Visual Data Mining: Framework and Algorithm Development

M. Ganesh, Eui-Hong (Sam) Han[†], Vipin Kumar, Shashi Shekhar, Jaideep Srivastava[*]
Dept. of Computer & Information Sciences
4-192 EECS Bldg., 200 Union St. SE
University of Minnesota
Minneapolis, MN 55455, USA

Technical Report 96-021 (March 12, 1996)

Abstract

Visual data mining is the use of visualization techniques to allow data miners and analysts to evaluate, monitor, and guide the inputs, products and process of data mining. It can help introduce user insights, preferences, and biases in the earlier stages of the data mining life-cycle to reduce its overall computation complexity and reduce the set of uninteresting patterns in the product. Even more useful may be the new insights developed by the data miners and analysts concerning the quality and implications of the decisions made by the data mining process. These new insights may facilitate the development of better algorithms and processes for data mining. This paper provides a framework for visual data mining via the loose-coupling of databases and visualization systems. The paper applies visual data mining towards designing new algorithms that can learn decision trees by manually refining some of the decisions made well–known algorithms such as $C4.5$. Experiments with a set of benchmark data sets from the literature show that a new algorithm consistently outperforms the well–known algorithms. The paper concludes with a discussion of the implications of visual data mining for the data mining, databases, and visualization systems.

Keywords: Data mining, visualization, classification, decision trees, algorithm development.

[†]**Contact author:** Eui-Hong (Sam) Han
Tel: (612) 626–7515
Fax: (612) 624–6539
email: han@cs.umn.edu

[*] The authors are listed in alphabetical order and no seniority is implied.

1 Introduction

Data mining [SAD+93] is the efficient and possibly unsupervised discovery of interesting, useful and previously unknown patterns in a data warehouse, which is a historical database designed to facilitate analysis and knowledge discovery. Common patterns of interest include classification [Qui86], associations [AS94, HS95], clustering [Fis95], and sequential patterns [MTV95]. The process is often very slow, particularly when databases are large. The success of the data mining process is critically dependent upon the availability of user insights and biases[Mit77], even though the process may use unsupervised learning algorithms [Lan96]. User insights and biases includes abstract preferences for attributes and attribute sets that reflect the user's interests and purposes. It could also include more detailed guidance in terms of preferences on the partial patterns being selected for further growth and exploration during the mining process.

In this paper, we propose Visual Data Mining as a mechanism to enable users to monitor the data mining process, inputs, and products, as well as to interact with the data mining process and possibly influence the decisions being made. Visual Data Mining is useful for several reasons. First, early user feedback about level of interest, usefulness, and domain knowledge can filter out uninteresting patterns early in the process and help focus computing resources on more interesting patterns. Second, user feedback can improve the selection of appropriate learning algorithms based on the application domain. Third, visual inspection of datasets can provide direct clues towards interesting patterns in the data [KKS94]. Most importantly, visualization of the data mining process can provide new insights and may result in the design of better algorithms for all aspects of data mining.

1.1 Data Mining Life Cycle

We view the life cycle of the data mining operation as a three stage process of preparing the data for mining, deriving the model, and using the knowledge obtained from the data, as shown in Figure 1.

The data preparation stage deals with improving the data quality and summarizing the data to facilitate the analysis and discovery process. Data mining can be done on either operational databases or on a data warehouse [IH94], which is usually a summary database of the various businesses of an enterprise. The quality of the data in the data warehouse is constantly monitored by data analysts. Due to the heterogeneity and non-standard policies enforced on data quality at the different source databases, the warehouse data is usually cleaned or standardized via data scrubbing [Inm92, IH94].

The model derivation stage focuses on choosing learning samples, testing samples and learning algorithms. Due to the large volume of available data, data mining may be done

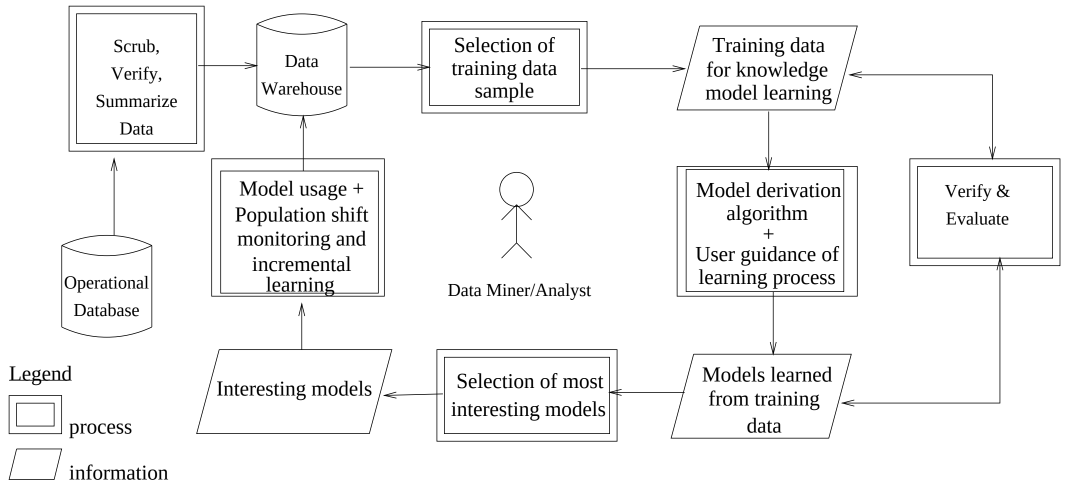

Figure 1: Information flow in the data mining life cycle

on subsets of the data from the data warehouse. An appropriate data sample is selected from the data in the warehouse and is checked for representativeness. This process may have to iterate a few times before a suitable sample set can be selected. The selected sample dataset forms the training data for the data-mining algorithm. The data-mining process is viewed in our framework as the derivation of an appropriate knowledge model of the patterns in the data that are interesting to the user. The algorithm for model derivation, together with the guidance provided by the user, will generally produce several models of the information contained in the data. The data-mining algorithms use guidance from the analyst to decide various parameters of the model being learned from the data, such as its accuracy and prevalence, and to control the computational complexity of the learning process.

Among all the models generated, the users may only select a few interesting models to be included in their applications. The usage and maintenance phase is concerned with monitoring of database updates and continued validation of patterns learned in the past. Even though the learning process may have used guidance, not all the knowledge models generated will have business applications. Only the interesting models are selected and applied performing business tasks. Another important task in this stage of the life cycle is to continuously monitor the validity of the knowledge models in the context of changes to data in the warehouse. When the population in the warehouse shifts significantly, the previously learned models will no longer be applicable, and new models will have to be derived. We may also be able to learn new models incrementally from the new data.

1.2 Scope and Outline

Visual Data Mining (VDM) refers to the use of visualization techniques to evaluate, monitor and guide the data mining process and its inputs and products, as needed. Evaluation includes the validation of training samples, test-samples, and learned models against the data in the database plus the appropriateness of data and learning algorithms for specific data-mining situations. Monitoring includes activities such as tracking the progress of the data-mining algorithms, evaluating the continued relevance of learned patterns in the context of database updates, etc. Guidance includes activities such as user-initiated biasing or altering inputs, learned patterns and other system decisions.

We find the prospect of improving the data-mining algorithms through the new insights developed about the mining process, to be the most exciting promise of visual data mining. This paper proposes a loose-coupling of existing conceptual data models and visual interfaces as a tool for supporting visual data mining. It illustrates the tool in the context of the problem of learning decision trees for classification problems, e.g., visualizing the progress of ID3-like [Qui86] algorithms. It presents and evaluates two new algorithms designed interactively via the tool.

The rest of the paper is organized as follows. In section 2 we provide brief surveys of visualization and data mining and summarize our contributions. In section 3 we propose a language which integrates query-language features and visual interface controls. We provide some examples to show the application of this extended visual query language to various data-mining algorithms. Section 4 shows the step-by-step processes involved in developing algorithms for data mining using visualization, and we develop new algorithms for inducing decision trees from examples. A performance evaluation of the algorithms developed in section 4 is done in section 5. Section 6 presents our conclusions and future work.

2 Related Work and Our Contributions

The goal of visual data mining is to provide a synthesis of visualization and data mining, to enhance the effectiveness of the latter using the techniques from the former. Since this synthesis is rather new, there is very little work which covers both aspects and is thus directly relevant. Since visual data mining draws upon techniques from both data mining and visualization, we believe there is a need to refer to work in these areas. However, since both are established fields in themselves, we provide only a brief reference to the techniques in them, and point to comprehensive surveys in these areas. To do more than this, we believe, is outside the scope of the present paper. In the following, we first briefly survey techniques in data mining and visualization and then evaluate the potential of visualization techniques for enhancing the effectiveness of data mining.

2.1 Data Mining: A Brief Survey

Data mining is an active research area with the promise of high payoffs in many business and scientific domains [FPSM91, MCPS93, AIS93, AP95]. Industry, government, and scientific communities are being inundated with huge volumes of data that is routinely stored in online databases. Analyzing this data and extracting meaningful patterns from it is almost impossible without computer assistance to the human analyst. Much of the data-mining research to date has focused on approaches to applying traditional machine learning and discovery methods to data stored in relational databases [FPSM91]. Recent years have seen more of a "systems" approach to the process [MCPS93, AP95, HS95], where knowledge discovery and machine-learning techniques are components of an entire data mining system.

