Knowledge Visualizer: a Software System for Visualizing Data, Patterns and Their Relationships

Q. Zhang

P 97-19
MLI 97-14

KNOWLEDGE VISUALIZER: A SOFTWARE SYSTEM FOR VISUALIZING DATA, PATTERNS AND THEIR RELATIONSHIPS

Qi Zhang

Machine Learning and Inference Laboratory
School of Information Technology and Engineering
George Mason University
Fairfax, Virginia 22030-4444
qzhang@aic.gmu.edu

Publication No.
P 97-19

Reports of the Machine Learning and Inference Laboratory
MLI 97-14

September 1997

KNOWLEDGE VISUALIZER: A SOFTWARE SYSTEM FOR VISUALIZING DATA, PATTERNS AND THEIR RELATIONSHIPS

ABSTRACT

This report describes a novel knowledge visualization tool: Knowledge Visualizer or KV. KV is specifically designed to visualize various aspects of **concept learning**. These visualized aspects include: representation space (or instance space), qualitative and quantitative distribution of training or testing examples, relationships between examples and rules characterizing concepts, qualitative and quantitative visual display of concept rules, changes in the representation space done by constructive induction. This software system employs a planar model of a multidimensional space spanned over a set of discrete attributes. The model is in the form of a diagram, in which each cell represents a unique combination of attribute values. The diagram can represent examples, rules, and rulesets (DNF). KV is very useful in many aspects such as analyzing behavior of existing learning algorithms or a new learning system being developed in its every stage, teaching or training people in machine learning, data mining and knowledge discovery in databases.

KEYWORD: Knowledge visualization, machine learning.

ACKNOWLEDGMENTS

This research was conducted in the Machine Learning and Inference Laboratory at George Mason University. The Laboratory's activities are supported in part by the Defense Advanced Research Projects Agency under grant F49620-95-1-0462, administered by the Air Force Office of Scientific Research, in part by the National Science Foundation under grants DMI-9496192 and IRI-9020266, and in part by the Office of Naval Research under grant N00014-91-J-1351

8/12/03

Also

Users' Guide for

Knowledge Visualizer 1.0

Table of Contents

1 INTRODUCTION 5

2 PREPARATION 7

2.1 TERMINOLOGY 7

2.2 THE FORMAT OF INPUT 8

2.3 EXEMPLARY DATA 8

2.3.1 Monk1's Problem 8

2.3.2 Designs of Wind Bracing 11

3 USING KV 12

3.1 STARTING KV 12

3.2 INPUT DATA 12

3.2.1 Open 13

3.2.2 View 13

3.3 ARRANGE ATTRIBUTES IN A REPRESENTATION SPACE 13

3.4 VISUALIZATION 16

3.4.1 Representation Space 16

3.4.2 Y Attributes and X Attributes 17

3.4.3 Examples of All Classes 17

3.4.4 Examples of Selected Classes 18

3.4.5 Positive and Negative Examples 19

3.4.6 Example Distribution of a Selected Class 19

3.4.7 Examples Corresponding to a Cell 20

3.4.8 Learned Rules of All Classes 21

3.4.9 Learned Rules of Selected Classes 22

3.4.10 Colored Rules of a Selected Class 22

3.4.11 Learned Rules of One Class by Total Cover 23

3.4.12 Learned Rules of One Class by Unique Cover 23

3.4.13 Rules and Examples of a Selected Class 23

3.4.14 Examples and Colored Rules of a Selected Class 24

3.4.15 Rules of a Selected Class against Examples of All Classes 25

3.4.16 Rules Corresponding to a Cell 25

3.4.17 Rules of All Classes of the Target Concept 26

3.4.18 Rules of Selected Classes of the Target Concept 26

3.4.19 Colored Rules of a Selected Class of the Target Concept 26

3.4.20 Errors of Commission, Errors of Omission, and Total Errors 26

3.4.21 Generate Examples from Scratch 27

3.4.22 Generate Examples Based on Previous Examples 27

3.5 OTHER OPERATIONS 27

3.5.1 Another Visualizer 27

4 OTHER TOPICS 28

4.1 CONSTRUCTIVE INDUCTION 28

4.2 PROJECTION OF EXAMPLES 28

4.3 INTERNAL VALUES OF A REPRESENTATION SPACE 29

5 CONCLUSION ..... ERROR! BOOKMARK NOT DEFINED.

APPENDIX 1: MONK1 DATA 30

APPENDIX 2: WIND BRACING DATA 33

REFERENCES 37

1 INTRODUCTION

This report describes a novel knowledge visualization tool: Knowledge Visualizer or KV. KV is specifically designed to visualize various aspects of **concept learning**. This system is very useful in many aspects such as analyzing behavior of existing learning algorithms or a new learning system being developed in its every stage, teaching or training people in machine learning, data mining and knowledge discovery in databases, providing a quick, preliminary yet effective data analysis tool.

KV employs a planar model of a multidimensional space spanned over a set of discrete attributes. The model is called General Logic Diagram (GLD) and was introduced by Michalski (1978). The model is in the form of a diagram, in which each *cell* represents a unique combination of attribute values. Each attribute partitions the diagram into areas corresponding to individual values of this attribute. Conjunctive rules correspond to certain regular arrangements of cells that can be easily recognized visually. The diagram can represent examples, rules, and rulesets (DNF)

The main goal of KV is to provide a tool for a visual interpretation of various aspects of concept learning. These include: representation space (or instance space), qualitative and quantitative distribution of training or testing examples; relationships between examples and rules characterizing concepts, qualitative and quantitative visual display of rules, changes in the representation space done by constructive induction.

Besides visualization of steps in a learning process, KV is also able to display the errors in concept learning. The set of cells representing the target concept (the concept to be learned) is called *target concept image (T)*. The set of cells representing the learned concept is called *learned concept image (L)*. The areas of the *target concept* not covered by the *learned concept* represent *errors of omission (T \ L)*, while the areas of the learned concept not covered by the target concept represent *errors of commission (L \ T)*. The union of both types of errors represents the *error image*. Display of these errors gives users a further insight into learned knowledge.

KV is based on the pioneering work DIAV (Wnek, 1995) which is the first software system for rule induction. Actually many of the KV's features are similar to those of DIAV. However, compared with DIAV, KV has several new practical and attractive ones:

(1) DIAV is coded in SmallTalk for Macintosh and its code for visual display of aspects of learning process cannot run in other machines. Hence, it is very machine-dependent and cannot be

accessible to most researchers or users who are actually using other types of machines. KV is coded in Java and its visual display is workable on any computer where a Java interpreter was installed, even through Web browsers such as Netscape[1].

(2) DIAV has some system limitations introduced in its design. For instance, DIAV, can display description spaces up to 4M events, i.e., spaces spanned over up to 22 binary variables (or correspondingly smaller number of multiple-valued variables). In contrast, KV does not impose any hardware or software limitations such as this. It allows problems as large as a particular machine's hardware or software system can deal with. It simply lets users themselves beat a system's limitations. This is very important because machine learning techniques are now being applied to larger and larger practical problems in data mining. KV also has convenient features for visualizing large problems.

(3) KV adopts combinations of symbols and colors to display examples and rules of different classes. It differs from DIAV which uses different kinds of texture patterns to label examples and rules. Using symbols and colors accommodates problems with many classes. However, diagrams using texture patterns are not visually comfortable and distinguishable in case of a large number of classes, and further, it is difficult to find or design new texture patterns for problems which has many classes.

(4) KV provides not only qualitative but also quantitative visualization of spatial distribution of examples and coverage of rules. It also allows for quantitative examination of a single rule, a ruleset or rule strength. These quantitative features are missing in DIAV.

Summarized are main KV features:

(1) Allow problems as large as possible. No restrictions are placed on the number of classes, attributes and attribute levels. No restrictions are placed on file size and memory requirement as long as the machine is able to run a Java interpreter. KV also provides auxiliary functions for visualizing large representation spaces.

(2) Window and menu driven. An input to the system comes from data files generated by a learning system (e.g. AQ15c or AQ18) or created manually. Output is directed to a graphical terminal with regard to visual effects. The graphical terminal is able to display and control multiple windows. Scroll bars on sides of the panes enable to display larger than screen size images.

(3) Flexible input manners. KV accommodates many cases in which users want to input their data. Note that in KV 1.0 the input means AQ-style descriptions of attributes, examples and

[1] The current version of KV is not an applet and so it cannot run in a Web browser. However, it is easy to produce an equivalent applet version.

rules.

(4) Pleasant visual display. Colors and symbols are combined for displaying different classes.

(5) Visualization of many aspects of examples and rules such as distribution of examples, spatial coverage of rules, relationships between examples and rules. and relationships between representation space cells and rules and examples. It provides not only qualitative but also quantitative display of them and so it enables an accurate understanding of various aspects of data and learning.

(6) Projection of a given representation space onto another one by removing and/or adding attributes. Easily visualizing such changes provides the user a very convenient and effective way of understanding the effects of constructive induction.

2 PREPARATION

2.1 Terminology

Here are concepts used throughout this manual. Not that some are defined loosely.

Attribute: an aspect characterizing an object or entity. It is composed of a set of values which can be continuous, discrete or nominal, etc. In the GLD model, only discrete, nominal or discretized continuous attributes are allowed.

Cell: an unique combination of attribute values which occupies a single grid in a GLD diagram.

Example: an item or member of a class. It is described with a set of attribute values. This term is exchangeable with **event**. It can be further classified as **training/learning example** used for learning by a learning system and **testing example** used for testing the performance of the learning system.

Concept: a set of examples. Usually, members of this set have some similarities. Learning is to find these relationships or similarities. This term is exchangeable with *class*.

Representation space: a Cartesian product of all attributes. It is the set of all possible combinations of attribute values. It is exchangeable with **instance space**.

Hypothesis: the similarities, relationships, or patterns identified by a learning system and represented by its representation language. In the case of rule learning, hypothesis is exchangeable with **rules**.

Cover: this word is used here in three situations: “A rule covers an example” or “an example is covered by a rule”. This means this rule logically follows this example or this example can be characterized by this rule. The second usage means spatial coverage of rules and the third means a ruleset. In AQ rule learning, the *total cover* of a rule means the number of examples

coverd by this rule; its *unique cover* means those examples covered only by this rule, not any other rules. The total cover and unique cover represents the *strength* or *weight* of a rule.

Target Concept: the exact set of examples defining a concept. Usually it is represented as a hypothesis which exactly covers this set of examples. No more or less. It is what a learning system attempts to output based on only some members.

Learned Concept: the set of rules output by a learning system. Usually it is not the same as the target concept. Represented as a hypothesis.

Error of commission: an example or a cell covered by the learned concept but not by the target concept.

Error of omission: an example or a cell covered by the target concept but not by the learned concept.

Image: an actual graphical display of rules or examples. It consists of three types.

Concept image: a diagram with only rules displayed in it. A rule covers a set of cells in this diagram.

Example image: a diagram with only examples displayed in it. Each example corresponds to a cell in this diagram.

Error image: a diagram with either errors of commission or errors of omission displayed in it.

Working window: the window where users start KV.

2.2 The Format of Input

Any valid input to AQ15c (Wnek et al., 1995) or AQ18 and any valid output from them. For the feel-and-look of input and output, see particular data in section 2.3. It would be better for users to arrange data in a file according to this sequence: variable section, names section, events section, and rule section. It would be better for each section to be separated by one blank line. Data of four problems, Monk1, Monk2, V concept, Windbracing, are provided. When encountering running problems with KV, be sure to make the input data have a correct format.

2.3 Exemplary Data

Throughout this report, data of two problems are used. One is Monk1 (Thrun et al., 1991). The other is about optimal designs of wind bracing in steel skeleton structures of tall buildings (Szczepanik et al., 1995).

