Knowledge Visualization Using Optimized General Logic Diagrams

Bartłomiej Śnieżyński[1], Robert Szymacha[1], and Ryszard S. Michalski[2,3]

[1] Institute of Computer Science, AGH University of Science and Technology, Kraków, Poland
[2] Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, USA
[3] Institute of Computer Science, Polish Academy of Science
e-mail: sniezyn@agh.edu.pl, robert@bystrze.org, michalski@gmu.edu

Abstract. Knowledge Visualizer (KV) uses a General Logic Diagram (GLD) to display examples and/or various forms of knowledge learned from them in a planar model of a multi-dimensional discrete space. Knowledge can be in different forms, for example, decision rules, decision trees, logical expressions, clusters, classifiers, and neural nets with discrete input variables. KV is implemented as a module of the inductive database system VINLEN, which integrates a conventional database system with a range of inductive inference and data mining capabilities. This paper describes briefly the KV module and then focuses on the problem of arranging attributes that span the diagram in a way that leads to the most readable rule visualization in the diagram. This problem has been solved by applying a simulated annealing.

Keywords: knowledge visualization, GLD diagrams, diagram optimization, machine learning.

1 Introduction

Data and knowledge visualization tools are components of many commercial data mining tools, such as Microsoft SQL Server 2000 Analysis Services, IBM DB2 Intelligent Miner, and Oracle Data Miner. They are also present in noncommercial software, e.g. Weka [11] and YALE [10]. The authors are not aware, however, of any tool with capabilities of Knowledge Visualizer (KV) described in this paper. KV is a unique knowledge visualization system that uses a General Logic Diagram (GLD) for representing examples and knowledge derived from them.

GLD, introduced by Michalski [6], is a planar model of a multidimensional space spanned over discrete variables (attributes). Each variable partitions the diagram into a set of disjoint areas corresponding to the values of the variable. The lines separating these areas for a single attribute are called *axes* of this attribute. An intersection of the areas corresponding to single values of each variable constitutes a cell of the diagram. Every cell of the diagram thus represents a unique combination of attribute values.

Decision rules and decision trees are represented by regular configurations of cells. Logical operations AND,OR, and NOT correspond to intersection, union and complement, respectively, of the groups of cells representing arguments of the operations (Fig. 2).

Depending on the way the attributes are assigned to the axes of the diagram, knowledge representation in a diagram may be less or more readable and visually attractive. Therefore, a problem arises as to how to optimize the attribute assignment for representing a given form of knowledge. In this paper, we will be concerned with the optimization of the diagram for representing attrbutional rules learned by AQ-type learning program [8].

The KV program has been implemented as a module of inductive database system VINLEN [3] that aims at integrating conventional databases and a wide range of inductive inference capabilities.

The following sections review the concept of GLD, describe the KV program for visualizing examples and/or decision rules using GLD, and present a method for solving the above-mentioned attribute assignment problem. The working of KV is illustrated by an example.

2 General Logic Diagrams

General Logic Diagrams were developed by Michalski and applied in many different areas, such as the design and optimization of switching circuits, the optimization of logic functions, conversion of decision tables into decision trees, and, most widely, for visualizing concept examples and decision rules induced from them [7]. A GLD can be viewed as a multivalued extension of the Marquand's binary diagram [5] with some additional properties. Specifically, the diagram is structured by drawing axes of different variables with different thickness. Such a structuring facilitates the readability of knowledge represented in the diagram even when it is spanned over a relatively large number of attributes.

The procedure for creating a GLD (Generalized Logic Diagram) for representing a space $E(x_1, x_2, \ldots, x_n)$ spanned over n discrete attributes $x_1, x_2, \ldots, x_n$, with domains $D_1, D_2, \ldots, D_n$, is as follows:

1. Divide attributes $x_1, x_2, \ldots, x_n$ into two subsets: *Row-att* and *Coll-att*.
2. Order attributes in these subsets.
3. Order values in the domains of nominal attributes.
4. Draw a rectangle, and divide it into the numer of rows equal the number of values of the first attribute in *Row-att*. Draw the separating lines with the greatest chosen thickness. These lines constitute the axes of the first attribute. Assign to rows values of the first attribute.
5. Divide each row into the number of subrows equal the number of values of the second attribute in *Row-att*. Separating lines, which are axes of the second attribute, are drawn with the lower thickness than the axes

of the previous (first) attribute. Continue such process for the remaining attributes in *Row-att*.

6. Repeat the same process for attributes in *Coll-att* by dividing the rectangle into columns.

Individual cells in the so-created diagram correspond to single events, that is, to distinct combinations of attribute values. To represent a function that maps the event space (the cartesian product of attribute domains) into the domain of output variable, the cells of the diagram are marked by function values. Thus, every such function can represented in the diagram. Because standard decision rules, attributional rules, decision trees, neural nets, and other forms of knowledge involving discrete input attributes (or discrtized continuous attributes) represent functions over an event space, they all can be represented using GLDs.