Knowledge extracted during the mining process can be of various kinds, depending on the nature of the knowledge, its complexity, its uses, etc. Examples include classification rules, association rules, temporal sequences, causal graphs, etc. [AIS93]. In addition, this categorization helps in developing efficient algorithms for the specific categories of knowledge. The focus of traditional machine-learning research [Car90] was on the accuracy of the learned knowledge. Given the huge volume of data in databases and the computational complexity of the learning process, the scope of the research program has been expanded to address efficiency in addition to accuracy. In recent years substantial advances have been made in developing data-mining techniques that provide high efficiency in addition to improved accuracy.

Since a detailed survey of data mining research is beyond the scope of the present paper, we point readers to the following references [MCPS93, AIS93, AP95]. From the perspective of visual data mining, any of these data-mining techniques can be used as the engine in the "model derivation algorithm" component of our system.

2.2 Visualization Techniques: A Brief Survey

Visualization has been an active research area for some time now, and it has shown its power in helping us better understand scientific phenomena [REE$^+$94, Kau94a]. Any visualization system focuses on the following four issues: (i) the conceptual model of visualization data, (ii) the control interface, (iii) the control methods, and (iv) the visualization representation. A detailed survey of these is beyond the scope of this paper and can be found in [NS90, REE$^+$94, com94]. In the following, we provide a brief overview of each of these.

Conceptual Data Model: Almost all work in visualization has used some variation of a model where (point) datum is represented as a vector with values for each component/dimension. Given the roots of visualization in graphics, this is very natural. While often adequate for representing information about physical phenomena such as convective and laminar flows, vortex formation, etc., this approach may not be appropriate for representing the complex relationships found in business data[Mac86].

The VisDB [KK94] system demonstrates one example of conceptual data modeling of complex business data. The VisDB system uses each pixel of the screen to visualize the data items resulting from a query on whole data. The resulting data items are sorted according to their relevance to the query. The VisDB system maps the relevance factors to colors and places the perfectly matched items in the middle of the screen with the less relevant ones creating a rectangular spiral according to their relevance factors. The system also generates a separate window for each selection predicate of the query. The pixel position for each data item is the same in all the windows. The location of each data object represents a unique object identity. The users can perceive data characteristics such as multidimensional clusters or correlations by studying corresponding regions in the different windows.

Control Interface: Control of most visualization systems is provided as a "point-and-click" interface, typically handled by a mouse. A new interface like one using DataGlove has been also proposed [BF94]. The ease of use of PCs over Unix machines has shown the clear superiority of the desktop paradigm over a textually oriented command line paradigm.

Control Methods: Control methods refer to the various techniques that are available to an analyst to control navigation through the data. Specific examples include querying, dimension reduction, aggregation, setting of orientation/perspective/hierarchy, animation, and slider control. Current visualization systems provide some of these. Subspace methods find significant features and subspaces of high-dimensional data sets that can be visualized [Gro94]. Cluster-analysis techniques are used to find an optimal representation of the density distribution of the multidimensional data sets. The problem of finding a low dimensional representation of these clusters that preserves their most important features is solved by an analysis of the principal components. We believe there is a need to go substantially beyond this and include the other control methods mentioned above to visualize the complex nature of relationships extracted from business data.

Visualization Representations: Volume visualization is a method of extracting information from volumetric datasets and is concerned with the representation, manipulation and rendering of these datasets [Kau94b]. The volumetric dataset is usually represented as a 3D discrete regular grid of volume elements called *voxels*. Each voxel has values associated with it which represent properties of the object residing in the voxel. Volume graphics [Kau94b] provides a mechanism for synthesis, manipulation and rendering of volumetric objects stored in a volume buffer of voxels. Volume graphics has several advantages over conventional surface graphics. These are viewpoint independence, insensitivity to object and scene complexity, support for block operations and the capability of representing the inner structure of objects [Kau94b]. The major disadvantage of volume graphics based on the voxel model is that the method is approximate and requires a very large amount of memory. However, octree representations, filtering schemes and transformation schemes have been proposed to reduce the memory requirement of the voxel model by compressing the information [Kau94b].

AutoVisual [BF94] designs visualizations for the n-Vision visualization system that use *worlds within worlds*, an interactive method of representing multivariate relations with a hierarchy of nested heterogeneous coordinate systems (*worlds*). Each world may contain a graph encoding a subset of the relation encoded by its parent world. The subset is determined by the position of the world's origin relative to its parent. The user can grab each world using a DataGlove and move it throughout the space defined by its parent, thereby interactively exploring the data space. Glyphmaker is a tool for data visualization/analysis [RAEM94] that allows users to build customized representations of multivariate data easily and also provides interactive tools to explore the patterns in and relations between the data. Glyphs, which are graphical objects whose elements (e.g. position, size, shape, color, orientation, transparency, etc.) are bound to data, are used to visualize complex data. Glyphmaker provides different kinds of glyphs like lines, spheres, cuboids, cylinders and arrows. It also provides an interface to bind data variables to the several elements of the glyphs. This binding makes data characteristics be displayed through the visual aspects of glyphs.

Visualization Needs of Data Mining have been explored by few researchers. IMACS [BST+94] uses conventional graphs and plots as an interface for the analyst to segment data with mouse clicks. The data segements appear as breaks in a graph to indicate segment boundaries. MVV [MTS91] uses bar charts and slider bars to locate clusters in multidimensional space, allowing the display of multiple views of a given dataset. The nested histograms and the group bars in WinViz [LHLQ95] and Netmap [Dav93] are line-based visualization tools that use a circle as the basic graphical device. Its circumference is divided into several groups, one for each attribute of interest. Individuals are represented by nodes within the group. Lines drawn across circles indicate relationships between subgroups or individuals linking their nodes. Netmap uses clustering algorithms to group individuals and fuzzy matching techniques to identify near matches.

Of the various components of visualization, we believe that the control interface and visualization representations are in the most mature form for use by data mining. Thus, unless a specific shortcoming is identified, we believe there is no need to go beyond what is provided by visualization systems today. For the other two, namely data model and control methods, we believe more work needs to be done. Earlier we described possible enhancements to control methods. We now describe possible enhancements to the data model.

First, we believe that the vector-type data models are too low level to appropriately capture the rich semantics found in most business data. We believe there is a need to use semantically rich data models, e.g. the *entity relationship model* [EN94], to model the concepts. Some initial efforts in this direction include [ABN92, KKS94, LHLQ95]. Next, we need the ability to visualize database concepts like schemas, queries, indices, etc. in addition to the data. This will enhance the understanding of an analyst about the underlying database, thus aiding in the mining process.

2.3 Our Contributions

The first contribution made by this paper is in developing an approach where techniques from data mining and visualization are synthesized to provide user input, preference and bias during earlier phases of data mining. The second contribution made by this paper is in developing a framework where learning algorithms can be developed and refined in an incremental manner, with visual feedback from the analyst. This provides a very powerful method for the analyst to incrementally refine the parameter settings, thus improving the convergence of the learning process.

The final contribution made by this paper is to develop a new class of learning algorithms in the proposed framework, which uses feedback from the user at runtime to refine some of the decisions made by well-known learning algorithms for decision trees. We have designed new algorithms via this mechanism. A new algorithm outperformed the well-known algorithms for several benchmark datasets.

3 A Visual Query Language Based Control Interface

In this section we present a Visual Query-Language-Based Control Interface (VQLBCI), as a framework which enables user participation in the data-mining process. Users are allowed to make queries on and manipulate the visual representations just like data. The query-language-based framework is described in terms of the entity-relation data model. We describe how the common tasks in visual data mining can be enabled in this framework, and we also use the specific example of classification-rule learning to show the utility of the VDM process.

3.1 Basic Concepts

VQLBCI is a tool to visualize the database objects (e.g. the results of queries) using user-defined visual representations and to query and manipulate the visual representations of database objects, via a modern user interface such as the mouse based X window system. The goal of VQLBCI is to integrate the visual representations with conceptual data models and query languages in a loosely coupled and extensible manner. The loose coupling is intended to facilitate the reuse of existing database technology and visualization technology in the implementation. Extensibility is desired in such a system to allow the user to include new diagrams and visual representations, since the visual representation can often be domain-dependent.

One approach to integration is to specify the visual representations using a conceptual data model, just like other data in the database. The behavior of these visual representations in response to the user input, e.g. mouse events, can be modelled as data manipulation queries over the specification of visual representation. We also assume that the result of a query is presented to the user in a visual manner by creating new visual representations or updating existing ones. Several associations between pairs of concepts from the two domains need to be specified. These include the association between query results and their visual representations, as well as the association between external events and implied queries on the visual representations.

This paper uses the Entity-Relationship Diagram (ERD) [EN94], a well-known conceptual data-model, to specify visual representations just like other data in a database. We hypothesize that the Entity-Relation model can represent a reasonable set of diagrams needed for visualizing the inputs, products and process of data mining, even if there are some visual representations that are difficult/impossible to model in this framework. We take the liberty of allowing procedure-valued composite attributes in ERDs to model the procedure-valued fields supported by modern databases. The externally specifiable parameters of the conversion procedures are represented in the ERD as parts of the composite attributes.

