# Visual Data Mining Using Monotone Boolean Functions.

F. Delizy *
Computer Science Department
Central Washington University
Ellensburg, WA, U.S.A.

B. Kovalerchuk
Computer Science Department
Central Washington University
Ellensburg, WA, U.S.A.

**Abstract** *This paper describes a new technique for extracting patterns and relations visually from multidimensional binary data using monotone Boolean functions. Visual Data Mining (VDM) has shown benefits in many areas when used with numerical data, but that technique is less beneficial for binary data. This problem is especially challenging in medical applications tracked with binary symptoms. The proposed method relies on monotone structural relations between Boolean vectors in the n-dimensional binary cube, $E^n$, and visualizes them in 2-D or 3-D as chains of Boolean vectors. Actual Boolean vectors are laid out on this chain structure. Currently the system supports two visual forms: the multiple disk form (MDF) and the "Yin/Yang" form (YYF). In the MDF, every vector has a fixed horizontal and vertical position. In the YYF, only the vertical position is fixed. The method is illustrated with an example from a breast cancer diagnosis based on mammographic X-ray images.*

*Keywords:* Visual Data Mining, explicit data structure, Boolean data, Monotone Boolean Function, Hansel Chains, Binary Hypercube

## 1 Introduction

The goal of visual data mining (VDM), as detailed in [1], is to help a user to get a feeling for the data, to detect interesting knowledge, and to gain a deep visual understanding of the data set. The traditional table of records format is convenient for presenting records, but it does not give a feel for the overall character of the data set. Visualization of such tables is difficult because, in contrast with scientific data visualization, data visualization lacks inherent dimensions that can be naturally mapped to three spatial and one temporal dimension [4]. Simple presentation graphics such as bar charts are intuitive and easy-to-use, but they show only highly aggregated data and very simple patterns [7].

VDM methods have shown benefits in many areas when used with numerical data, but these methods do not address the specifics of binary data, where there is no much variability in visual representation of objects for each individual Boolean attribute. The purpose of this paper is to develop a technique for extracting patterns and relations visually from *multidimensional binary data* using the technique of monotone of Boolean functions. We start from the analysis of currently available methods of data visualization.

A *glyph* is a 2-D or 3-D object (icon, cube, or more complex *"Lego-type"* object). Glyph or iconic, visualization is an attempt to encode multidimensional data within the parameters of the icons, such as the shape, color, transparency, orientation [2, 12, 13]. Typically, glyphs can visualize up to *nine attributes* (three positions $x$, $y$, and $z$; three size dimensions; color; opacity; and shape). Texture can add more dimensions. Shapes of the glyphs are studied in [14]. They concluded that with large super-ellipses, about 22 separate shapes can be distinguished on the average. An overview of

*Exchange student from the Computer Engineering Department of University of Technology of Belfort-Montbéliard (France).

multivariate glyphs is presented in [15]. This overview includes a taxonomy of glyph placement strategies and guidelines for developing such visualization. Some glyph methods use data dimensions as positional attributes to place glyphs; other methods place glyphs using *implicit or explicit structure* within the data set. From our viewpoint, the placement based on the use of data structure is a promising approach. We argue that placements of glyphs on a data structure is a way to increase the data dimensions that can be visualized. We call this the GPDS approach (glyph placement on the data structure). It is important to notice that in this approach, some attributes are implicitly encoded in the data structure and some explicitly in a glyph. Thus, if the structure carries ten attributes and a glyph carries nine attributes, we can encode nineteen attributes total. The number of glyphs that can be visualized is relatively limited because of possible glyph overlap and blockage.

Alternative techniques such as Generalized Spiral and Pixel Bar Chart are developed in [7]. These techniques work with large data sets without overlapping, but only with a few attributes (from a *single attribute* to *4-6 attributes*). Another set of visualization methods, Scatter, Splat, Map, Tree, and Evidence Visualizer, implemented in MineSet (Silicon Graphics) permits up to *eight dimensions* to be shown on the same plot by using color, size, and animation of different objects [11]. Parallel coordinate visualization [6] can work with ten or more attributes, but suffers from record overlapping and, thus, is limited to tasks with well distinguished cluster records. In parallel coordinates, each vertical axis corresponds to a data attribute and a polyline corresponds to a record. Technically, the number of dimensions is limited only by the screen resolution, and is typically too overwhelming to permit any real understanding of the data [3].