2.3.1 Monk1's Problem

The Monk1's problem relys on an artificial robot domain, in which robots are described by six different attributes:

x1: head_shape ∈ {round, square, octagon}
x2: body_shape ∈ {round, square, octagon}
x3: is_smiling ∈ {yes, no}
x4: holding ∈ {sword, balloon, flag}
x5: jacket_color ∈ {red, yellow, green, blue}
x6: has_tie ∈ {yes, no}

The learning is a two-class classification task. This problem defines a class of robots as "(head_shape = body_shape) or (jacket_color = red)" (we call this class class1) and all other robots which do not conform with this description form another class (we call it class2). There are all 432 possible robots. The learning task is then to accept a randomly drawn subset of examples and generalize over this subset and output a set of rules characterizing these two classes. The following is part of the training data file (for the full set of data, see the software package):

```
parameters
 run   mode   ambig   trim   wts   maxstar  echo    criteria  verbose
  1    ic     neg     mini   cpx   10       pv      default   1

 variables
   #   type   levels   cost   name
   1   nom       3    1.00   x1
   2   nom       3    1.00   x2
   3   nom       2    1.00   x3
   4   nom       3    1.00   x4
   5   nom       4    1.00   x5
   6   nom       2    1.00   x6

x1-names
 value name
 0     1
 1     2
 2     3

x2-names
 value name
 0     1
 1     2
 2     3

x3-names
 value name
 0     1
 1     2

x4-names
 value name
```

```
 0    1
 1    2
 2    3


x5-names
 value name
 0    1
 1    2
 2    3
 3    4

x6-names
 value name
 0    1
 1    2

class1-events
 x1 x2 x3 x4 x5 x6
 3  3  1  3  4  2
 2  2  1  1  2  2
.....

class2-events
 x1 x2 x3 x4 x5 x6
 3  2  1  1  4  2
 2  1  2  1  3  1
.....
```

In the parameters section are defined some learning parameters for the AQ learning program. *Note that this section is useless and unnecessary to KV*. Following is the variables (i.e., attributes) section which is important to KV because KV uses this section to build representation space. As can be seen, each attribute's values are replaced with integers starting from 1 (this is not required). The name section specifies the correspondence of these integers to AQ's internal values starting from 0.

The learned rules are the following:

```
 class1-outhypo
   #   cpx
   1   [x5=1]   (t:16, u:11)
   2   [x1=3] [x2=3]   (t:15, u:11)
   3   [x1=2] [x2=2]   (t:10, u:10)
   4   [x1=1] [x2=1]   (t:7, u:6)

class2-outhypo
   #   cpx
   1   [x1=1,3] [x2=2] [x5=2,3,4]   (t:18, u:18)
   2   [x1=2] [x2=1,3] [x5=2,3,4]   (t:13, u:13)
   3   [x1=1] [x2=3]   (t:9, u:9)
   4   [x1=3] [x2=1] [x5=2]   (t:3, u:3)
```

One version of the target concepts is the following:

```
class1-outhypo
 #   cpx
 1   [x5=1]
 2   [x1=3] [x2=3]
 3   [x1=2] [x2=2]
 4   [x1=1] [x2=1]

class2-outhypo
 #   cpx
 1   [x1=1] [x2=2..3] [x5=2..4]
 2   [x1=2..3] [x2=1] [x5=2..4]
 3   [x1=3] [x2=2] [x5=2..4]
 4   [x1=2] [x2=3] [x5=2..4]
```

2.3.2 Design of Wind Bracing

The following is part of the input file to the AQ learning system (for the full data set, see the software package).

```
parameters
 run   mode   ambig   trim   wts   maxstar   echo  verbose
  1      ic     empty   spec   cpx    10        pcdv       3

variables
# type size  name
1 nom 5  x1.l5
2 nom 2  x2.l2
3 nom 2  x3.l2
4 nom 3  x4.l3
5 nom 3  x5.l3
6 nom 4  x6.l4
7 nom 4  x7.l4


 l2-names
value name
0     1
1     2

 l3-names
value name
0     1
1     2
2     3

 l4-names
value name
0     1
1     2
2     3
3     4
```

```
 l5-names
value name
0     1
1     2
2     3
3     4
4     5

class1-events
 x1 x2 x3 x4 x5 x6 x7
 1  2  2  3  2  1  3
 1  2  2  3  2  1  2
......

class2-events
 x1 x2 x3 x4 x5 x6 x7
 2  1  1  1  3  1  1
 5  2  2  1  3  1  1
....

class3-events
 x1 x2 x3 x4 x5 x6 x7
 3  2  1  2  1  2  3
 2  2  1  2  1  2  3
....

class4-events
 x1 x2 x3 x4 x5 x6 x7
 5  2  2  3  1  1  2
 5  2  1  3  1  1  4
….
```

3 USING KV

3.1 Starting KV

A Java 1.0.2 interpreter must be installed first. In Sparc workstation or in Windows95, open a wind and type java KV. The following window will be popped out:

Fig. 1. The main window.

3.2 Input Data

The current KV version requires all data (attributes, examples, rules) to be stored in files first. Click the File menu and you can see Fig. 2.

3.2.1 Open

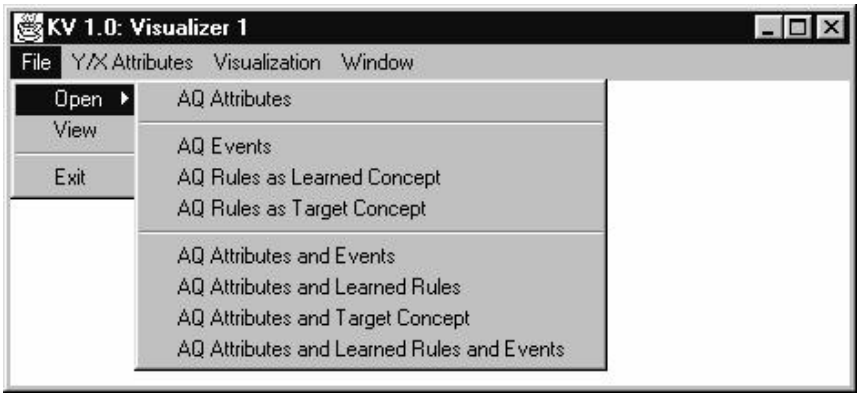

Fig. 2 Various Open operations.

In this menu, Open means opening a file and getting the data. The current KV requires that all data be input *through a file*. KV can input all the data (AQ-style) in the file or you can *extract* part of it. This flexibility is also shown in Fig. 2. Note that if you want to input examples or rules, you must select an option which will input variables section or this variables section was already input., because KV parses rules and examples according to this section. For a large file, KV takes some time to do a analysis job because KV does not assume any restrictions upon data, for example, predefined numbers of examples or classes.

Example: First input the Monk1 data from the file m1.aqin which contains attributes and training examples. Then input learned rules stored in the file m1.aqout. Finally input the file m1.target which contains the rules as the target concepts. There is no restriction on the input sequence of the last two files.

3.2.2 View

View in the File menu is only for viewing the contents of a file. Note that KV does not provide any editing function. To edit or create a file, use a native editor.

3.3 Arrange Attributes in a Representation Space

In a representation space, an attribute can be put either in Y dimension of a diagram or in X dimension and is called Y attribute or X attribute respectively. One can also ignore an attribute. KV provides three ways of arranging attributes. See Fig. 3.

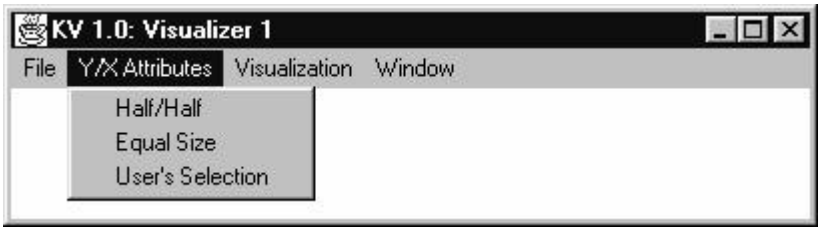

Fig. 3. Three ways of arranging attributes in a representation space.

Half/Half means KV automatically takes the first half of all the attributes, defined in the variables section and in the order there, as Y attributes and all others as X attributes. Half/Half is very likely to generate a representation space which is a rectangular, very long in one dimension and hence inconvenient to visual examination. Equal Size is provided to address this issue. KV automatically examines the number of attributes and their levels and attempt to define a space which is approximately a square. See Fig. 4 for the effects of using this feature on the data of wind bracing design problem.

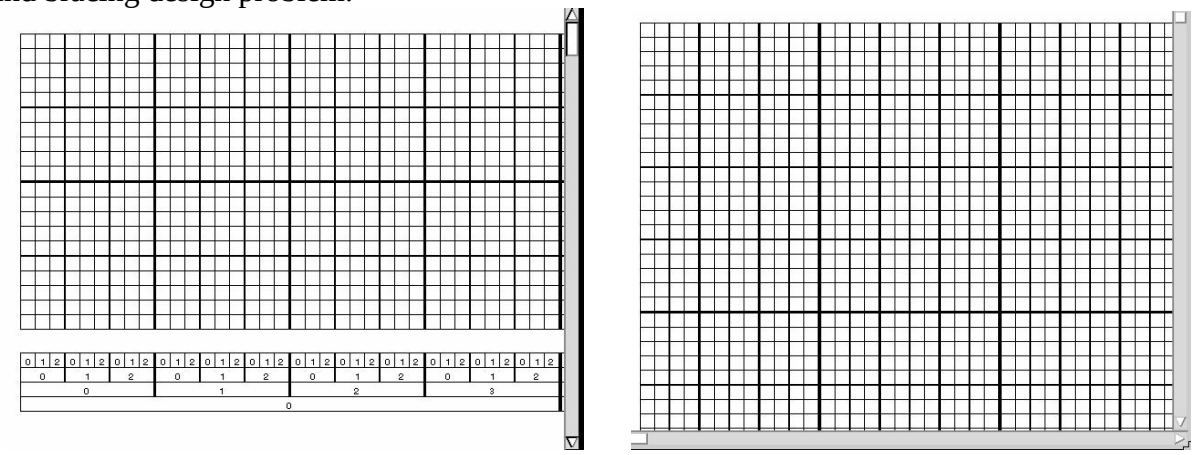

Fig. 4. Representation spaces of wind bracing design problem. The left is generated by Half/Half, and the right by Equal Size.

Users can also select and put any attributes in either dimension or put an attribute in both dimensions or even ignore some attributes. User's Selection is designed for this purpose. Fig. 5 shows the selection of attributes x1, x2, x3 in Y dimension, and x6, x7 in X dimension. Attributes x4, x5 are ignored. *Note that at least one attribute must be selected for Y dimension and there can be no attribute in X dimension*. Fig. 6 is the representation space after this selection. Ignoring, i.e.,

deleting some attributes is actually performing constructive induction (Michalski, 1983; Bloedorn, 1996).

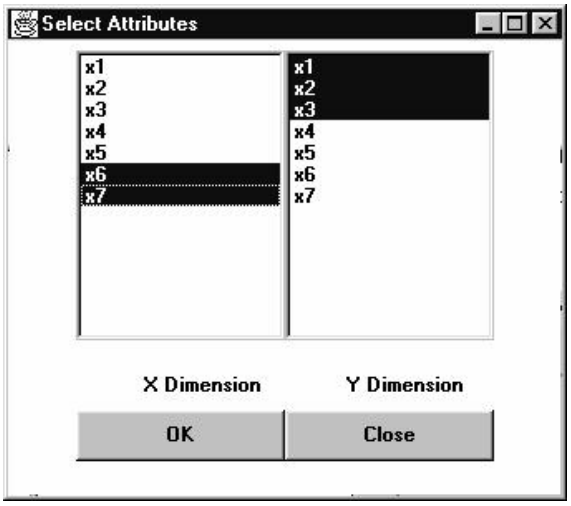

Fig. 5. Selection of attributes of wind bracing design data for Y and X dimensions.