Single decision or attributional rules correspond to regular configurations of cells in the diagram, and thus can be easily recognized visually even in diagrams with a relatively large number of attributes (their number depends on the sizes of attribute domains).

When the event space is large, the diagram can be spanned only over the attributes that are present in the rules to be represented in the diagram (whose number if usually significantly smaller than the total number of attributes).

To illustrate a GLD rule presentation, let us use the attributional rules generated by a learning program for very simple robot domain. The rules are listed in Fig. 1.

```
A robot is classified as friendly, if:
  - its body is round, or
  - its head is triangular.
A robot is classified as unfriendly, if:
  - the shape of its head is round or square,
    and its body is square or triangular,
    and it is holding a balloon or a sword, or
  - its body is triangular,
    and the antenna is green.
A robots is unclassified, if:
  - its head is pentagonal or square,
    and it is holding a flag.
```

Fig. 1. Rules generated using AQ21 program for simple robot domain

These rules are visualized using GLD that is presented in Fig. 2. The following shortcuts are used in the diagram:

- attribute names: holding=$x0$, head=$x1$, antenna=$x2$, body=$x3$;

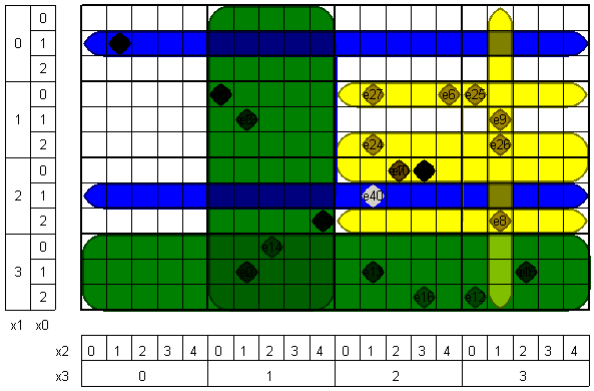


Fig. 2. GLD for rules generated from robots data

- holding values: balloon=0, flag=1, sword=2;
- body and head values: pentagon=0, round=1, square=2, triangle=3;
- antenna values: blue=0, green=1, red=2, white=3, yellow=4.

As we can see, *Coll-att*= $\{x3, x2\}$, *Row-att*= $\{x1, x0\}$. Big rectangles in the middle and on the bottom represent rules for class "good". Two long, horizontal rectangles represent rule for class "do_not_know". Rectangles on the right represent rules for class "bad". Circles represent events and their identifiers used to generate rules. They are darker if more then one event are printed is the same cell.

3 Current capabilities of the KV module

The KV program was implemented as a module of the VINLEN system. Input data (event space definition, examples, and rules) are taken from the VINLEN database. The module has following capabilities:

1. It can automatically draw a GLD for the given attributes.
2. The assignment of axes to attributes can be done by the user or automatically by the program.
3. It can represent training and testing examples for a given learning problem.
4. User can add events by choosing cells on a diagram.
5. It can automatically visualize attributional rules learned by AQ-type learning program and supplied to KV via VINLEN. Rules are represented by collections of linked rounded rectangles. Rectangles corresponding to rules of the same class are given the same color.
6. A diagram can be enlarged (to see details) or decreased (to fit the computer screen).

7. A diagram can be printed or saved as a graphic file. The user can choose the way the diagram is printed: in black-and-white or in color.
8. The assignment of axes of attributes can be automatically optimized to improve the rule visualization. Because the number of possible assignments can be very large, automatic optimization is very useful. An optimization method is described in the next Section.

4 Diagram optimization algorithm

To optimize the assignment of axes to attributes two features of the diagram are measured:

1. $C(d)$, the number of compact regions in the diagram, d, that represent given set of attributional rules.
2. $S(d)$, the closeness of the ratio of the height to width of the diagram to the golden ratio, $S(d) = |width(d)/height(d) - 1.62|$.

The following diagram cost function aggregating these features is used:

$$f(d) = C(d) + pS(d), \tag{1}$$

where p is a user defined parameter (integer number) representing the importance of shape.

In the current implementation, simulated annealing algorithm is used to minimize f. This is done in a similar way as in graph optimization [2]. Details of the algorithm can be found in [4]. Generally, it works as follows. Initial partitioning and ordering is randomized. Next, the algorithm makes random changes in partitioning and orderings. Probability to switch to the state with worse diagram (higher f value) depends on a "temperature," which is initially high, but reduced during simulation by multiplying by user-defined dT factor. After user-defined number of steps, the best diagram is presented to the user.

