UNIVERSITY OF WATERLOO
DEPARTMENT OF COMPUTER SCIENCE
MACHINE LEARNING GROUP

# USER'S MANUAL

## ELEM2 RULE INDUCTION SYSTEM

VERSION 3.0

Aijun An and Nick Cercone

# USER'S MANUAL

## ELEM2 RULE INDUCTION SYSTEM VERSION 3.0

## TABLE OF CONTENTS

**TABLE OF CONTENTS** .......... **2**

**INTRODUCTION** .......... **3**

**DATA PREPARATION** .......... **3**

FILE NAMES .......... 3
TRAINING DATA FILE .......... 3
DATA DESCRIPTION FILE .......... 4
TEST DATA FILE .......... 6

**GENERATING RULES** .......... **6**

HOW TO RUN THE RULE INDUCTION PROGRAM .......... 6
FILES GENERATED BY THE SYSTEM .......... 8
HOW TO INTERPRET THE RULES .......... 8

**TESTING RULES** .......... **10**

HOW TO RUN THE TESTING PROGRAM .......... 10
FILES GENERATED BY THE TESTING PROGRAM .......... 10

**CROSS VALIDATION** .......... **11**

HOW TO RUN THE CROSS VALIDATION PROGRAM .......... 11
FILES GENERATED BY THE CROSS VALIDATION PROGRAM .......... 11

**EXAMPLES** .......... **12**

EXAMPLE 1 .......... 12
*Problem Description* .......... *12*
*Training data and its description* .......... *12*
*Generating rules without using pruning* .......... *13*
*Generating rules with the pruning option* .......... *14*
EXAMPLE 2 .......... 15
*Problem Description* .......... *15*
*Training and Testing data* .......... *16*
*Rules generated by elem2* .......... *16*
*Testing on the test data* .......... *18*
EXAMPLE 3 .......... 19
*Dataset Description* .......... *19*
*Generating Rules* .......... *20*
EXAMPLE 4 .......... 22
*Dataset description* .......... *22*
*Cross validation* .......... *22*

**ACKNOWLEDGEMENT** .......... **23**

**REFERENCES** .......... **24**

## INTRODUCTION

ELEM2 [1] is a rule induction system that generates rules from a set of data and uses the generated rules to classify (new) cases in another set of data. In this manual, we describe how to use the ELEM2 (Version 3.0) rule induction system. The system contains three executable programs: *elem2v3.exe*, *test.exe* and *cvelem2.exe*. The program elem2v3.exe is a rule induction program that generates a set of classification rules from a set of training data. The program test.exe is a classification program, which uses the rules generated by elem2v3.exe and classifies the cases in a test data file. The program cvelem2.exe is a cross-validation program that can be used to evaluate ELEM2 on a data set using *n*-fold cross-validation. To run these programs, the user has to prepare the training and testing data. In this manual, we describe how to prepare these data files, how to run the three programs with the data files from the command line of an operating system, and how to interpret the result files generated from the programs. We also illustrate the use of the ELEM2 system by running the programs with some examples, which demonstrates the use of elem2v3.exe, test.exe and cvelem2.exe.

## DATA PREPARATION

In this section, we describe how to prepare the training data file, test data file and their description file. The training data file is used by ELEM2 to generate rules. The test data is used to test generated rules. The description file describes the training or testing data and is required by the system for rule induction and cross validation.

### FILE NAMES

All files read and written by the ELEM2 system are in plain text format and have a name of *filestem.extension*, where *extension* characterizes the type of information contained in the file. A filestem can be any string of characters that is acceptable as a file name to your operating system. The maximum length of a filestem is 60 characters.

### TRAINING DATA FILE

The training data file is used to represent the training cases from which ELEM2 constructs decision rules. Training cases are described in terms of condition attributes and a decision attribute. The decision attribute, also called the class attribute, is used to describe the class that a case belongs to. Each value of the decision attribute represents a class. For example, if a case in a risk-evaluating domain belongs to a *low, medium,* or *high* class, then the decision attribute (say, named as risk) for the training data in this domain has a value domain which contains *low, medium,* and *high*, each of which represents a class. Condition attributes are used to describe the conditions under which a case belongs to a certain class. Each condition attribute also has a name and a value domain. For example, age can be a condition attribute for the risk-evaluating domain and it takes a value between 0 and 150. A case in the training data file is represented by values of a set of condition and decision attributes

The training data file used by ELEM2 is named as *filestem*.**dat**. Each line in the file describes one case, providing the values of all the condition and decision attributes, separated by spaces or tabs. For example,

```
20   9.8    red   0   12.7   8.5   1

9    10.6   blue  2   11.6   7.9   0
```

are two cases in a training data file, where each case consists of values for six condition attributes and one class attribute. The attribute values must appear in the same order in all the cases and in the same order that the attributes are given in the description file (see below). The order of cases themselves does not matter. Numeric values may be given in integer, fixed-point, or floating-point form, so that all the following are acceptable as attribute values:

7, 12.7, +2.5, -.3, -23E-5, 0.000026

## DATA DESCRIPTION FILE

The description file is fundamental to the induction task. The file, named *filestem*.**fmf**, provides information about each attribute and class, including attribute names, class names, attribute types, priorities of attributes, attribute values for symbolic attributes, and a numeric range for each continuous attribute.

