Data visualization in RDBMS

Maria Camila Barioni[1], Elisângela Botelho[1], Christos Faloutsos[2],
Humberto Razente[1], Agma J. M. Traina[1], Caetano Traina Júnior[1]

[1] Department of Computer Science and Statistics
University of São Paulo at São Carlos - Brazil
Av. Trab. Sãocarlense, 400 - Centro - C. Postal 668, São Carlos, SP, 13560-970
{mcamila, ebotelho, hlr, agma, caetano}@icmc.usp.br
[2] Department of Computer Science
Carnegie Mellon University - USA
5000 Forbes Avenue, Pittsburgh, PA, 15213-3891
christos@cs.cmu.edu

Abstract

Completely automated data analysis techniques often fail to meet their requirements, due to their inability to exploit peripheral knowledge associated with the data. Human beings are very good at interpreting data represented in graphical format, and usually have the wisdom to recognize the associated knowledge. This paper addresses this dichotomy through a data visualization tool which displays, in a graphical manner, data stored in database relations, without requiring any native spatial data distribution, thus involving human beings in the main stream of KDD processes. It develops the conceptual framework which supports the data transformations enabling the visualization of data composed by attributes of many data types (numbers, dates and texts). This is achieved through the mapping of the attributes taken as multidimensional data into a 3-dimensional space, applying a user-defined distance function. Experimental evaluation shows that this tool is scalable to any database size, regarding number of tuples and attributes.

Key Words: Data Visualization, Dimensionality Reduction, Distance Functions

1. Introduction

Searching for a better position in the market, companies have been trying to obtain more benefits from the data they have collected during years of processing. The volume of data to be analyzed is huge, and in most cases no one knows where or how to start an information retrieval process. In other words, the effort and cost spent to collect and store data over the years, only to keep the information regarding products and clients, are no more enough to keep the companies in business. Moreover, all the data stored can be much more valuable if we can retrieve innovative information from that data, new facts not yet known and advantageous for the information's owner.

Recently, many works have been done in order to develop techniques and algorithms for Knowledge Discovery in Databases – KDD. According to [1], KDD is "the process of identifying structures that represent valid, novel, potentially useful, and ultimately understandable information, in a mass of data".

An important problem faced by the KDD process is the volume of data to be mined can be very large. Techniques and algorithms for data searching that are adequate for small datasets may not be adequate when the volume of data grows, considering both the number of attributes (dimensions) involved, and the number of items treated. When the number of attributes grows to more than a hundred, these algorithms become inappropriate, as time spent goes to weeks of processing. Nevertheless, the number of applications demanding this number of attributes is growing, such as systems of handling images and other complex data, as genetic structures or temporal sequences.

It is also important to note that the searching goal is not always clearly defined. For example, the search for some pattern in a relationship among several attributes, or the search for clusters, are frequent, but what attributes or what pattern would match this searching is not known beforehand [2]. This also brings to the problem of identifying how useful a discovered relation is, and how to interpret its results.

This work assumes that human beings are not "efficient" when interpreting large volumes of data in numerical or textual formats, specially when the data involve a high number of variables, or dimensions. However, they have very good perception of data presented as graphics. On the other hand, techniques fully automated for pattern detection, classification, clustering, among others, are frequently distorted by their inability to take advantage of other knowledge that humans easily recognize [3] [4].

This work explores the visualization of data stored in databases as a first step of analysis, where the user exploits his/her capacity of visual perception to interpret data. In general, the existing techniques of visualization represent statistic information about the data, such as the representation of aggregates (count, sum, minimum, maximum, average) through bar charts, pie charts, line charts, etc. In some cases where the data present an inherent spatial distribution, there are techniques of spatial visualization for it [5]. In this work, we present a technique which allows the creation of spatial visualizations of data, even when the data has no inherent spatial distribution. Moreover, this visualization does not rely on statistical summarizations.

