Interactive Visualization for Knowledge Discovery

by

Jianchao Han

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2001

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Knowledge discovery in databases (KDD) has been actively pursued. Currently, two approaches are extensively researched and developed. One is algorithm-based and the other is visualization-based. The algorithm-based methods specify the target formats and pursue the correlations between the outcome variable and the independent variables. Once the algorithms are determined, the user can hardly participate in the discovery process. The visualization-based methods specify the hypothesis by means of visualization metaphors and pursue interactive visualization of large data sets and the patterns behind the data sets. Most visualization systems lack the ability to visualize the entire process of knowledge discovery, neither consider to include the users' perception into the systems.

We propose an interactive visualization model, RuleViz, for the KDD process, which stresses the human-machine interaction and visual representation. The interaction between the user and the machine helps the KDD system navigate through the enormous search spaces and recognize the intentions of the user. Thus, the user can easily provide the system with heuristics and domain knowledge and specify parameters. On the other hand, the visual representation of data and knowledge resulted in the KDD process helps users gain better insight into multidimensional data, understand the intermediate results, and interpret the discovered patterns.

RuleViz consists of five components: raw data preparation and visualization, interactive data reduction (horizontally and vertically), visual data preprocessing such as missing values handling, numerical attribute discretization and data transformation, pattern discovery like correlation mining, rule induction and decision tree construction, and pattern visualization like neural networks, decision trees, and classification list.

To implement the RuleViz model, we suggest three implementation paradigms: image-based algorithmic implementation, embedded-algorithm-based implementation, and user-supervised interactive implementation, and implement four interactive knowledge discovery systems: AViz – an image-based system for discovering numerical association rules based on data plots and optimized rectangles; CViz – an embedded-algorithm-based system for classification rule induction based on the parallel coordinates visualization technique; CVizT – a user-supervised interactive system for building classification rules based on the Table Lens visualization technique; and DTViz – a user-supervised interactive system for constructing decision trees based on the parallel segments pixel-oriented visualization technique and the tree structure visualization algorithm.

Our experimental results with the UCI repository data sets and artificial data sets demonstrate that the RuleViz model can provide a methodology for developing interactive KDD systems. The systems developed according to the RuleViz model take advantage of both algorithm-based and visualization-based approaches. They provide the user with straightforward observation and control of the KDD process, integrate the user's perception and domain knowledge into the KDD process easily, make it convenient for the user to understand and interpret the discovered knowledge, and remain flexible for distributing the KDD functions between the user and machine.

Acknowledgements

First of all, I would like to thank my supervisor Dr. Nick Cercone, for his insightful guidance, grateful discussion and continuous support throughout this study.

I would also like to thank my external Dr. Yiyu Yao, internal Dr. Andrew Wong, and reviewers Dr. Forbes Burkowski and Dr. Dale Schuurmans for their valuable advice and suggestions.

Many thanks to Dr. Aijun An for her kindly support with the ELEM2 system. Much credit goes to my friend Yingwei Wang, Richard Brown, Mike Xie for their provision of time to experiment with the systems developed in this thesis.

My special thanks go to my wife Jessie Hou for her moral support and everlasting encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Knowledge discovery in databases (KDD) is a process of searching for knowledge represented as relationships and patterns from large data sets and employment of this knowledge to solve problems or interpret phenomena. The KDD process is best suited to be interactive and iterative, involving numerous steps with many decisions being made by the user [53]. *Data mining* is an important activity in *knowledge discovery in databases*, and develops techniques and approaches to discovering new patterns from databases [35]. Data mining attracts many researchers from the database and artificial intelligence communities, as well as other disciplines such as logic and statistics. Their interest is motivated by the availability of huge amounts of computerized data that many organizations possess about their business. Research on data mining techniques concentrates on the solution of various problems, including classifying data into different categories, characterizing a set of data, discovering associations and correlations between data, finding sequential patterns and similarities in ordered data, and so on.

Currently, two important approaches for knowledge discovery can be char-

acterized as algorithm-based and visualization-based [24]. The algorithm-based approach includes rule induction [11], concept learning [123], association mining [3, 4, 117], decision tree construction [26, 136], neural networks [42], etc. These methods specify the target formats and pursue the correlations between the outcome variables and the independent variables. Usually, patterns are hidden in the raw data set, and may have particular structured or unstructured forms. It is essential to discovering knowledge to determine how the raw data should be cleaned and preprocessed and where the patterns are to be found in the database. Machine learning techniques and statistics are extensively exploited in data mining algorithms.

The other approach, the visualization-based approach, specifies the relationships among data items or variables by means of visualization metaphors. A working definition of visualization views it as *a process of transforming the abstract data into a meaningful visual form so that users can understand and use the results much better* [37]. The visualization method integrates techniques from computer graphics, image processing, graphical user interfaces, and other related fields, and takes into account human perceptual cognitive capabilities, human variations, and task characteristics. Interactive visualization provides the user with straightforward observation and control of the data mining process.

Many visualization techniques and systems have been developed and implemented [8, 13, 20, 44, 46, 30, 70, 76, 97]. These visualization systems can be divided into two categories, one which visualizes the raw data like Spotfire [8], Independence Diagrams [20], Periodic Data [44], and VisDb [97], and the other which visualizes final results like Decision Tree Construction [13], Map Visualizer and Tree Visualizer developed by Silicon Graphics [30, 70], association rules visualization [61], and visualization of various patterns [76]. One common feature of

these existing visualization systems is their dependence on computer graphics and scientific visualization.

Most existing methods for knowledge discovery reported in the literature [4, 12, 42, 133, 110, 149, 154] stress the interactions between the human and the machine. Their implementations, however, focus on the algorithmic approaches for data mining rather than the interactions. Some visualization systems develop either the raw data visualization or the final results visualization, but few of them emphasize visualization of the entire KDD process and the interactions for navigating through the search space.

The data set on which the knowledge discovery process works is usually huge, and hence it is very difficult to grasp the patterns or rules behind the raw data. Visualization techniques provide us with various approaches to gain insight into the meaning of the data and help us to interpret the patterns discovered. Most visualization systems, however, lack the ability to visualize the entire process of knowledge discovery. In data visualization systems, complex data are carefully arranged and displayed in a specific visual form, and the knowledge behind the raw data is left for the user to observe and determine the meaning of the pictures, which usually requires a wealth of background knowledge in graphics and applications. On the other hand, the knowledge visualization systems visualize the discovered knowledge according to different techniques such as decision trees, neural networks, etc. However, these systems display only results based on some parameters specified by the user, and the user cannot understand how the results are obtained and interpreted. An interactive visualization system of knowledge discovery should provide a user with not only the raw data and/or the discovered results but also the entire process in visual forms so that the user can participate in the discovery process, provide heuristics, guide the navigation of search, and interpret the discovered knowledge.

1.1 Contributions

In order to take advantages of algorithm-based approach and visualization-based approach for discovering knowledge from large data sets, in this thesis, we propose an interactive visualization model, RuleViz, which stresses two aspects of interactive visualization, human-machine interaction and visual representation during the entire KDD process. The RuleViz model consists of five components: data visualization, data reduction, data preprocess, pattern learning, and knowledge visualization. The interaction between the user and the machine helps the KDD system navigate through the enormous search spaces and carry out the intentions of the user, and the user easily provides the system with heuristics and domain knowledge and specifies parameters. The visualization of the KDD process helps users gain better insight into the multidimensional data, understand the intermediate results, and interpret the discovered patterns.

To support the RuleViz model, we implement four interactive systems, AViz, CViz, CVizT, and DTViz, for visualizing the process of learning classifiers, mining association rules, and constructing decision trees, respectively. The reason that these patterns are chosen in our implementations is that they are actively pursued [3, 6, 9, 10, 11, 39, 66, 103].

In our implementations, we focus on the data visualization, feature selection and tuple selection for data reduction, missing value handling and continuous attribute discretization for data preprocessing, learning strategies, and discovered pattern visualization. Three approaches for implementing the RuleViz mode are developed in these implemented systems. The first one is based on image processing, the second approach is to embed existing algorithms into the visualization systems, and the third one is fully interactively finding patterns. We experiment and evaluate

these systems with UCI repository data sets and artificial data sets.

We make the following contributions in this thesis:

1. We propose an integrated interactive model, RuleViz, consisting of five components, raw data visualization, data reduction, data preprocess, pattern learning, and pattern visualization, for visualizing the entire process of knowledge discovery, and suggest possible approaches for implementing each component of this model;

2. We implement an interactive visualization system, AViz, for mining association rules, in which approaches for learning patterns and discretizing continuous attributes based on image processing techniques are developed;

3. We implement a visualization system, CViz, for rule induction, in which an approach for interactively learning classification rules with visualization by embedding existing algorithms into different components is explored;

4. We implement an interactive system, CVizT, for constructing classifiers, in which an approach for fully interactive classification rules construction is presented; and

5. We implement an interactive system, DTViz, for interactive building decision trees, which provides a novel approach for "supervised" learning in which the user is the "supervisor".

1.2 Organization of This Thesis

This thesis is organized as follows. In Chapter 2, we present an overview of the algorithm-based KDD techniques, especially focusing on statistical classification

techniques, decision tree techniques, and association rule mining techniques. Other issues, including pattern evaluation, missing value handling, continuous attribute discretization, etc. are also considered. We also summarize techniques for data visualization and knowledge visualization, analyze the visualization framework and process, and present some data and knowledge visualization systems.

In Chapter 3, we propose an integrated interactive model, RuleViz, for visualizing the entire process of discovering knowledge from large data sets. The RuleViz model consists of five components, data visualization, data reduction, data preprocess, pattern learning, and knowledge visualization. The problems, objectives, available techniques, etc. for each component are discussed, and some novel approaches for each component are suggested.

In Chapter 4, the implementation issues of the RuleViz model are presented, including choosing the visualization space, choosing visualization techniques for different targets, choosing implementation methods, data traversability and navigability, human-machine interactions, and windowing strategies.

In Chapter 5, we design an interactive system, AViz, for visualizing the process of mining numerical association rules. The implementation of each component of AViz is presented, and our AViz experiments with census data sets are reported. Image-based continuous attribute discretization and association rule mining algorithms are developed.

In Chapter 6, we embed an existing learning algorithm ELEM2, continuous attribute discretization algorithm EDA-DB, and feature selection algorithm RELIEF into a *parallel coordinates*-based data visualization technique to explore a visual classification rule induction system, CViz. Additionally, interactive data reduction and a novel technique for visualizing classification rules are implemented. Our ex-

periments of CViz with the UCI data sets [126] and artificial data sets are reported at the end of this chapter.

In Chapter 7, we develop an interactive visualization system, DVizT, for constructing classifiers (a set of classification rules) based on Table Lens, a graph-based data visualization technique. This is a fully interactive system. A visualization-based continuous attribute discretization and a Table Lens based visual representation of classification rules are developed in DVizT. The experiments of DVizT with the UCI repository data sets are reported.

In Chapter 8, another fully interactive visualization system, DTViz, is described. The DTViz system is used to interactively build decision trees. Two interaction windows are developed, one for data, one for decision tree. The user can interactively expand tree nodes, evaluate tree nodes, prune tree branches, and assign labels to leaf nodes by interacting with the decision tree window, and visualize the data covered by tree nodes and select split attributes and split points by interacting with the data window, respectively. A pixel-oriented data visualization technique, Parallel Segments, is developed for ordered attribute values with class labels. The DTViz system is experimented with the UCI data sets and the results are reported.

In Chapter 9, we analyze and compare the characteristics of the four systems above, AViz, CViz, CVizT, and DTViz, from different angles. We also compare these systems with other visualization-based and algorithm-based KDD systems.

Finally, Chapter 10 is the conclusion. We provide a summary of accomplishments and make suggestions for our future work.

Chapter 2

Knowledge Discovery and Visualization

2.1 Introduction

Knowledge discovery in databases (KDD) is a multi-disciplinary research endeavour, which has been defined as *the non-trivial extraction of implicit, previously unknown and potentially useful knowledge from data* [133]. The purpose of knowledge discovery is to search for knowledge represented as relationships and patterns and employ this knowledge to solve unseen problems or interpret phenomena.

Two approaches for knowledge discovery can be characterized as algorithm-based and visualization-based [24]. The former includes rule induction [11], concept learning [123], association mining [3, 4, 117], decision tree induction [26, 136], neural networks [42], etc. These methods specify the target formats and pursue the correlations between the outcome variable and the independent variables. The latter approach specifies the hypothesis by means of visualization metaphors. Many

techniques and systems for data visualization and/or knowledge visualization have been developed and implemented [8, 13, 20, 44, 46, 30, 70, 76, 97]. These visualization systems can be further divided into two categories, one visualizing the raw data like Spotfire [8], Independence Diagrams [20], Periodic Data [44], and VisDb [97], and the other visualizing the final results like Decision Tree Construction [13], Map Visualizer and Tree Visualizer developed by Silicon Graphics [30, 70], assoication rules visualization [61], and visualization of various patterns [76]. One common feature of these existing systems is their dependence on computer graphics and scientific visualization.

In this chapter, we briefly introduce some algorithm-based KDD approaches which will be used in later chapters, and then discuss the concepts and techniques of data and knowledge visualization for underlying visualization-based approaches.

2.2 Algorithm-Based Knowledge Discovery

We summarize the important algorithm-based approaches for knowledge discovery, including statistical classfication, decision tree construction, associations mining, and other approaches.

2.2.1 Statistical Classification

Generally, the classification problem can be understood as follows [68, 93, 85]. We are given a set of attributes or features, denoted $\{A_1, A_2, \ldots, A_p\}$, with attribute A_j defined in the domain $Dom(A_j)$ for $j = 1, 2, \ldots, p$, a set of predefined classes, denoted $\{C_1, C_2, \ldots, C_q\}$, and a training set with n examples $\{(e_1, c_1), (e_2, c_2), \ldots, (e_n, c_n)\}$. Each training example can be featured with the given attribute values,

called p-dimensional attribute vector, $e_i = < x_{i1}, x_{i2}, \ldots, x_{ip} >, i = 1, 2, \ldots, n$, where $x_{ij} \in Dom(A_j), j = 1, 2, \ldots p$, is the value of attribute A_j with respect to e_i, and has been already assigned to one and only one class c_i. The problem is how to construct a classifier that is able to assign a class to a new example that is not in the training set.

Statistical classification, as the term suggests, uses probability theory and statistical methods to build a classifier from the given data. Many approaches have been developed, which can be divided into two categories [123]. One is the *parametric* approach [49, 127, 23, 33, 34, 67, 105, 118, 139, 145] which assumes that the form of distribution density function of a p-dimensional attribute vector for each class is known, and the aim of classification is to estimate the distribution density function by estimating some parameters from the training set. When a new example is observed, it can be assigned to a class for which it has the greatest probability. Typical *parametric* approaches include Linear Discriminant [127], Quadratic Discriminant [49], Logistic Discriminant [33], and the method of using the mixture likelihood model to solve classification problems with a mixture of labeled and unlabeled examples [23, 62, 67, 118, 129, 130, 146]. The second statistical approach is the *non-parametric* or *distribution-free* approach [42, 48, 49, 59, 127, 162] which assumes that no form of underlying distribution density function is known, and the classification procedure builds some mechanism to estimate the distribution density or probability for each class directly from the known data when a new example is presented to it. Typical *non-parametric* approaches include Kernel-based Density Estimation [42], K-nearest Neighbours Method [123], Projection Pursuit [49], etc.

2.2.2 Decision Tree Techniques

Construction of decision trees from a given set of examples has been the subject of extensive research in machine learning and data mining in the past over ten years [26, 66, 123, 135, 136, 159, 162]. It is typically performed in two steps, *growing* and *pruning*. In the first step, a training set is sampled from the given set of examples and used to grow a decision tree. Then, in the second step, the tree is reduced based on a *testing set* to prevent *overfitting*. The testing set is generally the complement of the training set with respect to the given set of examples.

Decision Tree Generation

The skeleton for growing a decision tree from a training set T is elegantly simple [136]. Let the decision classes be denoted $C_1, C_2, \ldots, C_k$.

If T contains no cases, then the decision tree is only a leaf, but the class to be associated with the leaf must be determined from information other than T.

If T contains one or more cases, all belonging to a single class C_j, then the decision tree for T is a leaf identifying class C_j, $1 \leq j \leq k$.

If T contains cases that belong to a mixture of classes, then T is partitioned into subsets $T_1, T_2, \ldots, T_n$, where T_i contains all cases in T that have output O_i of the chosen test, based on a single attribute which has one or more mutually exclusive outcomes $O_1, O_2, \ldots, O_n$. The decision tree for T consists of a decision node identifying the test, and one branch for each possible outcome. The same tree-building machinery is applied recursively to each subset of training cases, so that the i-th branch leads to the decision tree constructed from the subset T_i of training set $T, i = 1, 2, \ldots, n$.

Building a decision tree depends a great deal on the choice of appropriate tests, which we have apparently extracted from thin air. Any test that divides the training set in a nontrivial way, so that at least two of the subsets are not empty, will eventually result in a partition into single-class subsets under the condition of no noise examples, even if all or most of them contains a single training case. The tree-building process is to build a tree that reveals the structure of the domain and so has predictive power. The partition must have as few blocks as possible. Ideally, we would like to choose a test at each stage so that the final tree has small number of nodes and high classification accuracy.

Unfortunately, the problem of finding the smallest decision tree with a training set is NP-complete under the measure of internal length [90]. Of course, what is the "best" depends on how one intends to use the tree. The most common criterion of measuring a decision tree is called *accuracy of prediction*, or, in contrast, *error rate*, when classifying previously unseen objects. There are several popular measures for selecting attributes to extend the decision tree, for example, *information gain and gain-ratio measure* used in the ID3 and C4.5 systems [135, 136], *the GINI-index of diversity measure* used in CART [26], *the Marshall correction measure* [121], *the* χ^2 *statistical measure* [123], etc.

Decision Tree Pruning

There are two broad classes of pruning algorithms [114, 122]. The first class includes algorithms like *cost-complexity* or *error-complexity* pruning, that use a separate set of examples for pruning, distinct from the set used to grow the tree. The second class of decision tree pruning algorithms, like pessimistic pruning, uses all of the training examples for tree generation and pruning.

Assume we have N testing examples, t is a node of the decision tree, T_t a sub-tree rooted at t, L_t a set of leaf nodes in T_t, $N(t)$ the number of examples contained in t, $e(t)$ the number of misclassified examples, and $r(t) = \frac{e(t)}{N(t)}$ the error rate at t.

Error-complexity pruning [26]: Define the data rate $p(t)$ as $\frac{N(t)}{N}$. Then the error cost of node t is

$$R(t) = r(t)p(t) = \frac{e(t)}{N}, \tag{2.1}$$

and the error cost for sub-tree T_t is

$$R(T_t) = \sum_{i \in L_t} \frac{e(i)}{N} = \frac{1}{N} \sum_{i \in L_t} e(i). \tag{2.2}$$

The complexity cost is the cost of one extra leaf, $\alpha(t)$. Then the total cost of the sub-tree T_t is $R(T_t) + \alpha(t)|L_t|$, and the total cost of node t is $R(t) + \alpha(t)$ (only one leaf node t). Let them be equal, then

$$\alpha(t) = \frac{R(t) - R(T_t)}{|L_t| - 1}, \tag{2.3}$$

which gives a measure of the value of the sub-tree, the reduction in error per leaf.

The node that is to be pruned should be the following:

$$t_{prune} = \arg\min_t \{\alpha(t)\}.$$

The final tree is selected as the smallest tree with a misclassification rate (per test data) within 1 standard error of the minimum (1 SE Rule):

$$SE = \sqrt{\frac{R(100 - R)}{N}}, \tag{2.4}$$

where R is the misclassification rate of the pruned tree.

Pessimistic error pruning [136]: The number of misclassifications that need corrected is estimated as

$$n(t) = e(t) + \frac{1}{2}, \quad \text{for node } t, \tag{2.5}$$

and the rate with continuity correction of sub-tree T_t as

$$n(T_t) = \sum_{i \in L_t} e(i) + \frac{|L_t|}{2}, \quad \text{for subtree } T_t. \tag{2.6}$$

The standard error for the number of misclassifications is calculated as

$$SE(n(T_t)) = \sqrt{\frac{n(T_t)(N(t) - n(T_t))}{N(t)}}. \tag{2.7}$$

To prune the decision tree, each node starting at the root is evaluated. A node along with the sub-tree rooted at it is pruned unless its corrected number of misclassifications is lower than that for the node by at least one standard error:

$$n(T_t) < n(t) - SE(n(T_t)).$$

2.2.3 Classifier Evaluation and Other Considerations

Classification error rate, or *accuracy*, is often used to evaluate a classifier, a set of classification rules, acquired by using a classification learning algorithm [26, 136]. Suppose, there are n cases in the training set T and N cases in the testing set T'. The classifier generated from T is denoted by D. For any case $x \in T'$, assume its actual class is $C(x)$, but D classifies it into $D(x)$, then the classification accuracy of D is defined as

$$R(D) = \frac{|\{x \in T'; D(x) = C(x)\}|}{N} \tag{2.8}$$

There are three methods of estimating the classification accuracy which are called *internal estimates* [26, 123, 124].

1. *Resubstitution estimate:* The test cases are the same as training cases;
2. *Test sample estimate:* Given a set of cases, divide it into two parts, one as a training set and the other a test set;

3. *Cross-validation estimate:* Divide the case set into more subsets $S_1, S_2, \ldots, S_K$. Let $T_k = \cup_{j \neq k} S_j$ be training set, $T'_k = S_k$ testing set. Assume the classifier generated from T_k is D_k, then

$$R(D_k) = \frac{|\{x \in T'_k; D_k(x) = C(x)\}|}{N_k}, \tag{2.9}$$

where N_k is the size of T'_k. Thus, $R(D) = \frac{1}{K} \sum_{k=1}^{K} R(D_k)$.

During the construction of classifiers, it is easier to deal with discrete-valued attributes than continuous-valued attributes. How the continuous-valued attributes are handled has been become an important topic of machine learning and data mining [12, 40, 42, 51, 52, 136, 152]. Many methods of transforming continuous-valued attributes into discrete-valued ones have been developed in recent years, e.g., in C4.5, the cut-points were used to replace the corresponding intervals [136]. CAL5 is a learning system based on a statistical approach to splitting real values into discrete intervals, each of which is treated as a discrete value [125]. The real-valued attribute is thus transformed into a discrete one. There have been other approaches to discretizing continuous attributes into discrete ones which we will disscuss in the following chapters.

A data set is *incomplete* if it contains tuples with missing values, such a tuple being called an *incomplete tuple.* Missing data may result from errors, measurement failures, difficult observation, changes in the database schema, etc. Incomplete tuples are still very useful in the process of discovering rules although they may provide us with less information than tuples without missing values. For the task of learning classification rules, the missing values may be the condition attributes, or the class labels (decision attribute values). For the former case, the missing values can be estimated in some way or the incomplete tuples are simply removed

in the pre-precessing. In the latter case, the missing labels may lead to completely different results.

Missing values present additional problems. Several solutions to the problem of generating classifiers from the training set of examples with unknown values have been proposed [26, 123, 124, 135, 136]. The simplest among them consists in removing examples with unknown values or replacing missing values with the most common values. More complex approaches are to use a Bayesian formalism to determine the probability distribution of the missing value over the possible values from the domain, and either choose the most likely value or divide the example into fractional examples, each with one possible value weighted according to the probabilities determined.

2.2.4 Mining Association Rules from Large Data Sets

Discovering association rules from large data sets is actively pursued currently [2, 3, 4, 7, 27, 83, 88, 106, 113, 117, 131, 144, 151, 156]. An *association rule* is an implication of the form $A \Rightarrow B$, where A, B are subsets of regarded attributes, and $A \cap B = \phi$. For different types of attribute values the *implication* of the association form may have different meanings. Currently, the proposed algorithms can be classified into three categories, *boolean* , *quantitative*, and *generalized* association rules.

Boolean Itemset Association Rules

Agrawal, Imielinski, and Swami presented the *basket data* problem [3], in which, given a large database of customer transactions (*basket data*) that consists of items purshased by a customer in a separate transaction, we are to look for significant

associations between items, for example, which items are always or often bought together with which items. The typical examples are *most often transactions that purchase bread and butter also purchase milk*, and *a customer purchasing tea is likely to also purchase coffee.*

Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of items, each I_k being an *item*, where m is the number of items. Let D be a database of transactions, where each transaction corresponds to a tuple in D, denoted T. A transaction T is a set of items represented as a binary vector, with $T[k] = 1$ if T contains the item I_k, and $T[k] = 0$ otherwise, for $1 \leq k \leq m$. Assume $X \subseteq I$ is a subset of items in I. A transaction T satisfies X if for all items $I_k \in X, T[k] = 1$.

An *assocation rule* is an implication of the form $X \Rightarrow Y$, where $X, Y \subset I$, and $X \cap Y = \phi$. The rule $X \Rightarrow Y$ holds in D with *confidence* c if $c\%$ of transactions in D that contain X also contain Y, and *support* s if $s\%$ of transactions in D contain $X \cup Y$.

The problem of mining association rules from a large set of transactions D is to generate all association rules with support and confidence greater than or equal to the user-specified support threshold (δ) and confidence threshold (σ), respectively. Confidence is a measure of the rule's strength, while support corresponds to statistical significance. The problem of mining association rules can be decomposed into two subproblems:

1. Discovering all large itemsets: A subset X of I is called a *large* itemset or an *interesting* itemset or a *frequent* itemset if its support is at least δ. Otherwise, it is a *small* itemset.

2. Generating association rules: For each large itemset X, construct rules of the form $X - Y \Rightarrow Y$, where $Y \subset X$. If the confidence of a rule in such a form is

greater than or equal to σ, then it is an association rule that we are finding. Otherwise it is not.

From statistics, the support for a rule $X \Rightarrow Y$ is given as the probability of occurrence of $X \cup Y$ in D, denoted as $p(X \cup Y)$, while its confidence is the conditional probability of Y given X. Confidence can be expressed as $\frac{p(X \cup Y)}{p(X)}$.

Currently many approaches to solving the *basket data* problem have been developed and efficient algorithms have been implemented. The classical solution is based on breadth-first or level-wise search to generate all candidate large itemsets and then to test them [3, 4, 6, 28, 111]. Another solution is based on sampling over the full data set of an appropriate size to avoid excess scanning passes over the data set [132, 144, 156]. In addition, alternative methods such as the connectionist approach [110] have also been proposed.

Quantitative Association Rules

Generally, *quantitative values* consist of *numerical values* and *categorical values.* Typical numerical values are salary, age, quantity, and so forth, while country, sex, major, etc. are categorical values. Handling quantitative attributes is to partition the domains into disjoint intervals [152].

To transform quantitative attributes into positive consecutive integer attributes, assume any attribute has the uniform format: <*attribute, integer-value*>, and tuple in the data set is *a set of* <*attribute, integer-value*>.

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of attributes, P the set of positive integers, and I_v the set $I \times P$. The pair $< x, v > \in I_v$ means that attribute x has value v. Let $I_R = \{< x, l, u > \in I \times P \times P | l \leq u$, if x is quantitative; $l = u$, if x

is categorical}. Item $< x, l, u >$ means that the value of x is between l and u. For $X \subseteq I_R$, $attribute(X) = \{x| < x, l, u > \in X\}$. A tuple $R \subseteq I_v$ supports $X \subseteq I_R$ if for all $< x, l, u > \in X, \exists < x, q > \in R$ such that $l \leq q \leq u$. Thus, the quantitative attributes are transfered to a set of items with form of $< x, l, u >$. The quantitative association rule is of the following form: $X \Rightarrow Y$, where $X, Y \subset I_R$ and $attribute(X) \cap attribute(Y) = \phi$.

As long as the partitions are made, the problem can be solved by means of the algorithms developed in the previous section. The main problem to be addressed is how to partition the domains. If the number of intervals is large, then the support for any single interval can be low, while if the number of intervals is small, the interval length is large and the confidence can be low. Thus many rules could be missed. There are currently many approaches to partitoning quantitative values, such as *equi-space partitoning*, *equi-depth partitoning*, *distance-based partitoning*, *entropy-based partitioning*, etc. [12, 52, 75, 117].

Generalized Association Rules

Most often, *taxnomies* over attributes are available. When hierarchies are present, the association rules between different levels of hierarchies are usually interesting, especially in the case that most leaf nodes have not sufficient support. Climbing up the hierarchies would generate *generalized* association rules. Such generalized association rules and algorithms for discovering such rules are discussed by Cai, Cercone and Han [31], Srikant and Agrawal [151], Han and Fu [83], Srikant, Vu, and Agrawal [153], etc.

The interestingness degree of a rule is defined as its support divided by its *expected* support, and the expected support is based on the support of the rule's

antecedent and consequent [151]. A *generalized association rule* is an implication of the form $A \Rightarrow B$, where $A, B \subset I, A \cap B = \phi$, and no item in B is an ancestor of any item in A. A generalized association rule is interesting if it has no ancestors or its interestingness degree is greater than some interest threshold. The ancestors of a rule $A \Rightarrow B$ consist of $\hat{A} \Rightarrow B$, $A \Rightarrow \hat{B}$ and $\hat{A} \Rightarrow \hat{B}$, where $\hat{A}$ and $\hat{B}$ are the ancestors of A and B respectively.

The problem of mining generalized association rules can be decomposed into three parts, finding all generalized large itemsets, generating desired rules, and pruning all uninteresting rules. The generalized large itemsets include the classical large itemsets and those large itemsets that contain higher level items. Since the support of an ancestor may not be equal to the summation of the support of its direct descendants, all items at all levels have to be counted to decide their support.

To handle hierarchical relationships, the background knowledge for hierarchies could be stored into a table consisting of two columns: *item_name* and *level_difference*. The mining algorithm for generalized association rules is based on the algorithms discussed in previous sections but running on the extended transaction data set. For each transaction t, an extended transaction t' of t is obtained by adding all the ancestors of each item in t to t. Thus, t supports itemset X iff t' is a superset of X. The algorithm consists of two phases, the first phase generating candidate frequent itemsets and the second phase determining support of those candidates [83, 153].

2.2.5 Other Techniques Employed in Data Mining

Chen et al [38] suggest several classification schemes to categorize data mining techniques, for example, based on the kinds of databases to be studied, the kinds of knowledge to be discovered, and the kinds of techniques to be utilized, etc.

According to the kinds of knowledge to be mined, except for data classification and association rules mining discussed previously, the data mining techniques also contain the following categories [38, 73, 84, 89]:

- Data generalization and summarization: This technique presents the general characteristics or a summarized high-level view over a set of user-specified data in a database.
- Interesting/exceptional/Peculiar rules mining: Similar to association rules mining which is based on support-confidence framework, other kinds of interesting rules can be discovered using different metrics [47, 58, 65, 69, 86, 107, 163].
- Data clustering: This technique is to group a set of data (unsupervised) based on the conceptual clustering principle: maximizing the intraclass similarity and minimizing the interclass similarity.
- Characteristic rule mining: A characteristic rule is an assertion that shows the characteristics of the data classes, though the rule may not cover all classes and all examples of any given class.
- Discriminant rules mining: A discriminant rule is an assertion that discriminates concepts of the class being examined (the target class) from other classes (called contrasting classes).
- Temporal or spatial-temporal (sequential) pattern mining: This pattern is also called *trend*, such as financial data for stock price indices, customer data, medical data, etc. It is to search for similar patterns in order to discover and predict the risk, causality, and trend associated with a specific pattern.

- Path traversal patterns mining: This technique is to capture user access patterns in a distribued information enviroments, such as the Web page access patterns.
- Neural networks: The problems that neural networks are expected to solve include prediction, pattern recognition, classification, optimization, associative memory, dignosis, and control [22, 36, 57, 92, 94, 142].
- Fuzzy set and rough set: Using fuzzy/rough set theory to discover fuzzy rules or approaximation of classification rules [74, 89].

2.3 Data Visualization

Data visualization combines the techniques from computer graphics, image processing, computer vision, graphical user interfaces, and other related research to visually represent abstract data that are often retrieved from large document collections such as banking databases, digital libraries, the World Wide Web, text or other databases. Data visualization takes into account human perceptual cognitive capabilities, human variations, and task characteristics. The data visualization techniques deal with the effective portrayal of data with the further goal of providing insights about large amounts of data [44, 46, 98].

Formally, **Data** or **information visualization** is defined as *the use of computer-supported, interactive, visual representations of abstract data to amplify cognition* [32] in the following six ways: by increasing the memory and processing resources available to the user; by reducing the search for information; by using visual representations to enhance the detection of patterns; by enabling perceptual inference

operations; by using perceptual attention mechanisms for monitoring; and by encoding information in a manipulable medium.

Recent research on data visualization pursues interactive visualization of large data sets and the patterns behind the data sets. Interactive visualization allows researchers to visualize the results of presentations on the fly and in different perspectives, e.g., angles, magnitude, layers, levels of detail, charts, curves, and so on, and thus helps users to understand and access the patterns behind the data sets better and easier. Interactive visualization systems are most useful especially when the results of models or simulations have multiple or dynamic forms, layers, or levels of detail [37].

2.3.1 Data Visualization Framework

Among various data visualization techniques, there is a basic common framework which is an extension of Abowd and Beale's model of human-computer interaction [99]. This basic framework consists of four main components: User, Database, Visualization and Interaction, as shown in Figure 2.1.

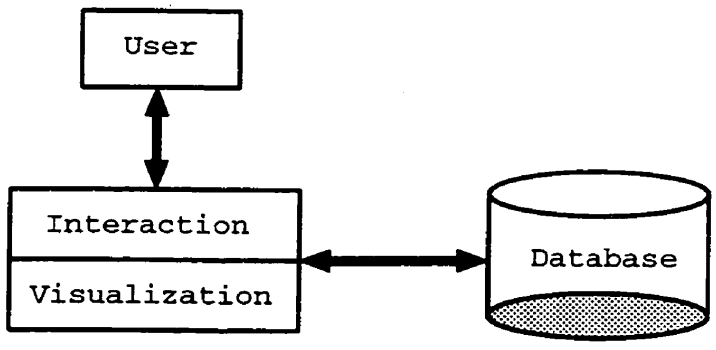


Figure 2.1: The Basic Framework for Interactive Visualization

The **User** component describes the human user of the system, providing a

description of his/her sophistication (e.g., expertise with the data visualized or with the system itself), the tasks which he/she wishes to perform (e.g., discovering associations between data, classifying data into categories, etc.), and his/her level of authority to view or modify data.

The **Database** component represents the actual database to be "mined" and visualized, encompassing the data model used, the schema, and the instances which populate the database.

The **Visualization** component presents the data to the user by using different predefined metaphors such as color, shape, shadow, texture, etc. to represent different items of data and arranging the physical layout on the screen. Generally, this component contains a subcomponent that extracts relevant information from the database, processes the extracted information, and transfers the results into metaphors.

The **Interaction** component addresses how the user alters the presentation of the data, or the data itself. This component interprets the user's intention, the medium selected to realize the intention, and the effect of the action selected. Interaction effects may force a recomputation of results for subsequent and simultaneous display, and change the data or visualization, or both.

Figure 2.1 illustrates this framework, where the interaction and the visualization components are combined to form the user-friendly interface, which is usually implemented as an interactive visualization.

This basic framework provides an integrated model, coordinating the subcomponents. This point is particularly important to ensure that the resulting system is open and able to cooperate with other software systems. This framework facilitates incorporation of interactive components from various sources and provides

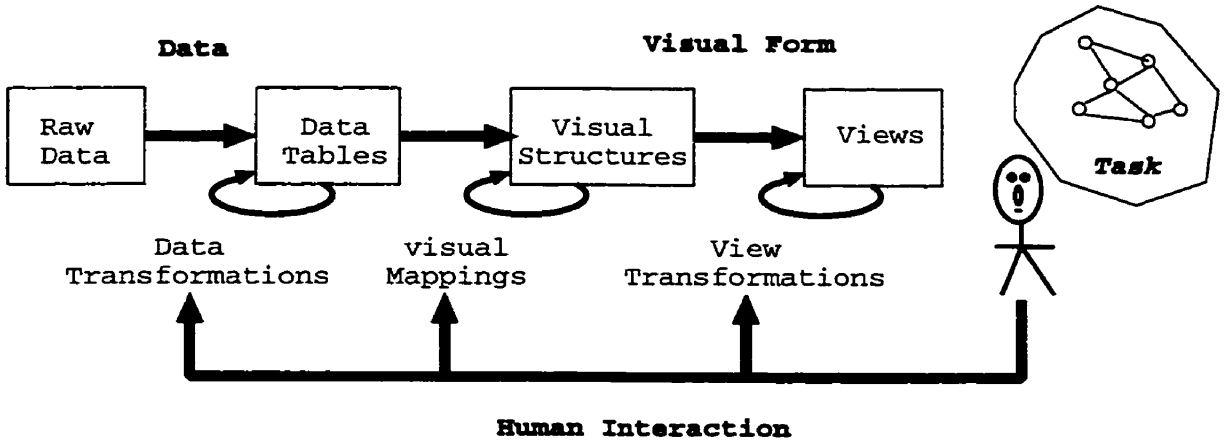


Figure 2.2: Data Visualization Process

for multiple visualizations of the same data.

2.3.2 Data Visualization Process

Data visualization can be imagined as a process of transforming the generated abstract data into a meaningful visual form to the human perceiver so that users can see and understand the results better. Card, Mackinlay, and Shneiderman [32] propose a simple reference model, which consists of three mappings, as shown in Figure 2.2. All mappings are fulfilled through human-system interactions. The human interaction is needed for the user to specify parameters, data ranges, visual forms, primitives, rendering methods, display positions, etc. in the service of tasks by moving sliders, pressing buttons, clicking mouse, and so forth.

Data Transformations

Data transformations map Raw Data into Data Tables. Raw Data comes in diverse forms, from spreadsheets to Web pages, from relational database tables to

multimedia data. Usually, the raw data is transformed into a relation or a set of relations that are more structured and thus easier to map to visual forms. Data Tables are usually *relational tables*, similar to the "tables" in relational databases. They consist of relations and *metadata*, where a relation is a set of *tuples*, while the *metadata* are variables or labels of columns or rows of the relations.

Variables may be one of the following three basic types: *nominal, ordinal*, or *quantitative*. For the purpose of visualization, different types of variables can be transformed into each other so that the generated abstract data to be visualized can have convenient formats for being displayed on screen. For example, quantitative variables can be transformed into ordinal variables by discretizing them into disjoint intervals; nominal variables can be transformed into ordinal variables by lexicographically sorting them; while ordinal variables can be transformed into nominal variables by igoring the ordering. Additionally, a hierarchical data type is very possible, where different values of variables constitute a tree or layered structure. For example, temporal variables may take values of "days", "months", or "years".

Metadata is descriptive information about data and important in choosing visualizaton. Generally, metadata is collected or generated directly from real-world applications. For example, individual ages, income, height, etc. are all metadata.

Raw data may not be in table forms, especially for unstructured data like hypertext documents. For our purpose, the data collected or generated from the real-world applications must be first transformed into Data Tables. This transformation typically involves the loss or gain of the informaton contained in the raw data. Usually, data transformations consist of a chain of transformations. As Tweedie [158] pointed out, data transformation often changes the structure of a data table, or the values of variables. The following are examples of data transformations: statis-

tical calculations of *mean* and *variance* from the raw data, estimating the missing values, changing non-fixed length transactions into fixed length tuples, discretizing numerical variables.

Visual Structures

The second step of data visualization is to map Data Tables to Visual Structures, which must preserve the data [21]. Visual Structures encode the information in Data Tables and represent the information in multiple ways.

Different data types have different properties, hence they should be distinguished to make their different topologies (geometries) reflect their properties. Data variables are depending on the coordinate system of the base space. The common coordinate systems include Cartesian coordinates, polar coordinates, cylindrical coordinates, spherical coordinates, and curvilinear coordinates. For a specific problem, the most appropriate coordinate system is chosen to take advantage of its inherent properties.

Usually, before Data Tables are transformed into Visual Structures, some visual primitives should be defined to reflect the features of the data in the data tables. These primitives may consist of following [21, 32]:

- Axes: According to variable types, four different axes can be created, *unstructured axis* (no axis) representing unstructured data, *nominal axis* (a region is divided into subregions), *ordinal axis* (the ordering of subregions is meaningful), and *quantitative axis* (a region has a metric). In addition, different types of axes may be represented in different ways. For example, a quantitative axis can be represented as a slider, while an ordinal axis or a nominal axis can be represented as a row of radio buttons.

- Marks: Marks are the visible things that occur in space, which include *points* (0D), *lines* (1D), *areas* (2D), *volumes* and *surfaces* (3D).
- Connection and Enclosure: Links between marks can be used to represent relations among objects or encode hierarchies.
- Arrows: Arrows can be used to indicate specific relations between objects or the trends of variable evolution.
- Time Series: Most data are obtained over a certain period of time. These temporal data should be sampled in *fixed time step size* or *variable time step size* and mapped to a series of frames or blocks.

After the data in Data Tables are cast into visual primitive representations, the visualization objects can be transformed into pictures, which are to be completed by rendering. Rendering techniques are dependent on primitives and also human perception. The cues of object shapes, depth, and motion are important factors of rendering techniques [29, 72].

View Transformations

The purpose of a view transformation is to turn the static presentations of Visual Structures into visualizations in space-time which can be viewed from specific angles. Three types of view transformations are widely used [32]:

1. **Location probes** use location in a Visual Structure to reveal additional Data Table information. This is a very common view transformation. Usually, an overview distribution of data is visualized, which can not reflect the details of all data due to the limitation of display window size. By choosing

specific points or regions (e.g., clicking, moving sliders, etc.), the corresponding data (perhaps in another data table) are visualized in a pop-up window. The resulting details-on-demand visualization gives details about more Data Tables. There are several probe techniques in use. *Brushing* creates visual effects at other marks when the cursor passes over one location [112]; *slicing plane probes* is to access the interior of 3D solid objects [72]; *Stream lines* render vector fields visible with additional depth cues to locate the curves in 3D spaces; and *magic lenses* give an alternate view of a region in the Visual Structure [55].

2. **Viewpoint controls** are similar to *location probes* in that both use probes of data points or regions to view details. However, location probes generally use the same window for probes, while viewpoint controls use affine transformations to zoom (in or out), pan, and clip the viewpoints in different windows. For example, Shneiderman [148] proposes a technique, *overview+detail*, where two windows are used together: an overview of the Visual Structure and a detail window that provides a magnified focus on area.

3. **Distortion** is a visual transformation that modifies a Visual Structure to create focus + context views. Overview and details are combined into a single Visual Structure. Distortion is effective when the user can perceive the larger undistorted Visual Structure through the distortion. Distortions can be roughly classified by what the human perceives as invariant [32]. *The bifocal lens* is an example of a 1D distortion that horizontally compresses the sides of the workspace by direct scaling and leaves ordering invariant, which supports the perception of linear sequence, although objects outside the focal area have distorted aspect ratios. *The table lens* is a 2D distortion

that leaves ordering invariant (for more information about the Table Lens, see Chapter 7). *Perspective wall* and *Hyperbolic tree* are 3D distortions that leave topological relationships invariant [104]. Figure 2.3 shows a perspective wall. The perspective wall is similar to the fisheye view. It is a variant of the bifocal lens. The data inside the focal area are presented on a perspective wall, while the data outside the focal area are perspectivelly reduced in size.

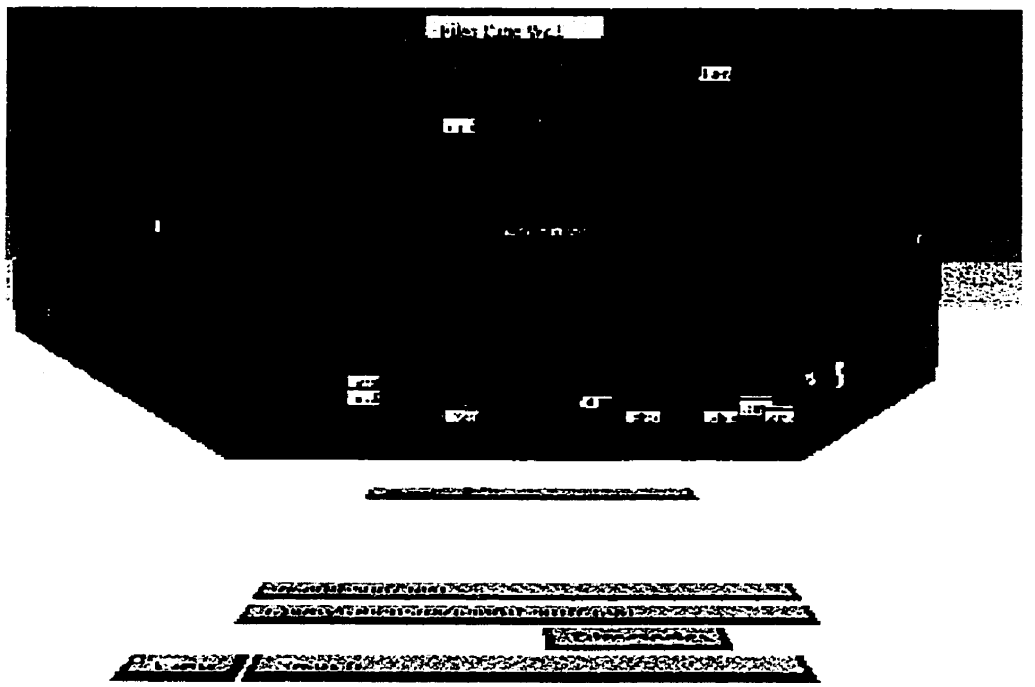


Figure 2.3: A Perspective Wall Visualization

2.4 Data Visualization Techniques and Systems

There are two related but quite distinct purposes for data visualization. One is for communicating ideas, concepts, rules, and so forth, because "a picture is worth

ten thousand words". Data visualization for this purpose is called *knowledge visualization* in our discussion in the sense that ideas, concepts, rules, etc. are all "knowledge". Knowledge visualization will be discussed in the next section. The other purpose is to exploit graphics and image processing to build concepts, ideas, and relations, or to verify hypotheses or discover structures or regulations. Put in another way, it is using the special properties of human visual perception to resolve logical problems [21]. This is called *data visualization.* The goals of data visualization can be summarized as following three aspects [98]:

- **Exploratory Analysis**

  Based on the data, without hypotheses about the data, data visualization searches for structures, trends, and clusters interactively, and finally displays the data in visual forms to provide hypotheses about the data.

- **Confirmatory Analysis**

  Data visualization accepts hypotheses about the data as the input, makes goal-oriented examination of the hypotheses, and outputs the data in a visual form which allows the confirmation or rejection of the hypotheses.

- **Presentation**

  Data visualization fixes the facts to be presented a priori, chooses an appropriate presentation technique, and gives the high-quality visualization of the data to present the facts.

There have been many well-known systems for visualizing multidimensional data sets, such as Spotfire [8], Visage [46], KnowledgeSeeker [70], DataMind [70], DViz [76], VisDB [97], MinSet [30], etc. The techniques that have been exploited in the existing visualization systems for visual data mining can be classified into five categories [98].

2.4.1 Geometric Projection Techniques

Geometric projection techniques [95, 98] aim at finding "interesting" projections of multidimensional data sets. The class of geometric projection techniques includes techniques of exploratory statistics such as principle component analysis, factor analysis and multidimensional scaling, many of which are subsumed under the term "projection pursuit". Since there are an infinite number of possibilities to project high-dimensional data onto the two display dimensions, "projection pursuit" systems aim at automatically finding the interesting projections or at least helping the user to find them. So the key to this technique is in the presentation of data and/or relationships among them.

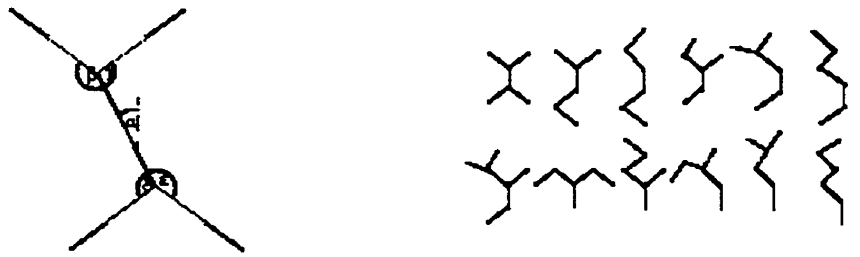
Another geometric projection technique is the parallel coordinate visualization technique [91]. The parallel coordinate technique maps the n-dimensional space onto the two display dimensions by using n equidistant axes which are parallel to one of the display axes. The axes correspond to the dimensions and are linearly scaled from the minimum to the maximum value of the corresponding dimension. Each data item is presented as a polygonal line, intersecting each of the axes at that point which corresponds to the value of the considered dimension. Although the principle idea of the parallel coordinate visualization technique is quite simple, it is powerful in revealing a wide range of data characteristics such as different data distributions and functional dependencies. However, since the polygonal lines may overlap, the number of the data items that can be visualized on the screen at the same time is limited. For more information about the parallel coordinate technique, see Chapter 6.

2.4.2 Icon-based Techniques

Icon-based (or iconic display) techniques map each multidimensional data item to an icon. There are several well-known iconic display techniques developed, e.g., Shape Cording, Stick Figures, Chernoff-Faces, Group Technique.

The Shape Coding technique allows the visualization of an arbitrary number of dimensions [18]. In this approach, the icon maps each dimension to a small array of pixels and arranges the pixel arrays of each data item into a square or rectangle, and each pixel represents one attribute value, which is mapped to grey scale or color according to the data value of the dimension. The small squares or rectangles corresponding to the data items are then arranged line-by-line according to a given sorting, e.g., the time attribute for time-series data.

The Stick Figure technique allows a visualization of larger amounts of data and is therefore more adequate for data mining [98]. As indicated by the name, the icon is some type of stick figure. In addition, two dimensions are mapped to the display dimensions and the remaining dimensions are mapped to the angles and/or limb lengths of the stick figure icon. Figure 2.4 shows a) the individual stick figure icon, and b) a family of stick figure icons [96]. If the data items are relatively dense with respect to the display dimensions, the resulting visualization presents texture patterns that vary according to the characteristics of the data and are therefore detectable by preattentive perception. Different stick figure icons with variable dimensionalities may be used. Figure 2.5 illustrates two visualizations of the properties of the triangulation of molecule data using stick figure technique [96].



a. Stick Figure Icon

b. A Family of Stick Figures

Figure 2.4: Stick Figure Icons

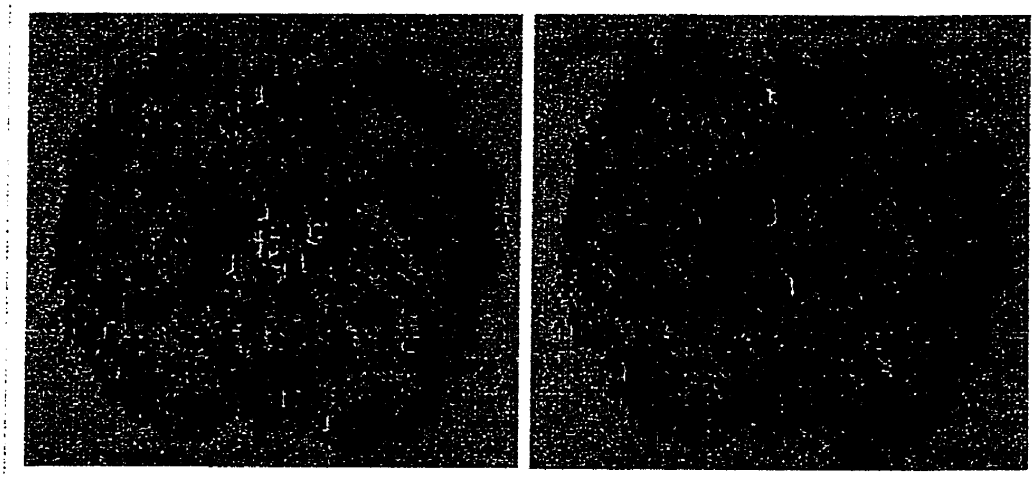


Figure 2.5: Stick Figure Visualizations

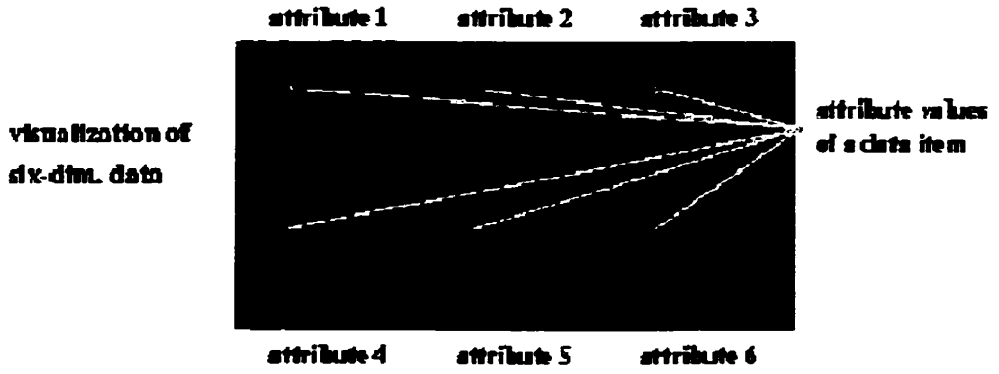


Figure 2.6: Pixel-oriented Visualization with Six Dimensions

2.4.3 Pixel-oriented Techniques

Using pixel-oriented techniques, each attribute value is represented by one colored pixel, or the value ranges of the attribute are mapped to a fixed colormap, and the attribute values for each attribute are represented in separate subwindows. Figure 2.6 shows a pixel-oriented visualization of data set with six attributes (six dimensions), where each dimension occupies a rectangle. A tuple consisting of six value items is visualized with six pixels which are distributed in different rectangles and rendered with different colors in terms of the item value [96]. This technique has been successfully used in the VisDB system [95]. If each data value is represented by one pixel, the main question is how to arrange the pixels on the screen.

There are two main techniques to implement pixel-oriented data visualization, one is the querying-dependent visualization technique, the other is the query-independent visualization technique. The former visualizes the relevance of the data items with respect to a query, calculates the distances between data and query values, combines the distance for each data item into an overall distance, and then visualizes the distances for attributes and overall distance sorted according to the

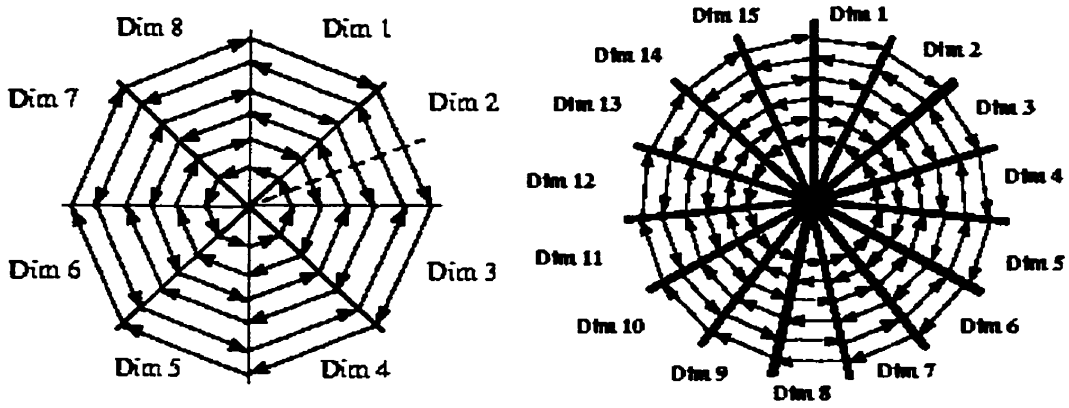


Figure 2.7: Circle-segments Visualizations with Eight and Fifteen Dimensions

overall distances. There are many visualization patterns used to present the results of a query. The Spiral technique and Circle-segments technique are two important ones. Spiral technique produces the curve like a rectangular spiral shape around the centre. The Circle-segments technique is to display the distances for the attributes as segments of a circle. If there are k attributes, the circle is partitioned into k sections equally and the distance values are arranged from the centre to outside in a back and forth manner orthogonal to the line that halves the segments. Figure 2.7 illustrates two examples of circle-segments visualizations with eight (left) and fifteen (right) attributes (dimensions), respectively.