The description file consists of a series of entries, Each entry occupies one line and describes one attribute.[1] An entry starts with a "<" and ends with a ">". There are three kinds of attributes: *symbolic, integer,* and *real-valued* attributes. A symbolic attribute has discrete values; an integer attribute has integer values; and a real-valued attribute has continuous values. Depending on the type of the attribute, the entry can be in one of the following formats:

1. For a symbolic attribute, its entry takes the following fomat:

   <**C/D/X Priority Name** S **Number_of_values List_of_values**>

   where **C/D/X** takes the values of C, D or X and indicates whether the attribute is a condition attribute (indicated by C), a decision (class) attribute (indicated by D), or an ignored attribute (indicated by X),[2] **Priority** specifies the priority of this attribute among all the condition attributes and takes an integer value with lower number indicating higher priority, **Name** specifies the name[3] for this attribute, S means symbolic attribute, **Number_of_values** indicates the number of symbolic values of the attribute, and **List_of_values** lists the symbolic values. For example, a condition attribute *color* has the values of *red, blue, yellow* and *green*. Suppose its priority is 5. The entry for this attribute is

[1] The maximum number of attributes currently allowed by ELEM2 (V2.0) is 100.

[2] If an attribute is labeled as an X attribute, then its corresponding column of data in the training data file will be ignored by the induction program. We allow X attributes in a training set because at times a data set contains irrelevant attributes, such as the index of the training cases, which is usually irrelevant to the learning task.

[3] An attribute name can be any string of characters without spaces in between. The maximum length of an attribute name is 19 characters.

<C 5 color S 4 red blue yellow green>

2. For an integer or real-valued attribute, its entry takes one of the following three formats, depending on how you would like the attribute to be discretized:

**<C/D/X Priority Name I/R>**

or

**<C/D/X Priority Name I/R Min_value Max_value D Number_of_partitions>**

or

**<C/D/X Priority Name I/R Min_value Max_value M List_of_cutpoints>**

where **C/D/X**, **Priority, Name** have the same meaning as for symbolic attributes, **I/R** takes the values of I or R and indicates whether the attribute is an integer or a real-valued attribute, **Min_value** gives the minimum value of the attribute and **Max_value** is the maximum value for the attribute. If the first format is used, ELEM2 invokes an automatic supervised discretization method[4] to symbolize the values of this attribute in the training cases. For example, if a real-valued condition attribute named *pressure* has the priority of 3 and you would like to use the automatic supervised method to discretize this attribute, then its entry in the description file is

<C 3 pressure R>

If the second format is used, ELEM2 uses an equal width interval binning method that divides the range of values for this attribute into **Number_of_partitions** equal sized bins, where **Number_of_partitions** is a user-supplied parameter in the entry. For example, if you would like to discretize the above *pressure* attribute into 10 equal sized value ranges, the entry for this attribute in the description file is (suppose the range of values for this attribute is between 0 and 500):

<C 3 pressure R 0 500 D 10>

which means that the value range of the *pressure* attribute is divided into [0, 50], (50, 100], ……, (450, 500].

If the third format for the attribute entry is used, ELEM2 discretizes the attribute using the provided cut-points. For example, if you want to discretize the pressure attribute into ranges of [0, 80], (80, 120], (120, 170], (170, 250], (250, 400], and (400, 500], then the attribute's entry in the data description file is as follows:

<C 3 pressure R 0 500 M 80 120 170 250 400>

Note that either the second or the third format has to be used for a decision attribute if the decision attribute is an integer (I) or a real-valued (R) attribute. That is, the user has to provide

[4] A supervised discretization method makes use of the class labels in the training cases in the discretization process, while unsupervised methods do not utilize the class labels. The supervised discretization method used in ELEM2 is based on an entropy minimization theory [3].

either the number of equal width intervals or the cutpoints for a decision attribute of type I or R. ELEM2 cannot use a supervised method to discretize a decision attribute automatically.