5 Experiments

The experiments were designed to test the performance of the KV module. In the experiment described here, the UCI's Mushroom dataset was used [1]. Two sets of rules were generated: RS1 using AQ21 program in strong patterns mode[1] for "cap-surface" target attribute, (see Fig. 3), and RS2 using C4.5 algorithm for "classes" target attribute.

Diagrams were optimized for displaying RS1 and RS2. Optimization was performed 10 times with the following parameters: 100 steps, initial temperature $T = 1$, temperature factor dT=0.9, and shape importance p=100. Diagrams are presented in Fig. 4, and 5. Axes descriptions are omitted because cells are very small. Average execution time on a Pentium 4 2.4GHz machine was 143 seconds for RS1 and 81 seconds for RS2.

[1] Rules generated using AQ21 in the strong pattern mode are simple but approximate.

```
cap-surface is FIBROUS, if:
  - gill-color is BLACK,BROWN,CHOCOLATE,GRAY,PINK,PURPLE;
  - classes is EDIBLE;
cap-surface is GROOVES, if:
  - stalk_surface_above_ring is SMOOTH;
cap-surface is SCALY, if:
  - cap-color is not WHITE;
cap-surface is SMOOTH, if:
  - cap-shape is BELL,KNOBBED;
```

Fig. 3. RS1 rules generated using AQ21 program, with "cap-surface" target attribute and strong patterns mode

6 Conclusion and Further Research

The KV module allows one to represent graphically rules and data from a VINLEN knowledge system. The visualization method uses the General Logic Diagram. KV allows one to check how rules cover given training examples, and to analyze the relationship between the rules and the examples.

The integration of the KV module in the VINLEN system facilitates experimental investigation of the rule learning process. The diagram optimization feature improves the readability of the rules in the diagram.

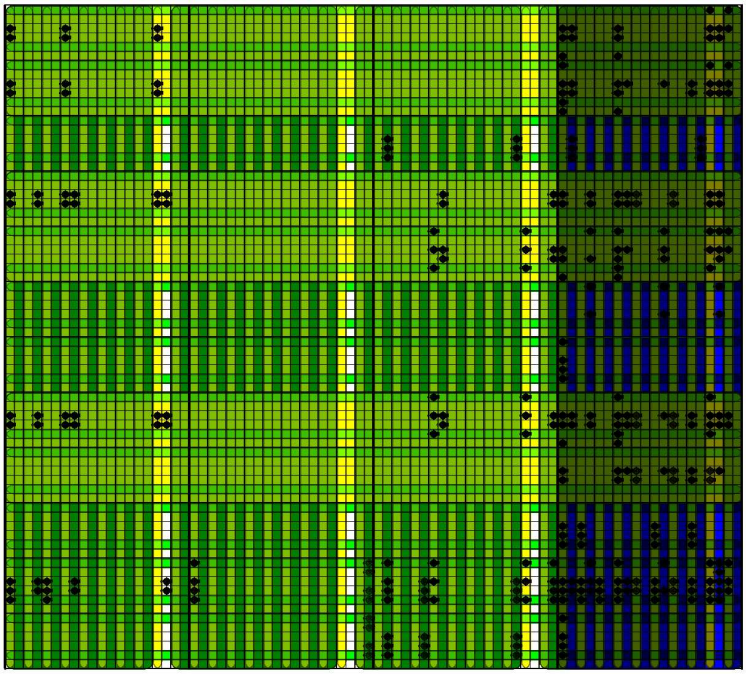
Further research will concern adding more features to the KV module, in particular, the ability to:

- display results of constructive induction;
- display results of abstraction and concretion operation on the representation space in the case of structured input attributes;
- place images in the cells, which will enable KV to visualize more complex knowledge;
- represent decision trees and results of clustering algorithms for unsupervised learning;