The query-independent visualization technique is especially useful for data with a natural ordering according to one attribute. This technique sorts the data according to some attributes and uses a screen-filling pattern to arrange the data values on the display. One major problem is to find meaningful arrangements of the pixels on the screen. Simple arrangements are to arrange the data from left to right in a line-by-line fashion, or from top to bottom in a column-by-column fashion. However, more useful techniques of arrangement provide a better clustering of closely

related data items. One such technique is the space-filling curve. For data mining, it is more important that arrangement techniques provide nice clustering properties as well as an arrangement which is semantically meaningful. The recursive pattern techniques [97] can provide this function. The recursive pattern is based on a simple back and forth arrangement which arranges a certain number of elements from left to right, and then below backwards from right to left, then again forward from left to right, and so on. The same basic arrangement is done on all recursion levels with the only difference that the basic elements which are arranged on level i are the patterns resulting from level $i-1$ arrangements.

2.4.4 Hierarchical Techniques

Hierarchical techniques subdivide the k-dimensional space and present the subspaces in a hierarchical fashion. There are also several well-known hierarchical techniques, such as n-Vision, Treemaps, and the dimensional stacking.

The n-Vision technique is also known as "worlds within worlds" [54]. It partitions the k-dimensional space into several three dimensional subspaces. Figure 2.8 [96] shows the visualization of a six-dimension space by having a new coordinate system where the last three dimensions sit inside the coordinate system for the first three-dimension, where the outside three dimensions contain *Strike, Foreign,* and *Time*, while the inside dimensions consist of *Butterfly, Volatility,* and *Spot.*

The treemaps technique focuses on visualizing multivariate functions and can be therefore used to particularly represent data relationships and patterns with tree structure such as file systems and decision trees [147]. This technique is a screen-filling method that uses a hierarchical partitioning of the screen into regions whose size depends on the attribute values, while whose color may correspond to an

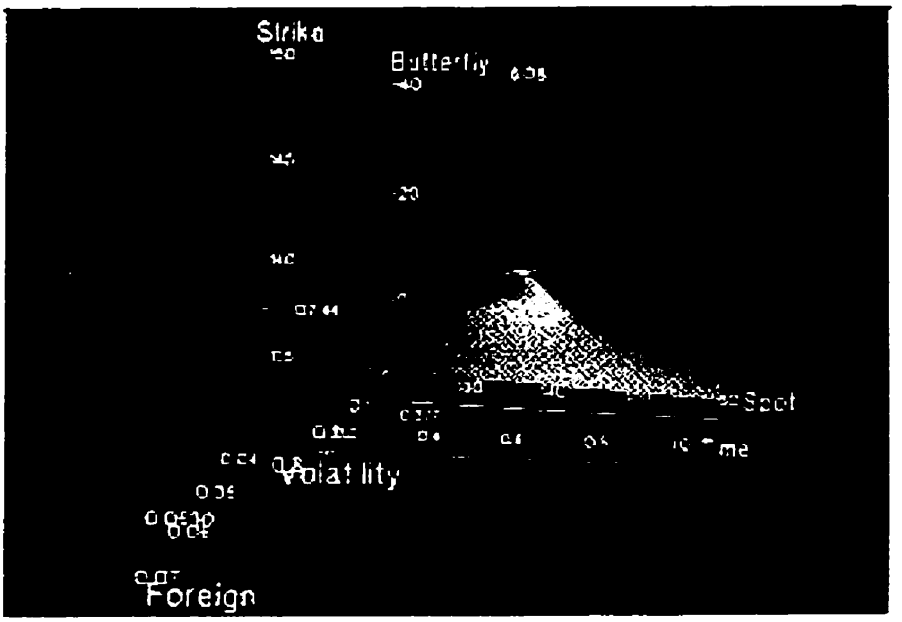


Figure 2.8: The n-Vision Visualization of a Six-dimension Space

additional attribute. This technique is very suitable to get an overview over large amounts of hierarchical data and for data with multiple ordinal attributes. Figure 2.9 illustrates a treemap of a file system containing about 1000 files [96].

Moreover, dimension stacking is also a very useful hierarchical technique. It subdivides the k-dimensional space into two dimensional subspaces which are "stacked" into each other, and partitions the attribute-value ranges into classes. The attributes are used from the inner levels to the outer levels according to their importance. This technique is adequate especially for data with ordinal attributes of low cardinality.

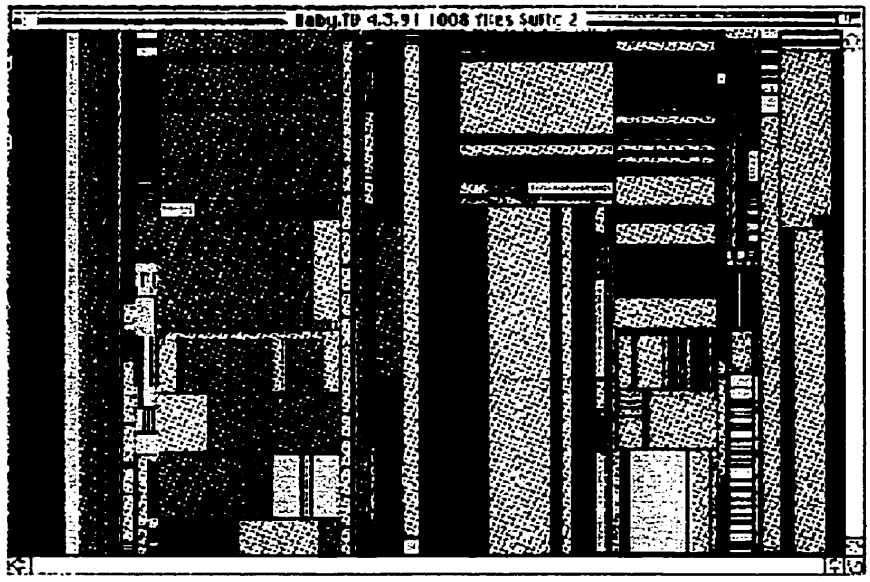


Figure 2.9: The Treemap of a File System Containing 1000 Files

2.4.5 Graph-based Techniques

The graph-based techniques effectively present a graph using specific layout algorithms, query languages, and abstraction techniques. The most important approaches of the graph-based techniques include Hy^+, SeeNet, etc.

Hy^+ is an integrated query and visualization system and used to visualize structural data as hygraphs which are a generalization of several diagrammatic notations [32]. Hy^+ queries are described as expressions of GraphLog, a special query language, and may be used to define new relationships and reduce the data to be visualized. This technique is used to visualize a Web browsing session.

SeeNet is a visualization system of hierarchical networks with weighted links [17]. It has special features such as semantic node placement (minimizing the distance of nodes with high-weighted links), attributes mapped to size and color of nodes and

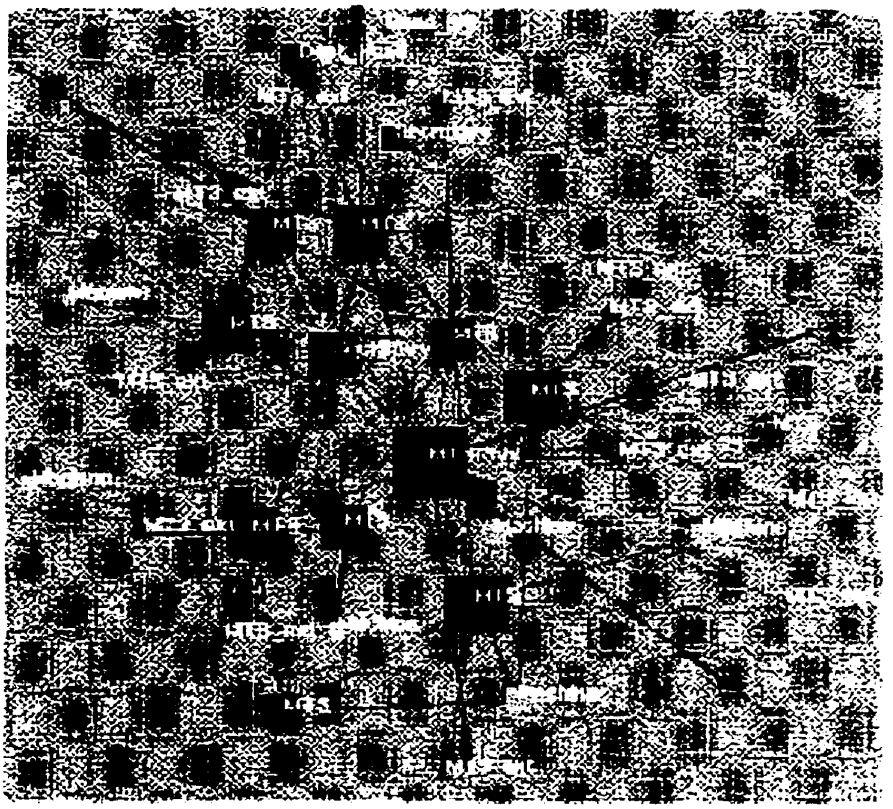


Figure 2.10: The SeeNet Visualization of E-mail Messages

links, and interactivity for changing the mappings, expanding or collapsing nodes within the hierarchy and getting additional information. Figure 2.10 illustrates the visualization of all e-mail connections in a department over a period of time [96], where the size of node represents the number of e-mail messages of a person, the color of node corresponds to the function of staff members, and the size of links are mapped from the number of e-mail messages of the link.

2.5 Knowledge Visualization

As mentioned in the previous section, knowledge (or pattern) visualization is to communicate ideas, concepts, and rules, which are usually obtained from the raw data. For the purpose of communication, knowledge must be visualized in visually perceptible ways so that the human beings can easily understand what it is. Just as different real-world data sets with different properties require different techniques for visualization, different patterns have different visual perceptions. For example, classification rules and association rules can be visualized as relations between attributes and their values, while neural networks and decision trees have their own natural structures. Recent research on visualizing different knowledge patterns represents a growing interest in machine learning and data mining [13, 20, 30, 61, 74, 75, 76, 77, 81, 101, 128, 137]. In this section, several systems and techniques for visualizing different patterns are introduced, including decision trees visualization, associations and correlations visualization, and the MineSet system.

2.5.1 Decision Trees Visualization

The decision tree is an important knowledge structure which is constructed from a training set. There have been several algorithms for visually constructing decision trees. For example, the MineSet system implements the 3D Tree Visualizer [30]; Rao and Potts implemented a system, CAT (Classification Aggregation Tablet), for visualizing a bagged decision tree [137]; KnowledgeSEEKER developed by Angoss [70] is a comprehensive program for classification tree analysis and can automatically grow the decision tree under the user's guidance who selects menu items and specifies parameters.

A decision tree is different from the usual *tree*, although both have the same

structure. Some systems merely visualize the decision tree structure only [1, 128], while the others are trying to visualize the construction of the decision tree [13].

Nguyen et al. [128] implement a tree visualizer for their CABRO system. They focus on two problems of decision tree visualization: integration of tree visualization into an interactive KDD process and visualization of large trees. The tree visualizer can be invoked during the discovery process and can handle large trees by combining several techniques in information visualization. The tree visualizer provides several *views*, each of which uses variant techniques and drawing algorithms to display trees, and serves different usage purpose. The available views include:

- *Standard*: The tree is drawn in proportion to size of nodes, length of labels, number of children, etc.
- *Tightly-coupled*: Use *viewpoint control* technique to do the view transformation. Both overview window and detail window are exploited.
- *Fish-eye*: Distortion view transformation is used so that the interesting nodes are magnified.
- *T2.5D*: The focused paths of the tree are drawn in the front and in highlight colors, while the other paths are in background and dim colors.

The tree visualizer also provides the following operations for the user to customize different views: *Zoom* in or out of the drawn tree; *collapse/expand* the tree paths to view different parts; and *display* the content of a node such as attribute-values, labels, population, etc.

Ankerst et al. [13] develop an interactive approach, PBC (Perception Based Classification), for constructing decision trees which is based on a pixel-oriented vi-

sualization technique, *circle segments*. It assumes that all attributes are numerical. The decision tree construction consists of the following steps:

- *Visualizing* the corresponding data by using a circle segments technique, which maps d-dimensional data to a circle that is partitioned into d segments, each of which represents an attribute;
- *Selecting* the attribute to be split in each node according to the class labels distribution by viewing the circle segments;
- *Determining* the split points for the attribute in the current node, where the spliting strategies include *best pure partition, largest cluster partition, best complete partition*, and *different distribution partition*;
- *Growing* the decision tree by spliting the node;
- *Updating* the decision tree by specifying and labeling the leaves or removing one level (similar to pruning).

Two visualization windows are used, the *data interaction window* for interactive operation such as selecting attributes, specifying split points, etc., and the *knowledge interaction window* for growing and pruning decision tree. The experiment of PBC with the shuttle data shows they can achieve the same accuracy as well-known decision tree approaches including CART and C4.5, even better.

2.5.2 Associations and Correlations Visualization

Many data mining algorithms focus on the discovery of associations and correlations among attributes or features. We introduced their basic ideas in the previous

chapter. There have been some visualization systems attempting to visually display the associations or correlations discovered from the large data sets.

Adriaans and Zantinge [1] describe the visualization of transaction (itemset) associations and experiment their system with the magazine data set to find the associations between magazine types (music, sports, shopping, etc.), customers, ages, and so forth. The natural structure, the *lattice*, among the frequent itemsets (frequent itemsets are nodes and *subset* relations are arcs for linking nodes), is displayed on the fly with support and confidence in each of the frequent itemsets. The algorithm for finding the frequent itemsets is running as a background process, while the results are visualized in the front.

Fukuda et al. [60, 61] implemented the SONAR system which discovers numerical association rules from two dimensional data by visualizing the raw data and finding an optimized rectangular or admissible region. The two dimensional data are mapped to the Euclidean plane, which is divided into *buckets* by discretizing the two numerical attributes. Buckets are rendered in terms of the points falling in them. Two classes of optimized regions on this plane, *rectangles* and *admissible regions* consisting of buckets, are to be found according to the optimized *gain* which is computed based on the association rule parameters: support and confidence. Each optimized region represents an optimized association rule. This is an image-based knowledge extraction system.

Hofmann et al. [87] show how Mosaic plots, especially their variants called Double Decker plots[1], can be used to visualize association rules. These plots visualize

[1]Mosaic plots and Double Decker plots are achieved based on the contingency table of attributes. They are constructed hierarchically, including the attributes one at a time in the user-specified order. Mosaic plots always alternate between height-splitting and width-splitting, while Double Decker plots always split the width (or the height). For more information, see [87].

the contingency table that yields the association rules as well as other potential rules in that table, whether they meet the thresholds or not. Thus a deeper understanding on the nature of the correlation between the left-hand side and the right-hand side of a rule can be achieved.

Berchtold et al. [20] propose another technique: independence diagrams, for visually mining correlations between two numerical attributes. This technique can be described as follows. Each attribute is discretized into disjoint intervals; for each pair of attributes, the combination of these intervals defines a two-dimensional grid. Each cell in this grid stores the number of data points in it. The grid is displayed by scaling each attribute axis so that the displayed width of an interval is proportional to the total number of data points within that inteval. The brightness of a cell is proportional to the density of data points in it. Thus, both attributes are independently normalized by frequency, ensuring insensitivity to outliers and skew, and allowing specific focus on attribute dependency. The process of generating an independence diagram consists of following steps:

- *Numerical attribute discretization*: A modified equi-depth histogram method is proposed to partition numerical attributes with two constraints: the width and depth of any interval should be smaller than the given constants, respectively.

- *Scaling*: Transform the attributes by applying some mathematical function, such as taking the logarithm, square root, etc., so that attribute values can be "properly scaled" to produce the maximum information to the visual analyst.

- *Shading pixels*: Map a count value for a cell into a grey-scale value or color value for the display.

- *Morphing*: Map a given two-dimensional equi-slice histogram in an equi-width manner or an equi-depth manner to aid the interpretation of the data.

The displayed images show the correlations between attributes. The general form of such correlation is: *While a fraction x of all points lie in range R_A in attribute A, among those data points lying in range R_B in attribute B, the fraction is y.*

2.5.3 MineSet: An Integrated Knowledge Visualization System

MineSet is an integrated system developed by SGI for data warehousing, data mining, and visualization for analyzing and understanding a vast wealth of information [120]. It supports a scalable client/server model for extracting knowledge from a data warehouse and revealing newly discovered patterns and connections through intuitive, interactive, and visual displays. MinSet integrates analytical algorithms for knowledge discovery including Decision Tree Inducer and Classifier, Option Tree Inducer and Classifier, Evdience Inducer and Classifier, Decision Table Inducer and Classifier, Regerssion Tree, Clustering Algorithm, and Column Importance. These analytical algorithms can be complemented with data visualization techniques taking advantage of the human's perception and intuitive pattern recognition capability. It provides the following visualizer:

- Map Visualizer: Display quantitative and relational characteristics of spatially oriented data, where data items are associated with graphics "bar chart" objects in the visual landscape which can consist of a collection of spatially related objects, each with individual heights and colors.

- Tree Visualizer: Display quantitative and relational characteristics of data by showing them as hierarchically connected nodes. It takes the data at the lowest level of the hierarchy as input, visualizes the data as a three-dimensional "landscape", and presents the data as clustered, hierarchical blocks (node) and bars through which part or all of the data set can be dynamically navigated and viewed.

- Scatter Visualizer: Disply data points in one, two, or three dimensions, and map additional attributes to color, size, and shape. Two more attributes may be mapped to sliders, allowing animation and fly-through, for a total of eight dimensions. The column importance operation in MineSet can help the user identify the important dimensions for a given task.

- Splat Visualizer: Similar to Scatter Visualizer, with the distinction that the data density is shown by opacity of color, which appears as a blurred translucent cloud. The result approximates the effect of rendering each data points individually.

- Statistic Visualizer: Display the data in the form of box plots and histograms, one attribute per column. Continuous columns are shown as box plots, and discrete columns are shown as histograms.

- Histogram Visualizer: Display the data in the form of histograms. Continuous columns are binned (broken down into ranges).

- Record Viewer: Display the raw data as a spreadsheet.

Chapter 3

RuleViz: An Interactive Model for Visualizing the KDD Process

3.1 Introduction

As mentioned in the previous chapters, there are two main categories of knowledge discovery approaches: algorithm-based and visualization-based. Both approaches were developed from different research fields and applied to different applications at their first emergence. Now they find a large territory in knowledge discovery and are growing mutually supportively. However, each of them has its own drawbacks. Let's consider their disadvantages respectively.

For algorithm-based knowledge discovery, once the algorithms [1] used in the process are determined, the user can hardly participate in the discovery process. What

[1] A KDD system may contain many algorithms for different purposes in different steps, e.g. continuous attribute discretization algorithm, data mining algorithm, output generation algorithm, etc.

the user can do is to provide the raw data and algorithms' input parameters. KDD systems look like black boxes. When the process terminates, the end results are output. Unfortunately, the results are often difficultly understood by the user, or it is hard for the user to interpret the results. For example, given a rule induction system with data preprocessing algorithms and a rule induction algorithm, assume the output of the system is a list of classification rules which are represented as logical implications. One may ask questions like why the continuous attributes were discretized in such a way, how the results were obtained, what the prediction accuracy of the rules is. Of course, one can look up related documents and explanations to interpret such questions. But for non-specialists or beginners, understanding these questions is not easy.

On the other hand, the visualization-based approaches provide users with straightforward outputs and can interact with users. But a common feature of existing visualization systems is their dependence on computer graphics and scientific visualization. Most visualization systems neither have the ability to visualize the entire process of knowledge discovery nor include the user's perception into the systems. In data visualization systems, the complex data are carefully arranged and displayed in a specific visual form, and the knowledge behind the raw data is left to the users who must observe and determine the meaning of the pictures. This usually requires a wealth of background knowledge on graphics and applications. Moreover, the knowledge visualization systems visualize the discovered knowledge according to different techniques such as decision trees, neural networks, etc. However, these systems only display the results on the basis of some parameters specified by the user, and the user can not understand how the results are obtained and interpreted. An interactive visualization system of knowledge discovery should provide a user with not only the raw data and/or the discovered results but also the entire process

in visual forms so that the user can participate in the discovery process, provide heuristics, guide the navigation of search, and interpret the discovered knowledge.

We propose an interactive visualization model, RuleViz, which stresses the human-machine interaction and visual representations during the entire KDD process. The interaction between the user and the machine helps the KDD system navigate through the enormous search spaces and recognize the intentions of the user, and the user easily provides the system with heuristics and domain knowledge and specifies parameters. On the other hand, the visualization of the KDD process helps users gain better insight into the multidimensional data, understand the intermediate results, and interpret the discovered patterns.

3.2 The *RuleViz* Model

As discussed before, the KDD process uses the database along with any required selection, preprocessing, sub-sampling, and transformations to apply data mining algorithms to enumerate patterns and to evaluate the products of data mining to identify the subset of the enumerated patterns deemed as "knowledge" [42]. The KDD process underlines an interactive and user-friendly character of overall activities and involves numerous steps with many decisions being made by the user [53]. Much discussion about the KDD process and its main ingredients are present in the literature [24, 35, 38, 42, 53, 56, 70]. Basically, the KDD process consists of data collection, target data creation, data cleaning, data preprocessing, data reduction, data mining, pattern interpretation, and knowledge application [24]. The KDD process relies on a human-system interaction. It is impossible to envision a fully automated process of knowledge discovery because this process requires that the system possess all domain knowledge and be capable of recognizing

the intentions of the user. Hence, effective human-machine interaction plays an important role in the KDD process. Various visualization tools can be used to provide the interaction facilities and help incorporate the user's intentions and navigate through the enormous search spaces interactively.

In order to construct such interactive systems, we suggest an interactive visualization model, RuleViz, which addresses the following aspects:

- Combine algorithm-based and visualization-based approaches for data mining. As mentioned in the previous section, both algorithm-based and visualization-based approaches have their own advantages and disadvantages, while the RuleViz model attempts to take their advantages and avoid their disadvantages;
- Visualize the entire process of knowledge discovery. There are two different but related ways for visualizing the KDD process, one presenting the intermediate results of algorithm-based process in each step, the other visualizing each step of the KDD process;
- Include the user's perception to guide the KDD process and navigate the search for various tasks, like data reduction, continuous attribute discretization, decision tree construction, threshold setting, etc.;
- Investigate human-machine interactions for the machine to accept the user's intentions and perceptions;
- Develop new visual forms for presenting intermediate results and various kinds of final patterns.

According to the KDD process, the RuleViz model consists of five components: raw data preparation and visualization, interactive data reduction (horizontally

and vertically), visual data preprocess such as missing values handling, numerical attribute discretization and data transformation, pattern discovery like correlation mining, rule induction and decision tree construction, and pattern visualization like neural networks, decision trees, and classification list, shown in Figure 3.1. In this model, the *Raw Data Set* is visualized by using the *Data Visualization Routines.* The user can view the visualized data and decide which *Data Reduction Methods* are appropriate for reducing the raw data due to the resource and effectiveness and cleaning the data due to noise, impreciseness and uncertainty, and the result is the *Reduced Data Set* which usually is the representative portion of the raw data set and contains the most informative data areas. The reduced data are preprocessed before they are used to discover patterns. The *Data Preprocess Approaches* consist of data transformation, data scaling, data normalization, missing values handling, noise removing, continuous attribute discretization, etc. In practice, data reduction and preprocess components can be interchanged and repeated until the *Preprocessed Data Set* is good enough for knowledge discovery. Based on the cleaned and preprocessed data, various *Algorithms for Pattern Discovery* can be used to find knowledge. Finally, the *Discovered Patterns* are visualized by using corresponding *Pattern Visualization Routines.*

At any point of the process of the RuleViz model, the user can instantly observe the progress, determine how to respond the system, and decide the next task and approaches for fulfilling the task. Each component can be repeated until the results are satisfying. In the following sections, we discuss each of the components in the RuleViz model. Since the data visualization, knowledge discovery algorithms, and knowledge visualization were discussed in previous chapters, we will focus on the data reduction and data preprocess components in this chapter.

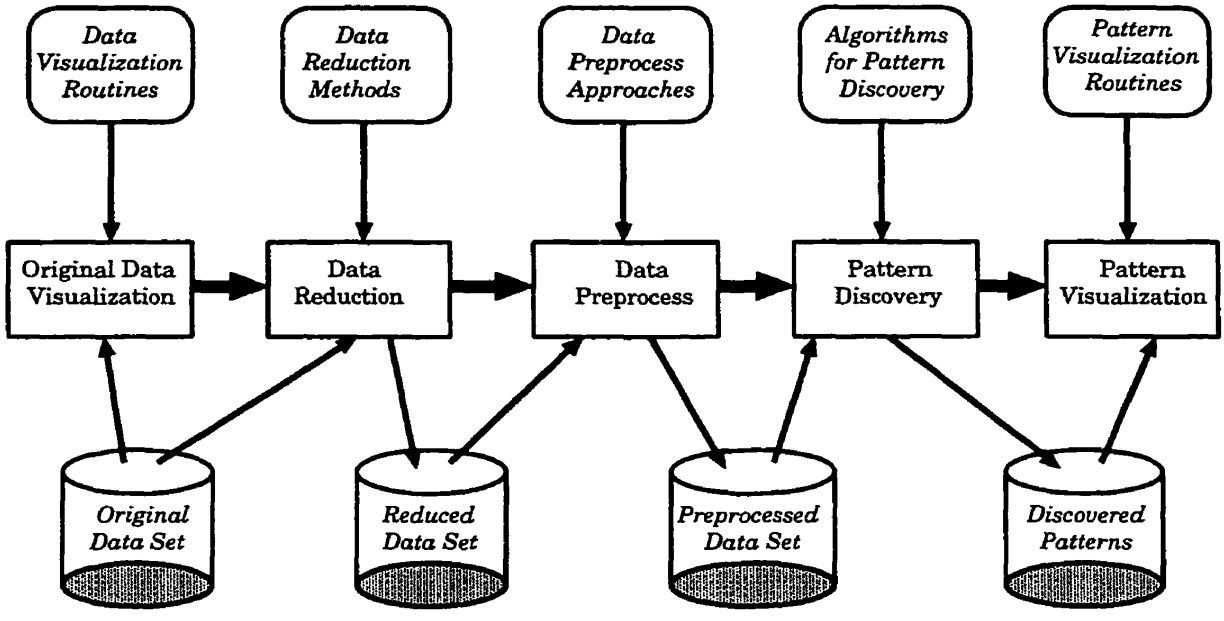


Figure 3.1: The RuleViz Model

3.3 Data Preparation and Visualization

The data preparation and visualization follow the procedure of data visualization (see Chapter 2), and its purpose is to present the raw data to the user in visual forms so that the user can view the data from different angles, perceive the data (features and tuples) distribution, and decide what patterns are interesting and which tools can be used.

The raw data may come from the business databases or other systems. The data preparation is to specify the data source and raw data form. It can be done either manually or automatically. Raw data visualization is to use various data visualization techniques to appropriately display the raw data in some visual form. How the raw data is visualized depends on the patterns that are to be discovered. The goal of this component is to give the user insight that is virtually impossible

to get from looking at tables of output or simple summary statistics. The raw data visualization can tell how the raw data is distributed, where the interesting data are located, and whether the patterns can be observed. The patterns to be searched for might be classification rules or association rules, flat rules or hierarchical rules, etc.

Different patterns may require different visual forms in order for the user to observe and determine the interesting areas. For example, classification rules may require that the focus be on the relationships between the condition attributes and their values and decision attribute and its class labels. The association rules may require the focus to be on the relationships among item subsets for item transaction data sets, or the relationships among the value ranges of different attributes for numerical data sets. The decision trees may require the focus to be on the hierarchy of concepts or the distribution of data in different regions (branches).

According to [98], the goals of data visualization could be one of three aspects: *exploratory analysis*, *confirmatory analysis*, and *presentation* (see Chapter 2). In the RuleViz model, however, we assume the raw data visualization is to present the data in the visual form for the user to view conveniently.

All visualization techniques discussed in Chapter 2 can be used in RuleViz, including geometric techniques, icon-based techniques, pixel-oriented techniques, hierarchical techniques, and graph-based techniques. It must be emphasized that the distance measure between data points when visualized is so important that inappropriate measure will lead the user in a wrong direction or into a mistaken action in the following steps. Usually, the distance measure can be Euclidean, Mankowski, Tschebyschev, and others [42].

3.4 Data Reduction

There are many factors affecting the performance of knowledge discovery systems. One prominent factor is the data [108]. Too much data may cause a KDD system to be unable to handle the data (not only because of algorithm complexity, but due to computer memory capacity), and consequently, nothing can be discovered. On the other hand, too little data can not provide a KDD system (such as a miner of association rules) the sufficient evidence to find anything meaningful and reliable. In addition, most data sets collected from real world applications contain noisy data, which may distract the KDD system and mislead to nonsense conclusions. Thus, the raw data need to be cleaned to not only reduce the size of the data set but remove noise as well.

Generally speaking, data reduction is done to reduce the redundant or insignificant data, or select a small portion to represent the entire set due to time complexity and/or main memory capacity. This is necessary for real-world applications as databases are usually huge. For example, the database in Wal-Mart increases by over 20 million transactions a day [53]. However, many knowledge discovery techniques are very sensitive to the size of data in terms of time complexity and storage complexity. For interactive knowledge discovery, the raw data can be cleaned by the user who may observe the data distribution through the visual form of the raw data and determine which data area s/he is interested in, which attributes are more important than others, and which attribute values are redundant and can be removed.

Many approaches to data reduction have been proposed and implemented for knowledge discovery [78]. Basically, two categories of algorithms have been investigated and developed extensively. One category focuses on the reduction of features

or attributes (vertical reduction) which are used to represent the raw data, called *feature selection* [42, 108]. Feature selection is trying to find the best feature subset from the raw feature set according to a given selection criterion. The objective is to reduce the data dimensionality by selecting important attributes without or with less loss of information behind the data so that the data can be expressed in some succinct formation and the data size is decreased to the extent that the algorithms can efficiently execute in the main memory.

The other category addresses *example selection*, or *tuple selection* (horizontal reduction) which selects or searches for a representative portion of the raw data set that can fulfill a knowledge discovering task as if the whole data set is exploited [42, 78]. The tuple selection can be algorithm-based or visualization-based.

Numerical attribute discretization and concept hierarchy climbing can also be used to reduce the raw data set horizontally. By discretizing the numerical attributes into interval attributes, the tuples that fall in the same hyper-intervals can be combined if these intervals are viewed as individual values. In the RuleViz model, numerical attribute discretization is done in the next component, *data preprocessing*. By climbing the concept hierarchy, feature values can be substituted by their higher level values in the predefined hierarchy of categorical values of this attribute [31, 151].

The outcome of data reduction is a subset of the raw data set which has been cleaned vertically and horizontally, and will be used to be the representative portion of the entire data set. The data reduction can be repeated with different approaches until the selected data subset is satisfactory in terms of the user's intentions or some pre-specified criteria.

3.4.1 Feature Selection

Features refer to attributes, properties, or characteristics. In most real-world applications, the collected data are usually represented as feature-values. This data representation is primitive, convenient, and widely used in pattern recognition, databases, statistics, as well as machine learning and data mining. Unfortunately, real application data sets often contain hundreds or even thousands of features so that it is very difficult to transform such high dimensional data into visual form and impossible, in most cases, to discover useful patterns from the mountains of data.

There are two categories of feature selection methods, algorithm-based methods and visualization-based methods. The algorithm-based feature selection methods can be viewed as a search problem in the feature space to find an optimal subset of features to represent the entire set of features. The visualization-based methods provide the raw data distribution in visual form and take into account the user's perception. The interesting (relevant or important to the task) features are selected through the human-machine interaction. We discuss the algorithm-based feature selection methods, consider some cases in which the visualization-based methods are helpful, and suggest possible combinations of both approaches.

Algorithm-based Feature Selection

Four issues must be considered carefully when one designs an algorithm-based feature selection method: search directions, search strategies, evaluation measures, and stop criteria [2]. We briefly discuss each of them below.

[2]Some articles do not consider *stop criterion* as an independent issue, instead the same as *evaluation measures* [108]. We argue, however, they are two different concepts. The reason is that

Search directions for feature selection

Commonly, the feature space forms a lattice through the set-inclusion relation. In this lattice, the lowest element is the empty set, and the top element is the full set. Between them are the subsets of the features. Two fundamental search directions are *upward* and *downward*. The combination of *upward* and *downward* searches is possible.

1. Sequential upward search: Initially, the optimal feature subset is assumed to be empty. Each feature not in the current optimal subset is evaluated according to the specified evaluation function. The best one is chosen to fill in the subset. Thus the subset is sequentially increased. Each time a new feature is added to the subset, the subset is evaluated and compared with the stop criterion. If the stop criterion is satisfied, then the process ends; otherwise the selection is repeated.

2. Sequential downward search: This search is along the opposite direction with upward search. It begins with the full feature set. Each feature in the current feature set is evaluated according to the specified evaluation method, and the least important one is removed from the current feature set. Thus, the feature set sequentially shrinks. Each time a feature is removed from the feature set,

---

the evaluation measures for the features may be different from the measures of stop condition. For example, the stop criterion may be the number of features. Once the size of the feature subset reaches to the specified number, the search stops. On the other hand, the evaluation measure may be the entropy that a feature contains. In some applications, to avoid too long runtime, the approximate optimal feature subsets are also acceptable. In such cases, some threshold is specified as the stop criterion and the evaluation measures are compared with the threshold to determine whether the search stops.

the set is evaluated and compared with the stop criterion. This process is repeated until the stop criterion is met.

3. Bidirectional search: This is a combination of above two searches. Two searches along *upward* and *downward* respectively proceed concurrently and separately. The search process stops when either one of two searches satisfies the stop criterion or both searches meet at some feature subset in the feature space.

4. Random search: The search direction is not deterministic. Whether adding or removing a feature in each step depends on a random number generator. This search attempts to avoid trapping into local optima, though it is hard to monitor the stop criterion.

Search strategies for feature selection

The size of the feature space depends on the number of the features. Assume the number of features is n, then the feature space size is 2^n. When n is large, it is intractable to find the optimal feature subset. On the other hand, in many real world applications, the optimal feature subset is not only not necessary, but may cause other problems if the size of the optimal feature set is still large enough so that the resource is not sufficient [109]. Therefore, the objective of the feature selection is to find the approximately optimal subsets instead of the optimal ones. To this end, several search strategies have been developed.

1. Complete search: Actually, this is a variation of brute-force search, and similar to the level-wise search for finding frequent itemsets in mining association rules algorithms *AIS* and *Apriori* [3, 4, 6]. The search direction can be upward or downward. For the upward search, it first evaluates each one-feature set,

and chooses the best one. Then it evaluates all two-feature sets, and chooses the best one. Repeatedly, it evaluates all k-feature sets and chooses the best one until the resource is exhausted, $k = 1, 2, \ldots, \leq n$. Finally, the end result is the best feature subset among all best k-feature sets.

2. Heuristic search: To avoid brute-force search, some heuristics can be used to conduct the feature selection. It follows the general principle of heuristic search in artificial intelligence community [143]. The challenging problem is how to design the heuristics to make the result as good as possible. However, heuristic search runs fast because it doesn't need to search all possible paths.

3. Nondeterministic search: It has no definite search direction and always works with random search discussed above.

Evaluation measures for feature selection

In the process of feature selection, the feature with the best or worst evaluation measure is added into or removed from the current feature subset. Therefore, the evaluation measures of the features decide the performance of the selected feature subset, while the search direction and strategy affect the efficiency of feature selection alone. Many evaluation methods have been developed [19, 108, 109, 136]. We briefly review some of them in the following.

1. Information measures: For the given data set with features, the information content conveyed in the data set can be calculated according to information entropy. The information gain from a feature is the difference between the information contents with and without this feature. Generally, the greater gain, the better. Thus, the feature with the greatest information gain is chosen as the best.

2. Distance (dependence) measures: Distance measure is a discrimination method. The distance between the current state and the target state is calculated and the state with smaller distance to the target is considered as better or worse one, depending on the applications. For example, in data classification, the distance can be defined as the class-conditional difference. Assume q class labels, $c_1, c_2, \ldots, c_q$. For feature X, $D(X)$ can be defined as :

$$D(X) = \frac{2}{q(q-1)} \sum_{1 \le i \ne j \le q} |P(X|c_i) - P(X|c_j)|, \tag{3.1}$$

where $P(X|c)$ is the probability of X given class c. We say X is better than Y if $D(X) > D(Y)$.

The dependence measure is actually an opposite measure of distance measures. The mining correlations between features is an example for dependence measure, where the dependence (relevance) of a feature X on the other features can be defined as the average normalized covariance between X and all other features [50]. Since the covariance may be positive or negative, indicating positive correlation or negative correlation, respectively, the more distant from zero the covariance, the better the feature.

3. Correctness measures: This measure is based on the comparison of the results obtained with the selected features with the real application. If the result obtained with the selected features is very close to the actual situation, then the selected feature subset is good. For example, in data classification, the correctness can be replaced by prediction accuracy; in the discovery of associations, the rule *support* and *confidence* can be used as the correctness measure. The problem of using this measure is that before the features are selected, at least one pattern discovery algorithm must be developed and used to measure the correctness of the feature subsets. For example, to select fea-

tures for a classifier, a known classifier construction algorithm such as Naive Bayesian Classification or C4.5 decision tree construction is used to construct a candidate classifier based on a candidate feature subset. The accuracy of these classifiers are compared, and the feature subset corresponding to the most accurate classifier is chosen [108].

4. Other measures: Other feature measures are possible. Data consistency, data redundancy, data noisy, etc. can be also used to measure features. For example, *data inconsistency rate* is used as consistency measure for data classification [109]. Two tuples are said inconsistent if they have the same values for all conditional attributes but are labeled as different classes. Removing a feature will increase the number of inconsistent paired tuples. The increment (or increment ratio) can be applied to select features. The bigger increment means the better feature (the more discriminant power).

Stop criteria for feature selection

The last issue for algorithm-based feature selection is the stop criteria. Summarizing the feature selection systems, we can divide the stop criteria into the following categories.

1. Evaluation measure threshold criterion: In this category, the stop criteria are usually specified as some threshold. When the evaluation measure of the current feature subset is greater or less than the threshold, the process stops.

2. Feature subset size criterion: In real applications, the data set may contain hundred or thousand of features, making the data set too large to be stored in computer memory and degrading the efficiency of knowledge discovery. Thus, the main purpose of feature selection is to reduce the data set size instead of

finding the optimal feature subset. If this is the case, the number of selected features is limited by the memory capacity. Therefore, when the number of selected features is large enough with respect to the computer capacity, the selection stops.

3. Runtime criterion: This is a simple criterion. When the runtime surpasses the pre-specified value, the selection stops and the current best feature subset is returned.

These stop criteria are often mixed to form a compound criterion. For example, the evaluation threshold and feature subset size criteria can be used together. As long as one of them is satisfied, the feature search terminates.

Visualization-based Feature Selection

The other approach to feature selection is the visualization-based method, which attempts to include humans perception into the system through the human-machine interaction. By looking into the distribution of the raw data which are visualized in the first step, the user may have a feeling indicating which portion of the data is significant and which features are relevant to the task. Of course, how to judge whether a feature is significant depends on the knowledge discovery task and the data visualization techniques used. We examine the following cases.

Case 1: *Assume the task is to induce classification rules, and the data visualization technique used in the first step is parallel coordinates [91]. In the extreme case in which all polylines indicating tuples go through the same value of a feature, this feature is trivial and is worthless of consideration.*

Case 2: *Assume the task is to discover exception rules [154, 149], and the hierarchical visualization technique is used. Then, the features with unevenly distributed*

values are usually more significant than those with evenly distributed values, because for the evenly distributed values, there is less chance that the exception occurs.

Case 3: *Assume the task is to construct a decision tree and the data visualization technique used in the first step is* circle segments *[13]. Then, generally, the features with the segments in which the class labels are randomly distributed are less significant than the features with the segments in which the class labels are distributed aggregately.*

3.4.2 Tuple Selection

The purpose of tuple selection is to select a small representative portion of the raw data such that the discovered results are good enough as if the whole raw data were used. Most existing tuple selection approaches are algorithm-based, meaning the tuples are either calculated according to some evaluation measure and data distribution to decide whether they are chosen or just randomly sampled. Another tuple selection approach is visualization-based, in which the raw data are distributed on the picture by visualization and the user can pick up any interesting areas as the selected data.

Algorithm-based Tuple Selection

Existing algorithm-based tuple selection techniques can be divided into two main classes: *parametric techniques* and *non-parametric techniques* [16]. The former assumes a parametric model for the data distribution, like *Gaussian* or *exponential* distributions, and then estimates the parameters of the model. After the parameter estimate is obtained, the typical data, with respect to the parametric model, can be drawn according to their typicality measure, which forms the representative

portion of the raw data [78]. Actually, learning algorithms such as decision trees, neural networks, and other classification and/or clustering algorithms can operate on the typical sample. In order to use parametric techniques, the form of the data distribution density function must be assumed. In real world applications, however, this is not the case.

Non-parametric tuple selection techniques do not assume any model for the data. The raw data can be randomly sampled or sampled based on the use of criterion functions. We list some of techniques below.

- Divide-and-Conquer methods: This method is to divide a whole data set into small parts, and each part is used to discover patterns. Breiman considers the problem that the raw data is too large to hold in memory of any currently available computer, and proposes an approach to solving the problem [25]. According to this approach, the data set is divided into many small subsets such that each subset can be held in the main memory. A classifier can be learnt based on each data subset. Thus, a set of classifiers is obtained. Finally, all classifiers learnt from the data subsets are combined to form an aggregation classifier that has higher accuracy [25]. Savasere et al. propose a partition method for discovering large itemsets in mining association rules [144]. The data set is partitioned into sections small enough to be processed in memory. For each data section, the locally large itemsets are discovered as the candidates which are further evaluated in memory.

- Density-estimate methods: Fukunaga and Mantock present a non-parametric approach to tuple selection [63]. The technique is iterative and based on the use of a criterion function and *nearest neighbor* density estimates.

- Database-oriented methods: Many techniques of tuple selection from database

perspective can be used in data reduction for knowledge discovery, including singular value decomposition, discrete wavelet transform, linear regression, histogram clustering, index trees, etc. [16].

- Continuous-discretization methods: When a continuous attribute is discretized, the values that fall in the same interval can be treated as the same. Hence, once all continuous attributes are discretized, many tuples will become the same, and only one is needed and sampled. Similarly, for a nominal attribute, if there exists an hierarchy between its values, then climbing the hierarchy causes different tuples identical. The redundant tuples can be removed so that the data is reduced.

- Sample-based methods: Sample techniques can be used to select tuples [155], including incremental sampling, average sampling, strategic sampling, sampling by adjusting prevalence, and random sampling.

Visualization-based Tuple Selection

Visualization-based tuple selection is an interactive process between the user and the system. The raw data must be transformed into a visual form so that the user can directly view the data distribution. According to his/her perception, s/he chooses the interesting data areas by using the convenient interaction tools. Figure 3.2 shows two cases for visualization-based tuple selection. In Figure 3.2 a), a data set to be classified is visualized and each tuple is drawn as its class label. Totally, there are three classes in this data set. Different classes aggregate in different areas with few exceptions. One can use an interaction tool to specify the areas for each class, thus divide the data set into three sections, each containing the most part of a class tuples and being used to learn the corresponding class description. On the

other hand, Figure 3.2 b) shows another case in which the knowledge discovery task may be classification or mining correlation. The raw data aggregate in two areas. One can use interaction tools such as "rubber band" to specify each of them, and once the results are discovered based on each area, the final patterns can be formed by combining the separate results obtained from different areas. The points not in any interesting areas are considered as *outliers* and not used to find patterns.

3.5 Data Preprocess

Data preprocess is a sequence of operations that convert the raw data to the target format for the data mining algorithms. It includes several aspects such as data normalization, data scaling, data transformation and feature extraction, handling the missing attribute values, numerical attribute discretization, and so forth [42].

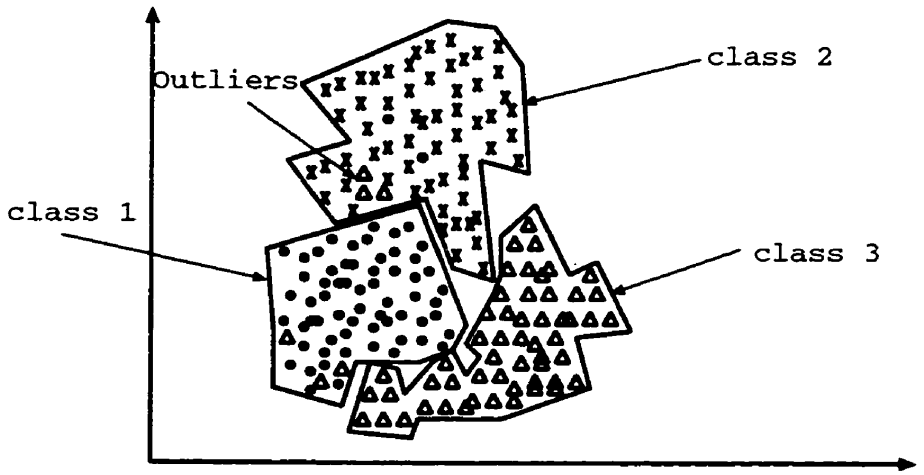
3.5.1 Data Transformation and Feature Extraction

Data transformation is to transform the current data into appropriate format which can be used directly by the specific knowledge discovery algorithms. When data are visualized or processed for knowledge discovery, they may require scaling and normalization. A simple normalization formula is:

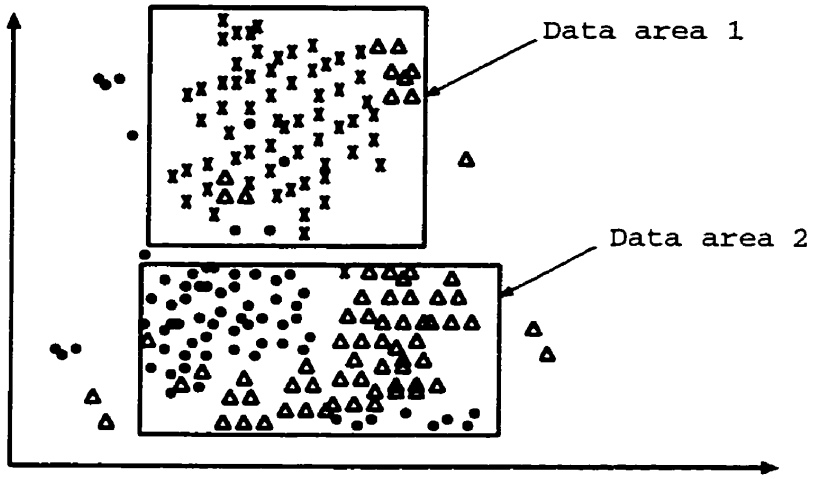
$$x_{normalized} = \frac{x - \mu_x}{\sigma_x}, \tag{3.2}$$

where μ_x and σ_x are the mean and standard deviation of attribute x, respectively.

For simplifying the computation, one may need to translate the data so that the centroids (with respect to some dimensions as well as overall centroid) are located at the origin.



a) Partitioning data based on visualization



b) Two interesting data areas based on visualization

Figure 3.2: Tuple Selection Based on Visualization

Another transformation is *feature extraction*, which transforms a high dimensional data set into a low dimensional data set. However, feature extraction is different from feature selection. The latter selects the most significant or important features or an optimal feature subset, according to the specified criteria, to replace the full feature set to represent the data; while the former finds the linear independent features and transforms the dependent ones into independent ones. Feature extraction not only reduces the data dimensionality, it also transforms the data themselves. From the visualization perspective, each feature being viewed as a dimension, high dimensional data sets are often more difficult to be visualized than low dimensional data sets. To obtain low dimensional data without or with less loss of the information contained in the raw data set, the dependent features are removed, and the axes consisting of independent features can be rigidly rotated along different vectors so that all axes are orthogonal.

The commonly used feature extraction method is the Principle Component Analysis [42, 59, 108]. This method is based on statistics and results in a linear Karhunen-Loeve transformation. The linear Karhunen-Loeve transformation is produced in terms of the covariance matrix of the data, and its eigenvalues and corresponding eigenvectors.

Assume the raw data set is characterized by an n-dimensional feature space. N data points are denoted as $x_i, i = 1, 2, \ldots, N$, and x_i is a n-dimensional vector, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^T$. The raw data can be represented as a $N \times n$ matrix X, whose elements are $x_{ij}, i = 1, 2, \ldots, N; j = 1, 2, \ldots, n$. Estimate the mean value for each dimension based on the data set,

$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_{ij}, \quad j = 1, 2, \ldots, n. \tag{3.3}$$

The square $n \times n$ dimensional covariance matrix is defined as

$$\Sigma = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^T, \tag{3.4}$$

where $\mu = (\mu_1, \mu_2, \ldots, \mu_n)^T$ is the mean vector.

The intrinsic characteristic of the given data set can be found as a set of n eigenvalues λ_j and the corresponding eigenvectors e^j by solving the eigenvalue problem:

$$\Sigma e^j = \lambda_j e^j, \quad j = 1, 2, \ldots, n. \tag{3.5}$$

Since the covariance matrix Σ is symmetric and real-valued, its eigenvectors are orthogonal, which means $(e^j)^T e^k = 0, 1 \leq j \neq k \leq n$, and $(e^j)^T e^j = 1, j = 1, 2, \ldots, n$. The eigenvalues are arranged as in descending order such that

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n, \tag{3.6}$$

and the corresponding orthonormal eigenvectors form the square $n \times n$ matrix:

$$E = [e^1, e^2, \ldots, e^n], \tag{3.7}$$

with the jth column representing the eigenvectors e^j corresponding to the eigenvalue $\lambda_j, j = 1, 2, \ldots, n$. The most dominant eigenvalue of the covariance matrix Σ is λ_1, and the most dominant eigenvector of the matrix E is e^1. Assume our task is to reduce the n-dimensional data into m-dimensional data ($m < n$), then simply take the first m principle components corresponding to the first m dominant eigenvalues. The corresponding eigenvectors are used as the transformation $m \times n$ matrix W:

$$W = (e^1, e^2, \ldots, e^m)^T. \tag{3.8}$$

Thus, the raw data X is transformed to the new data set Y:

$$Y = WX, \tag{3.9}$$

which minimizes the mean least square errors [42].

In real applications, the number of non-zero eigenvalues may be large. If this is the case, we can take a small m and m largest eigenvalues, and ignore others. Thus, the n-dimensional data are transformed into m-dimensional data with less loss of information contained in the raw data set. The lost information can be measured as the ratio of the sum of ignored eigenvalues to the sum of all eigenvalues:

$$ratio_{loss} = \frac{\sum_{i=m+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i}. \tag{3.10}$$

When the ratio is sufficiently small (e.g. < 0.05), the m new features can replace the n old features to represent the data set approximately.

The Principle Coordinate Analysis is related to the Principle Component Analysis [68]. It considers another problem: Given a set of values which are known or assumed to represent the squared distances within a set of N points in some Euclidean space, how can one build the coordinates of the points, including determining the number of dimensions? We summarize this method as three stages, finding squared Euclidean distance in the raw data space, transforming the squared distance matrix into an inner product matrix, and transforming the raw data set into a new data set.

- Finding squared Euclidean distance matrix:

  Given a data set of N tuples with n features (n dimensions), denoted $t_i = (y_{i1}, y_{i2}, \ldots, y_{in})$, $i = 1, 2, \ldots, N$. The Euclidean distance between any two points t_i and t_j can be simply calculated:

$$d_{ij} = \sum_{k=1}^{n} (y_{ik} - y_{jk})^2, \quad i, j = 1, 2, \ldots, N. \tag{3.11}$$

- Transforming the squared distance matrix into an inner product matrix:

Assume the new data set has m features (m-dimensional data set) with $m < n$, and $t_i = (x_{i1}, x_{i2}, \ldots, x_{in}), i = 1, 2, \ldots, N$, where $x_{ik} = 0, k = m+1, m+2, \ldots, n$. Before m is determined, we use n to take its place. For simplicity, assume $\sum_{i=1}^{N} x_{ik} = 0, k = 1, 2, \ldots, n$, since we expect the centroid of the new data set to lie at the origin of the coordinate system.

With this assumption in mind, let A be a $N \times N$ matrix, $A = (a_{ij}), i, j = 1, 2, \ldots, N$, where

$$a_{ij} = \sum_{k=1}^{n} x_{ik} x_{jk}, \qquad i, j = 1, 2, \ldots, N. \tag{3.12}$$

It can be shown that

$$a_{ij} = -\frac{1}{2}(d_{ij} - d_{i.} - d_{.j} + d_{..}), \quad i, j = 1, 2, \ldots, N, \tag{3.13}$$

where

$$d_{i.} = d_{.i} = \frac{1}{N} \sum_{j=1}^{N} d_{ij}, \qquad i = 1, 2, \ldots, N, \tag{3.14}$$

and

$$d_{..} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij}. \tag{3.15}$$

- Determining new coordinates:

  Because A is a real symmetric matrix, it can be diagonalized as

  $$A = XX^T = VWV^T, \tag{3.16}$$

  where W is a $n \times n$ diagonal matrix, and V is a $N \times n$ orthogonal matrix with columns being the eigenvectors corresponding to the eigenvalues of W. Let $V = (\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n})$ and the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$ are arranged in descending order, meaning $W = (w_{ij})$, with $w_{ii} = \lambda_i$ and $w_{ij} = 0, i \neq j, i, j = 1, 2, \ldots, N$, and

  $$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N. \tag{3.17}$$

Thus we have

$$X = VW^{\frac{1}{2}} = (\sqrt{\lambda_1}\mathbf{v}_1, \sqrt{\lambda_2}\mathbf{v}_2, \ldots, \sqrt{\lambda_N}\mathbf{v}_N). \tag{3.18}$$

Take m as the minimum integer $k < n$ such that $\lambda_k > 0$. The raw n-dimensional data set is reduced to a m-dimensional data set.

3.5.2 Missing Values Handling

Many solutions to the problem of knowledge mining from a data set with missing values have been proposed [43, 71, 123, 136]. Generally, the missing values can be either viewed as **unknown** or as inapplicable, or both. We summarize some of them as follows.

- Ignore the tuples with missing value(s): The simplest method is just ignoring the tuples with any missing values. All tuples with missing values are removed from the raw data sets and the remaining tuples are used in the following steps. This method is easy to process, but the information contained in the tuples with missing values will be lost.

- Treat missing values as a special attribute value: In this method, any missing values are assigned to a special value **unknown** as if **unknown** is a new value of all attributes and dealt with in the same as other normal values. This method is also simple, but since all **unknown** values are viewed as the same, for an actual unknown value this assignment might cause confusion.