This visualization approach turned out to be an expressive tool in data interpretation, mainly when statistical information may mask the existence of unexpected distributions on the data, outliers, etc. In this paper we show it is possible to build spatial representations from almost any mass of data, considering it is possible to calculate a grade of similarity between any pair of elements in it. In the process of visualization, categorical data can be used to classify the data, helping on the interpretation process of data classification.

Briefly, the technique adopted in this paper considers that data are represented in one relation of the database, and each attribute represents a numeric value (i.e., measures, monetary value, etc.), a date, or a short text (usually names). For each attribute is assigned a procedure to calculate differences from a pair of values, such as the absolute value of the difference between two numeric values, the number of days between two dates, etc. The individual differences are then combined to define the distance between each tuple of the relation, allowing one tuple of N attributes to be interpreted as a dot in a N-dimensional space. After that, an algorithm for dimensionality reduction is applied, which maps the original space to a 3-dimensional one, enabling the visualization of the relative distribution of the tuples of the relation. Categorical attributes can be used to color or visually modify the representation of each dot.

The rest of the paper is organized as follows. Section 2 discusses the concepts involved, and the main related work. Section 3 describes how to create distance functions for data stored in relational databases, taking into account different data types and domain properties, so it can be used to generate the data visualization. Section 4 presents the FastMapDB tool, created to demonstrate the concepts proposed here, and shows performance measurements of this tool for large datasets, and some results of applying it to visualize both well-known datasets of the literature and real application data. Section 5 presents the conclusions of this work.

2. Related works

Human beings have the capacity of absorbing and understanding information represented as graphics very quickly. Thus, when it is necessary to summarize huge amounts of numeric data, usually histograms, graphs or any other visual synthesis mechanisms are used. However, when the information to be presented is in some high dimensional space, or even in a non-dimensional space (e.g., words set), traditional visualization techniques are not suitable.

Existing visualization tools for data mining allow navigation through complex data structures, creating initial views, which are modified as the user navigates through data. In [6] it is presented a summary of the techniques used on visual data mining. Among them are the geometric projection, icon-based, pixel-oriented, hierarchical, and combinations of them.

According to [6] it's important to note that in order to obtain an effective data exploration it's necessary the use of interaction and distortion techniques in addition to visualization techniques. Interaction techniques let the user interact with the visualization realizing manipulations such as mapping, projection, filtering, zooming, etc. The distortion techniques help in the process of interactive exploration of the visualization, such as focusing regions while preserving a general view of the dataset.

The data visualization process also helps in clustering certification and identification of the dataset in analysis [7]. Among the most representative works about clustering is Clarans [8], which uses partitioning algorithms and BIRCH [9]. For spatial data domains, there are several studies and techniques already developed [10] [11]. For metric data domains, where only objects and distances among them are considered, [12] is one of the most divulged.

Data treatment in high dimensional data domains is expensive for any kind of processing that one may be interested. Thus, dimensionality reduction techniques have been developed. One of the landmarks among these techniques is the FastMap [13], which maps data in high or non-dimensional spaces to any dimensional space. The FastMap receives as input the dimensionality E of the target space, a set of objects and a distance function defined over the elements of this set, and produces a mapping of each element in a 3-dimensional space. In the mapping process, the FastMap algorithm uses the cosine law and the distances between the objects in the original space to generate the coordinates of each element in the target space preserving as much as possible the original distances between them.

3. Creating distance functions over relations

Besides being developed as a technique aiming dimensionality reduction of datasets, the FastMap algorithm can also be used to create spatial visualizations of the dataset. To this intent, if the dataset has a distance function defined, it is enough to set the target dimension E to 3, and display the resulting mapping using an interactive 3D visualization tool. However, to define a distance function to visualize data stored in a database relation, some considerations must be taken. First of all, although the attributes in each tuple of the relation describe one object in the real world, they relate to different measurements and concepts, thus defining different spaces where the attributes are correlated. Therefore, the distance function must be defined in such a way as to integrate these diverse conceptual structures. Second, a tuple can consist of many attributes, each one with different types, like numbers in continuous or discrete domains, dates, currency and strings. Also, some numeric or textual attributes are categorical (i.e., discrete, and perhaps unordered), making the direct comparison of values meaningless. There are many distance functions that can be used, and each of them will lead to a different visualization of the dataset, as the distance function is the base for the mapped result.