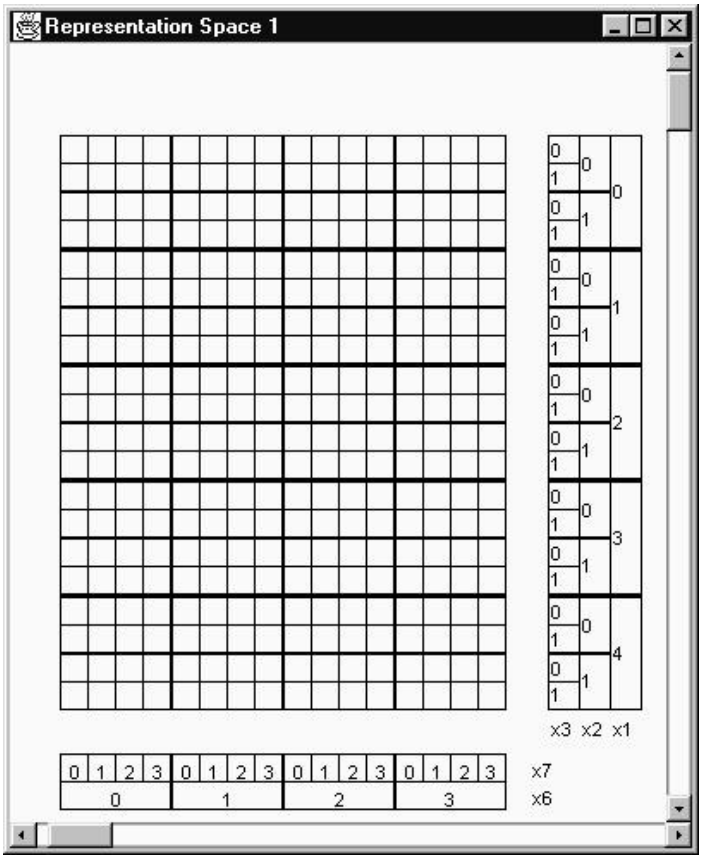

Fig. 6. The representation space of the wind bracing design problem after selection of attributes x1, x2, x3, x6 and x7.

Note that in any input operation containing input of attributes, Half/Half division of attributes is automatically performed, i.e., Half/Half is the default division.

3.4 Visualization

For various visualization operations provided in KV 1.0, see Fig. 7. *Note that these various visualization functions are effective only after Y/X attributes have been defined and any visualization operation is ineffective if attributes are not defined or input.*

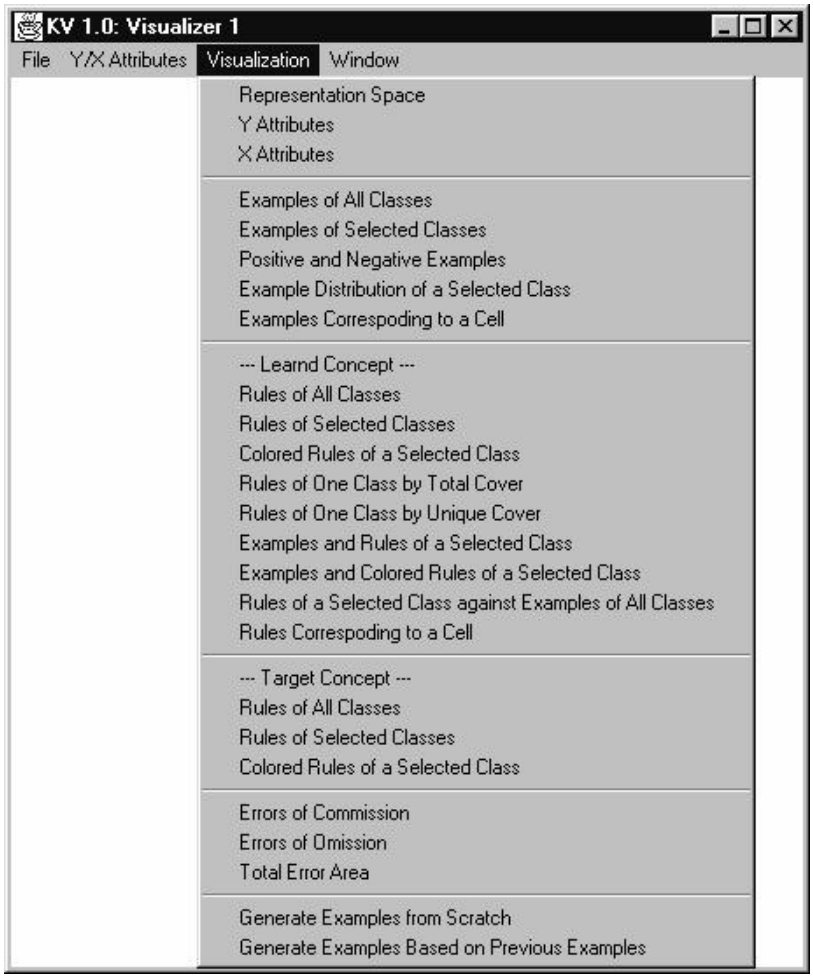

Fig. 7. Various operations in the Visualization menu.

3.4.1 Representation Space

It simply displays a representation space similar to Fig. 6 after Y/X attributes were determined. No examples or rules are displayed.

3.4.2 Y Attributes and X Attributes

These two functions are very useful when a displayed representation space is too large to be held in one screen. In this case, users can lose accurate understanding of specific cells in this diagram. To address this, users can display Y an X dimensional attributes in two separate windows and move their scroll bars for exact quantitative relationships. See Fig. 8.

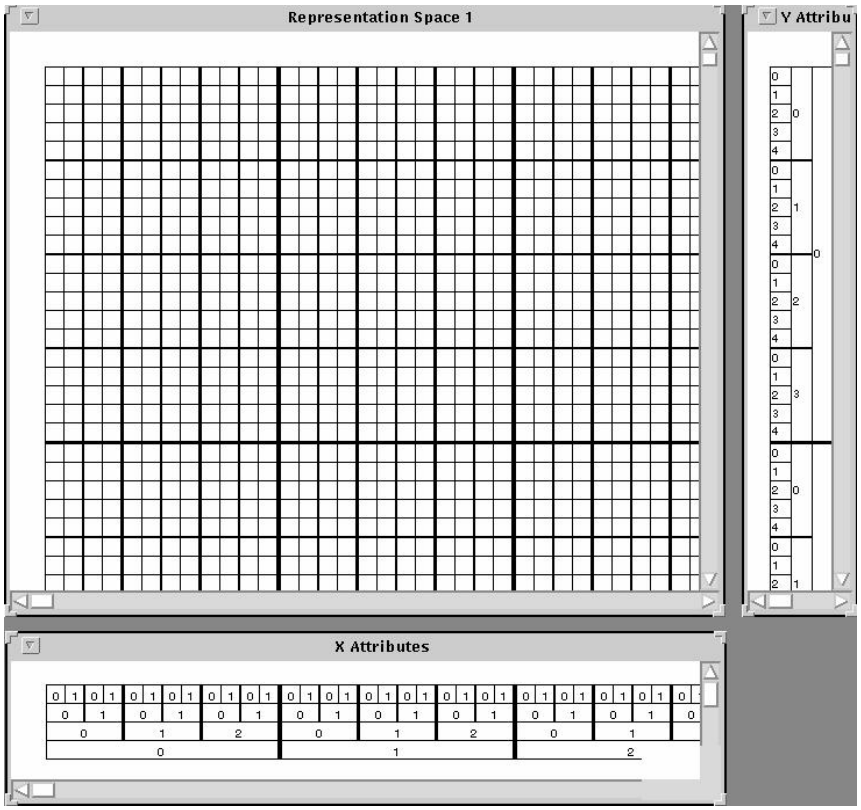

Fig. 8. The Y Attributes and X Attributes are used for an auxiliary purpose.

3.4.3 Examples of All Classes

Take the Monk1 problem as an example, and see Fig. 9.Actually this diagram shows a qualitative distribution of all the examples. Each class is labeled by using a different symbol (starting from 1 to 9, then followed by a..z, by A..Z) and color (in the order of red, green, pink, light blue, orange, yellow, magenta, cyan and light gray. After these nine colors, KV generate a random color for other classes). If the examples of two different classes occupy the same cell (i.e., the same event appears as a member of more than one class), then an X will be displayed in that cell instead of any other symbol and color.

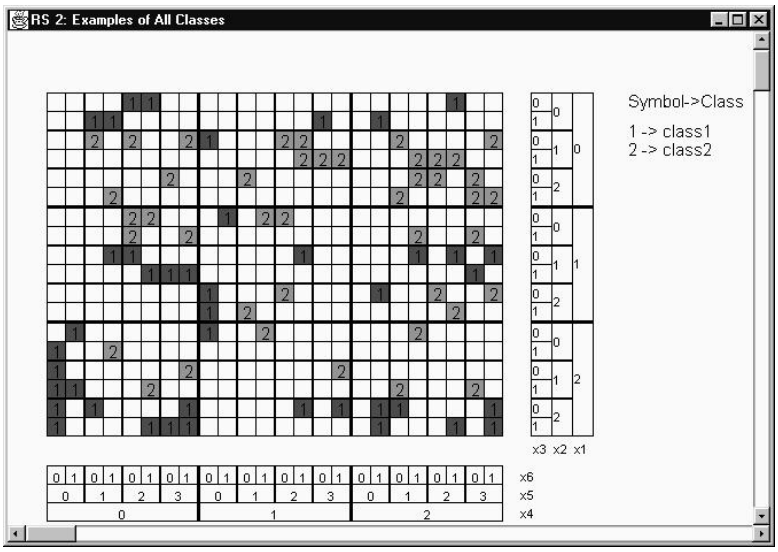

Fig. 9. The training examples of all classes of the Monk1 problem.

3.4.4 Examples of Selected Classes

Through this function, one can select examples of any number of classes to view. Take the data of the wind bracing design problem as an example, and select the examples of class1 and class3 for display. When examples of more than one class occupy the same cell, this function can give users an a more accurate visualization. See Fig. 10.

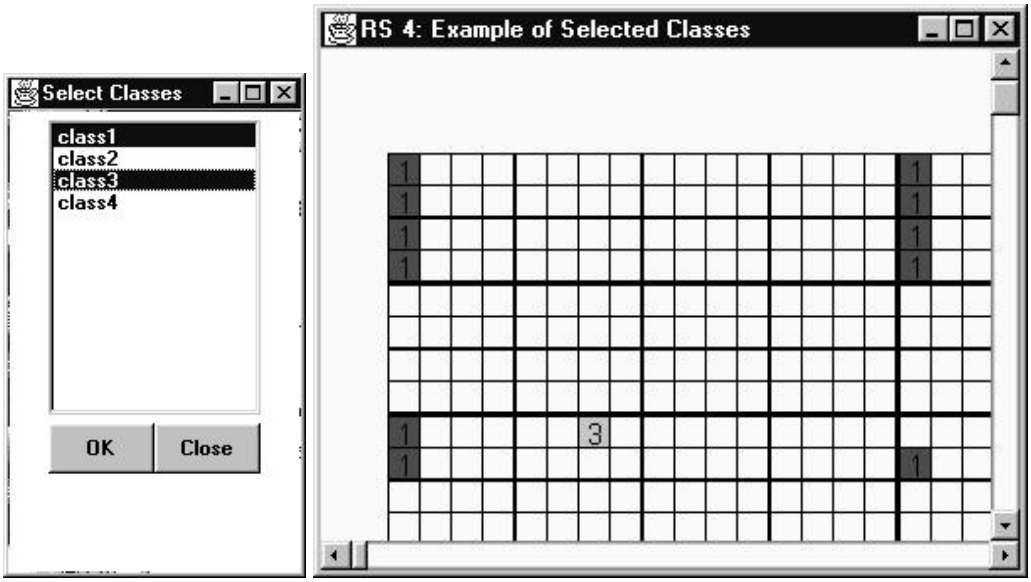

Fig. 10. Select the examples of class1 and class3 of the wind bracing design problem for display. 1 corresponds to class1 and 2 class3.

3.4.5 Positive and Negative Examples

If one class is selected as positive, then all other classes are considered negative. The examples of the positive class is displayed using red symbol "+" and the examples of all other classes using blue symbol "-". See Fig. 11 for an example.