- Replace the missing values with the most common values: This is a more complex approach that counts the number of occurrence of each distinct values and the most common value (with the largest number of occurrence in the

data set) is chosen to replace the missing value. There are different versions to define what the *most common* means. The simplest method, for each attribute, is to select the value that occurs most often as the most common value of the attribute without paying attention to the correlations between attributes. The CN2 induction algorithm employs this definition [43]. The complicated definition also considers the relationships between attributes. For example, in data classification, the most common value can be defined based on the class labels to be the value that occurs most often among the tuples that have the same class label as the tuple with the missing value.

- Split the tuple with missing value(s) into many tuples each of which has a distinct possible value in the attribute domain. This method considers all possibilities of the missing values and includes them in the data set [71]. For example, assume tuple t has two missing values of attributes A and B, with domains of N and M values, respectively. Then t is split into $N \times M$ tuples, each of which has a distinct value of A and a distinct value of B. A variation of this method can be seen in C4.5 [136], which uses a Bayesian formalism to determine the probability distribution of the missing value over the possible values from the domain, and either choose the most likely value or divide the example into fractional examples, each with one possible value weighted according to the probabilities determined. Another split method is related to the decision attribute in classification, similar to the definition of the *most common value* with respect to the class labels.

- Other methods: Other approaches to dealing with missing values include the *event-covering* method, fuzzy methods, and hybrid method. The event-covering method selects a subset of statistically interdependent events in the

outcome space of attribute-value pairs, disregarding whether or not the attributes are statistically independent [41]. The fuzzy method uses membership functions to assign memberships to each possible values. The hybrid method combines several approaches and for each missing value votes which method is the best and chooses the best method to handle the missing values.

3.5.3 Numerical Attributes Discretization

As mentioned previously, an important task of data preprocess is to discretize numerical attributes. Discretization is a process of transforming a continuous attribute value range (usually containing infinite number of possible values) into a finite number of intervals, each of which can be dealt with as a discrete value. Most approaches for discovering knowledge discretize the numerical attributes into intervals and then treat the discretized attributes as categorical ones and intervals as categorical values. A discretization process consists of two general steps. The first is deciding the number of discrete intervals, and the second is determining the boundaries of each intervals. The number of intervals has a profound effect on the mining efficiency and accuracy, and is often pre-specified by the user, though some approaches can automatically determine it according to the data distribution. Ching et al. [40] suggest that for classification problems, the number of intervals for each continuous attribute be $\frac{N}{3q}$, where N is the number of training examples and q the number of class labels.

There are many approaches to determining the interval boundaries, and these approaches are either used as a part of the mining algorithm itself, or performed as a preprocess step. The existing approaches to discretizing continuous attributes are all algorithm-based. So, we first discuss the algorithm-based discretization

methods, and then propose some visualization-based discretization approaches.

Algorithm-based Continuous Attributes Discrtization

Cios et al. [42] divide the existing algorithm-based discretization scheme into *unsupervised vs. supervised, global vs. local,* or *static vs. dynamic* based on the criteria that the discretization algorithms use. Supervised discretization uses the decision attributes while unsupervised discretization doesn't. Global discretization partitions each attribute into intervals regardless of other attributes, while local discretization partitions attributes based on a portion of data set that is partitioned from the raw data set after other attributes discretization. Finally, the static approach performs discretization only once, while the dynamic discretization repeats partitions according to the current available data (e.g. in decision tree construction, the partition is based on the current node).

- *Equi-length partition* [61, 76]: This is the simplest partition method, which finds the minimum and maximum values for each numerical attributes and divides the range into a number of user-specified equal width discrete intervals. The boundary points are the two end points, and the midpoints

  $$(maximum - minimum)/(number\ of\ intervals) \times i + minimum,$$

  where $i = 1, 2, \ldots$, the number of intervals.

- *Equi-depth partition:* This method sorts the values of each attribute in the data set and then divides these values into the user-specified number of intervals such that each interval contains the roughly same number of sorted attribute values (the number of tuples falling in the interval) along the ordering of the data. The algorithm KID3 uses this partition [133]. For a given bucket width W, the sorted values are initially divided into approximately

equal buckets, each of which has low and high bounds and a count of values between the bounds. If a bucket has the same last value as the first value of next bucket, then the next first value is reassigned into the previous bucket until different buckets have different values. If a bucket has the first value less than the last value and the count of the last value is W or more, then the bucket is split in two, one containing all the last value, the other containing all other values.

Another equi-depth approach, proposed by Srikant and Agrawal [152], for mining numerical association rules is based on the measure of the partial completeness over itemsets which handles the amount of information lost by partitioning. The intuition behind this measure is that the information lost due to partitioning should be as small as possible. The information lost is measured by the distance between the set of rules obtained from the raw data and the set of rules obtained from the partitioning.

- *Distance-based partition*: This method partitions the attribute value range by identifying and combining clusters of data points. The approach proposed by Miller and Yang [117], also for mining numerical association rules, consists of two phases, identifying clusters and combining clusters to form rules. This approach addresses the following problems: the measure of the quality of an interval which reflects the distance between the data points in the interval (intra-distance), and the distance between the data points in the adjacent intervals (inter-distance), the definition of an association rule which ensures that items in the antecedence will be close to satisfying the consequence, and the measure of rule interests which should reflect the distance between the data points.

Another *distance-based partition* is called the K-means algorithm [42], which is based on the minimization of the sum of the squared distances of all tuples in a cluster to the cluster centroid. Initially, randomly choose K points as the centroids of K clusters (e.g. use equi-length method). Then the values of the attributes occurring in the raw data set are distributed among the K cluster centers based on the minimal squared distance criterion. Third, the K new cluster centers are computed such that for each cluster the sum of the squared distances from all points in the same cluster to the new cluster center is minimized. Finally, check if the K new cluster centers are the same as the previous ones. If not, repeat above procedure to update the cluster centers; if yes, the approximately optimal cluster centers are found and the boundary points are the midpoints between any two adjacent cluster centers plus the two end points.

- *Entropy-based partition*: This approach is widely used and has many different implementations [42]. This method discretizes the attributes based on the maximal entropy of the partition. Fayyad and Irani develop a discretization algorithm for decision tree construction which improves the efficiency of ID3 algorithm by partitioning continuous attributes into binary attributes using class-entropy minimization heuristic [51]. Their another discretization algorithm uses the Description Length Principle together with a heuristic for a recursive entropy minimization as a stopping criterion to control the number of intervals generated [52]. The discretization algorithm built into the C4.5 learning algorithm is also based on entropy-based partition [136]. For a given continuous attribute F, sort the training examples on the values of F, and then split the values into two intervals based on a threshold α, one containing points with which $F \leq \alpha$, and the other containing points with which $F > \alpha$.

To find the threshold α, the *information gain* for each value occurring in the training examples is examined and compared, and the value corresponding to the largest information gain or gain ratio is chosen as the threshold to discretize the feature F. An and Cercone develop an entropy-based discretization algorithm EDA-DB and build it into the ELEM2 learning system [12]. EDA-DB first divides the value range of the attribute into several *big* intervals and then selects in each interval a number of cut-points based on the entropy calculated over the current entire data set. The number of cut-points selected for each interval is determined by estimating the probability distribution of the boundary points over the data set. The maximum number of selected cut-points is determined by the number of class labels and the number of distinct observed values for the continuous attribute.

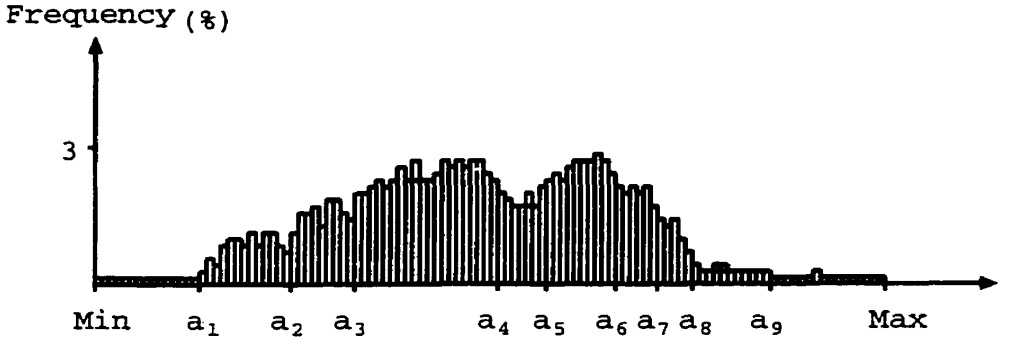
Visualization-based Continuous Attributes Discretization

The algorithm-based approaches for discretizing continuous attributes introduced above show that the distribution of attribute values plays an important role in determining the number of intervals as well as the cut-points (interval boundaries). Thus we argue that visualizing the distribution of attribute values must be helpful for the user to figure out and choose the number of intervals and the cut-points. Consider an example, in which the frequency histogram of a continuous attribute is shown in Figure 3.3. The range of this attribute values is from Min to Max. The frequency of each value that appears in the data tuples is drawn as a bar. From this frequency histogram, the user can look into the values distribution and determine the number of intervals according to the distribution. Figure 3.3 a) shows that the user determines the number of intervals as 10, and decides the cut-points at $a_1, a_2, \ldots, a_9$, plus the two end points Min and Max. Figure 3.3 b)

illustrates another visualization-based discretization, where the user also determines 10 intervals, but chooses the boundary points in terms of equi-depth scheme. To this end, the user first observes the frequency histogram and estimates the cut-points, and then selects the corresponding points to specify the boundaries.

For data classification, an approach for discretizing continuous attributes based on visualization with respect to the class labels is shown in Figure 3.4, which illustrates the visualization and discretization of a continuous attribute. First, the attribute values appearing in the data tuples are sorted in ascending order. Repeated values are allowed. Since each value must be in a data tuple and hence corresponds to a class label [3], we represent class labels as different shapes (icons) or render them in different colors, and then draw the sorted values with the representation of the corresponding class labels. In Figure 3.4, there are totally five classes which are labeled as $1, 2, 3, 4$, and 5, respectively. The attribute values are drawn one by one, starting from the first pixel (or cell) at the left-bottom of the drawable area, from left to right and from bottom to top, and ending at the right-top. It is apparent that five cut-points, d_1, d_2, d_3, d_4, and d_5, are straightforward, since the tuples corresponding to the attribute values between d_1 and Min fall in class 1, the tuples corresponding to the values between d_1 and d_2 belong to class 2, and the tuples corresponding to the values between d_2 and d_3 or between d_5 and Max are in class 5. The tuples corresponding to values between d_3 and d_5 are not deterministic, but between d_3 and d_4 are either class 3 or class 4, while between d_4 and d_5 are either class 1, 2 or 3. From this observation, it is reasonable to set d_4 to a cut-point, and the intervals $[d_3, d_4]$ and $[d_4, d_5]$ can be discretized further by using other visualization techniques or algorithm-based approaches.

[3]We assume no missing values in the data set, and no noise labeling. An attribute value may appear in more than one tuples with different labels.



a) Discretizing according to value occurrence

b) Discretizing according to equi-depth

Figure 3.3: A Frequency Histogram for A Continuous Attribute

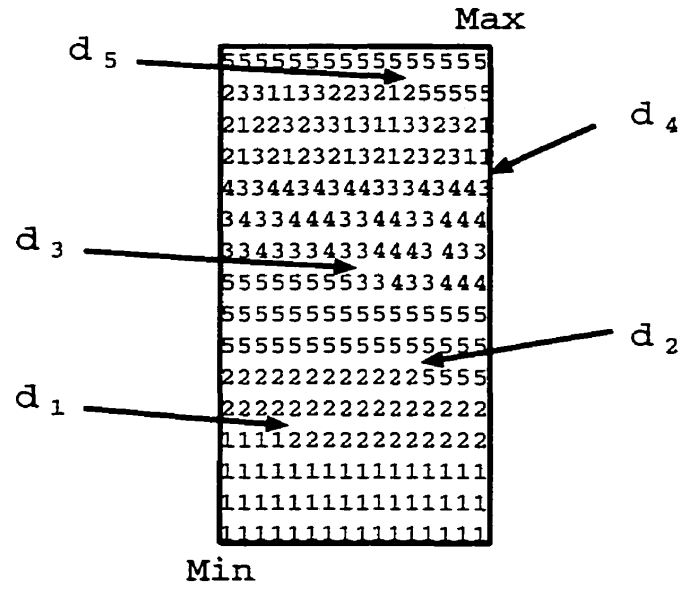


Figure 3.4: Visualizing the Sorted Attribute Values with respect to the Class Labels

3.6 Pattern Discovery

Pattern discovery is the task of data mining, an important step of the KDD process, which searches for patterns of interest in a particular representation form.

Pattern discovery can be either *algorithm-based* or *image-recognition-based.* Algorithm-based pattern discovery approach stresses the logic and/or statistics reasoning and attempts to find the patterns behind the data mathematically. Decision tree techniques [136], neural networks [105], statistics methods [125], association rule mining algorithms [4, 6, 117], rule induction [11], and case-based reasoning [123] belong to this category.

The *image-recognition-based* approach for finding patterns from large data sets is to directly use the images that visualize the preprocessed data [61, 77]. The characteristics such as shape, color, brightness, etc. of the image provides us diverse

information which can be developed to discover patterns. This approach depends on the visualization techniques and the visualized results.

3.7 Pattern Visualization

The last component is pattern visualization which is to visualize the patterns achieved in pattern discovery on the screen. Since different patterns may have different characteristics, they must be drawn with different techniques. A collection of visualization routines should be developed to adapt to different patterns to reflect their characteristics. For example, a tree structure can be used to represent hierarchical concepts. Icon shapes and colors which change with sliders can be used to represent the associations between attributes. Animation and fluid flow can be used to show the evolution of attributes.

For a given preprocessed data set, the user may specify different pattern discovery algorithms to find knowledge and thus obtain the different patterns. The pattern visualization component provides the user an opportunity to observe the final results and compare these patterns discovered by different algorithms.

In a complicated implementation of the RuleViz model, pattern discovery and visualization components can be repeated. One can first specify a pattern discovery algorithm to get a set of patterns, and visualize these patterns, and then choose another discovery algorithm to find a new set of patterns and visualize them. Two sets of patterns can be compared through the visualization so as to interpret the results.

Chapter 4

Implementation Issues of the *RuleViz* Model

4.1 Introduction

RuleViz is an abstract model for visualizing the entire KDD process. In the previous chapter, we discussed its components and techniques that can be exploited in each component. However, there is still a long way to go to implement practical interactive visualization systems for knowledge discovery, though the RuleViz model provides us with a methodology of doing so.

Before we implement visual KDD systems according to the RuleViz model, we consider some general implementation issues independent of applications, such as how to explore each component, how to communicate between components, and how to allocate functionality between the user and machine. Then we choose three knowledge patterns, including numerical association rules, classification rules, and decision trees, as our discovery tasks, and implement four interactive systems to

visualize the discovery process of these patterns in the following chapters.

4.2 Implementation Issues

The RuleViz model provides us with a guideline of designing an interactive visualization system for knowledge discovery. To implement the RuleViz model, however, many decisions need to be made and some details should be refined, such as choosing techniques for each component, selecting language and workspace for implementation, deciding data and view transformations, allocating interactions between the user and machine, including human perceptions, developing discovery algorithms, exploring visualization spaces and visual forms for raw data and final results, etc. because there exist many methods for each part, and different methods have different features and can adapt to different tasks. In this section, we summarize and discuss these issues for implementing interactive visualization systems.

4.2.1 Choosing visualization spaces

The first decision in implementing a system with the RuleViz model is how to use the visualization space. In general, the visualization of the data is based on the display screen which is currently a two-dimensional device. The visual structures may be 1D, 2D, 3D and 3D+time[1] [21, 32]. One-dimensional visual structures are simple and easy for implementation but have weak representation power, and are typically used as part of larger visual structures, for example, a slider for control, a list for timeline or lifeline. The popular debate is about 2D versus 3D presentations

[1]Three dimensions plus time as the fourth dimension are usually represented by animated three-dimensional pictures.

[141, 157], though 3D+time and other higher dimensional visual structures exist. On one hand, the proponents of 2D presentations argue that the display screen is two-dimensional and human vision perceptions are based on seeing only a 2D projection of the 3D world. Usually, 2D pictures are not only faster on computers but also are familiar and straightforward to the user, as well as easy to interact. On the other hand, the promoters of 3D presentations argue that the real world is three-dimensuinal and human experience and perception come from this space. Additionally, three-dimensional space provides much more room for the data and more realistic pictures.

We believe that this debate will continue for a long time, depending on the development of display devices and control widgets. The tradeoff is possible. In the implementation of the RuleViz model with the current devices and technologies, we suggest that two-dimensional visualization be preferred for the raw data, data reduction and preprocess, while two-dimensional and three-dimensional visualizations be appropriate according to the applications. The reason is that the visualizations for raw data, data reduction and preprocess are for the purpose of data mining (pattern learning) and contain a lot of interactions between the user and machine. For interactive feature selection, tuple selection, continuous attribute discretization and so on, it is difficult in 3D pictures for the user to specify the interesting features, data areas, and cut-points due to fewer 3D control widgets. The user may be confused sometimes about whether the interesting data subset is an area or a volume when s/he picks up an area on a three-dimensional picture. For the final results, no more interactivity is needed and the more realistic pictures are preferred, therefore the three-dimensional visualization is better, but complex.

4.2.2 Choosing Visualization Techniques

The second decision in designing a visualization system for the KDD process is whether to choose the existing visualization techniques or to develop new techniques for each component in the RuleViz model. In general, different visualization techniques have different features and can be used for different data sets and tasks. The nature and characteristics of a data set can be revealed by many visual forms, while a visualization technique can be adapted for various purposes. In another word, the questions that a user wants to ask require different presentations for the same data sets. It should be addressed that different visualization techniques can be used in different components in the RuleViz model, but they must be consistent such that the user can easily understand what is going on and conveniently react with the intermediate results between different components [79, 80]. On the other hand, when choosing the techniques for visualizing the raw data, data reduction, and data preprocess, it must be kept in mind that they serve for discovery tasks, not for obtaining beautiful pictures.

There are three decisions to be made for choosing visualization techniques for different components:

- Choosing raw data visualization techniques: The raw data visualization should focus on the revealing of the data values and attribute distributions, attribute correlations, interesting data areas, etc. Therefore, at a glance, the user can have a sense of the data so as to decide the following actions (reducing data and transforming data). Generally speaking, data visualization could be task-driven or data-driven [32]. In the RuleViz model we strongly suggest the task-oriented raw data visualization. Here, the task includes the raw data visualization task (giving the user a feeling) and the discovery task (knowledge

pattern).

- Choosing knowledge visualization techniques: The discovered knowledge is the final result that we look for. Some knowledge patterns have their own natural structures. For example, decision trees have a tree structure, while neural networks have a net structure. However, most of patterns are in rule forms (e.g. classification rules, attribute association rules, characteristic rules, sequential rules) and lack natural visual structures.

- Choosing visualization techniques for data reduction and data preprocess: When designing data reduction and preprocess components, two decisions need to be made: whether visualization-based or algorithm-based reduction and preprocess are exploited, and what interactive visualization and control widgets should be provided. If algorithm-based method is chosen, the system should provide the user with the interface of parameter input (e.g. sliders, buttons, check boxes, choice controls, lists). On the other hand, if the visualization-based method is chosen, more complex interaction functions are needed, for example, selecting interesting data areas by using a "rubber band" approach, picking correlated features by clicking mouse, and specifying cut-points for continuous attributes by moving sliders.

4.2.3 Choosing Implementation Methods

A KDD process consists of many algorithms[2], including reduction algorithms, handling missing values, discretizing continuous attributes, mining (learning) algorithms, and so forth. All approaches can be algorithm-based as well as visualization-based. For the algorithm-based approach, only the results are displayed. For

[2]Here, algorithms refer to mathematical analysis algorithm, not to visualization algorithms.

example, if the discretization algorithm of continuous attributes is an entropy-based algorithm, the cut-points found by the algorithm are visualized after the algorithm ends or as the algorithm proceeds [79]. While for the visualization-based approaches, the algorithm process should be visualized. For example, if a visualization-based discretization algorithm of continuous attributes is exploited, the process of specifying the cut-points one-by-one and therefore its effects should be displayed. The visualization-based approaches can be interaction-based or image-based. Interaction-based approaches[3] provide the user with rich control tools to operate on pictures, and the system only needs to response the user. On the other hand, image-based approaches are similar to algorithm-based approaches but they analyze the images obtained by visualizing the data instead of the data set itself.

The combination of algorithm-based and visualization-based approaches is possible. Different approaches can be used in different components, and in the same components different approaches can also be explored, which depends on applications. For example, handling missing values is difficult to implement based on interaction, while discretizing continuous attributes might be easy in some data sets. Therefore, in the data preprocess component, we can employ the algorithm-based approach for the former and the interaction-based approach for the latter.

4.2.4 Traversability and Navigability

For very large amount of data, it is impossible to accommodate all data items in a relatively small screen. Therefore, traversal and navigation among the data are needed. For example, the user might have window centered on a particular

[3]Interaction can be active or passive. In the active interaction, the user activates the system and the system responds to the user, while in the passive interaction, the user only answers the questions posted by the system.

current item, and scroll up or down the items by clicking on an item at the top or the bottom of the window or moving a slider up or down. Usually, efficient view traversibility and strong navigability are required [64]. By *view traversal* we mean the iterative process of viewing, selecting something seen, and moving to it, to form a path through the logical data structure which can be characterized by a logical structure graph and used to organize the data items. Each data item is represented as a node in the logical structure. Efficient view traversability requires a small number of out-going links of nodes and short paths between pairs of nodes.

The *view traversal* concerns the logical structure itself and ignores the decision where to go next. On the other hand, the *view navigation* cares about the problems like how to find the shortest path between pairs of nodes and how to get to the target. *Strong navigability* requires that the logical structure and its attached information allow finding the shortest path from the initial to the target as fast as possible.

4.2.5 Human-machine Interactions

Interaction between the user and machine plays an essential role in the RuleViz model. An important decision to be made in designing an interactive system is how to allocate the functions between the user and the machine as well as how to communicate between them. The concept of **balance of control** can be used to characterize the interactions between the user and machine [158], where two extremes are "do-it-yourself mode" and "command mode", respectively. In "do-it-yourself mode", none or less user intervention occurs, while in "command mode", the machine follows the user instructions step by step. Most interaction schema are between these two extremes and can be categorized as follows.

- Manual intervention: e.g. physically dragging a mouse in order to drag an screen object, or clicking on the screen to specify an object.
- Mechanized intervention: e.g. moving a slider or a scroll bar.
- Instructable intervention: e.g. using relations between features such as formula's in a spreadsheet.
- Steerable intervention: e.g. directing an algorithm to perform in a certain way such as making choices.
- No intervention: or automatic manipulation, where an algorithm performs automatically.

Tweedie divides the interactivity in the visualization tools into two categories [158], *direct manipulation* and *indirect manipulation*. The most existing visualization tools exploit direct manipulations in which physical behavior in the real world is replicated literally. Indirect manipulation, however, can add more "magical" functionality that does not rely on direct physical metaphors. For example, in a visualization system based on a parallel-coordinate technique, sliders can be attached to each axis and used as filters for selecting data ranges. As the first slider moves, the data tuples within the current range are selected and the other data are hidden. However subsequent filters (the second slider, third slider, etc.) become more complex to interpret.

In addition, response time and processing cost also need consideration because they affect the system performance. Well-established interface for a visualization system should take less processing time and response fast.

4.2.6 Windowing Strategies

The amount of data to be visualized may be huge. In order to clearly display all parts of data and concentrate on the specific objects when demand, many visualization tools provide more than one window and many panels in a window. The decision of how to layout the windows and how to distribute the data among windows and panels must be made in designing a visualization system. More windows can provide more spaces for the data but may distract the user. Few windows accommodate less data and cause distortion but attract the user's attention and present correlations better. Several strategies can be studied.

- Zoom: Zooming is simple and operates in the same window. Basically, all the data can be visualized in a window to give the user an overview of the data set. Zooming in on a specific object or data area can jump to the next level of details. The problem is that the user may lose the overview when zooming in.

- Overview+Detail: This strategy coordinates two display windows [32], one maintaining an overview and the other showing a detail view. The overview window contains a field-of-view box indicating what is shown in the detail view window. By specifying the portion of the overview window, the user can adjust the contents of the detail view window. A variant of this strategy is *details-on-demand*, where the detail view window pops up within the overview window only when requested.

- Focus+Context: This strategy is different from overview+detail in that it maintains only one display window, but it follows the idea of overview+detail display, dividing the window into two parts, *focus* and *context*. The focus

area (detail view) is embedded in the context (overview). To implement this strategy, perspective transformation such as perspective wall, bifocal view, and fisheyes, may be needed to compress the display in the contextual area (overview display) and keep the normal display in the focus area (detail display). Thus, distortion is produced [32, 141].

4.2.7 Understanding Human Perception

Finally, the implementation of the RuleViz model is highly dependent on the properties of human perception. Therefore, human perception must be studied carefully, including color perception, shape perception, cluster perception, scale perception, etc. The misjudgement for the visualization will mislead interpretations of data, and hence cause the user's wrong reactions. Understanding human perception is difficult, especially different users may have different perceptions for the same picture. This is beyond the scope of this thesis, but should be noticed.

4.3 Implementations of RuleViz

To verify the feasibility of the RuleViz model and compare it with the existing visualization systems as well as knowledge discovery systems, we propose three knowledge patterns: numerical association rules, classification rules and decision trees, as our discovery tasks, and implement four interactive visualization systems for discovering these knowledge patterns. The reason of choosing these patterns is that they are typical knowledge patterns, and extensively researched and widely used in machine learning and data mining communities. The association rules and classification rules do not have natural visual structure, and they are usually in

logical forms, while the decision trees have an intrinsic tree structure.

The four interactive visualization systems we implemented explore different techniques for different components of the RuleViz model. In the following, we briefly present their fundamental approaches and point out the basic techniques used in each component.

1. **AViz: Discovering numerical association rules**

   AViz is an interactive visualization system for discovering numerical association rules, where the rule condition part contains two numerical attributes and the decision part is a nominal or numerical attribute. The task is to find an optimal region consisting of intervals of condition attributes for each decision attribute value or interval. The raw data visualization technique is scatter plot. The data reduction is interactively performed by using "rubber band" and the interesting data area is zoomed in. Three continuous attribute discretization methods are provided, one algorithm-based, one image-based, and one interaction-based. The learning algorithm is image-based. Raw data, data reduction, and data preprocess are visualized in the 2D space, while the final result is displayed in the 3D space. The interaction between the user and machine is performed through either manual manipulation (mouse clicking, list selecting) or mechanized manipulation (moving sliders). All components are completed in a single window.

2. **Learning classifiers**

   Two interactive visualization systems are implemented for learning classifiers, CViz and CVizT. They explore two different approaches for visualizing the KDD process.

   - *CViz: inducing classification rules*

In CViz, data reduction (including feature selection and tuple selection), data preprocess (data transformation, continuous numerical attribute discretization and missing values handling), and rules learning are all algorithm-based. Only are the final and intermediate results visualized. The parallel-coordinate is the basic visualization technique and the classification rules are displayed as strips across the parallel axes. Entropy-based method is used to discretize continuous attributes, the missing values are replaced with the most common values, and the learning algorithm ELEM2 is embedded in the CViz system. The interaction between the user and machine is performed either by dialog (input parameters for discretization and learning algorithms) or by clicking (choosing current class labels) or by moving sliders (specifying accuracy threshold). All components are visualized in a single window and 2D space.

- *CVizT: building classifiers*

  In contrast, all components in the CVizT system are visualization-based. The raw data is sorted according to the class labels and visualized in a *Table Lens* [134], in which the tuple values are drawn as lines proportional to the feature range. The user can examine the correlations between features and class labels and determine to choose the features closely related to the class label, if necessary. For the continuous attributes, the user can specify the cut-points by clicking the mouse on the specific points which can be judged in terms of the Table Lens shape and the tuple values distribution. This is a visualization-based discretization with respect to class labels. The classification rules are interactively built by the user who specifies the condition part for each class label. Once a rule is built, its accuracy rate is displayed. The user can adjust

the rule if the rule's accuracy rate is not satisfying. The final results are visualized as a variant of Table Lens. The most of interactivity is completed manually. Only a single window is needed and all components are visualized in two-dimensional space.

3. **DTViz: constructing decision trees**

   DTViz is a visualization system for interactively constructing decision trees. It is similar to CVizT. All components are visualization-based. A pixel-oriented visualization technique is explored to visualize the raw data, in which a tuple value corresponds to a pixel which is rendered according to the tuple's class label. The feature values are sorted. The features that weakly distinguish class labels can be removed. At the beginning, the decision tree root contains all remaining features and tuples. In each node, the feature that aggregates class labels most closely can be picked by clicking on the feature area as the split attribute. The split attribute is discretized into intervals each of which leads to a branch. Once a terminal node (leaf) is reached, the user informs the system by *finish-clicking* on the terminal node, and the tuples that are covered in the node are removed. The discretization is performed as the decision tree is being constructed. Two windows are used, one for raw data visualization, data reduction, data preprocess, and decision tree node split, the other for the final decision tree. This is different from the overview+detail strategy. The user can adjust the decision by removing a node or exchange the order of split attributes. Most interactions are manual manipulation, and the visualization spaces are two-dimensional.

Chapter 5

AViz: An Interactive System for Visual Mining Association Rules

5.1 Introduction

According to the RuleViz model, we develop an interactive system, AViz, for visualizing the process of discovering three-dimensional numerical association rules from large data sets. The AViz system consists of five components: preparing and visualizing the raw data set, interactive data reduction, interactive numerical attributes discretization with visualization, association rules mining based on the discretized numerical attributes, and the discovered association rules visualization. In this chapter, we present the structure of the AViz system, discuss each components, and illustrate the experiments with the census data sets.

The AViz system is developed to visualize the discovery of a special kind of association rule. Consider association rules of the form: $A \Rightarrow B$, where A consists of two different numerical attributes and B is a quantitative attribute (*numerical or*

nominal attribute). Suppose X and Y are two such different numerical attributes and Z is a quantitative attribute. Our goal is to find an interesting region in the $X \times Y$ plane for each discretized interval $[z_1, z_2]$ of Z, as shown below:

$$Rule: \quad X \in [x_1, x_2], Y \in [y_1, y_2] \Longrightarrow Z \in [z_1, z_2]. \tag{5.1}$$

If Z is a nominal attribute, then each discretized interval corresponds to one of its distinct categorical values.

The *support* of the above rule is defined as the percentage of tuples that meet $X \in [x_1, x_2]$, $Y \in [y_1, y_2]$ and $Z = z$ (or $Z \in [z_1, z_2]$), while its *confidence* is defined as the ratio of its support over the percentage of tuples that meet $X \in [x_1, x_2]$ and $Y \in [y_1, y_2]$. Formally,

$$Supp(Rule) = \frac{number\ of\ tuples\ with\ X \in [x_1, x_2] \wedge Y \in [y_1, y_2] \wedge Z \in [z_1, z_2]}{total\ number\ of\ tuples\ in\ data\ set} \tag{5.2}$$

$$Conf(Rule) = \frac{number\ of\ tuples\ with\ X \in [x_1, x_2] \wedge Y \in [y_1, y_2] \wedge Z \in [z_1, z_2]}{number\ of\ tuples\ with\ X \in [x_1, x_2] \wedge Y \in [y_1, y_2]} \tag{5.3}$$

For example, assume X is *age*, Y is *salary*, and Z is *position*. Further assume that *age* has domain [0, 100], *salary* has domain [10k, 1000k], and *position* takes the values of *programmer, analyst, project leader*, and *manager*. Then the association rules to be discovered might be

$$age \in [20, 25], salary \in [42k, 48k] \Longrightarrow position = programmer$$
$$age \in [26, 35], salary \in [50k, 65k] \Longrightarrow position = analyst$$
$$age \in [30, 40], salary \in [60k, 75k] \Longrightarrow position = project\_leader$$
$$age \in [35, 50], salary \in [65k, 80k] \Longrightarrow position = manager$$

In order to discover and visualize association rules like those shown above, the AViz system emphasizes the following major problems: *interactive data prepara-*

tion and reduction, discretization of numerical attributes, mining algorithm, and *visualization scheme.*

5.2 The AViz System

AViz is an interactive visualization system of knowledge discovery, which employs visualization techniques to preprocess the data and interpret the patterns discovered. It also exploits numerical attributes discretization approaches and mining algorithm to discover numerical association rules according to the requirement (support threshold and confidence threshold) specified by the user.

The visualization in AViz consists of three aspects: raw data visualization, discretized intervals visualization, and association rules visualization. The interaction in AViz includes interactive data preparation, interactive data reduction, interactive discretization, and interactive specification of support and confidence threshold for finding optimized association rules. Since we only consider three numerical attributes, two in antecedence and one in consequence, the data tuple is mapped to a Data Table containing three columns, each of which can be interactively specified and dynamically changed, further projected as a point to the numerical plane consisting of two antecedent attributes, and finally plotted on the screen. The points in the numerical plane may have a different distribution. The user can directly observe the distribution and choose an interesting area to reduce the data. Based on the reduced data set, the numerical attributes can be discretized by using one of three discretization approaches provided in the AViz system. Thus, the X-Y plane is meshed. The grid on the X-Y plane is visualized in the scheme extended from SONAR [61]. In terms of the discretized intervals of numerical attributes and the cell support, the algorithm for discovering association rules is developed to find the

optimized area.

The structure of the AViz system is shown in Figure 5.1, which consists of five components and briefly introduced as follows.

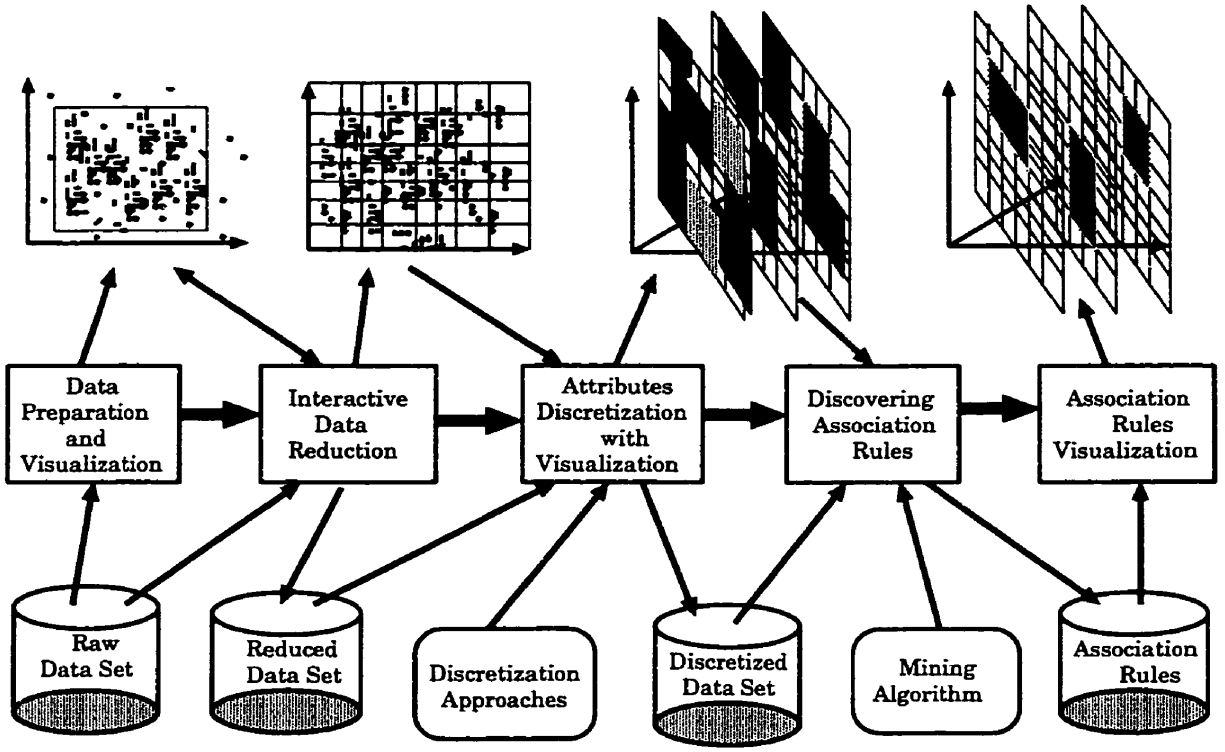


Figure 5.1: The AViz System

To prepare the data sets for numerical association rules discovery, the raw data are transformed into Data Table, and three attributes are interactively specified by the user. The chosen data subset is projected on to a two-dimensional space and plotted on the drawable window. The user can interactively pick up the interesting area by using "rubber band" to horizontally reduce the data. We develop three numerical attribute discretization approaches, including *equal-sized, bin-packing based equal-depth*, and *interaction-based approaches*. For the discretized attributes, a grid is formed. Each cell in the grid is rendered according to the cell "support". We

also define the cell "confidence" in terms of the cell support for different X-Y planes parallel with Z-axis. The algorithm for mining numerical association rules is to find the gain optimized rectangles, each of which constitutes an association rule. Finally, these optimized rectangles are highlighted in the corresponding planes and used as rules. We experiment the AViz system with the census data sets, which illustrates that the AViz system is straightforward for discovering and visualizing numerical association rules and it is easy to use.

5.3 Data Preparation and Visualization

In the AViz system, data preparation is used to specify the raw data file and attributes file with specific format, and the numerical attributes X and Y and the quantitative attribute Z, which constitute the antecedence and the consequence of the association rules to be discovered, respectively. This specification is interactively given by the user and implemented by using file dialog and choice windows. The data set prepared for discovering association rules is a Data Table consisting of three columns $< X, Y, Z >$.

Once the attributes are specified, the data tuples are read from disk, mapped to the points on a numerical plane, and plotted on the drawable window. Since we only consider two numerical attributes as the antecedence of the association rules and one quantitative attribute as the consequence, we project points in 3D space $X \times Y \times Z$ on to the $X \times Y$ plane. Thus it is easily observed how the data distributes in the space, and the joint distribution of the antecedence is visualized so that the interesting area is obviously observed. Generally, on the $X \times Y$ plane, the denser the points in a region are, the more support the region has. The support of a region is important to discovering association rules because each rule must have support

not less than the specified support threshold.

To prepare the raw data, three kinds of files are required. The *data file* provides the AViz system with the raw data set, the *attribute file* is used to describe the characteristics of each attribute contained in the data file, and the *nominal value description file* is to map the discrete values of each nominal attribute to integer values. All files are in plain text format.

The attribute file contains the descriptions of all attributes, each attribute being described in one line as the following format:

rank, name, type, start, length, max, min

where *rank* gives the index of an attribute, *name* the attribute name, *type* attribute type which takes value of *numeric* or *nominal*, *start* and *length* describe the start position and value range of an attribute in the data file, and *max* and *min* define the domain of an attribute.

For example, Table 5.1 shows an attribute file which characterizes attributes *Age, Race, Sex, Class-of-worker, Total-person-income, Total-taxable-income, Weeks-worked-in-year*, and *Hours-worked-weekly.*

The data file contains a list of data tuples, each tuple consisting of a list of fields with fixed length. The format of tuple is as follows:

$$< field_1 >< field_2 > \cdots < field_n >$$

where $< field_i >$ is the value of the i-the attribute $i = 1, 2, \ldots, n$, and n is the number of attributes.

The nominal value description file create mappings from discrete values of nominal attributes to integer values, which counts starting from 0. Thus only integer values are used in the data file to save storage space. All nominal attributes are

Table 5.1: Attributes File Format

```
1, Age, numeric, 6, 2, 99, 0
2, Class-of-worker, nominal, 8, 1, 8, 0
3, Race, nominal, 9, 1, 5, 1
4, Sex, nominal, 10, 1, 2, 1
5, Total-person-income, numeric, 58, 12, 1000000, 0
6, Total-taxable-income, numeric, 70, 12, 1000000, 0
7, Weeks-worked-in-year, numeric, 82, 4, 52, 0
8, Hours-worked-weekly, numeric, 86, 4, 80, 0
```

included in this file and each attribute values mapping is enclosed with *attribute-name* and *End of attribute-name*. For instance, Table 5.2 shows a nominal values description file which maps the values of attribute *Class-of-worker* to integers 0, 1, and 2. Thus the integer values corresponding to the nominal values of attribute *Class-of-worker* occupies only one character. As shown in Table 5.1, their position in the tuples of the data file is 8 with length 1. Similarly, the values of attributes *Race* and *Sex* are mapped to integers 0 through 4 and 0 and 1, respectively.

The AViz system provides file dialog windows to specify the data file, attribute file and nominal value description file. Since we are restricted to discover three-dimensional association rules, three attributes must be specified. This is accomplished by using attribute setting *Choice* windows, each attribute window being used to choose an attribute and set the default number of discrete intervals.

Table 5.2: Nominal Values Mapping to Integer Values

```
Class-of-worker
   0,     Private
   1,     Government
   2,     Self-employed
End of Class-of-worker
Race
   0,     White
   1,     Black
   2,     Amer-Indian-Aleut-or-Eskimo
   3,     Asian-or-Pacific-Islander
   4,     Other
End of Race
Sex
   0,     Male
   1,     Female
End of Sex
```

To visualize the raw data stored in the data file, each tuple is mapped to a point in the draw window. Our purpose is to find the association rules of the form (5.1), so we must first find the potential rectangle $[x_1, x_2], [y_1, y_2]$, that is an *interesting area*. Since the association rules have at least the support threshold, all rectangles that have support less than the threshold will not be considered, which means that these areas can be removed from our consideration. To this end, the raw data tuples is read from disk, the triple values $< x, y, z >$ of the chosen attributes as the antecedence and consequence are taken off according to the attributes' position and length in the data file described in the attribute file, and then the triples are projected to a point in the plane $X \times Y$ with respect to the domain of X and Y.

For each tuple t in the data set, let $t[X], t[Y]$, and $t[Z]$ represent its values of attributes X, Y, and Z, respectively, and $P_X[t]$, $P_Y[t]$, and $P_Z[t]$ be the coordinate of the point corresponding to t, respectively. To plot the data, we map the real world (two-dimensional plane consisting of X and Y) into a screen window (viewport). The mapping function from the real world to the viewport can defined as follows.

Assume the ranges of X and Y are $[Min_X, Max_X]$ and $[Min_Y, Max_Y]$, and the screen window coordinate ranges (viewport) are $x \in [MIN_x, MAX_x]$, and $y \in [MIN_y, MAX_y]$, respectively. Then

$$P_X[t] = MIN_x + \frac{MAX_x - MIN_x}{Max_X - Min_X} \times (t[X] - Min_X),$$

$$P_Y[t] = MIN_y + \frac{MAX_y - MIN_y}{Max_Y - Min_Y} \times (t[Y] - Min_Y).$$

5.4 Interactive Data Reduction

Generally, business data sets contain millions or billions of data items. To discover associations between attributes from the huge data set, the raw data must be read

from disk or input from other systems and stored in memory. Even if the data set is scanned once from disk, it is time-wise expensive. So we must reduce the passes of disk scans as much as possible. On the other hand, Data is not always clean, which means that there may exist uninteresting data items or *data derivation*. To grasp the nature of the interesting data, the raw data set should be cleaned.

In the AViz system, two approaches to cleaning the data are provided, *vertical reduction* and *horizontal reduction*. The former reduces the number of attributes, while the latter reduces the number of tuples.

To reduce the data set vertically, we restrict the AViz system to process only three attributes each time. This has been fulfilled interactively in data preparation.

To reduce the data set horizontally, by observing the data distribution, the user can interactively specify the data area in which he/she is interested and remove other data that are outside the interesting area. The interesting region on the $X \times Y$ plane is picked up by using a rubber band. This region usually contains dense points and has high support. The points outside the region are cleaned. Therefore, the size of the data set used to discover association rules is reduced. The result of the reduction is redisplayed on the screen window so that it can be reduced further. This step can be repeated until the user is satisfied with the final result.

In the AViz system, the full data set is read twice, the first read is to visualize the raw data and clean the data, and the second read is to discover and visualize the association rules. The full pass read can overcome the drawback of sampling approach especially when the data set is not distributed evenly. To avoid the memory requirement for storing the full data set, AViz system exploits an *aggregate* array. Each element of array represents an interval along axis or a square on the

$X \times Y$ plane and characterizes the subset of the full data set that falls in the corresponding interval or grid cell. Thus the storage depends on the number of the discretized intervals (buckets) of X, Y, and Z, regardless of the size of the full data set.

After data tuples are projected to the $X \times Y$ plane and mapped to the draw window, the data distribution can be directly observed. Since the observed distribution is limited by the draw window size, many distinct points in the $X \times Y$ plane may be overlapped when mapped to the window, so the picture only roughly reflects the data distribution. To reflect the real distribution, we attempt to reduce the domain of attributes and restrict to the interesting area. All points outside the interesting area are dropped. What is the interesting area depends on the user's observation and perception.

To reduce the data set horizontally by dropping uninteresting points, the user can interactively specify the interesting data area by using a rubber band. This region usually contains dense points and has high support.

Once the interesting area is selected, the AViz system rereads the data from disk and redraws data tuples in the same way as above. Thus, the user can further choose the data area in which s/he is interested. These two steps, visualizing raw data and choosing an interesting area, can be repeated until the final data subset used to find association rules is satisfiable.

5.5 Discretizing Numerical Attributes with Visualization

After the raw data is reduced, the specified interesting data area is scaled and redrawn on the whole window. To discretize the numerical attributes into disjoint intervals (buckets), the AViz system provides three approaches in its *Discretization Approaches* library, shown in Figure 5.1, including equi-sized, bin-packing based equi-depth, and interaction-based approaches. The user can choose one of three approaches to discretize jointly two numerical attributes of the antecedence. If the consequence attribute is a nominal attribute, its distinct values are seen as disjoint buckets.

The bin-packing based equi-depth approach is based on *bin-packing* and also employs equi-depth on the basis of the packed bin. The equi-sized and bin-packing based equi-depth approaches require the user to specify the number of intervals for both numerical attributes. The interaction-based approach is based on equi-sized or bin-packing based equi-depth approach, by which the user can intuitively adjust the partition of above two discretizations through observing the distribution of the data.

Equi-sized discretization

The *equi-sized approach* partitions the continuous domain into intervals with equal length. For example, if the domain of attribute *age* is $[0, 99]$, then it can be divided into small intervals of length 10, thus we have intervals $< age, 0, 9 >$, $< age, 10, 19 >, \ldots, < age, 90, 99 >$. This approach is simple and easily implemented. The main drawback of this approach is it may miss many useful rules since it does not consider the distribution of the data values.

Suppose the domains of numerical attributes X and Y are $[Min_X, Max_X]$ and $[Min_Y, Max_Y]$, respectively. $X \times Y$ forms an Euclidean plane. Each tuple t in the data set can be mapped to a point $(t[X], t[Y])$ in $X \times Y$. Assume X and Y are discretized into N_x and N_y buckets, respectively. Then the size of buckets is, in average, $(Max_X - Min_X)/N_x$ for X, and $(Max_Y - Min_Y)/N_y$ for Y. For a region P in $X \times Y$, we say a tuple t meets condition $(X, Y) \in P$ if t is mapped to a point in region P.

Bin-packing based equi-depth discretization

Bin-packing based equi-depth approach is different from the existing approaches. The domain of the numerical attributes may contain an infinite number of points. To deal with this problem, several equi-depth discretization methods have been proposed, for example, KID3 employs an *adjustable buckets method* [133], while the approach proposed in [152] is based on the concept of a partial completeness measure. The drawback of these approaches is in time-consuming computation and/or large storage requirements. AViz exploits a simple and direct method.

Assume the window size used to visualize the data set is M (width or height) in pixels, and each pixel corresponds to a *bin*. Thus we have M bins, denoted $B[i], i = 0, \ldots, M-1$. Map the raw data tuples to the bins in terms of the mapping function. Suppose $B[i]$ contains $T[i]$ tuples, and the attribute is to be discretized into N buckets. According to equi-depth approach, each bucket will contain $d = \sum_{i=0}^{M-1} T[i]/N$ tuples. We first assign $B[0], B[1], \ldots$, to the first bucket until it contains d or more tuples, and then assign the following bins to the second bucket. Repeat this operation until all buckets contain a roughly equal number of tuples. This process is depicted in Figure 5.2.

The storage requirement in bin-packing based equi-depth approach is $O(M+N)$,

```
j=0;
for (i = 0; i < N; i++)
    Bucket[i] = 0;
    for (k = j, K < M, k++)
        Bucket[i]+ = T[j++];
        if Bucket[i] ≥ d
            break;
```

Figure 5.2: Bin-packing Based Equi-depth Discretization

depending on the number of buckets and the size of the visualization window, regardless of the domain of the attributes and the size of the data set. It does not need to sort the data and the execution time is linear in the size of the data set. This method, however, may not produce enough buckets, because each bin must be assigned to only one bucket, and cannot be broken up. For instance, if the data concentrates on several bins, then the buckets that contain these bins will contain many more tuples than others. This case could happen especially when the visualization window has a small size.

Interaction-based

The third discretization approach that AViz employs is interaction-based. This method consists of two steps. First, the user can specify one of the two above approaches to simply discretize the attributes. AViz displays the discretization result. In the second step, the user can intuitively observe the data distribution and the discretization, and then move discretization lines to wherever s/he thinks appropriate by clicking and dragging the mouse. In this interaction process, the

user can actively decide the discretization of numerical attributes. However, since the visualized data has been preprocessed and mapped onto the screen, the user can only observe the graphics to obtain a rough idea about the data distribution. For a small visualization window, distortion inevitably occurs. This may cause discretization errors.

To visualize the discretization of continuous attributes, three sets of buckets are created, one for each attribute. Thus a collection of squares is obtained, each *square* consisting of two intervals, one from each antecedent numerical attribute. These collections of buckets and squares are stored in *Discretized Data Set* in Figure 5.1.

After the attributes are discretized, the raw data in the interesting area is read from the disk again, and the mapped points are redrawn on the screen. While reading the interesting data, count the support and hit for each square with respect to the buckets of Z, which can be used to compute the RGB-color of the square. The support of a square is the number of points that fall in it, and the hit of a square with respect to each bucket of attribute Z is the number of points that fall in this square and have a value that falls in this bucket of Z. For each square, the sum of its all hits is equal to its support. The visualization of the discretized attributes is to render all squares for all buckets of Z based on the support and hit. In the graphics, each bucket of the attribute Z corresponds to a plane parallel with $X \times Y$ plane.

Assume that attribute X is partitioned into N_x buckets, and Y into N_y buckets, then the total number of squares is $N_x \cdot N_y$. Usually, N_x and N_y are between 20 and 300 in practice [61]. Hence the data set is mapped into $N_x \cdot N_y$ squares, no matter how large the data set size is.

AViz is based on the two-dimensional model for visualizing numerical associa-

tion rules proposed by Fukuda et al [61]. The buckets may or may not be equi-sized, depending on the discretization approach. The screen axes are partitioned correspondingly. Thus the $X \times Y$ plane is divided into $N_x \cdot N_y$ *unit squares.* A tuple t in the data set is projected to the unit square containing the point $(P_X[t], P_Y[t])$.

For each bucket, denoted z, of attribute Z, consider the unit square G_{ij}, which is composed of the i-th bucket of X and the j-th bucket of Y. Let u_{ij} denote the number of total tuples and v_{ij}^z the number of tuples satisfying $Z \in z$, which are projected to cell G_{ij}, that is,

SELECT count(*)
FROM dataset INTO u_{ij}
WHERE $< P_X[t], P_Y[t] > \in G_{ij}$).

SELECT count(*)
FROM dataset INTO v_{ij}^z
WHERE $< P_X[t], P_Y[t] > \in G_{ij}$ and $t[Z] = z$.

The confidence of a square G_{ij} with respect to the bucket z can be easily calculated as

$$Conf(G_{ij}^z) = \frac{v_{ij}^z}{u_{ij}} \in [0, 1]. \tag{5.4}$$

Thus, on the plane $Z = z$, G_{ij} is rendered with color RGB=$(v_{ij}^z, u_{ij} - v_{ij}^z, 0)$. The *red* component v_{ij}^z represents the support of the rule, while the *green* component reflects the confidence. So the redder the square, the higher its support , and the brighter the square, the higher its confidence.

5.6 Discovering Association Rules

For each value or interval of Z, we attempt to find an optimized region on the $X \times Y$ plan, and construct an association rule in the form of (5.1). To find the optimized regions, we extend the concepts *confidence* and *support* for a square to those for any form of region on the plane. The support and confidence of a region P are defined as follows:

$$\begin{aligned} Supp(P) &= \textstyle\sum_{G_{ij} \in P} u_{ij} \\ Hit(P^z) &= \textstyle\sum_{G_{ij} \in P} v_{ij}^z \\ Conf(P^z) &= \frac{Hit(P^z)}{Supp(P)} \end{aligned} \tag{5.5}$$

A region is said to be *ample* if its support is greater than or equal to the support threshold. A region is said to be *confident* if its confidence is greater than or equal to the confidence threshold.

The algorithm for discovering numerical association rules by visualization is discussed in [60, 61]. To start with, we define the concept *gain*. For the given confidence threshold θ, the *gain* of a square G_{ij} on the plane $Z = z$ is defined as

$$gain(G_{ij}^z) = v_{ij}^z - \theta \times u_{ij}. \tag{5.6}$$

Obviously, when the confidence of $G_{ij} = \theta, gain(G_{ij}) = 0$ with respect to $Z = z$. The gain reflects the relationship between the confidence and the threshold. When the confidence is greater than the threshold, the gain is positive, while the gain is negative when the confidence is less than the threshold.

The *gain* concept can also be extended to any form of regions. The *gain* of a region P with respect to $Z = z$ is defined as

$$gain(P^z) = \sum_{G_{ij} \in P} gain(G_{ij}^z) = \sum_{G_{ij} \in P} (v_{ij}^z - \theta \times u_{ij}). \tag{5.7}$$

It is easy to show that the following problems are all intractable [61]:

- find the optimized gain region, ample region, or confident region with the maximum gain;
- find the optimized support region or confident region with the maximum support; and
- find the optimized confident region or ample region with the maximum confidence

To circumvent the intractable problems, we restrict the optimized region to a specific form. In particular, we consider rectangular regions.

We design a dynamic programming algorithm, depicted below, to find the optimized gain rectangle. The basic idea is, for each Z value z, to choose randomly a pair of rows, say i-th and j-th rows, $1 \leq i \leq j \leq N_y$, and consider rectangles $G^z([i,j],m)$ on the plane $Z = z$, which consists of the squares from the i-th row to the j-th row in the m-th column, for $m = 1, 2, \ldots, N_x$. Then compute the gain for each $G^z([i,j],m)$,

$$gain(G^z([i,j],m)) = \sum_{k=i}^{j} gain(G^z_{km}) = \sum_{k=i}^{j} (v^z_{km} - \theta \times u_{km}). \tag{5.8}$$

Finally, for $1 \leq i \leq j \leq N_y, 1 \leq r \leq k \leq N_x$, compute the gain of the rectangular region $G^z([i,j],[r,k])$,

$$gain(G^z([i,j],[r,k]) = \sum_{m=r}^{k} gain(G^z([i,j],m)), \tag{5.9}$$

which consists of rows from i to j, and columns from r to k. The optimized gain rectangle is the rectangle with the highest gain.

In the algorithm **OptimizedRectangle**, without loss of generality, we assume $N_y \leq N_x$. This algorithm takes time of $O(N_z \cdot N_x \cdot N_y \cdot min\{N_x, N_y\})$ time, where N_z

is the number of buckets of attribute Z, that is, the number of values of attribute Z, if Z is nominal, or the number of discrete intervals of Z, if Z is numerical, because it contains four nested loops.

