrow (Flu,\ no)$,

4. $(Cough,\ yes) \wedge (Headache, yes) \longrightarrow (Flu,\ maybe)$.

Replace Set Values with Attribute Variables This sequential method substitutes attribute variables for attribute-values containing sets. For example, any instance of $(Temperature, \{high,\ very\_high\})$ is replaced with the single attribute variable $temp_1$. After rules have been created, attribute variables are replaced with the corresponding set of attribute values.

As an example, for the data set shown in Table 5.1, $(Temperature, \{high, very\_high\})$ becomes the variable $temp_1$ and $(Temperature, \{normal, high\})$ becomes the variable $temp_2$. The data set now contains only single-value attributes. The results of these substitutions can be seen in Table 6.3.

Table 6.3: Attribute variables data set

Case	Attributes			Decision
	Temperature	Headache	Cough	Flu
1	$temp_1$	yes	no	yes
2	high	no	yes	yes
3	very_high	no	no	no
4	$temp_2$	yes	yes	maybe

Table 6.4: Computing rules for data set with attribute variables

(a, v) = t	[(a, v)]	{1,2}	{1}	{3}	{4}
[(Temperature, high)]	{2}	{2} •	∅	∅	∅
[(Temperature, very_high)]	{3}	∅	∅	{3} •	∅
[(Temperature, $temp_1$)]	{1}	{1}	{1} •	∅	∅
[(Temperature, $temp_2$)]	{4}	∅	∅	∅	{4} •
[(Headache, yes)]	{1,4}	{1}	{1}	∅	{4}
[(Headache, no)]	{2,3}	{2}	∅	{3}	∅
[(Nausea, yes)]	{2,4}	{2}	∅	∅	{4}
[(Nausea, no)]	{1,3}	{1}	{1}	{3}	∅
	T	{2}	{1}	{3}	{4}
	$\mathbb{T}$	{2}	{{2},{1}}	{{3}}	{{4}}

Rules After the rule set has been created, each attribute variable is replaced with the original set of attributes values. An attribute-value set as part of a condition is represented as the union of each attribute-value in the set and logically combined with Exclusive Or.

1. $(Temperature,\ high) \longrightarrow (Flu,\ yes)$,
2. $(Temperature,\ temp_1) \longrightarrow (Flu,\ yes)$ becomes
   $((Temperature,\ high) \vee (Temperature,\ very\_high)) \longrightarrow (Flu,\ yes)$,
3. $(Temperature,\ very\_high) \longrightarrow (Flu,\ no)$,
4. $(Temperature,\ temp_2) \longrightarrow (Flu,\ maybe)$ becomes
   $((Temperature,\ normal) \vee (Temperature,\ high)) \longrightarrow (Flu,\ maybe)$.

The downside to replacing set values with attribute variables is the potential for broad rules with inconsistencies. For example, variables $temp_1$ and $temp_2$ both use $(Temperature,\ high)$ as an attribute value. The overlap causes an inconsistency between Rule 1, Rule 2, and Rule 4. Each rule covers cases in more than one concept for the data set shown in 5.1, meaning the rule set is inconsistent.

6.2 Parallel Method

Ideally a single local covering of a data set with set-value attributes can be created without increasing the complexity of LEM2's execution. Doing so requires the modification of how the data set is processed instead of a costly change to the data set. Unlike sequential methods, no data preprocessing is used with parallel methods. Instead the LEM2 algorithm induces rules directly from a data set with set-value attributes.

Concept Determining concepts for a data set with set-value attributes is no different than determining concepts for single-value data sets. The concepts for the data set shown in Table 5.1 are as follows:

$$[(Flu,\ yes)] = \{1,\ 2\},$$
$$[(Flu,\ no)] = \{3\},$$
$$[(Flu,\ maybe)] = \{4\}.$$

Consistency To check set-value data sets for consistency we need to extend the definition of characteristic sets. By changing the way characteristic sets combine attribute-value blocks, the existing method can be used to determine consistency.

Attribute-Value Block Creation The interpretation of attribute-value blocks will change slightly to accommodate changes made to characteristic sets. Attribute-value blocks of data sets which contain set-value attributes are created the same as for data sets with only single-value attributes. An important difference between single-value attributes and set-value attributes is the existence of cases in more than one attribute-value block. Cases in a set-value attribute may exist in more than one attribute-value block, while cases in a single-value attribute exist in a maximum of one attribute-value block. This will not affect execution of the LEM2 algorithm.

Attribute-Value Block Example The attribute-value blocks of the set-value data set shown in Table 5.1 are as follows:

$$[(Temperature,\ high)] = \{1,\ 2,\ 4\},$$
$$[(Temperature,\ very\_high)] = \{1,\ 3\},$$
$$[(Temperature,\ normal)] = \{4\},$$
$$[(Headache,\ yes)] = \{1,\ 4\},$$
$$[(Headache,\ no)] = \{2,\ 3\},$$
$$[(Cough,\ yes)] = \{2,\ 4\},$$
$$[(Cough,\ no)] = \{1,\ 3\}.$$