To be able to use the FastMap algorithm to map sets of tuples in relations, we propose a generic distance function that handles each attribute individually, so the difference of each attribute value from a pair of tuples are collected, and all of them are composed to create the final distance function. In this way, the set of tuples are viewed as a set of points in an E-dimensional space, where I is the cardinality of each tuple. Many distance functions have been used in different applications, including machine learning, neural networks and statistics [14]. When applied to multi-dimensional spaces, many of these distance functions are based on the differences between the values of each attribute. That is, let X=<x_1, x_2,... x_E> and Y=<y_1, y_2, ... y_E> be tuples from the dataset. Therefore, the distance function is $d(X,Y) = f(\partial(x_i, y_i))$, where $\partial(x_i, y_i)$ is a difference function between the values of each attribute. The most common of such distance functions are the Minkowsky distance functions, also known as the L_p norm:

$$d_{Lp}(X,Y) = \sum_{i=1}^{N} \sqrt[p]{(x_i - y_i)^p}$$

The value p, usually an integer number, enables the representation of several common distance functions, such as p=2 gives the Euclidean distance, p=1 comes to the Manhattan distance, and p=∞ is the Chebychev, or $L_{Infinite}$ distance function.

The L_p norm is usually applied over numeric attributes in continuous domains. However, when applied over attributes stored in a generic relation, the attributes can be in discrete domains, or texts, dates, etc. Then, we generalized the (x_i-y_i) component into a $\partial_t(x_i, y_i)$ function, which is chosen depending on the data type of each attribute in the relation. Note that a relation with attributes of different data types will use different $\partial_t()$ functions. Hence, we proposed a extended, weighted, Minkowsky distance functions to be applied to selected attributes of a database relation. As distance functions are based on the difference of each attribute in pairs of tuples, we represent the attributes used to calculate each difference, proposing the following definition.

Definition 1 (difference function). Let A_i be an attribute of a database relation,and let x_i,y_i be the values of attribute A_i in tuples X and Y of that relation, getting its values from the domain $Dom_u(A_i)$. Then $\partial_{tu}[A_i](x_i, y_i) \to \mathbb{R}$ is a *difference function* that calculates the difference between values x_i *and* y_i, using different procedures in the calculation, depending on the type t and $Dom_u(A_i)$ The following forms a basis set of attribute domains supported:

Continuous numeric domain: $\partial_N[A_i](x_i, y_i) = |x_i - y_i|$, where $\partial_N[A_i]$ calculates the absolute difference between the attribute values. This function can also be used when the attribute domain is ordered and discrete with equal steps between values;

Date domain: $\partial_D[A_i](x_i, y_i) = |days(x_i) - days(y_i)|$, where $\partial_D[A_i]$ calculates the number of days between dates, plus the fraction of time if x_i or y_i have time values specified;

Currency domain: $\partial_C[A_i](x_i, y_i) = |x_i - y_i|$, where $\partial_C[A_i]$ calculates the absolute difference between the attributes values;

String domain: $\partial_L[A_i](x_i, y_i) = L_{Edit}(x_i, y_i)$, where $\partial_L[A_i]$ calculates the Levenshtein distance function between two strings (the number of characters that should be inserted, removed or replaced to transform one text into the other);

Categorical (numeric or textual) domain: $\partial_M[A_i](x_i, y_i) = Mi_{I_{x_i} I_{y_i}}$, where Mi is a matrix representing the distance function in the domain of attribute A_i. This matrix must meet the requirements of a metric distance function (that is, $Mi_{jk} = Mi_{kj}$, $Mi_{kk} = 0$, $Mi_{jk} \geq 0$ and $Mi_{jk} \leq Mi_{jl} + Mi_{lk}$, $\forall jkl \in Dom(A_i)$).