Serious limitations in traditional visualization techniques are summarized in [11]:

- *Subjectivity of visual representation* can cause different conclusions looking to the same data.
- *Poor scalability* to represent hundreds of attributes for visual data analysis.
- *Human inability* to perceive more than 6-8 dimensions at the same graph.
- *Low speed* of manual interactive examination of the multi-dimensional, multi-color charts.

Automating the process of human perception in connection with the data visualization task is suggested in [11] as a way to address listed problems.

We are interested developing a technique that can work with ten or more Boolean attributes. Many data mining problems can be encoded using Boolean vectors, where each record is a set of binary values $\{0, 1\}$ and each record belongs to one of two classes (categories) that are also encoded as 0 and 1. For instance, a patient can be represented as a Boolean vector of symptoms and an indication of the diagnostic class (e.g., benign or malignant tumor) [9, 10].

For $n$-dimensional Boolean attributes, traditional glyph-based visualizations offer a limited scope. Attributes of a Boolean vector can be encoded in glyph lengths, widths, heights, and other parameters. There are only two values for the length, width, and other parameters for each Boolean vector. Thus, there is not much variability in visual representation of objects. When plotted as nodes in a 3-D binary cube, many objects will not be visually separated.

The proposed method does not follow the traditional glyph approach that would put $n$-dimensional Boolean vectors ($n > 3$) into 3-D space, making them barely distinguishable. The method relies on monotone structural relations between Boolean vectors in the $n$-dimensional binary cube, $E^n$, and visualizes them in 2-D or 3-D as chains of Boolean vectors. Actual Boolean vectors are laid on this *chain structure*. Every n-dimensional Boolean data set can be encoded as a Boolean function in DNF or CNF. Thus, visualization of a

Boolean data set is equivalent to visualization of a Boolean function. Next, every Boolean function can be decomposed into a set of monotone Boolean functions [8].

The monotone *structure* is important for the data mining tasks, because most data mining methods are based on the hypothesis of local compactness: if two objects have similar features, then they belong to the same class.

Now we will describe the structure used to allocate Boolean vectors. In rendering, Boolean vectors are ordered vertically by their Boolean norm (the number of 1 values in the vector) and a partial order on Boolean vectors. This number is referred to as a *level*. The *partial order* is defined as follows: vector $a = (a_1, a_2, \ldots, a_n)$ is greater or equal to vector $b = (b_1, b_2, \ldots, b_n)$ if $\forall i = 1, \ldots, n, a_i \geq b_i$. We will use the notation $a \geq b$. A set of vectors $v_1, v_2, \ldots, v_n$ is called a *chain* if $v_1 \geq v_2 \geq \ldots \geq v_n$.

Currently, the system supports two visual forms: *Multiple Disk* Form (MDF) and *"Yin/Yang"* Form (YYF). In the MDF, every vector has a fixed horizontal and vertical position. In the YYF, only the vertical position is fixed for each vector.

The procedure $P_1$ for MDF places Boolean vectors in the natural numerical order in each level, which is called a disk. In this way, every Boolean function has exactly the same layout in the disks, allowing multiple functions to be compared at the same time. The second procedure $P_2$ for MDF permits direct comparison of Boolean data sets and functions and is based on the decomposition of the binary cube, $E^n$, into special chains of vectors called Hansel chains [5, 8]. The Hansel chains are computed and then aligned vertically.