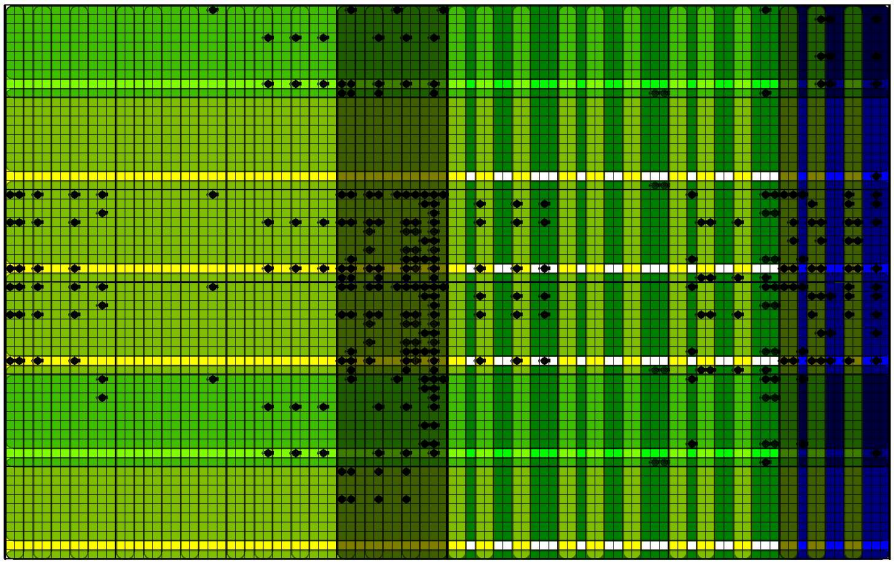
It is also planned to improve the current diagram optimization algorithm, for example, by applying the Learnable Evolution Model [9]. An associated problem is to develop and test other cost functions that take into consideration other diagram features. Another important problem is how to represent large representation spaces for which diagrams are too complex. A method of automatic generation of subsets of attributes to span different diagrams (views) needs to be developed.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
2. R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.
3. K. A. Kaufman and R. S. Michalski. The development of the inductive database system VINLEN: A review of current research. In M. Kłopotek et al., editor, *Intelligent Information Processing and Web Mining*, Advances in Soft Computing, pages 393–398. Springer, 2003.
4. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by genetic annealing. *Science*, (220):671–680, 1983.
5. A. Marquand. On logical diagrams for n terms. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, XII(75):266–270, 1881.
6. R. S. Michalski. Graphical minimization of normal expressions of logic functions using tables of the veitch-karnaugh type. *Journal of the Institute of Automatic Control*, (52), 1978.
7. R. S. Michalski. A planar geometrical model for representing multi-dimensional discrete spaces and multiple-valued logic functions. Technical Report 897, Department of Computer Science, University of Illinois, Urbana, 1978.
8. R. S. Michalski. *Attributional Calculus: A Logic and Representation Language for Natural Induction.* Reports of the Machine Learning and Inference Laboratory, MLI 04-2. George Mason University, 2004.
9. R. S. Michalski, K. A. Kaufman, and G. Cervone. Speeding up evolution through learning: LEM. In M. Kłopotek et al., editor, *Proceedings of the Ninth International Symposium on Intelligent Information Systems*, 2000.
10. I. Mierswa, R. Klinkberg, S. Fischer, and O. Ritthoff. A Flexible Platform for Knowledge Discovery Experiments: YALE – Yet Another Learning Environment. In *LLWA 03 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivitt*, 2003.
11. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, 1999.

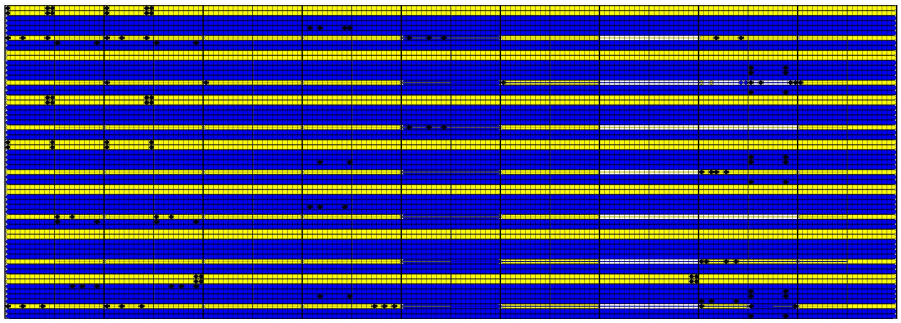


(a)

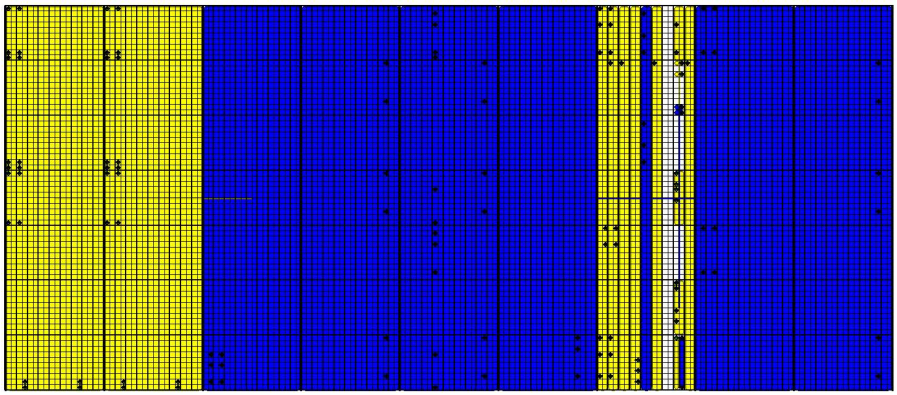


(b)

Fig. 4. GLD Diagrams for rules RS1: initial (a) and optimized (b)



(a)



(b)

Fig. 5. Diagrams for rules RS2: initial (a) and optimized (b)