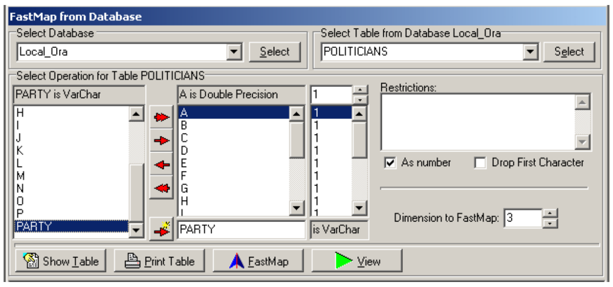
The categorical difference function $\partial_M(\,)$ stands for any difference function that can be expressed by the set of differences between every possible pair of instances from the domain $Dom(A_i)$. Examples of the usage of this difference function are the difference of terms based on the meaning of a set of terms (semantic difference), physical distance of the objects represented by its names (e.g the distance of cities identified by its names), etc. Being able to calculate the difference between values of any attribute of a relation, a function to calculate a distance between any pair of tuples can be created, using the following definition.

Definition 2 (Distance Function). Given a relation of the form $\boldsymbol{R}$={A_1, ... A_N}, a *relation distance function* is a metric $d_{Lwp_u}(\boldsymbol{R}) = \sum_{i=1}^{N} \sqrt[p]{w_i \cdot m_i((\partial_{t_i}[A_i])^p)} \to \mathbb{R}$ that, applied to the attributes of the relation returns a real positive number. Parameter w_i is the weight associated to the A_i attribute, and $\partial_{t_i}(A_i)$ is a difference function that varies with both the data type of attribute A_i, and with modifiers associated individually to each attribute A_i.

The weight w_i is used to deal with variations in the magnitude of the values stored at each attribute. It can be both a constant or calculated as the span of the values of this attribute in the database, providing a way to normalize the space. The modifiers $m_i(\,)$ associated to each $\partial_{t_i}(A_i)$ difference function are the logarithm (l), no modifier (null) and exponential (e) functions. Using them, different behavior regarding the scale of values at each single attribute can be accommodated. Notice there are many different distance functions that can be specified to each relation, changing weights, modifiers, power, the use of categorical or continuous difference functions, or distinct definitions for the categorical attributes. The usability of each of them have definite implications in the visualizations obtained, but the description of them are beyond the scope of this paper.

4. The FastMapDB tool

Using the proposed extended weighted Minkowsky distance functions and the FastMap algorithm [13], we built the visual analyzer tool called FastMapDB (Figure 1). It aims to assist, through visual resources, the data analysis step in the KDD process using data stored in relational databases. The purpose of the tool is to allow the user to "see" the object distribution in a 3D space. It allows, for example, to verify the existence of outliers, to assist on tasks of cleaning and data preparation, to help the user to choose reduced sets of attributes to submit to data mining operations, and to verify the adequacy or not of a chosen distance function or the answer generated by an automatic process.


FastMap from Database

Select Database: Local_Ora — Select

Select Table from Database Local_Ora: POLITICIANS — Select

Select Operation for Table POLITICIANS

PARTY is VarChar: H, I, J, K, L, M, N, O, P, PARTY

A is Double Precision: A, B, C, D, E, F, G, H, I — 1, 1, 1, 1, 1, 1, 1, 1, 1, 1

PARTY — is VarChar

Restrictions:

[✓] As number   [ ] Drop First Character

Dimension to FastMap: 3

Fig. 1. The FastMapDB main screen.

Describing briefly, the FastMapDB tool might be used in a sequence of 6 steps:

1 - to select the database;

2 - to select the relation ***R*** to be visualized;

3 - to select the attributes $\boldsymbol{A'}=\{\boldsymbol{A_i} \in \boldsymbol{R}\}$ that are going to compose the visualization, and each respective difference function $\partial_i[A_i]$;

4 - to define the distance function $d_{Lwp}[\boldsymbol{R}]$;

5 - to define visualization parameters;

6 - to generate and interact with the visualization.