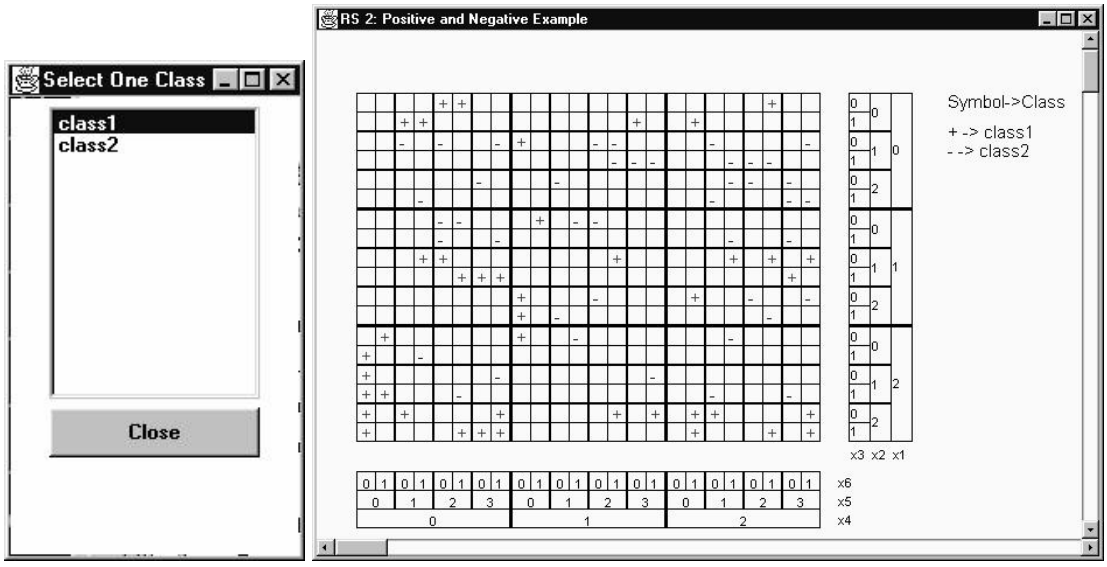

Fig. 11. The examples of class1 of the Monk1 problem are selected as positive.

3.4.6 Example Distribution of a Selected Class

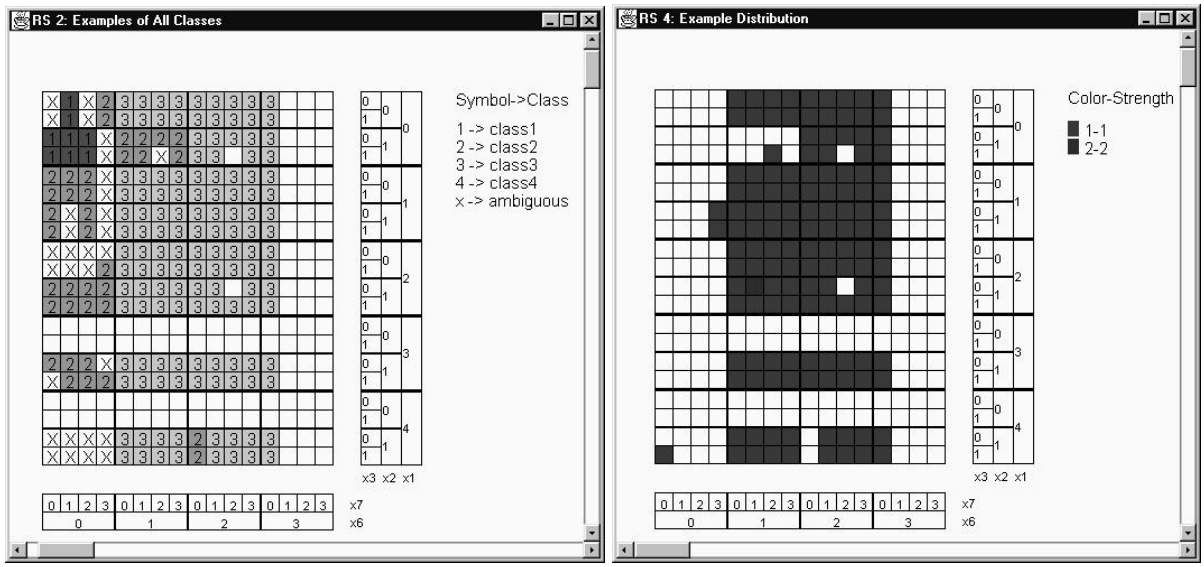

Fig. 12. The left is all examples of the wind bracing design problem based only on attributes x1, x2, x3, x6, x7. The right is example distribution of class3 only.

KV provides users an opportunity of viewing the quantitative distribution of examples of only one class per diagram. It uses colors to indicate the number of examples, called *strength* in the diagram, that occupy the same cell. Fig. 12 shows the distribution of examples of class3 of the wind bracing design problem. KV uses only ten levels of strength. More precisely, it uniformly divides the strength into ten levels, with darker color showing stronger strength. Comparing the two images in Fig. 12, one can have better understanding.

3.4.7 Examples Corresponding to a Cell

When examining a cell in a representation space, one probably wants to know exactly how many examples occupy this cell and what their classes are. "Examples Corresponding to a Cell" is implemented for this purpose. Fig. 13 shows all examples of the Monk1 problem when only attributes x1, x3, x4, x5 x6 are used.

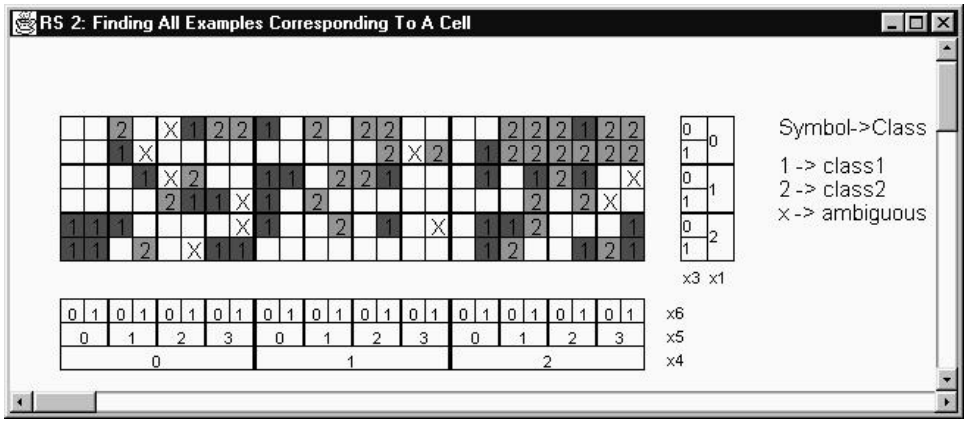

Fig. 13. The training examples of all classes of the Monk1 problem when only attributes x1, x3, x4, x5 x6 are used.

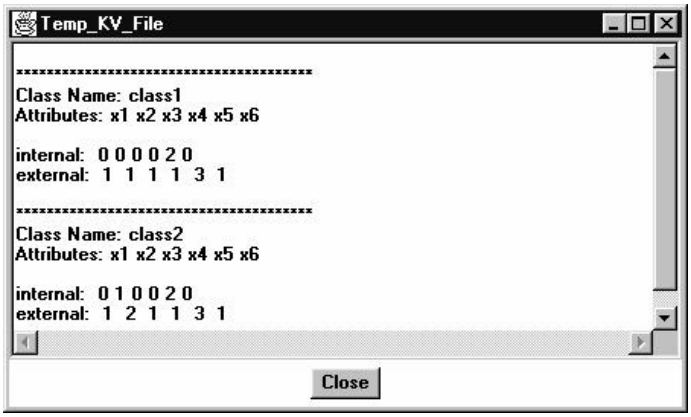

Fig. 14. The examples occupying the cell x1=0, x3=0, x4=0, x5=2, x6=0.

Clicking the cell x1=0, x3=0, x4=0, x5=2, x6=0, one can get a list of examples occupying this cell. See Fig. 14. Note that “internal” means the internal attribute values used by KV and that “external” means their nominal values seen in an input file. Clicking the cell x1=2, x3=1, x4=0, x5=0, x6=0, one can get a list of examples occupying this cell. See Fig. 15.

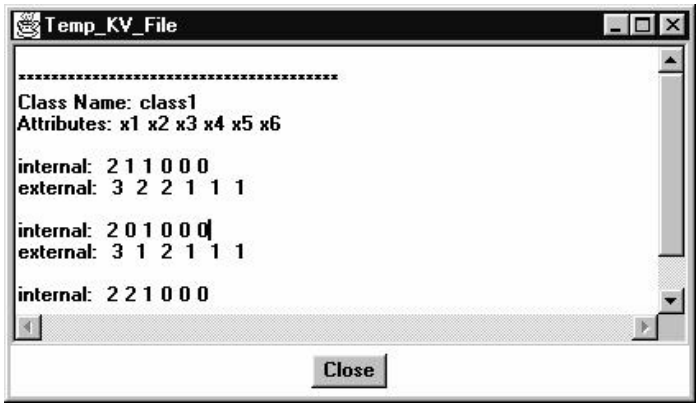

Fig. 15. The training examples occupying the cell x1=2, x3=1, x4=0, x5=0, x6=0.

Please check Fig. 10 and exact the correctness of Fig. 16 and Fig. 17.

3.4.8 Learned Rules of All Classes

Fig. 16 shows all the rules (spatial coverage precisely) learned by AQ15c from the training examples in the file m1.aqin. A cell with the symbol “X” means rules of at least two different classes cover this cell.

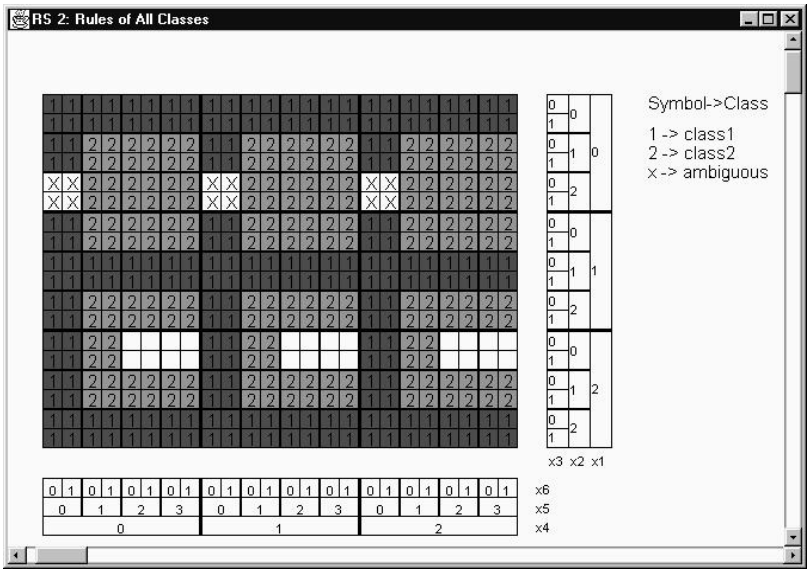

Fig. 16. The learned rules of all classes of the Monk1 problem.

3.4.9 Learned Rules of Selected Classes

Users can select any number of classes and then display their rules. Fig. 17 shows the rules of class1 of the Monk1 problem.

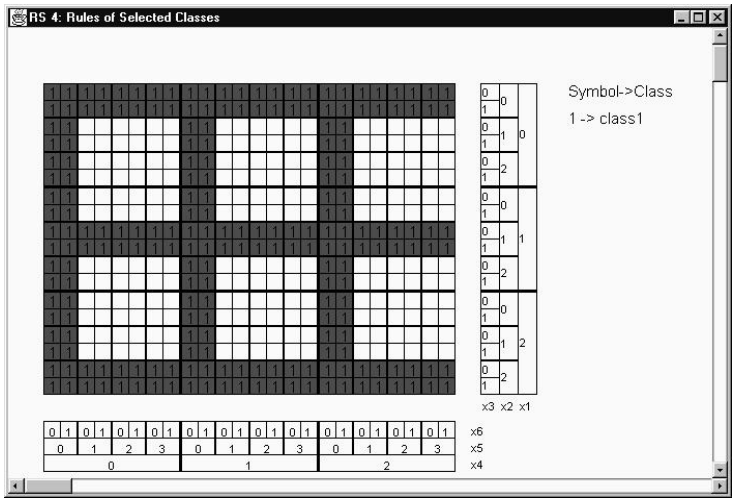

Fig. 17. The learned rules of class1 of the Monk1 problem.