An example for a description file is

```
<C 3 a1 I>
<C -2 a2 R 0 5.0 D 5>
<C 0 a3 I –30 60 M –5 0 20 35 51>
 <C 1 color S 4 red blue yellow green>
<D 0 class S 2 0 1>
```

where there are four condition attributes and a decision attribute in total. The attribute a2 has the highest priority among the three condition attributes and a1 the lowest. The order of these entries should be consistent with the order of attribute values in each case in the training data file. In this example there are three integer or real-valued attributes, named a1, a2 and a3 respectively. The attribute a1 will be discretized using an automatic supervised discretization method, the attribute a2 will be discretized into 5 equal width intervals, and a3 will be discretized using the specified cut-points.

Depending on the training data file, the entry for the decision attribute is not necessary to be the last one. If no attribute is specified as a decision attribute, i.e., all the attributes are labeled as either C or X, then the last attribute with C is considered to be the decision attribute. If more than one attributes are specified as decision attributes, ELEM2 only considers the first attribute with D is the decision attribute and others as condition attributes. The value for the priority of a decision attribute does not affect the induction results because priority is designed to specify the relative importance of condition attributes. However, a number must be provided for the priority of a decision attribute for the purpose of syntax checking of the description file.

### TEST DATA FILE

To evaluate the classification rules the system has produced from the training data, you may reserve part of the available data as a test data set or generate a separate test data set. The test data appears in the file *filestem*.**tst**, in exactly the same format as the training data file. Testing the classification rules is optional.

## GENERATING RULES

### HOW TO RUN THE RULE INDUCTION PROGRAM

After training data and description files have been prepared, inducing rules from the training data can be as simple as running the following command:

elem2v3.exe *filestem*

in a Command Prompt window under the directory where elem2v3.exe, filestem.dat and filestem.fmf reside. The options that can be used with this command are

–p (Default: no pruning)

This option allows the program to use a pruning technique [1] to post-prune the rules in order to deal with possible noise in the training data. It is used as follows:

```
elem2v3.exe filestem –p
```

The default is no pruning, in which case the program generates rules that fit the training data as well as possible. Use of the pruning option is recommended.

-q *rule_quality_no* (Default: 1)

The option allows the user to choose one of the 12 rule quality formulas [4] encoded in the elem2v3.exe program by specifying a number between 1 and 12, inclusive. The selected formula is used in the ELEM2's post-pruning and classification procedures. The 12 formulas are shown in the following table (see [4] for the description of these formulas):

| Formula Number | Rule Quality Formula |
|---|---|
| 1 | Measure of Discrimination |
| 2 | Weighted Sum of Consistency and Coverage |
| 3 | C2 |
| 4 | C1 |
| 5 | Degree of Logical Sufficiency |
| 6 | Coleman's Formula |
| 7 | Product of Consistency and Coverage |
| 8 | G2 Likelihood Ratio Statistic |
| 9 | Measure of Information |
| 10 | Pearson Chi-Square Statistic (Version 1) |
| 11 | Cohen's formula |
| 12 | Pearson Chi-square Statistic (Version 2) |

The default value of this option is 1, in which case the formula *Measure of Discrimination* is used. An example of using this option is

```
elem2v3.exe filestem –p –q 5
```

which causes the elem2v3.exe program to use *Degree of Logical Sufficiency* as the rule quality formula.

In summary, the elem2v3.exe program can be invoked as elem2v3.exe *filestem* [-p] [-q *rule_quality_no*]

The two options can appear in either order and either of them can be missing.

## FILES GENERATED BY THE SYSTEM

The program elem2v3.exe generates some intermediate and results files with the same filestem as used by the training data and description files. The intermediate files are generated at the beginning of the program execution, and will be deleted automatically by the program after rules are generated. As a user, you will not have to worry about the intermediate files, other than to make sure that you do not delete or modify them while they are still relevant. The intermediate file extensions are: *fmc1, qzf, qzf1,* and *neg*.

Three results files are generated by the program. Their extensions are:

- **rule**, containing description and information about the rules that generated by the program. This file is the results file for users.

- **intr**, also containing description of rules, but in a format not readable by users. The file is for the program test.exe to read induced rules in order to classify testing data.

- **fmc**, containing the information about condition and decision attributes, especially the cutpoints that the system generated for continuous attributes. This file is needed when running test.exe. It serves as a data description file for test.exe as *filestem.fmf* for elem2v3.exe.