Use of ERDs to specify visual models allows users to define different visual representations (e.g. histograms, networks, pie-charts, etc.) in the database schema. A mechanism is needed to associate these user-defined visual representations and the data stored in or derived from the database and data warehouse. This mechanism can be modeled as a relationship between visual representations and other data entities within ERD. Given an association between a query and a visual representation, the results of traditional queries can be converted to visual representations. Default procedures can be provided to convert a database object (e.g. a relational table) to a set of system-defined diagrams along the lines of [Mac86]. The conversion procedures are not natural in traditional ERDs; however, we will take the liberty of viewing these procedures as composite attributes in ERDs. These composite attributes are not too far from modern databases, since they support procedure-valued fields and triggers.

This paper will also use the interaction semantics of common window-system based user interfaces. As new user interfaces emerge based on newer input devices, e.g. gloves, helmets, eye-trackers, and body suits, there may be opportunities for improving the design and assumptions made in this paper. Modern window systems view the display as a hierarchy of windows. The root window represents the entire display area and has child windows such as menus, text windows, icons, scroll-bars, buttons, etc. The type of window determines the response to an event inside it. A button-press event may carry out different actions in a button window and in a text window. The behavior is often modeled as a collection of associations between the external events, e.g. mouse-related ones, and event-handler procedures. Often child windows displaying the result of an event within a parent window are drawn on top of the parent. The window system seldom tries to optimize the size, shape or placement of various windows. These aspects of the windows are often manipulated by the user to improve the layout for the purpose of viewing multiple windows simultaneously. The system also allows the user to interactively redefine some of the visual aspects of a composite window via sliders, menus, radio/check buttons etc. Selection of windows and focus can be made in different ways.

Thus, we assume that the users will start with a hierarchy of windows on their display space. Queries made within a specific window will have their results displayed in the same window or in a child of that window. The visual representation of the results may be determined by the enclosing window as well as the query. For example, queries made inside a text window may display results using textual tables as are used by traditional SQL language. Queries made via a query-by-example window may display results using a spread-sheet interface. Queries could change the default visual representations offered by the enclosing windows via associations which may already be in the database or be specified interactively by the user. For example, an aggregate query using a group-by clause may use a histogram representation. A query fetching only two columns could use a bar chart or plot representation based on the data-types of the selected fields.

3.2 Components of VQLBCI

There are three major concepts in VQLBCI, namely *visual representations, computations* and *events*, as shown in Figure 2 using the conceptual data model of ERD. Visual representations consist of a set of primitive visual representations such icons, geometry etc. Each primitive object has a location with respect to the enclosing visual representation. Visual representations include user-defined and system-defined diagrams and pictures to visualize the inputs, products and process of data mining. Different diagrams already proposed in the literature [GSSW92, LHLQ95, KKS94, KK94, RAEM94] can be thought of as concrete instances of visual representation for data mining. We believe that newer visual representations will be forthcoming based on the needs of the application domains and data mining life-cycles in the future. VQLBCI allows the inclusion of additional user-defined diagrams, as illustrated later in this section.

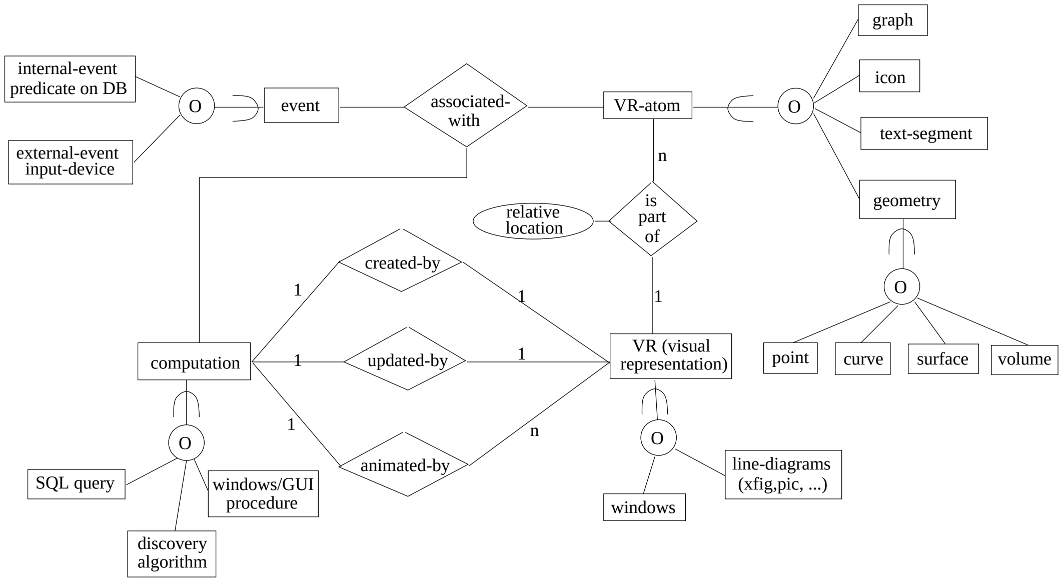

Figure 2: Conceptual data model (ER) for VQLBCI

The computations represent various attributes of database queries, windows-system procedures, and other algorithms. It is not always desirable for the ER model to capture the executable specifications of all computations. However, several attributes of a computation including its name, documentation, parameters, and invocation-commands, may be modeled. Stored procedures in modern databases illustrate the trade-offs inherent in modeling computations in a data model. Events include internal database predicates as well as external events from the input devices such as the mouse and keyboard.

The key relationships in this framework include *associated-with, created-by, updated-by*, and *animated-by.* The <event, visual-representation-atom> pairs are *associated with* computations, which are invoked when the event occurs in the context of a primitive visual representation. The other three relationships are between computations and visual representations. Thus the result of a computation such as a database query can be presented by creating a new diagram, updating an existing diagram, or animating a sequence of diagrams.

Now, we illustrate the modeling of visual representations and primitive visual representation atoms via a couple of examples. We select simple diagrams associated with decision trees and association rules. Figure 3 shows an entropy-gain comparison diagram to help choose among the alternative candidates for extending a frontier node in a decision tree. Each candidate represents the entropy gain by using an attribute to grow the tree. Bar-chart visual representation is used to facilitate visual ranking of the candidates in terms of entropy gain, as shown in the upper left corner of Figure 3. The primitive visual representation atoms are illustrated via an ERD in the figure. The key primitives include data attributes

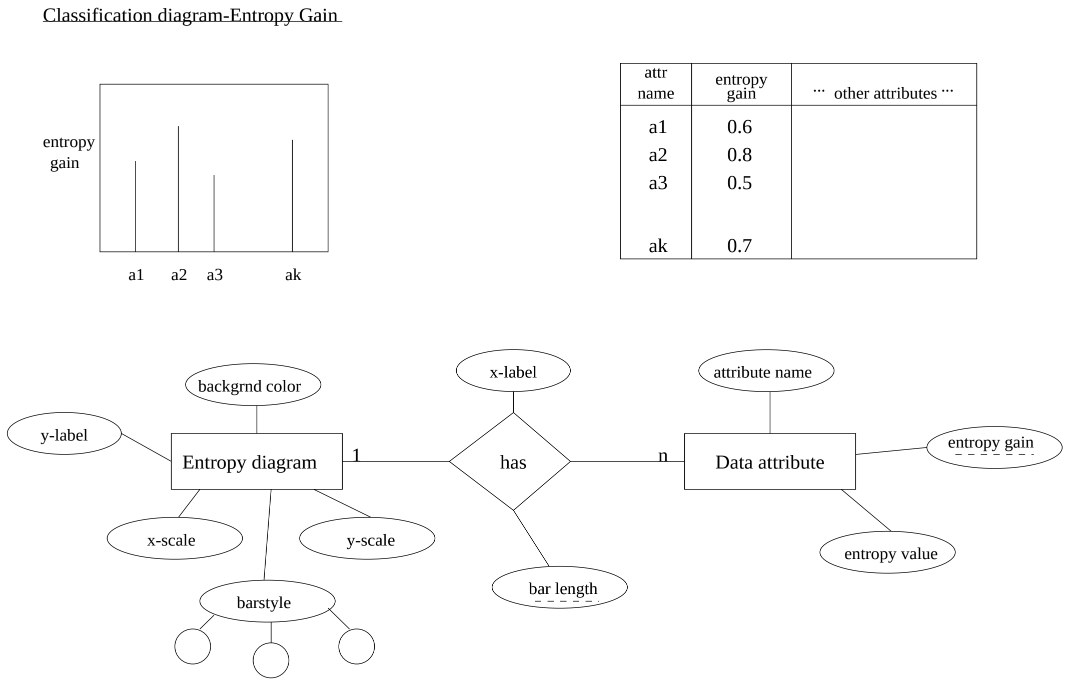

Figure 3: A diagram to compare Entropy Gains

with the visual representation of the bars in the bar diagram. The visual attributes (e.g. bar length) are derived from the entropy gain attributes of the data attributes by using the geometric properties of the diagram.