These steps are executed sequentially by the user. It is possible to return to any previous step, however, when steps 1 or 2 are executed, the data of subsequent steps are discarded. The tool presents a graphical interface and the options of attribute selection are displayed. For example, to choose a relation to be visualized, the user might select the desired one among the list of all database relations shown. In the same way, once the relation ***R*** is chosen, its attributes are displayed. Then, the user selects the desired attributes to compose the set ***A'***.

Once the set ***A'*** is defined, the user may proceed to define the distance function $d_{Lwp}(\ )$. The options include: defining the respective weights of each attribute used with each attribute; choosing each difference function $\partial_i[A_i]$ including its modifiers to determine if it will be used in linear, exponential or logarithm scales; the distance matrix for categorical attributes; and the power *p*.

The user can also choose some filters to select only part of the tuples in the relation, or he/she can select the full relation. The selection is done by the conjunction of one or more selection conditions over the original attributes of the relation ***R***. In this step, it is also possible to choose a categorical attribute to be used as a classifier attribute, allowing tuples of different classes to be represented in different shapes and colors. After the distance function and the selection criteria were defined, the program performs the mapping algorithm and generates the visualization. Finally, the user can explore the generated visualization, and interactively manipulate it through operations as rotation, zoom and translation.

4.1. Performance Measures and Examples

In this section we present the results of applying FastMapDB tool on synthetic and real datasets. The results highlight two aspects: the scalability of the tool regarding the number of attributes and the number of tuples, and how visualization resources assist the understanding of datasets.

Scalability

In these experiments, we have used 5 tables with 10, 20, 40 and 80 thousand tuples. Each table has 10 floating-point attributes generated randomly, and a classifier categorical attribute with 3 possible values, also generated randomly. The first experiment was performed measuring the time spent by the tool to map the 10 numeric attributes for each table. The results of this experiment are shown in Figure 2. Figure 3 shows the results of another experiment, that evaluated the time spent by the tool, considering only the table with 20 thousand tuples, varying the number of attributes from 5 to 10.

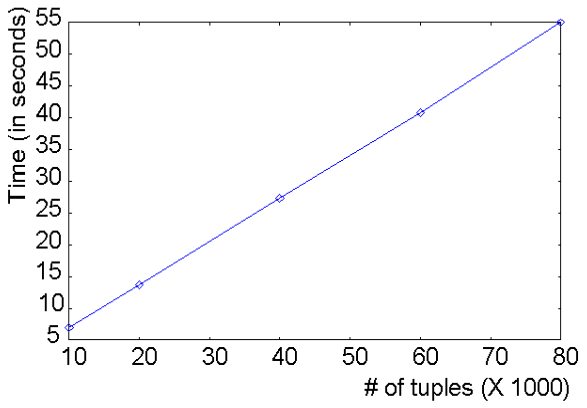


Fig. 2. Scalability graphic of FastMapDB. Regarding number of tuples.

The times shown in these figures refer to the total time since the user requested the mapping, including the reading of all data from the database, the mapping process, until the data visualization. As we can see in Figure 2 the tool presents linear performance, either varying the number of attributes or the number of tuples in the relation. All tests were realized on a Pentium III 800 MHz, with 512 Mbytes of memory, running MS Windows 2000 operating system and accessing an Oracle 8i database.

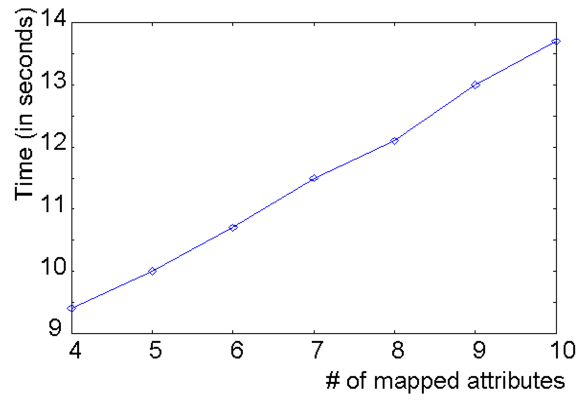


Fig. 3. Scalability graphic of FastMapDB. Regarding number of attributes.