Characteristic Set Creation Characteristic sets must account for cases belonging to multiple attribute-value blocks for the same attribute. The union of each block intersecting the case will be used instead of a single attribute-value block containing the case. Given B is a subset of A and $a(x)$ is a set of all values of a for case x, the updated definition of characteristic sets is as follows:

$$K_B(x) = \bigcap_{a \in B} K_a(x) \textit{ where } K_a(x) = \cup\{[(a,v)] : v \in a(x)\}.$$

Characteristic Set Example The characteristic sets for the set-value data set shown in Table 5.1 are as follows:

$$\begin{aligned}
K_A(1) &= (\{1, 2, 4\} \cup \{1, 3\}) \cap \{1, 4\} \cap \{1, 3\} \\
&= \{1\}, \\
K_A(2) &= \{1, 2, 4\} \cap \{2, 3\} \cap \{2, 4\} \\
&= \{2\}, \\
K_A(3) &= \{1, 3\} \cap \{2, 3\} \cap \{1, 3\} \\
&= \{3\}, \\
K_A(4) &= (\{4\} \cup \{1, 2, 4\}) \cap \{1, 4\} \cap \{2, 4\} \\
&= \{4\}.
\end{aligned}$$

Inconsistent Set-Value Data Set Example Approximations become increasingly important for data sets with set-value attributes due to the greater possibility of inconsistent cases. See Table 6.5 for an example of an inconsistent set-value data set.

Table 6.5: Inconsistent data set with set-value attributes

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

Attribute-value blocks for the data set shown in Table 6.5 are as follows:

$$\begin{aligned}
[(Temperature,\ very\_high)] &= \{1,\ 5\},\\
[(Temperature,\ high)] &= \{1,\ 2,\ 6,\ 7\},\\
[(Temperature,\ normal)] &= \{2,\ 3,\ 4\},\\
[(Headache,\ yes)] &= \{2,\ 4,\ 6,\ 7\},\\
[(Headache,\ no)] &= \{1,\ 3,\ 5\},\\
[(Nausea,\ yes)] &= \{1,\ 5,\ 6,\ 7\},\\
[(Nausea,\ no)] &= \{2,\ 3,\ 4\}.
\end{aligned}$$

Below we can see the characteristic sets for Table 6.5 and identify inconsistent cases 6 and 7.

$$\begin{aligned}
K_A(1) &= (\{1, 2, 6, 7\} \cup \{1, 5\}) \cap \{1, 3, 5\} \cap \{1, 5, 6, 7\} \\
&= \{1, 5\}, \\
K_A(2) &= (\{2, 3, 4\} \cup \{1, 2, 6, 7\}) \cap \{2, 4, 6, 7\} \cap \{2, 3, 4\} \\
&= \{2, 4\}, \\
K_A(3) &= \{2, 3, 4\} \cap \{1, 3, 5\} \cap \{2, 3, 4\} \\
&= \{3\}, \\
K_A(4) &= \{2, 3, 4\} \cap \{2, 4, 6, 7\} \cap \{2, 3, 4\} \\
&= \{2, 4\}, \\
K_A(5) &= \{1, 5\} \cap \{1, 3, 5\} \cap \{1, 5, 6, 7\} \\
&= \{1, 5\}, \\
K_A(6) &= \{1, 2, 6, 7\} \cap \{2, 4, 6, 7\} \cap \{1, 5, 6, 7\} \\
&= \{6, 7\}, \\
K_A(7) &= \{1, 2, 6, 7\} \cap \{2, 4, 6, 7\} \cap \{1, 5, 6, 7\} \\
&= \{6, 7\}.
\end{aligned}$$

Approximation The LEM2 algorithm can be executed using either the upper or lower concept approximations. Approximations for set-value data sets are created using the updates to the interpretation of attribute-value blocks and creation of characteristic sets.

Set-Value Approximation Example The approximations created for the data set shown in Table 6.5 are as follows:

$$\begin{aligned}
\bar{A}\{1,\ 5,\ 6\} &= K_A(1) \cup K_A(5) \cup K_A(6) \cup K_A(7) \\
&= \{1,\ 5\} \cup \{1,\ 5\} \cup \{6,\ 7\} \cup \{6,\ 7\} \\
&= \{1,\ 5,\ 6,\ 7\}, \\
\bar{A}\{2,\ 3,\ 4,\ 6,\ 7\} &= K_A(2) \cup K_A(3) \cup K_A(4) \cup K_A(6) \cup K_A(7) \\
&= \{2,\ 4\} \cup \{3\} \cup \{2,\ 4\} \cup \{6,\ 7\} \cup \{6,\ 7\} \\
&= \{2,\ 3,\ 4,\ 6,\ 7\},
\end{aligned}$$

$$\begin{aligned}
\underline{A}\{1,\ 5,\ 6\} &= K_A(1) \cup K_A(5) \\
&= \{1,\ 5\} \cup \{1,\ 5\} \\
&= \{1,\ 5\}, \\
\underline{A}\{2,\ 3,\ 4,\ 7\} &= K_A(2) \cup K_A(3) \cup K_A(4) \\
&= \{2,\ 4\} \cup \{3\} \cup \{2,\ 4\} \\
&= \{2,\ 3,\ 4\}.
\end{aligned}$$

Chapter 7

LEM2 Execution on Set-Value Data Sets

A modification of characteristic sets extends the use of the LEM2 algorithm to data sets containing set-value attributes. Because LEM2 relies on attribute-value blocks to create rules, the new interpretation of attribute-value block creation used for checking consistency can be used. This can be seen in the execution of the algorithm on the approximation created in Chapter Six on Table 6.5.