Figure 4 illustrates a common matrix diagram used to browse the association rules produced by the data mining process. This diagram may be used to select the most interesting rules. As shown in the ERD in figure 4, the primitives include the right-hand-side (RHS) of the association rule with the visual representation of a row in the matrix. The second primitive is the left-hand-side (LHS) of the association rule with the visual representation of a column in the matrix. LHS consists of multiple data attributes, whereas the RHS has a single attribute. The most important primitive is the association rule, with the visual representation of a glyph in the matrix entry determined by its LHS and RHS. The color and size of the glyph may represent the prevalence/support and confidence of the rule.

4 Visual Development of Algorithms

We believe that the most interesting use of visual data mining is the development of new insights and algorithms. The visualization of data mining often provides new insights about the quality and implications of the decisions made by the data mining system. Poor decisions made by the data mining systems can be detected and corrected via a visual query–

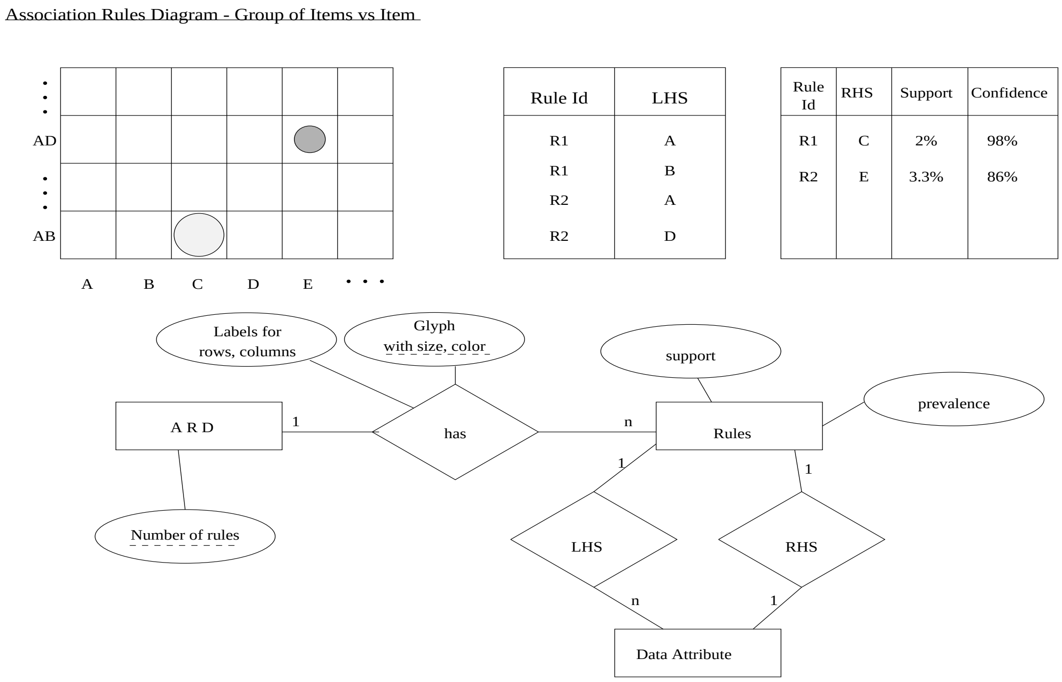

Figure 4: Association Rules Diagram showing two example rules AB→C, and AD→E

language–based control interface.

4.1 General Framework

We have modeled the problem of classification–rule–learning as a search [Kor90] process in the state space of different models. Figure 5 shows the ER diagram for learning decision trees. Such a model allows the user to monitor the quality and impact of decisions made by the learning procedure. The learning procedure can be refined interactively via a visual interface, e.g. a form shown in Figure 5. These extensions are performed by doing queries on the current visual representation of the model. Thus the operations provided as part of the VQLBCI framework allow a user to modify and/or create new procedures for data mining. A visual data mining framework thus also provides a basis for developing new algorithms for data mining. Section 4.2 discusses how some of the interesting algorithms for learning decision trees are developed using this method.

Learning a classification decision tree from a training data set can be regarded as a process of searching for the best decision tree that meets user–provided goal constraints. The problem space of this search process consists of *Model Candidates*, a *Model Candidate Generator*, and *Model Constraints*. Many existing classification–learning algorithms like $C4.5$ [Qui93] and $\mathcal{CDP}$ [AIS93] fit nicely within this search framework. New learning algorithms that fit users' requirements can be developed by defining the components of the problem

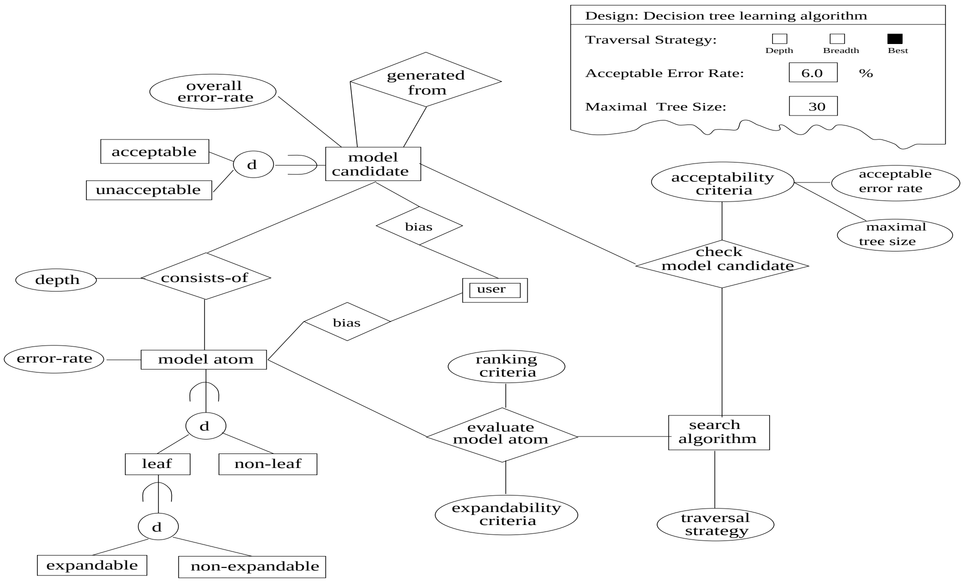

Figure 5: ER diagram for the search space of decision-tree learning algorithms and a form based interface for the manipulation of search parameters

space. We discuss the components of the search problem space in the context of ID3 like decision–tree–learning.

A *Model Candidate* corresponds to a partial classification decision tree. Each node of the decision tree is a *Model Atom.* A non–leaf model atom has an attribute with k associated values and has k child model atoms, each corresponding to a value of the attribute. The arc from a non–leaf model atom to its child model atom is labeled with the corresponding value of the attribute. Leaf model atoms have an attached label that indicates the majority class. Each model atom contains a subset of the training data set, such that the values of the attributes on the path from the root to the model atom are consistent with the data. An initial model candidate corresponds to a decision tree that has a single leaf model atom containing all of the training data set. The search process can be defined as a process to find a final model candidate starting from the initial model candidate, such that the final model candidate meets user goal specifications. Table 1 shows a training data set with four data attributes and two classes. Figure 6 shows different model candidates that are constructed during the search process of finding a final model candidate with the training data set given in Table 1. Figure 6(a) shows the initial model candidate and Figure 6(c) shows the final model candidate. The leftmost leaf model atom of the final model candidate labeled “Play” contains the complete training data set for which the value of *Outlook* is sunny and the value of *Humidity* is less than or equal to 75.

The model candidate generator transforms the current model candidate into a new

Outlook	Temp(F)	Humidity(%)	Windy?	Class
sunny	75	70	true	Play
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play
rain	65	70	true	Don't Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Table 1: A small training data set [Qui93]

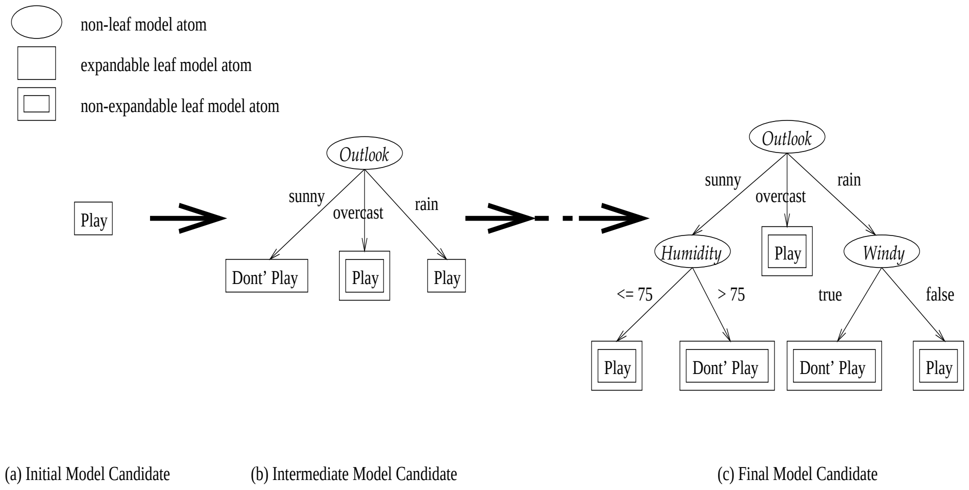

Figure 6: Search process

model candidate by selecting one model atom to expand from the expandable leaf model atoms. The model candidate generator first checks to see whether the current model candidate satisfies the acceptability constraints. If the current model candidate satisfies the acceptability constraints, the search process ends and the current model candidate is returned as the final model candidate. Otherwise, the model candidate generator selects an expandable leaf model atom to expand according to the traversal strategy.