## HOW TO INTERPRET THE RULES

We describe how to interpret the generated rules in the file *filestem.rule*. A rule is listed according to the class value it predicts. For example, if there are two classes in the problem, denoted as 0 and 1 respectively in the entry for the decision attribute of the file *filestem.fmf* as follows:

<D 0 Class 2 0 1>

then in the file *filestem.rule*, the rules that predict Class=0 is listed first under the title "Rules for Class=0" before the rules predicting Class=1, which are listed later under the title "Rules for Class=1" as follows:

```
            Rules for Class=0

Rule 1:
(a1=1)(a2!=1)(a5!=1)
Rule Accuracy = 1.000000
Rule Quality  = 2.096910

The positive cases covered by the rule are: (31 cases)
11 12 13 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 32
```

```
33 34 35 36 38 39 40 41 43 44 45
The negative cases covered by the rule are: (0 cases)


Rule 2:
(a1!=1)(a2=1)(a5!=1)
Rule Accuracy = 1.000000
Rule Quality  = 1.780275

The positive cases covered by the rule are: (20 cases)
46 47 50 51 52 53 54 55 56 57 58 59 60 61 91 92 94 95 96 97
The negative cases covered by the rule are: (0 cases)

            :
            :
            :

      Rules for Class=1

Rule 1:
(a5=1)
Rule Accuracy = 1.000000
Rule Quality  = 2.041687

The positive cases covered by the rule are: (29 cases)
9 10 14 31 37 42 48 49 66 67 71 77 78 80 84 85 88 89 90 93
98 100 102 103 105 108 113 117 121
The negative cases covered by the rule are: (0 cases)


Rule 2:
(a1=3)(a2=3)
Rule Accuracy = 1.000000
Rule Quality  = 1.681937

The positive cases covered by the rule are: (17 cases)
108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124
The negative cases covered by the rule are: (0 cases)

         :
         :
         :

Rule Numbers are:
  Rule number for Class=0 is 4
  Rule number for Class=1 is 4

Total number of rules = 8

Average Length of the rules = 2.375000
```

```
Evaluation on training data: accuracy = 100.0000%
```

Description of each rule consists of the condition part of the rule, rule accuracy on the training data, rule quality in which higher numbers indicate better rules, and the positive and negative cases covered by the rule. The condition part of a rule consists of one attribute-value pair or a conjunction of two or more attribute-value pairs. For example, the second rule for Class=1 is interpreted as

*If the attribute a1 is equal to 3 and the attribute a2 is equal to 3, then Class=1*

At the end of the file *filestem.rule*, a summary is given which states how many rules are generated for each class, the total number of rules, average length of the rules (in terms of the number of attribute-value pairs in the condition part of rules), and classification accuracy of the rules evaluated over the training data.

## TESTING RULES

After rules are generated, you can test the predictive performance of the rules on another set of available data, called test data. The test data should appear in the file *filestem.tst*. The file can be the same as or different from the training data file *filestem.dat*, but it must be in the same format.

### HOW TO RUN THE TESTING PROGRAM

The command for invoking the testing program is

test.exe *filestem*

This program reads attribute descriptions from *filestem.fmc* and the rules from *filestem.intr*. It uses these rules to classify the cases in the file *filestem.tst*, then compares the class it predicts for each case with the class label for that case in the *filestem.tst* file, and calculates the predictive accuracy of the rules over these testing cases.

### FILES GENERATED BY THE TESTING PROGRAM

The test.exe program also generates some intermediate and results files. The intermediate files will be deleted by the program after testing is done. The extensions of these files are *qzf* and *qzf1*. Users should not be concerned with intermediate files other than to avoid deleting or modifying them while the program is still running. Two results files are generated by test.exe. Their extensions are

- **tcover**, containing information about rules, the test cases covered by each rule, and at the end the predictive accuracy of the rules on the testing data. The content of this file is the same as the *filestem.rule* except that the cases covered by each rule and the predictive accuracy are in terms of testing data, not the training data.

- **result**, containing all the cases in the *filestem.tst* file with the predicted class at the end of each case. An example of this file is given in the next section.

## CROSS VALIDATION

The cross validation program is to evaluate the ELEM2 rule induction system on a data set by using *n*-fold cross validation. At the expense of computational resources, cross validation gives a more reliable estimate of accuracy of a learning system than a single run on a held-out test set. A *n*-fold cross validation program randomly partitions a data set into *n* disjoint subsets, then provides the learning program with *n-1* of them as training data and uses the remaining one as test cases. This process is repeated *n* times using different possible test subsets. Each time a classification accuracy is obtained on the test subset. The mean of the accuracies from the *n* runs and the standard deviation of the accuracy are then calculated to measure the testing performance.