7.1 Execution Walkthrough

Table 7.1 is an example data set generated randomly with a mixture of attribute types. There exists an attribute of only single values (Single), an attribute of set-values (Multiple), and an attribute with both set-values and single values (Mixed). This data set will be used to demonstrate the described process with simulated real-world data.

Table 7.1: Randomly generated data set

		Attributes		Decision
Case	Single	Multiple	Mixed	Letter
1	S9	M2,M6	X3	D
2	S9	M0,M2,M7,M9	X1	C
3	S2	M1,M2,M4,M7	X2,X3	B
4	S1	M6,M7,M10	X1,X3	C
5	S10	M0,M5	X1	C
6	S4	M0,M2,M3,M10	X1	A
7	S1	M4,M9	X1	A
8	S1	M4,M5,M8	X1,X3	D
9	S9	M5,M9	X3	B
10	S1	M3,M9	X1,X2,X3	C

Attribute-value Blocks The attribute-value blocks of the data set shown in Table 7.1 are as follows:

$$[(Single,\ S1)] = \{4,\ 7,\ 8,\ 10\},$$

$$[(Single,\ S2)] = \{3\},$$

$$[(Single,\ S4)] = \{6\},$$

$$[(Single,\ S9)] = \{1,\ 2,\ 9\},$$

$$[(Single,\ S10)] = \{5\},$$

$$\begin{aligned}
[(Multiple,\ M0)] &= \{2,\ 5,\ 6\},\\
[(Multiple,\ M1)] &= \{3\},\\
[(Multiple,\ M2)] &= \{1,\ 2,\ 3,\ 6\},\\
[(Multiple,\ M3)] &= \{6,\ 10\},\\
[(Multiple,\ M4)] &= \{3,\ 7,\ 8\},\\
[(Multiple,\ M5)] &= \{5,\ 8,\ 9\},\\
[(Multiple,\ M6)] &= \{1,\ 4\},\\
[(Multiple,\ M7)] &= \{2,\ 3,\ 4\},\\
[(Multiple,\ M8)] &= \{8\},\\
[(Multiple,\ M9)] &= \{2,\ 7,\ 9,\ 10\},\\
[(Multiple,\ M10)] &= \{4,\ 6\},\\
[(Mixed,\ X1)] &= \{2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10\},\\
[(Mixed,\ X2)] &= \{3,\ 10\},\\
[(Mixed,\ X3)] &= \{1,\ 3,\ 4,\ 8,\ 9,\ 10\}.
\end{aligned}$$

Characteristic Sets The attribute-value block combinations of the attribute *Multiple* in the data set shown in Table 7.1 are as follows:

$$\begin{aligned}
K_{Multiple}(1) &= \{1, 2, 3, 6\} \cup \{1, 4\} \\
&= \{1, 2, 3, 4, 6\}, \\
K_{Multiple}(2) &= \{2, 5, 6\} \cup \{1, 2, 3, 6\} \cup \{2, 3, 4\} \cup \{2, 7, 9, 10\} \\
&= \{1, 2, 3, 4, 5, 6, 7, 9, 10\}, \\
K_{Multiple}(3) &= \{3\} \cup \{1, 2, 3, 6\} \cup \{3, 7, 8\} \cup \{2, 3, 4\} \\
&= \{1, 2, 3, 4, 6, 7, 8\}, \\
K_{Multiple}(4) &= \{1, 4\} \cup \{2, 3, 4\} \cup \{4, 6\} \\
&= \{1, 2, 3, 4, 6\}, \\
K_{Multiple}(5) &= \{2, 5, 6\} \cup \{5, 8, 9\} \\
&= \{2, 5, 6, 8, 9\}, \\
K_{Multiple}(6) &= \{2, 5, 6\} \cup \{1, 2, 3, 6\} \cup \{6, 10\} \cup \{4, 6\} \\
&= \{1, 2, 3, 4, 5, 6, 10\}, \\
K_{Multiple}(7) &= \{3, 7, 8\} \cup \{2, 7, 9, 10\} \\
&= \{2, 3, 7, 8, 9, 10\}, \\
K_{Multiple}(8) &= \{3, 7, 8\} \cup \{5, 8, 9\} \cup \{8\} \\
&= \{3, 5, 7, 8, 9\}, \\
K_{Multiple}(9) &= \{5, 8, 9\} \cup \{2, 7, 9, 10\} \\
&= \{2, 5, 7, 8, 9, 10\}, \\
K_{Multiple}(10) &= \{6, 10\} \cup \{2, 7, 9, 10\} \\
&= \{2, 6, 7, 9, 10\}.
\end{aligned}$$

The attribute-value block combinations of the attribute *Mixed* in the data set shown in Table 7.1 are as follows:

$$\begin{aligned}
K_{Mixed}(3) &= \{3,\ 10\} \cup \{1,\ 3,\ 4,\ 8,\ 9,\ 10\} \\
&= \{1,\ 3,\ 4,\ 8,\ 9,\ 10\}, \\
K_{Mixed}(4) &= \{2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10\} \cup \{1,\ 3,\ 4,\ 8,\ 9,\ 10\} \\
&= \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\}, \\
K_{Mixed}(8) &= \{2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10\} \cup \{1,\ 3,\ 4,\ 8,\ 9,\ 10\} \\
&= \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\}, \\
K_{Mixed}(10) &= \{2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10\} \cup \{3,\ 10\} \cup \{1,\ 3,\ 4,\ 8,\ 9,\ 10\} \\
&= \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\}.
\end{aligned}$$