Data Visualization

The next experiment was done to illustrate the data visualization resources of our proposed tool. The data used for this experiment is the "Congressional Voting Records" set of 1984, from the UCI Machine Learning Archive [15].

This dataset is constituted by the register of the votes of the congressmen, where each tuple corresponds to the vote of one congressman in sixteen congressional matters. The attributes have the value 1 (approved), -1 (not approved) or zero (neutral or abstention). Each tuple has also a categorical attribute, indicating which party the correspondent congressman belongs, Republican (168 tuples) or Democrat (267 tuples). This experiment illustrates the non-continuous data visualization. It is known this set presents a good separability between democrats and republicans, and Figure 4 presents it, showing how these two classes are polarized with little overlapping. The visualization shown in Figure 4 presents the results of the mapping on all 16 attributes.

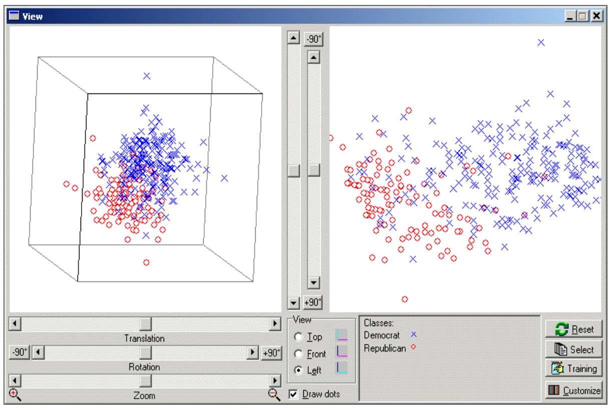


Fig. 4. The Votes set visualization.

4.2. Applications

This section describes some real applications where FastMapDB is being used. In the first one, customer data came from a financial institution, containing fraudulent households and a random sample of non-fraudulent households[3]. For each household were provided 90 fields, containing information such as the amount of accounts of a certain type (e.g. savings, credit lines), the balance in each of these accounts and the days between transactions at accounts of a given type. Using the available data it's possible to visualize how fraudulent households are mixed with non-fraudulent households, according to some selected fields, and detect the most probable ones to

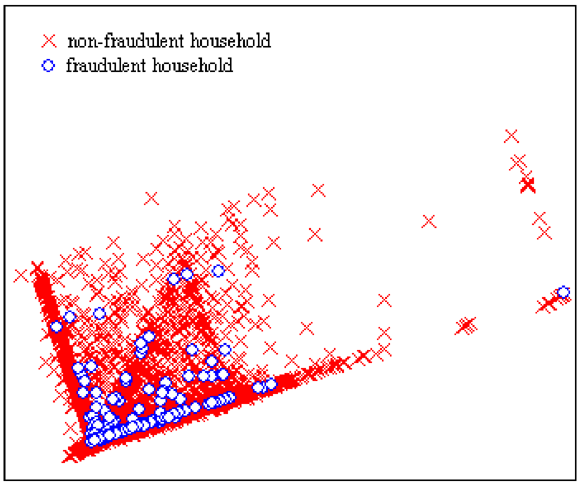


Fig. 5. The customer dataset visualization.

[3]Due to confidentially requirements we are not allowed to publish the institution's name and any other related information.

capture frauds. An example of visualization obtained from this dataset is shown in Figure 5. Other application where FastMapDB is being used is in analyzing data from a radiology image center of an hospital. In this application, factors that affect the quality of image production are being analyzed.

5. Conclusions

There are many tools to visualize information extracted from databases, but most of them allow the visualization of statistic information about data, as totals, averages, etc., in resembling pie charts, histograms, x-y graphics, etc. The few tools that allow spacial visualization of how data are distributed, use an inherent distribution, in other words, data must have a natural spacial distribution in a subset of its attributes. FastMapDB, on the other hand, creates spatial distributions for any kind of data stored in a relational database, and presents a rich pallette of tools to allow exploring many inter-relationships in the data to generate the visualizations.