Algorithm: **OptimizedRectangle**

Dynamic Programming Algorithm for Finding Optimized Gain Rectangle

Input: u_{ij}, number of tuples mapped to unit square G_{ij},

v_{ij}^z, number of tuples mapped to unit square G_{ij} with value z of Z,

support threshold δ and confidence threshold θ

Output: an optimized rectangular region with maximum gain

Variables: $gain(G_{ij}^z)$–the gain of square G_{ij} with $Z = z$

$X^z([i,j],m)$–the gain of rectangle which consists of

squares from i-th row to j-th row in column m with $Z = z$

BEGIN

$OptRect = null$;

for each value z of Z

for $m = 1, \ldots, N_x$

for $i = 1, \ldots, N_y$

if $u_{im} \geq \delta$

$gain(G_{im}^z) = v_{im}^z - \theta \times u_{im}$;

else $gain(G_{im}^z) = -\infty$;

$X^z([i,i],m) = gain(G_{im}^z)$;

for $j = i+1, \ldots, N_y$

if $u_{jm} \geq \delta$ **Then** $gain(G_{jm}^z) = v_{jm}^z - \theta \times u_{jm}$;

else $gain(G_{jm}^z) = -\infty$;

$X^z([i,j],m) = X^z([i,j-1],j) + gain(G_{jm}^z)$;

for $i = 1, \ldots, N_y$

for $j = 1, \ldots, N_y$

$Max^z(1) = X^z([i,j],1);$

$MAX^z(\leq 1) = Max^z(1);$

for $m = 1, \ldots, N_x - 1$

$Max^z(m+1) = max\{0, Max^z(m)\} + X^z([i,j], m+1);$

$MAX^z(\leq m+1) = max\{Max^z(m+1), MAX^z(\leq m)\};$

$OptRect = OptRect \cup MAX^z(\leq N);$

Return $OptRect$;

END

5.7 Visualizing Association Rules

Finally, the association rules discovered by the mining algorithm are visualized. The visualization scheme is very simple. Since each association rule corresponds to an optimal region on the plane parallel with Z-axis with respect to $Z \in [z_1, z_2]$ in the three-dimensional space $X \times Y \times Z$, the visualization of these association rules is to focus the optimized regions with the deep color, and the other regions are rendered with light color.

The optimized regions depend on the support and confidence threshold. As the threshold varies, the optimized regions change, and the corresponding association rules vary. To reflect this change, the mining algorithm is repeatedly executed as the threshold sliders move.

5.8 AViz Implementation and Experiment

The AViz system has been implemented in JDK1.2 and Java3D. The data preparation is accomplished by choosing a data file and attributes file, and specifying the attributes to be mined. The data file is formatted in tuples which consists of a series of fixed length fields. The attributes file characterizes each attribute, including attribute name, type, length, the position in the data file, and domain. This is implemented in dialog windows (under the file menu and setting menu). The steps of discovering knowledge is controlled by the *control menu.* Two sliders are used to control the support threshold and confidence threshold. By moving these sliders in the discovering step, the resulting rules (focus areas in all planes parallel with $X \times Y$ plane) vary quickly.

AViz has been applied to the U.S. census data in 1996 to find the association rules between attributes. The data set contains about 1.4 million tuples, each tuple consisting of 5 numerical attributes *age, total-person-income, taxable-income-amount, tax-amount, hours-usually-worked-per-week* and 3 nominal attributes *sex, race, class-of-work*, as shown in Table 5.1.

In this section, we illustrate our experiments with this data set and different attributes, respectively. For each experiment, we keep track of the process of association rules discovery and visualization step by step.

5.8.1 AViz Experiment 1

In the first experiment, we attempt to find associations between *Taxable-income-amount*, *Total-person-income*, and *Race.*

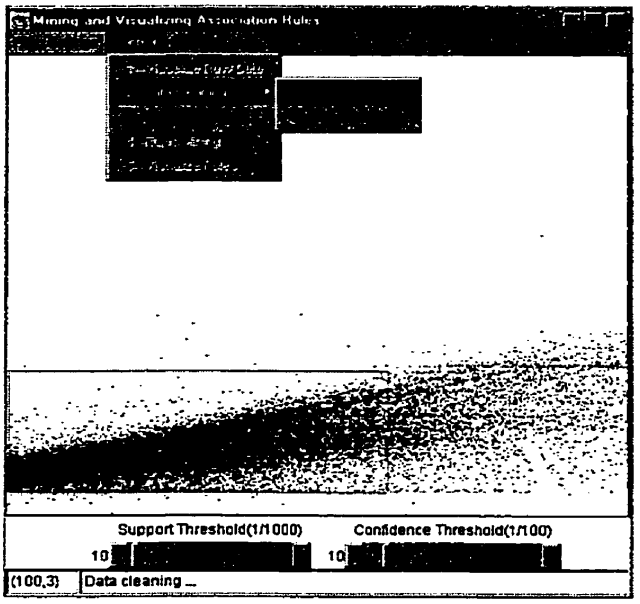


Figure 5.3: The Raw Census Data and Interesting Area with *Taxable-income-amount* and *Total-person-income*

Step 1: Data Preparation and Visualization

Choose two numerical attributes X=*Taxable-income-amount* and Y=*Total-person-income*, and an nominal attribute Z=*Race*. The domain of X and Y is $[0, 1000K]$ and $[0, 500K]$, respectively. Z takes the following values: *White, Black, Amer-Indian-Aleut-or-Eskimo, Asian-or-Pacific-Islander*, and *Other*. Also, we specify that X and Y are to be discretized into 20 intervals.

Map the raw data into the visualization window, shown in Figure 5.3.

Step 2: Interactive Data Cleaning

From Figure 5.3, it is clearly seen that most data concentrates on a strip which is interesting to us. The other data can be cleaned. For now, we pick this strip by using rubber band. After cleaning, the remaining data set contains about 1.08 million tuples.

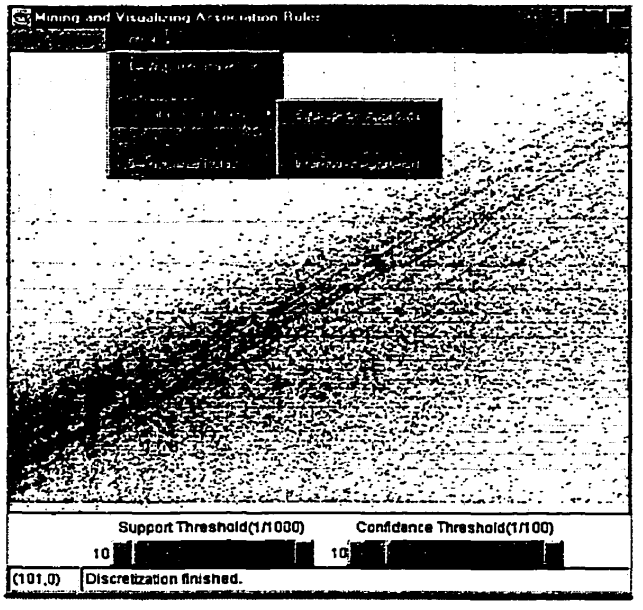


Figure 5.4: Discretizing the Numerical Attributes in Figure 5.3 with *Bin-packing Based Equi-depth Method*

Step 3: Discretizing Numerical Attributes with Visualization

We choose the second approach of attribute discretization, bin-packing based equi-depth, and then utilize the interaction-based method by moving discretization lines to adjust the discretization. The result is shown in Figure 5.4.

Figure 5.5 visualizes the discretization of *Taxable-income-amount* and *Total-person-income* for each value of Z = *White, Black, Amer-Indian-Aleut-or-Eskimo, Asian-or-Pacific-Islander*, and *Other*. Each Z value corresponds to a plane and the volume consisting of all planes rotates around Y axis so that all planes can be viewed clearly.

Step 4: Discovering the Optimized Regions

To find the association rules, we move the threshold sliders and specify the support threshold and confidence threshold as 0.2% and 20%, respectively. we obtain

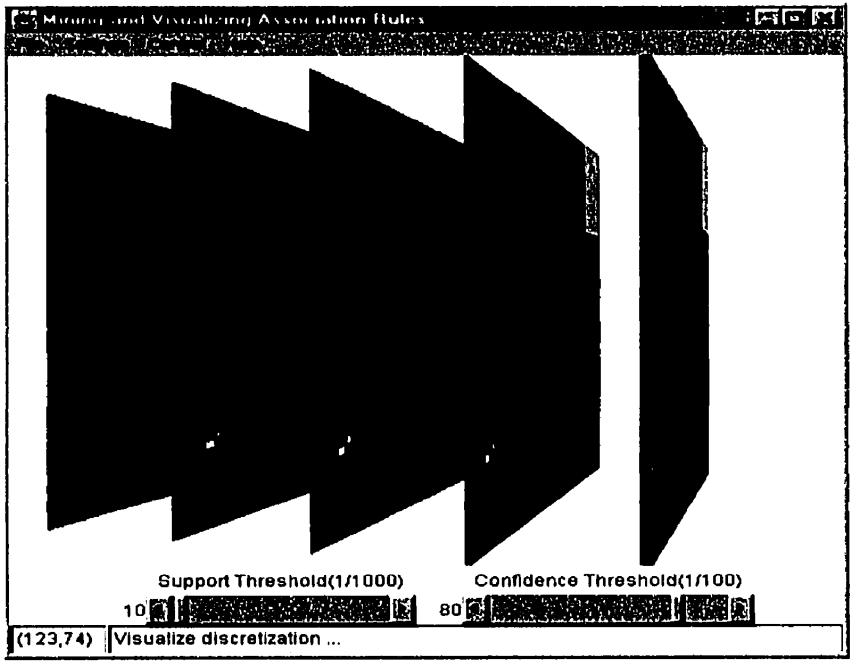


Figure 5.5: Visualizing the Discretization Shown in Figure 5.4

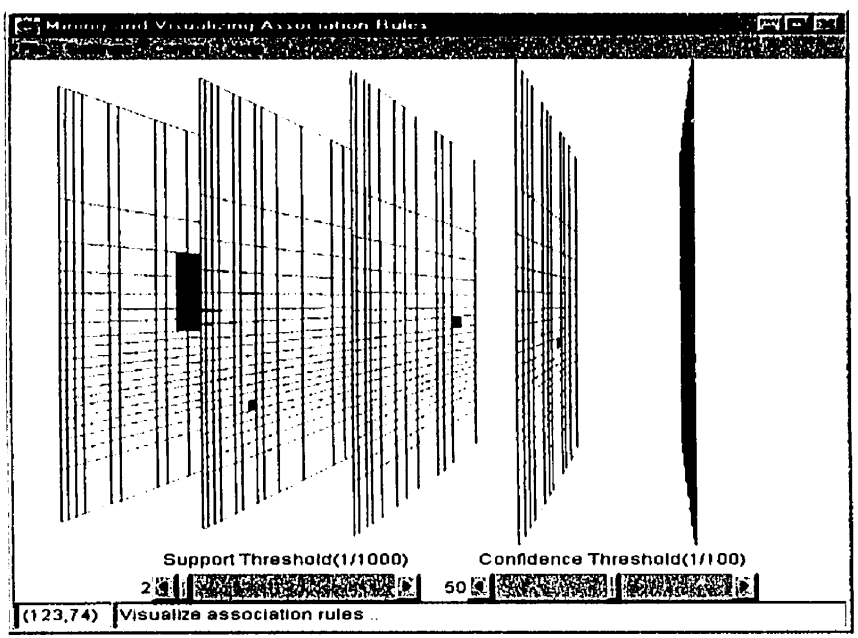


Figure 5.6: Visualizing the Optimal Rectangles (Association Rules)

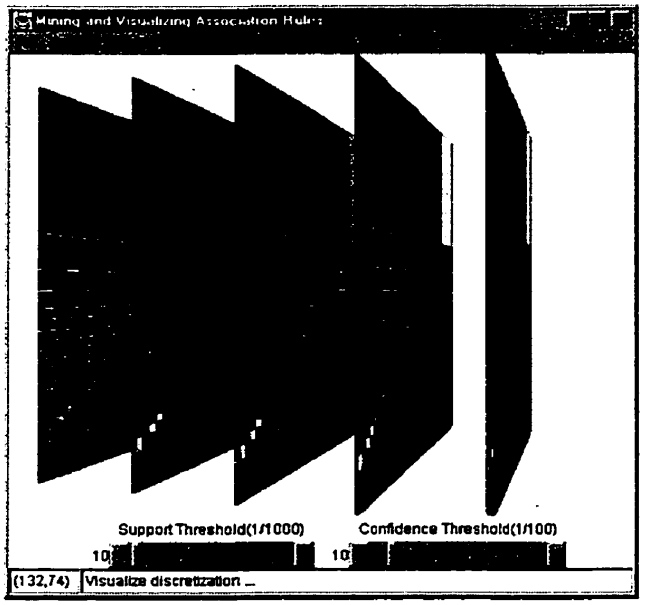


Figure 5.5: Visualizing the Discretization Shown in Figure 5.4 by Rotating Around the Y Axis

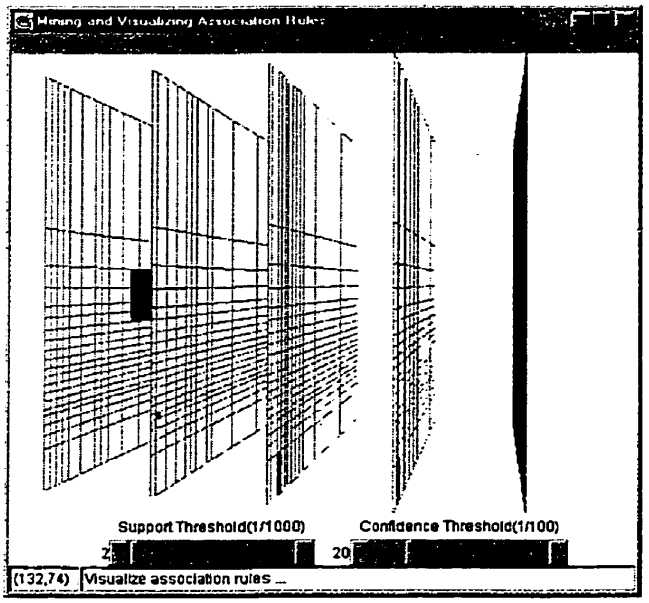


Figure 5.6: Visualizing the Optimal Rectangles Representing Association Rules Discovered from Figure 5.5

five rules, each corresponding to a value of Z, which are described as following.

$X \in [16.74K, 17.86K], Y \in [42.78K, 43.73K] \Rightarrow Z = White$

$X \in [13.37K, 14.49K], Y \in [35.24K, 35.53K] \Rightarrow Z = Black$

$X \in [9.99K, 11.12K], Y \in [32.20K, 32.73K] \Rightarrow Z = Amer\text{-}Indian\text{-}or\text{-}Eskimo$

$X \in [9.99K, 11.12K], Y \in [32.20K, 32.73K] \Rightarrow Z = Asian\text{-}or\text{-}Pacific\text{-}Islander$

$X \in [9.99K, 11.12K], Y \in [32.20K, 32.73K] \Rightarrow Z = Other$

The result shows *Amer-Indian-Aleut-or-Eskimo*, *Asian-or-Pacific-Islander* and *other* have the same optimized area, but *White* and *Black* have the different ones.

Step 5: Visualizing the Association Rules

Each optimized region represents an association rule for the specified Z value. Figure 5.6 illustrates the result of optimized regions which are highlighted in the rotating planes.

5.8.2 AViz Experiment 2

In the second experiment, we choose different attributes: *Total-person-income, Age,* and *Class-of-work.* The following is the procedure of discovering association rules between these attributes.

Step 1: Data Preparation and Visualization

Through a dialog, choose two numerical attributes X=*Total-person-income* and Y=*Age*, and an nominal attribute Z=*Class-of-work.* The domain of X and Y is $[0, 1000K]$ and $[10, 100]$, respectively. Z takes the following values: *Private, Government*, and *Self-employed*, which represent "works in private company", "works in

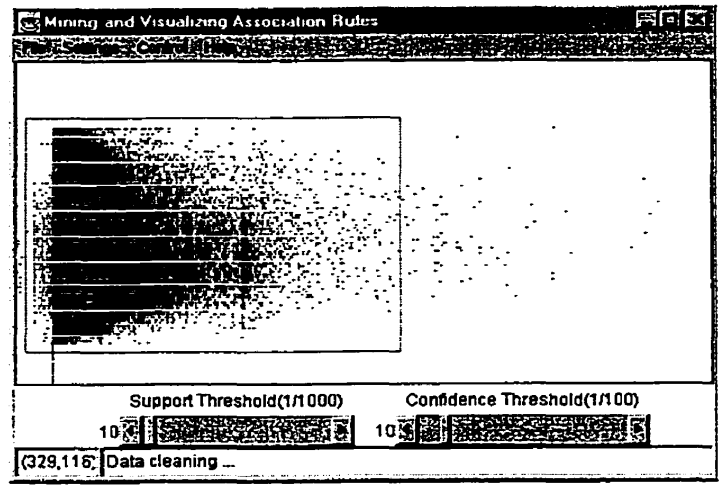


Figure 5.7: Visualizing the Raw Census Data with Attributes *Total-person-income* and *Age*, and Picking the Dense Area Using Rubber Band

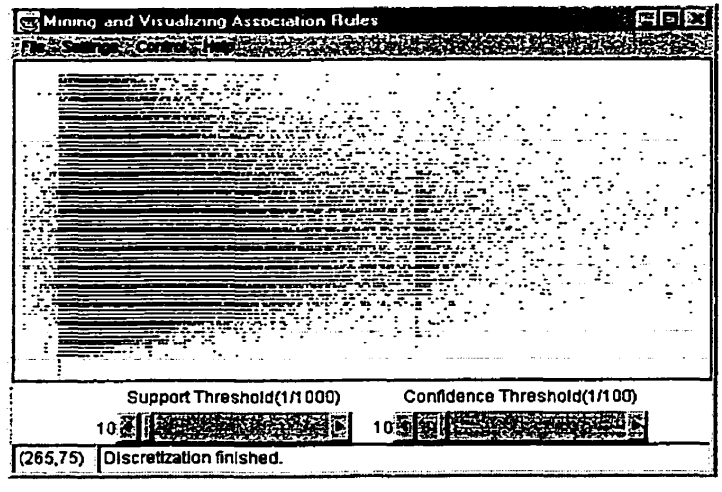


Figure 5.8: Numerical Attributes Discretization with *Bin-packing Based Equi-depth Method* and *Interaction-based Method* According to the Picked Area in Figure 5.7

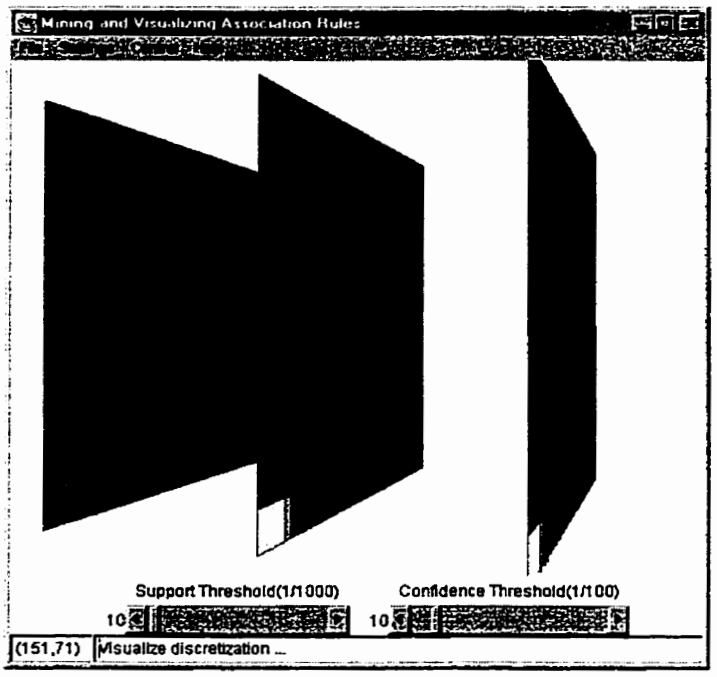


Figure 5.9: Visualizing the Discretization Obtained from Figure 5.8

government (including federal, state and local government)", and "self-employed", respectively.

Similarly, after the attributes are specified, the raw data are visualized in the two-dimensional plane, shown in Figure 5.7.

Step 2: Interactive Data Cleaning

Figure 5.7 shows that the census data follows approximately the Gaussian distribution with the mean being around the point of $Age = 40$. We might be only interested in the data points concentrating on the center of the image and the other data can be cleaned. For now, we pick this area by using rubber band. After cleaning, the remaining data set contains about 1.18 million tuples. The picked data is revisualized, showed in Figure 5.8.

Step 3: Discretizing Numerical Attributes with Visualization

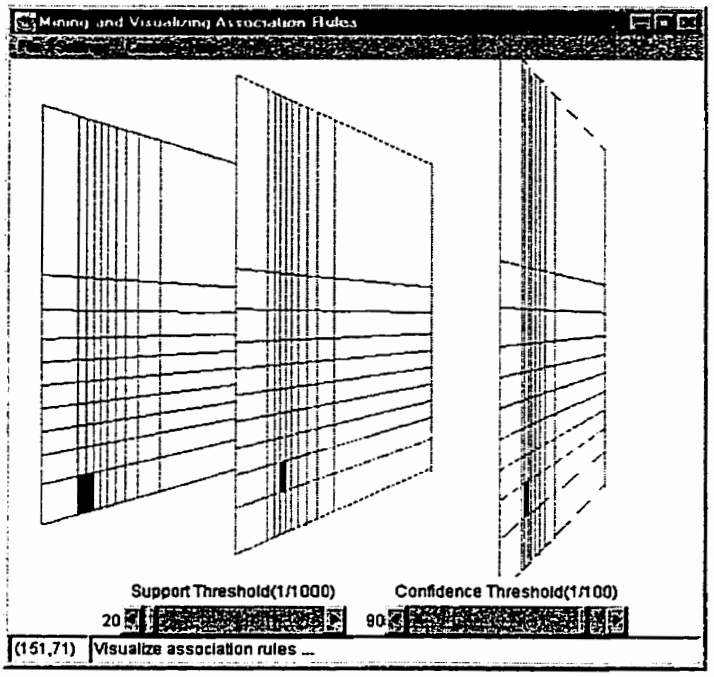


Figure 5.10: Rotating the Association Rules for Support Threshold 2% and Confidence Threshold 90%

Choose the second approach of attribute discretization, *bin-packing based equi-depth*, and then utilize the *interaction-based* method by moving discretization lines to adjust the discretization. The result is shown in Figure 5.9. Each Z value corresponds to a plane and all planes rotate around Y axis so that all planes can be viewed clearly from different angles.

Step 4: Mining Association Rules

The algorithm for discovering the association rules is executed to find the optimal region in terms of the user-specified support and confidence threshold by moving threshold sliders, and each rule is represented as an optimal region on the plane $Z = z$ of the 3D space $X \times Y \times Z$. The discovered results are visualized in the next step.

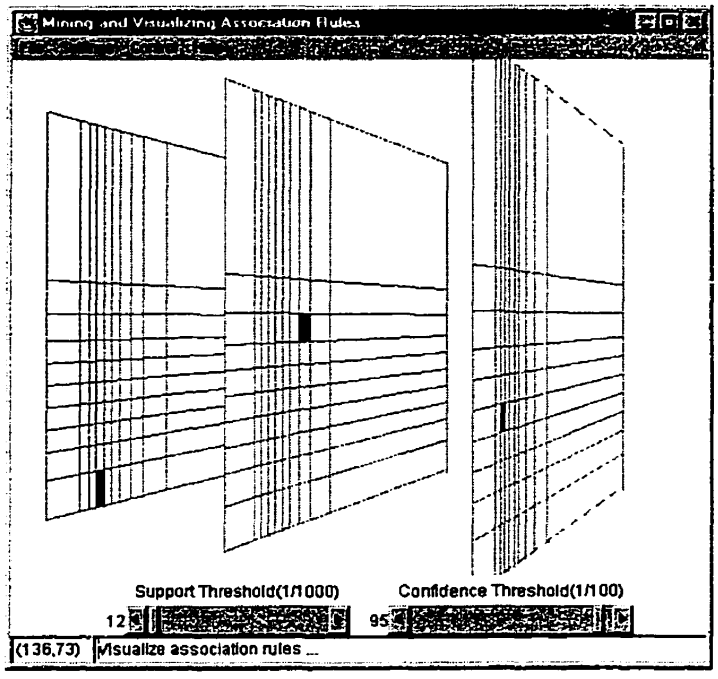


Figure 5.11: Visualizing the Association Rules Achieved with Support Threshold 1.2% and Confidence Threshold 95%

Step 5: Visualizing Association Rules

Figures 5.10 and 5.11 show two groups of association rules for different support and confidence thresholds. In Figure 5.10, the support and confidence threshold are specified by the threshold sliders as 2% and 90%, respectively. The three optimal regions on three planes correspond to three association rules, described as follows:

$X \in [16, 24], Y \in [25.56K, 32.34K] \Longrightarrow Z = \textit{private}$

$X \in [24, 30], Y \in [32.34K, 35.23K] \Longrightarrow Z = \textit{government}$

$X \in [24, 30], Y \in [29.15K, 32.34K] \Longrightarrow Z = \textit{self-employed}$

In Figure 5.11, the support and confidence threshold are set to 1.2% and 95%, respectively, and another three associations are obtained as follows:

$X \in [16, 24], Y \in [32.34K, 35.23K] \Longrightarrow Z = \textit{private}$

$X \in [48, 55], Y \in [41.60K, 45.48K] \Longrightarrow Z = \textit{government}$

$X \in [35, 39], Y \in [29.15K, 32.34K] \Longrightarrow Z = \textit{self-employed}$

Chapter 6

CViz: A Visualization System for Rule Induction

6.1 Introduction

Interactive visualization techniques allow people to visualize the results of presentations on the fly in different perspectives, and thus help users understand the discovered knowledge better and more easily. This interactive process makes the knowledge discovery process straightforward and accessible.

In this chapter, we introduce an interactive visualization system, CViz, for rule induction. The process of learning classification rules is visualized, which consists of five components: preparing and visualizing the raw data, reducing the raw data, preprocessing the data, learning classification rules, and visualizing the discovered rules. In contrast to the AViz system, in which most components are imaged-based, all components of the CViz system, including feature selection, tuple reduction, missing values handling, continuous attribute discretization, and rule induction, are

algorithm-based except perception-based feature deleting and tuple cleaning. The process visualization is based on *parallel coordinates* technique. The visualization task is to display the raw data, intermediate results, and final rules.

Basically, the raw data tuples are scaled attribute by attribute to a pixel range determined by the display window. To select features in the case of too many features in the raw data sets, the attribute relevance degree with respect to the class labels of each condition attribute is calculated and ranked, and the features with higher relevance degree are selected. To reduce the size of tuples in the case of too many tuples, the tuples are randomly drawn with the condition that each class must not contain too few (e.g., less than the number of class labels) tuples. A simple method for handling missing attribute values is adopted, where the missing condition attribute values are estimated as the most probable values, while the tuples with missing class labels are discarded. The continuous attributes are discretized with an *entropy-based approaches*, called EDA-DB. The algorithm ELEM2 for learning classification rules is used to induce rules. The final results (classification rules) are displayed as *color strips* across the related parallel coordinates, where the color or grey-scale represents rule accuracy and rule quality [11], the intersection with coordinates indicates the attribute values or intervals, and forward hatch and backward hatch illustrate positive and negative rules, respectively.

The CViz system stresses the visualization of intermediate results during the discovery of classification rules and the interaction between the user and the embedded algorithms. Interaction tools are provided for choosing approaches, specifying parameters, input thresholds, and selecting menu and commands.

In this chapter, we outline the structure of the CViz system and its components, introduce the data visualization technique, *parallel coordinates*, suggest the feature selection and tuple selection, discuss the missing values handling and continuous

attribute discretization approaches used in CViz, and present the learning algorithm ELEM2 used in the CViz system and the rule visualization algorithm. Our experiments with the UCI repository data sets and artificial data set show that the CViz system is useful and helpful for visualizing and understanding the learning process of classification rules.

6.2 The CViz System

CViz is developed for classification rule induction according to the RuleViz model. The CViz system exploits *parallel coordinates*, a data visualization technique, to visualize the raw data, help the user determine the data cleaning and preprocessing, provide the user with interaction tools to understand the intermediate results, and interpret the rules discovered. The CViz system embeds feature selection and tuple selection methods, conducts numerical attribute discretization and missing values handling, and encompasses a rule induction algorithm to learn classification rules based on the training data set. CViz consists of five components, shown in Figure 6.1.

In CViz, the raw data is visualized based on the *parallel coordinates* technique [98]. Suppose the training data are represented as n-ary tuples. n equidistant axes are created to be parallel to one of the screen axes, say the Y-axis, and correspond to the attributes. The axes are scaled to the range of the corresponding attributes and normalized, if necessary. Every tuple corresponds to a polygonal line which intersects each of the axes at the point that corresponds to the value for the attribute [91]. Figures 6.4 and 6.5 show the effects of using parallel coordinates technique to visualize multidimensional data sets. Each coordinate is interpreted further with a list of values for a categorical attribute, while a numeric attribute can be attached

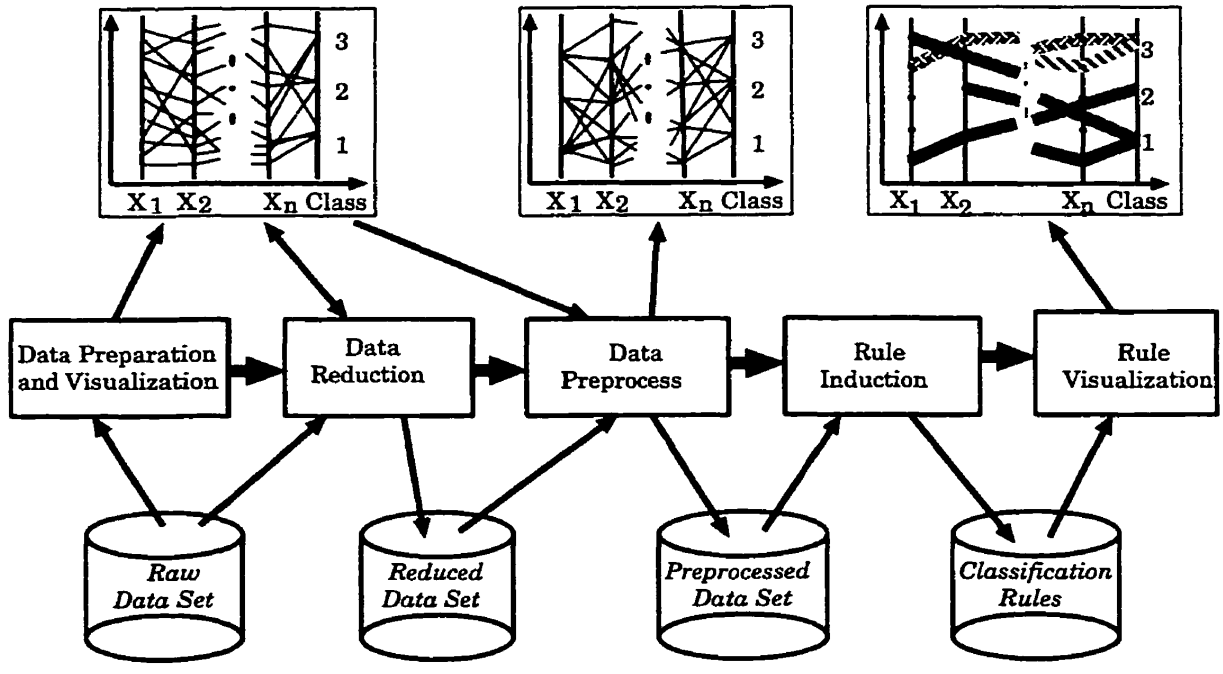


Figure 6.1: The CViz System

with the minimum and maximum, even mean, variance, etc. As the algorithm proceeds, numeric intervals can be added, including the start point and the end point for each interval and the number of total intervals once the numeric attributes are discretized.

The second component is to reduce the raw data, if necessary. Once the raw data is visualized on the parallel coordinates, the user can get a rough idea about the data distribution so that data cleaning might be performed. The data reduction involves two aspects: *horizontal* and *vertical* reduction. Two approaches for data reduction are developed. One is based on interaction. An attribute that the user thinks irrelevant to the knowledge discovery can be interactively removed by *deleting-click*, thus the data is horizontally reduced. While an attribute value can be deleted if the distribution of the data tuples passing the value is too sparse, thus the data

tuples that have the value could be removed so that the data is vertically reduced. The data reduction is highly dependent on the user [76]. The other data reduction approach is algorithm-based. We exploit attribute ranking of relevance degree with respect to the class label to select features, and randomly sampling to draw data tuples.

The third component is to discretize numerical (continuous) attributes. The CViz system provides three approaches: *equi-length*, *equi-depth*, and *entropy-based* methods [12]. The user can select the method that s/he likes, or select all of them for different learning runs for comparison of the learning results. The discretized attributes are visualized again where an attribute interval corresponds to a point on the attribute coordinate. Figures 6.9 and 6.14 show the discretized parallel coordinates.

To induce classification rules, we embeded the ELEM2 learning algorithm [11] in the CViz system. ELEM2 employs sequential covering learning strategy, general-to-specific heuristic search strategy, and unlearnable region in the induction and classification processes to improve the rule accuracy. ELEM2 also develops a rule quality measure for post-pruning.

The final result is a set of classification rules which are visualized as colored strips (polygons). Each rule is illustrated by a subset of coordinates with corresponding values and/or intervals. The coordinates in the subset are connected together through the values or intervals. The rule accuracy and quality values are used to render the rule strips. Figures 6.10 through 6.12 and 6.15 through 6.17 illustrate a part of the classification rules obtained in our experiments.

6.3 Raw Data Visualization

The underlying visualization technique used in the CViz system is *parallel coordinates* which induces a non-projective mapping between high dimensional and two-dimensional data sets, and visualizes analytic and synthetic multi-dimensional geometry [95]. Basically, for N-dimensional data sets with each dimension being denoted $X_i, i = 1, 2, \ldots, N$, a tuple $t = (x_1, x_2, \ldots, x_N)$ is viewed as a point in the N-dimensional space $X_1 \times X_2 \times \ldots \times X_N$. On the Cartesian plane, N copies of the vertical line, labeled $X_1, X_2, \ldots, X_N$, are placed equi-distant and parallel. These vertical lines are the axes of the *parallel coordinate* system for Euclidean N-dimensional space, shown in Figure 6.2. A tuple t is represented by the polygonal line whose N vertices are on the N vertical lines (axes) with x_i on $X_i, i = 1, 2, \ldots, N$.

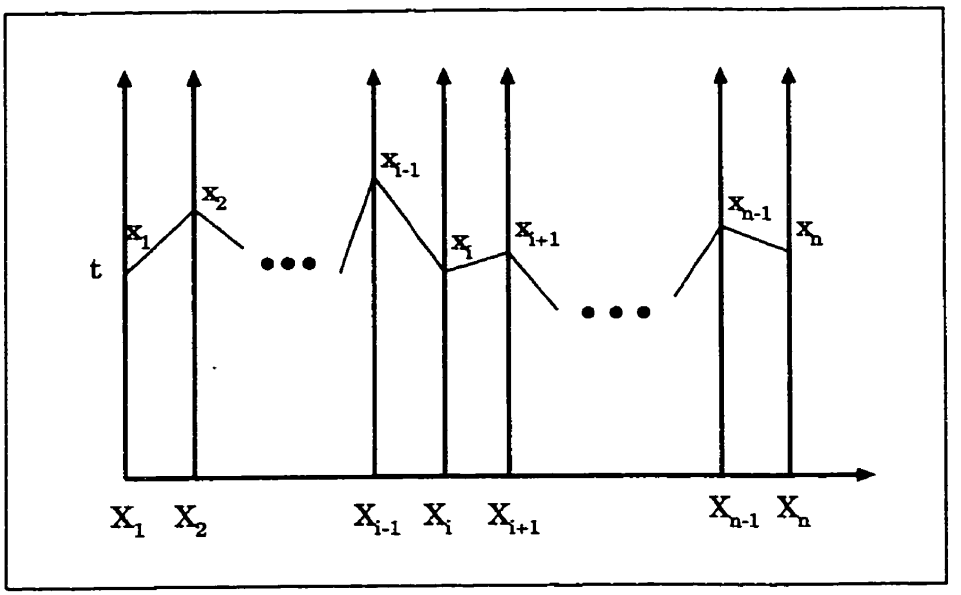


Figure 6.2: The *Parallel Coordinates* System

To visualize the raw data in the parallel coordinate system, the raw data needs to be scaled. Generally, N axes have the same height, especially when they are

displayed on the screen. Assume the axes in the display window is limited from m_p to M_p in pixel, and the i-th dimension has range from m_v to M_v. Then tuple t is scaled as follows: for $i = 1, 2, \ldots, N$, if X_i is a continuous dimension,

$$\bar{x}_i = m_p + \frac{M_p - m_p}{M_v - m_v}(x_i - m_v); \tag{6.1}$$

if X_i is a discrete dimension,

$$\bar{x}_i = m_p + \frac{M_p - m_p}{M_v - m_v + 2}(x_i - m_v + 1). \tag{6.2}$$

Here, the values of discrete dimensions (categorical attributes) must be first encoded as ordinal numbers. For example, encode the values $Color$={*red, green, blue, yellow*} into $\{0, 1, 2, 3\}$. The factors 1 and 2 in the discrete dimension scale are used to avoid two values are placed in the ends in the case that a dimension only contains two values.

The parallel coordinates visualization technique yields a graphical representation of multi-dimensional relations rather than just finite point sets. These presentations have a rigorous mathematical structure leading to algorithms for synthetic constructions and can be used to solve real world applications, such as Robotics, Statistics, Computation Geometry, and Air Traffic Control system [91].

For our purpose, the parallel coordinates can help us see the data distribution, identify outliers, and perceive interesting data tuples. The properties of the parallel coordinates technique aid to do data reduction, presented in the next section. In addition, we develop a visual representation of classification rules on the basis of the underlying parallel coordinates.

6.4 Data Reduction

In the CViz system, feature selection and tuple selection methods are provided. The feature selection is based on the rank of attributes according to their relevance to the class labels, while tuple selection can be randomly drawn and/or interactively chosen based on the user's perception.

6.4.1 Feature Selection

When the parallel coordinates technique is used to visualize the data, the number of dimensions[1] is limited by the size of display window. Thus, if there are too many features, the important ones with respect to the class labels must be selected. The approach used in CViz for selecting features is based on the RELIEF feature selection algorithm [100] and its extension RELIEF-F [102].

The underlying idea of the RELIEF algorithm is that good feature should differentiate between instances from different classes and should have the same value for instances from the same class. To measure this kind of "goodness", define the relevance degree of a feature A to class variable C as follows:

$$Rd(A) = P(t_1(A) \neq t_2(A) | t_1(C) \neq t_2(C)) - P(t_1(A) \neq t_2(A) | t_1(C) = t_2(C)), \quad (6.3)$$

where P represents probability, $t_i(A)$ denotes the value of A in tuple t_i, and $t_i(C)$ the class label of t_i, $i = 1, 2$. It means, the relevance degree of A with respect to C is the difference between the probability that any two tuples with different class labels have the same value of A and the probability that any two tuples with the same class label have the same value of A.

[1]Dimension, feature, and attribute are used interchangeably.

It can be seen that the higher the relevance degree of a feature, the more discriminant power the feature. The probabilities, however, are usually unknown in prior. To estimate the probabilities, a random sample of tuples is drawn. For each sampled tuple, calculate its distance from the nearest hit tuple in the same class and the nearest miss tuples from different classes[2], and use the distance to estimate *Rd*. The algorithm is described in Figure 6.3, assuming N features are denoted $A_1, A_2, \ldots, A_N$, and the sample size of tuples is m.

Procedure **for feature selection**

for $i = 1$ to N

$Rd(A_i) = 0$;

for $j = 1$ to m

randomly select a tuple t;

find one of hit tuples h nearest to t in the same class as t;

for each class label c not the same as the class label of t

find one of miss tuples s_c nearest to t in class c;

for $i = 1$ to N

$Rd(A_i) = Rd(A_i) - diff(A_i, t, h)/m + \sum_{c \neq t(C)}[P(c) \times diff(A_i, t, s_c)]/m$;

Figure 6.3: Procedure for Feature Selection Based on RELIEF

In Figure 6.3, the distance function $diff$ is defined as follows:

[2]The nearest hit tuple of a tuple t is the closest tuple to t among the tuples that belong to the same class as t. The nearest miss tuple of t in class c is the closest tuple to t among the tuples that belong to the class c, where c is not the same class as t.

for discrete attribute A,

$$diff(A,t,s) = \begin{cases} 0, & if\ t(A) = s(A), \\ 1, & if\ t(A) \neq s(A); \end{cases} \tag{6.4}$$

for continuous attribute A,

$$diff(A,t,s) = \frac{t(A) - s(A)}{max(A) - min(A)}, \tag{6.5}$$

where $max(A)$ and $min(A)$ are the maximum and minimum values of A, respectively.

The distance between two tuples t and s is the sum of differences over all attributes:

$$diff(s,t) = \sum_{i=1}^{N} diff(A_i,t,s). \tag{6.6}$$

The nearest tuple to t in class c is one with the lowest value of $diff(s,t)$ for all s in class c.

The class probability $P(c)$ can be estimated as the frequency that c occurs in the data set.

Once the relevance degree of each attribute with respect to the class label has been calculated, they are ranked in descending order of relevance degree, and the first M features are selected as the learning features, where M is determined by the size of display window and can also be specified by the user through the dialog window.

The CViz system also provides a method for interactive feature reduction. If one is not interested in a feature by looking into the distribution of data on the corresponding coordinate in the display window, he can directly remove the feature by "right double click (delete-click)" on the coordinate. For example, if most tuples

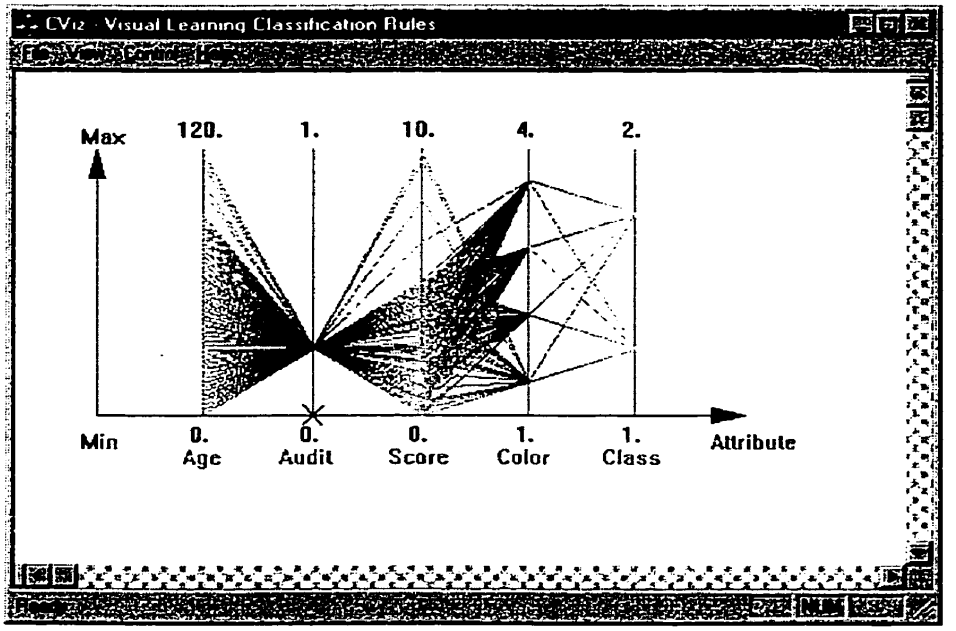


Figure 6.4: Removing Attribute *Audit*

concentrate on one value of attribute A, and few or none tuples have other values, then A can be removed by "delete-click". Which features are deleted depends on the user's intention and interests. Figure 6.4 shows an example of perception-based feature reduction, where attribute *Audit* has two possible values: 0 – pass, 1 – fail. From the figure, one can see that, in the training data set, all tuples have *Audit* of value *pass*. Therefore, attribute *Audit* provides no information for the classification task, and can be removed by "delete-click".

6.4.2 Tuple Selection

The CViz system develops two methods for tuple selection. The first one is randomly sampling in the case that too many tuples are included in the training data set. The size of the sample and the minimum number of tuples for each class label

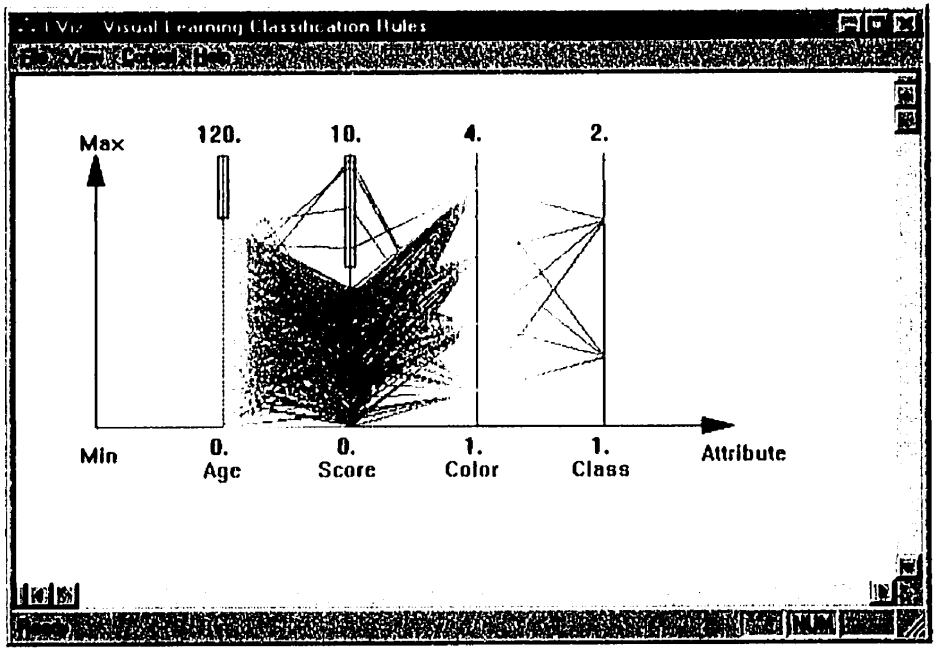


Figure 6.5: Cleaning Data by Removing Outliers

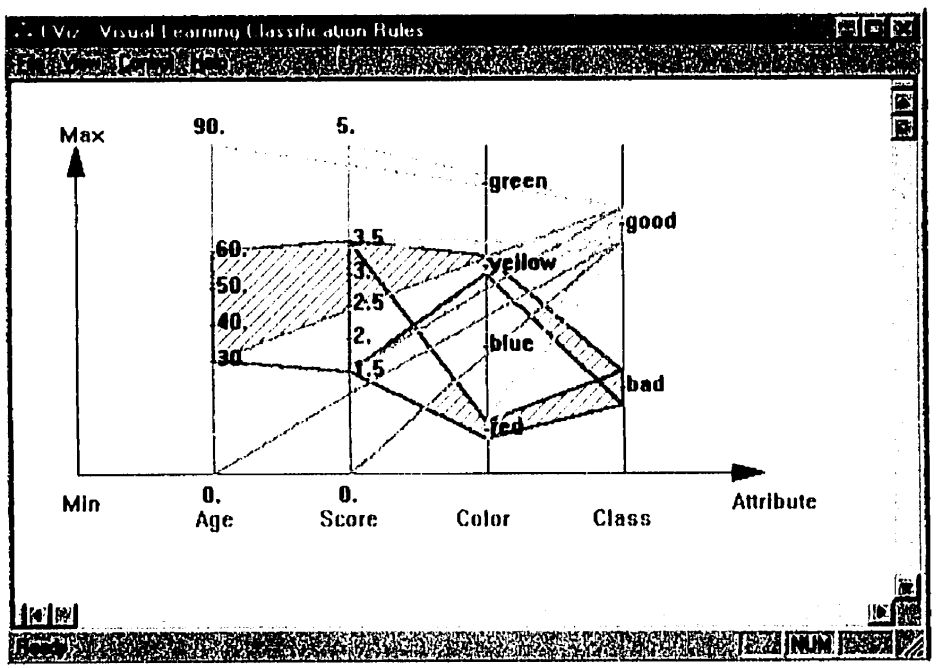


Figure 6.10: All Rules Discovered with Artificial Data Set

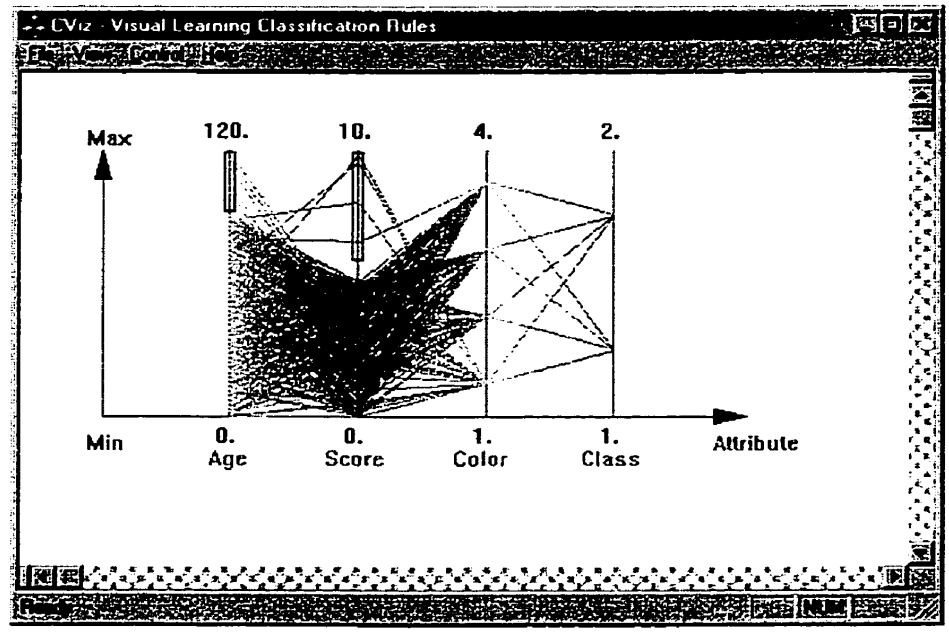


Figure 6.5: Cleaning Data by Removing Outliers

can be determined by the user through dialog window. If no sampling is required, the entire data set is used.

The second approach for tuple selection is visualization-based. Once one perceives that some tuples should be cleaned because they are outliers or not interesting, the intervals or the values of the attributes, depending on whether they are continuous or categorical, that identify outliers or uninteresting tuples can be specified and the tuples passing through the ranges or points are removed. Look at Figure 6.5, which is obtained from Figure 6.4 after attribute *Audit* is deleted. Very few polygonal lines (corresponding to tuples) fall in the upper quartile segment of feature *Age* or in the upper half segment of feature *Score*. Thus, these ranges may be noise. One can specify the noise ranges by using "rubber band". The tuples falling in the noise ranges are identified as outliers, and one can remove the noise ranges by "noise-click". Therefore, the tuples having attribute value in the noise

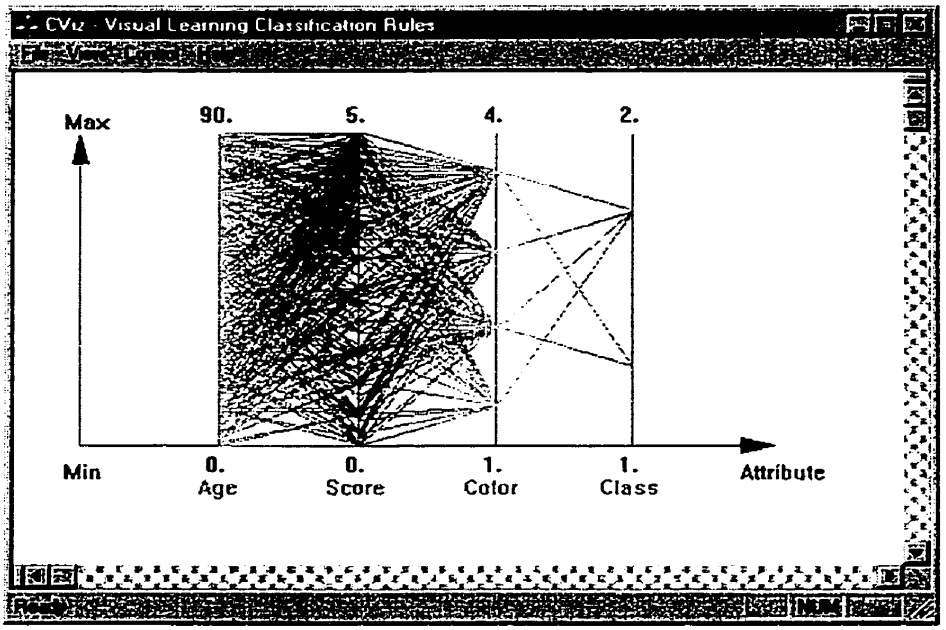


Figure 6.6: The Raw Artificial Data

range are cleaned. The cleaned data set has feature *Age* with range $[0, 90]$, and feature *Score* with range $[0, 5]$. The result is illustrated in Figure 6.6.

6.5 Data Preprocess

In the CViz system, the data preprocess consists of two aspects, missing values handling and continuous attribute discretization. Both approaches are algorithm-based and embedded in CViz.

6.5.1 Missing Values Handling

There are two kinds of missing values, condition attribute values, the decision attribute value. For missing condition attribute values, CViz chooses the most

frequent values with respect to the same class labels to replace them. Formally, assume that tuple t has a missing value for attribute A, and t is labeled as class c. Count each individual value of A occurring in the training data set. Let $D(A)$ be the set of such values. For each $a \in D(A)$, count the number of tuples that have value of a for attribute A and are labeled as c. The value with the highest count replaces the missing value of attribute A in tuple t.

If the decision value (class label) is missing, then the corresponding tuple is removed from the training data set, because this kind of noise can not be processed in ELEM2.

6.5.2 Discretizing Numerical Attributes

The CViz system provides three methods to discretize numerical attributes, *equi-length*, *bin-packing based equi-depth*, and *entropy-based* approaches. The former two are the same as those in the AViz system, described in previous chapter.

The third approach uses EDA-DB (Entropy-based Discretization According to Distribution of Boundary points) method [12]. Unlike solely entropy-based discretization methods (such as the method based on the Minimum Description Length Principle [52]), EDA-DB first divides the value range of the attribute into several *big* intervals and then selects in each interval a number of cut-points based on the entropy calculated over the current entire data set. The number of cut-points selected for each interval is determined by estimating the probability distribution of the boundary points over the data set. The maximum number of selected cut-points is determined by the number of class labels and the number of distinct observed values for the continuous attribute.

Let l be the number of distinct observed values for a continuous attribute A, b

be the total number of boundary points[3] for A, and k be the number of classes in the data set. Then the discretization of A is described in the procedure illustrated in Figure 6.7.