Using the combination of attribute-value blocks above, the characteristic sets of the data set in Table 7.1 are as follows:

$$\begin{aligned}
K_A(1) &= \{1,\ 2,\ 9\} \cap \{1,\ 2,\ 3,\ 4,\ 6\} \cap \{1,\ 3,\ 4,\ 8,\ 9,\ 10\} \\
&= \{1\}, \\
K_A(2) &= \{1,\ 2,\ 9\} \cap \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 9,\ 10\} \cap \{2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10\} \\
&= \{2\}, \\
K_A(3) &= \{3\} \cap \{1,\ 2,\ 3,\ 6,\ 7,\ 8\} \cap \{1,\ 3,\ 4,\ 8,\ 9,\ 10\} \\
&= \{3\}, \\
K_A(4) &= \{4,\ 7,\ 8,\ 10\} \cap \{1,\ 2,\ 3,\ 4,\ 6\} \cap \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\} \\
&= \{4\}, \\
K_A(5) &= \{5\} \cap \{2,\ 4,\ 5,\ 6,\ 8,\ 9\} \cap \{2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10\} \\
&= \{5\}, \\
K_A(6) &= \{6\} \cap \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 10\} \cap \{2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10\} \\
&= \{6\}, \\
K_A(7) &= \{4,\ 7,\ 8,\ 10\} \cap \{2,\ 3,\ 7,\ 8,\ 9,\ 10\} \cap \{2,\ 4,\ 5,\ 6,\ 7,\ 8,\ 10\} \\
&= \{7,\ 8,\ 10\}, \\
K_A(8) &= \{4,\ 7,\ 8,\ 10\} \cap \{3,\ 5,\ 7,\ 8,\ 9\} \cap \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\} \\
&= \{7,\ 8\}, \\
K_A(9) &= \{1,\ 2,\ 9\} \cap \{2,\ 5,\ 7,\ 8,\ 9,\ 10\} \cap \{1,\ 3,\ 4,\ 8,\ 9,\ 10\} \\
&= \{9\}, \\
K_A(10) &= \{4,\ 7,\ 8,\ 10\} \cap \{2,\ 6,\ 7,\ 9,\ 10\} \cap \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 10\} \\
&= \{7,\ 10\}.
\end{aligned}$$

Given the concepts $[(Letter,\ A)] = \{6,\ 7\}$, $[(Letter,\ B)] = \{3,\ 9\}$, $[(Letter,\ C)] = \{2,\ 4,\ 5,\ 10\}$, $[(Letter,\ D)] = \{1,\ 8\}$, inconsistencies identified in the data set shown in Table 7.1 are as follows:

$$
\begin{aligned}
K_A(7) &= \{7,\ 8,\ 10\} \nsubseteq \{6,\ 7\},\\
K_A(7) &= \{7,\ 8,\ 10\} \nsubseteq \{1,\ 8\},\\
K_A(7) &= \{7,\ 8,\ 10\} \nsubseteq \{2,\ 4,\ 5,\ 10\},\\
K_A(8) &= \{7,\ 8\} \nsubseteq \{6,\ 7\},\\
K_A(8) &= \{7,\ 8\} \nsubseteq \{1,\ 8\},\\
K_A(10) &= \{7,\ 10\} \nsubseteq \{6,\ 7\},\\
K_A(10) &= \{7,\ 10\} \nsubseteq \{2,\ 4,\ 5,\ 10\}.
\end{aligned}
$$

Case 7 belongs to three different concepts and cases 8 and 10 belong to two different concepts. Therefore the data set is inconsistent and an approximation will be created.

Approximations Upper and lower approximations of the data set shown in Table 7.1 are as follows:

$$[(Letter,\ A)] = \{6,\ 7\},$$
$$\overline{A}[(Letter,\ A)] = \{6\} \cup \{7,\ 8,\ 10\} \cup \{7,\ 8\} \cup \{7,\ 10\} = \{6,\ 7,\ 8,\ 10\},$$
$$\underline{A}[(Letter,\ A)] = \{6\},$$

$$[(Letter,\ B)] = \{3,\ 9\},$$
$$\overline{A}[(Letter,\ B)] = \{3\} \cup \{9\} = \{3,\ 9\},$$
$$\underline{A}[(Letter,\ B)] = \{3\} \cup \{9\} = \{3,\ 9\},$$

$$[(Letter,\ C)] = \{2,\ 4,\ 5,\ 10\},$$
$$\overline{A}[(Letter,\ C)] = \{2\} \cup \{4\} \cup \{5\} \cup \{7,\ 8,\ 10\} \cup \{7,\ 10\} = \{2,\ 4,\ 5,\ 7,\ 8,\ 10\},$$
$$\underline{A}[(Letter,\ C)] = \{2\} \cup \{4\} \cup \{5\} = \{2,\ 4,\ 5\},$$

$$[(Letter,\ D)] = \{1,\ 8\},$$
$$\overline{A}[(Letter,\ D)] = \{1\} \cup \{7,\ 8,\ 10\} \cup \{7,\ 8\} = \{1,\ 7,\ 8,\ 10\},$$
$$\underline{A}[(Letter,\ D)] = \{1\}.$$

Execution Execution of the LEM2 algorithm on the data set shown in Table 7.1 is shown in the following tables.