The goal of our VDM is to show patterns in a simple visual form. If points of one class (e.g., benign tumor) are located in one part of the visual space, points of another class (e.g., malignant tumor) are located in the other part and the border between classes is simple, then the goal of VDM is reached. Monotone Boolean functions permit the production of a border that helps interpret data. In this way, the border between classes of vectors can be revealed visually. Procedure $P_1$ produces a border that can be very complex and, thus, not easy to interpret. However, $P_1$ produces the same border scheme for different Boolean functions, a condition that is necessary for direct comparison. In general, direct comparisons are not always easy. Moreover, often the goal is finding the differences between functions more than similarities. Hence, a third procedure, $P_3$, for MDF has been introduced. This procedure tries to move all Hansel chains to the center of the disk. It is based on: the level of the first 1 value in each chain for a given Boolean function, and the requirement that the disk architecture should be preserved. In this way, two different functions will be visualized distinctly.

The borders produced by $P_3$ can still be complex and thus difficult to interpret visually. Therefore, the YYF structure is introduced. In the YYF, the movement of all chains is based on only the level of the first 1 in each chain. The set of chains is sorted from left to right according to this level and providing a clear, simple border between the two classes of Boolean vectors.

## 2 Definitions and Data Structures

Before entering upon the subject more formally, it is important to define the terms that will be used below.

A *Boolean vector* is an ordered set of Boolean values 0 and 1. It can be represented as a sequence of number such as 0101110010, which would be a 10 value Boolean vector. We can assign a set of properties to a Boolean vector such as its size and its norm.

The *level of a Boolean vector*, also referred to as the *Boolean norm*, is the sum of the components of the Boolean vector. We use these norms for splitting the set of vectors into $n+1$ levels. For instance, the level of the vector 0000000000 would be 0, whereas the level of the vector 1111111111 would be 10.

Boolean vectors can be represented as a cube

(or an *hypercube* if $n > 3$). For instance, if $n = 1$, the cube is formed by the two elements $\{0, 1\}$. For $n = 2$, the cube is a square formed by the elements $\{00, 01, 10, 11\}$. We need to have a way to browse a hypercube without overlapping. This way is provided by the *Hansel chains* [5, 8].

We build Hansel chains recursively. The Hansel chain for the size of 0 is the trivial segment $(0, 1)$. To obtain the Hansel chains for the level 2, we first duplicate the Hansel chains of the level 1 by adding a 0 or a 1 prefix. Then the two following set $E_{min} = (00, 01)$ and $E_{max} = (10, 11)$ are genereted. We then cut the maximum element of $E_{max}$ and add it to $E_{min}$. Thus, the Hansel chains for the size 2 which are $\{(00, 01, 11); (10)\}$. By repeating those operations of *duplicating* and *cutting*, we will be able to build the Hansel chains for any size vectors.

A Boolean function $f$ is *monotone* if given two Boolean vectors $x = (x_1, x_2, x_3, \ldots, x_s)$ and $y = (y_1, y_2, y_3, \ldots, x_n)$, if $x$ precedes $y$, that is $\forall i \in \{1..n\}, y_i \geq x_i$, the following relation is true: $f(y) \geq f(x)$.

Such functions divide the set of Boolean vectors into two classes: vector assigned to value 0 and vectors assigned to value 1, thus forming a border between the two classes. This research intends to visualize the border between the two classes in a clear way. Each vector will be first placed in the view and then drawn as colored bar: white for the 0 class, black for the 1 class

Next, consider the structure of an MDF. All vectors are first ordered vertically in regard to their level (recall there are $n + 1$ levels). The number of vectors on each level $l$ is obtained by the formula $C_n^l = \frac{n!}{l! \cdot (n-l)!}$. For instance if $n = 10$ there is 1 element on the level 0, and 10 elements on the level 1 . Note the maximum number of elements is reached on the level $\frac{n}{2}$ with 252 elements for $n = 10$ ($C_{10}^5 = 252$). The number of elements increases from the level 0 to the level $\frac{n}{2}$ and decreases to the level $n$. All vectors are then centered in the view, thus giving the MDF a *"symmetrical Hanoi tower"* shape with each level of vector forming a so-called *disk* (see Figure 1).