6.6 Learning Classification Rules

The CViz system exploits ELEM2 as the approach to discovering knowledge. ELEM2 is a rule induction system that learns classification rules from a set of data [11]. Given a set of training data, ELEM2 sequentially learns a set of rules for each of classes in the data set. To induce rules for a class C, ELEM2 conducts a general-to-specific heuristic search over a hypothesis space to generate a disjunctive set of propositional rules. ELEM2 uses a *sequential covering* learning strategy; it reduces the problem of learning a disjunctive set of rules to a sequence of simpler problems, each requiring that a single conjunctive rule be learned that covers a subset of positive examples.

Once continuous attributes are discretized, all attributes can be processed in the same way. Consider an *attribute-value* pair av, which is a relation between the attribute and its values or value intervals. The relation may be one of $=, \neq, \leq, >, \in$, e.g., $a = v$, $a \in \{1, 3, 5, 7\}$, $a \leq 12$. A classification rule is a set of such pairs which are connected with logical operators AND, OR and NOT.

For a given class label, ELEM2 generates decision rules by performing a general-to-specific search in a hypothesis space. Then it iteratively evaluates attribute-value pairs and selects the most significant one. This procedure begins by considering

[3]Fayyad and Irani [52] proved that the value $t(A)$ for attribute A that minimizes the class entropy $E(A, T_A; S)$ for a training set S must always be a value between two examples of different classes in the sequence of sorted examples. These values are called boundary points.

Procedure **for EDA-DB discretization**

Calculate: $m = max\{2, k * log_2(l)\}$

Estimate the probability distribution of boundary points:

Divide the value range of A into d intervals, $d = max\{1, log_e(l)\}$

Calculate the number b_i of boundary points in each interval iv_i,

where $i = 1, 2, \cdots, d$ and $\sum_{i=1}^{d} b_i = b$

Estimate the probability of boundary points in each interval iv_i

$(i = 1, 2, \cdots, d)$ as $p_i = b_i/b$

Calculate the quota q_i of cut-points for each interval iv_i $(i = 1, 2, \cdots, d)$ according to m and the distribution of boundary points: $q_i = p_i * m$

Rank the boundary points in each interval iv_i $(i = 1, 2, \cdots, d)$ by increasing order of the class information entropy of the partition induced by the boundary point. The entropy for each point is calculated globally over the entire data set. If we are given a set S of instances, an attribute A, and a cut-point T, the class information entropy of the partition induced by T, denoted as $E(A, T; S)$ and defined as

$E(A, T; S) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2)$,

where $Ent(S_i)$ is the class entropy of the subset S_i, defined as

$Ent(S_i) = -\sum_{j=1}^{k} P(C_j, S_i) log(P(C_j, S_i))$,

where there are k classes $C_1, \cdots, C_k$ and $P(C_j, S_i)$ is the proportion of examples in S_i that have class C_j.

For each interval iv_i $(i = 1, 2, \cdots, d)$, select the first q_i points in the above ordered sequence. A total of m cut-points are selected.

Figure 6.7: EDA-DB Discretization of Continuous Attribute

the most general rule precondition (e.g., the empty test that matches every training example), then greedily searches for an attribute-value pair that are most relevant to the class label C according to the following attribute-value pair evaluation function:

$$SIG_C(av) = P(av)(P(C|av) - P(C)), \tag{6.7}$$

where av is an attribute-value pair and P denotes probability [11]. The range of SIG_c is $(-1, 1)$. If the value is positive, then the higher the value, the more relevant the attribute-value pair av to the class label; if the value is negative, then the lower the value, the more relevant the negative pair $\neg av$ to the class label.

Comparing their significance values for all possible attribute-value pairs, choose the pair with the highest absolute value and add it to the rule precondition as a conjunct. The process is repeated by greedily adding a second attribute-value pair, and so on, until the hypothesis reaches an acceptable level of performance. In ELEM2, the acceptable level is based on the rule consistency principle: it forms a rule that is as consistent with the training data as possible.

ELEM2 develops a concept, *unlearnable region*, to handle the problem of inconsistent training data. Two examples are inconsistent if they have identical condition attribute values but different class labels. The *unlearnable region* of a class label C is defined as the set of positive examples X of C satisfying $P(C|X) < P(C)$. The unlearnable region can be used to avoid the inconsistent examples to be involved in the construction of classification rules.

The *consistent* rule obtained may be a small disjunct that overfits the training data. ELEM2 provides a method to *post-prune* the rule after the initial search. For this purpose, ELEM2 defines rule quality value for a rule R as follows:

$$log \frac{P(R|C)(1 - P(R|\bar{C}))}{P(R|\bar{C})(1 - P(R|C))}. \tag{6.8}$$

The rule quality value measures the extent to which a rule R can discriminate between the positive and negative examples of class label C. ELEM2 then checks each attribute-value pair in the rule in the reverse order in which they were selected to see if removal of the attribute-value pair will decrease the rule quality value. If not, the attribute-value pair is removed and the procedure checks all the other pairs in the same order again using the new rule quality value resulting from the removal of that attribute-value pair to see whether another attribute-value pair can be removed. This procedure continues until no pair can be removed.

After rules are induced for all the classes, the rules can be used to classify new examples. The classification procedure in ELEM2 considers three possible cases when a new example matches a set of rules.

- *Single match.* The new example satisfies one or more rules of the same class. In this case, the example is classified to the class indicated by the rule(s).
- *Multiple match.* The new example satisfies more than one rule that indicates different classes. In this case, ELEM2 activates a conflict resolution scheme for the best decision.
- *No match.* The new example is not covered by any rule. In this case, the decision score is calculated for each class and the new example is classified into the class with the highest decision score.

6.7 Rule Visualization

To visualize classification rules, the CViz system represents a rule as a strip, called *rule polygon*, which covers the area that connects the corresponding attribute values.

A classification rule induced by the ELEM2 algorithm is a logical statement that consists of two parts: the *condition part* and the *decision part.* The *decision* part is simply the class label. The *condition* part is composed of a set of simple relations, each of which corresponds to a specific attribute.

(1) *For categorical attributes :*

$A = a_1, a_2, \ldots,$ *or* a_n, meaning attribute A has any values of $a_1, a_2, \ldots,$ *or* a_n, where $n >= 1$;

$A\ != a_1, a_2, \ldots,$ *or* a_n, meaning attribute A has any values except $a_1, a_2, \ldots,$ *and* a_n, where $n >= 1$.

(2) *For continuous attributes :*

$A <= a$, meaning attribute A has any values between the minimum and a;

$a < A$, meaning attribute A has any values between a and the maximum;

$a_1 < A <= a_2$, meaning that attribute A has any values between a_1 and a_2.

For example, the following is a classification rule obtained in our experiment with an artificial data set, and is visualized in Figure 6.11:

$$(30 < Age <= 60)(1.5 < Score <= 3.5)(Color = red\ or\ yellow) \Rightarrow Class = bad.$$

It means that if numerical attributes *Age* and *Score* have values between 30 and 60 and between 1.5 and 3.5, respectively, and categorical attribute *Color* has value *red* or *yellow*, then the instance will be in class *bad.* The corresponding *rule polygon* consists of two polygons which are enclosed by the points that correspond to value *red* and *yellow* on coordinate *Color*, respectively, the points correspond to 30 and

Procedure **for drawing rule polygons**

preCond = first condition;
repeat
  postCond = next condition;
  for each *or* component of *preCond* **do**
    if *preCond* corresponds to a categorical attribute
    then compute the points p_0, p_1 according to the specific categorical value;
    else compute the points p_0, p_1 according to a numerical interval;
    for each *or* component of *postCond* **do**
      if *postCond* corresponds to a categorical attribute
      then compute the points p_2, p_3 according to the specific categorical value;
      else compute the points p_2, p_3 according to a numerical interval;
      draw a small polygon with four points p_0, p_1, p_2, and p_3;
  preCond = *postCond*;
until all conditions are processed;
draw the small polygon from last condition to the decision attribute.

Figure 6.8: Procedure for Drawing Rule Polygons

60 on coordinate *Age*, the points that correspond to 1.5 and 3.5 on coordinate *Score*, and the point corresponding to class *bad* on the decision coordinate.

The *rule polygon* is constructed by a *rule polygon generating* procedure, which is described in Figure 6.8.

To distinguish between *positive* conditions and *negative* conditions represented with the unequal symbol (!), the positive condition is drawn with a backward hatch, while the negative condition is drawn with a forward hatch. In addition, the rule accuracy and the rule normalized quality are used to calculate the color of the rule polygon. The more accurate the rule, the redder the polygon, while the lower the rule quality, the brighter the polygon,

6.8 CViz Implementation and Experiment

The CViz system has been implemented in Visual C++ 6.0. The data preparation is accomplished by choosing a data file and an attributes file which describe the attributes, including attribute name, type, length, position in the tuple, domain, etc. This procedure is implemented in dialog windows (under the file menu and setting menu). The steps of visualizing and discovering knowledge are controlled by the *control menu*, which consists of five steps.

CViz has been experimented with several data sets from UCI Repository of Machine Learning Databases [126], including IRIS, Monk's, etc. and also with artificial data sets. The IRIS data are represented by only continuous attributes except the class label, while the Monk data set consists of only discrete attributes. The artificial data set is designated to contain both numerical and discrete attributes.

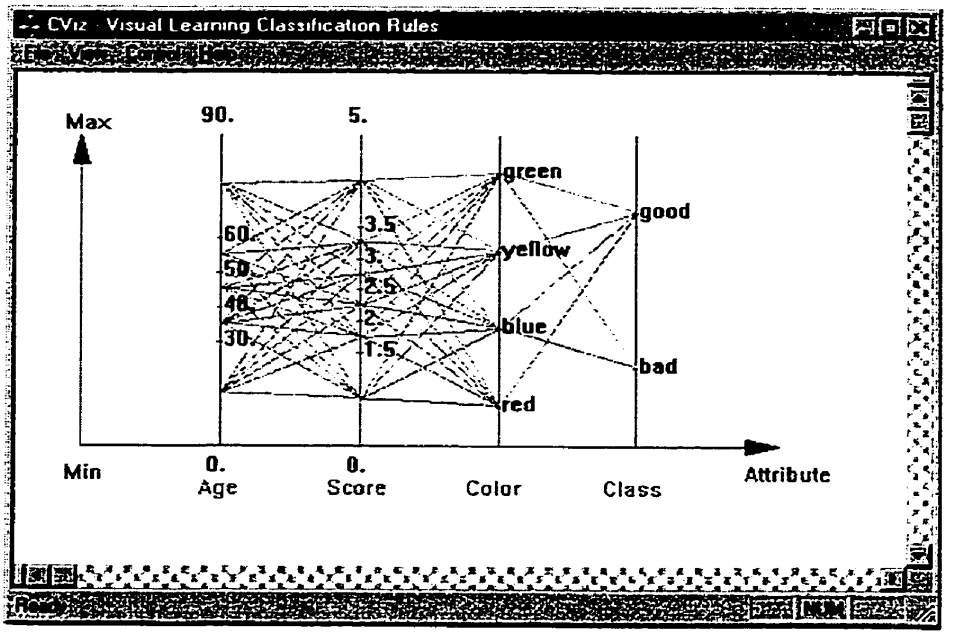


Figure 6.9: The Numerical Attributes Discretization of the Artificial Data Set

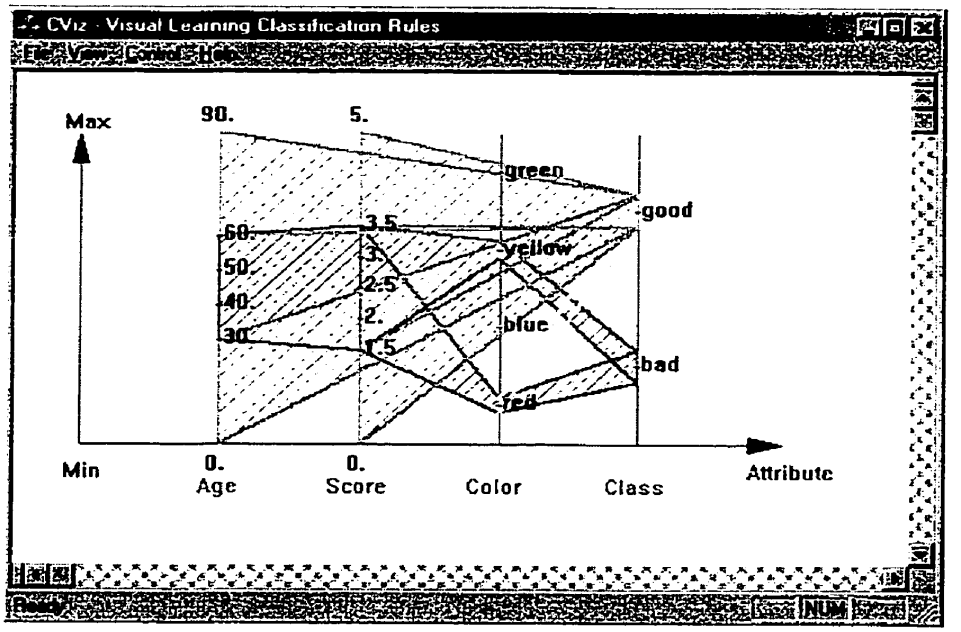


Figure 6.10: All Rules Discovered with Artificial Data Set

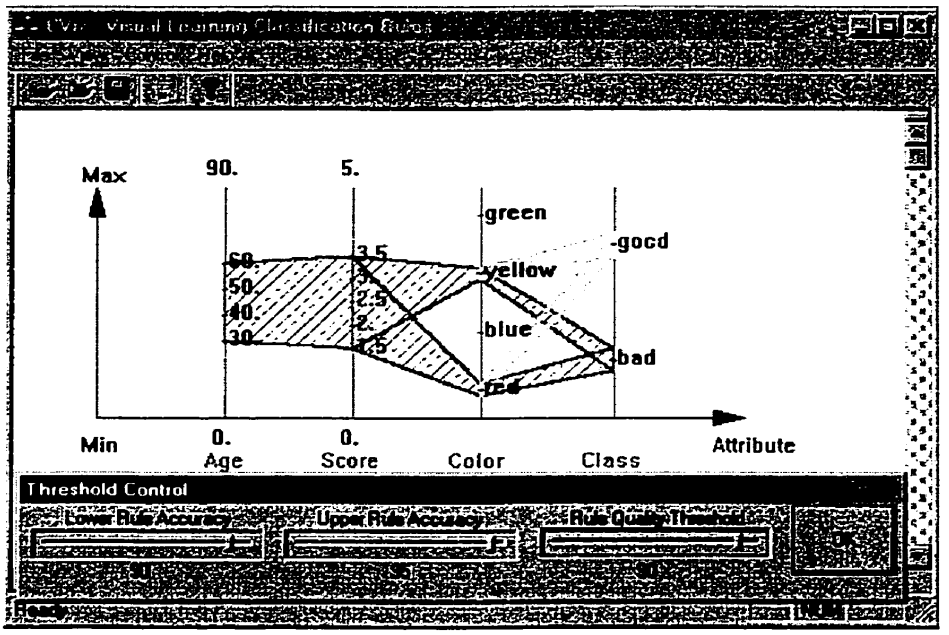


Figure 6.11: The Rules from Artificial Data with Accuracy Between 90% And 95%

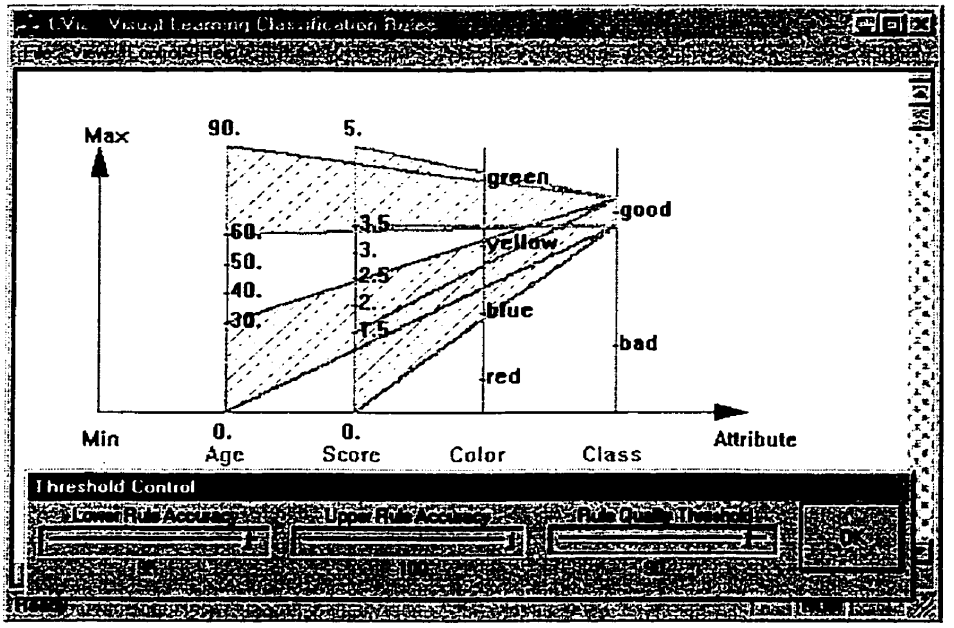


Figure 6.12: The Rules for the *Good* Class with Accuracy Between 95% And 100%

6.8.1 Experiment with Artificial Data

An artificial data set is designed to involve two numerical attributes *Age* and *Score*, two condition categorical attributes *Audit* and *Color*, and one decision categorical attribute *Class*. *Age* is of integer type and has range $[0, 120]$. *Score* is of real-valued type and has range $[0.0, 10.0]$. *Audit* has two categorical values *pass* and *fail*. *Color* has four discrete values: *red, blue, yellow*, and *green*. The decision attribute has two class labels: *good* and *bad*. The data set is designed as follows:

- The training data set contains 160 tuples.
- The underlying relationship between the class labels and the condition attributes in the data set is:

```
If (30<Age<=60) and (1.5<Score<=3.5) and (Color=red or yellow)
Then Class=bad
Otherwise Class=good.
```

- All tuples have the same value for attribute *Audit*, thus, *Audit* provides no information for classification task, and can be removed by perception-based "delete-click", before the learning starts.
- 10 noise tuples are included in the training data set, and the noise range for *Age* is $(90, 120]$, and the noise range for *Score* is $(5.0, 10.0]$. After the noise ranges are cleaned, 150 tuples with 3 condition attributes are effective for learning classification rules by using the ELEM2 algorithm.

Figures 6.4 and 6.5 illustrates the feature reduction and tuple cleaning based on the user's perception and intention. Figures 6.6 through 6.12 illustrate the rest

of our experiment with the artificial data set. Figure 6.6 visualizes the cleaned artificial data, and Figure 6.9 is the result of discretizing the numerical attributes *Age* and *Score* with EDA-DB method [12]. Figure 6.10 illustrates all discovered rules, including one rule for class *bad* and three rules for class *good*, which are the same as our designation and correspond to the following three conditions:

```
Age<=30 or Age>60,
Score<=1.5 or Score>3.5,
Color!=(red or yellow).
```

Figure 6.11 shows, we have only two rules with accuracy between 90% and 95% and quality greater than 90%, one for class *good*, one for class *bad*. Figure 6.12 illustrates the rules for class *good* with accuracy greater than 95%.

6.8.2 Experiment with the UCI Reporsitory Data Sets

The IRIS data set contains three classes of iris plants with 50 objects belonging to each class. One of the iris types is linearly separable from the other two while the other two are not linearly separable from each other. Each object is described by four numerical attributes, *sepal-length, sepal-width, petal-length*, and *petal-width*. Three classes are IRIS Setosa, IRIS Versicolor, and IRIS Virginica.

Figures 6.13 through 6.17 demonstrate the experimental results with the IRIS data set. Figure 6.13 is the raw data visualization, while Figure 6.14 is the visualization of discretized numerical attributes with EDA-DB method. From Figure 6.14, we see that four numerical attributes are discretized into different number of intervals with different width. For example, attribute *petal-width* is discretized into eight disjoint intervals, and the boundary points are 0.0, 0.6, 1.3, 1.4, 1.5, 1.6,

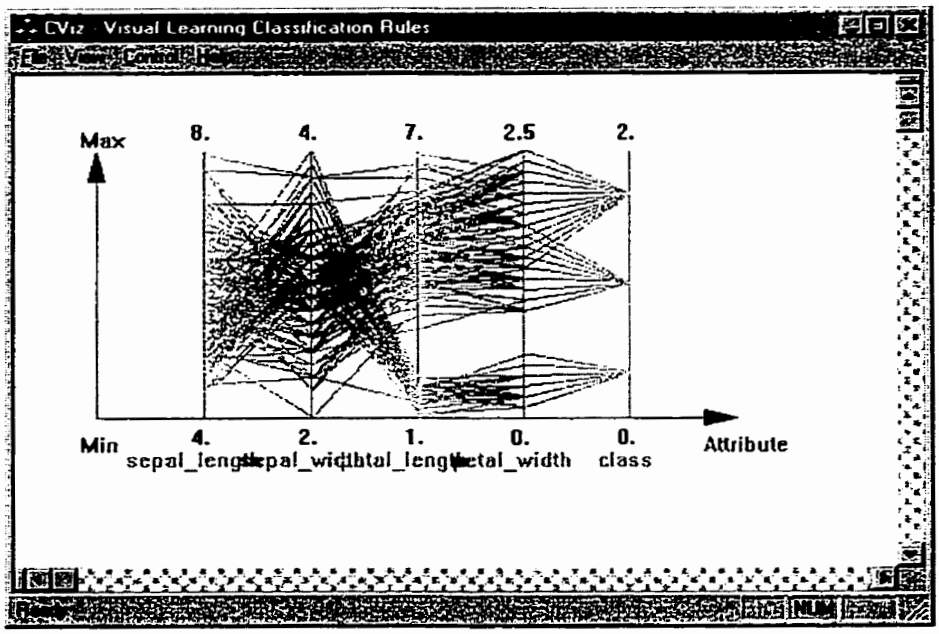


Figure 6.13: The Raw IRIS Data

1.7, 1.8, and 2.5, respectively. Figure 6.15 shows all rules that ELEM2 learns from the IRIS data sets, which involves eight classification rules, including four rules for class *3*, three rules for class *2*, and one rule for class *1*, respectively. Figure 6.16 illustrates all rules for class *2* discovered from the IRIS data set, and the rules for class *1* and class *3* discovered from the IRIS data set have the similar display form. Figure 6.17 shows the rules for class *2* with accuracy greater than 90% and quality greater than 85%. From Figure 6.17, it is easy to see we obtain two rules that satisfy the accuracy and quality specification through sliders, which are

$$< sepal\_length > 5.8 >< sepal\_width > 3.0 >$$
$$< petal\_length <= 4.8 > \quad \Rightarrow \quad Class = 2.$$

and

$$< petal\_length \in (1.9, 4.8] >< petal\_width < 1.6 > \quad \Rightarrow \quad Class = 2.$$

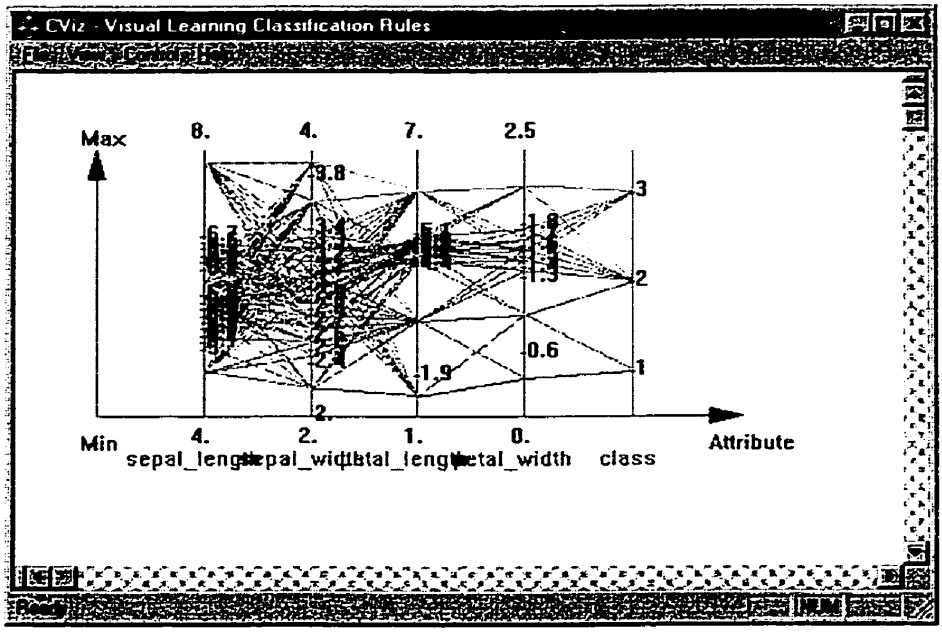


Figure 6.14: The IRIS Attributes Discretization with EDA-DB Method

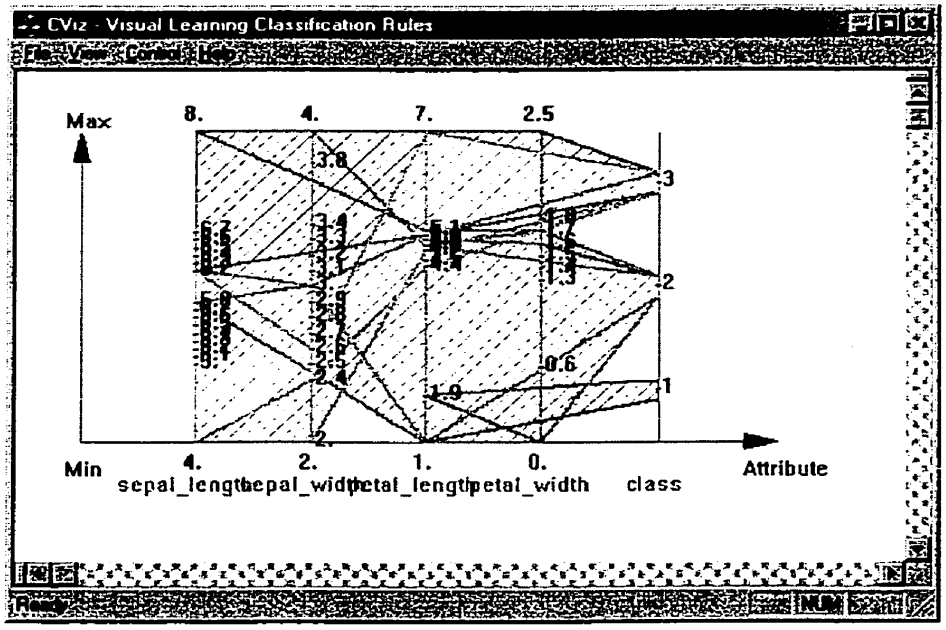


Figure 6.15: The Rules for the *IRIS* Data

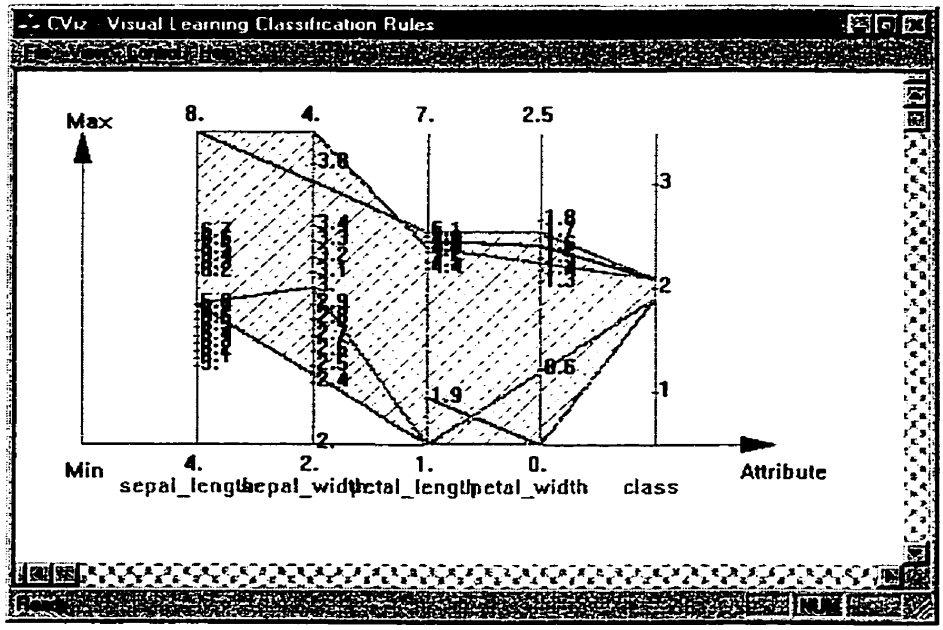


Figure 6.16: The Rules for *IRIS* Class 2

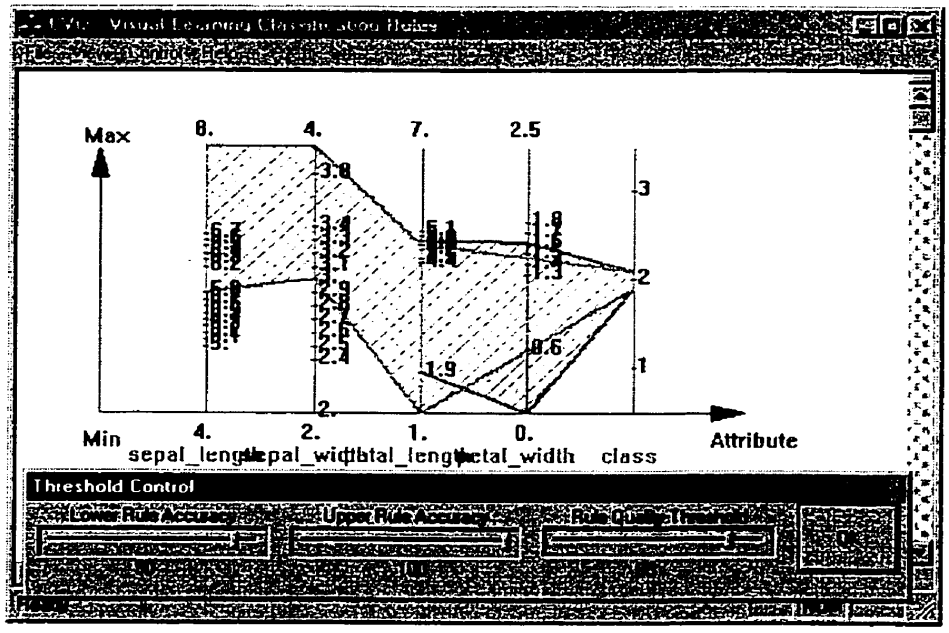


Figure 6.17: The Rules for *IRIS* Class 2 with Quality Greater Than 85%

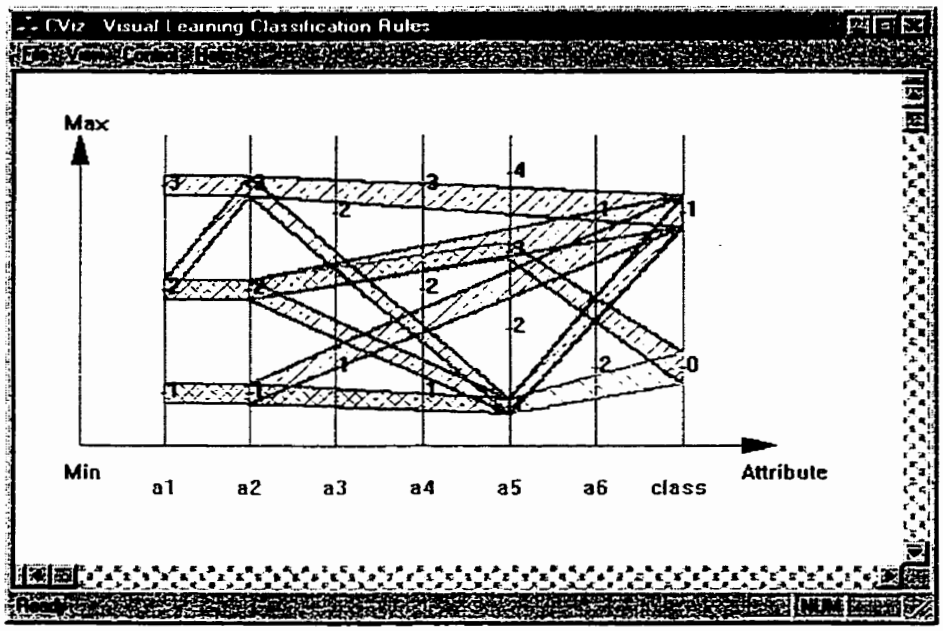


Figure 6.18: The Rules for the `Monk-1` Data

We also experiment the CViz system with the Monk-1 data set, which consists of a set of robots described by six attributes: head shapes, body shapes, facial expressions, objects being held, jacket colors, and whether or not the robot is wearing a tie. The decision attribute describes whether a robot belongs to a positive or negative class. This data set contains 432 examples. The rules obtained are illustrated in Figure 6.18, from which we can see there are nine classification rules where five rules are for class *0* (negative class) and four rules for class *1* (positive class).

6.9 Discussion

CViz is an interactive system for visualizing and learning classification rules. It visualizes the raw data on a parallel coordinates system. The user can interactively

reduce the data set horizontally and vertically by removing irrelevant attributes and/or attribute values. The user can also interactively select his/her favorite approaches to discretizing numerical attributes. The discretized attributes are treated as categorical ones and each interval corresponds to a discrete value. The ELEM2 induction algorithm is used to learn classification rules which are displayed in visual forms. The user can interactively choose a class to view the corresponding rules. Classification rules may have a complex logical form. In our implementation, we emphasize the human-machine interaction, since we believe that interactive visualization plays an important role in the process of discovering knowledge. Our experimental results have also demonstrated that it is useful for users to understand the relationships among data and to concentrate on the meaningful data to discover knowledge. The capability of CViz has been expanded so that the user can interactively specify the rule accuracy threshold and/or the rule quality threshold, which might be used to limit the search space for final rules.

It would be easy to adapt CViz to other rule-based learning systems because the CViz system just encompasses the learning algorithm as a learning step of the visual KDD process, and the difference between rule-based learning systems is that each of them uses a different learning algorithm. It is not easy, however, to adapt CViz to learning systems other than rule-based ones because CViz is based on the parallel coordinate technique which may not be plausible for representing decision trees, attribute correlations, and neural networks.

The picture obtained from a given data set depends on the order of attributes. Different permutations of attributes may generate different pictures. Therefore, the user may select different features and tuples. A new operator for permuting attributes should be provided for the user to see the data distribution from different ways so as to decide how to reduce and clean the data set.

Chapter 7

CVizT: An Interactive System for Building Classifiers

7.1 Introduction

Building classifiers from large multivariate data sets requires finding a way to assign a new object to one of a number of predetermined classes based on a set of measurements. Many approaches have been proposed for building classifiers in statistics, machine learning and data mining communities [123]. Usually, a classifier consists of, or is equivalent to, a set of classification rules, and each rule includes a condition part and a decision part. The decision part specifies the class label, while the condition part is composed of attribute-value pairs. For *categorical* attributes, the value in the pairs is a value of the attribute. However, for *continuous* attributes, the value is an *interval* of the attribute domain. Continuous attribute discretization plays an important role in building classifiers because the boundaries must be specified to build classification rules. Most classification approaches are based on

sophisticated statistical algorithms [127]. These algorithms, however, are not easily adapted to different data sets and by only changing input parameters may result in widely different requirements.

Alternatively, visualization-based approaches for building classifiers have also been developed recently [80], which stress the human-machine interaction and include a role for the user's perception. Geometrical representations of the data set can be very helpful in this regard because these representations portray the data items in a two- or three-dimensional space, and thus help the user gain better insight into multidimensional data [32, 138]. Domain knowledge and expert perception can also help to construct the classifiers.

Rather than inducing classification rules by sophisticated algorithms, we present a novel approach for interactively building classifiers by visualizing the data sets and classification rules according to the RuleViz model. This approach is fully interactive. It builds classifiers from large multivariate data sets based on the Table Lens [138], a multidimensional visualization technique, and appropriate interaction capabilities. Constructing classifiers is an interaction with a feedback loop and allows us to integrate the domain knowledge of experts and perception of the user on demand during the classifier construction. Both continuous and categorical attributes are processed uniformly, and continuous attributes are partitioned on the fly.

In this chapter, we describe the CVizT system and its components, introduce the visualization technique Table Lens, and discuss its applications in visualizing data. We also propose the approach for feature selection and tuple selection to clean the data and present the process of interactively constructing classifiers, including discretizing continuous attributes on the fly, rule formation, rule evaluation, and rule visualization. Finally, our experimental results with UCI repository data sets

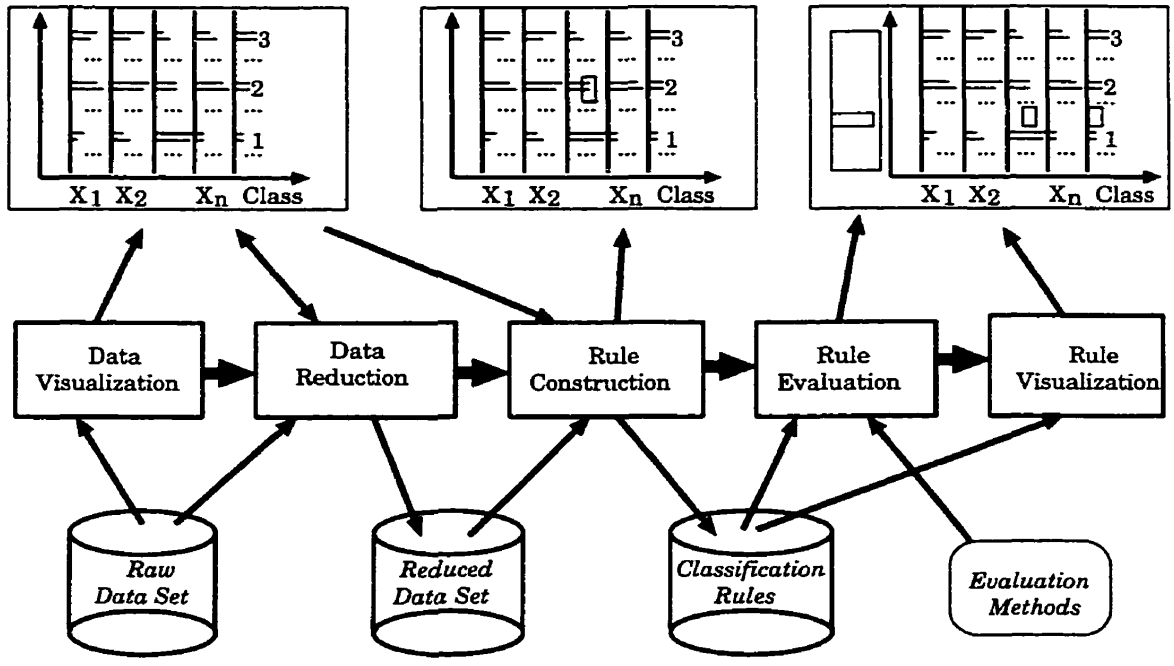


Figure 7.1: The CVizT System

are reported and the characteristics of the CVizT system are analyzed.

7.2 The CVizT System

The CVizT system developed according to the RuleViz model consists of the following components: data visualization, data reduction, rule construction, rule evaluation, and rule visualization, shown in Figure 7.1.

The raw data are visualized using the Table Lens, a visual representation technique of data tuples, where each tuple is represented as a row in the picture, which in turn is divided into columns corresponding to attributes of the data set. The attribute value is drawn as a color line in the corresponding column with the length proportional to the value. Users can sort the tuples according to the specified at-

tributes and use other features of the Table Lens to view the data distribution and the relationships between attributes from different angles so as to perceive the classifications of the data tuples.

Based on these perceptions, the user can remove some attributes that do not provide useful or helpful information to the user. In addition, attributes that correlate closely enough can be aggregated.

After selecting the attributes, the user can iteratively construct classification rules until all raw data are covered by a set of rules. To construct a classification rule, the condition attributes are chosen one by one. For each of the chosen attributes, the remaining data tuples are sorted with respect to it, and either a value or an interval of the chosen attribute is specified depending on whether it is continuous or categorical attribute. The continuous attributes need to be discretized, which is done interactively by the user by perceiving the data distribution.

Each constructed classification rule is evaluated according to the evaluation criteria. The *bad* rules can be removed or reconstructed.

Finally, the classification rules with their evaluation are visualized on a variant of the Table Lens, where a rule is visually represented as a set of rectangles, each rectangle corresponding to a condition attribute and sitting in the corresponding column with the height representing its *coverage* and the length representing its value or interval. The metric of the rule (prediction accuracy) is displayed on the right of the rule.

In the rest of this chapter, each component is described in a greater detail. The experimental results with some UCI data sets will be reported and the performance of the CVizT system is discussed.

7.3 Data Visualization Based on the Table Lens

In order to support interactive construction of classifiers, the raw data set and the intermediate result are visualized with the **Table Lens** for the user to observe and control the construction process. Basically, the Table Lens integrates the relational data table or spreadsheet with graphical representations to support browsing of the values for hundreds of tuples and tens of columns on a typical workstation display [138]. Each tuple is represented as a row in the graph, and each attribute (variable) is represented as a column. Figure 7.2 illustrates the Table Lens that visualizes the IRIS flower data set, which consists of five attributes that are shown in the columns. The last variable is *class label*. The attribute values of tuples are drawn as lines proportional to the width of the corresponding columns. Assume attribute A has maximum and minimum values m and M, respectively. The width of the column corresponding to A is L, and a tuple t has value a of A. Then the length of the line corresponding to t in column A is determined to be

$$l(t) = \frac{a - m}{M - m} \times L \tag{7.1}$$

pixels.

Particular rows and columns can be assigned variable widths according to the number of points in attribute domains. In Figure 7.2, however, all columns have the same width. Figure 7.2 also shows the range of attributes (maximal and minimal values) above the columns.

The Table Lens also provides a small set of direct manipulation operators for discovering correlations among the observed variables, such as finding the attribute mean, standard deviation, variables (columns) permutation, etc.[1]

[1]Most operations are defined for multivariate data analysis. For more information about these operations, see [134, 138]

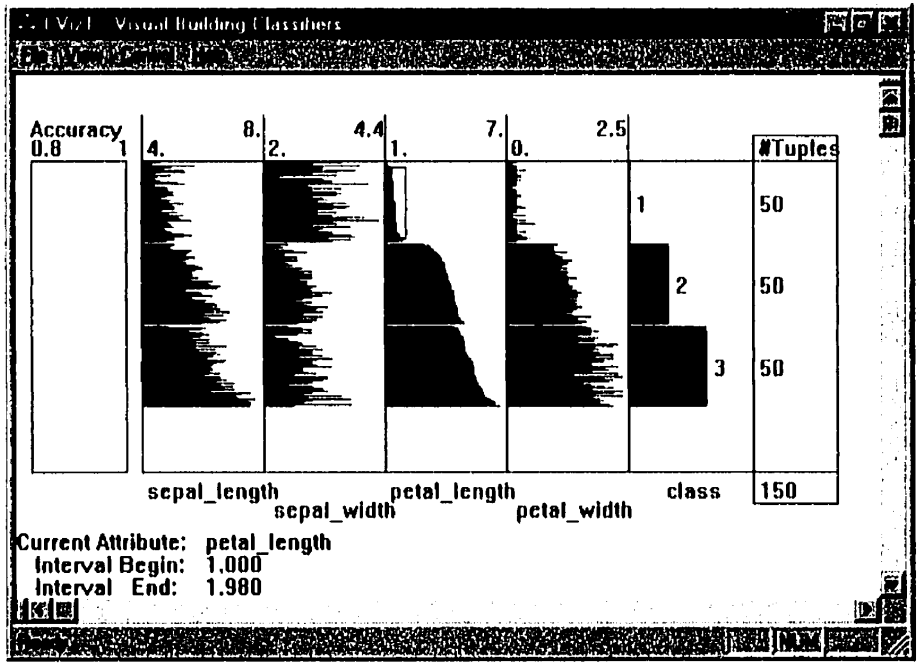


Figure 7.2: Table Lens Visualizing the IRIS Data Set

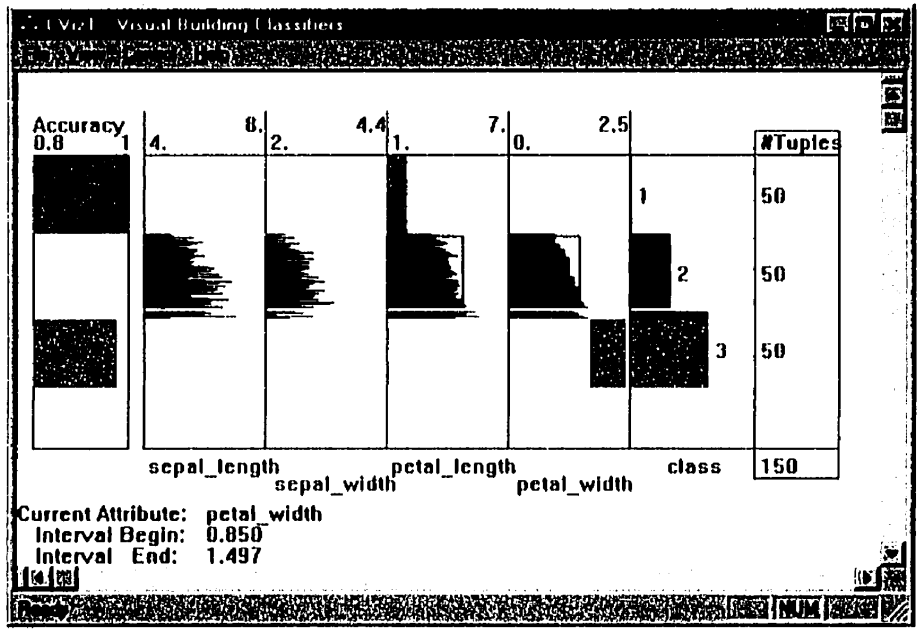


Figure 7.4: Table Lens Visualizing the Classification Rules

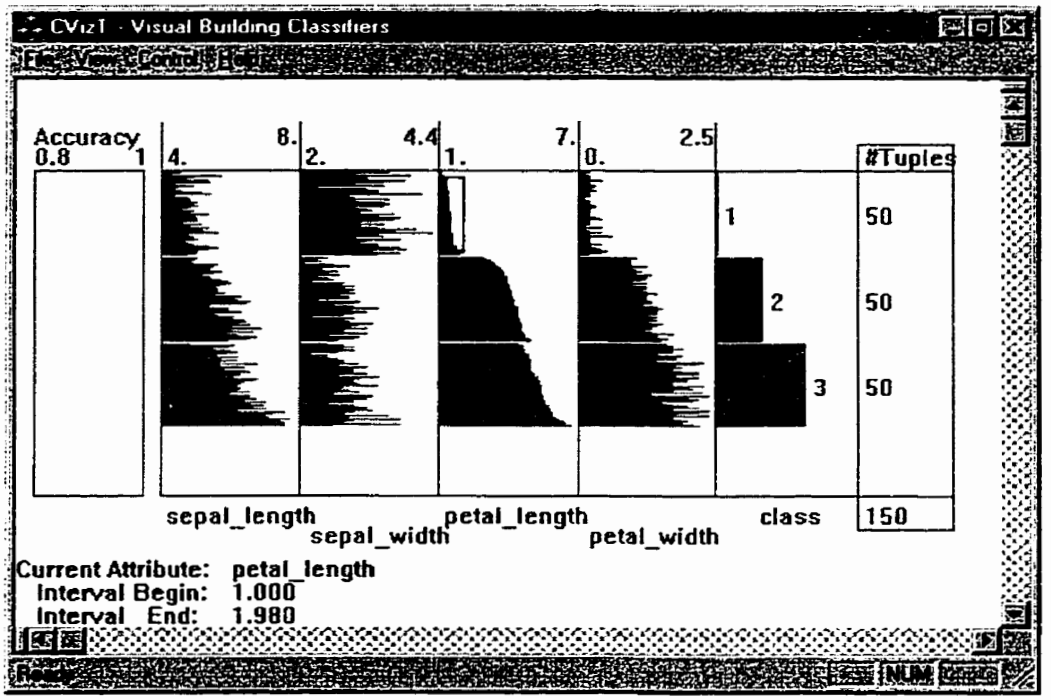


Figure 7.2: Table Lens Visualizing the IRIS Data Set

Sorting columns is a major operation on the Table Lens. Once a column is sorted, properties of the batch of values can be estimated by graphical perception and some amount of display manipulation, for example, the central values in the middle and extreme values on the ends of the column. In addition, the correlations among the columns can be observed if other columns are also sorted. In our method, tuples are aggregated in terms of their class labels, and the number of tuples in each class is displayed on the right side. From Figure 7.2, one can see that the IRIS data set contains three classes labeled as 1, 2, and 3, and once the data tuples are sorted in ascending order according to the *class* values, attributes *petal_length* and *petal_width* are roughly sorted in the same order. The other two attributes, however, have not been observed such strong correlations with *class* by perception. Another useful observation from Figure 7.2 is that *petal_length* and *petal_width* are

correlated strongly. The strongly correlated features can be aggregated together to reduce the number of features.

7.4 Data Reduction

As discussed before, the raw data can be reduced horizontally and vertically by selecting tuples and features, respectively. Let us consider these selections in the CVizT system.

Usually, the display window for the Table Lens is always limited in size. Each row corresponds to a tuple and the number of rows is usually less than 1000. If the window is designed to be scrollable, the scrolled rows will be hidden so that the user can not observe the distribution of the entire data set at the first glance. Therefore, the number of tuples displayed in the Table Lens is better to be limited in a range less than 1000. Our experiments show the number of tuples between 150 and 500 can achieve good results. To select tuples, random sampling is made by the CVizT system if the raw data set is large.

On the other hand, the number of columns (attributes) in the Table Lens is also limited because of the width of the display window. According to the Table Lens, the length of the line in each column is proportional to the width of column in terms of the corresponding attribute (determined by the minimum and maximum values of the attribute). If the number of attributes is large, the average width of each column must be small. In this case, different points may be mapped to the same length and can not be distinguished when they are displayed in the Table Lens.

Additionally, some attributes may be correlated very strongly. For example,

from Figure 7.2 one can see that *petal_length* and *petal_width* are correlated strongly. Thus, one of them could be removed or combined.

The CVizT system provides a *feature delete* mechanism. How many features and which features should be deleted, however, are determined by the user.

7.5 Interactive Rule Construction

To construct a classification rule, one can interactively specify the intervals for the attributes strongly correlated to the class labels to form the condition part of the rule. For example, from Figure 7.2, one can observe that the tuples belonging to *class 1* have *petal_length* and *petal_width* values concentrating on the left ends of the domains. Thus, one can conclude that

```
If the petal_length or petal_width of a tuple are around
     the minimum values,
Then it belongs to class 1.
```

However, we still have a question that is to determine how the *petal_length* or *petal_width* is *around* the minimum value. The user can specify an interval of the variable to define *around*. Generally, this kind of attribute intervals is used to define the condition part of a classification rule. In the CVizT system, the user can click the selected column and use a "rubber band" to draw an area to represent the attribute intervals.

Building classifiers is to interactively and iteratively construct classification rules. This process is illustrated in Figure 7.3 and described below.

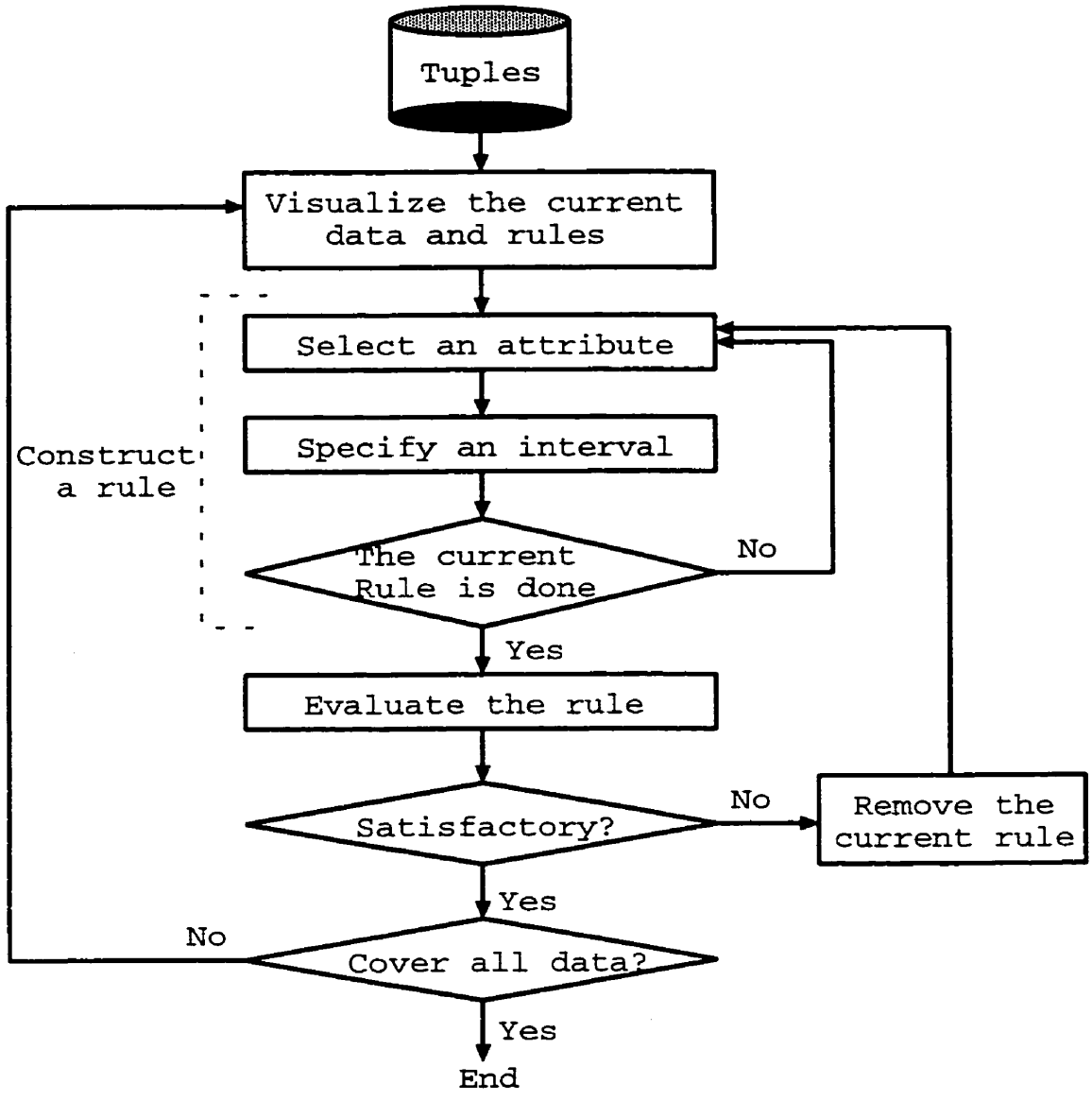


Figure 7.3: The Process of Building Rules in CTViz

1. **Visualize the current data and rules:**

   The classifier construction process operates on the cleaned data set. The current available data and the rules constructed already are visualized on the Table Lens. By looking into the distribution of the available data set, one can perceive the correlations between attributes.

2. **Construct a rule:**

   If there are still tuples that are not covered by the current classifier, one can construct a classification rule by iteratively performing the following two tasks:

   - *Select an attribute:*

     Double click any one of columns. The attribute corresponding to the clicked column is selected as the current attribute. Tuples within each class are sorted by the selected column in ascending order.

   - *Specify an interval:*

     Use a "rubber band" to draw an interesting interval for the selected attribute. The current attribute with interval encompassed by the rubber band is displayed at the left bottom corner.