### HOW TO RUN THE CROSS VALIDATION PROGRAM

The command for invoking the cross validation program is

cvelem2.exe *elem2_rule_induction_program filestem n* [-p] [-q *rule_quality_no*]

where *elem2_rule_induction* is the name of the ELEM2 rule induction program to be evaluated, *filestem* is the prefix of the data file (filestem.dat) that the cross validation is conducted on, and *n* is the number of folds. Similar to the elem2v3.exe program, two options (-p and –q *rule_quality_no*) can be used with cvelem2.exe. They are used in the same way as used with elem2v3.exe and passed from the cvelem2.exe program to the rule induction program to be evaluated (such as elem2v3.exe). Other inputs to the cvelem2.exe program, which are not shown in the command, include a filestem.fmf file and the test.exe program. The filestem.fmf file provides description of the data in filestem.dat. The test.exe program is used to classify test cases during cross validation. An example of running cvelem2.exe is

cvelem2.exe elem2v3.exe iris 10 –p –q 3

which conducts 10-fold evaluation on the elem2v3.exe program on the data set *iris.dat* with the pruning option and the rule quality formula *C2*.

### FILES GENERATED BY THE CROSS VALIDATION PROGRAM

The cvelem2.exe program generates some intermediate and result files. The intermediate files are the training, testing, intermediate or result files used or generated in each of the *n* runs. These files will be deleted by the program after the evaluation is done. The cvelem2.exe program generates one result file. Its extension is

- **cv**, containing the information about each run, such as the number of generated rules, the average length of the rules and the predictive accuracy on the test subset. The file also contains a summary of the results from the *n* runs, such as the average number of rules, the average length of rules, the average testing accuracy and standard deviation over the *n* runs.

## EXAMPLES

We illustrate the use of ELEM2 with some examples.

### EXAMPLE 1

PROBLEM DESCRIPTION

This example is designed to illustrate how to use elem2v3.exe with or without pruning, the option –p. The problem contains two classes and three symbolic condition attributes (A, B, and C), each of which has two values (0 or 1). The two classes, denoted as D=1 and D=0 respectively, are described as follows:

*If (A=1) and (B=1) or (C=0), then D=1; otherwise D=0.*

TRAINING DATA AND ITS DESCRIPTION

The training data file (*example1.dat*) contain all the eight possible examples, which are

```
0    0    0    1
0    0    1    0
0    1    0    1
0    1    1    0
1    0    0    1
1    0    1    0
1    1    0    1
1    1    1    1
```

The description file (*example1.fmf* ) is as follows, in which we give all the attributes the same priority:

```
<C 0 A S 2 0 1>

<C 0 B S 2 0 1>

<C 0 C S 2 0 1>

<D 0 D S 2 1 0>
```

## GENERATING RULES WITHOUT USING PRUNING

Use the following command to generate rules that describe the data exactly without using the –p option:

```
elem2v3.exe example1
```

The following rules are generated, which are contained in the file *example1.rule*.

```
    Rules for D=1


Rule 1:
(C=0)
Rule Accuracy = 1.000000
Rule Quality  = 1.322219

The positive cases covered by the rule are: (4 cases)
1 3 5 7
The negative cases covered by the rule are: (0 cases)


Rule 2:
(A!=0)(B!=0)
Rule Accuracy = 1.000000
Rule Quality  = 0.698970

The positive cases covered by the rule are: (2 cases)
7 8
The negative cases covered by the rule are: (0 cases)


    Rules for D=0


Rule 1:
(A=0)(C!=0)
Rule Accuracy = 1.000000
Rule Quality  = 1.263241

The positive cases covered by the rule are: (2 cases)
2 4
The negative cases covered by the rule are: (0 cases)


Rule 2:
(B=0)(C!=0)
Rule Accuracy = 1.000000
Rule Quality  = 1.263241

The positive cases covered by the rule are: (2 cases)
```

```
2 6
The negative cases covered by the rule are: (0 cases)

Rule Numbers are:
  Rule number for D=1 is 2
  Rule number for D=0 is 2

Total number of rules = 4

Average Length of the rules = 1.750000

Evaluation on training data: accuracy = 100.0000%
```

GENERATING RULES WITH THE PRUNING OPTION

If you consider the data set may contain noise, you can use the pruning option when running elem2v3.exe as follows:

elem2v3.exe example1 –p

In this case, the file *example1.rule* contains the following generated rules:

```
    Rules for D=1


Rule 1:
(C=0)
Rule Accuracy = 1.000000
Rule Quality  = 1.322219

The positive cases covered by the rule are: (4 cases)
1 3 5 7
The negative cases covered by the rule are: (0 cases)


Rule 2:
(A!=0)(B!=0)
Rule Accuracy = 1.000000
Rule Quality  = 0.698970

The positive cases covered by the rule are: (2 cases)
7 8
The negative cases covered by the rule are: (0 cases)
```

```
     Rules for D=0


Rule 1:
(C!=0)
Rule Accuracy = 0.750000
Rule Quality  = 1.322219

The positive cases covered by the rule are: (3 cases)
2 4 6
The negative cases covered by the rule are: (1 cases)
8

Rule Numbers are:
  Rule number for D=1 is 2
  Rule number for D=0 is 1

Total number of rules = 3

Average Length of the rules = 1.333333

Evaluation on training data: accuracy =  87.5000%
```

Note that in this case the eighth example (1 1 1 1) is interpreted by ELEM2 as a noisy case for the class D=0.