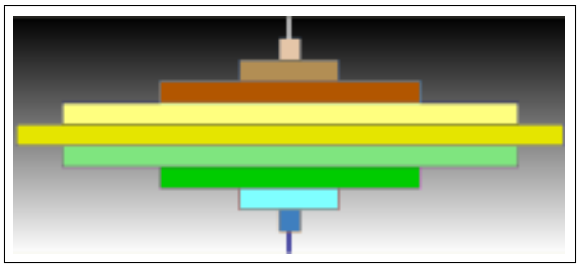

Figure 1: *MDF structure without data.*

The YYF is derived from the MDF. In our attempt to visualize the borders between the two classes of elements, we again moved the data out of the MDF structure. In the YYF vectors are ordered vertically in the same way as in the MDF but they are not centered anymore. All vectors are moved with regard to the data in order to visualize to the border between the two classes. In this way, a clear border will appear, the 1 class being up to the 0 class thus giving the YYF a *"Yin/Yang"-like* shape responsible for its name.

## 3 Procedures

The Boolean vectors are placed vertically with every structure in regard to their level. Then vectors are placed horizontally inside each structure using specialized procedures. We developed three procedures working with the MDF and one procedure with the YYF.

The procedure $P_1$ relative to the MDF places the vectors relative to their natural order. Each binary vector is converted to its decimal equivalent. For instance, the decimal equivalent of the vector 0000000010 would be 2. Each vector is then placed based on this value with value 0 being on the right side. The advantage of this procedure is to allow the user to compare more than one function at a time. For every function, and without regards to the data, the placement of every vector will be exactly the same, allowing direct comparison.

$P_1$ does not really visualize any border or structure of the Boolean function; therefore, we created the second procedure $P_2$. Procedure $P_2$

relative to MDF moves the vectors with regard to Hansel chains. The Hansel chains for vectors of size $n$ are first computed, then every vector belonging to the chain will be moved in order to align the Hansel Chain vertically. Hansel chains have different lengths, from 1 element to $n$ elements. To keep the integrity of the MDF structure, we have to place these chains so that no elements fall out of the disks. Hence, the longest chain will be placed on the center disk and the others chains will be placed alternatively right and left from the first chain. Moreover, procedure $P_2$ will assign the same position to vectors without regards to the data. This again allows, direct comparison to be done between different Boolean functions.

$P_2$ visualizes a certain level the structure of the Boolean function, but does not really visualize the border between the classes. Hence, we created the last procedure $P_3$. Procedure $P_3$ relative to MDF is derivative of $P_2$. After computing and placing the vectors using $P_2$, every Hansel chain will be given a value $l$ equal to the level of the first 1 value present within the chain. Next, every Hansel chain will be moved so that the chain with the highest $l$ value is located in the center so that the MDF structure is kept. Using this procedure, we are able to group the class within the MDF. Nevertheless to keep the MDF structure, the chains have to be placed with regards to their length. This introduces a possibly complex border between classes because of a possible *gap* between groups of vectors of the same class.

Because of the possibly complex border provided by $P_3$ we developed the new YYF structure as well as the procedure $P_4$ relative to this structure. The YYF does not keep the disk form. This structure allows filling the gaps between the groups. The procedure $P_4$ extends every Hansel chains created before placing them according to the same value calculated for procedure $P_3$.

The first step consists in extending the Hansel chain with elements in relation with the edges elements up and down. To extend up a chain, we try to find the first element belonging to the 1 class above the edge element. That is given the edge element $x$ we look for the element $y$ verifying $y \geq x$, if no such $y$ is found on the level just above the $x$ level, we then add an element $z$ from the 0 class so that $z \geq x$ and so that the path to the first element $y \geq z$ of the 1 class is minimized. We repeat these step until we find an element $y$ from the 1 class and add it to the chain. To expand down the chain, we apply the same steps reversing the relation $y \geq x$ and swapping the classes 0 and 1: we try to find an element $y$ from the 0 class verifying $x \geq y$ . If no $y$ is found on the level just bellow the $x$ level, we add a $z$ from the 1 class verifying $x \geq z$ and minimizing the path to the first $y$. Using this procedure, we duplicate some of the vectors which could then be displayed more than once. This is justified because we keep the relation between same vectors.