Table 7.2: Computing rules for the upper approximation of [(Letter, A)]

(a, v) = t	[(a,v)]	{6,7,8,10}	{6,7,8,10}	{7,8,10}	{6,10}
[(Single, S1)]	{4,7,8,10}	{7,8,10}	{7,8,10} •	-	{10}
[(Single, S2)]	{3}	∅	∅	∅	∅
[(Single, S4)]	{6}	{6}	{6}	∅	{6}
[(Single, S9)]	{1,2,9}	∅	∅	∅	∅
[(Single, S10)]	{5}	∅	∅	∅	∅
[(Multiple, M0)]	{2,5,6}	{6}	{6}	∅	{6}
[(Multiple, M1)]	{3}	∅	∅	∅	∅
[(Multiple, M2)]	{1,2,3,6}	{6}	{6}	∅	{6}
[(Multiple, M3)]	{6,10}	{6,10}	{6,10}	{10}	{6,10} •
[(Multiple, M4)]	{3,7,8}	{7,8}	{7,8}	{7,8} •	∅
[(Multiple, M5)]	{5,8,9}	{8}	{8}	{8}	∅
[(Multiple, M6)]	{1,4}	∅	∅	∅	∅
[(Multiple, M7)]	{4,5,6}	{6}	{6}	∅	{6}
[(Multiple, M8)]	{8}	{8}	{8}	{8}	∅
[(Multiple, M9)]	{2,7,9,10}	{7,10}	{7,10}	{7,10}	{10}
[(Multiple, M10)]	{4,6}	{6}	{6}	∅	{6}
[(Mixed, X1)]	{2,4,5,6,7,8,10}	{6,7,8,10} •	-	-	{6,10}
[(Mixed, X2)]	{3,10}	{10}	{10}	{10}	{10}
[(Mixed, X3)]	{1,3,4,8,9,10}	{8,10}	{8,10}	{8,10}	{10}
	T	{2,4,5,6,7,8,10}	{4,7,8,10}	{7,8}	{6,10}
	$\mathbb{T}$	∅	∅	{{7,8}}	{{7,8},{6,10}}

Table 7.3: Computing rules for the upper approximation of [(Letter, B)]

(a, v) = t	[(a,v)]	{3,9}	{3,9}	{9}	{9}
[(Single, S1)]	{4,7,8,10}	∅	∅	-	∅
[(Single, S2)]	{3}	{3}	{3} •	∅	∅
[(Single, S4)]	{6}	∅	∅	∅	∅
[(Single, S9)]	{1,2,9}	{9}	{9}	{9} •	-
[(Single, S10)]	{5}	∅	∅	∅	∅
[(Multiple, M0)]	{2,5,6}	∅	∅	∅	∅
[(Multiple, M1)]	{3}	{3}	{3}	∅	∅
[(Multiple, M2)]	{1,2,3,6}	{3}	{3}	∅	∅
[(Multiple, M3)]	{6,10}	∅	∅	∅	∅
[(Multiple, M4)]	{3,7,8}	{3}	{3}	∅	∅
[(Multiple, M5)]	{5,8,9}	{9}	{9}	{9}	{9} •
[(Multiple, M6)]	{1,4}	∅	∅	∅	∅
[(Multiple, M7)]	{4,5,6}	{3}	{3}	∅	∅
[(Multiple, M8)]	{8}	∅	∅	∅	∅
[(Multiple, M9)]	{2,7,9,10}	{9}	{9}	{9}	{9}
[(Multiple, M10)]	{4,6}	∅	∅	∅	∅
[(Mixed, X1)]	{2,4,5,6,7,8,10}	∅	∅	∅	∅
[(Mixed, X2)]	{3,10}	{3}	{3}	∅	∅
[(Mixed, X3)]	{1,3,4,8,9,10}	{3,9} •	-	{9}	{9}
	T	{1,3,4,8,9,10}	{3}	{1,2,9}	{9}
	$\mathbb{T}$	∅	{{3}}	{{3}}	{{3},{9}}

Table 7.4: Computing rules for the upper approximation of [(Letter, C)]