## EXAMPLE 2

PROBLEM DESCRIPTION

Example 2 is also artificially designed. It involves two continuous condition attributes, one symbolic condition attribute, and two classes. The description file *example2.fmf* describes the domains of each attribute:

```
<C 0 a1 I>
<C 0 a2 R>
<C 0 color S 4 red blue yellow green>
<D 0 class S 2 1 0>
```

where the attribute a1 is of  integer type; the attribute a2 is of real-valued type; the attribute color has 4 symbolic values: *red, blue, yellow* and *green*. The relationship between the classes and the condition attributes in the data set is designed as follows:

*If (30<a1<=60) and (1.5<a2<=3.5)and (color=blue or green), then class=1; otherwise class=0.*

## TRAINING AND TESTING DATA

The file *example2.tst* contains 440 examples that satisfy the above relationship. 60% of these examples are randomly chosen as training examples, which are contained in the file *example2.dat*.

## RULES GENERATED BY ELEM2

Using the following command:

```
elem2v3.exe example2
```

the following rules (shown in *example2.rule*) are generated from the data file *example2.dat*.

```
    Rules for class=1


Rule 1:
(30<a1<=60)(1.500000<a2<=3.500000)(color=blue or green)
Rule Accuracy = 1.000000
Rule Quality  = 4.132932

The positive cases covered by the rule are: (13 cases)
118 120 122 124 126 146 148 150 152 170 172 173 175
The negative cases covered by the rule are: (0 cases)


    Rules for class=0


Rule 1:
(color!=blue or green)
Rule Accuracy = 1.000000
Rule Quality  = 1.427917

The positive cases covered by the rule are: (125 cases)
1 3 11 13 16 18 20 23 26 27 28 30 32 33 34 36 39 40 42 44
46 49 50 51 53 55 57 61 62 64 66 68 70 71 73 78 81 83 84 86
88 89 90 92 94 96 97 101 102 104 106 107 109 110 111 114 116 119 121 123
125 128 130 132 134 138 140 142 144 147 149 151 153 155 156 157 159 160 161 163
165 167 169 171 174 177 179 182 186 189 191 193 195 199 204 206 208 210 213 216
219 221 224 225 226 227 229 232 234 236 237 240 242 244 246 249 250 252 253 255
257 258 259 260 262
The negative cases covered by the rule are: (0 cases)


Rule 2:
(a1<=30)
Rule Accuracy = 1.000000
Rule Quality  = 1.316963
```

```
The positive cases covered by the rule are: (109 cases)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
101 102 103 104 105 106 107 108 109
The negative cases covered by the rule are: (0 cases)


Rule 3:
(a2<=1.500000)
Rule Accuracy = 1.000000
Rule Quality  = 1.231440

The positive cases covered by the rule are: (97 cases)
1 2 3 4 5 6 7 8 30 31 32 33 34 35 36 37 38 57 58 59
60 61 62 63 64 65 66 67 88 89 90 91 92 93 94 95 110 111 112 113
114 115 116 117 136 137 138 139 140 141 142 143 144 145 159 160 161 162 163 164
165 166 167 168 183 184 185 186 187 188 189 190 191 192 193 194 212 213 214 215
216 217 218 219 220 236 237 238 239 240 241 242 243 244 245 246 247
The negative cases covered by the rule are: (0 cases)


Rule 4:
(a1>60)
Rule Accuracy = 1.000000
Rule Quality  = 1.118648

The positive cases covered by the rule are: (82 cases)
183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202
203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222
223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242
243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262
263 264
The negative cases covered by the rule are: (0 cases)


Rule 5:
(a2>3.500000)
Rule Accuracy = 1.000000
Rule Quality  = 1.054322

The positive cases covered by the rule are: (74 cases)
22 23 24 25 26 27 28 29 49 50 51 52 53 54 55 56 79 80 81 82
83 84 85 86 87 104 105 106 107 108 109 127 128 129 130 131 132 133 134 135
153 154 155 156 157 158 176 177 178 179 180 181 182 203 204 205 206 207 208 209
210 211 231 232 233 234 235 258 259 260 261 262 263 264
The negative cases covered by the rule are: (0 cases)
```

```
Rule Numbers are:
  Rule number for class=1 is 1
  Rule number for class=0 is 5

Total number of rules = 6

Average Length of the rules = 1.833333

Evaluation on training data: accuracy = 100.0000%
```