   Repeat above steps until all conditions for the current rule are specified. If the chosen attribute or the current interval is not appropriate, it can be canceled by choosing the corresponding menu item. Figure 7.4 shows the situation in which two rules have been constructed, one with the condition of small *petal_length* and the other with the condition of large *petal_width*, respectively, and the third rule is under construction with two attributes and intervals being specified.

3. **Evaluate the rule:**

   The current rule constructed in the previous step must be evaluated before it is put in the classifier. In the CVizT system, evaluating a rule consisting of computing its accuracy, coverage, and quality. Two methods of rule evaluation are provided, *resubstitution estimate*, where the test cases and the training cases are the same, and *test sample estimate*, where the raw data set is divided into two parts, one for training set and the other for test set (see Chapter 2).

   If the rule accuracy is equal to or greater than a pre-specified threshold, then it is accepted, otherwise it is discarded automatically.

4. **Update the data set:**

   If the user is satisfied with the current rule (according to its evaluation), then the rule is appended to the classifier, and the available data set is updated. The tuples covered by the current rule are marked and will not be displayed. The display space saved by the marked tuples is used to visualize the rule. The rule condition is displayed as a set of attribute-interval pairs (see Figure 7.4), where the rule accuracy threshold is set as 80%.

   If the current rule is not satisfactory, one can delete it from the classifier. The marked tuples are restored to be available.

Repeat steps 2 through 4 until all produced rules are appropriate and all tuples are covered by these rules. The final classifier is composed of these classification rules.

From above discussion, one can see that the principle of learning classification rules behind the CVizT system is based on sequential covering learning strategy [115, 116, 160].

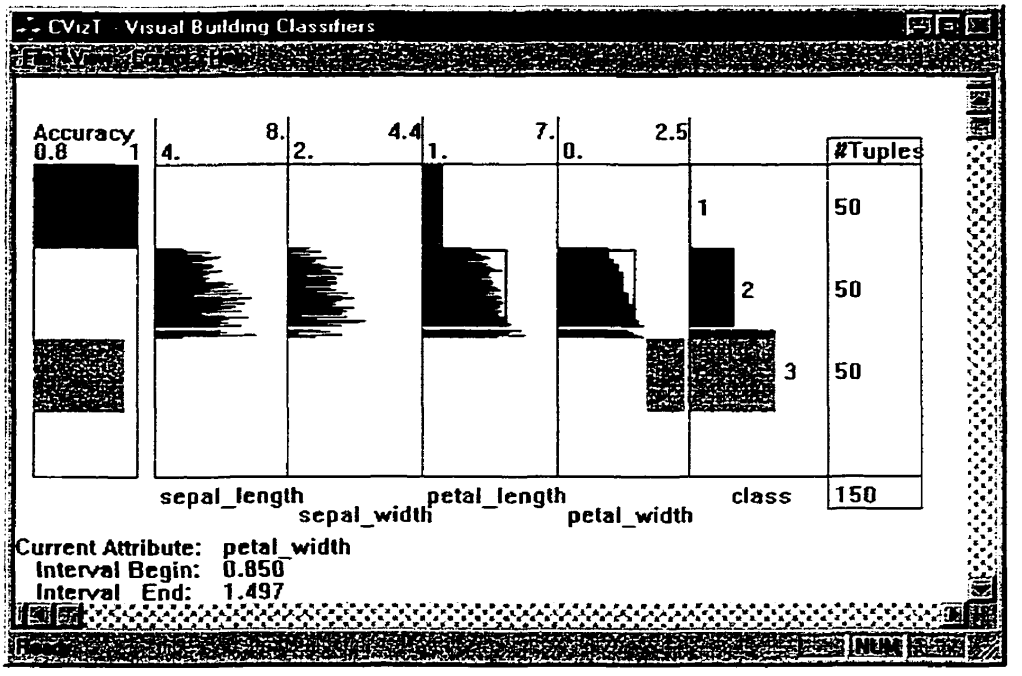


Figure 7.4: Table Lens Visualizing the Classification Rules

7.6 Classification Rule Visualization

The obtained rules are also visualized in the Table Lens, with conditions corresponding to attribute intervals and decision to the class label. Figure 7.4 shows two classification rules currently acquired, and a new rule is being constructed. One may argue that another condition with around minimum *petal_width* should be **OR**ed. This condition, however, is not included because it is unnecessary. From Figure 7.2, one can observe: *petal_width is around minimum if and only if petal_length is around minimum.*

In the left side of Figure 7.4, the rule accuracy is displayed as the area whose height is proportional to the rule coverage and whose width is proportional to the rule accuracy [123]. In Figure 7.4, the first rule has accuracy of 100%, while the

second about 98%. Rules with accuracy less than 80% are ignored. In addition, the color of the rules can be used to represent the rule quality [80].

7.7 CVizT Implementation and Experiment

We implement the approach described in this chapter with Visual C++ 6.0 on Windows 98. We experiment the CVizT system with some of the UCI repository data sets [126], including Glass, Diabetes, Iris, Monks, Parity5+5, etc. In this section, we illustrate our experiments with the Glass data set.

The Glass Identification Database was rawly used for rule induction in Forensic Science. The study of classification of types of glass was motivated by criminological investigation. At the scene of a crime, the glass left can be used as evidence, if it is correctly identified. This data set contains 214 instances, and 10 attributes (including an Id number) plus the class attribute. Except the ID and the class label, all attributes are continuously valued. The attribute information is described in Table 7.1.

The class attribute (Type) has 7 possible values which are labeled as 1 through 7. Table 7.2 illustrates the glass types with the class distribution.

Initially, 11 attributes including ID number and class label are visualized, shown in Figure 7.5, from which one can see some attributes are strongly correlated with the class label, while the others are not. Actually, the ID number is absolutely not correlated with the class label. The attributes uncorrelated to the class label can be removed to save display space for the correlated attributes.

From Table 7.1, one can see that attributes Na, Mg, Al, and Ba have high correlation strength with the class label, and the other attributes have low correla-

Table 7.1: The Attributes Information of the Glass Identification Database

Attribute	Mininum	Maximum	Mean	Standard Deviation	Correlation with class
RI (refractive index)	1.5112	1.5339	1.5184	0.0030	-0.1642
Na (Sodium)	10.73	17.38	13.4079	0.8166	0.5030
Mg (Magnesium)	0	4.49	2.6845	1.4424	-0.7447
Al (Aluminum)	0.29	3.5	1.4449	0.4993	0.5988
Si (Silicon)	69.81	75.41	72.6509	0.7745	0.1515
K (Potassium)	0	6.21	0.4971	0.6522	-0.0100
Ca (Calcium)	5.43	16.19	8.9570	1.4232	0.0007
Ba (Barium)	0	3.15	0.1750	0.4972	0.5751
Fe (Iron)	0	0.51	0.0570	0.0974	-0.1879

tion strength (positive strength means positive correlation while negative strength means negative correlation). Thus, select attributes Na, Mg, Al, and Ba, as well as the class label. The result is illustrated in Figure 7.6.

Figure 7.6 also shows that the instances are sorted by attribute Ba in ascending order. Clearly, most of instances belonging to Type 7 (*headlamps*) have middle values of Ba (*Barium*). To construct a classification rule to reflect such a fact, the attribute Ba is selected and the middle interval is specified. The rule consisting of condition part as *Ba in the middle* and decision part *Type 7* is formed and illustrated in Figure 7.7. From Figure 7.7, one can see that the accuracy of this rule is about 96%.

Furthermore, Figure 7.7 also shows that only part of Type 5 instances have

Table 7.2: Classes with Distribution in the Glass Identification Database

Label	Glass Type	Number of Instances
1	building_windows_float_processed	70
2	building_windows_non_float_processed	76
3	vehicle_windows_float_processed	17
4	vehicle_windows_non_float_processed	0
5	containers	13
6	tableware	9
7	headlamps	29
	Total	214

attribute values of Al (*Aluminum*) close to the maximum value 3.5. Now, select attribute Al and draw the interval close to the maximum end. A new rule is formed and illustrated in Figure 7.8. From Figure 7.8, one can see that the rule has the highest accuracy, 100%.

The above process can be repeated until a group of rules are achieved with high evaluation. No specific stop criterion is defined.

7.8 Discussion

Compared to other approaches for building classifiers, The CVizT system has the following characteristics:

(1) Easy to use: This system is very easily learned. We trained two non computer major students for five minutes with the IRIS flower data set. They built

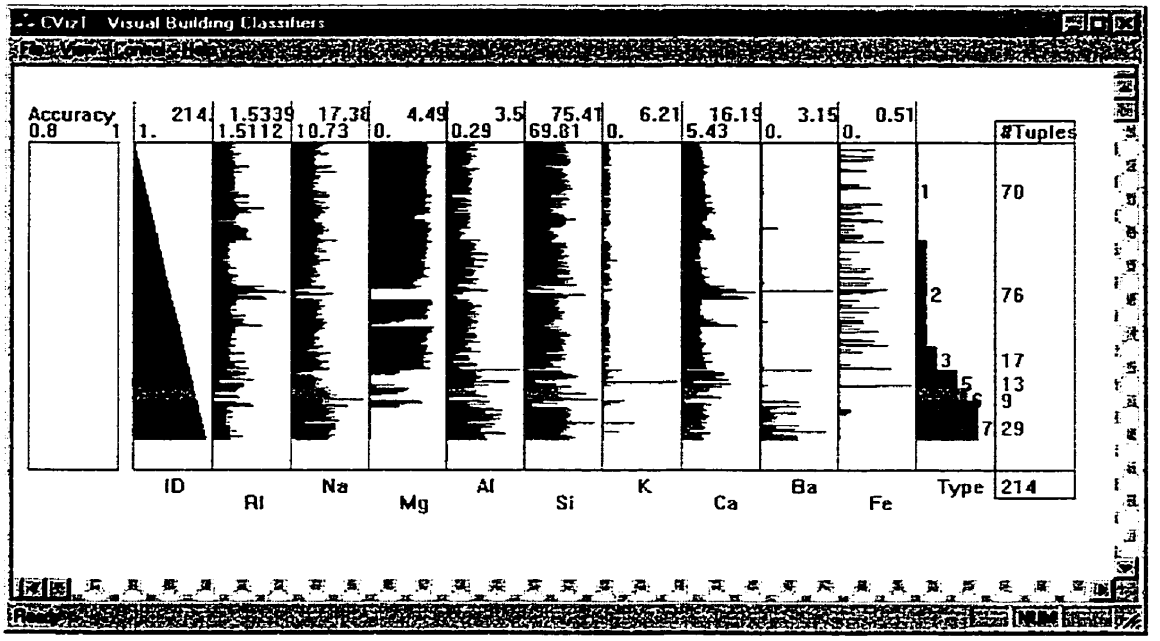


Figure 7.5: The Glass Data Set with 11 Attributes

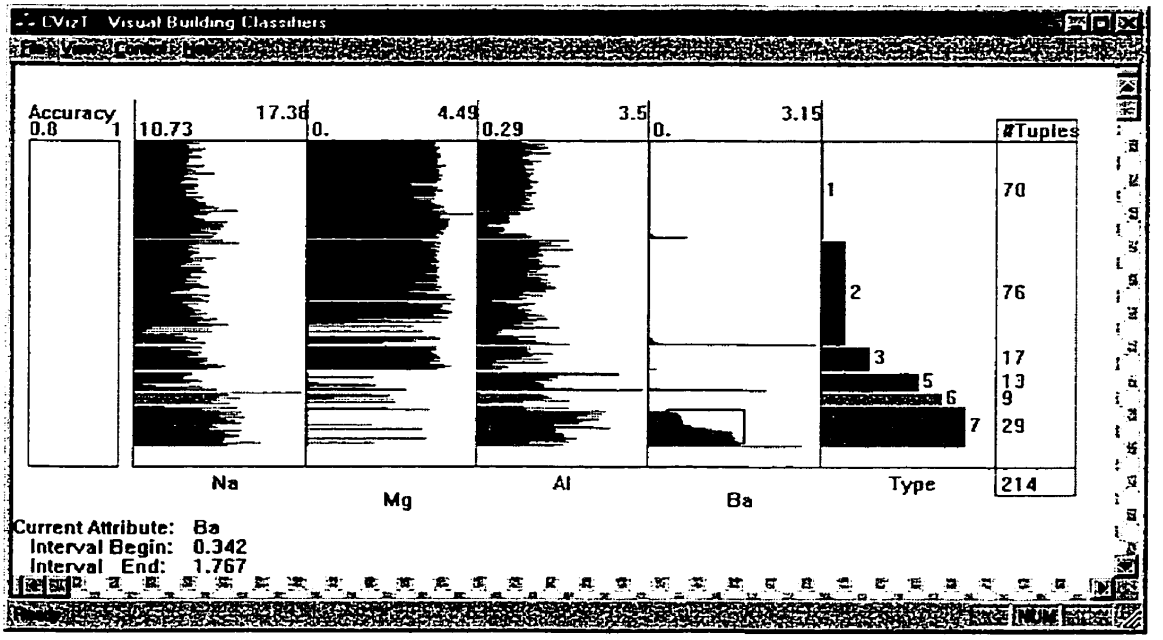


Figure 7.6: The Reduced Glass Data Set with 5 Attributes

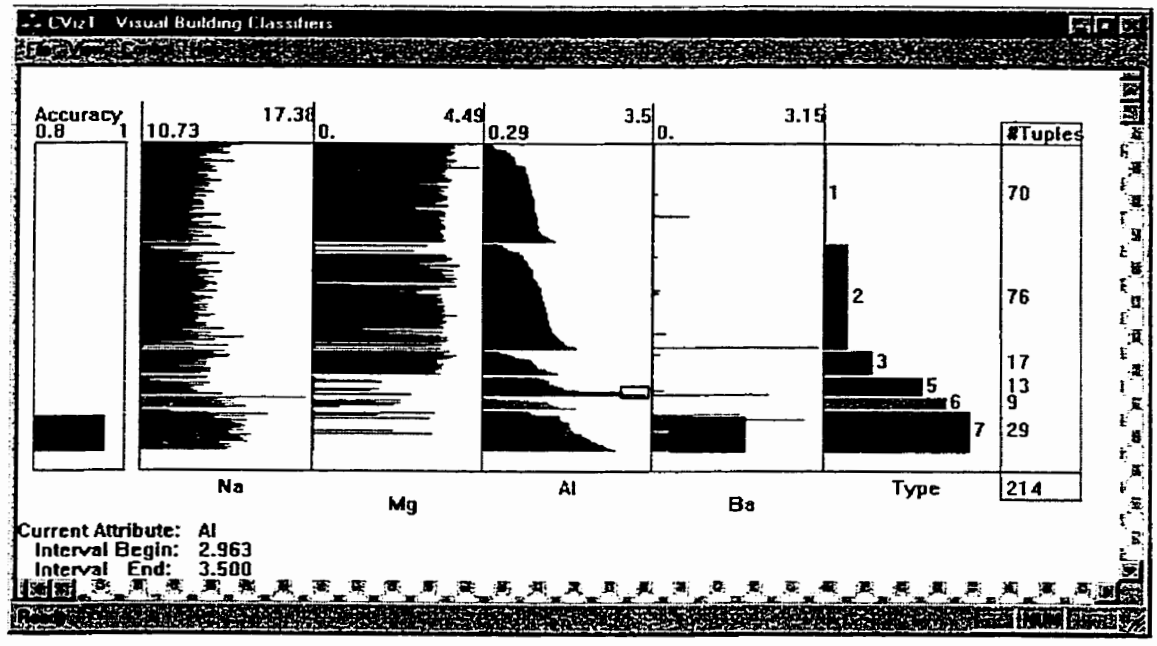


Figure 7.7: The Reduced Glass Data Set with One Rule

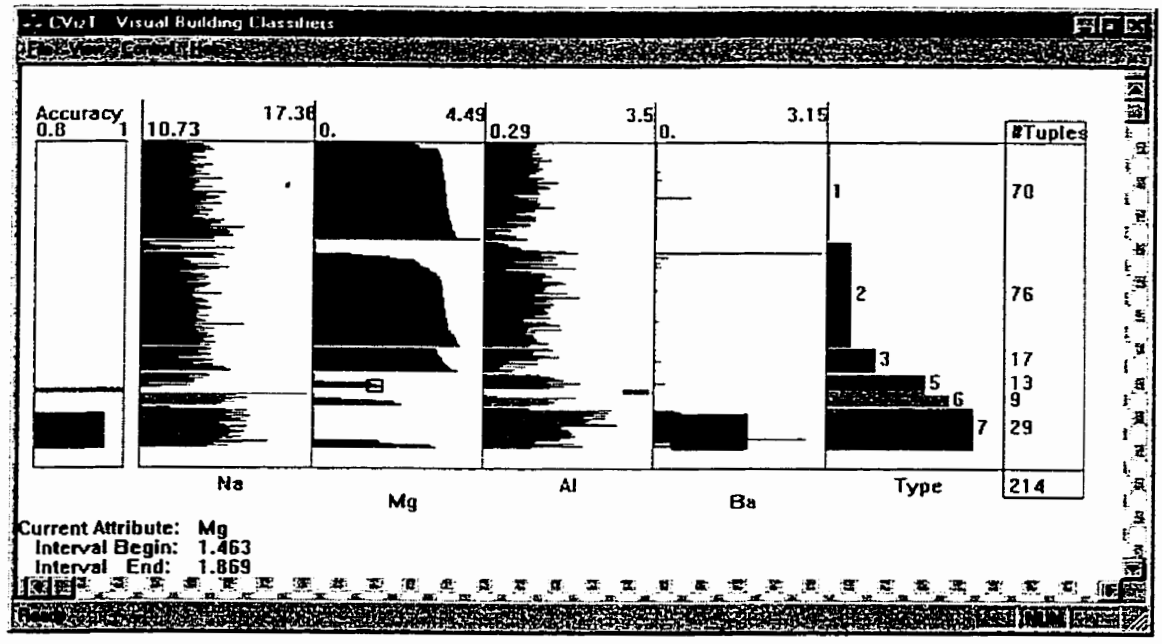


Figure 7.8: The Reduced Glass Data Set with Two Rules

classifiers with an average accuracy higher than 90%.

(2) Uncertain classifiers: For the same data set, the generated classifier is uncertain and user-dependent.

(3) Varying accuracy: The accuracy of generated classifiers also depends on the user. Moreover, the same user may obtain different accuracies of classifiers for different executions of the system.

(4) Understandable classifiers: Generally, most classification rules in the final classifier contain only one or two condition attributes.

(5) On-demand discretization: The continuous attributes are not partitioned in advance. Only the needed intervals are specified on the fly. The ranges not included in any rules are not discretized.

(6) Uniformed process of categorical and continuous attributes: Categorical attributes are processed in the same way as continuous attributes.

However, there is a main disadvantage with the CVizT system. The attribute range is proportional to the width of the display column. If there are many different points in the attribute range, the tuples with the closed attribute values could be mapped to the same pixels and drawn with the same length. In this case, it is difficult to look into the data distribution and discretize attributes, thus the classification rules constructed might have low classification accuracy.

Chapter 8

DTViz: An Interactive System for Constructing Decision Trees

8.1 Introduction

Another kind of classifier is the decision tree. As discussed in Chapter 2, constructing decision trees from a large set of multivariate data has been extensively developed by researchers from different communities such as machine learning, data mining, and statistics [5, 10, 26, 66, 123, 136, 162]. Usually, the decision tree construction consists of two main steps, *growing* and *pruning*. The first step builds a tree root and expands the leaf nodes iteratively until the resulting decision tree satisfies the basic requirements of accuracy with respect to the training set. The second step exploits the testing data to reduce the decision tree sizes (usually the number of nodes) to prevent *overfiting* and improve the understandability of the decision tree.

Different methods have been proposed for the construction of decision tree clas-

sifiers, which are primarily aimed at binary and categorical attributes. The numerical attributes must be discretized into distinct intervals so that they can be used in the same way as categorical attributes. Moreover, the well-known algorithms for building decision tree classifiers perform a binary split of the form $a \leq v$ or $a \geq v$ for a numerical attribute a and a cut point v. For example, the C4.5 decision classifier [136] discretizes numerical attributes based on an entropy-based partition. For each value v occurring in the training set of attribute a, it calculates the *information gain* with respect to $a \leq v$ and $a \geq v$. The value corresponding to the largest information gain or gain ratio is chosen as a split point to discretize the attribute a. Thus a is partitioned into two parts. Repeat the partition, the numerical attribute a is finally partitioned into a set of binary attributes. The SPRINT decision tree classifier discretizes numerical attributes as follows [5]: there are $n-1$ possible splits for n distinct values of a. The GINI index is calculated at each of these $n-1$ points and the attribute value yielding the minimum GINI index is chosen as the split point. In addition, the CLOUDS decision tree classifier [10] draws a sample from the set of all attribute values and evaluates the GINI index only for this sample thus improving the efficiency.

There are several interactive systems for constructing decision trees. Some of them are to visualize the decision tree structure only [1, 128], while the others are trying to interactively construct the decision tree classifiers [13, 150]. SPSS CHAID [150] provides an approach for interactive construction of decision tree classifiers, but it does not visualize the training data and does not interactively grow the decision tree. It only interactively accepts parameters such as accuracy threshold and tree size threshold from the user to control the tree construction and visualizes the final decision tree generated. The PBC system [13] is very close to our work (See Chapter 2). However, it uses *circle segments* technique to visualize the raw data,

and some visualization space (on the window corners) is wasted and the tuples in the training data are limited to a small number. Secondly, the decision tree being constructed is not visually displayed so that the user is not able to see clearly what is going on and to decide what to do next. Finally, the PBC system does not provide any approaches for cleaning the raw data. When the number of attributes in the raw data is huge, say 100, the number of tuples is limited to less than 2000 for fitting a 516×516 display window [13].

Like the CVizT system, rather than constructing decision trees by sophisticated algorithms, we present a novel approach and develop a visualization system, DTViz, according to the RuleViz model, for interactively building decision tree classifiers by visualizing the current available data sets and the current tree. This approach is fully interactive: including feature selection, numerical attribute discretization, node split, labeling nodes, and pruning. The current available data set is interactively determined by choosing the current tree node. The training data is visualized in a pixel-oriented technique, *parallel segments*, similar to *circle segments* [15]. Constructing decision tree classifiers is an interactive process consisting of selecting a node and an attribute, discretizing the attribute and splitting the node, pruning a node with its all children, looking at a node's state, labeling or unlabeling a node, etc. During the decision tree construction, the user can integrate the domain knowledge of experts, see the intermediate decision tree, check the state of nodes, and feedback his/her perception.

We experimented with the DTViz system using data sets from the UCI repository. Our performance evaluation demonstrates that the interactive construction of a decision tree is straightforward and easily learned, and the decision tree generated in this way is satisfactory with the basic requirements of accuracy and understandability. The user's preference can also be integrated into the construction process.

8.2 The DTViz System

DTViz is a visual interactive system for constructing decision tree classifiers. Like the other systems developed according to the RuleViz model, the DTViz system consists of the following components: data tuple visualization with a pixel-oriented technique, feature selection and tuple selection, decision tree node construction with interactive operations, decision tree node evaluation, and decision tree visualization, shown in Figure 8.1.

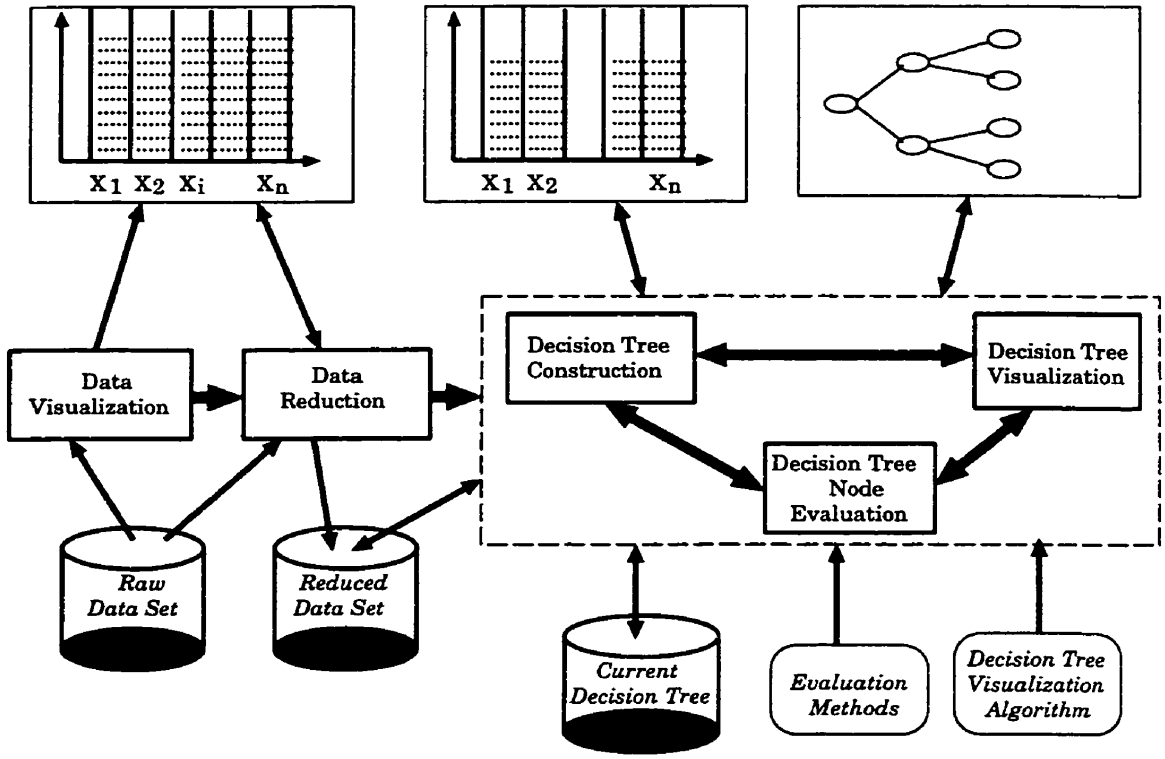


Figure 8.1: The DTViz System

The training data are visualized in order for the user to see the data distribution vertically and horizontally. We develop a pixel-oriented data visualization technique, *parallel segments*, obtained by combining with the *circle segments* [15]

and *parallel coordinates* [91] techniques, where each parallel coordinate (column) corresponds to an attribute. The pixel-oriented visualization techniques map each attribute value of each tuple to a colored pixel and represent the values belonging to different attributes in separate areas or subwindows [95, 98]. The tuples are sorted in ascending order by each attribute and the attribute values are displayed in the separate columns in pixels row by row with the pixel being rendered as the class label color. Therefore, the user can easily perceive the degree of impurity with respect to class memberships (see Chapter 3).

The display window is often limited and can not accommodate a large amount of tuples. If scrollable windows are used, the user is not capable of observing the entire data set at first glance and thus the user's perception is affected. To get around this problem, we use an unscrollable window to visualize the data tuples and allow the user to select attributes strongly related with class labels. The selected attributes and randomly sampled tuples (if necessary) are used as the training examples for constructing decision tree classifiers.

The construction of decision trees is an interactive loop process with feedback from the user. This process consists of the following five operations on the decision tree currently built: selecting a node and splitting it, labeling a node as the final leaf node, unlabeling a labeled node to allow it to be resplit if necessary, checking the state of any nodes, and pruning a non-leaf node with all its children. Splitting a node means selecting an attribute and partitioning the data set contained in the node according to the tuple values. The attribute selected to be split may be categorical or numerical. But the DTViz system deals with them uniformly: the categorical attribute values are numbered and processed as numerical ones. Thus, splitting an attribute is to discretize it. The method for discretizing attributes in the DTViz system follows the principle discussed in Chapter 3. The user can use

these operations to control the growing and shrinking of the decision tree being constructed.

When a node is checked or labeled, it is evaluated and the evaluation result is displayed around it. The node evaluation is to calculate the node's support (coverage) and classification accuracy.

The decision tree that is currently constructed is visualized in a separate window from the data tuple visualization window, and is drawn from left to right with the root at the middle of the left side. The nodes in the same level of the tree are uniformly arranged. The non-leaf node is represented as a rectangle, while the leaf node is represented as a rectangle with round corners and rendered according to its support and accuracy.

Actually, decision trees are interactively constructed by iteratively performing the last three components and switching between the data tuple visualization window and the decision tree visualization window. Usually, the data tuple visualization window reflects the data tuples contained in the current decision tree node (being labeled, unlabeled, split, or pruned). Initially, it visualizes the entire data set corresponding to the decision tree root.

8.3 Parallel Segments: A Pixel-Oriented Data Visualization Technique

Circle segments [15] is a pixel-oriented data visualization technique, and *parallel coordinates* [91] is a geometric projection data visualization technique. Both of them are for visualizing high-dimensional data from different angles. We combine these two techniques together to develop a new visualization technique, *Parallel*

Segments. Figure 8.2 illustrates this technique with an eight dimensional data set.

Following the idea of the *parallel coordinates* technique, the visualization area is divided into d equal sized segments for d-dimensional data set with each segment corresponding to an attribute. Within each segment, the pixels are arranged to start from the left bottom and end at the right top in a row-by-row and bottom-up fashion. The pixel arrangement is similar to that in *circle segments* where a circle is partitioned into d segments and the arrangement starts in the middle of the circle and continues to the outer border of the corresponding segment in a line-by-line fashion, and lines upon which the pixels are arranged are orthogonal to the segment halving lines. For more information about *circle segments*, see [15]. The *Parallel segments* technique has two advantages over *circle segments*. Firstly, the parallel segments technique utilizes the entire drawing area, while the circle segments technique wastes the four corners [1], especially, for low-dimensional data set the waste is considerable. Secondly, *parallel segments* arranges pixels in horizontal lines (parallel with window edges), while *circle segments* does it in arbitrary lines, depending on the number of attributes. Usually, the former arrangement is easily perceived.

To visualize the training examples (data tuples), we map the attribute values occurring in the data set to the pixels in the attribute columns. The attribute values are sorted by each attribute and mapped to the pixels in the arrangement order illustrated in Figure 8.2. The pixels are rendered as the color determined by the class label of tuples to which attribute values belong. The color scale for class labels used in the DTViz system is derived from the PBC system [13] which is based on the HSI color model, a variation of the HSV model. In this model, each color is represented by a triple (hue, saturation, intensity), and the most distinctly

[1]Usually, the visualization window (or area) is a rectangle, and an inside circle can not occupy the entire rectangle.

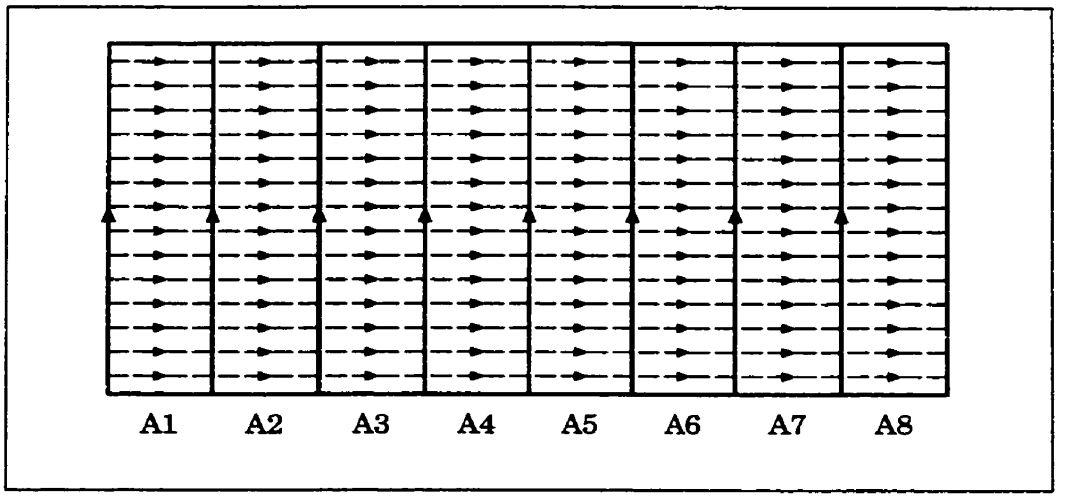


Figure 8.2: Illustration of the *Parallel Segments* Visualization Technique

perceived colors can be achieved by setting the following parameters: Assuming we have k class labels that are to be rendered with k different colors, denoted $color_i, i = 1, 2, \ldots, k$, the colors are obtained by setting saturation and intensity to 1.0, and partitioning the hue scale into $k - 1$ equadistant intervals and setting the hue to the k boundary points, respectively. Thus, the i-th color is defined to

$$(hue, saturation, intensity) = (0.5 + 2 \times \frac{i-1}{k-1}, 1.0, 1.0), \tag{8.1}$$

$i = 1, 2, \ldots, k$.

The number of data tuples that can be visualized at one time is determined by the area of each attribute, which is roughly $\frac{1}{k}$ the area of the window. Figure 8.3 shows the data tuple visualization window of the DTViz system. The visualized data set is *Adult* from the UCI repository [126], which contains 30,162 data tuples and 14 attributes plus a class attribute.

In order to visualize categorical and continuous attribute values uniformly and for the convenience of later processing, the categorical attributes are digitized into

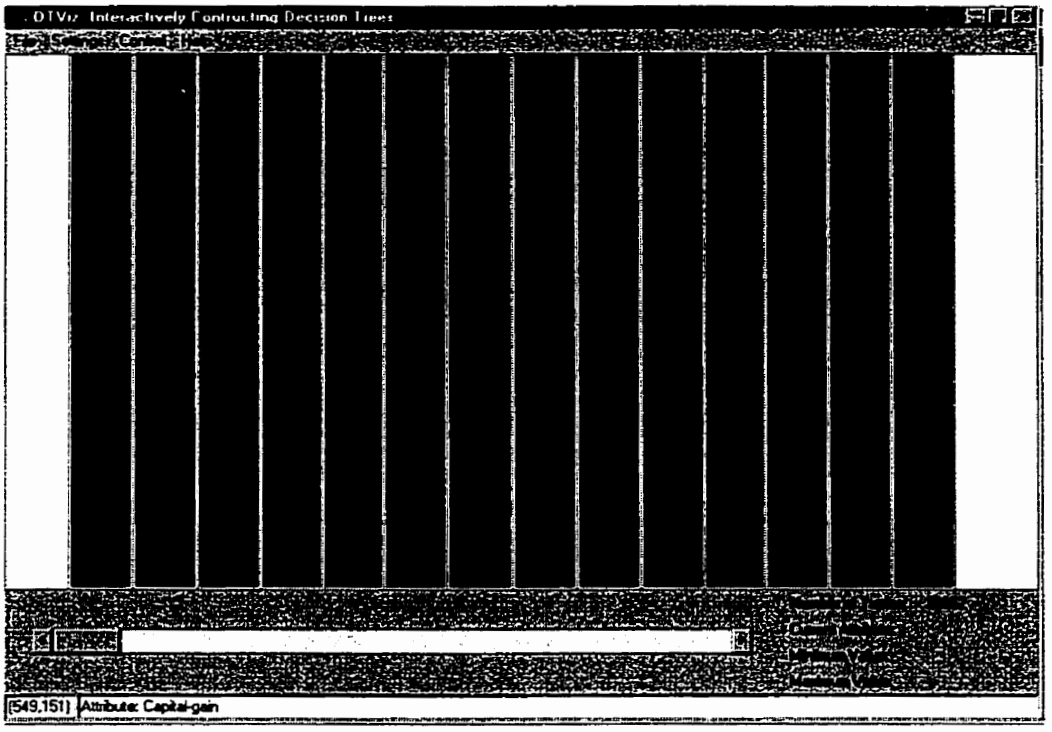


Figure 8.3: The Raw *Adult* Data Set with 30,162 Tuples

numbers from 0 to $N-1$, where N is the number of categorical values of an attribute.

8.4 Data Reduction

The interactive data reduction implemented in the DTViz system consists of horizontal selection of tuples and vertical selection of attributes.

As discussed previously, the size of data that can be visualized in *parallel segments* is determined by the data visualization window size and the number of attributes in the data set. In real world applications, the number of attributes may be hundreds or thousands, and the number of tuples may be millions or billions.

In such cases, it is necessary to select important attributes with respect to the class labels and sample the typical tuples. Like the Table Lens used in the CVizT system, if the window for *parallel segments* is designed to be scrollable, the user is not able to see the global distribution of attribute values and the class labels at first glance. Therefore, the number of tuples displayed in the parallel segments is limited. On the other hand, to visualize the data values in parallel segments, the tuples are sorted by each attribute. The well-known sorting algorithms are $O(n \log n)$ in running complexity, where n is the number of tuples. Actually, the *quicksort* algorithm is used in the DTViz system, which is $O(n \log n)$. Assuming k attributes, the sorting algorithm will run in $O(kn \log n)$. We will see that the sorting must be done as long as the data set is updated as the decision tree construction proceeds. For large k and n, the system efficiency is very low. We tried to construct a decision tree with the data set *Adult* shown in Figure 8.3. The response time of the machine is in average over 5 minutes for each construction operation at the beginning of the decision tree construction. To get around this problem, the DTViz system randomly sample the raw data set if the data set is large. Figure 8.4 illustrates such a case where $5,000$ tuples are randomly sampled from the raw *Adult* data set.

The DTViz system also provides an *interactive feature selection* mechanism, in which the user can arbitrarily delete any attributes that s/he thinks irrelevant or not strongly related to the class attribute. Similarly, how many attributes and which attributes should be deleted or retained are the user's responsibilities.

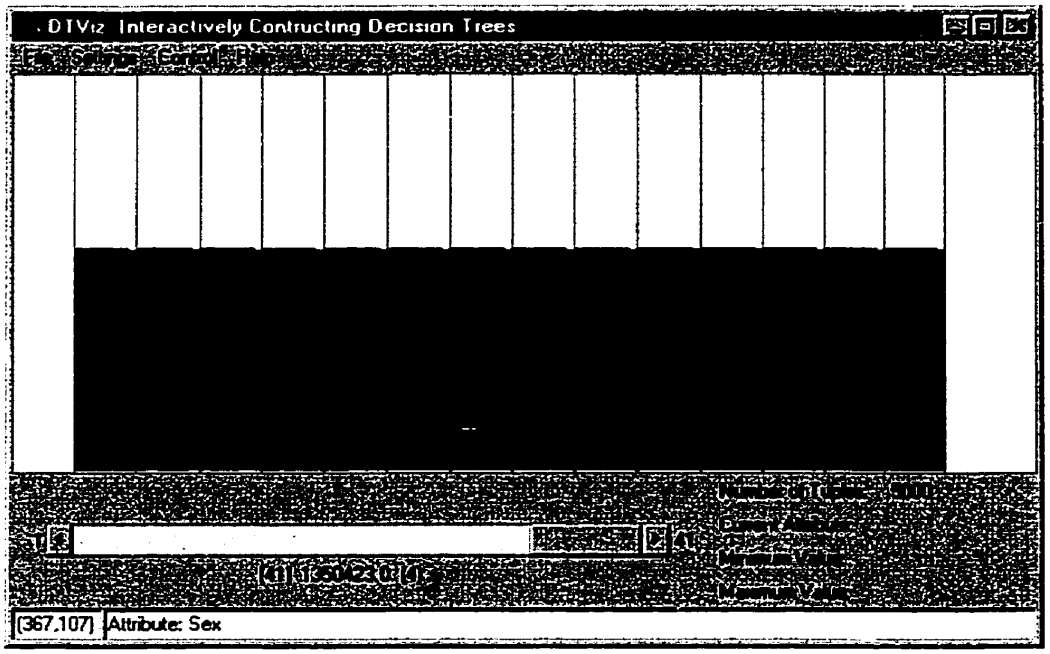


Figure 8.4: Randomly Sample of 5,000 Tuples from the Data Set *Adult*

8.5 Decision Tree Visualization

The intermediate decision trees during the construction and the final decision tree are visualized in another window, called the *decision tree interaction window.* Here, a decision tree is drawn as the tree structure from left to right and from middle to top and bottom. Figure 8.5 shows the final decision tree that is obtained from the IRIS flower data set.

The root of the decision tree is in the middle of the first column. The children of the root are evenly distributed in the second column. Generally, all the tree nodes at the i-th level are evenly distributed in the i-th column ($i \geq 1$).

There are two kinds of tree nodes that must be distinguished, *labeled* nodes and *unlabeled* nodes. Labeled nodes represent the leaf nodes of the final decision tree that can not be split again and are labeled with class labels such that the most

instances satisfying the path from the root to this node have the same class labels. The labeled nodes are drawn as rounded rectangles, and rendered in terms of the node evaluation and the HSI color model. Assuming that the prediction accuracy and support (coverage) of a labeled node are a and s, respectively, like the pixel rendering in the parallel segments technique, the color of this node is calculated as follows:

$$\begin{aligned} hue &= 0.5 + 2 \times a, \\ intensity &= 0.5 + 2 \times \tfrac{s}{n}, \\ saturation &= 1.0, \end{aligned} \tag{8.2}$$

where n is the size of the data set (training set or testing set depending on the evaluation method selected by the user, see Section 8.6.3).

Moreover, the class label and the pair $(support, accuracy)$ are displayed in the labeled nodes. For example, in the second level of the decision tree shown in Figure 8.5, there are two labeled nodes and one unlabeled node. The top labeled node shows that the class label is 1, the node support is 50, and the node classification accuracy is 100%. The bottom labeled node shows that the class label is 3, and the node support and classification accuracy are 51 and 92%, respectively.

The unlabeled nodes are drawn as rectangles, which are filled in the split attributes. If an unlabeled node is under construction, that is, it has not been labeled as a final leaf node yet, and currently does not have children, then it is left unfilled and ready to be split. In Figure 8.5, three unlabeled nodes are contained in the decision tree. For example, the root is filled in *petal_length*, meaning that the root is split with the attribute *petal_length*.

The nodes associated with the *parent-children* relations are connected by lines. The lines are marked with the split attribute intervals. In Figure 8.5, for example, the root is connected to its three children, and the connections are marked by

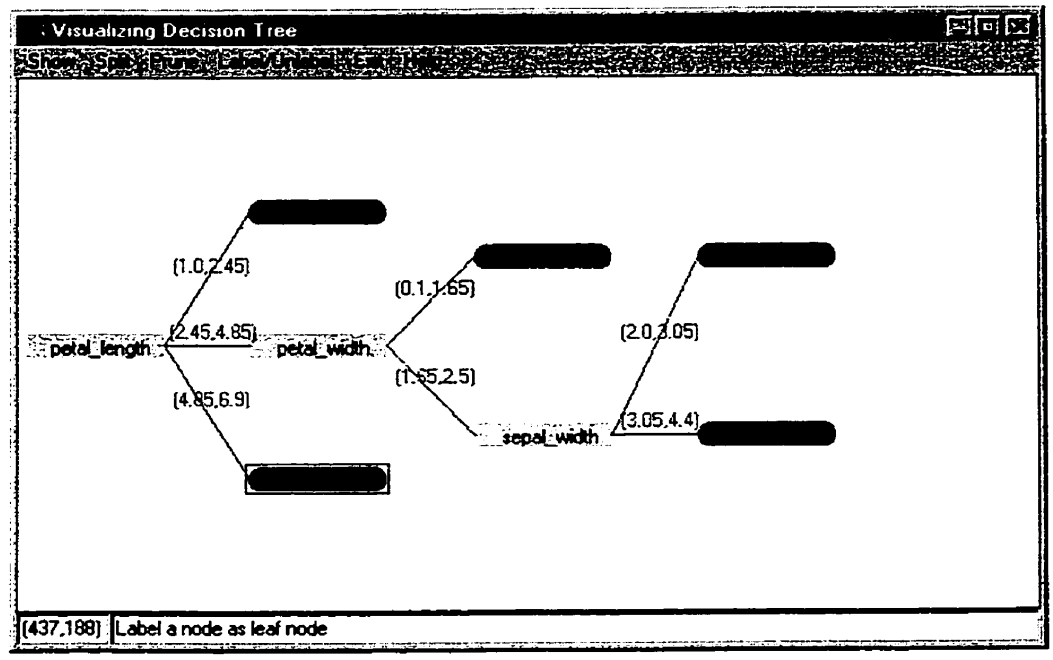


Figure 8.5: Visualizing a Decision Tree

$[1.0, 2.45]$, $(2.45, 4.85]$ and $(4.85, 6.9]$, respectively, which signifies that the attribute *petal_length* filled in the root is partitioned into three intervals as above.

In the decision tree interaction window, interactive operations are provided, including *showing* the node states, *splitting* non-leaf nodes without children currently, *labeling* and *unlabelling* leaf nodes, *evaluating* nodes, etc. These interactive operations are used to interactively construct the decision trees, and will be described in the next section.

8.6 Interactive Construction of Decision trees

Constructing a decision tree classifier is to build a decision tree from the training set that satisfies the basic requirements of classification accuracy, support and un-

derstandability. The construction of decision trees consists of operations such as selecting attributes, splitting tree nodes, evaluating classifiers, and pruning decision trees to prevent "overfitting", etc. The DTViz system provides various interaction tools for the user to build decision trees based on his/her perception of the training set and the decision tree evaluation. Figure 8.6 depicts the interaction model developed in the DTViz system.

The interaction model for constructing decision trees consists of two display windows and five operations. One window is to visualize the training data that is covered by the current node and to perform the selection of the split attribute and the partition (discretization) of the selected attribute. This window is called the *data interaction window*, illustrated in Figure 8.4. The other window is to visualize the decision tree being constructed and perform the other interactive operations, including labeling and unlabeling nodes, checking/evaluating/showing nodes, and pruning nodes. This window is called the *decision tree interaction window* and is illustrated in Figure 8.5.

Once the raw data set is cleaned and the training set is determined, the system enters the interactive construction of decision trees. At the beginning, the decision tree to be constructed contains only one node, *decision tree root*, that covers the entire training set. The decision tree interaction window displays the single root node with nothing filled in the node rectangle. At this moment, the root is the current node for splitting. The data interaction window displays the data covered by the current tree node, which is the entire training set in this case.

As the decision tree construction proceeds, the user can arbitrarily select one of five interaction operations to view the state of each tree node and to control the growing and shrinking of the decision tree. These interactions are listed as follows and will be described in the following subsections:

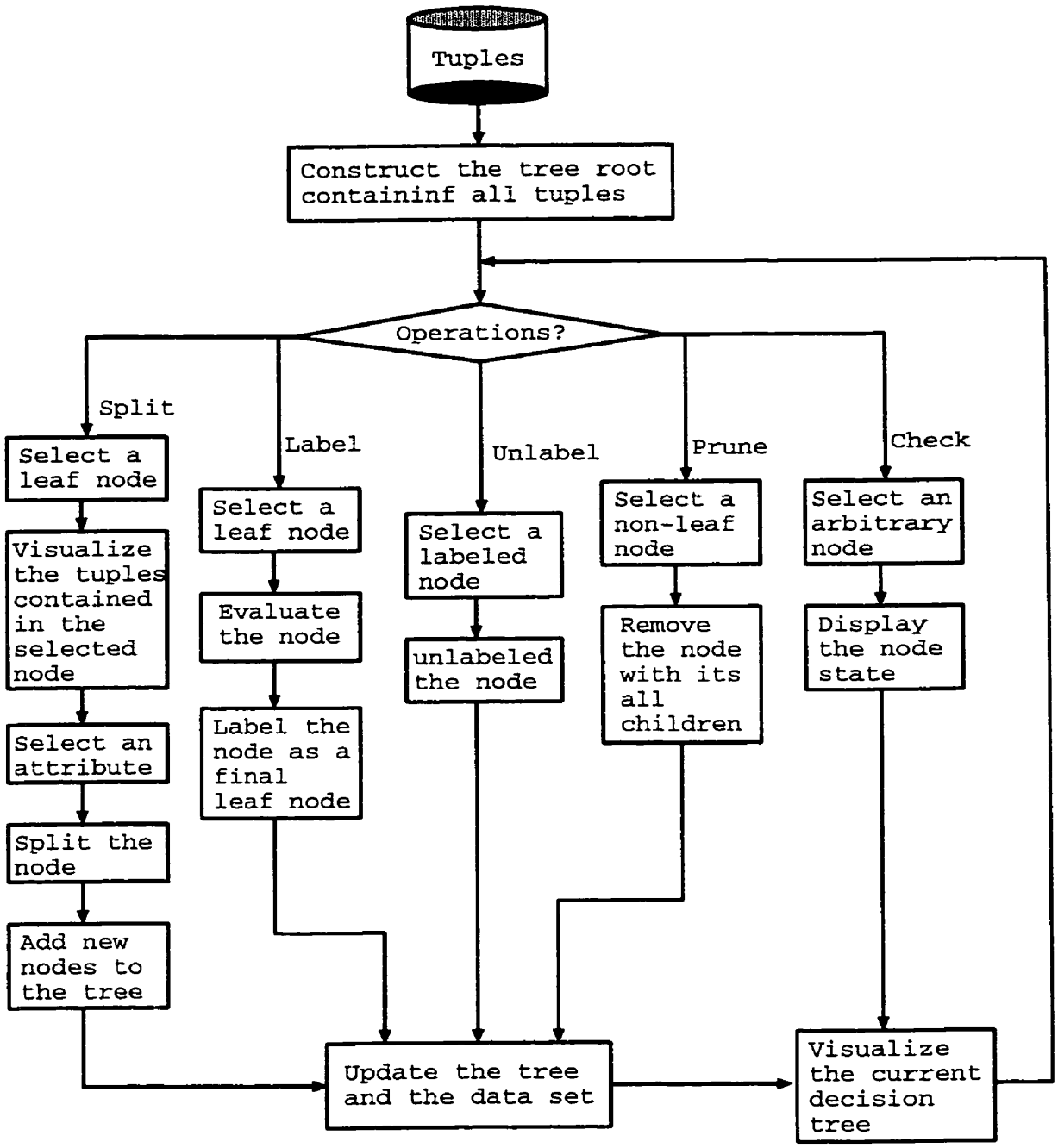


Figure 8.6: The Interaction Model of Constructing Decision Trees Used in DTViz

- *split* a node to grow the decision tree;
- *label* a leaf node so that it can not be split again;
- *unlabel* a labeled node so that it can be split further if necessary;
- *prune* a non-leaf node to shrink the decision tree or to re-split the node if its previous split (split attribute selection and/or the attribute discretization) is not satisfactory; and
- *check/show* the state of a node (coverage and classification accuracy if the node is labeled) by evaluating the node with the specified evaluation method.

The first four operations will change the current decision tree (new nodes being added, old nodes being removed, or state of nodes being changed), and hence the training set may need to be updated. For example, when a node is split, its children are generated, thus the data sets that these children cover must be calculated accordingly and attached to these nodes, and also the node being split should be changed to a non-leaf node. Finally, the decision tree updated already is re-displayed in the decision tree interaction window.

The interactive construction of a decision tree is finished when all leaf nodes are labeled.

8.6.1 Node Split

The *node split* includes three steps, selecting a node for splitting, selecting a splitting attribute, and selecting split points of the attribute. The first step is performed in the decision tree interaction window by specifying the *Split* menu item and clicking the node to be split. The labeled nodes can not be selected for further splitting.

The second and third steps are performed in the data interaction window. When the node to be split is specified as the current node in the decision interaction window, the data interaction window only visualizes the data tuples that are covered by this node, and the segments corresponding to the attributes that occur in the nodes on the path from the root to the current node are left blank because these attributes will not be utilized along any paths starting from the current nodes to its descendants.

The splitting attribute can be specified by selecting the *Pick an Attribute* item under *Control* menu to enter split status, and then clicking the attribute to be split. When an attribute is specified, its name, maximum and minimum values are displayed at the right-bottom corner of the data interaction window. This corner also displays the number of the current data set (the set of tuples covered by the current decision tree node). In the left-bottom side, a scroll bar is set to contain all values of the splitting attribute that occur in the current data set. These attribute values are also sorted in ascending order to reflect the values visualized in the parallel segments. This scroll bar is an assistant tool to help the user select split points of the current attribute. Below the scroll bar, the tuple number in the current data set, the splitting attribute value, and the class label of the current tuple are displayed. Moving the slider in the scroll bar will change these values.

Attribute selection should follow two strategies:

- *The clearer the clusters in a parallel segment, the better the corresponding attribute for splitting.* For example, in Figure 8.4, the segment on the left side has two very clear clusters at the bottom, while on the top, class labels are distributed randomly. On the other hand, the fourth segment from the right side also has two clear clusters, one at the bottom and the other on

the top. The middle part of this segment also contains two clusters which are a little bit vague and are separated by a part in which class labels are evenly distributed. Similarly, the sixth segment from the right side contains two clear clusters which are separated by only a non-cluster area. According to this attribute selection strategy, each of them is appropriate for splitting.

- *The more approximate the area size of clusters as well as non-clusters, the better the corresponding attribute.* Generally, clusters and non-cluster areas correspond to children nodes of the current node if the current attribute is selected as splitting attribute, although clusters easily lead to leaf nodes of the decision tree, and non-cluster areas must be split further. If the cluster and non-cluster areas are approximately equal in size, the decision tree constructed with such splitting will be balanced, and has less levels and probably less tree size (fewer tree nodes). Refer Figure 8.4. According to this strategy, the sixth segment from the left size is preferred to the other two segments mentioned in above strategy.

Once the splitting attribute is selected, the system enters the state of interactively selecting split points. To select a split point, one just needs to click upon the pixel that separates clusters. Note that the separation of two different colors is not the only criteria for determining the exact split point because the same attribute values may belong to tuples of different classes. Hence, it would not be reasonable to set a split point between two differently colored pixels. To solve this problem and help the user identify the reasonable split points, the DTViz system provides feedback to the user in the following ways:

- The attribute value of the pixel at the position of the mouse pointer appears in the status area at the bottom line of the data interaction window when the

mouse is moving. Therefore, the user can check if two adjacent but differently colored pixels have the same attribute values by moving the mouse through the separation of two clusters. If they have the same attribute values, just ignore this separation; otherwise it is appropriate to set a split point here.

- The scroll bar provides another means of viewing attribute values and class labels. First, point the mouse around the separation of two differently colored clusters if it is hard to point at the exact pixel which separates two clusters. The attribute value of the pixel under the mouse pointer, called the *reference value*, is displayed in the status area. The *reference value* gives the user a rough idea of where the possible split point is because the attribute values are sorted. Then the user can move the slider to the point corresponding to the *reference value*, and slightly move the slider left and right to see if the class labels of tuples that have attribute values around the *reference value* change or not (the current tuple number, attribute value, and class label are displayed below the scroll bar). If attribute values keep unchanged while class labels are changing, it shows the same attribute values belong to tuples of different classes, hence the value is not a reasonable split point. If class labels change as attribute values change around the *reference value*, then there must be a good cut point around the *reference value*. In this case, one can carefully look into this separation by moving mouse, scroll slider, and observing attribute values in the status area to find an appropriate pixel as a split point.

After a separating pixel is found, a spilt point is formed as follows: Assume the attribute value corresponding to the separate pixel is u and the next value is v. The split point is set to the middle between u and v. Hence, the separating pixel should be in the lower cluster (corresponding to the maximum value in the lower

cluster).

The cut points are displayed above the scroll bar. The DTViz system allows the user to split an attribute into more intervals at one time, overcoming the limitation of binary splits in continuous attributes. Assume r cut points are $s_1, s_2, \ldots, s_r$, and the minimum and maximum values are s and S, respectively, then the intervals will be $[s, s_1], (s_1, s_2], \ldots, (s_r, S]$. Correspondingly, $r+1$ children nodes will be generated for the current node of the decision tree being constructed.