Once the chains are expanded, a value $l$ will be assigned to each chain, just as in the procedure $P_3$. Then, the chains will be sorted in regard to this value. This approach will visualize a simple border between 0 and 1 classes.

## 4 Experiments

In this section we illustrate the methods for visualizing a data set that can be described by a simple Boolean function $f(x_1, x_2, \ldots, x_{10}) = x_1$. First, Figure 2 shows the system allowing all elements at a fixed place using MDF and $P_1$.

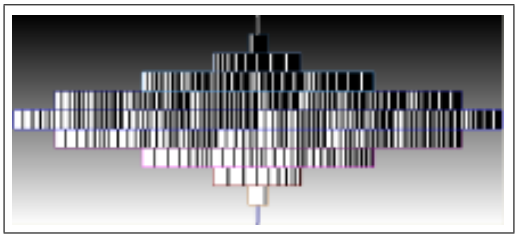

Figure 2: *MDF,* $P_1$ *with* $f(x_1, \ldots, x_{10}) = x_1$

Clearly, $P_1$ does not permit visualizing the structure itself. Nevertheless, because elements have a fixed place, this procedure permits comparison between multiple functions. Next, the procedure $P_2$ will unveil parts of the

structure of the Boolean function, see Figure 3.

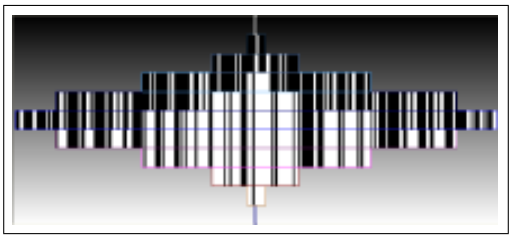

Figure 3: *MDF, $P_2$ with $f(x_1, \ldots, x_{10}) = x_1$*

$P_2$ still has a fixed place for each Boolean vector. Hence, it still permits the comparison between multiple functions. As Figure 3 shows, the border visualized can be very complex. In this case, we then need $P_3$ to more easily visualize the border between the two classes of elements, see Figure 4.

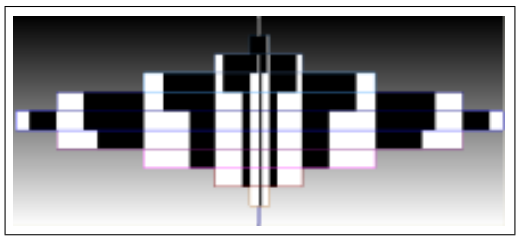

Figure 4: *MDF, $P_3$ with $f(x_1, \ldots, x_{10}) = x_1$*

Using $P_3$ makes the border obvious, but the border is still cut into pieces because of the use of the MDF itself. We can see the gaps between the black parts. However, each white element placed on the top (belonging to 0 class) actually expands to an element of the 1 class. Therefore, the border should be visualized as continuous rather than interrupted. This is the reason why we created the YYF structure. The YYF structure will show this continuous border.

## 5 Conclusion

VDM had shown benefits in many areas when used with numerical data. However, classical VDM methods do not address specifics of binary data. By not using the traditional glyph approach, that would put $n$-dimensional Boolean vectors ($n > 3$) into 3-D space, we tend to visualize the real patterns contained in Boolean data.

The first attribute to consider while ordering Boolean vectors is its norm. Using the Boolean norm of the vectors, we are able to split the data into $n+1$ groups ($n$ being the number of elements of vectors). Each group is assigned to a vertical position. Then, multiple methods can be used to assign the horizontal position.

We created two different structures to handle data: MDF and YYF. In every structure, horizontal position of vectors is then handled by a specific procedure. We created three procedures specific to the MDF structure and one for the YYF structure.