TESTING ON THE TEST DATA

To evaluate how these rules perform on the testing data in file *example2.tst*, use the following command:

```
test.exe example2
```

The program will read rules from *example2.intr* and classify each case in *example2.tst*. The classification results are shown in the file *example2.result* as follows:

```
0 0.000000 red 0 -> 0
0 0.000000 blue 0 -> 0
0 0.000000 yellow 0 -> 0
0 0.000000 green 0 -> 0
0 0.500000 red 0 -> 0
0 0.500000 blue 0 -> 0
0 0.500000 yellow 0 -> 0
0 0.500000 green 0 -> 0
0 1.000000 red 0 -> 0
          :
          :
          :
40 2.000000 red 0 -> 0
40 2.000000 blue 1 -> 1
40 2.000000 yellow 0 -> 0
40 2.000000 green 1 -> 1
40 2.500000 red 0 -> 0
40 2.500000 blue 1 -> 1
40 2.500000 yellow 0 -> 0
40 2.500000 green 1 -> 1
40 3.000000 red 0 -> 0
40 3.000000 blue 1 -> 1
          :
          :
          :

Number of testing cases = 440    Number of cases classified correctly = 440
```

```
Predictive Accuracy = 100.000000%
```

In this file, each row shows a test case with the column before -> representing the actual class for the case as shown in the test date file and the column after -> representing the predicted class generated by the test.exe program. A summary of the prediction results is shown at the end of the file.

## EXAMPLE 3

### DATASET DESCRIPTION

Example 3 is taken from the UCI repository of machine learning databases [5]. The data set (*housing.dat*) concerns housing values in suburbs of Boston. It contains 506 training examples. Each example is described by 13 condition attributes (12 continuous and 1 binary-valued attributes) and 1 decision attribute (continuous and representing house values). This data set is used here to illustrate the use of continuous decision attributes.

The condition and decision attributes of the data set are listed as follows:

| Attribute Names | Meaning |
|---|---|
| CRIM | per capita crime rate by town |
| ZN | proportion of residential land zoned for lots over 25,000 square feet |
| INDUS | proportion of non-retail business acres per town |
| CHAS | Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) |
| NOX | nitric oxides concentration (parts per 10 million) |
| RM | average number of rooms per dwelling |
| AGE | proportion of owner-occupied units built prior to 1940 |
| DIS | weighted distances to five Boston employment centres |
| RAD | index of accessibility to radial highways |
| TAX | full-value property-tax rate per $10,000 |
| PTRATIO | pupil-teacher ratio by town |
| B | $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town |
| LSTAT | percentage of the population with lower status |
| MEDV | median value of owner-occupied homes in $1000's |

where MEDV is the decision attribute and others are condition attributes. A description file (*housing.fmf*) for this data set is:

```
<C 0 CRIM R>
<C 0 ZN R>
<C 0 INDUS R>
<C 0 CHAS S 2 0 1>
<C 0 NOX R>
<C 0 RM R>
<C 0 AGE R>
<C 0 DIS R>
<C 0 RAD I 1 24 D 10>
<C 0 TAX R>
<C 0 PTRATIO R>
<C 0 B R>
<C 0 LSTAT R>
<D 0 MEDV R 5.0 50.0 M 10 20 30 40>
```

where MEDV is a real-valued decision attribute and cut-points (10, 20, 30 and 40) are specified to discretize the attribute whose values range from 5.0 and 50.0. Thus, there are five classes in this data set, denoted as (MEDV<=10.0), (10<MEDV<=20), (20<MEDV<=30), (30<MEDV<=40), and (MEDV>40).