The model candidate generator checks whether the model atom is expandable according to the expandability constraints. If the model atom is not expandable, the model candidate generator marks the model atom as non–expandable. Otherwise, the model candidate generator marks the model atom as non–leaf and then generates child model atoms. The model candidate generator considers all possible attributes that can be used to generate child model atoms. Using a data–entropy–calculation function, the model candidate generator evaluates the entropy gain of the possible child model atoms with different data attributes. One data attribute that maximizes the entropy gain is chosen for the expansion. One model atom is created for each possible value of the attribute chosen, and the training data set is also split accordingly and assigned to each corresponding child model atom. All the generated model atoms are initialized as leaf and expandable. After the expansion, model candidate attributes such as the total number of model atoms and overall error rate must be updated with new model atoms and their attributes.

Model constraints, which are used by the model candidate generator, provide controls and boundaries to the search space. Model constraints consist of acceptability constraints, expandability constraints, a traversal strategy and a data–entropy–calculation function. These constraints are specified based on several attributes of the model candidate and model atom. Each model candidate has the global attributes of the decision tree, the overall error rate of the model candidate, the total number of model atoms in the model candidate, and the total number of expandable leaf model atoms. The overall error rate shows the accuracy of the model candidate for the classification task. The total number of model atoms shows the complexity of the classification rules learned in the model candidate. The total number of expandable leaf model atoms indicates the different possible ways in which the model candidate can be expanded further.

A model atom has the following attributes: leaf or non–leaf, expandable or non–expandable, depth of the node in the tree, and a subset of the training data set that falls into the node of the atom. A model atom also has the derived attributes of the subset, the size of the subset, the class distribution of the subset by each data attribute, the data attribute chosen for the model atom expansion (only for non–leaf model atoms), the entropy gain (only for expandable model atoms), majority class, and local error rate.

An **acceptability constraint predicate** specifies when a model candidate is acceptable and thus allows the search process to stop. The predicate is specified based on the model candidate attributes. Some examples of the acceptability predicate are:

A1) total number of expandable leaf model atoms $= 0$ (default)

A2) overall error rate of the model candidate $\leq$ acceptable error rate

A3) total number of model atoms in the model candidate $\geq$ maximum allowable tree size

A4) other predicates that are logical combinations of the above predicates

The acceptability predicate A1 is used in $C4.5$ and $\mathcal{CDP}$. The predicate A2 can be used to terminate the search process early if the overall error rate is acceptable. The predicate A3 limits the decision tree size with the maximum allowable number of model atoms. Predicate A2 and A3 prevent the search process from expanding fully and force the search process to stop as soon as the user's need is met.

An **expandability constraint predicate** specifies whether a leaf model atom is expandable or not. The predicate is specified based on model atom attributes. Some examples of the expandability predicate are:

E1) local error rate > 0.0 (default)

E2) entropy gain > 0.0 (default)

E3) $(\frac{depth-1}{maxlength})^2 <$ local error rate, where $maxlength$ is user provided constant

E4) $(\frac{adj\_depth-1}{maxlength})^2 <$ local error rate, where $maxlength$ is a user–provided constant and adj_depth is adjusted depth (see Section 4.2 for details)

E5) other predicates on the combination of model atom attributes

$C4.5$ uses the expandability predicate E1 and E2. $\mathcal{CDP}$ uses the predicate E2 and E3.

The **traversal strategy** ranks expandable leaf model atoms based on the model atom attributes. Some of the examples are:

T1) increasing order of depth (Breadth–First)

T2) decreasing order of depth (Depth–First)

T3) orders based on other model atom attributes (Best–First)

$C4.5$ uses Depth–First strategy and $\mathcal{CDP}$ uses Breadth–First strategy.

The **data–entropy–calculation function** provides a mechanism for calculating the entropy of training cases with respect to classes. Some examples of the entropy–calculation function are based on information theory [Qui86], and other user–provided functions such as the Gini function [WK91].

Post–processing modifies the final model candidate generated. It usually involves removing the model atoms from the model candidate. One example of post–processing is $C4.5$ error-based pruning on a decision tree generated from the $C4.5$ algorithm. This pruning technique has been shown to be effective in remedying the problem of over–generalization in decision trees.

Bias is an interactive mechanism for a user to override choices made by the model candidate generator. User bias can override the stopping decision of the model candidate generator or force the model candidate generator to expand a specific model atom. User bias can also override a decision by the model candidate generator by making an expandable model atom non–expandable, or vice versa. For model atom expansion, a user might select a data attribute which is different from the one suggested by the algorithm, based on the data–entropy–calculation function. The child model atoms generated can be merged into one model atom by grouping several discrete data attribute values together, or a single child model atom can be split up into several model atoms by multiple cuts in continuous data attribute values. For example, in Figure 6(b), user might want to merge two values of the *Outlook* attribute, "overcast" and "rain" into one called "not sunny" and then merge two corresponding child model atoms into one child model atom.

4.2 Steps In Visual Algorithm Development

The set of all possible model candidates is too large to allow any classification–learning algorithm to perform an exhaustive search for an optimal model candidate. All of the existing algorithms are greedy in nature, since a decision to expand one model atom by a chosen data attribute is based on local entropy gain and is final. As the result of this greedy nature, it is entirely possible that a different model atom expansion, based on another data attribute of lower entropy gain, leads to a better final model candidate. This fact implies that no single algorithm is the best all the time and that performance is highly data–dependent.

The VQLBCI framework proposed in this paper enables users to find an algorithm that works the best on a given data set. By changing different predicates of model constraints, users can construct a new classification–learning algorithm.

We show how to develop an algorithm, BF, based on the Best–First search idea. For acceptability criteria, in addition to the default predicate which has the total number of expandable leaf model atoms, we include the acceptability predicate 2 with an overall error rate. The predicate 2 allows a user to specify an acceptable error rate for the decision tree. The traversal strategy chosen is Best–First, where the expandable leaf model atoms are

Algorithm	Accept. Predicate	Traversal Strategy	Exp. Predicate	Post–processing
$C4.5$	A1	T1	E1,E2	none
$C4.5P$	A1	T1	E1,E2	error-based pruning
BF	A1,A2	T3	E1,E2	none
$\mathcal{CDP}$	A1	T2	E3	none
$\mathcal{CDP}+$	A1	T2	E4	error-based pruning

Table 2: Comparison of different classification learning algorithms

ranked according to the decreasing order of the number of misclassified training cases. The number of misclassified training cases can be calculated by (local error rate) $*$ (size of the subset training data set). The traversal strategy will expand a model atom that has the most misclassified training cases, thus possibly reducing the overall error rate the most. For the expandability predicate, default predicates on the local error rate and entropy gain are used. For the data–entropy–calculation function, a function based on information theory is chosen for the algorithm.

We can also construct a new algorithm by modifying existing algorithms and combining useful features of different algorithms. We propose a new algorithm $\mathcal{CDP}+$, which is a modification of $\mathcal{CDP}$. $\mathcal{CDP}$ has dynamic pruning using the expandability constraint $(\frac{depth-1}{maxlength})^2 <$ local error rate. We modify the depth in the predicate according to the size of the training data set of the model atom. We want our algorithm to generate a deeper subtree at the model atom if the size of the training data set at the model atom is larger than expected. We set $adj\_depth = -log_b(\frac{t}{T})$, where b is the estimated branching factor of the decision tree, t is the size of training data set belonging to the model atom, and T is the size of the whole training data set. We also adopt the error–based pruning of $C4.5$ in the post–processing of the $\mathcal{CDP}+$ algorithm.

Table 2 compares different learning algorithms based on 3 different criteria of model constraints and post–processing. Data–entropy–calculation function is omitted, since all the algorithms use a function based on information theory.

5 Experimental Evaluation of Visually Designed Algorithms

To validate the usefulness of our methodology of algorithm development using visualization we carried out experiments that compare the new algorithms BF and $\mathcal{CDP}+$, developed in our framework, to other well–known classification algorithms. In this section we describe our experimental methodology and the results obtained from the experiments.