(a, v) = t	[(a,v)]	{2,4,5,10}	{2,4,5,10}	{2,5}
[(Single, S1)]	{4,7,8,10}	{4,10}	{4,10}	∅
[(Single, S2)]	{3}	∅	∅	∅
[(Single, S4)]	{6}	∅	∅	∅
[(Single, S9)]	{1,2,9}	{2}	{2}	{2}
[(Single, S10)]	{5}	{5}	{5}	{5} •
[(Multiple, M0)]	{2,5,6}	{2,5}	{2,5} •	-
[(Multiple, M1)]	{3}	∅	∅	∅
[(Multiple, M2)]	{1,2,3,6}	{2}	{2}	{2}
[(Multiple, M3)]	{6,10}	{10}	{10}	∅
[(Multiple, M4)]	{3,7,8}	∅	∅	∅
[(Multiple, M5)]	{5,8,9}	{5}	{5}	{5}
[(Multiple, M6)]	{1,4}	{4}	{4}	∅
[(Multiple, M7)]	{4,5,6}	{2,4}	{2,4}	{2}
[(Multiple, M8)]	{8}	∅	∅	∅
[(Multiple, M9)]	{2,7,9,10}	{2,10}	{2,10}	{2}
[(Multiple, M10)]	{4,6}	{4}	{4}	∅
[(Mixed, X1)]	{2,4,5,6,7,8,10}	{2,4,5,10} •	-	-
[(Mixed, X2)]	{3,10}	{10}	{10}	∅
[(Mixed, X3)]	{1,3,4,8,9,10}	{4,10}	{4,10}	∅
	T	{2,4,5,6,7,8,10}	{2,5,6}	{5}
	$\mathbb{T}$	∅	∅	{{5}}

Table 7.5: Computing rules for the upper approximation of [(Letter, C)] (continued)

(a, v) = t	[(a,v)]	{2,4,10}	{2,4,10}	{10}	{10}
[(Single, S1)]	{4,7,8,10}	{4,10}	{4,10}	{10}	∅
[(Single, S2)]	{3}	∅	∅	∅	∅
[(Single, S4)]	{6}	∅	∅	∅	∅
[(Single, S9)]	{1,2,9}	{2}	{2}	∅	∅
[(Single, S10)]	{5}	∅	∅	∅	∅
[(Multiple, M0)]	{2,5,6}	{2}	{2}	∅	∅
[(Multiple, M1)]	{3}	∅	∅	∅	∅
[(Multiple, M2)]	{1,2,3,6}	{2}	{2}	∅	∅
[(Multiple, M3)]	{6,10}	{10}	{10}	{10} •	-
[(Multiple, M4)]	{3,7,8}	∅	∅	∅	∅
[(Multiple, M5)]	{5,8,9}	∅	∅	∅	∅
[(Multiple, M6)]	{1,4}	{4}	{4}	∅	∅
[(Multiple, M7)]	{4,5,6}	{2,4}	{2,4} •	∅	∅
[(Multiple, M8)]	{8}	∅	∅	∅	∅
[(Multiple, M9)]	{2,7,9,10}	{2,10}	{2,10}	{10}	{10}
[(Multiple, M10)]	{4,6}	{4}	{4}	∅	∅
[(Mixed, X1)]	{2,4,5,6,7,8,10}	{2,4,10} •	-	∅	∅
[(Mixed, X2)]	{3,10}	{10}	{10}	{10}	{10} •
[(Mixed, X3)]	{1,3,4,8,9,10}	{4,10}	{4,10}	{10}	{10}
	T	{2,4,5,6,7,8,10}	{2,4}	{6,10}	{10}
	$\mathbb{T}$	{{5}}	{{5},{2,4}}	{{5},{2,4}}	{{5},{2,4},{10}}

Table 7.6: Computing rules for the upper approximation of [(Letter, D)]

(a, v) = t	[(a,v)]	{1,8}	{1,8}	{1}	{1}
[(Single, S1)]	{4,7,8,10}	{8}	{8}	∅	∅
[(Single, S2)]	{3}	∅	∅	∅	∅
[(Single, S4)]	{6}	∅	∅	∅	∅
[(Single, S9)]	{1,2,9}	{1}	{1}	{1}	{1} •
[(Single, S10)]	{5}	∅	∅	∅	∅
[(Multiple, M0)]	{2,5,6}	∅	∅	∅	∅
[(Multiple, M1)]	{3}	∅	∅	∅	∅
[(Multiple, M2)]	{1,2,3,6}	{1}	{1}	{1}	{1}
[(Multiple, M3)]	{6,10}	∅	∅	∅	∅
[(Multiple, M4)]	{3,7,8}	{8}	{8}	∅	∅
[(Multiple, M5)]	{5,8,9}	{8}	{8}	∅	∅
[(Multiple, M6)]	{1,4}	{1}	{1}	{1} •	-
[(Multiple, M7)]	{4,5,6}	∅	∅	∅	∅
[(Multiple, M8)]	{8}	{8}	{8} •	∅	∅
[(Multiple, M9)]	{2,7,9,10}	∅	∅	∅	∅
[(Multiple, M10)]	{4,6}	∅	∅	∅	∅
[(Mixed, X1)]	{2,4,5,6,7,8,10}	{8}	{8}	∅	∅
[(Mixed, X2)]	{3,10}	∅	∅	∅	∅
[(Mixed, X3)]	{1,3,4,8,9,10}	{1,8} •	-	{1}	{1}
	T	{1,3,4,8,9,10}	{8}	{1,4}	{1}
	$\mathbb{T}$	∅	{{8}}	{{8}}	{{8},{1}}

Possible Rules

1. $(Single,\ S1) \wedge (Multiple,\ M4) \longrightarrow (Letter,\ A)$,

2. $(Multiple,\ M3) \longrightarrow (Letter,\ A)$,

3. $(Single,\ S2) \longrightarrow (Letter,\ B)$,

4. $(Single,\ S9) \wedge (Multiple,\ M5) \longrightarrow (Letter,\ B)$,

5. $(Single,\ S10) \longrightarrow (Letter,\ C)$,

6. $(Mixed,\ X1) \wedge (Multiple,\ M7) \longrightarrow (Letter,\ C)$,

7. $(Multiple,\ M3) \wedge (Mixed,\ X2) \longrightarrow (Letter,\ C)$,

8. $(Multiple,\ M8) \longrightarrow (Letter,\ D)$,

9. $(Multiple,\ M6) \wedge (Single,\ S9) \longrightarrow (Letter,\ D)$.