Following the splitting strategy discussed in [13], one can partition the coherent regions of values in the splitting attribute that s/he intends to cut by split pixels. The splitting strategy includes the following four options listed in the priority order, which results in a decision tree optimized in terms of classification accuracy and the number of tree nodes:

1. *Best Pure Cluster Partitions.* Choose the largest pure cluster in which all tuples have the same class label (only one color).

2. *Largest Cluster Partitions.* Choose the largest cluster in which almost all tuples have the same class label (only one clearly dominant color).

3. *Best Complete Partitions.* Choose the largest cluster that contains the most pixels that can be partitioned into subclusters further each of which has one clearly dominant color.

4. *Different Distributed Partitions.* Choose the cluster where different distributions of class labels can be best identified and thus separated through split points.

Generally, the first two partitions lead to labeled leaf nodes in the decision tree, hence reducing the size of the decision tree. The first partition results in

very high accuracy (in most case, 100%), and the second partition results in high accuracy (usually about 90% in our experiments). For example, in Figure 8.5, the top leaf node in the second level of the decision tree is obtained by *best pure cluster partitions*, which has 100% of accuracy, while the bottom leaf node in the same level is acquired by *largest cluster partitions*, which has 92% of accuracy.

Clusters that are chosen by one of the last two partitions lead to non-leaf nodes in the decision tree, meaning that the desired nodes need to be split further. For example, the middle node in the second level of the decision tree shown in Figure 8.5 is obtained by *best complete partitions*.

8.6.2 Node Labeling/Unlabeling

Two operations for labeling and unlabeling decision tree nodes are provided in the DTViz system. If a node is obtained by using *best pure partitions* or *largest cluster partitions*, this node is probably an appropriate final leaf node of the decision tree. To make such a node a final leaf node, choose the *Label a node* item under the *Label/Unlabel* menu and click the node to be labeled. The labeled node is redrawn as a rounded rectangle and rendered with the color calculated by the method discussed in Section 8.5. The class label that occurs most frequently in the sub data set covered in this node is found to be the node label and displayed in the node with the node classification accuracy and coverage which are computed by *node evaluation*, described in next section.

The node to be labeled must be a leaf node of the current decision tree. To assure that the labeled node has high accuracy so that the decision tree classifier is optimized, one can first evaluate the node to be labeled to see its classification accuracy and coverage before labeling it.

If one wants to change his/her mind after a node is labeled, the labeled node can be unlabeled by selecting *Unlabel a node* item under the *Label/Unlabel* menu and clicking the node to be unlabeled. The unlabeled node is restored to a leaf node that can be split again.

8.6.3 Node Evaluation

At any time during the construction of decision trees, one can specify any nodes in the current decision tree to evaluate by choosing *Evaluate a node* item under the *Show* menu. The node evaluation includes the following three aspects.

- *Find node class label.* The node class label is the label that most frequently occur in the set of tuples that are covered by the node. A tuple is covered by a node if it satisfies the path from the root to the node. For example, in Figure 8.5, the tuples covered by the node *sepal_width* (the bottom node in the third level) have attribute values of *petal_length* in $(2.45, 4.85]$, and of *petal_width* in $(1.65, 2.5]$.

- *Find node support.* The node support is the number of tuples that are covered by the node.

- *Find node classification accuracy.* The node classification accuracy is calculated as the occurring frequency of the node label in the set of tuples covered by the node.

Like the CVizT system, DTViz also provides two methods of node evaluation, *resubstitution evaluate*, which uses the same data set as both training set and testing set, and *test sample evaluate*, which uses different training and test sets. The user

can specify one of the two methods through the *Node Evaluate Method* item under the *Settings* menu in the data interaction window. If the *test sample evaluate* method is selected, one must specify the test data file. The DTViz system does not automatically divide a data set into training and testing sets. Actually, two evaluation methods can be applied alternatively for the same node of the decision tree being constructed to compare the evaluation results.

8.6.4 Decision Tree Pruning

Simply stated, decision tree pruning in the DTViz system is to remove a part from the decision tree that has been constructed. Generally, decision tree pruning is needed in the following cases:

- The user is not satisfied with the structure of a part of the current decision tree. For example, the user finds out, when the decision tree grows large enough, that a node was not split properly (too many or too few split points) and hopes to resplit it. In this case, all descendants of this node can be removed from the decision tree by pruning this node.

- The user thinks it is difficult to split some unlabeled leaf nodes to get high node evaluation. Consider an extreme example. The user may find out that an unlabeled leaf node covers a small set of tuples that have the same values for all remaining attributes [2] but have two different class values with half labeled as one class and half as another class. The node evaluation is around 50%. To promote the decision tree classification accuracy, the node needs to

[2]Remaining attributes of a decision tree node are those attributes that have not been used as a split attribute in all nodes on the path from the root to this node.

be split further. However, the further node split will lead to more tree nodes and hence increase the size of the decision tree, and will also result in nodes that have small support. Therefore, the user may want to revise the node splits made previously. In this case, the user can look into the nodes on the path from the current node to the root and decide where to revise by pruning the corresponding node.

- The final decision tree is very large, though it has very high accuracy. To reduce the decision tree size, some parts of the tree need to be removed, thus the classification accuracy of the decision tree could be decreased. The decision tree pruning in this sense in the DTViz system is the same as the traditional decision tree pruning discussed in Chapter 2.

To prune a subtree from the decision tree constructed so far, one can choose the *Prune a node* menu item in the decision tree interaction window and click the node to be pruned. Note that only a non-leaf node can be pruned, and the pruned node is not removed, while its all descendants are removed. The pruned node can be re-split or simply labeled to be a final leaf node.

8.7 DTViz Implementation and Experiment

We implement the approach described in this chapter with Visual J++ on Windows 98. We experiment the DTViz system with data sets from the UCI repository [126], including Adult, Iris, Car, Flag, Breast-Cancer, etc. In this section, we illustrate our experiment with the data set Adult (Income).

The Adult database was extracted from the 1994 Census database and used for data mining and visualization. Especially, the Adult database was used for scaling

up the accuracy of naive Bayes classifiers with a decision tree hybrid. A set of reasonably clean records was extracted using the following conditions:

$$((AAGE > 16)\&\&(AGI > 100)\&\&(AFNLWGT > 1)\&\&(HRSWK > 0)).$$

It contains 48,842 instances with a mixture of continuous and nominal attributes where training set has 32,561 instances and testing set has 16,281 instances. In the Adult database, 7% instances contains missing values.

The Adult database consists of 14 condition attributes with 6 continuous and 8 nominal attributes. The class attribute *Salary* has two values (two class labels), $> 50K$ and $<= 50K$. The attribute information is summarized in Table 8.1.

In our experiment, we only use the instances without missing values. This data set contains 30,162 training instances and 15,060 testing instances, in total 45,222 instances. In addition, 24.78% instances are labeled as $> 50K$, and 75.22% as $<= 50K$.

The initial training data set of 30,162 instances with 14 attributes is visualized in the *parallel segments*, shown in Figure 8.3. Because the data interaction window is not large enough to accommodate the entire training set, a random sample is made from the training set. 5,000 instances are sampled and visualized in Figure 8.4.

As discussed before, the decision tree root node is automatically generated and covers the entire sample set. To grow the decision tree, one must split this root node. To this end, one must select a split attribute. With the aid of status area and scroll bar in the data interaction window, we choose the attribute *Capital-gain* (the fourth segment from right) as the split attribute according to the first strategy of split attribute selection: *the clearer the clusters in a parallel segment, the better the corresponding attribute for splitting.*

Table 8.1: The Attributes Information of the Adult Database

Attribute	Type	Number of Values
Age	Continuous	
Workclass	Nominal	8
Fnlwgt	Continuous	
Education	Nominal	16
Education-Num	Continuous	
Marital-Status	Nominal	7
Occupation	Nominal	14
Relationship	Nominal	6
Race	Nominal	4
Sex	Nominal	2
Capital-Gain	Continuous	
Capital-Loss	Continuous	
Hours-Per-Week	Continuous	
Native-Country	Nominal	41

Using the *Best Pure Cluster Partitions*, this segment can be partitioned into three clusters. By means of scroll bar, however, we can see that around the separation between the middle cluster and the bottom cluster, pixels correspond to the same attribute values: *Capital-gain = 0*, though they are rendered to different labels. This observation shows that the separation between these two clusters is not a reasonable split point. In contrast, the separation between the middle and top clusters is an appropriate split point because the class label changes when moving

the mouse pointer from one side to another. Split the attribute *Capital-gain* at this point, then the root node is split into two children nodes, shown in Figure 8.7.

For simplicity, we choose the *resubstitution sample* method to evaluate decision tree nodes. Using the interactive operation *Show* (check/evaluate) to evaluate these two new nodes, one can find out that the top node has the classification accuracy of 62%, while the bottom node 98% with coverage of 205 instances. If this evaluation is satisfying, one can label it, or can split it. In this experiment, we think this accuracy is pretty high and satisfactory, thus label the node as a final leaf node of the decision tree. Figure 8.7 illustrates that this node is labeled as $> 50K$.

On the other hand, the top node in the second level in Figure 8.7 has very low accuracy, hence needs to be split further. Choose this node and observe the sub data set covered by this node in the data interaction window, illustrated in Figure 8.8. From Figure 8.8, one can see the fourth column from right is blank. This is because it corresponds to the attribute *Capital-gain* that is already split in the root, a node on the path from the root to the current node.

Repeat the above process to split the top node. According to the attribute selection strategy and attribute split strategy, we choose the attribute *Relationship* and three split points. Therefore, the top node is split into three children nodes, shown in Figure 8.9. Check these three new leaf nodes (without labels), one can find out that the middle node has high classification accuracy (99%) and large coverage (645), and thus can be labeled as the most frequently occurring label in the node data set, $<= 50K$ in this case, illustrated in Figure 8.9. However, the other two nodes are not satisfying by evaluation, and must be split again.

Figures 8.10 and 8.12 show the data set covered by these two nodes, respectively. Figure 8.11 illustrates the decision tree that is obtained from Figure 8.9 by splitting

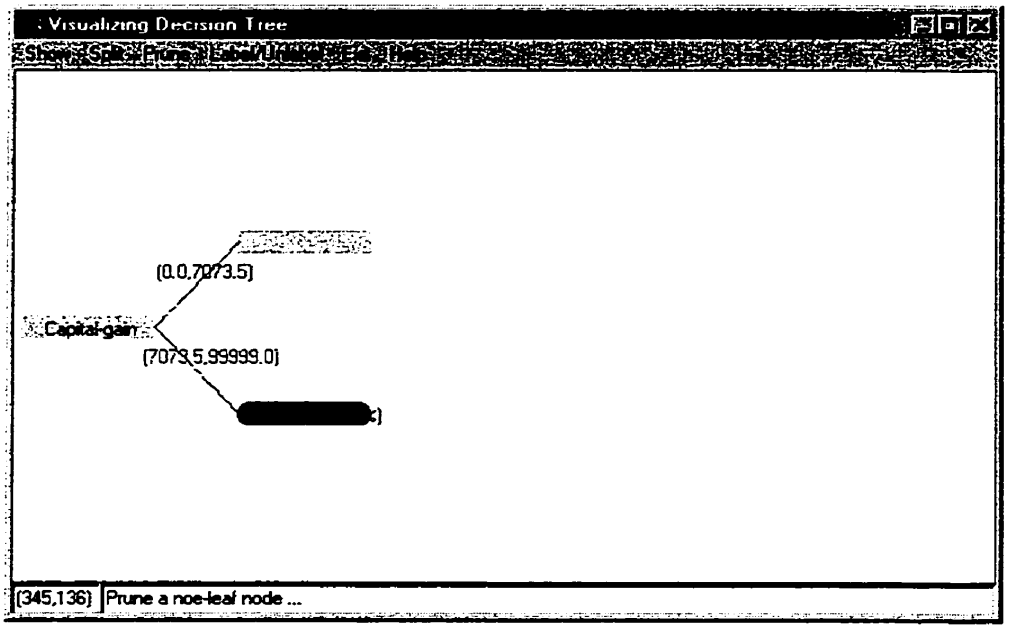


Figure 8.7: Split the Attribute *Capital_gain* into Two Parts at the Root

the attribute *Sex* into two partitions at the bottom node. The node *Sex* has two children, one having accuracy of 95% and coverage of 955 instances and thus being labeled as the final leaf node, and the other 68% accuracy and 2,566 instances. If the latter node is labeled, the resulting decision tree will not be satisfactory. Hence, one can unlabel it and split it since it has a large coverage.

Figure 8.13 illustrates the decision tree that is obtained from Figure 8.9 by splitting the attribute *Fnlwgt* into two partitions at the top node. Like the node *Sex*, the node *Fnlwgt* also has two children, one having accuracy of 100% and coverage of 420 instances and thus being labeled as the final leaf node, and the other 50% accuracy and 206 instances.

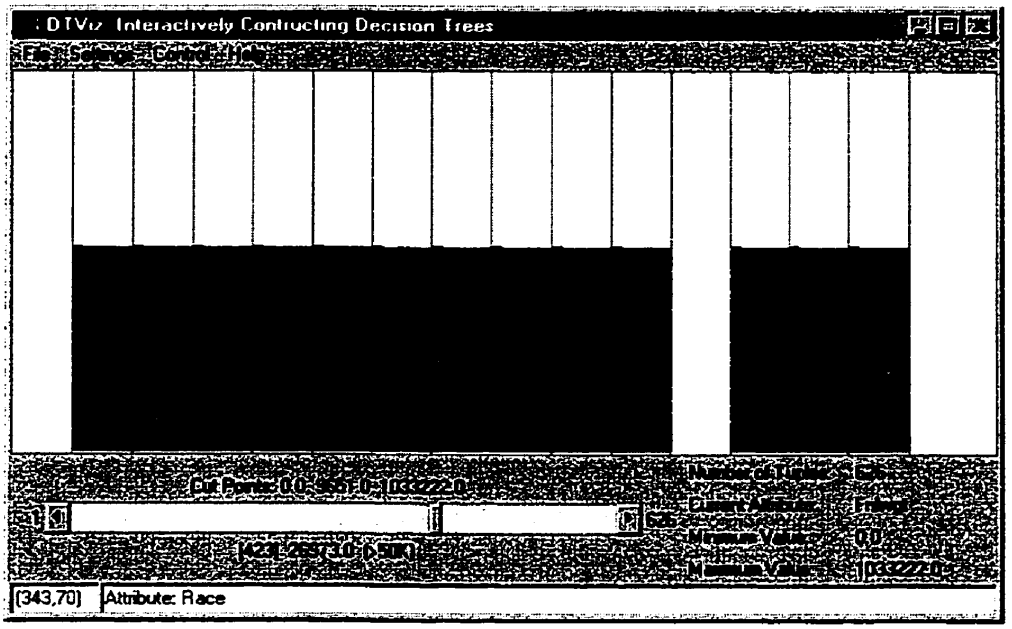


Figure 8.8: The Data Tuples Covered by the Top Node in Figure 8.7

8.8 Discussion

In this chapter, we presented an approach for interactively constructing decision tree classifiers and implemented the DTViz system. DTViz contains two interaction windows, *data interaction window* and *decision tree interaction window*. The former visualizes the training data covered by the current decision tree node, while the latter visualizes the decision tree being constructed and provides five interacive operations.

To visualize the training data, a pixel-oriented technique, *parallel segments*, is developed. The strategies for selecting split attributes and split points are discussed. The interactive operations help the user grow, prune, and revise the decision tree iteratively until the final result is satisfactory.

Like the CVizT system, the DTViz system also has the following characteristics,

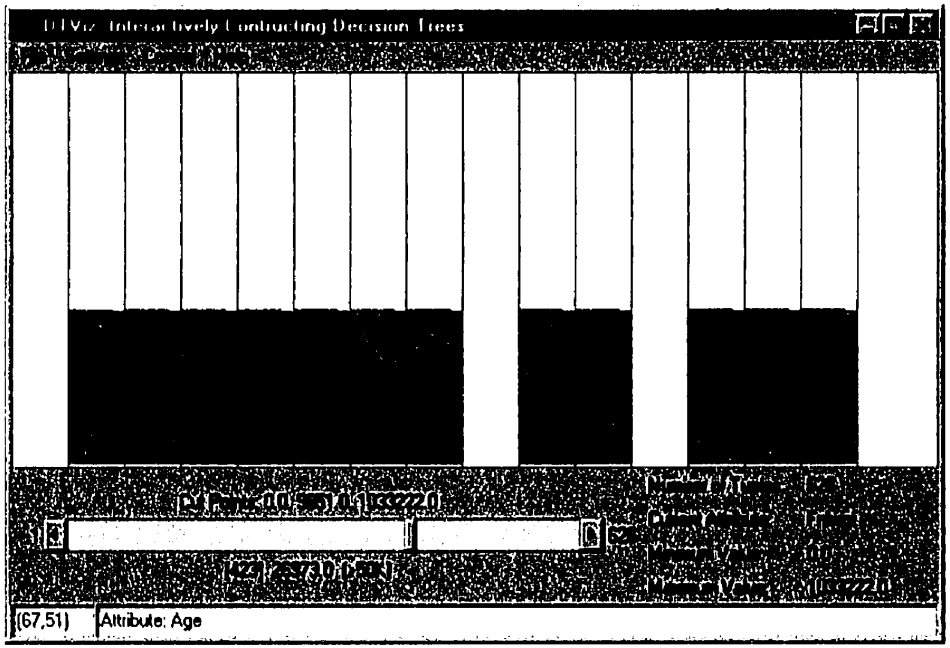


Fig. 8.10: The Data Tuples Coveed by the *Sex* node in Fig. 8.11

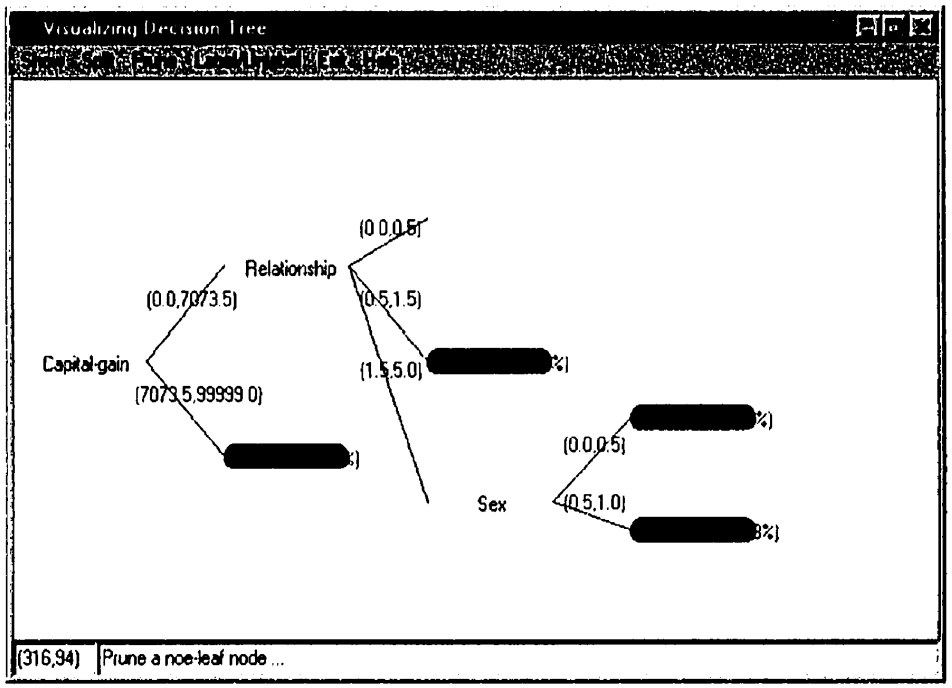


Figure 8.11: Split the Attribute *Sex*

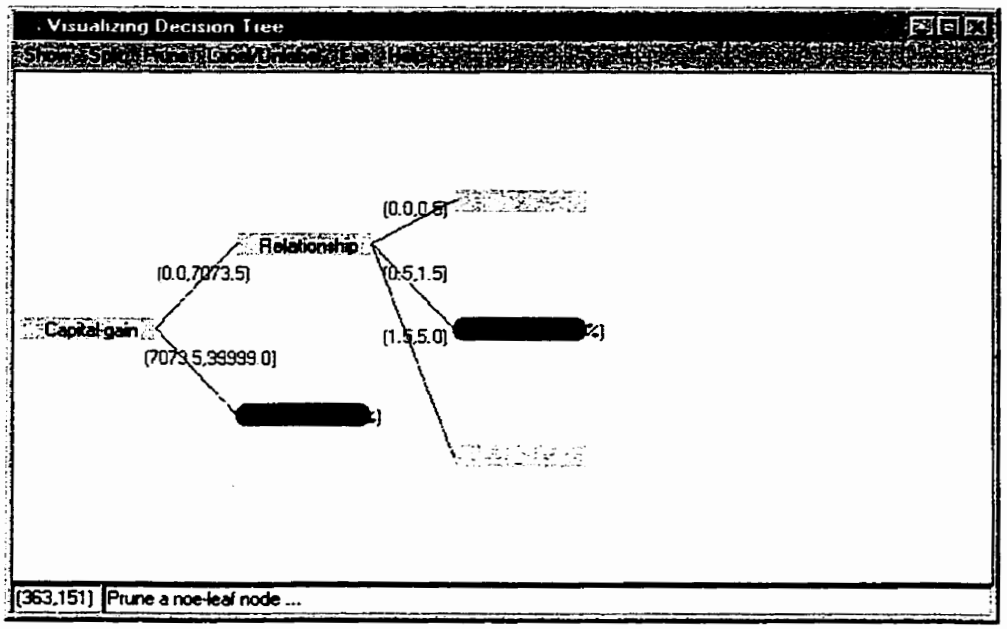


Figure 8.9: Split the Attribute *Relationship* into Three Parts

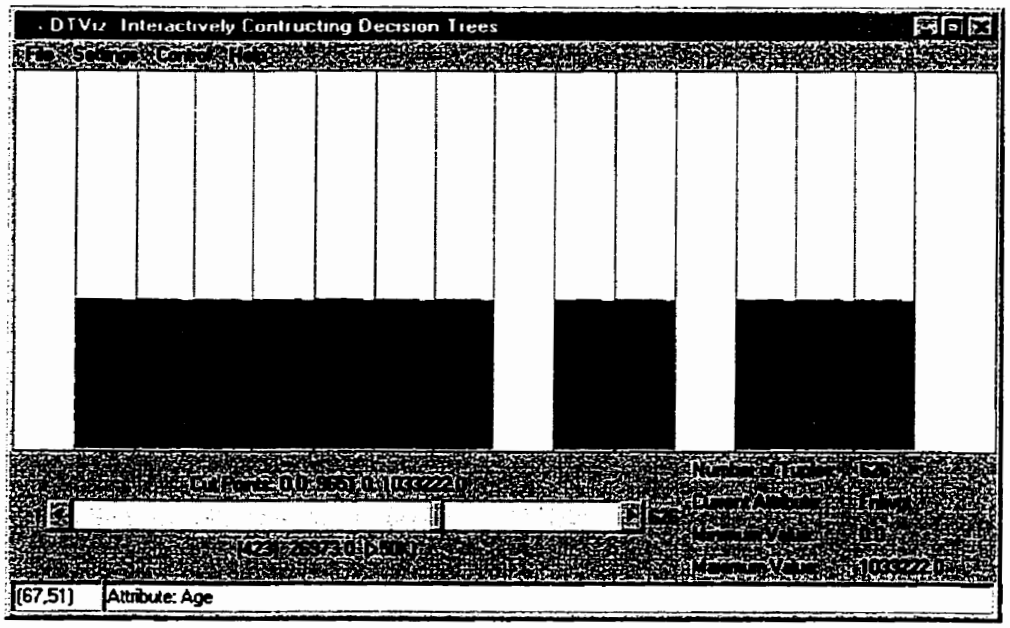


Figure 8.10: The Data Tuples Covered by the *Sex* Node in Figure 8.11

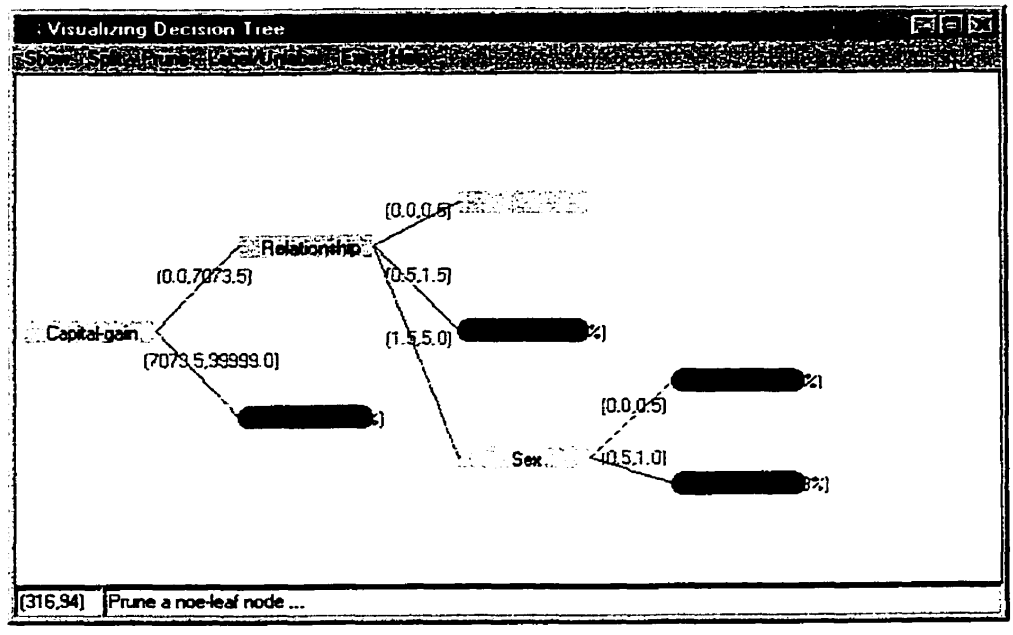


Figure 8.11: Split the Attribute *Sex*

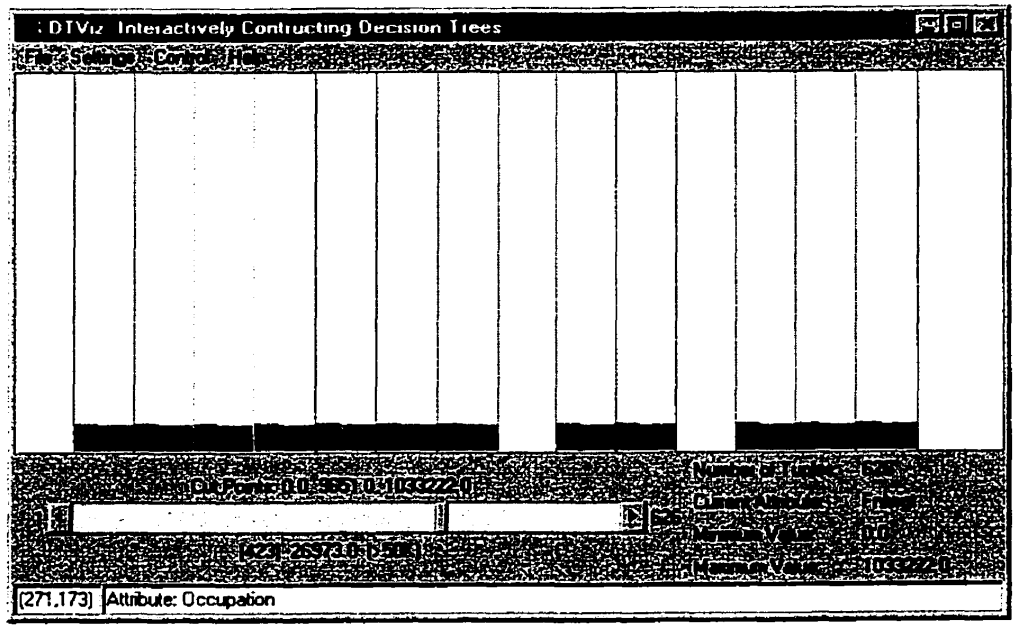


Figure 8.12: The Data Tuples Covered by the Node *Fnlwgt* in Figure 8.13

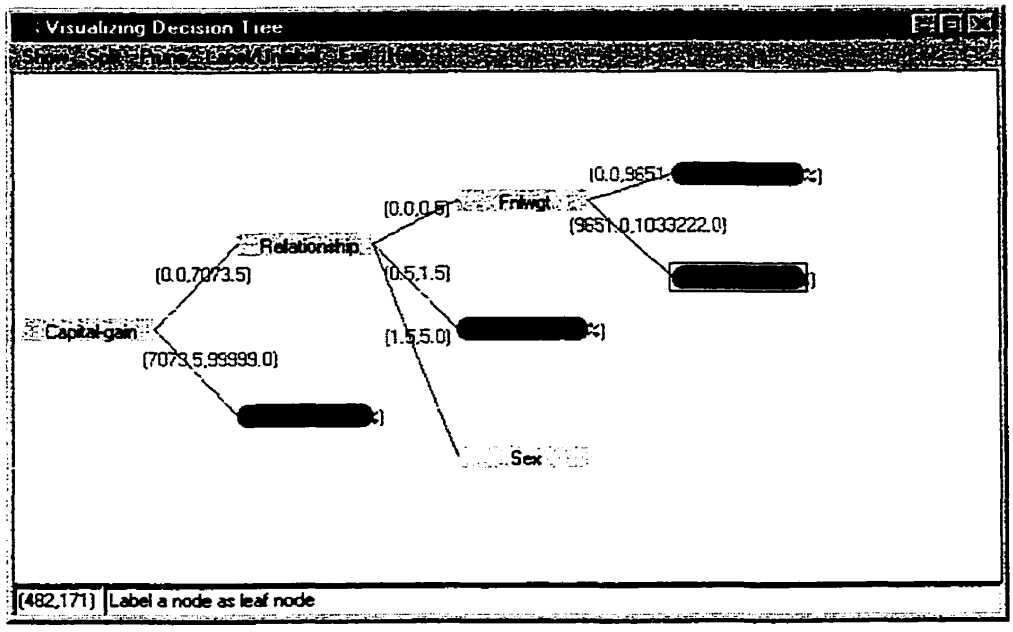


Figure 8.13: Split the Attribute *Fnlwgt*

compared to other approaches for constructing decision tree classifiers:

- easy to use,
- different resulting decision trees for the same data set,
- varying accuracy,
- understandable decision tree structure,
- on-demand node split and continuous attributes discretization, and
- uniformed process of categorical and continuous attributes.

However, in our experiment, we find that it is still difficult for the user to determine the splitting attributes and split points, especially the split points, for some

data sets. The selection of splitting attributes is important because it dominates the decision tree size and its classification accuracy. Therefore, we conclude that some facilities based on human perception should be developed for aiding the selection of split attributes and split points.

Chapter 9

Evaluation and Comparison

9.1 Introduction

In previous chapters, we developed four interactive knowledge discovery systems with visualization, AViz, CViz, CVizT, and DTViz, respectively. Although these systems have different targets and exploit distinct techniques, they share a common characteristic: following the general process of knowledge discovery with the process being divided into five components, including data preparation and visualization, data reduction, data preprocess, pattern learning, and knowledge visualization. This process is modeled in our RuleViz model.

As discussed in Chapter 3, the RuleViz model can be implemented in different ways, not only because the different components of RuleViz can be implemented with different techniques, but also because some components can be interchanged and/or interleaved. For example, the *data reduction* component can be done before or after the *data preprocess* component. Another possibility has the following order: *selecting features*, *preprocessing data* by discretizing continuous attributes,

selecting tuples, and then *handling missing values*. The RuleViz model can also be implemented in such a way that components are executed iteratively.

The four interactive systems discussed in previous chapters are implemented based on the RuleViz model. They, however, exploit different techniques for different tasks. For example, the AViz system explores an image processing-based approach to map the data items to an image and then to find the optimized area in the image. The CViz system, on the other hand, embeds different existing algorithms into different components, allows the user to interactively specify parameters for the algorithms, and prompts the results of algorithms to the user. Nevertheless, the CVizT and DVizT systems implement truly "supervised" learning systems, because the learning systems are completely supervised by the user.

In this chapter, we characterize these four systems and compare their techniques and effectiveness. Three implementation paradigms of the RuleViz model are identified from the four implemented systems. The techniques used in the components of these implemented systems are compared, and the capabilities and limitations of the four systems are analyzed. We also compare these four systems with the well-known algorithm-based systems and visualization-based systems, respectively.

9.2 Implementation Paradigms

From the implementations of the AViz, CViz, CVizT, and DTViz systems presented in previous chapters, three implementation paradigms can be identified as follows: *image-based algorithmic approach*, like AViz; *embedded-algorithm-based approach*, like CViz; and *user-supervised interactive approach*[1], like CVizT and DTViz. In

[1]The term *algorithmic* is contrast to the term *interactive*. In the interactive systems such as CVizT and DTViz, no algorithms is needed to fulfill pattern discovery tasks in their implemen-

this section, we briefly discuss these three implementation paradigms.

9.2.1 Image-based Algorithmic Approach

As we have seen in Chapter 5, the AViz system is used to visualize the discovery process of association rules. AViz first maps tuples from the raw data sets to three-dimensional tuples in terms of two conditional continuous attributes and one continuous or categorical attribute specified by the user, and then plots the tuples on the visualization window. The obtained image reflects the data distribution and attribute value distribution. The user can interactively choose areas in which s/he is interested by using a rubber band to reduce the data if necessary. The reduced data are redrawn and a new image is obtained. This can be repeated until the user is satisfied with the obtained image. The final data distribution image is discretized according to *bin-packing based equal-depth* method, one of discretization approaches provided in the AViz system, and a grid is acquired. Each cell in the grid is rendered with the color that is calculated on the basis of *support* and *confidence* of the cell. Finally, the optimized rectangles with the highest *gain* is found according to the colors of cells contained in the rectangles.

We call the implementation of AViz as *image-based algorithmic approach* because each component in the AViz system is implemented based on the algorithms that process images. We say that the AViz system is *image-based*, meaning that the raw data are mapped into an image and are reduced based on the image, the continuous attributes are discretized in terms of the image, and the optimized rectangles are found on the basis of the image. Three kinds of images are exploited

tation. This, however, does not mean that no algorithms is used to control the systems, e.g., the window layout and coordination between components.

in the AViz system. The first kind of image is obtained by plotting projected tuples in the data visualization component, data reduction component, and data preprocess component. The second kind of image is generated by rendering cells in the grid produced through attribute discretization, where the cell colors reflect the cell *support* and cell *confidence*. These two kinds of images are two-dimensional. The third kind of image is three-dimensional and rotated continuously, generated by rendering the gain optimized rectangles and leaving other cells unrendered.

Moreover, we say that the AViz system is *algorithmic*, meaning that its components are implemented with specific algorithms. For example, the raw data are projected and mapped with the corresponding algorithms, the continuous attributes are discretized by algorithms and the optimized rectangles are discovered through an algorithm.

However, the AViz system is not fully *image-based algorithmic approach*. For example, the data reduction is image-based but not algorithmic. In contrast, it is interactive because the raw data is reduced by the user using rubber band to specify the interesting area, instead of finding interesting area by calculating some interestingness measures. In addition, the AViz system provides three approaches for discretizing continuous attributes, only the *bin-packing based equal-depth* approach is image-based, while the other two approaches are not. The *equal-sized* approach discretizes continuous attributes in terms of the number of intervals and the attribute ranges, having nothing to do with the images. The *interaction-based* approach is not an independent approach and assists the other two.

9.2.2 Embedded-Algorithm-Based Approach

The CViz system presented in Chapter 6 is used to visualize the entire process of classification rules induction. CViz first divides the display window into parallel coordinates and then draws the tuples from the raw data sets as polylines across the parallel coordinates by executing a transformation and scale algorithm. The raw data can be reduced by randomly sampling with the condition that a certain number (a specified threshold) of tuples must be contained in each class although the visualization-based tuple selection is also available, and/or by selecting significant features based on the RELIEF algorithm that is embedded in CViz. The missing attribute values in the reduced data set, if any, can be estimated as the most frequent values of the same attribute with respect to the same class labels, which is performed by the embedded *missing values handling* algorithm, while the continuous attributes can be discretized by running the embedded EDA-DB discretization algorithm based on entropy of intervals. The preprocessed data are input as parameters to the learning algorithm ELEM2 that induces a set of classification rules from the input data by performing a general-to-specific search in a hypothesis space. Finally, the resulting classification rules are visualized as *strips* across the related parallel coordinates by an embedded *rule visualization* algorithm.

The implementation of the CViz system is recognized as a paradigm, called *embedded-algorithm-based approach.* The characteristics of this implementation paradigm are as follows:

- Each component in the system is implemented based on one or more embedded algorithms;
- The input parameters for each algorithm are controlled by the user, newly added or prespecified; and the input data are usually the outputs of previous

components.

- The outputs of each algorithm are presented to the user in some predefined-visual form;
- The user can perceive the visual outputs, choose to repeat some algorithm or related algorithms by adjusting parameters to see the effects of parameters, and decide what parameters are the best or satisfactory.

The CViz system is, of course, not a full *embedded-algorithm-based approach* because it contains components that are implemented with a mixture of embedded algorithms and interactions. For example, the data reduction can be interactive if the user chooses to specify the attributes that are not interesting by *delete-clicking* the corresponding coordinates and/or removing continuous attribute intervals and/or categorical attribute values that probably contain many outliers by using rubber band. The continuous attribute discretization can also be interactive if one chooses to use the *bin-packing based equi-depth* method. In spite of this, the CViz system can be actually a complete *embedded-algorithm-based* implementation of the RuleViz model. As a matter of fact, the interactive parts in the CViz system can be removed without affecting the completeness of CViz as an interactive visualization system for knowledge discovery. The reason that these interactive parts were included in the CViz system is in order to promote the flexibility of CViz and to compare the effects of different methods.

9.2.3 User-supervised Interactive Approach

The CVizT and DTViz systems provide another paradigm for implementing the RuleViz model. In the implementation of CVizT, the raw data are visualized in

the Table Lens with each attribute value of a tuple being drawn as a color line in the corresponding column. The user can interactively remove uninteresting attributes and/or aggregate several correlated attributes into one. The continuous attributes are discretized on demand to construct the condition part of classification rules, and the discovered rules occupy the display space in which the tuples covered by the rules were drawn. The main characteristic of this implementation is the full interaction between the user and machine, meaning that all components of the CVizT system are interaction-based.

Similarly, the DTViz system is also fully interactive. The data visualization window is used to interactively visualize the raw data and decision tree node data, select features, and discretize continuous attributes, while the decision tree visualization window is used to interactively grow and prune the decision trees according to the current tree structure and node evaluation.

Another important characteristic of CVizT and DTViz is that three components, including data preprocess (e.g., continuous attribute discretization), pattern learning, and pattern visualization are combined together to form an iterative process. In the CVizT system, a classification rule is generated in each iteration, while in the DTViz system, a decision tree node is split in each iteration.

We call this paradigm of implementation of the RuleViz model as the *user-supervised interactive approach.* By *user-supervised* we mean that the knowledge discovery process is completely supervised by the user, and the user decides how each step is performed. In summary, the user-supervised approach for implementing the RuleViz model has the following features:

- The relevant data identification, data preprocess, and the pattern recognition capabilities of the human beings can be explored to increase the effectiveness

of pattern learning.

- The user specifies the task, focuses on the search, and evaluates the intermediate results of the process. Therefore, the domain knowledge can be easily integrated into the systems.

- The user has a deep understanding of the resulting patterns because the discovery process is guided by the user. Hence, the user usually trusts into the results.

- The data preprocess is not an independent component, and usually is done on demand. Thus the preprocessing cost is often reduced. For example, the continuous attributes are discretized only when they are used to split decision tree nodes in the DTViz system, while the attributes that are never chosen at any nodes will not be discretized. Moreover, the same continuous attribute may have different splits in different decision tree nodes. In contrast, most algorithm-based systems discretize continuous attributes before the mining algorithms begin [4, 11, 40, 89, 106, 111, 113, 117, 125, 133, 135], and can not be changed during the algorithm execution.

- The user can *undo* and *redo* tasks in any steps at will. The knowledge pattern to be discovered usually consists of more than one components, e.g., a classifier consists of a set of classification rules and rules in turn are composed of a set of *attribute-value* or *attribute-interval* pairs, while a decision tree includes a set of nodes connected together. The user-supervised approach interactively constructs the knowledge pattern by building its components, e.g., constructing classifiers by building classification rules, constructing decision trees by building tree nodes. Using *undo* and *redo* can add new components and delete the components that are constructed already until the result is satisfactory.

- Compared to naive Bayes classification or other subject probability based approaches [123], this approach is totally subjective, fully depending on the user.

In addition, a combination of above three implementation paradigms is also possible. In a hybrid implementation paradigm, different components of the RuleViz model can be implemented in different ways. For example, as illustrated above, the data reduction can be interactively performed under the supervision of the user in the CViz system that is mainly embedded-algorithm-based, if the user decides to select significant features by observation and interaction and/or to choose significant intervals for continuous attributes by using rubber band. Moreover, the user-supervised interactive approach can be integrated with algorithms such as feature ranking, split attribute evaluation, etc. in decision tree construction to aid the user to decide which attribute is the best for the split in a specific decision tree node in the case that it is difficult to directly observe the attribute values distribution. We will discuss in more detail about the cooperation of the user and the machine for knowledge discovery in next chapter.

9.3 Components Comparison

From the point of view of component implementation, the AViz, CViz, CVizT, and DTViz systems explore different techniques for their own different tasks, although they have the same components. In this section, we take into account the implementation techniques used in these system implementations. Tables 9.1, 9.2, and 9.3 enumerate the techniques explored in data/pattern visualization, data reduction, and data preprocessing and pattern learning, respectively, of these four systems.

9.3.1 Learning Targets and Visualization Techniques

The four systems have different learning tasks, which decides that they use different visualization techniques. The visualization consists of the following aspects:

- *Raw data visualization*, visualizing the raw data tuples with attributes and attribute values;
- *Current available data visualization*, visualizing the data tuples and/or attribute values available for the current tasks or components, for example, node data in a decision tree construction, a subset of data as the input of an embedded algorithm in an embedded-algorithm-based approach;
- *Intermediate result visualization*, visualizing the intermediate results of components, for example, the cleaned data after data reduction in AViz, attribute value intervals after discretizing continuous attributes in AViz and CViz, the classification rule being constructed in CVizT, and the current decision tree once a node is split or a branch is pruned in DTViz; and
- *Final pattern/knowledge visualization*, visualizing the resulting patterns such as association rules, classification rules, and decision trees.

Table 9.1 shows the learning tasks of the AViz, CViz, CVizT, and DTViz systems, as well as the data visualization and knowledge/pattern techniques developed in these systems, respectively. From the table, one can see that the data visualization techniques and knowledge visualization techniques are strongly related, which means that the knowledge visualization is based on the corresponding data visualization, especially in the CViz and CVizT systems. Similarly, the AViz system has a three-dimensional image for association rules discovered from the two-dimensional

image of data tuples. However, the DTViz system is an exception. Its knowledge visualization (tree structure) has nothing to do with the corresponding data visualization (parallel segments). This is because the decision tree as a pattern has its own intrinsic structure which is understandable by most people. No matter what data visualization techniques are used, the pattern should be visualized as the tree structure. Otherwise, the purpose (e.g., understandable, straightforward, etc.) of visualization diminishes. On the other hand, the learning tasks of the AViz, CViz, and CVizT systems are finding association rules and classification rules, respectively, which have no intrinsic physical structures. Although they have their own internal logical structures, this kind of structure is hard to understand. Thus, the raw data, intermediate data and result, and final result follow the same or similar visualization techniques could be very helpful for the user to understand what is going on and to participate easily in the entire process such as providing parameters for algorithms, choosing appropriate points for attribute discretization, evaluating intermediate results, and interpreting the final patterns.

Table 9.1 does not show the visualization techniques for intermediate result and current available data. Actually, in these four systems, the current available data visualization adopts the same techniques as the raw data visualization techniques. However, the intermediate result visualization adopts different strategies in different systems. Recall their implementations presented in previous chapters.

- In the AViz system, the discretized attributes are visualized as interval values corresponding to *cells*. Cells are the primary and undividable units for finding the gain optimized rectangles. This is different from both the raw data visualization technique (2D plots) and the pattern visualization (3D rotation planes).

Table 9.1: Learning Targets and Visualization Techniques of AViz, CViz, CVizT, and DTViz

	Patterns to be learned	Data Visualization Techniques	Knowledge Visualization Techniques
AViz	Association Rules	2D Plots (Geometric Projection)	3D Rotation Planes (Geometric Projection)
CViz	Classification Rules	Parallel Coordinates (Geometric Projection)	Parallel Coordinates-based Rule Strips (Geometric Projection & Graph-based)
CVizT	Classification Rules	Table Lens (Graph-based)	Table Lens-based Rule Lens (Graph-based)
DTViz	Decision Trees	Parallel Segments (Pixel-oriented)	Tree Structure (Graph-based)

- In the CViz system, the discretized attributes are treated as categorical ones, and the tuple lines that fall in the same intervals are aggregated. This is neither the same as the raw data visualization nor the same as the final patterns (rule strips), although they are similar when visualized on parallel coordinates.

- In the CVizT and DTViz systems, the intermediate results are treated the same as the final patterns, because both are user-supervised interactive systems and the user needs to know exactly whether what is being constructed is the same as expected.

In addition, in order to visualize the raw data sets and current available data sets, the data tuples must be transformed into appropriate forms from the Data Tables (see Chapter 2). According to the data visualization techniques used in our implementations, the AViz, CViz, CVizT, and DTViz systems develop different data transformations. In AViz, attribute values are scaled in terms of the attribute ranges and the projected points are proportional to the number of pixels along the axes. In CViz, attribute values are scaled and mapped onto the parallel coordinates. The coordinate points transformed from attribute values are proportional to the coordinate height. In CVizT, the attribute values are mapped to segments of lines which are proportional to the column width. While in DTViz, the attribute values are not scaled. They are sorted and mapped onto the corrseponding pixels line-by-line in ascending order.

9.3.2 Data Reduction Techniques

We only discussed two aspects of data reduction, *feature selection* and *tuple selection*. Table 9.2 enumerates the feature selection techniques and tuple selection techniques used in the AViz, CViz, CVizT, and DTViz systems, respectively.

From Table 9.2, one can see that three kinds of feature/tuple selection techniques are used in our implementations.

- *Manual selection*: Through the dialog window, the user can arbitrarily select features, even tuples, without seeing how the data are distributed. Only the AViz system uses this technique to select features. Manual selection is simple and easy to implement. However, this selection is blind. Before the user has a feel about the data tuple and attribute value distribution by visualizing the data set, s/he does not have any idea about which features are strongly

Table 9.2: Data Reduction Techniques Used in AViz, CViz, CVizT, and DTViz

	Feature Selection	Tuple Selection
AViz	Manual (through dialog window)	Interactive (using rubber band)
CViz	Algorithmic (Embedded Algorithm RELIEF) and Interactive (by clicking coordinates to be removed)	Algorithmic (random sampling) and Interactive (using rubber band)
CVizT	Interactive (by clicking columns to be removed)	Algorithmic (random sampling)
DTViz	Interactive (by clicking columns to be removed)	Algorithmic (random sampling)

correlated and thus are better to be selected as significant features. To do better selection, the user must have prior knowledge about the data set or background knowledge about the application domain.

- *Algorithmic selection*: This technique executes an algorithm for selecting features and tuples. In our implementation of CViz, CVizT, and DTViz, tuple selection are drawn by random sampling algorithms. Usually, many tuple selection algorithms can be used [16, 155], but the random sample is still the most common algorithm, because, in most realistic cases, nothing about the

data tuple distribution is known *a priori*. In the cases that the data distribution is known in advance or as background knowledge, special algorithms for selecting tuples can be used [33, 78].

For the feature selection, the CViz system embeds an algorithm based on RELIEF, while other systems use interactive selection.

- *Interactive selection*: This technique provides the user with an interaction tool to select interesting data tuples and/or significant features. Usually, tuples and features are selected on the basis of data distribution. By visualization, the user can easily observe the distribution of the *entire* data set, but it is difficult to decide which *individual* tuples are interesting, especially for high dimensional data sets. Moreover, the interaction tools for selecting tuples are also not easily designed because multidimensional data must be visualized on the two-dimensional visualization window. This depends on the visualization techniques. Therefore, in our implementations, only *rubber band* are provided for interactively selecting tuples in the AViz and CViz systems, because the AViz system only visualizes the two-dimensional tuples projected from high dimensional data set, and the CViz system maps data tuples onto parallel coordinates. On the other hand, according to the visualization technique *parallel segments* used in the DTViz system, the user can not see what is an individual tuple because the attribute values of the same tuples are distributed in different segments. In the CVizT system, the interaction tool similar to *rubber band* can be used to select tuples. However, the tuples in the same classes are aggregated. Selecting an interesting area will cover a subset of data tuples concentrating in few classes, even only one class. This is unfair for other classes. Our experiments illustrate that the interactive tuple selection in CVizT is not easy to obtain good results.

Interactive feature selections are explored in CViz, CVizT, and DTViz. These three systems use visualization techniques based on parallel coordinates or segments. The values of the same attributes concentrate on one column or one coordinate. It is comparatively easy to see how these values are distributed, and the user is able to decide which attributes are significant and which are not in terms of their values distribution.

Actually, the techniques for feature and tuple selections depend not only on the data visualization techniques but also on the learning tasks. For learning tasks other than classification, the interactive feature selection developed in the CViz, CVizT, and DTViz systems may not be helpful, while other interaction tools for interactive data reduction can be developed. This is a challenging question, and becomes our future work.

9.3.3 Data Preprocess and Learning Techniques

In the implementations of the AViz, CViz, CVizT, and DTViz systems, two aspects of data preprocessing are considered, continuous attribute discretization and missing value handling. Table 9.3 lists the data preprocessing methods used in these systems.

There are two kinds of missing values, condition attribute values and class labels. All tuples with missing class labels are simply removed, because estimating class labels leads to another problem [80]. To handle condition attribute missing values, the CViz system embeds an algorithm for replacing the missing attribute values with the values that most frequently occur in the tuples with the same class labels as the tuples with missing values. This algorithm is similar to that explored in the CN2 induction algorithm [43]. It only considers the tuples in the same class without

paying attention to the correlations between attributes. The other three systems adopt the simplest method: remove the tuples with missing values. For large data sets, the tuples without missing values are sufficient to provide information needed by the learning algorithm. For small data sets, however, removing tuples with missing values is suspectable to information loss and the learning results are inadequate. Because the AViz and DTViz systems usually handle large data sets[2], the effect of removing tuples with missing values is not highlighted. The reason that the CVizT system simply ignores the tuples with missing values is that CVizT is built to interactively construct classification rules so no algorithms are included. We attempted to develop interactive approaches for handling missing values, but no good methods were found.

For continuous attribute discretization, the four systems provide interactive approaches. In the AViz and CViz systems, both interactive and algorithmic approaches are developed. The interactive attribute discretization is auxiliary to algorithmic approaches. That means, after continuous attributes are discretized by the given algorithms, the user can slightly adjust the cut points by dragging the discretization lines and moving to the expected points if the algorithm results are not satisfactory. In our experimemts, however, the interactive discretization is seldom used in the AViz and CViz systems, because the algorithm results are usually sufficiently good enough that any changes of cut points will lead to bad results (low confidence in AViz or low accuracy in CViz). Both AViz and CViz provide an image-based discretization algorithm, called *bin-packing based equi-depth*, which

[2]For small data sets, AViz and DTViz are difficult to discover satisfying patterns, because the data tuples are distributed in the visualization windows very sparsely. For AViz, the cells will have low supports, while for DTViz, it is difficult for the user to observe and decide the split points.

Table 9.3: Data Preprocess and Learning Techniques Used in AViz, CViz, CVizT, and DTViz

	Attribute Discretization	Missing Values Handling	Learning Techniques
AViz	Image-based Algorithmic & Interactive	Remove	Image-based Algorithmic
CViz	Image-based & Entropy-based Algorithmic & Interactive	Algorithmic (Most frequent value)	Embedded Induction Algorithmic (ELEM2)
CVizT	Interactive (Graph-based)	Remove	Interactive (Graph-based)
DTViz	Interactive (Pixel-oriented)	Remove	Interactive (Piexl-oriented & Graph-based)

was described in Chapter 5. Moreover, they also provide another approach, *equi-length*, for discretizing continuous attributes. In addition, the CViz system embeds an entropy-based discretization algorithm, EDA-DB. Table 9.4 shows the learning results of the ELEM2 algorithm for diffierent discretization approaches provided in CViz for at most 10 intervals, where the *Artificial* data set is described in Chapter 6, and the other four data sets are from the UCI repository [126]. From this table, one can see that EDA-DB and *bin-packing based equi-depth* methods have almost the same effect, but the *equi-length* leads to the worse results.

Table 9.4: Comparision of Three Continuous Attribute Discretization Algorithms Provided in the CViz System for at most 10 Intervals

	Equi-Length		Bin-packing based Equi-depth		EDA-DB	
Data Sets	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
Artificial	8	89.23%	6	92.98%	6	92.69%
Monk-1	9	91.23%	9	93.12%	9	94.38%
IRIS	8	81.45%	8	94.22%	8	94.67%
Glass	33	62.54%	32	73.44%	30	72.88%
Bupa	4	56.46%	4	67.76%	4	66.93%

The interactive construction systems CVizT and DTViz only develop interactive discretization methods. This is not only because both systems attempt to provide the user with interaction tools for every component of knowledge discovery process, but also because the data visualization techniques explored in them are convenient for the user to look into the cut points that separate different class labels. For more information about this claim, see Chapters 7 and 8.

Table 9.3 also lists the learning algorithms developed in the AViz, CViz, CVizT, and DTViz systems. Three techniques for learning patterns are developed in the four systems, respectively, as follows:

- The AViz system discovers the gain optimized rectangles each of which contain a set of cells from the image that is generated according to the data tuples covered in the cells. We call this technique an *image-based algorithmic approach* because the patterns are learned in terms of the image. There are

two distinct methods to implement the image-based approach. One is through image processing. That means the image is yielded from the data set, and the learning algorithm analyzes the hue, satuation, and intensity, or the red, green, and blue color components of pixels in the image, transforms, splits and merges pieces of images, as well as uses other image processing techniques [161] to find the interesting pieces of images that represent expected patterns. The other method is to record the mappings between the image and the data distribution (e.g., mapping between the cell color and the cell data such as cell support and cell confidence), and then process the image through the cell data. The final patterns are actually obtained from the cell data. The AViz system adopts the latter method.

- The CViz system embeds an induction algorithm ELEM2 into the pattern learning component. In the implementation, there are many approaches for embedding well-known algorithms. The CViz system employs a simple but succinct approach that looks like a client/server structure, where the CViz system is the client, and the embedded algorithm ELEM2 is the server. The request from the client contains the input data and parameters that is stored in a data file. The server accepts the request and the data and parameter file name, and executes the induction algorithm. The answer from the server are stored into another data file, and the client reads and processes the results from the data file. Other embedding approaches are possible. For example, including the learning algorithm as a part of the learning component is one choice. This approach, however, is not flexible for changing or revising the embedded algorithm.