Its data visualization concept allows using the human capacity of graphic data understanding, to effectively including the user as an important link in data analysis process of knowledge discovery in huge databases. An additional achievement is its suitability to validate either the data or the data analysis algorithms. This enables, for example, to verify if a data mining process such as classification or clustering has a good result, and if not, it may provide visual clues of the reason of the problem.

The main contributions of this work are:

- the development of a technique for incremental data visualization from relations built on attributes of any kind;
- the development of an algorithm for visualization, which is fast and scalable for huge datasets, presenting linear computational complexity concerning the number of tuples and the number of attributes in a relation;
- the implementation of a tool for use in real databases, stored in commercial DBMS;

Future developments expected involves academic activities and practical extensions. Among the academic activities are the possibility of using the concepts for classification and visual clustering discovery, and the possibility of analyzing more than one relation simultaneously, orienting the tool for Data Warehousing and On Line Analytical Processing (OLAP). Among the practical extensions are the inclusion of different manners of data manipulation and other tools to help the creation of distance functions, and to help compare changes in the settings of different visualizations.

Acknowledgments

This research has been partially supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo, Brazil) under grants 00/04184-9 and 00/04183-2, and by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil) under grants 52.1685/98-6, 860.068/00-7 and 35.0852/94-4.

References

1. Fayyad, U.M., *Mining Databases: Towards Algorithms for Knowledge Discovery.* Bullettin of Tech. Committee on Data Engineering, 1998. 21(1): p. 29-48.
2. Hinneburg, A. and D.A. Keim. *Clustering Methods for Large Databases: From the Past to the Future*. in *ACM Int'l Conference on Data Management (SIGMOD)*. 1999. Philadelphia, PA: ACM Press.
3. Ankerst, M., et al. *Visual Classification: An Interactive Approach to Decision Tree Construction*. in *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. 1999. San Diego, CA: ACM Press.
4. Louie, J.Q. and T. Kraay. *Origami: A New Data Visualization Tool*. in *Fifth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. 1999. San Diego - CA USA: ACM Press.
5. Keim, D.A., *Designing Pixel-Oriented Visualization Techniques: Theory and Applications.* IEEE Trans. on Visualisation and Computer Graphics, 2000. 6(1): p. 59-78.
6. Hinneburg, A., D.A. Keim, and M. Wawryniuk, *HD-Eye: Visual Mining of High-Dimensional Data.* IEEE Computer Graphics and Applications, 1999. 19(5): p. 22-31.
7. Ribarsky, W., et al., *Discovery Visualization Using Fast Clustering.* IEEE Computer Graphics and Applications, 1999. 19(5): p. 32-39.
8. Ng, R.T. and J. Han. *Efficient and Effective Clustering Methods for Spatial Data Mining*. in *Intl. Conf. on Very Large Databases (VLDB)*. 1994.
9. Zhang, T., R. Ramakrishnan, and M. Livny. *BIRCH: An Efficient Data Clustering Method for Very Large Databases*. in *ACM SIGMOD Intl. Conf. on Management of Data - SIGMOD*. 1996. Montreal, Quebec, Canada: ACM Press.
10. Ester, M., et al. *Algorithms for Characterization and Trend Detection in Spatial Databases*. in *Fourth Intl. Conf. on Knowledge Discovery and Data Mining*. 1998. New York City, NY: AAAI Press.
11. Syed, N.A., H. Liu, and K.K. Sung. *A study of support vectors on model independent example selection*. in *Fifth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. 1999. San Diego, CA USA: ACM Press.
12. Ganti, V., J. Gehrke, and R. Ramakrishnan, *Mining Very Large Databases.* IEEE Computer, 1999. 32(8): p. 38-45.
13. Faloutsos, C. and K.-I.D. Lin. *FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets*. in *ACM Int'l Conference on Data Management (SIGMOD)*. 1995. Zurich, Switzerland: Morgan Kaufmann.
14. Wilson, D.R. and T.R. Martinez, *Improved Heterogeneous Distance Functions.* Journal of Artificial Intelligence Research, 1997. 6: p. 1-34.
15. Hettich, S., *The UCI Machine Learning Repository*. 2000, University of California at Irvine.