3.4.10 Colored Rules of a Selected Class

This feature is for displaying individual rules of a selected class. Using this feature, one can find out the exact coverage of each rule. Each rule uses a different symbol and color for display. Fig. 18 shows the four rules of class1 of the Monk1 problem. When two or more rules cover the same cell, the symbol and color of the strongest rule is selected.

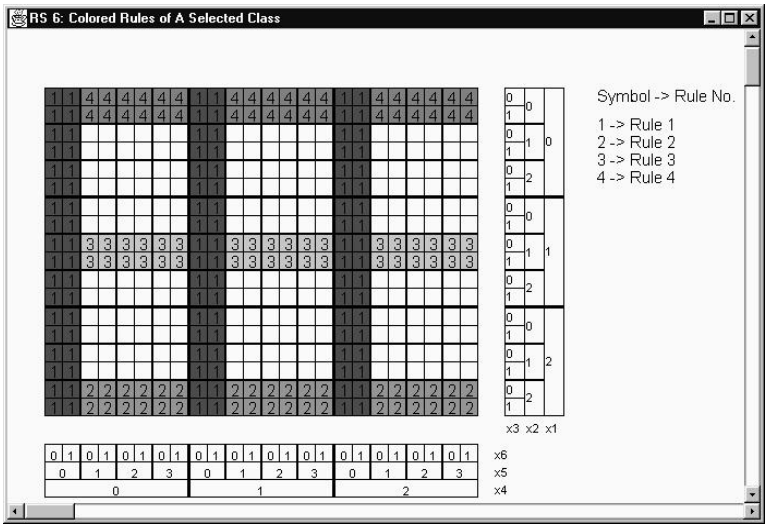

Fig. 18. The individual learned rules of class1 of the Monk1 problem.

3.4.11 Learned Rules of One Class by Total Cover

The cover measures used in AQ rules reveal the strength of each rule. KV allows users to view rule covers of only one class per diagram. Fig. 19 is the diagram of rule total covers of class1 of the Monk1 problem. The darker color represents the stronger cover, i.e., stronger pattern. KV uses only ten levels to differentiate covers strength. Clearly, Fig. 19 gives users more understanding of learned rules.

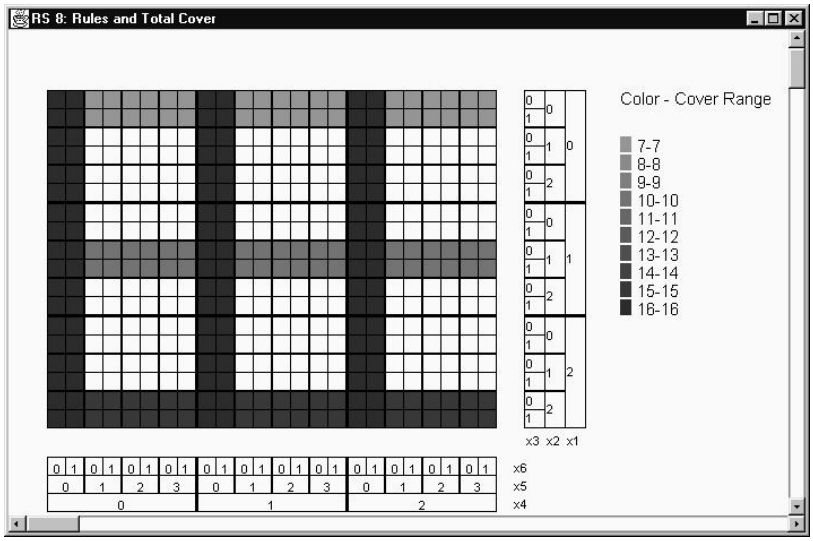

Fig. 19. The rules' total cover of class1 of the Monk1 problem.

3.4.12 Learned Rules of One Class by Unique Cover

KV also provides the visualization of rule unique covers.

3.4.13 Rules and Examples of a Selected Class

One can input any rules and examples (AQ style) and then visualize their covering relationships. Fig. 20 shows how training examples of class1 of the Monk1 problem are covered by the learned rules of this class. Red color means intersecting, i.e., both examples and rules occupy the same cells; green color represents an uncovered example; blue color represents unused rule cells, i.e., the generalization area. Since in this example, AQ output a set of rules consistent with these example, no examples were uncovered. The *example covered rate* in Fig. 20 means the percentage of examples covered by the rules of their own class.

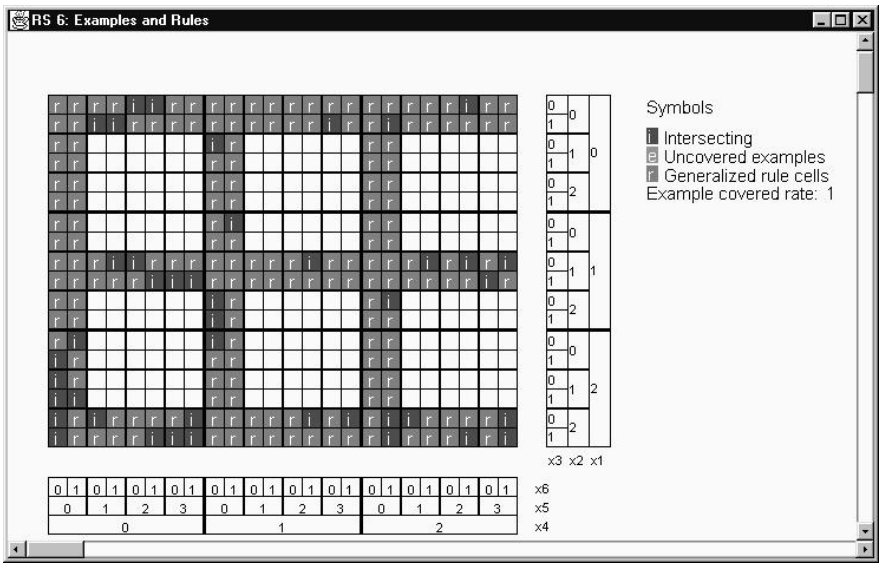

Fig. 20. The covering relationship between examples and rules of class1 in the Monk1 problem.

3.4.14 Examples and Colored Rules of a Selected Class

This feature is similar to the above one, except for each rule being colored. From Fig. 20, it can be seen that no example is uncovered. Just for illustration, let's delete the first rule of class1 in the Monk1 problem and input this modified rule set and display examples and rules of class1 in Fig. 21.

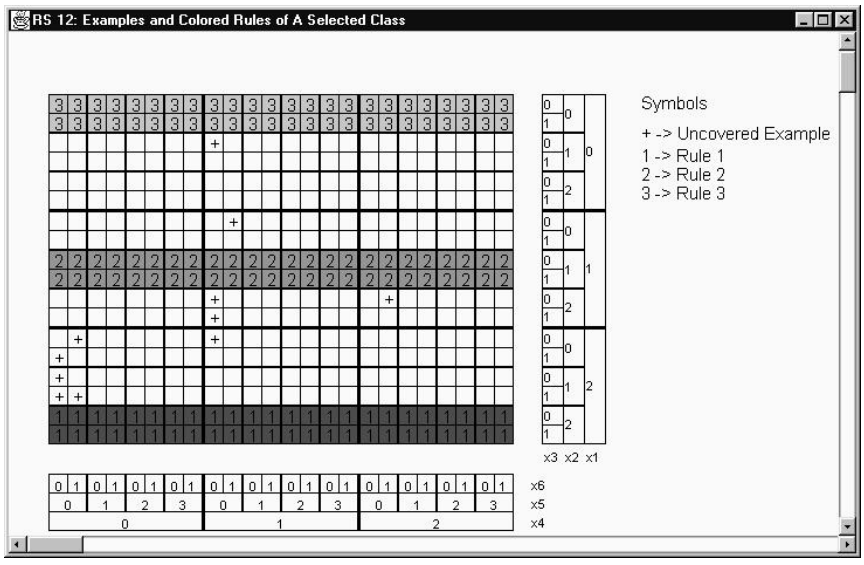

Fig. 21. The examples and colored rules (after deleting the first rule temporarily) of class1 in the Monk1 problem.

3.4.15 Rules of a Selected Class against Examples of All Classes

In this feature, the first five rules (i.e., the strongest five rules) of the selected class are displayed in five different colors, darker color meaning stronger cover, and all other rules are displayed by using the lightest green color. Using this feature, one can visualize the covering relationship between the selected rules and examples of all classes. See Fig. 22.

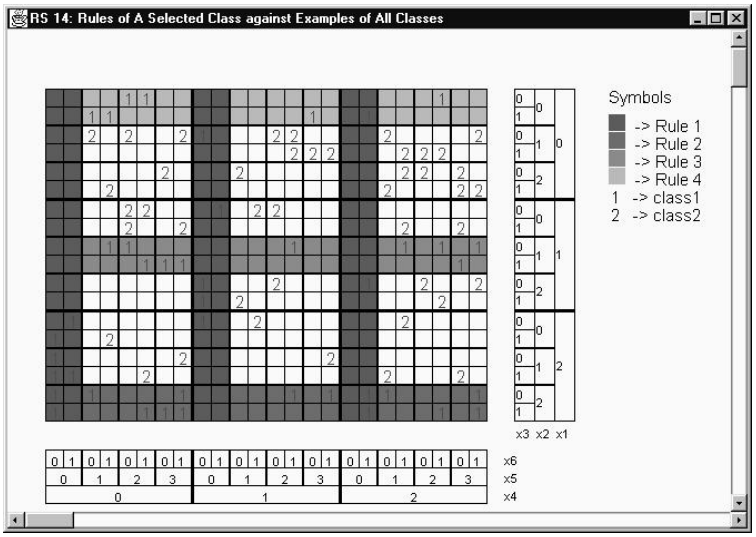

Fig. 22. The rules of class1 in the Monk1 problem against examples of all classes.

3.4.16 Rules Corresponding to a Cell

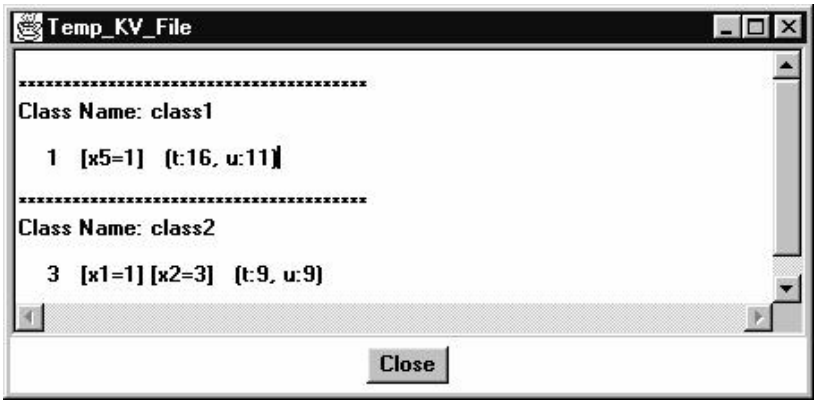

Fig. 23. The list of rules occupying the cell x1=0, x2=2, x3=0, x4=0, x5=0, x6=0.

It is often the case that rules of the same or different classes occupy some same cells. If one wants to know what these rules are and their classes, then use this feature and click a representa-

tion space cell. See Fig. 16, and click the cell x1=0, x2=2, x3=0, x4=0, x5=0, x6=0, one can get a list of rules covering this cell.(see Fig. 23). Note that the attribute values in Fig. 23 are their nominal values, not KV's internal values.