The first procedure $P_1$ is specific to MDF and orders vectors in regard to the natural order, converting the Boolean value into a numerical decimal value. This procedure does not visualize the real structure of data, but, permits direct comparison between multiple functions.

The second procedure $P_2$ is specific to MDF and orders vectors to visualize Hansel chains. This procedure visualizes the structure of the data itself. However, it does not really visualize relations among data but it still permits direct comparison to be made between several Boolean functions.

The third procedure $P_3$ is specific to MDF and orders the Hansel chains. This procedure unveils the border between the two classes of elements (0 and 1). In order to keep the MDF structure intact, the border is visualized as being interrupted and thus differences can be visualized between multiple Boolean functions. However, in monotone Boolean functions, vectors belonging to 0 class actually all expand to a vector of the 1 class. Hence, the border should be continuous.

The last procedure, $P_4$, is specific to YYF and then solves this final problem. This procedure visualize the real border that exists between the two classes of elements. This is done by expanding up and down the Hansel chains, thus duplicating some elements.

This new approach proved to be appropriate to handle discovery of patterns in binary data. By further developing these procedures and data structures, the new approach can be used in variety of applications.

## References


[1] C. Beilken, M. Spenke. Visual interactive data mining with InfoZoom. *The Medical Data Set Contribution to the "Discovery Challenge" at the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD '99, Sept 15-18, 1999, Prague, Czech Republic.

[2] D. Ebert, C. Shaw, A. Zwa, E. Miller, D.Roberts. Two-handed interactive sereoscopic visualization. *IEEE Visualization '96 Conference.*

[3] A. Goel. *Visualization in Problem Solving Environments.* Virginia Polytechnic Institute and State University, 1999.

[4] D. P. Groth, E. L. Robertson. Architectural support for database visualization, *Workshop on New Paradigms in Information Visualization and Manipulation*, 53-55,1998.

[5] G. Hansel. *Sur le nombre des fonctions Booléenes monotones de n variables.* C.R. Acad. Sci., Paris (in French), 262(20):1088–1090, 1966.

[6] A. Inselberg, B. Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. *Proceedings of IEEE Visualization '90*, 360–375, Los Alamitos, CA, 1990, IEEE Computer Society Press.

[7] D. A. Keim, Ming C. Hao, U. Dayal, Meichun Hsu. Pixel bar charts: a visualization technique for very large multi-attributes data sets. *Information Visualization*, March 2002, Vol. 1, N. 1:20–34.

[8] B. Kovalerchuk, E. Triantaphyllou, A.S. Despande, E. Vityaev. Interactive Learning of Monotone Boolean Functions. *Information Sciences* Vol. 94, issue 1-4:87–118, 1996.

[9] B. Kovalerchuk, E. Vityaev, J. Ruiz. Consistent knowledge discovery in medical diagnosis. *IEEE Engineering in Medicine and Biology, (Special issue on Data Mining and Knowledge Discovery)*, v. 19, n. 426–37, 2000.

[10] B. Kovalerchuk, E. Vityaev, J.F. Ruiz. Consistent and complete data and "expert" mining in medicine. *Medical Data Mining and Knowledge Discovery*, Springer, 2001:238–280.

[11] M. Last, A. Kandel. Automated perceptions in data mining, invited paper. *1999 IEEE International Fuzzy Systems Conference Proceedings*, Part I:190–197, Seoul, Korea, Aug 1999.

[12] F. J. Post, T. van Walsum, F.H. Post, D. Silver. Iconic techniques for feature visualization. *In Proceedings Visualization '95*, 288–295, 1995.

[13] W. Ribarsky, E. Ayers, J. Eble, S. Mukherja. Glyphmaker: creating customized visualizations of complex data. *IEEE Computer*, 27(7):57–64, Jul 1994.

[14] C. Shaw, Hall, J., Blahut, C., Ebert, D., Roberts. Using shape to visualize multivariate data. *CIKM'99 Workshop on New Paradigms in Information Visualization and Manipulation*, ACM Press, 1999.

[15] M. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization* 1:194–210, 2002.