Attribute	Description	Value
salary	salary	uniformly distributed from 20000 to 150000
commission	commission	salary $\geq$ 75000 $\Rightarrow$ commission = 0 else
		uniformly distributed from 10000 to 75000
age	age	uniformly distributed from 20 to 80
elevel	education level	uniformly chosen from 0 to 4
car	make of the car	uniformly chosen from 1 to 20
zipcode	zip code of the town	uniformly chosen from 9 available zipcodes
hvalue	value of the house	uniformly distributed from 0.5k100000 to 1.5k100000
		where $k \in \{0 \cdots 9\}$ depends on the **zipcode**
hyears	years house owned	uniformly distributed from 1 to 30
loan	total loan amount	uniformly distributed from 0 to 500000

Table 3: Description of Attributes

5.1 Experiment Design

Candidate Algorithms: We have chosen to compare the new BF and $\mathcal{CDP}+$ algorithms with the $C4.5$ and the $\mathcal{CDP}$ algorithms. The $C4.5$ algorithm generates a decision tree for the given training data set by recursive partitioning of the data. The algorithm considers all the possible tests that can split the data set and selects a test that gives the best information gain. The algorithm generates a large decision tree which can either fully classify the entire training data set, or which will stop without further partitions along branches which do not give any information gain. The over–fitted tree is then pruned, using a predicted error criterion, by replacing some subtrees with leaves or some intermediate nodes with some subtrees. For comparison purposes, we show the results with pruning ($C4.5P$) and without pruning ($C4.5$).

$\mathcal{CDP}$ has been developed as part of the Quest project [AIS93] at the IBM Almaden Research Center. This algorithm has been shown to have comparable accuracy to the $C4.5$ algorithm, while generating a relatively smaller number of nodes. The $\mathcal{CDP}$ algorithm uses a dynamic pruning criterion to stop the generation of decision nodes if the error rate at a node of the tree is below some adaptive precision threshold.

Benchmark Data: We have followed the experimental methodology described in [AIS93] very closely. Our data set consists of records with the attributes described in table 3. Attributes **elevel, car** and **zipcode** are discrete attributes, while all the other attributes are continuous. The data records are divided into classes based on classification functions of varying complexity. These functions are explained in detail in the Appendix.

Dataset Generation: Data records are generated by randomly generating values for the different attributes using independent random streams of numbers. The data sets generated are divided into training sets and test sets. The records in both the training set as well as the test set are then assigned class labels using the selected classification function.

To model noise in the data we also have a noise factor which is an input to the data–generation program. After assigning classes to the data records, we perturb the continuous attribute values based on the noise factor. Given a perturbation factor of p, the value of a continuous attribute is changed to lie within $v \pm \frac{pr}{2}$, where v is the original value of the attribute, and r is the range of values the attribute can take. In our experiments we have set the noise factor at 5%.

Following the strategy in [AIS93], we have selected training set sizes of 2500 tuples and test set sizes of 10000 tuples. We have generated 10 sets of data for the experiment. We used the same set of 10 functions described in [AIS93] for our experiments.

Performance Metrics: We have selected the following metrics to compare the efficiency, accuracy and size of final decision trees of the classification algorithms. The generation efficiency of the nodes is measured in terms of the total number of nodes generated. The size of a final decision tree is measured in terms of the number of the nodes in the final tree. To compare the accuracy of the various algorithms we have measured the mean classification error on the test data sets.

5.2 An Experiment and Its Results

Our BF algorithm can use two different model candidate attributes, overall error rate and total number of model atoms, for the acceptability constraint predicate. For this experiment, we chose a 6% overall error rate as the acceptable error rate. We do not have a copy of the $\mathcal{CDP}$ algorithm, and thus we simulated the algorithm based on the description in [AIS93]. $\mathcal{CDP}$ uses a parameter $maxlength$ for the adaptive precision threshold. We have chosen 10 for the parameter, which is the choice in [AIS93]. We also used the same value for $\mathcal{CDP}+$. The experiments are repeated 10 times to obtain confidence intervals of 95 percent for the mean error values.

The experimental results of various classification algorithms are shown in Figure 7, Figure 8, and Figure 9. Figure 7 compares the classification errors of the different algorithms on the y-axis for each of the 10 classification functions plotted on the x-axis. Figure 8 compares the generation efficiency of the different algorithms. On the y-axis, the number of nodes generated are shown for each classification function. Figure 9 compares the size of the final decision trees of the different algorithms. On the y-axis, the number of nodes in the final decision trees is shown for each classification function. The final tree size for all algorithms except $C4.5P$ is the same as the number of nodes generated.

As noted in [AIS93], the result shows that $\mathcal{CDP}$ has accuracy comparable to $C4.5$, while

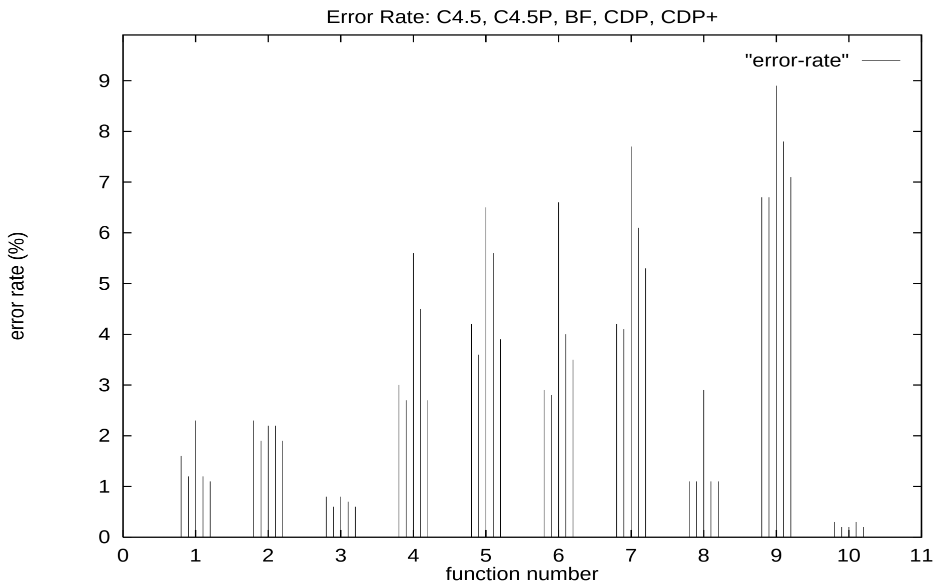

Figure 7: Comparison of classification error for 10 data-sets. Five bars are shown for each problem. They represent C4.5, C4.5P, BF, CDP and CDP+ from left to right.

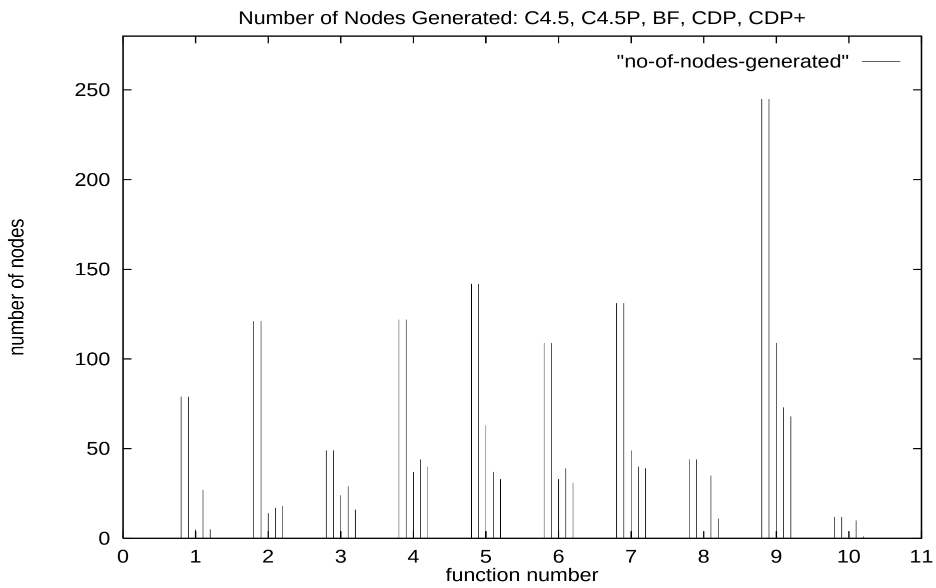

Figure 8: Comparison of nodes generated for 10 data-sets. Five bars are shown for each problem. They represent C4.5, C4.5P, BF, CDP and CDP+ from left to right.