## GENERATING RULES

To generate rules from *housing.dat*, use command:

```
elem2v3.exe housing
```

The following are samples of rules in the *housing.rule* file generated by the above command.

```
    Rules for (MEDV<=10.000000)


Rule 1:
(CRIM>7.526010)(NOX>0.671000)(RM>5.272000)(DIS<=2.002600)(LSTAT>26.639999)
Rule Accuracy = 1.000000
Rule Quality  = 2.616550

The positive cases covered by the rule are: (7 cases)
386 399 400 401 405 416 439
The negative cases covered by the rule are: (0 cases)


Rule 2:
(NOX>0.605000)(RM<=6.152000)(DIS<=2.002600)(B<=68.949997)(LSTAT>19.879999)
Rule Accuracy = 1.000000
Rule Quality  = 2.325986

The positive cases covered by the rule are: (4 cases)
419 426 438 439
```

```
The negative cases covered by the rule are: (0 cases)

                         :
                         :
                         :

     Rules for (10.000000<MEDV<=20.000000)


Rule 1:
(CRIM<=15.023400)(NOX>0.583000)(RM<=6.525000)(AGE>82.500000)(B>50.919998)(14.100000
<LSTAT<=19.879999)
Rule Accuracy = 1.000000
Rule Quality  = 2.279597

The positive cases covered by the rule are: (44 cases)
128 129 130 134 135 136 137 138 140 147 154 155 156 157 171 357 362 364 391 394
395 396 397 421 422 431 434 435 442 443 444 447 448 449 450 453 459 460 462 475
477 479 489 492
The negative cases covered by the rule are: (0 cases)

                         :
                         :
                         :

     Rules for (MEDV>40.000000)


Rule 1:
(RM>7.079000)(DIS<=6.640700)(PTRATIO<=14.900000)(LSTAT<=7.440000)
Rule Accuracy = 1.000000
Rule Quality  = 3.005333

The positive cases covered by the rule are: (16 cases)
162 163 164 167 196 203 204 205 258 262 263 268 269 281 283 284
The negative cases covered by the rule are: (0 cases)

                         :
                         :
                         :
```

## EXAMPLE 4

### DATASET DESCRIPTION

Example 4 is also taken from the UCI repository of machine learning databases [5]. The data set (*iris.dat*) concerns classification of iris flowers. It contains 150 examples. Each example is described by 4 condition attributes (all continuous) and one decision attribute. A description file (*iris.fmf*) for this data set is:

```
<C 0 sepal_length R>
<C 0 sepal_width R>
<C 0 petal_length R>
<C 0 petal_width R>
<D 0 class S 3 1 2 3>
```

### CROSS VALIDATION

To evaluate ELEM2 on the iris data set, we can use command:

```
cvelem2.exe elem2v3.exe iris 10 –p
```

which evaluates *elem2v3.exe* (with the pruning option and the default rule quality formula) on the *iris.dat* data set using 10-fold cross validation. The evaluation result is recorded in the *iris.cv* file, the content of which is shown below:

```
Run 1:
Total number of rules = 7
Average length of the rules = 2.142857
Prediction accuracy in the testing data = 93.333336%

Run 2:
Total number of rules = 8
Average length of the rules = 2.750000
Prediction accuracy in the testing data = 100.000000%

Run 3:
Total number of rules = 7
Average length of the rules = 2.000000
Prediction accuracy in the testing data = 100.000000%

Run 4:
Total number of rules = 5
Average length of the rules = 1.800000
Prediction accuracy in the testing data = 86.666664%
```

```
Run 5:
Total number of rules = 7
Average length of the rules = 2.000000
Prediction accuracy in the testing data = 100.000000%

Run 6:
Total number of rules = 7
Average length of the rules = 2.000000
Prediction accuracy in the testing data = 100.000000%

Run 7:
Total number of rules = 7
Average length of the rules = 1.857143
Prediction accuracy in the testing data = 100.000000%

Run 8:
Total number of rules = 6
Average length of the rules = 1.666667
Prediction accuracy in the testing data = 93.333336%

Run 9:
Total number of rules = 8
Average length of the rules = 2.250000
Prediction accuracy in the testing data = 100.000000%

Run 10:
Total number of rules = 6
Average length of the rules = 2.000000
Prediction accuracy in the testing data = 100.000000%

  ------ Summary -------
Average number of rules = 6.800000
Average length of rules = 2.046667
Average testing accuracy = 97.333328%
Standard deviation of the accuracy = 4.661306
```

## ACKNOWLEDGEMENT

The authors are members of the Institute for Robotics and Intelligent Systems (IRIS) and wish to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, the Natural Sciences and Engineering Research Council, and the participation of PRECARN Associates Inc.

## REFERENCES

1. An, A. and Cercone, N. 1998. ELEM2: A Learning System for More Accurate Classifications, In *Proceedings of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, AI'98*, Vancouver, Canada.

2. An, A., Cercone, N., Chan, C. and Shan, N. 1995. ELEM: A Method for Inducing Rules from Examples, In *Proceedings of the 15th Annual Technical Conference of the British Computer Society Specialist Group on Expert Systems*, Cambridge, U.K.

3. An, A. and Cercone, N. (1999) Discretization of Continuous Attributes for Learning Classification Rules, *Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99*), Beijing, China.

4. An, A. and Cercone, N. (2000) Rule Quality Measures Improve the Accuracy of Rule Induction: an Experimental Approach**,** *Proceedings of 12th International Symposium on Methodologies for Intelligent Systems*, Charlotte, North Carolina.

5. Murphy, P.M. and Aha, D.W. 1994. UCI Repository of Machine Learning Databases. URL: http://www.ics.uci.edu/AI/ML/MLDBRepository.html.