3.4.17 Rules of All Classes of the Target Concept

This feature is similar to Learned Rules of All Classes. Input the target rules in the file m1.target and display all the rules of the target concepts of the Monk1 problem in Fig. 24.

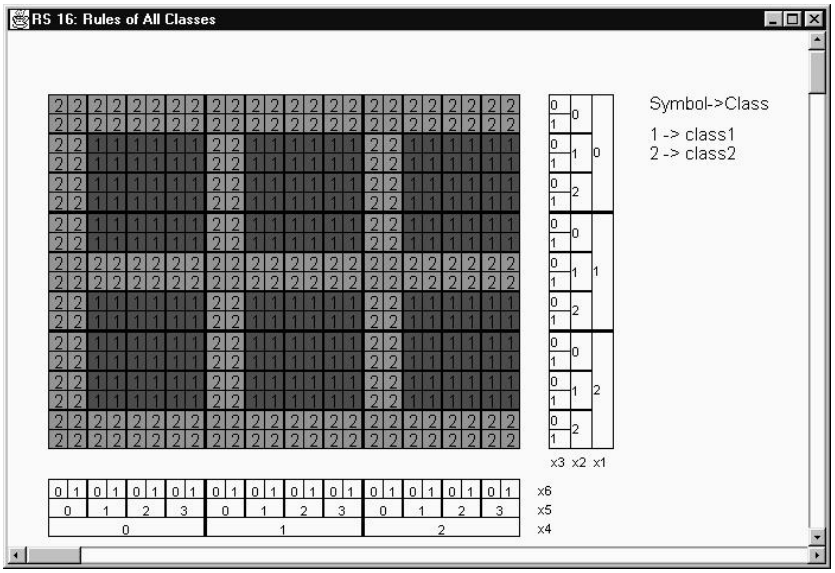

Fig. 24. The target concept of the Monk1 problem.

3.4.18 Rules of Selected Classes of the Target Concept

This function is totally the same as Learned Rules of Selected Classes.

3.4.19 Colored Rules of a Selected Class of the Target Concept

This function is totally the same as Learned Rules of Selected Classes.

3.4.20 Errors of Commission, Errors of Omission, and Total Errors

Compare Fig. 24 to Fig. 16 and find the difference, i.e., the areas covered by the learned rules for class2 is not exactly the same as those of the target concept. This difference can be displayed in three ways: errors of commission, errors of omission and total errors. Fig. 25 provides visualiza-

tion of total errors.

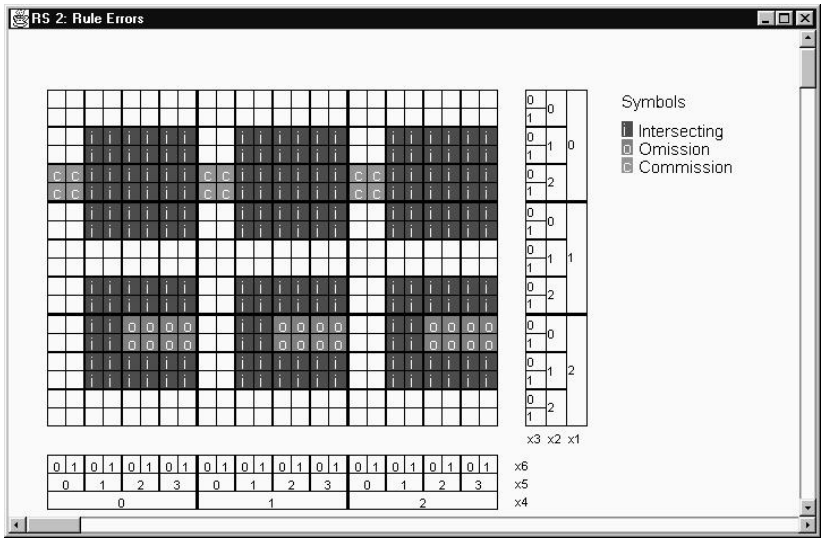

Fig. 25. Errors of learned rules for class2 in the Monk1 problem.

3.4.21 Generate Examples from Scratch

It is not unusual for one to desire to designing some learning problems to challenge or test existing or being-developed learning algorithms. With KV's attractive visual display, one can select this feature and generate examples at his/her will. The first step is to select a class for which he/she want to create examples. The second step is clicking the left mouse key for creating an example or clicking the right mouse key to delete an example. Note that the definition of attributes and classes must be input from a file. One must create a file similar to m1.aqin (with one or more valid examples under each class). Also note that one mouse click creates or deletes only one example.

3.4.22 Generate Examples Based on Previous Examples

This feature is similar to the above one, except that one can directly work on previous examples he/she has.

3.5 Other Operations

3.5.1 Another Visualizer

Knowledge visualizer only remembers the data of and display diagrams for the problem whose

data was input last recently. If one wants to visualize different problems or different data of the same problem, he/she can select this function.. This feature is useful for visualization of the effects of constructive induction or modifications to the current data (attributes, examples or rules). Note that choosing this feature needs more memory.

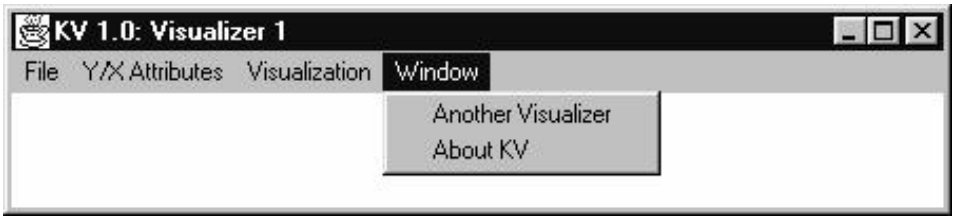

Fig. 29. The window menu.

4 OTHER TOPICS

4.1 Constructive Induction

A popular view of constructive induction is that learning via any changes in the representation space in order to achieve better results is considered constructive induction. We consider two situations: deleting or adding attributes.

For visualization of the effects of deleting attributes, one can first perform the operation of selecting attributes in the menu of Y/X Attributes. Using User's Selection to select some attributes and ignore others lead to the effects of deleting attributes. Only after this can one perform various operations listed in the menu of Visualization. Any operation generates shrunk representation spaces.

For visualization of the effects adding attributes, one has to put all new data in *files* and input them. Then go through the menus of Y/X Attributes and Visualization.

If one wants to change values of an attribute, he/she must put changes in a file first and then input changed data.

4.2 PROJECTION OF EXAMPLES

Users can project examples of only one class per diagram onto one attribute and visualize this projection quantitatively. For projection, users have to go back to the menu of Y/X Attributes and select that attribute for Y dimension and then visualize the projection using Example Distribution of a Selected Class in the menu of Visualization. Fig. 30 shows the projection of the

examples of class1 of the Monk1 problem onto attribute x5. Users can check the quantitative relationship between this figure and other figures above. Generally speaking, users can project examples of any class onto any subset of the given attributes.

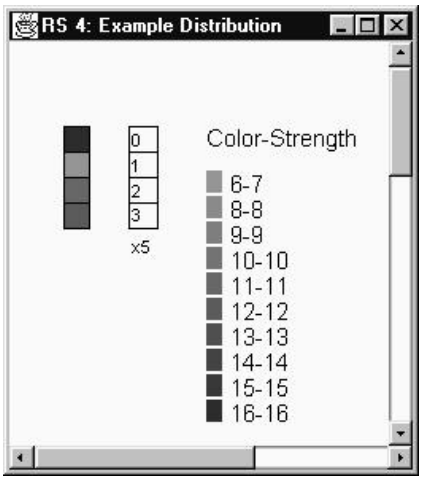

Fig. 30. The projection of the examples of class1 of the Monk1 problem onto attribute x5.

4.3 Internal Values of a Representation Space

In the two dimensions of any diagram, the attribute values displayed are actually the internal values of the AQ learning program, integers starting form 0. If an attribute is numerical, its discretized attributes should be integers and start from 0 for AQ to work. If an attribute is nominal, the name section of its input file should specify how its nominal values correspond to AQ's internal values. So in any case, users know the correspondence of the attribute values in a diagram to their external actual values. Users can use his/her native editor or View function in the menu of File to check the correspondence simultaneously.

5 CONCLUSION

This report describes a novel knowledge visualization software called KV. It has many advantages over previous work. It is especially good for machine learning research and data mining. The above simply describes individual operations; however, creative combination of various visualization operations can make KV more powerful and effective. Some of planned features are listed in the following.

(1) Upgrade KV to Java 1.1. Currently, users have to use additional software (e.g., in Sparc

workstation, a software called snapshot; in Mac, use three keys apple+shift+3 to capture a window; in IBM PC, use two keys alt+print_screen) to cut or capture a graphic display of a diagram in order to save it or to insert it in a file. Using Java 1.1, representation spaces can be easily saved in a file as images for future use.

(2) Add interfaces to formats adopted by other important or popular learning systems. KV 1.0 now only accepts inputs from AQ learning systems. It will provide interfaces to C4.5 (Quinlan, 1993), INLEN (Michalski et al., 1992; Kaufman and Michalski, 1996), etc.

(3) Zooming function. For a large representation space, it is difficult for a user to manage to examine the representation space as a whole. He or she can only visualize it portion by portion. Future KV will address this issue and make users able to get the whole look-and-feel for some moderately large problems.

(4) Accepting rules or examples from a window. It would be more convenient for a user, only through a window, not through a file, to input and test just one or a few rules invented by him or her to see the effects.

APPENDIX 1: Monk1 Data

The following is the input file (called m1.aqin in the KV 1.0 package) to the AQ learning program. In this file, there are 86 training examples.

```
parameters
 run  mode  ambig  trim  wts  maxstar  echo    criteria  verbose
  1   ic    neg    mini  cpx  10       pv      default   1


 variables
   #  type  levels  cost  name
   1  nom      3    1.00  x1
   2  nom      3    1.00  x2
   3  nom      2    1.00  x3
   4  nom      3    1.00  x4
   5  nom      4    1.00  x5
   6  nom      2    1.00  x6

x1-names
 value name
 0     1
 1     2
 2     3

x2-names
```

```
 value name
 0     1
 1     2
 2     3

x3-names
 value name
 0     1
 1     2

x4-names
 value name
 0     1
 1     2
 2     3

x5-names
 value name
 0     1
 1     2
 2     3
 3     4

x6-names
 value name
 0     1
 1     2
```

```
class1-events
 x1 x2 x3 x4 x5 x6
 3  3  1  3  4  2
 2  2  1  1  2  2
 2  3  2  2  1  1
 2  2  1  3  3  2
 3  3  1  3  2  1
 2  2  1  3  4  2
 3  3  2  3  3  2
 1  1  1  1  3  2
 2  2  2  1  3  2
 1  1  2  1  2  2
 3  3  1  1  1  1
 1  1  2  2  4  1
 1  1  1  1  3  1
 3  3  2  3  1  2
 1  1  2  3  1  2
 3  3  1  1  2  1
 3  3  1  2  4  2
 2  1  1  2  1  2
 3  2  2  1  1  1
 1  1  2  1  2  1
 3  2  2  1  1  2
 3  3  2  3  4  2
 1  1  1  3  3  2
 3  3  1  3  1  2
 2  2  2  1  4  1
 3  3  2  1  4  2
 2  2  1  2  3  2
 2  2  2  3  4  1
 2  2  2  1  4  2
 3  3  2  1  3  2
 2  3  1  3  1  2
 3  3  2  1  4  1
 3  1  1  2  1  1
 3  3  1  1  4  2
 3  1  1  1  1  2
 3  3  1  2  3  2
 3  3  2  1  1  1
 2  2  1  1  3  1
 2  2  1  3  2  2
 3  1  2  1  1  1
 1  2  1  2  1  1
 3  2  1  1  1  1
 2  3  1  2  1  1
```

```
class2-events
 x1 x2 x3 x4 x5 x6
 3  2  1  1  4  2
 2  1  2  1  3  1
 1  2  2  3  2  2
 2  3  1  3  4  2
 1  2  1  1  4  2
 2  1  1  2  3  1
 1  3  1  3  2  2
 3  2  1  2  4  2
 2  1  2  1  4  2
 2  1  2  3  4  1
 1  2  2  3  3  2
 3  1  1  2  2  2
 2  1  1  1  3  2
 1  2  1  1  3  1
 3  2  2  3  4  1
 2  3  2  3  3  2
 2  3  1  3  3  1
 1  2  2  3  3  1
 1  2  1  2  3  1
 3  2  2  3  2  1
 1  3  2  3  4  2
 1  3  1  1  4  1
 1  2  2  2  4  2
 1  2  1  3  4  2
 3  2  2  1  3  2
 1  2  2  2  3  2
 2  1  1  1  3  1
 1  2  1  2  3  2
 1  3  2  3  2  1
 2  3  2  2  2  1
 1  3  1  2  2  1
 1  2  2  2  4  1
 1  2  1  3  2  1
 2  1  2  3  2  2
 3  1  2  1  2  2
 1  3  1  3  4  1
 2  1  1  2  2  2
 1  3  1  3  3  1
 1  3  2  1  2  2
 1  2  1  1  2  1
 1  3  2  3  4  1
 2  3  1  2  3  1
 3  1  1  3  2  2
```