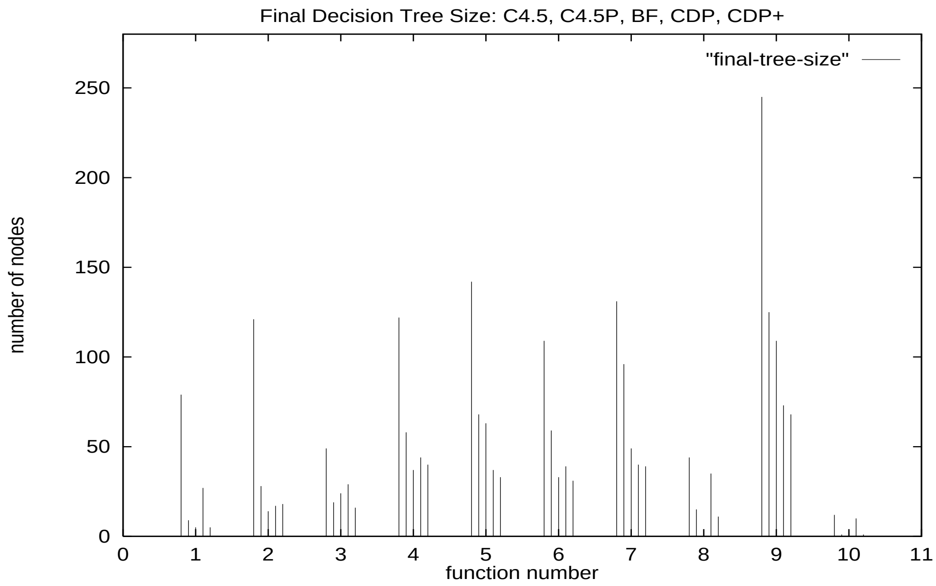

Figure 9: Comparison of final decision tree size for 10 data-sets. Five bars are shown for each problem. They represent C4.5, C4.5P, BF, CDP and CDP+ from left to right.

generating considerably fewer nodes. It can be seen that the error rate of $\mathcal{CDP}$ is higher than that of $C4.5P$ in most of the functions. BF produced decision trees that approximately met the user's acceptable error rate (6%) on the testing data set for most functions except function 7 and 9. In general, BF with a 6% acceptable error rate generated a smaller number of nodes than $\mathcal{CDP}$, but it has a higher error rate. We set a 6% acceptable error rate for all 10 functions in this experiment. However, as we adjusted the error rate for individual functions, we were able to produce decision trees with a lower error rate that is comparable to $\mathcal{CDP}$. $\mathcal{CDP}+$ consistently outperformed $\mathcal{CDP}$ in error rate and number of nodes generated. It can be also seen that the accuracy of $\mathcal{CDP}+$ is comparable to $C4.5P$ while generating a considerably smaller number of nodes than $C4.5P$. Even compared to the final tree size of $C4.5P$ trees, $\mathcal{CDP}+$ generated fewer nodes in all the functions.

The experimental result shows that $\mathcal{CDP}+$ is the best overall algorithm for the 10 functions and the data set we generated, when we consider all the performance metrics together. However, if you consider classification accuracy alone, $C4.5P$ is the winner. For function 8, even though BF has a tree times greater error rate than the other algorithms, it produced decision trees with 10 to 40 times fewer nodes. Depending on user requirements, this decision tree may be preferable over those found by other algorithms due to its exceptionally small size. Different data sets require different algorithms for the best results, and diverse user requirements put different constraints on the final decision tree. The experiment shows that our interactive visual data mining framework can help finding the most suitable algorithm for a given data set and group of user requirements.

6 Conclusions and Future Work

Visual data mining shows great promise. It can help bring in user insights, preferences and biases to the earlier stages of the data-mining life-cycle to reduce overall computation complexity and reduce the set of uninteresting patterns in the product. Even more useful may be the new insights developed by the data miners and analysts regarding the quality and implications of the decisions made by the data-mining process. These new insights may facilitate the development of better algorithms and processes for data mining.

We have proposed a conceptual data-model-based framework for visual data mining in terms of a visual query-language-based control interface (VQLBCI). This reuses common concepts from query languages such as the SQL which are widely used in the database community and can be related closely to the data mining context. The query language allows users to do powerful selections, subsetting, summarizations and updates to the visual representations. These operations are frequently required in a visual data-mining system. In addition, the visual representations have event-computation associations. Users can launch the pre-associated computations (e.g. queries) via a simple click of a mouse-button or another input device. The query results and other database objects have associated user-defined visual representations and hence, results of queries can be visually presented to the user. Implementation of VQLBCI reuses existing database engines as well as existing window systems (e.g. X-windows). In addition, VQLBCI manages the mapping between the database engines and window systems by managing the associations such as <visual representation, database objects> and <visual representation atoms, event, computations>. It also allows the definition of new visual representations for domain-specific diagrams and pictures.

We provide a generic search-based interface with VQLBCI to visualize the quality and nature of decisions made by decision tree learning algorithms such as $C4.5$ [Qui93]. We have developed two new algorithms, namely BF and $\mathcal{CDP}+$, and evaluated their performance with 10 datasets from the literature. BF uses best-first ordering to expand the frontier nodes of the decision trees instead of the conventional depth-first or breadth-first criteria. $\mathcal{CDP}+$ uses error-rate and prevalence-based pruning criteria instead of the $\mathcal{CDP}$ -pruning criteria based on error-rate and depth. Experimental results show that BF has a new capability to guide the construction of decision trees based on the overall error-rate criteria. In addition, $\mathcal{CDP}+$ often outperforms several decision-tree learning algorithms in error rate and number of nodes generated. This illustrates the promise of VQLBCI in developing new insights towards improving the data-mining processes and algorithms.

Future work includes further evaluation and refinement of VQLBCI. The evaluation of visual algorithm development using VQLBCI needs more experimentation with additional data-sets. We plan to use VQLBCI as an algorithm animation tool in our research, and to further explore interactive construction of decision trees by allowing users to override and refine the decisions made by the data-mining process at run-time.

VQLBCI may be extended in the future to support other conceptual data models such as multi-dimensional space, graphs/networks, etc. Performance issues in implementation need a lot of exploration. Even reused components such as databases and window systems may need refinement to better support visual data mining. For example, current window systems cannot handle billions of objects for interactive exploration. However, many databases contain billions of objects, and interactive data mining may occasionally require browsing of such large data-sets. Window systems may need to use hierarchical management of such large datasets by viewing visual representations as indices rather than as collections of actual objects.

We would like to explore the parallel implementation [KGGK94, KSA94] of search algorithms, which is more complex than sequential algorithms. We will use the VQLBCI framework as the front-end interface and will provide users with different ways to run data mining algorithms: in parallel or distributed over available computing resources. We would also like to explore whether a search framework can help in developing new insights and algorithms for discovering other patterns such as association rules, clusterings and sequences. Another area of interest is in exploring specific data mining application domains such as database integration [GSR96], semantic query optimization [SHKC93] and generalization [AS92].

7 Acknowledgments

We would like to thank Dr. Arun Swami from Silicon Graphics Inc. and Dr. Bamshad Mobasher for their valuable contributions in the various stages of the project. We would also like to thank the data mining group at the Department of Computer Science for its numerous contributions, whose comments improved the technical content, readability and presentation. The members include Rajat Agarwal, Mark Coyle, Brajesh Goyal, Raja Harinath, Namit Jain, Mohamed Qadir, Siva Ravada, and Anurag Srivastava. This work was supported by the Department of the Army, the Federal Highway Authority, and the Minnesota Dept. of Transportation. This work (project effort) is sponsored (in whole or in part) by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, ARO Grant DA/DAAH04-95-1-0538, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

References

[ABN92] T. M. Anwar, H. W. Beck, and S. B. Navathe. Knowledge mining by imprecise querying: A classification-based approach. In *Proc. of the 8th Int'l Conf. on Data Engg.*, pages 622–630, 1992.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engg.*, 5(6):914–925, December 1993.

[AP95] R. Argawal and G. Psaila. Active data mining. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 3–8, Montreal, Quebec, 1995.

[AS92] M. B. Amin and S. Shekhar. Generalization by neural networks. *Proc. of the 8th Int'l Conf. on Data Engg.*, April 1992.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th VLDB Conference*, pages 487–499, Santiago, Chile, 1994.

[BF94] C.G. Beshers and S.K. Feiner. Automated design of data visualizations. In L. Rosenblum, R.A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors, *Scientific Visualization*, pages 86–102. Academic Press Inc., San Diego, CA, 1994.

[BST+94] R. J. Brachman, P. G. Selfridge, L. G. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D. L. McGuiness, and L. A. Resnick. Integrated support for data archaeology. In *Proc. of AAAI-94 Knowledge Discovery in Databases Workshop*, Seattle, WA, 1994.

[Car90] J. G. Carbonell. Introduction: Paradigms for machine learning. In J. G. Carbonell, editor, *Machine Learning: Paradigms and Methods*, pages 1–9. MIT Press, 1990.

[com94] Special issue on visualization. IEEE Computer, July 1994.

[Dav93] C. Davidson. What your database hides away. *New Scientist*, pages 28–31, 9 January 1993.

[EN94] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Benjamin/Cummings, Redwood City, CA, second edition, 1994.

[Fis95] D. Fisher. Optimization and simplification of hierarchical clusterings. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 118–123, Montreal, Quebec, 1995.

[FPSM91] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. The AAAI Press, Menlo Park, CA, 1991.

[Gro94] M.H. Gross. Subspace methods for the visualization of multidimensional data sets. In L. Rosenblum, R.A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors, *Scientific Visualization*, pages 171–186. Academic Press Inc., San Diego, CA, 1994.

[GSR96] M. Ganesh, J. Srivastava, and T. Richardson. Rule learning for instance-level database integration. Technical Report TR-96-??, Department of Computer Science, University of Minnesota, Minneapolis, 1996.

[GSSW92] G. Grinstein, J. Seig, S. Smith, and M. Williams. Visualization for knowledge discovery. *Int'l Journal of Intelligent Systems*, 7:637–648, 1992.

[HS95] M. A. W. Houtsma and A. N. Swami. Set-oriented mining for association rules in relational databases. In *Proc. of the 11th Int'l Conf. on Data Engg.*, pages 25–33, Taipei, Taiwan, 1995.

[IH94] W. H. Inmon and R. D. Hackathorn. *Using the Data Warehouse.* Wiley-QED, 1994.