- The CVizT and DTViz systems develop interactive approaches for construct-

ing classification rules and decision trees, respectively. In these systems, no learning algorithms are included, but interaction tools are provided. By using interaction tools, one can construct the classifiers based on his/her perception of the data distribution and intermediate results. The final patterns constructed in this way have the following features[3]:

- *Unfixed results*: The outcome patterns vary with users and runs. Even if the input data and the user are the same, the results may change. On the other hand, for the algorithmic approach, the output results should be unchanged if the input data and parameters are the same.
- *Understandable results*: It is easy for the user to understand the constructed patterns because the construction is supervised by the user. Considering that the algorithmic approach is called "black-box" because the user can not see the deails of the learning process, the interactive approach is refered to as "white-box".

9.4 System Capabilities

We evaluate three capabilities of the four systems we implemented: the size of data sets that the system can deal with, the accuracy, and the understandability of the results that the system generates.

[3]The algorithmic approach in other components, e.g., data preprocess, data reduction, etc. have the similar features.

9.4.1 The Size of Data Sets

The size of the data sets that the system can handle well depends on their data visualization techniques. Table 9.5 lists the number of features and the number of tuples that the four systems can properly process respectively, as well as the abilities that they deal with different feature types. One may argue that if the visualization window can be scrollable, the size of data sets that the systems can process can be considerably enlarged, and theoretically unlimited. However, scrollable windows are impractical in our implemented systems. The purpose of data visualization is to not only allow the user to perceive the data distribution by a glimpse of the visualized pictures but also provide the user interaction tools to select features and/or tuples. Scrollable windows will distract the user's focus, although it can accommodate large size of data. This is because the user is not able to view the entire scrollable window simultaneously to grasp the significant features or data areas if the data are distributed in more than one screen. Our four systems do not support scrollable windows.

For the AViz system, three features can be selected each time, two condition attributes and one decision attribute, and only the two condition attributes, which must be continuous attributes, are visualized. No categorical condition attributes are allowed. Because we address the discretization of continuous attributes, all systems can handle continuous attributes. The number of tuples that AViz can properly visualize is affected by three factors: visualization window size, attribute value range, and attribute value distribution. In theory, the unlimited number of tuples can be visualized because the tuples are mapped onto the pixels in the display, and different tuples may be projected to the same pixel. The problem is, if many distinct tuples are projected to the same pixel, the image obtained will be

Table 9.5: The Size of Data Sets That AViz, CViz, CVizT, and DTViz Can Properly Process

	Number of Features*	Feature Types		Number of Tuples*
		Continuous	Categorical	
AViz	2/3	Good	No	Large (>1,000,000)
CViz	$6 \sim 12$	Good	Good	Medium ($\approx$ 1,000)
CVizT	$6 \sim 12$	Good	Good	Medium ($\approx$ 500)
DTViz	$6 \sim 20$	Good	Bad	Large (> 15,000)

$\star$ *The number of features and the number of tuples are with respect to unscrollable windows.*

hard to distinguish due to the density of pixels. This will occur if the visualization window is small, attribute ranges are large, and tuples distribute unevenly. Our experiments show that the AViz system can properly visualize over $1,000,000$ tuples with the visualization window size of 600×500. In contrast, if the data set size is small, say less than $10,000$, the AViz system can not generate good results, because the image is too sparse for the user to observe interesting areas and for the learning algorithm to find *ample* cells.

The CViz and CVizT systems employ similar data visualization techniques, and both can properly process continuous attributes and categorical attributes in a uniform way. In CViz, the attribute values are distributed on the parallel coordinates, and categorical attribute values are mapped into integers and distributed evenly along the coordinates. Similarly, in CVizT, categorical attribute values are also mapped into integers which are processed as numerical attribute values which, in turn, are drawn as lines proportional to the column width. The only difference of

processing continuous and categorical attributes is when the user draws an interval by using "rubber band", the rubber band can cover any range in the coordinates (for CViz to remove tuples) or columns (for CVizT to select cut points) for continuous attributes, but can only cover one value for categorical attributes each time.

The number of features that CViz and CVizt can handle depends on the visualization window width. More features will lead to more narrow columns (for CVizT) and closer parallel coordinates (for CViz). The narrow columns will cause more distinct attribute values to be mapped to the lines with the same length, and thus the user finds it difficult to discretize attributes. The close parallel coordinates will cause the parallel lines corresponding to tuples are messy and hard to distinguish between tuples. Our experiments show that the number of features between 6 and 12 results in good visualization effects.

On the other hand, the number of tuples that CViz and CVizT can properly process depends on the height of the display window. In CVizT, each tuple occupies a row, so the number of tuples is equal to the window height, usually about 500. For CViz, the different tuples may overlap, thus more tuples can be accommodated. But too many tuples will make the picture hard to understand. Our experiments illustrate that the appropriate number of tuples in CViz is about $1,000$.

For the same reason mentioned above, the DTViz system can properly process continuous attributes, but the categorical attribute values can not be processed well, because the attribute values are sorted and mapped to pixels in ascending order. In most real world applications, however, categorical attribute values do not have natural order.

Because of the characteristics of pixel-oriented visualization techniques, in the DTViz system, the display window can accommodate about the same number of

data items as the number of available pixels. For the 600×500 display area, the number of available pixels is about $300,000$. If we assume the data set contains N tuples with M features, then $M \times N = 300,000$. For example, if $M = 20$, then $N = 15,000$.

9.4.2 Accuracy of Results

Table 9.6: The Data Sets for Experiments with the CViz, CVizT, and DTViz Systems

Data sets	No. of Classes	No. of Cond. Attrs.	No. of Examples	Domain
australia	2	14	690	Credit card application approval
balance-scale	3	4	625	Balance scale classification
breast-cancer	2	9	683	Medical diagnosis
bupa	2	6	345	Liver disorder database
diabetes	2	8	768	Medical diagnosis
glass	6	9	214	Glass identification for cirminological investigation
heart	2	13	270	Heart disease diagnosis
iris	3	4	150	Iris plant classification
tic-tac-toe	2	9	958	Tic-Tac-Toe Endgame database
wine	3	13	178	Wine recognition data

In our implementations, three of four systems focus on learning classifiers. One of the important evaluation measures of classifiers is the *prediction accuracy* [123]. From the characteristics of the CViz, CVizT, and DTViz systems, one can see that the prediction accuracy of the CViz system depends on the embedded learning algorithms, while the accuracy of the CVizT and DTViz systems are determined by the user and the user-machine interaction. In this subsection, we compare the prediction accuracy of these three systems.

To compare our three systems, we have conducted experiments with them on 10 data sets from the UCI repository [126]. The 10 data sets were selected with no particular agenda in mind other than convenience so that they cover different domains. Table 9.6 describes these data sets in terms of their number of concepts, number of condition attributes, number of examples, and application domains.

Table 9.7 illustrates the prediction accuracy of the CViz, CVizT, and DTViz systems on the data sets listed in Table 9.6. The accuracy evaluation of the CViz system is the same as that of the ELEM2 learning algorithm[4]. For the CVizT and DTViz systems, the predictive accuracy of the final classifiers is evaluated using *resubstitution evaluate* method, where the same data set is used as both training set and testing set[5], and estimated as follows:

- Five people including the author are trained to construct classifiers by using the CVizT and DTViz systems. Each of them is taught how to operate CVizT

[4]The accuracy of ELEM2 is excerpted from [11] which is calculated by the ten-fold evaluation on the 10 data sets. The accuracy mean is the average accuracy of the ten runs.

[5]With *resubstitution evaluatation*, the selected data sets are used for users to construct decision trees and classification rules, which are in turn used to predict the class labels of tuples in the same data sets. Other evaluation methods can also be employed and will be explored in the future experiments. See Chapter 10.

and DTViz for about 10 to 30 minutes.

- For each data set, each trainee operates the CVizT and DTViz systems to construct classifiers. Thus, we obtain five classifiers for each system. The predictive accuracy of each classifiers is recorded. Three of five classifiers for CVizT and DTViz are arbitrarily picked to avoid alias (These people have different backgrounds. Some of them are familiar with decision trees, some familiar with classification rules, and some familiar with both).

- The average accuracy of the selected classifiers for each system is taken as the accuracy mean of that system.

Table 9.8 shows the accuracy mean estimation of the CViz, CVizT, and DTViz systems. The *Low, Medium* and *High* columns represents the evaluation of the classifiers that the three people constructed, and the *Mean* column is the average evaluation of these classifiers.

From Tables 9.7 and 9.8, one can conclude

- A good user can construct better classifiers using CVizT and DTViz more so than the CViz system (the ELEM2 algorithm), while the bad user always obtain worse classifiers. The accuracy mean of the CVizT and DTViz systems is almost the same as that of the CViz system. Thus, the results discovered by using interactive construction methods depend largely on the user, while the results discovered by algorithmic approaches do not.

- For the data sets with which CViz has low accuracy, CVizT and DTViz can achieve high accuracy, for example, the data sets *bupa, diabets, glass*, and *heart*. However, for the data sets with which CViz has very high accuracy (approaximate 100%), CVizT and DTViz can only achieve low accuracy, for

Table 9.7: The Predictive Accuracy Comparisons among the CViz, CVizT, and DTViz Systems

Data sets	Accuracy Mean (%)			Standard Deviation (%)		
	CViz	CVizT	DTViz	CViz	CVizT	DTViz
australia	86.52	85.20	87.10	3.55	6.21	5.20
balance-scale	81.14	80.80	82.17	5.29	8.40	5.87
breast-cancer	96.19	93.67	94.23	2.51	6.25	3.92
bupa	69.24	72.10	72.73	7.88	6.77	6.44
diabetes	74.10	76.17	77.13	4.82	4.17	4.21
glass	72.88	79.37	77.10	10.11	12.28	11.65
heart	79.63	82.70	82.10	7.46	7.13	7.06
iris	94.67	92.10	92.97	5.26	4.76	4.69
tic-tac-toe	99.17	94.20	95.30	1.28	2.45	1.78
wine	97.78	93.53	95.43	3.88	5.22	3.57
average	85.13	84.08	85.62	5.20	6.36	5.44

example, the data sets *breast-cancer, iris*, and *tic-tac-toe*. Therefore, the algorithmic approach is more sensitive to the data sets (application domains) than the interactive approaches.

- Generally, the DTViz system constructs better classifiers (represented as decision trees) compared with the CVizT system (represented as classification rules) for 8 out of 10 data sets except *glass* and *heart* for which the accuracy mean of both systems is very close. Hence, the interactive construction of decision trees is more effective than the interactive construction of classification

Table 9.8: The Accuracy Mean Estimation of the CViz, CVizT, and DTViz Systems

	CVizT (%)				DTViz (%)			
Data sets	Low	Medium	High	Mean	Low	Medium	High	Mean
australia	78.00	85.50	92.10	85.20	82.20	86.80	92.30	87.10
balance-scale	74.50	82.60	95.30	80.80	77.50	81.70	87.30	82.17
breast-cancer	88.30	94.20	98.50	93.67	90.20	94.30	98.20	94.23
bupa	63.40	72.30	80.60	72.10	65.60	71.50	81.10	72.73
diabetes	69.00	74.90	84.60	76.17	70.40	74.80	86.20	77.13
glass	71.40	78.50	88.20	79.37	68.50	76.40	86.40	77.10
heart	76.60	82.70	88.80	82.70	75.30	82.60	88.40	82.10
iris	85.00	93.20	98.10	92.10	88.60	94.00	96.30	92.97
tic-tac-toe	88.40	96.20	98.00	94.20	92.10	95.30	98.50	95.30
wine	86.10	96.00	98.50	93.53	92.00	95.80	98.50	95.43
average				84.08				85.62

rules.

Table 9.7 also shows the standard deviation[6] of the three systems' predictive accuracy. Among the 10 data sets, CViz gives the best results in terms of predictive accuracy for 4 data sets, CVizT for 2, and DTViz for 4. In terms of standard

[6]The standard deviation is calculated using the following formula:

$$\textit{Standard Deviation} = \sqrt{\frac{n\sum x_i^2 - (\sum x_i)^2}{n(n-1)}}$$

where x_i is the accuracy from the i-th run ($i = 1, 2, \ldots, 10$ for CViz) or the i-th person ($i = 1, 2, 3$ for CVizT and DTViz) on a data set.

deviation, on 5 out of 10 data sets, CViz gives the smallest number, CVizT does it on 1 data set, and DTViz on 4. The average of the accuracy means and deviations of each system over the 10 data sets indicate that CViz is the most stable representations of the hidden patterns in the data sets, while DTViz is generally able to learn the most accurate results.

9.4.3 Understandability of Systems

Table 9.9: The Understandability Evaluation of the AViz, CViz, CVizT, and DTViz Systems

	Evaluation People					
	Person 1	Person 2	Person 3	Person 4	Person 5	Average
AViz	3	2	1	3	2	2.0
CViz	2	3	3	2	1	2.2
CVizT	3	2	3	4	3	3.0
DTViz	4	4	4	4	4	4.0

We also asked five people to evaluate the understandabilities of the four systems. The evaluation results are classified into four levels corresponding to degrees of *4–Excellent, 3–Very good, 2–Good, 1–Bad*, and the process of the systems and their results were evaluated. In this evaluation, we just evaluate how the user is able to understand the learning process and the output results, regardless of the correctness and accuracy of the results. The evaluation result is listed in Table 9.9.

From Table 9.9, one can see that the DTViz system has the highest understandability (*Excellent*) , while the AViz system has the lowest understandability (*Bad*),

and the CViz and CVizT systems have the intermediate understandability (*Good* and *Very Good* respectively). This is because the decision trees constructed by the DTViz system have intrinsic structure (tree structure) and they are understandable for users. The classification rules and association rules, however, have only logic structures which are not straightforward for the user to understand. Furthermore, the CViz system embeds algorithms which are not well known by the user, while the CVizT provides interaction methods to help the user construct rules. Therefore, the CVizT system is more understandable than the CViz system. The AViz system has the lowest understandability because tuples are projected onto two-dimensions and the output results are three-dimensional images. Complex data transformation causes the user confused. Moreover, association rules are more difficult than classification rules for the user to understand without a data mining background.

9.5 Comparison of AViz with SONAR

The AViz system is developed from the SONAR system [61] which only considers the learning algorithm based on the image processing as well as *cell* confidence and support threshold adjustment by sliders. The AViz system, however, consists of all knowledge discovery components, and explores new algorithms for data reduction and continuous attribute discretization. In addition, the AViz system extends the pattern to be discovered to three dimensions, while the SONAR system can only yield two dimensional patterns. Table 9.10 lists their components and methods employed in each component. From the table, one can see that both systems plot the data tuples on a two-dimensional display. The AViz system provides a "rubber band" to help the user select the interesting data areas such that the "outliers" can be cleaned; while the SONAR system does not clean the raw data, and simply uses

Table 9.10: Comparison of AViz with SONAR

	Raw Data Visualization	Tuple Selection	Continuous Attribute Discretization	Learning Patterns	Visual Patterns
AViz	2D Plots	Interaction	Equi-length Equi-depth Interaction	Optimized Rectangles	3D Animation
SONAR	2D Plots	N/A	Equi-length	Optimized Rectangle/ Optimized Admissible Region	2D Image

the entire data set. The SONAR system only employs the *equi-length* method to discretize continuous attributes, while the AViz system provides three approaches for attribute discretization, including *equi-length, equi-depth*, and *interaction-based.* However, the SONAR can be used to discover not only two-dimensional optimized rectangle but also two-dimensional optimized *admissible regions* [60, 61]; while the AViz system can find a set of optimized rectangles with respect to the values of the decision attribute, not *admissible regions.* For pattern visualization, the SONAR system statically displays the two-dimensional image containing the optimized area, while the AViz system displays the three-dimensional pictures in motion. The final results can be changed with the movement of threshold sliders in both systems.

9.6 Comparisons with Classifier Learning Systems

Table 9.11: Accuracy Comparsion of CViz, CVizT, and DTViz with the C4.5 and CN2 Systems

Data sets	C4.5	CN2	CViz	CVizT	DTViz
australia	82.90	84.92	86.52	85.20	**87.10**
balance-scale	78.10	79.05	81.14	80.80	**82.17**
breast-cancer	95.60	95.17	**96.19**	93.67	94.23
bupa	68.40	62.00	69.24	72.10	**72.73**
diabetes	70.80	71.21	74.10	76.17	**77.13**
glass	68.70	55.17	72.88	**79.37**	77.10
heart	80.80	77.85	79.63	**82.70**	82.10
iris	**95.30**	88.68	94.67	92.10	92.97
tic-tac-toe	96.80	98.44	**99.17**	94.20	95.30
wine	92.80	92.09	**97.78**	93.53	95.43
average	83.02	80.46	85.13	84.08	85.62

To further compare CViz, CVizT, and DTViz with well-known systems, we chose C4.5 and CN2, and conducted experiments with these algorithms on the data sets listed in Table 9.6. The C4.5 system is used to construct decision trees[7], while the CN2 system can build classification rules. When running C4.5, we use the option '-s' to allow grouping of attribute values. Other options in C4.5 are kept as default settings. For CN2, all runs use default parameters so that the

[7]The C4.5 system also has another version to build classification rules. The C5.0 system is also available so far.

Laplacian estimate can be employed as the search heuristic and unordered rules are generated, which means that the improved version of CN2 is used. For each data set and algorithm, we report the average of accuracies from the ten runs. The accuracy mean estimation of CViz, CVizT, and DTViz are the same as those in Table 9.8.

The best result for each data set is highlighted in boldface in the table. Among the 10 data sets, C4.5 gives the best results of predictive accuracy for only 1, CN2 for none, CViz (ELEM2) for 3, CVizT for 2, and DTViz for 4. At the bottom of the table, the average of the accuracy means of each system over the 10 data sets indicate that the DTViz system can discover the most accurate classifier (represented as decision tree), the CViz and CVizT systems are next, and C4.5 and CN2 learn the least accurate classifiers.

Chapter 10

Conclusion and Future Work

10.1 Conclusion

In this thesis we developed an interactive model for knowledge discovery in datasets which takes advantages of both the algorithm-based approach and the visualization-based approach. An interactive visualization model, RuleViz, was proposed, which specifies the architecture model to attain this objective; RuleViz divides the knowledge discovery process into five components based on the common understanding of the KDD process:

- Data preparation and visualization: preparing the raw data and presenting the data in a visual form for the user to easily observe the data distribution;
- Data reduction: vertically and/or horizontally reducing the size of the data set (that is, feature selection and/or tuple selection) for knowledge discovery when the raw data set is too large for the user or algorithm to search efficiently for useful patterns;

- Data preprocessing: transforming, analyzing, merging data tuples from the raw form to a new form such that the data can be easily and precisely processed. Most common data preprocesses include data transformation, missing values handling, continuous attributes discretization, etc.;
- Pattern learning: discovering the useful patterns (associations, classifications, decision trees, etc.) from the data set; and
- Pattern visualization: presenting the discovered patterns in visual forms.

The RuleViz model attempts to combine data mining and machine learning approaches with visualization techniques to provide effective human-machine interactions by visualizing the entire KDD process so that the user's perception and domain knowledge can be integrated into the KDD process on-demand. By means of interaction, the KDD system can effectively navigate through the enormous search spaces and carry out the intentions of the user, while visualizing the KDD process helps users to gain better insight into multidimensional data, to understand the intermediate results, and to interpret the discovered patterns.

There are many decisions that must be made when implementing an interactive KDD system based on the RuleViz model, such as choosing an appropriate visualization space (2D, 3D, or even more dimensions), data and knowledge visualization techniques (iconic, pixel-oriented, geometric projection, hierarchical, or graph-based), interaction function distribution, implementation paradigms, windowing strategies, etc.

Three implementation paradigms of the RuleViz model were discussed in detail and are summarized as follows, and it was also pointed out that combinations of these paradigms are possible.

- *Image-based algorithmic approach* addresses the use of image processing techniques. The raw data is visualized into an image, and each component of the system is performed by means of operations on the image. The data and pattern images are used in the place of the data and patterns that the KDD process must accommodate.

- *Embedded-algorithm-based approach* embeds state-of-the-art algorithms for each component into the system, and the components, however, are connected by interaction and visualization. The objectives of interaction and visualization in such a system depend on the user to interactively choose appropriate algorithms and provide parameters for the algorithms in each component and to conveniently observe the data distribution so as to make decisions for next step, respectively.

- *User-supervised interactive approach* implements a fully interactive KDD system, in which the user plays an active role. The KDD process is completely "supervised" by the user, which means the user not only selects the dataset and sets parameter values like a typical interactive KDD system does, but also operates each component directly without the algorithm running. The actions that the machine performs are to visualize the current states (data and patterns) and store the results that the user directs.

Four interactive knowledge discovery systems with visualization were implemented for experimenting and evaluating the RuleViz model, including

- AViz – an image-based system for discovering numerical association rules based on data plots and optimized rectangles;

- CViz – an embedded-algorithm-based system for classification rule induction based on the *parallel coordinates* visualization technique;
- CVizT – a user-supervised interactive system for building classification rules based on the *Table Lens* visualization technique; and
- DTViz – a user-supervised interactive system for constructing decision trees based on the *parallel segments* pixel-oriented visualization technique and the tree structure visualization algorithm.

There are certainly many different functions that can be included in each component when an interactive KDD system is developed. In our implementation, however, the functions that were considered mainly concentrated on the following aspects:

- raw data visualization techniques;
- feature and tuple selection for data reduction;
- missing values handling and continuous attribute discretization for data preprocessing;
- pattern discovery algorithms and interactive pattern construction; and
- pattern (final pattern and intermediate result) visualization techniques.

Experiments were performed on these four systems using the artificial data set, census data, and several UCI repository data sets, including IRIS, Monk, Glass, Income, etc. Our experimental results demonstrate that the RuleViz model can provide a methodology for developing interactive KDD systems and the systems developed according to the RuleViz model take advantage of both algorithm-based and visualization-based approaches. They provide the following functionalities:

- provide the user with straightforward observation and control of the KDD process;
- integrate the user's perception and domain knowledge into the KDD process easily;
- make it convenient for the user to understand and interpret the discovered knowledge; and
- remain flexible for distributing the KDD functions between the user and machine.

These merits are what we have been pursuing in designing the RuleViz model and represent accomplishments of this thesis.

10.2 Future Work

The RuleViz model proposes a framework for developing interactive KDD systems, and the four implemented systems, AViz, CViz, CVizT, and DTViz, explore specific visualization techniques and learning strategies for specific learning tasks. Considerable research remains to be done in application of the RuleViz model. We would like to persue the following venues in our future work:

- More experiments with the implemented systems will be completed. We realized that our experiments presented in this thesis are not sufficient to make the persuasive conclusion. We intend to train more people with different backgrounds to operate the systems and construct the classifiers, and compare the results to achieve the improvement of the results. We only used *resubstitution*

evaluatation method to evaluate the predictive accuracy of the classifiers in our experiments. Other existing evaluation methods such as *resample evaluation* and *multi-fold evaluation* will be used to evaluate the experiment results. In addition, more exact evaluation methods should be developed. Our experiments are currently focus on the data sets from the UCI repository [126]. The systems implemented should be applied to practical data sets.

- More effective visualization techniques will be developed. Most existing data and knowledge visualization techniques, including those that are used in the AViz, CViz, CVizT, and DTViz systems, focus on the visual representation of data and knowledge, but lack the flexible and friendly interface for the user to operate the visualization so as to integrate the domain knowledge and user's perception into the system [29]. The visual discretization of continuous attributes developed in the CVizT and DTViz systems is our first attempt for developing an interactive data operation in the KDD process by means of effective visualization techniques. The other possibilities include effective interactive data reduction (tuple selection and feature selection), data clustering or pre-classification (usually requiring appropriate measure of distances between data points), missing value estimation (for example, the user can replace the missing values with the potential values by viewing the data or feature histograms), etc.

- More effective interaction interfaces for the user to manipulate the data will be researched. In the RuleViz model, the interaction between the user and machine is most important. By means of visualization, the user can gain insight into the data (feature) distribution. However, observing the data distribution is not our end. What is pursued is to operate the data according

to the observations. For example, through data visualization, one may find an interesting data area in the visualized graphics. S/he wants to use the subset of data that fall in this area and clean all data points outside the area. In such a case, the effective interactive tools for picking the interesting area is desired. In the AViz system, we implemented a "rubber band" to pick the interesting data rectangle on the two dimensional space. This "rubber band" has an inconvenient limitation that constrains the interesting data area to be rectangular. For most realistic applications this limitation is not the case. The "rubber band" can be expanded in two directions: from data rectangles in 2D spaces to a data cube in 3D spaces; from data rectangles with only four vertical and horizontal lines to data polygons with any number of straight lines or arcs or a mixture of both.

- Human perceptions and perceptual inferences will be explored for designing visualization techniques. Usually, visual representations of data create problems obvious for humans to solve by means of perceptions. How a user perceives the data (e.g., data distribution, feature association, or other data characteristics) through the data visualization and visualized images dominates the visualization techniques. Moreover, human perception research also helps to design the interaction tools for the user to operate the data and intermediate results. Human perception is a vast and frequently studied subject. The connection between perception and cognitive activities, however, has been tenuous, making external cognition difficult to study with any precision [32]. Perception research can be based on either the logical map of the eye or the logical three-level hierarchical organization of the visual perception system [140].

- More types of patterns will be considered to be the target patterns of new interactive KDD systems. The RuleViz model is not limited to discovery of (2D and 3D) associations (AViz), classification rule induction and construction (CViz, CVizT), and construction of decision trees (DTViz). New interactive KDD systems will be developed for discovering other types of patterns such as neural networks, data/document clusters, sequential patterns, and so forth. The AViz systems can be extended to visualize multidimensional tuples and discover multidimensional numerical association rules. An interactive visual mining system from transaction databases is being considered, where a visual representation of the layer structure of frequent itemsets can be characterized by the "lattice" structure of itemsets that are visualized by use of iconic and graph-based visualization techniques, and the "lattice" structure can be built in either bottom-up or top-down fashion. The neural network has its own intrinsic structure which can be used as the basis of visualizing data and patterns. Web and text mining systems with visualization are also under consideration. Each type of pattern usually requires distinct data and knowledge visualization techniques, interaction means, discovery algorithms, and implementation strategies.

- Separate KDD systems will be integrated together to form an interactive KDD environment. Some approaches for tuple selection, feature selection, discretization of continuous attributes, missing values handling, etc. can be developed for general purpose, not limited to specific learning patterns, and can be shared by many KDD systems. For example, both the AViz and CViz systems provide the *bin-packing-based equi-depth* discretization method of continuous attributes. In our future integrated system, they can share a common routine for this discretization method.

- The effective cooperation [14] of the user and the machine in the KDD system will be investigated and developed. Embedded-algorithm-based approaches for implementing the KDD system includes learning algorithms and data processing algorithms in the system as relatively independent components. What the user needs to do is to provide parameters and input data for the embedded algorithms. In the systems implemented in this paradigm, the user is passive while the machine is active. The embedded algorithms are not easily changed so that the generated systems are not flexible and have more limitations to include the user's perception and domain knowledge. User-supervised interactive approaches, in contrast, only provide visual data and/or intermediate results to the user and the user must make decisions in each step of the KDD process. In the systems implemented in such a paradigm, the user is active while the machine is passive, and the user's perception and domain knowledge can be flexibly integrated into the KDD process. The flexibility of the systems built in this way, however, may lead to a situation in which the user does not know how to perform the next step or has no idea how to make a decision. For example, during the construction of a decision tree using the DTViz system, after the user chooses a node to split, s/he must first choose the split attribute and then choose the split points for the attribute. Although strategies for selecting split attributes and split points were discussed in Chapter 8, the user may still find it difficult to make a decision to select the split attribute and/or the split points for the selected attribute, if the data and class labels distribute evenly or uniformly and no clear clusters exist in all attribute columns. To overcome this problem, user-supervised and embedded-algorithm-based approaches can be integrated. A possible combination of these two paradigms is to build an effective cooperation between

the user and the machine. In most cases, the user supervises the KDD process. In the cases in which the user does not know how to perform a specific task, embedded algorithms are called upon to compute the task and rank the possible solutions in terms of some criteria, and then the user makes the final decision to choose one solution from the many possible solutions.

Bibliography

[1] P. Adriaans and D. Zantinge, Data Mining, *Addison-Wesley*, 1996.

[2] C. C. Aggarwal, P. S. Yu, A New Framework For Itemset Generation, *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp.18-24, 1998.

[3] R. Agrawal, T. Imielinski, A. Swami, Mining Association Rules Between Sets of Items in Large Databases, *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 207-216, 1993.

[4] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo, Fast Discovery of Association Rules, in *Advances in Knowledge Discovery and Data Mining* ed. by U. M. Fayyard, G. Piatetsky-Shapiro, P. Smyth, and R. Uthrusamy, AAAI Press, Menlo Park, CA, pp.307-328, 1996.

[5] R. Agrawal, M. Mehta, J. C. Shafer, SPRINT: A Scalable Parallel Classifier for Data Mining", in *Proc. of VLDB-96*, pp. 544-555, 1996.

[6] R. Agrawal, R. Srikant, Fast Algorithm for Mining Association Rules in Large Databases, *Proc. of VLDB-94*, pp.487-499, 1994.

[7] R. Agrawal, J. C. Shafer, Parallel Mining of Association Rules, *IEEE Transactions on Knowledge and Data Engineering* 8(6), pp.962-969, 1996.

[8] C. Ahlberg, Spotfire: An Information Exploration Environment, *ACM SIGMOD Record* 25(4), pp.25-29, 1996.

[9] K. Ali, S. Manganaris, R. Srikant, Partial Classification Using Association Rules, *Proc. of KDD-97*, pp.115-118, 1997.

[10] K. Alsabti, S. Ranka, V. Singh, CLOUDS: A Decision Tree Classifiers for Large Datasets, *Proc. of KDD-98*, pp. 2-8, 1998.

[11] A. An and N. Cercone, ELEM2: A Learning System for More Accurate Classifications, *Proc. of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 426-441, 1998.

[12] A. An and N. Cercone, Discretization of Continuous Attributes for Learning Classification Rules, *Proc. of the 3rd Pacific-Asia Conference on Knowledge Discovery in Databases*, pp. 509-514, 1999.

[13] M. Ankerst, C. Elsen, M. Ester, and H. P. Kriegel, Visual Classification: An Interactive Approach to Decision Tree Construction, *Proc. of KDD-99*, pp. 392-397, 1999.

[14] M. Ankerst, M. Ester, and H. P. Kriegel, Towards an Effective Cooperation of the User and the Computer for Classification, *Proc. of KDD-2000*, pp.179-188, 2000.

[15] M. Ankerst, D. A. Keim, H. P. Kriegel, Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets, *Proc. of Visualization*, pp.254-260, 1996.

[16] D. Barbara, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross, and K. C. Sevcik, The New Jersey Data Reduction Report, *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pp. 3-45, 1997.

[17] R. A. Becker, S. Eick, and A. R. Wilks, Visualising Network Data, *Trans. on Visualisation and Computer Graphics* 1(1), pp. 16-28, 1995.

[18] J. Beddow, Shape Coding of Multidimensional Data on a Microcomputer Display, *Proc. Visualisation'90*, pp. 238-246, 1990.

[19] M. Ben-Bassat, Pattern Recognition and Reduction of Dimensionality, *Handbook of Statistics-II*, eds. P. R. Krishnaiah and L. N. Kanal, pp. 773-791, North Holland, 1982.

[20] S. Berchtold, H. V. Jagadish, and K. A. Ross, Independence Diagrams: A Technique for Visual Data Mining, *Proc. of KDD-98*, pp.139-143, 1998.

[21] J. Bertin, Graphics and Graphic Information Processing, *Berlin: De Gruyter*, 1981.

[22] C. M. Bishop, Neural Networks for Pattern Recogination, *Oxford University Press*, 1996.

[23] A. Blum and T. Mitchell, Combining Labeled and Unlabeled Data with Co-Training, *Proc. of the 11th Annual Conf. on Computational Learning Theory*, pp. 92-100, ACM press, New York, 1998

[24] R. J. Brachman and T. Anand, The Process of Knowledge Discovery in Databases: A Human-Centered Approach, in *Advances in Knowledge Dis-*

covery and Data Mining, eds. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI Press / The MIT Press, pp. 37-58, 1996.

[25] L. Breiman, Pasting Bites together for Prediction in Large Data Sets and On-line, Technical report, Statistics Department, University of California, Berkeley, CA, 1997.

[26] L. Breimann, J. H. Friedmann, R. A. Olshen and C. J. Stone, Classification and Regression Trees, *Wadsworth International Group*, 1984.

[27] S. Brin, R. Motwani and C. Silverstein, Beyond Market Baskets: Generalizing Association Rules to Correlations, *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 265-276, 1997.

[28] S. Brin, R. Motwani, J. D. Ullman and S. Tsur Dynamic Itemset Counting and Implication Rules for Market Basket Data, *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 255-264, 1997

[29] K. W. Brodlie, J. D. Wood, and H. Wright Scientific Visualization: Some Novel Approaches to Learning, *Proc. of ACM SIGCSE/SIGCUE Conference*, 1996

[30] C. Brunk, J. Kelly and R. Kohavi, MineSet: An Integrated System for Data Mining, *Proc. of KDD-97*, pp. 135-138, 1997.

[31] Y. Cai, N. Cercone and J. Han, Attribute Oriented Induction in Relational Databases, in *Knowledge Discovery in Databases* ed. by Gregory Piatetsky-Shapiro and William J. Frawley, pp.213-228, 1991.

[32] S. K. Card, J. D. Mackinlay and B. Shneiderman, Readings in Information Visualization: Using Vision to Think, *Morgan Kaufmann Publishrs*, Inc. 1999.

[33] V. Castelli and T. M. Cover, Classification Rules in the Unknown Mixture Parameter Case: Relative Value of Labeled and Unlabeled Examples, *Proc. 1994 IEEE Int. Symp. Information Theory*, pp. 111, Trondheim, Norway, 1994.

[34] V. Castelli and T. M. Cover, On the Exponential Value of Labeled Samples, *Pattern Recognition Letters* 16, pp. 105-111, 1995.

[35] N. Cercone and H. Hamilton, Database Mining, in *Encyclopedia of Electrical and Electronics Engineering* 4, *Wiley Interscience*, pp.576-604, 1999.

[36] Y. Chauvin and D. Rumelhart, Backpropagation: Theory, achitectures, and applications, *Lawrence Erlbaum Associates*, 1995.

[37] J. X. Chen, D. Rine and H. D. Simon, Advancing Interactive Visualization and Computational Steering, *IEEE Computational Science and Engineering*, Winter 1996.

[38] M. S. Chen, J. Han, and P. S. Yu, Data Mining: An Overview from Database Perspective, *IEEE Transactions on Knowledge and Data Engineering* 8(6), pp. 866-883, 1996.

[39] D. W. Cheung, L. Wang, S. M. Yiu and B. Zhou, Density-based Mining of Quantitative Association Rules, *Proc. of PAKDD-2000*, pp. 257-268, 2000.

[40] J. Y. Ching, A. K. C. Wong and K. C. C. Chan, Class-dependent Discretization for Inductive Learning from Continuous and Mixed-mode Data, *IEEE Trans. PAMI* (17), pp. 641-651, 1995.

[41] D. K. Chiu and A. K. C. Wong, Synthesizing Knowledge: A Cluster Analysis Approach Using Event-covering, *IEEE Trans. SMC* (16), pp. 251-259, 1986.

[42] K. Cios, W. Pedrycz and R. Swiniarski, Data Mining Methods for Knowledge Discovery, *Kluwer Academic Publishers*, 1998.

[43] P. Clark and T. Niblett, The CN2 Induction Algorithm, *Machine Learning (3)*, pp. 261-283, 1989.

[44] J. V. Curtis and J. A. Konstan, Interactive Visualization of Periodic Data, *Proc. of the ACM Symp. of User Interface Software and Technology, Visualization*, pp.29-38, 1998.

[45] A. P. Dempster, N. M. Laird and D. B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society* B, pp.1-38, 1977.

[46] M. Derthick, J. Kolojejchick and S. F. Roth, An Interactive Visualization Environment for Data Exploration, *Proc. of KDD-97*, pp.2-9, 1997.

[47] G. Dong and J. Li, Interestingness of Discovered Association Rules in terms of Neighborhood-based Unexpectedness, *Proc. of the PAKDD*, pp. 72-86, 1998.

[48] R. O. Duda and P. E. Hart, Pattern Recognition and Scene Analysis, *Wiley*, 1973.

[49] J. F. Elder IV and D. Pregibon, A Statistical Perspective on Knowledge Discovery in Databases, in *Advances in Knowledge Discovery and Data Mining*, eds. U. M. Fayyard, G. Piatetsky-Shapiro, P. Smyth, and R. Uthrusamy, AAAI Press, Menlo Park, CA, pp.83-116, 1996.

[50] V. Fabian and J. Hannan, Introduction to Probability and Mathematical Statistics, *John Wiley & Sons*, 1985.

[51] U. M. Fayyad and K. B. Irani, On the Handling of Continuous-valued Attributes for Classificaton Learning, *Machine Learning* (8), pp. 87-102, 1992.

[52] U. M. Fayyad and K. B. Irani, Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. *Proc. of IJCAI-93.* pp. 1022-1027, 1993.

[53] U. M. Fayyad, G. Piatetsky-Shapiro and P. Smyth, From Data Mining to Knowledge Discovery: An Overview, in *Advances in Knowledge Discovery and Data Mining*, eds. U. M. Fayyard, G. Piatetsky-Shapiro, P. Smyth, and R. Uthrusamy, AAAI Press / The Mit Press, Menlo Park, CA, pp. 1-36, 1996.

[54] S. Feiner and C. Beshers, Worlds within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds, *Proc. of User Interface Software and Technology*, pp. 76-83, 1990.

[55] K. Fishkin and M. C. Stone, Enchanced Dynamic Queries via Movable Filters, *Proc. of CHI'95, ACM Conf. on Human Factors in Computer Systems*, pp. 415-420, 1995.

[56] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, Knowledge Discovery in Databases: An Overview, in *Knowledge Discovery in Databases*, eds. G. Piatetsky-Shapiro and B. Frawley, pp.1-27, 1991.

[57] J. A. Freeman and D. M. Skapura, Neural networks, *Readings, MA: Addision Wesley*, 1991.

[58] A. A. Freitas, On Objective Measures of Rule Surprisingness, *Proc og the PKDD*, pp. 1-9, 1998.

[59] B. J. Frey, Graphical Models for Machine Learning and Digital Communication, *The MIT Press*, 1998.

[60] T. Fukuda, Y. Morimoto, S. Morishita and T. Tokuyama, Mining Optimized Association Rules for Numeric Attributes, *Proc. of the 15th ACM Symposium on Principles of Database System*, pp.182-191, 1996.

[61] T. Fukuda, Y. Morimoto, S. Morishita and T. Tokuyama, Data Mining Using Two-Dimensional Optimized Association Rules: Scheme, Algorithms, and Visualization, *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 13-24, 1996.

[62] K. Fukunaga and D. L. Kessell, Nonparametric Bayes Error Estimation Using Unclassified Samples, *IEEE Trans. on Infor. Theory*, IT-19(4), pp.434-440, 1973.

[63] K. Fukunaga, J. M. Mantock, Nonparametric Data Reduction, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No.1, pp.115-118, 1984.

[64] G. W. Furnas, Effective View Navigation, *Proc. of CHI, ACM Conference on Human Factors in Computing Systems*, pp. 367-374, 1997.

[65] P. Gago and C. Bentos, A Metric for Selection of the Most Promising Rules, *Proc. of the European Conference on the Principles of Data Mining and Knowledge Discovery*, pp. 19-27, 1998.

[66] J. Gehrke, V. Ganti, R. Ramakrishnan and W. Loh, BOAT–Optimistic Decision Tree Construction, *Proc. of the SIGMOD*, pp.169-180, 1999.

[67] Z. Ghahramani and M. Jordan, Supervised Learning from Incomplete Data via an EM Approach, In *Advances in Neural Information Processing Systems (NIPS)* 6, pp.120-127, 1994.

[68] A. D. Gordon, Classification, *Chapman and Hall*, 1981.

[69] B. Gray and M. E. Orlowska, Ccaiia: Clustering Categorical Attributes into Interesting Association Rules, *Proc of the PAKDD*, pp. 132-143, 1998.

[70] R. Groth, Data Mining: A Hands-on Approach for Business Professionals, *Prentice Hall PTR*, 1998.

[71] J. W. Grzymala-Busse, On the Unknown Attribute values in Learning from Examples, *Proc. of the International Symposium on Methodologies for Intelligent Systems*, pp. 16-19, 1991.

[72] H. Hagen, M. Miller and G. M. Nielson Focus on Scientific Visualization, *Springer-Verlag*, 1993.

[73] J. Han, Y. Cai and N. Cercone, Knowledge Discovery in Databases: An attribute-oriented approach, *Proc. of VLDB-92*, pp.335-350, 1992.

[74] J. Han, Using Table Lens to Interactively Build Classifiers, Applied Mathematics Letters, Elsevier Science Ltd., Pergamon, to appear.

[75] J. Han, A. An, and N. Cercone, CViz: A Visualization System for Rule Induction, *Proc. of the 13th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 214-226, Montreal, Canada, May 2000.

[76] J. Han and N. Cercone, DVIZ: A System for Visualizing Data Mining, *Proc. of the 3rd Pacific-Asia Knowledge Discovery in Databases*, pp. 390-399, Beijing, China, April 1999.

[77] J. Han and N. Cercone, AViz: A Visualization System for Discovering Numerical Association Rules, *Proc. of the 4th Pacific-Asia Conference on Knowledge Discovery in Databases*, pp. 269-280, Kyoto, Japan, April 2000.

[78] J. Han and N. Cercone, Typical Example Selection for Rule Induction, *Proc. of the 13th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 347-356, Montreal, Canada, May 2000.

[79] J. Han and N. Cercone, Visualizing the Process of Knowledge Discovery, *Journal of Electronic Imaging*, SPIE 9(4), pp. 404-420, 2000.

[80] J. Han and N. Cercone, RuleViz: A Model for Visualizing Knowledge Discovery Process, *Proc. of KDD-2000*, pp.223-242, Boston, USA, August 2000.

[81] J. Han, N. Cercone and X. Hu, An Interactive Visualization System for Mining Association Rules, to appear, 2000.

[82] J. Han, X. Hu and N. Cercone, Supervised Learning: A Generalized Rough Set Approach, *Proc. of RSCTC-2000*, Banff, Canada, October 2000.

[83] J. Han and Y. Fu, Discovery of multiple-level association rules, *Proc. of the 21th International Conference on VLDB*, pp.420-431, 1995.

[84] J. Han and Y. Fu, Attribute-Oriented Induction in Data Mining, in *Advances in Knowledge Discovery and Data Mining* ed. by U. M. Fayyard, G. Piatetsky-Shapiro, P. Smyth, and R. Uthrusamy, AAAI Press, Menlo Park, CA, pp.399-421, 1996.

[85] D. J. Hand, Construction and Assessment of Classification Rules, *John Wiley & Sons*, 1997.

[86] R. J. Hilderman, Mining Summaries From Databases Using Domain Generalization Graphs and Objective Measures of Interestingness, *PhD Thesis*, Department of Computer Science, University of Regina, Regina, SK, Canada.

[87] H. Hofmann, A. Siebes, and A. Wilhelm, Visualizing Association Rules with Interactive Mosaic Plots, *Proc. of KDD-2000*, pp. 227-235, Boston, USA, August 2000.

[88] M. Houtsma and A. Swami, Set-oriented Mining of Association Rules, *Proc. of the International Conference on Data Engineering*, pp.25-34, 1995.

[89] X. Hu and N. Cercone, Data Mining via Discretization, Generalization and Rough Set Feature Selection, *Knowledge and Information Systems* 1(1), pp. 33-60, 1999.

[90] L. Hyafil and R. L. Rivest, Constructing Optimal Binary Decision Trees is NP-complete, *Information Processing Letters*, 5(1), pp.15-17, 1976.

[91] A. Inselberg and B. Dimsdale, Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry, *Proc of IEEE Visualization Conference*, pp.361-370, 1990

[92] A. K. Jain and J. Mao, Artificial Neural Networks: A Tutorial, *Computer*, pp. 31-44, March 1996.

[93] M. James, Classification Algorithms, *William Collins Sons & Co. Ltd*, 1985.

[94] N. B. Karayiannis and A. N. Venetsanopoulos, Artificial Neural Networks, Learning Algorithms, Performance Evaluation, and Applications, *Kluwer Academic Publishers*, 1993

[95] D. A. Keim, Databases and Visualisation: Tutorial, *Proc. of ACM SIGMOD Internal Conf. on Management of Data*, 1996.

[96] D. A. Keim, Visual Data Mining, *Tutorial Notes*, *Proc. of VLDB*, 1997.

[97] D. A. Keim and H. P. Kriegel, VisDB: Database Exploration Using Multidimensional Visualization, *IEEE Computer Graphics and Applications*, pp. 40-49, 1994.

[98] D. A. Keim and H. P. Kriegel, Visualization Techniques for Mining Large Databases: A Comparison, *Transaction on Knowledge and Data Engineering*, 8(6), pp.923-938, 1996.

[99] J. B. Kennedy, K. J. Mitchell and P. J. Barchay, A Framework for Information Visualization, *SIGMORD Record* 25(4), pp. 30-34, 1996.

[100] K. Kira and L. A. Rendell, The Feature Selection Problem: Traditional Methods and a New Algorithm, *Proc. of ICML-92*, pp. 129-134, 1992.

[101] R. Kohai and D. Sommerfield, Targeting Business Users with Decision Table Classifiers, *Proc. of KDD-98*, pp. 249-253, 1998.

[102] I. Kononenko, Estimating Attributes: Analysis and Extensions of RELIEF, *Proc. of ECML-94*, pp.171-182, 1994.

[103] M. Kryszkiewicz, Association Rules in Incomplete Databases, in *Proc. 3rd Pacific Asia Conf. on KDD*, pp.84-93, 1999.

[104] J. Lamping and R. Rao, The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies, *Journal of Visual Languages and Computing*, 7(1), pp.33-35, 1996.

[105] R. P. Lippmann, Pattern Classification Using Neural Networks, *IEEE Communications Magazine* 27, pp.47-64, 1989.

[106] B. Liu, W. Hsu, and Y. Ma, Integrating Classification and Association Rule Mining, *Proc. of KDD-98*, pp.80-86, 1998.

[107] H. Liu, H. Lu, L.Feng, and F. Hussain, Efficient Search of Reliable Exceptions, *Proc. of the PAKDD*, pp. 194-203, 1999.

[108] H. Liu and H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, *Kluwer Academic Publishers*, 1998.

[109] H. Liu and H. Motoda, Feature Extraction, Construction and Selection: A Data Mining Perspective, *Kluwer Academic Publishers*, 1998.

[110] H. Lu, R. Setiono, and H. Liu, Neurorule: A Connectionist Approach to Data Mining, *Proc. of VLDB-95*, pp.478-489, 1995

[111] H. Mannila, H. Toivonen, and A. I. Verkamo, Efficient Algorithms for Discovering Association Rules, *Proc. of the AAAI Workshop on Knowledge Discovery in Databases*, pp.144-155, July 1994.

[112] J. A. McDonald, Painting Multiple Views of Complex Objects, *Proc. of ECOOP/OOPLSLA'90*, pp. 245-257, 1990.

[113] N. Megiddo and R. Srikant, Discovering Predicative Association Rules, *Proc of KDD-98*, pp.274-278, 1998.

[114] M. Mehta, J. Rissanen, and R. Agrawal, MDL-based Decision Tree Pruning, *Proc. of KDD-95*, pp. 216-221, 1995.

[115] R. S. Michalski, Pattern Recognition as Rule-guided Induction Inference, *IEEE Transaction on Pattern Analysis and Machine Intelligence* 2, pp.349-361, 1980.

[116] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains, *Proc. of AAAI-86*, pp. 1041-1045, 1986.

[117] R. J. Miller and Y. Yang, Association Rules over Interval Data, *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 452-461, 1997.

[118] D. J. Miller and H. S. Uyar, A Mixture of Experts Classifier with Learning Based on both Labeled and Unlabeled Data, In *Advances in Neural Information Processing Systems (NIPS)* 9, pp.571-577, 1997.

[119] D. J. Miller and H. S. Uyar, A Generalized Gaussian Mixture Classifier with Learning Based on both Labeled and Unlabeled Data, *Proc. of Conf. on Info. Sci. and Sys.*, 1996.

[120] MineSet, *http://www.sgi.com/software/mineset/.*

[121] J. Mingers, An Empirical Comparision of Selection Measures for Decision Tree Induction, *Machine Learning* 3(4), pp. 319-342, 1989.

[122] J. Mingers, An Empirical Comparison of Pruning Methods for Decision Tree Induction, *Machine Learning* 4(2), pp. 227-243, 1990.

[123] T. M. Mitchell, Machine Learning, *The McGraw-Hill Co. Inc.*, 1997.

[124] D. Mitchie, D. J. Spiegelhalter and C. C. Taylor, Machine Learning, Neural and Statistical Classification, *Ellis Horwood Ltd.*, 1994.

[125] W. Muller and F. Wysotzki, The Decision Tree Algorithm CAL5 Based on a Statistical Approach to Its Splitting Algorithm, in *Machine Learning and Statistics: The Interface*, eds. G. Nakhaeizadeh, C. C. Taylor, *John Wiley & Sons, Inc.*, pp.45-65, 1997.

[126] P. M. Murphy and D. W. Aho, UCI Repository of Machine Learning Databases, URL: *http://www.ics.uci.edu/~mlearn/MLRepository.html*, 1996.

[127] G. Nakhaeizadeh and C. C. Taylor, Machine Learning and Statistics: The Interface, *John Wiley & Sons, Inc.*, 1997.

[128] T. D. Nguyen, T. B. Ho, and H. Shimodaira, Interactive Visualization in Mining Large Decision Trees, *Proc. of PAKDD-2000*, pp. 345-348, 2000.

[129] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, Learning to Classify Text from Labeled and Unlabeled Documents, *Proc. of AAAI-98*, pp.792-799, AAAI press/MIT press, 1998

[130] Y.-H. Pao and D. J. Sobajic, Combined Use of Supervised and Unsupervised Learning for Dynamic Security Assessment, *IEEE Trans. Power Systems* 7(2), pp.878-884, 1992.

[131] J. S. Park, M. S. Chen, and P. S. Yu, An Effective Hash Based Algorithm for Mining Association Rules, *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp.175-186, May 1995.

[132] J. S. Park, P. S. Yu, and M. S. Chen, Mining Association Rules with Ad-

justable Accuracy, *Proc. of the 6th Internal. Conf. on Info. and Knowledge Management, CIKM'97*, pp.151-160, Nov. 1997.

[133] G. Piatetsky-Shapiro, Discovery, Analysis, and Presentation of Strong Rules, in *Knowledge Discovery in Databases* ed. by Gregory Piatetsky-Shapiro and William J. Frawley, pp.229-260, 1991.

[134] P. Pirolli and R. Rao, Table Lens as a Tool for Making Sense of Data, *Proc. of AVI, Workshop on Advanced Visual Interfaces*, pp. 67-80, 1996.

[135] J. R. Quinlan, Induction of Decision Trees, in *Readings in Machine Learning*, eds. by J. W. Shavlik and T. G. Dietterich, *Morgan Kaufmann Publishers*, pp.57-69, 1990.

[136] J. R. Quinlan, C4.5: Programs for Machine Learning, *Morgan Kaufmann Publishers*, 1993.

[137] J. S. Rao and W. J. E. Potts, Visualizing Bagged Decision Trees, *Proc. of KDD-97*, pp. 243-246, 1997.

[138] R. Rao and S. K. Card, The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information, *Proc. of ACM Conference on Human Factors in Computing Systems*, 318-322, New York, NY, 1994.

[139] J. Ratsaby and S. S. Venkatesh, Learning from a Mixture of Labeled and Unlabeled Examples with Parametric Side Information, *Proc. of the 8th Annual Conf. on Computational Learning Theory*, pp.412-417, 1995.

[140] H. L. Resnikoff, The Illusion of Reality, *Springer-Verlag*, New York, NY, 1987.

[141] G. G. Robertson, S. K. Card, and J. D. Mackinlay, Information Visualization Using 3D Interactive Animation, *Communications of the ACM*, 36(4), pp.57-71, 1993.

[142] D. Rumelhart, B. Widrow and M. Lehr, The Basic Ideas in Neural Networks, *Communications of the ACM* 37(3), pp. 87-92, 1994.

[143] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, *Prentice Hall*, 1995.

[144] A. Savasere, E. Omiecinski, and S. Navathe, An Efficient Algorithm for Mining Association Rules in Large Databases, *Proc. of VLDB-95*, pp.432-444, 1995.

[145] B. Shahshahani and D. Landgrebe, The Effect of Unlabeled Samples in Reducing the Small Size Problem and Mitigating the Hughes Phenomenon, *IEEE Trans. on Geoscience and Remoting Sensing* 32(5), pp.1087-1095, 1994.

[146] B. Shahshahani and D. Landgrebe, Using Partially Labeled Data for Normal Mixture Identification with Application to Class Definition, *Proc. IEEE Internal. Geosci. Remote Sensing Symp.*, pp.1603-1605, 1992.

[147] B. Shneiderman, Tree Visualisation with Treemaps: A 2D Space-Filling Approach, *ACM Trans. on Graphics* 11(1), pp. 92-99, 1992.

[148] B. Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization, *Proc. of IEEE Workshop on Visual Languages*, pp. 336-343, 1996.

[149] A. Silberschatz and A. Tuzhilin, On Subjective Measure of Interestingness in Knowledge Discovery, *Proc. of KDD-95*, pp. 275-281, 1995.

[150] http://www.spss.com.

[151] R. Srikant and R. Agrawal, Mining Generalized Association Rules, *Proc. of VLDB-95*, pp.407-419, 1995.

[152] R. Srikant and R. Agrawal, Mining Quantitative Association Rules in Large Relational Tables, *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp.1-12, 1996.

[153] R. Srikant, Q. Vu, and R. Agrawal, Mining Association Rules with Item Constraints, *Proc. of KDD-97*, pp.67-73, 1997.

[154] E. Suzuki, Autonomous Discovery of Reliable Exception Rules, *Proc. of KDD-97*, pp. 259-262, 1997.

[155] S. K. Thompson, Sampling, John Wiley & Sons, Inc, 1992.

[156] H. Toivonen, Sampling Large Databases for Finding Association Rules, *Proc. of VLDB-96*, pp.134-145, 1996.

[157] E. R. Tufte, Visual Explanations: Images and Quantities, Evidence and Narrative, *Cheshire, CT: Graphics Press*, 1997.

[158] L. Tweedie, Characterizing Interactive Externalizations, *Proc. of CHI'97, ACM Conf. on Human Factors in Computing Systems*, pp. 375-382, 1997.

[159] P. E. Utgoff, N. C. Berkman, and J. A. Clouse, Decision Tree Induction Based on Efficient Tree Restructuring, *Machine Learning* 29, p.5-46, 1997.

[160] G. I. Webb, A Heuristic Covering Algorithm Has Higher Predictive Accuracy Than Learning All Rules, Proc. of the Conference on Information, *Statistics and Induction in Science*, pp.20-30, 1996.

[161] J. A. Webb, Adapt: Global Image Processing with the Split and Merge Model, *Carnegie Mellon University*, Pittsburgh, PA., 1991.

[162] A. P. White and W. Z. Liu, Statistical Properties of Tree-based Approaches to Classification, in *Machine Learning and Statistics: The Interface*, eds. G. Nakhaeizadeh, C. C. Taylor, *John Wiley & Sons, Inc.*, pp.23-44, 1997.

[163] N. Zhong, Y. Yao, and S. Ohsuga, Peculiarity-oriented Multi-database Mining, *Proc. of the PKDD*, pp. 136-146, 1999.