Below is the output file (called m1.aqout in the KV 1.0 package) from AQ.

```
parameters
run   mode   ambig   trim   wts   maxstar  echo     criteria  verbose
 1    ic     neg     mini   cpx   10       pv       default   1

variables
  #   type    size    cost          name
  1   nom       3     1.00          x1.x1
  2   nom       3     1.00          x2.x2
  3   nom       2     1.00          x3.x3
  4   nom       3     1.00          x4.x4
  5   nom       4     1.00          x5.x5
  6   nom       2     1.00          x6.x6

class1-outhypo
  #   cpx
  1   [x5=1]   (t:16, u:11)
  2   [x1=3] [x2=3]   (t:15, u:11)
  3   [x1=2] [x2=2]   (t:10, u:10)
  4   [x1=1] [x2=1]   (t:7, u:6)

class2-outhypo
  #   cpx
  1   [x1=1,3] [x2=2] [x5=2,3,4]   (t:18, u:18)
  2   [x1=2] [x2=1,3] [x5=2,3,4]   (t:13, u:13)
  3   [x1=1] [x2=3]   (t:9, u:9)
  4   [x1=3] [x2=1] [x5=2]   (t:3, u:3)

This learning used:
 System time:      0.333 seconds
 User time:        0.00  seconds
```

One version of the target concepts expressed in rules and stored in the file m1.target in the KV 1.0 package is:

```
class1-outhypo
  #   cpx
  1   [x5=1]
  2   [x1=3] [x2=3]
  3   [x1=2] [x2=2]
  4   [x1=1] [x2=1]

class2-outhypo
  #   cpx
  1   [x1=1] [x2=2..3] [x5=2..4]
  2   [x1=2..3] [x2=1] [x5=2..4]
  3   [x1=3] [x2=2] [x5=2..4]
  4   [x1=2] [x2=3] [x5=2..4]
```

APPENDIX 2: Wind Bracing Data

Wind bracing design data are stored in two files: windbracing.aqin and windbracing.aqout. The following is the training data file, windbracing.aqin. In this file, there are 335 training examples.

```
parameters
run   mode   ambig   trim   wts   maxstar   echo   verbose
 1      ic     empty   spec   cpx   10        pcdv       3

 domaintypes
name type levels
l2   nom  2
l3   nom  3
l4   nom  4
l5   nom  5

 l2-names
value name
0     1
1     2

 l3-names
value name
0     1
1     2
2     3

 l4-names
value name
0     1
1     2
2     3
3     4

 l5-names
value name
0     1
1     2
2     3
3     4
4     5

variables
#   name
1   x1.l5
2   x2.l2
3   x3.l2
4   x4.l3
5   x5.l3
6   x6.l4
7   x7.l4

class1-events                x1 x2 x3 x4 x5 x6 x7          1  2  2  3  2  1  3                1  2  2  3  2  1  2
```

3 1 2 1 2 1 1
1 1 2 3 1 1 2
1 2 2 1 3 1 1
1 1 1 3 2 1 2
1 1 2 1 1 1 1
1 2 1 1 3 1 1
1 2 1 1 1 1 1
1 1 2 3 2 1 2
2 1 1 3 2 1 4
1 1 1 3 1 1 2
1 2 1 3 1 1 4
3 1 1 3 1 1 3
1 2 1 3 2 1 4
1 1 1 3 1 1 3
1 2 1 3 1 1 2
2 2 2 3 1 1 2
1 1 1 1 1 1 1
4 2 1 3 3 1 4
1 1 2 3 1 1 3
1 2 1 3 2 1 2
1 2 1 3 3 1 2
1 2 2 1 2 1 1
1 2 2 3 3 1 2
1 2 2 3 1 1 2
1 1 2 1 2 1 1
1 2 1 3 3 1 3
1 1 1 3 2 1 3
2 1 2 3 2 1 4
1 2 1 1 2 1 1
1 1 1 3 3 1 2
1 1 2 3 3 1 2
5 2 2 3 2 1 2
1 2 2 3 1 1 4
1 2 2 3 3 1 3
3 1 1 1 1 1 1
1 2 2 3 2 1 4
3 1 1 3 1 1 2
3 1 2 3 1 1 2
1 1 2 3 2 1 3
3 1 2 3 1 1 3
1 1 1 1 2 1 1
3 1 1 3 1 1 4
4 2 2 1 2 1 1
1 2 2 1 1 1 1
3 1 2 1 1 1 1
5 2 2 3 2 1 3
1 2 1 3 2 1 3
1 2 2 3 1 1 3
2 2 1 3 1 1 2
1 2 1 3 1 1 3
5 2 1 3 2 1 3

class2-events
x1 x2 x3 x4 x5 x6 x7
2 1 1 1 3 1 1
5 2 2 1 3 1 1
3 2 1 3 2 1 4

3 2 2 3 2 1 2
1 1 2 3 3 1 3
1 1 1 3 1 1 4
5 2 1 3 2 1 1
3 1 1 1 3 1 1
3 2 2 3 3 1 3
2 2 2 3 2 1 4
2 2 1 3 1 1 4
2 1 2 1 2 1 1
3 2 2 3 3 1 2
3 2 1 3 1 1 2
4 2 2 3 2 1 4
2 2 1 1 3 1 1
1 2 2 2 1 2 3
2 1 1 3 1 1 3
3 1 1 3 3 1 2
1 1 1 3 3 1 4
2 2 2 3 3 1 2
4 2 2 3 2 1 3
2 1 2 3 3 1 3
2 1 2 3 3 1 2
2 2 2 3 2 1 2
4 2 2 1 3 1 1
5 2 2 3 2 1 4
3 1 1 3 3 1 4
2 2 1 3 2 1 2
2 2 2 3 1 1 3
5 2 1 3 2 1 4
4 2 2 3 3 1 3
3 1 2 3 2 1 3
5 2 2 3 3 1 3
2 2 1 1 2 1 1
2 2 1 3 3 1 2
3 2 1 3 1 1 2
2 2 1 3 2 1 4
3 2 1 3 3 1 3
4 2 1 3 1 1 2
5 2 1 1 3 1 1
1 2 2 2 1 2 1
3 1 2 3 1 1 4
1 1 1 3 3 1 3
3 2 1 1 1 1 1
2 1 1 1 1 1 1
2 2 2 1 3 1 1
3 2 2 1 1 1 1
2 1 2 3 1 1 4

4 2 2 3 1 1 3
2 1 1 3 1 1 4
4 2 2 3 1 1 4
1 1 2 1 3 1 1
3 2 1 1 1 1 1
4 2 1 3 3 1 3
2 1 1 1 2 1 1
2 1 2 3 1 1 3
3 2 1 3 3 1 2
1 2 2 3 3 1 4
3 1 1 3 2 1 2
2 1 1 3 2 1 3
2 1 1 3 3 1 3
3 2 1 1 2 1 1
2 2 1 1 1 1 1
1 1 2 3 1 1 4
1 1 2 3 3 1 4
3 1 2 3 3 1 2
2 2 2 3 1 1 3
3 1 2 1 3 1 1
3 1 1 3 2 1 4
1 1 1 3 2 1 4
3 2 1 3 2 1 2
4 2 2 3 2 1 2
2 2 1 3 2 1 3
3 2 2 3 2 1 3
2 1 2 3 2 1 2
2 1 2 1 3 1 1
1 2 2 2 1 2 4
2 1 2 3 3 1 4
1 2 1 3 3 1 4
5 2 1 1 2 1 1
2 2 2 3 2 1 3
4 2 1 3 2 1 4
3 2 2 3 1 1 2
4 2 2 3 3 1 2
2 1 2 1 1 1 1
2 1 1 3 1 1 2
4 2 1 3 2 1 2
3 2 1 3 1 1 3
3 1 1 3 3 1 3
1 1 1 1 3 1 1
3 2 1 3 2 1 3
4 2 1 3 1 1 4
3 2 2 1 3 1 1
2 1 1 3 2 1 2
3 2 1 3 3 1 4
1 2 2 2 1 2 2
4 2 1 3 3 1 2
5 2 1 3 3 1 3
1 2 1 2 1 2 3
1 2 1 2 1 2 1
5 2 1 2 2 3 1
2 2 1 3 3 1 3
2 1 2 3 2 1 3
5 2 1 3 3 1 2
5 2 2 3 2 1 4

5 2 1 3 3 1 4
3 1 2 3 3 1 3
3 2 2 3 1 1 3
4 2 1 3 1 1 3
4 2 2 3 1 1 2
4 2 2 1 1 1 1
3 1 1 1 2 1 1
4 2 1 1 1 1 1
3 2 2 3 3 1 4
2 2 2 1 1 1 1
2 1 1 3 3 1 2
2 1 1 3 3 1 4
3 1 1 3 2 1 3
1 2 1 2 1 2 4
4 2 1 1 2 1 1
3 1 2 3 2 1 4
2 1 2 3 1 1 2
4 2 1 1 3 1 1
3 2 2 1 2 1 1
3 1 2 3 3 1 4

class3-events
x1 x2 x3 x4 x5 x6 x7
3 2 1 2 1 2 3
2 2 1 2 1 2 3
2 2 2 2 1 2 3
2 2 1 3 3 1 4
1 2 2 2 2 3 1
4 2 2 2 1 2 4
1 1 2 2 1 2 4
3 2 1 2 2 3 1
3 1 2 2 3 4 1
1 1 2 2 2 3 4
5 2 2 2 1 2 2
3 1 1 2 2 3 3
2 1 1 2 2 3 1
3 2 1 2 2 3 2
2 1 1 2 1 2 4
5 2 2 2 2 3 2
1 1 1 2 2 3 4
2 1 2 2 1 2 4
5 2 2 2 1 2 1
1 1 1 2 1 2 4
1 2 1 2 2 3 2
2 1 2 2 1 2 2
1 2 2 2 3 4 1
2 1 1 2 2 3 4
3 1 1 2 1 2 4
3 1 1 2 2 3 4
3 2 2 2 2 3 2
2 2 2 2 2 3 1
4 2 2 2 2 3 4
3 2 2 2 1 2 2
2 2 1 2 3 4 1
1 1 2 2 1 2 3
3 1 2 2 2 3 1
1 1 1 2 2 3 3

3 1 1 2 3 4 1
2 1 2 2 3 4 1
3 1 2 2 2 3 2
2 1 2 2 2 3 4
1 1 2 2 1 2 1
1 2 1 2 2 3 3
1 1 2 2 2 3 1
4 2 1 2 1 2 4
2 1 1 2 2 3 3
5 2 2 2 2 3 4
1 2 1 2 2 3 4
5 2 1 2 2 3 3
4 2 2 2 2 3 2
2 1 1 2 1 2 3
5 2 1 2 1 2 2
2 1 1 2 3 4 1
2 2 2 3 3 1 4
4 2 1 2 3 4 1
1 2 2 2 2 3 4
2 2 2 2 1 2 2
3 1 1 2 2 3 2
2 1 1 2 1 2 2
3 2 1 2 3 4 1
2 2 1 2 1 2 4
2 2 1 2 2 3 3
1 2 2 2 2 2 3
2 2 1 2 1 2 1
2 2 2 2 1 2 1
4 2 2 2 2 3 1