Certain Rules

1. $(Single,\ S4) \longrightarrow (Letter,\ A)$,

2. $(Single,\ S2) \longrightarrow (Letter,\ B)$,

3. $(Single,\ S9) \wedge (Multiple,\ M5) \longrightarrow (Letter,\ B)$,

4. $(Single,\ S10) \longrightarrow (Letter,\ C)$,

5. $(Multiple,\ M7) \wedge (Mixed,\ X1) \longrightarrow (Letter,\ C)$,

6. $(Multiple,\ M6) \wedge (Single,\ S9) \longrightarrow (Letter,\ D)$.

7.2 Method Evaluation

To evaluate the different methods presented of inducing rules from set-value data sets, we execute each method on a collection of set-value data sets. Five data sets of all symbolic data with missing values from the machine learning data repository, University of California at Irvine were used [7]. Missing values were replaced with the set of all possible values for the attribute to create a set-value data set. Four randomly generated data sets with a mixture of single-value and set-value attributes were created for use in the evaluations. Finally, the set-value data set from the test for foreign language ability in Shanxi University will be used [9]. Data sets used for evaluation are outlined in Table 7.7. Experiments are based on those performed in *Rough sets based matrix approaches with dynamic attribute variation in set-valued information systems* [10].

Table 7.7: Description of evaluation data sets

			Number of	
Data Set	Source	Cases	Attributes	Classes
Congressional Voting Records	UCI	435	16	2
Mushroom	UCI	8124	22	2
Soybean-large	UCI	307	35	19
Audiology (Standardized)	UCI	226	69	24
Dermatology	UCI	336	34	6
Set-value data one	Data Generator	2000	5	5
Set-value data two	Data Generator	200	100	4
Set-value data three	Data Generator	500	10	8
Set-value data four	Data Generator	800	8	20
Shanxi Foreign Language Test	Shanxi University	50	4	2

Rule complexity will be used to evaluate the different methods for handling set-value attributes. To compare rule complexity, the number of rules and the total number of conditions are measured for each rule set created. Results can be seen in Table 7.8 of the different methods executed on each data set.

Table 7.8: Experimental evaluation results

	Parallel		Split		Variables	
Data Set	Rules	Conditions	Rules	Conditions	Rules	Conditions
Congressional Voting Records	206	3290	36	220	32	120
Mushroom	12	28	14	33	11	26
Soybean-large	47	374	NA	NA	44	164
Audiology (Standardized)	40	152	47	149	44	163
Dermatology	30	118	35	145	30	118
Set-value Data One	1253	5616	10175	43072	1895	6866
Set-value Data Two	47	156	NA	NA	80	179
Set-value Data Three	211	834	NA	NA	464	1523
Set-value Data Four	557	1975	772	2416	737	2551
Shanxi Foreign Language Test	17	101	58	213	17	60

For the data sets Soybean-large, Set-value Data One, and Set-value Data Three the split set values approach produced data sets too large to be processed by the implementation described in Chapter 8. The large number of cases resulted in increased processing time and memory usage to unpractical levels. For example, a single case in the Set-value Data Two data set exploded into hundreds of thousands of split cases.

The parallel method produced rule sets with the fewest number of rules in seven out of ten data sets compared to only five for the attribute variables method and none for the split set values method. The parallel method produced rule sets with the fewest number of conditions in five out of ten data sets compared to only five for the attribute variables method and one for the split set values method. For real world data, the attribute variables method produced the least complex rule sets while the parallel method produced the least complex rule sets for generated data sets. Generally, the parallel method produced rule sets with fewer rules and the attribute variables method produced rule sets with fewer conditions.

Chapter 8

Implementation

As a proof of concept an implementation of the LEM2 algorithm was written in JavaScript. Implementation includes functionality to induce rules on data sets with set-value attributes. A web interface for the application is available at https://lrdodge.github.io/lem2.

Technology The technologies used to create the application are as follows:

- Bootstrap 3,
- jQuery,
- Mocha.js,
- Chai.js,
- PapaParse.js,
- GitHub Pages.

Data Source Data sets are input into the application using the Comma Separated Value (CSV) format. Any data processing errors will be displayed below the data input field and must be resolved before proceeding. The first row of the data source should be the column labels and the last column should be the decision. Omit case numbers and separate set values with the | character.

For example, the set $\{A, B, C\}$ would be represented as A|B|C. For optimum performance and rule coverage, data should be discretized prior to processing.

A data source must be selected or input to initialize the application. Select a data source by clicking one of the data source option buttons and then clicking the *Load Data* button. Example data sets are provided to quickly show the capabilities and functionality. Data source options include:

- Input File - Local CSV file with headers,
- Manual Input - Enter CSV data directly into the data field,
- Examples - Data from one of several existing data sets.