1 1 2 2 2 3 2
3 1 2 2 1 2 3
3 2 2 2 2 3 3
5 2 2 2 2 3 3
1 1 1 2 2 3 2
2 2 2 2 1 2 4
3 2 2 2 3 4 1
3 2 2 2 1 2 1
3 1 2 2 2 3 4
4 2 1 2 2 3 1
3 1 2 2 1 2 2
5 2 1 2 1 2 3
3 1 1 2 1 2 1
2 2 2 2 2 3 4
2 1 2 2 1 2 1
4 2 2 2 1 2 2
4 2 1 2 1 2 3
1 1 1 2 2 3 1
2 2 2 2 2 3 3
2 1 1 2 1 2 1
1 1 1 2 3 4 1
2 2 2 2 2 3 2
2 1 1 2 1 2 4
3 2 1 2 1 2 4
5 2 2 2 1 2 4
1 1 2 2 1 2 2
2 2 1 2 1 2 2
3 1 1 2 2 3 1
4 2 2 2 3 4 1
2 2 1 2 2 3 4

3 2 2 2 2 3 1
4 2 2 2 2 3 3
3 2 2 2 2 3 4
3 2 1 2 1 2 1
2 2 2 2 3 4 1
1 1 2 2 2 3 3
1 1 1 2 1 2 2
3 1 1 1 1 2 3
5 2 2 1 2 1 1
1 2 2 2 2 3 2
1 2 1 2 3 4 1
4 2 1 2 2 3 3
2 1 2 2 2 3 1
4 2 1 2 2 3 4
3 1 2 2 1 2 4
3 2 2 2 1 2 3
1 2 1 2 2 3 1
3 1 1 2 1 2 2
5 2 2 2 1 2 3
4 2 1 2 1 2 1
2 2 1 2 2 3 2
5 2 1 2 1 2 4
4 2 1 2 1 2 2
1 1 2 2 3 4 1
2 1 1 2 2 3 2
2 1 2 2 2 3 2
3 2 1 2 2 3 4
2 1 2 2 2 3 3
4 2 2 2 1 2 1

5 2 1 2 2 3 2
5 2 1 2 3 4 1
3 1 2 2 1 2 1
2 1 2 2 1 2 3
5 2 1 2 1 2 1
4 2 1 2 2 3 2
5 2 1 2 2 3 4
3 1 2 2 2 3 3
4 2 2 2 1 2 3
3 2 2 2 1 2 4
5 2 2 2 3 4 1
1 1 1 2 1 2 1
1 1 1 2 1 2 3
2 2 1 2 2 3 1
3 2 1 2 1 2 2
3 2 1 2 1 2 2

class4-events
x1 x2 x3 x4 x5 x6 x7
5 2 2 3 1 1 2
5 2 1 3 1 1 4
5 2 2 3 1 1 4
5 2 1 3 1 1 2
5 2 1 1 1 1 1
5 2 2 1 1 1 1
5 2 2 3 1 1 3
5 2 1 3 1 1 3

The output file, windbracing.aqout, from AQ15c is the following:

parameters
run mode ambig trim wts maxstar echo criteria verbose
1 ic empty spec cpx 10 pcdv default 3

default-criteria
criterion tolerance
1 maxnew 0.00
2 minsel 0.00

domaintypes
type size cost name
nom 2 1.00 l2
nom 3 1.00 l3
nom 4 1.00 l4
nom 5 1.00 l5

variables
type size cost name
1 nom 5 1.00 x1.l5
2 nom 2 1.00 x2.l2
3 nom 2 1.00 x3.l2

```
  4   nom      3    1.00          x4.l3
  5   nom      3    1.00          x5.l3
  6   nom      4    1.00          x6.l4
  7   nom      4    1.00          x7.l4

 class1-outhypo
  #   cpx
  1   [x1=1] [x4=1,3] [x5=1,2] [x6=1] [x7=1,2,3]   (t:24, u:24)
  2   [x1=3] [x2=1] [x4=1,3] [x5=1] [x6=1] [x7=1,2,3]   (t:6, u:6)
  3   [x1=1] [x2=2] [x4=1,3] [x5=3] [x6=1] [x7=1,2,3]   (t:6, u:6)
  4   [x1=1] [x2=2] [x4=3] [x5=1,2] [x6=1] [x7=4]   (t:4, u:4)
  5   [x1=5] [x2=2] [x4=3] [x5=2] [x6=1] [x7=2,3]   (t:3, u:3)
  6   [x1=2] [x2=1] [x4=3] [x5=2] [x6=1] [x7=4]   (t:2, u:2)
  7   [x1=2] [x2=2] [x4=3] [x5=1] [x6=1] [x7=2]   (t:2, u:2)
  8   [x1=1] [x2=1] [x4.x5=3] [x6=1] [x7=2]   (t:2, u:2)
  9   [x1=4] [x2=2] [x3=1] [x4.x5=3] [x6=1] [x7=4]   (t:1, u:1)
 10   [x1=4] [x2.x3=2] [x4=1] [x5=2] [x6.x7=1]   (t:1, u:1)
 11   [x1=3] [x2=1] [x3=2] [x4=1] [x5=2] [x6.x7=1]   (t:1, u:1)
 12   [x1=3] [x2.x3=1] [x4=3] [x5=1] [x6=1] [x7=4]   (t:1, u:1)
     (Ex:53, Amb:0, St:53, Su:53, Su/St*100:100)

 class2-outhypo
  #   cpx
  1   [x1=2,3,4] [x4=3] [x5=2,3] [x6=1] [x7=2,3]   (t:40, u:8)
  2   [x1=2,3,4,5] [x4=1,3] [x5=3] [x6=1] [x7=1,2,3]   (t:36, u:12)
  3   [x1=3,4] [x2=2] [x4=3] [x5=1,2] [x6=1] [x7=2,3,4]   (t:23, u:6)
  4   [x1=2,3] [x2=2] [x4=1,3] [x5=1,2] [x6=1] [x7=1,3,4]   (t:23, u:6)
  5   [x1=1,2,3] [x2=1] [x4=1,3] [x5=3] [x6=1] [x7=1,3,4]   (t:18, u:8)
  6   [x1=2,4] [x4=1,3] [x5=1] [x6=1] [x7=1,3,4]   (t:18, u:6)
  7   [x1=3,5] [x4=3] [x5=2,3] [x6=1] [x7=4]   (t:12, u:7)
  8   [x1=1] [x2=2] [x4=2] [x5=1] [x6=2]   (t:8, u:8)
  9   [x1=2] [x2=1] [x4=1,3] [x5=1,2] [x6=1] [x7=1,2]   (t:8, u:3)
 10   [x1=2,3,4,5] [x3=1] [x4=1,3] [x5=2] [x6.x7=1]   (t:7, u:4)
 11   [x1=3,4] [x3=2] [x4=3] [x5=1,3] [x6=1] [x7=4]   (t:5, u:2)
 12   [x1=1] [x2=1] [x4=3] [x5=1,2] [x6=1] [x7=4]   (t:4, u:4)
 13   [x1=5] [x2=2] [x4.x5=2] [x6=3] [x7=1]   (t:2, u:2)
 14   [x1=1] [x2=2] [x4.x5=3] [x6=1] [x7=4]   (t:2, u:2)
     (Ex:137, Amb:0, St:206, Su:78, Su/St*100:37)

 class3-outhypo
  #   cpx
  1   [x1=2,3,4,5] [x4.x5=1,2] [x6=2,3] [x7=2,3,4]   (t:72, u:43)
  2   [x1=1,2,3,4] [x4=2] [x5=2,3] [x6=2,3,4]   (t:69, u:30)
  3   [x1=2,3,4,5] [x4=2] [x5=1,3] [x6=2,4] [x7=1]   (t:24, u:14)
  4   [x1=1] [x2=1] [x4=2] [x5=1] [x6=2]   (t:8, u:8)
  5   [x1=2] [x2=2] [x4.x5=3] [x6=1] [x7=4]   (t:2, u:2)
  6   [x1=5] [x2.x3=2] [x4=1] [x5=2] [x6.x7=1]   (t:1, u:1)
     (Ex:137, Amb:0, St:176, Su:98, Su/St*100:55)

 class4-outhypo
  #   cpx
  1   [x1=5] [x2=2] [x4=1,3] [x5=1] [x6=1]   (t:8, u:8)
     (Ex:8, Amb:0, St:8, Su:8, Su/St*100:100)

This learning used:
 System time:      2.450 seconds
 User time:        2.00  seconds
```

REFERENCES

Arciszewski, T., Bloedorn, E., Michalski, R.S., Mustafa, M. and Wnek, J., "Machine Learning of Design Rules: Methodology and Case Study," *ASCE Journal of Computing in Civil Engineering*, Vol. 8, No. 3, pp. 286-308, July 1994.

Bloedorn, E.E., "Multistrategy Constructive Induction," Ph.D. Dissertation, School of Information Technology and Engineering, Reports of Machine Learning and Inference Laboratory, MLI 96-7, George Mason University, Fairfax, VA, 1996.

Kaufman, K. and Michalski, R.S., "A Method for Reasoning with Structured and Continuous Attributes in the INLEN-2 Knowledge Discovery System," Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, OR, August, 1996, pp. 232-237.

Michalski, R. S., "A Planar Geometrical Model for Representing Multi-Dimensional Discrete Spaces and Multiple-Valued Logic Functions," Report No. 897, Department of Computer Science, University of Illinois, Urbana, January 1978.

Michalski, R. S., "A Theory and Methodology of Inductive Learning" (Modified version of 1983-3), Artificial Intelligence, 20, pp. 111-161, 1983.

Michalski, R.S., Kerschberg, L., Kaufman, K.A. and Ribeiro, J.S., "Mining For Knowledge in Databases: The INLEN Architecture, Initial Implementation and First Results," Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies, Vol. 1, No. 1, pp. 85-113, August 1992.

Quinlan, J.R., “C4.5: Programs for Machine Learning”, Morgan Kaufmann, Los Altos, California, 1993.

Szczepanik, W., Arciszewski, T. and Wnek, J., "Empirical Performance Comparison of Two Symbolic Learning Systems Based On Selective And Constructive Induction," Proceedings of the IJCAI-95 Workshop on Machine Learning in Engineering, Montreal, Canada, August, 1995.

Thrun, S.B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K.A., Dzeroski, S., Fahlman, S.E., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R.S., Mitchell, T., Pachowicz, P., Vafaie, H., Van de Velde, W., Wenzel, W., Wnek, J., Sarma, J., Wahab, A. and Michalski, R.S., "Comparing Learning Paradigms via Diagrammatic Visualization: A Case Study in Concept Learning Using Symbolic, Neural Net and Genetic Algorithm Methods," *Proceedings of the 5th International Symposium on Methodologies for Intelligent Systems - ISMIS’90*, Knoxville, TN, pp. 428-437, October 1990.

Wnek, J. and Zhang, J., "The Monk's problems: A Performance Comparison of Different Learning Algorithms," *Computer Science Reports*, CMU-CS-91-197, Carnegie Mellon University (Revised version), Pittsburgh, PA, December 1991.

Wnek, J., "DIAV 2.0 User Manual: Specification and Guide through the Diagrammatic Visualization System," Reports of the Machine Learning and Inference Laboratory, MLI 95-5, George Mason University, Fairfax, VA, 1995.

Wnek, J., & Kaufman, K., & Bloedorn, E., & Michalski, R.S., “Inductive learning system AQ15c: the method and user's guide”, Reports of the Machine Learning and Inference Laboratory, MLI 95-4, George Mason University, Fairfax, VA., 1995.