Concept Selection After initial data processing (concept and attribute-value block determination), a dialog will display with the concepts derived from the input data. Each concept will show the value and number of cases within the concept. Clicking on the number badge will display the cases. Clicking the badge again will hide the cases.

Select a concept by clicking one of the concept option buttons in the dialog. A concept must be selected before rules can be induced. Once a concept has been selected, click the *Induce Rules* button. Depending on the size of the data set and number of attribute-value blocks, rule induction may take a significant amount of time. Size and complexity of the data set will be constrained by the memory available to the browser.

Rule Set Display Once rule induction is complete rules will be displayed in the Rules section at the bottom of the page. Rules are shown in the order in which they were created. Because consistent data is not enforced, it is possible for the goal to not be equal to the empty set and no blocks available for selection. This is an indicator the input data set is inconsistent. An exclamation sign will be displayed by any rules which were unable to be completed. In these instances it is recommended an approximation of the data set should be created instead.

Chapter 9

Conclusion

LEM2 is constrained to single-value data sets, limiting the algorithm's range of application. By examining each step of the LEM2 algorithm, the definition of characteristic sets was identified as a point of failure for set-value data sets. Modifying characteristic set creation and the interpretation of attribute-value blocks will expand the data sets which LEM2 can handle. Data sets with set-value attributes can be now processed without modification or expansion of the data set.

The parallel method to inducing rules for set-value data sets creates rule sets which are less complex than those created with sequential methods. Only for the Congressional Voting Records data set did the parallel method produce the most complex rule set. Given the attribute variable method's propensity towards inconsistent rule sets, the parallel method is the best way to induce rules from set-value data sets.

The change to the characteristic sets and attribute-value blocks solves issues of determining consistency as well as the algorithm's execution itself. The output of the algorithm is consistent for both single-value data sets and set-value data sets and will cover cases for each concept. The solution scales to data sets of any size and attribute count, opening the door to a solution where complete coverings of set-value data sets are possible.

The parallel method presented will produce a consistent, but not complete, set of rules for a set-value data set. Some values in a set-value attribute may not be used during rule induction, even when the attribute was selected for rule creation. Currently a case is considered covered if one, but not all, attributes in a set-value attribute are used in a rule. For a rule created with a set-value attribute, there is no guarantee all values in a particular set will be accounted for in the final rule.

For example, consider the rules induced for $(Letter, B)$ for the data set shown in Table 7.1:

1. $(Single, S2) \longrightarrow (Letter, B)$,
2. $(Single, S9) \wedge (Multiple, M5) \longrightarrow (Letter, B)$.

When $(Multiple, M5)$ was selected to create Rule 2, $(Multiple, M9)$ was not included in the rule set despite belonging to one of the cases covered. The rule set created for the data set shown in Table 7.1 does not cover all cases for the logically equivalent split version of cases 3 and 9 shown in Table 9.1.

Table 9.1: Split cases of concept (Letter, B)

		Attributes		Decision
Case	Single	Multiple	Mixed	Letter
3_1	S2	M1	X2	B
3_2	S2	M2	X2	B
3_3	S2	M4	X2	B
3_4	S2	M7	X2	B
3_5	S2	M1	X3	B
3_6	S2	M2	X3	B
3_7	S2	M4	X3	B
3_8	S2	M7	X3	B
9_1	S9	M5	X3	B
9_2	S9	M9	X3	B

As we can see, Rule 2 covers case 9_1, but not case 9_2. Additional work is needed to modify the process to produce a complete set of consistent rules for the data set.

Further exploration into updating attribute-value block selection heuristics is also needed. Possible new preferences include single value attributes and smallest set cardinality of set-value attributes. These should be compared for a potential increase in rule set coverage or a decrease in rule set complexity.

References

[1] C. Chan and J. Grzymala-Busse. On the two local induction algorithms: Prism and lem2. *Foundations of Computing and Decision Sciences*, 19:185–203, 1994.

[2] J. Grzymala-Busse. A comparison of three strategies to rule induction from data with numerical attributes. *Electronic Notes in Theoretical Computer Science*, 82:132–140, 2003.

[3] J. Grzymala-Busse. Rough set theory with applications to data mining. *Real World Applications of Computational Intelligence*, pages 221–244, 2005.

[4] J. Grzymala-Busse. Rule induction. *Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, 2nd edn.*, page 249–265, 2010.

[5] J. Grzymala-Busse and W. Grzymala-Busse. Handling missing attribute values. *Data Mining and Knowledge Discovery Handbook*, pages 33–51, 2010.

[6] J. Grzymala-Busse and W. Rzasa. A local version of the mlem2 algorithm for rule induction. *Fundamenta Informaticae*, 100:1–18, 2010.

[7] D. Newman, S. Hettich, C. Blake, and C. Merz. Uci repository of machine learning databases, 1998.

[8] Z. Pawlak. Rough set theory and its applications. *Journal of Telecommunications and Information Technology*, pages 1–10, 3 2002.

[9] Y. Qian, C. Dang, J. Liang, and D. Tang. Set-valued ordered information systems. *Information Sciences*, 179:1–24, 2009.

[10] J. Zhang, L. Tianrui, D. Ruan, and D. Liu. Rough sets based matrix approaches with dynamic attribute variation in set-valued information systems. *International Journal of Approximate Reasoning*, 53:620–635, 2012.