[Inm92] W. H. Inmon. *Building the Data Warehouse.* Wiley-QED, 1992.

[Kau94a] A. Kaufman. Guest editor's introduction: Visualization. *IEEE Computer*, 27(7):18–19, July 1994.

[Kau94b] A. Kaufman. Trends in volume visualization and volume graphics. In L. Rosenblum, R.A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors, *Scientific Visualization*, pages 3–19. Academic Press Inc., San Diego, CA, 1994.

[KGGK94] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to Parallel Computing: Algorithm Design and Analysis.* Benjamin Cummings/ Addison Wesley, Redwod City, 1994.

[KK94] D. A. Keim and H.-P. Kriegel. VisDB: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, pages 40–49, September 1994.

[KKS94] D. A. Keim, H.-P. Kriegel, and T. Seidl. Supporting data mining of large databases by visual feedback queries. In *Proc. of the 10th Int'l Conf. on Data Engg.*, pages 302–313, Houston, TX, 1994.

[Kor90] R. E. Korf. Search. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 994–998. John Wiley & Sons, Inc., 1990.

[KSA94] V. Kumar, S. Shekhar, and M. B. Amin. A scalable parallel formulation of backpropagation algorithm for hypercubes and related architectures. *IEEE Transactions on Parallel and Distributed Systems*, 5(10):1073–1090, October 1994.

[Lan96] P. Langley. *Elements of Machine Learning.* Morgan-Kaufman, San Francisco, CA, 1996.

[LHLQ95] H.-Y. Lee, H.-L.Ong, and L.-H. Quek. Exploiting visualization in knowledge discovery. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 198–203, Montreal, Quebec, 1995.

[Mac86] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, April 1986.

[MCPS93] C. J. Matheus, P. K. Chan, and G. Piatetsky-Shapiro. Systems for knowledge discovery in databases. *IEEE Transactions on Knowledge and Data Engg.*, 5(6):903–913, December 1993.

[Mit77] T. M. Mitchell. Version spaces: A candidate elimination approach to rule learning. In *Proc. of IJCAI 77*, 1977.

[MTS91] T. Mihalisin, J. Timlim, and J. Schwegler. Visualization and analysis of multi-variate data: A technique for all fields. In *Proc. of Visualization 91*, 1991.

[MTV95] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 210–215, Montreal, Quebec, 1995.

[NS90] G. M. Nielson and B. Shriver, editors. *Visualization in Scientific Computing.* IEEE Computer Society Press, Los Alamitos, CA, 1990.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[RAEM94] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: Creating customized visualizations of complex data. *IEEE Computer*, 27(7):57–64, July 1994.

[REE+94] L. Rosenblum, R.A. Earnshaw, J. Encarnacao, H. Hagen, A. Kaufman, S. Klimenko, G. Nielson, F. Post, and D. Thalmann, editors. *Scientific Visualization.* Academic Press Inc., 1994.

[SAD+93] M. Stonebraker, R. Agrawal, U. Dayal, E. J. Neuhold, and A. Reuter. Dbms research at a crossroads: The vienna update. In *Proc. of the 19th VLDB Conference*, pages 688–692, Dublin, Ireland, 1993.

[SHKC93] S. Shekhar, B. Hamidzadeh, A. Kohli, and M. Coyle. Learning transformation rules for semantic query optimization: A data-driven approach. *IEEE Transactions on Knowledge and Data Engg.*, 5(6):950–964, December 1993.

[WK91] S. M. Weiss and C. A. Kulikowski. *Computer Systems That Learn.* Morgan-Kaufman, San Mateo, CA, 1991.

Appendix

A: Common VDM tasks in the VQLBCI framework

Browsing or observation is a very frequent activity required in a data mining operation. Very often the user may need to look at the distribution of data along various parameter axes to obtain visual clues to the hidden patterns. [KKS94] suggests some methods for the visual representation of data for this purpose. Visual representations can also help in browsing the learned models of knowledge from the data. For example, in the case of classification rules the user can visually inspect various nodes in a decision tree learned from the data and can see the effect of various attributes on the classification rules. In the case of association rules, the user may want to browse through the rules to select and view those which contain specific attributes in either the implicant or the consequent. More information regarding the quality of the learned model, such as the accuracy of the decision tree or the prevalence of the association rules, may also be visually represented along with the models. These may be displayed by queries on the visual representations to create new visual representations of the requested information.

Another task in data mining which can be aided by the VQLBCI framework is the guidance of the learning process. Users may provide guidance to the mining process at any stage. For example, in the case of learning a decision tree for classification, the user may provide grouping of several discrete attribute values to improve the quality of the rules learned. If the quality of the classification tree is not substantially improved by the expansion of some nodes, the user can also suggest that the mining algorithm not expand such nodes. The user could compare the results of two queries by issuing the queries in two different windows and then resizing/rearranging the resulting windows properly. The resulting visual representations from a query may be refined by the user by redefining the parameters of the conversation procedure using a query language interface. For example, let us consider a query retrieving the current traffic speed in different sections of a highway network. A visual representation may convert the numeric range ($>$ 80% of speed limit) to green, (50% – 80% of the speed limit) to yellow and ($<$ 50% of the speed limit) to red color for a map-based visual representation. The translation between attribute-value range and color can be redefined by the user by updating the conversion procedure parameters via a query.

Some other tasks in the data-mining life cycle can also be performed better using the VQLBCI framework. Evaluation and modification of the data-mining process, including its inputs and products, can be done through visual inspection and queries. The data quality of the learning samples as well as the warehouse data can also be visually inspected. The user can specify range checks on values and other constraints, and violations of this may be fed back through visual representations.

B: Classification functions

For our experimental evaluation we have selected the classification functions used in [AIS93]. These functions are described below. Each record was classified as either Group A or Group B. We describe only the conditions for a record to be classfied as Group A; all others are classified as Group B. The functions below use a C language like syntax for the sequential conditional expressions P ? Q : R. This is equivalent to the logical expression $(P \wedge Q) \vee (\neg P \wedge R)$.

Function 1

$(40 > age \geq 60)$

Function 2

$((age < 40) \wedge (50K \leq salary \leq 100K)) \vee$
$((40 \leq age < 60) \wedge (75K \leq salary \leq 125K)) \vee$
$((age \geq 60) \wedge (25K \leq salary \leq 75K))$

Function 3

$((age < 40) \wedge (elevel \in [0 \cdots 1])) \vee$
$((40 \leq age < 60) \wedge (elevel \in [1 \cdots 3])) \vee$
$((age \geq 60) \wedge (elevel \in [2 \cdots 4]))$

Function 4

$((age < 40) \wedge (((elevel \in [0 \cdots 1])?(25K \leq salary \leq 75K) : (50K \leq salary \leq 100K)))) \vee$
$((40 \leq age < 60) \wedge (((elevel \in [1 \cdots 3])?(50K \leq salary \leq 100K) : (75K \leq salary \leq 125K)))) \vee$
$((age \geq 60) \wedge (((elevel \in [2 \cdots 4])?(50K \leq salary \leq 100K) : (25K \leq salary \leq 75K))))$

Function 5

$((age < 40) \wedge (((50K \leq salary \leq 100K)?(100K \leq loan \leq 300K) : (200K \leq loan \leq 400K)))) \vee$
$((40 \leq age < 60) \wedge (((75K \leq salary \leq 125K)?(200K \leq loan \leq 400K) : (300K \leq loan \leq 500K)))) \vee$
$((age \geq 60) \wedge (((25K \leq salary \leq 75K)?(300K \leq loan \leq 500K) : (100K \leq loan \leq 300K))))$

Function 6

$((age < 40) \wedge (50K \leq (salary + commission) \leq 100K)) \vee$
$((40 \leq age < 60) \wedge (75K \leq (salary + commission) \leq 125K)) \vee$
$((age \geq 60) \wedge (25K \leq (salary + commission) \leq 75K))$

Function 7

$disposable > 0$ where,
$disposable = (0.67 * (salary + commission) - 0.2 * loan - 20K)$

Function 8

$disposable > 0$ where,
$disposable = (0.67 * (salary + commission) - 5000 * elevel - 20K)$

Function 9

$disposable > 0$ where,
$disposable = (0.67 * (salary + commission) - 5000 * elevel - 0.2 * loan - 10K)$

Function 10

$disposable > 0$ where,
$disposable = (0.67 * (salary + commission) - 5000 * elevel + 0.2 * loan - 10K)$
$equity = ((hyears < 20)?(equity = 0) : (equity = 0.1 * hvalue * (hyears - 20)))$

Contents

1 Introduction **1**

1.1 Data Mining Life Cycle . . . 1

1.2 Scope and Outline . . . 3

2 Related Work and Our Contributions **3**

2.1 Data Mining: A Brief Survey . . . 4

2.2 Visualization Techniques: A Brief Survey . . . 4

2.3 Our Contributions . . . 7

3 A Visual Query Language Based Control Interface **7**

3.1 Basic Concepts . . . 7

3.2 Components of VQLBCI . . . 9

4 Visual Development of Algorithms **11**

4.1 General Framework . . . 12

4.2 Steps In Visual Algorithm Development . . . 17

5 Experimental Evaluation of Visually Designed Algorithms **18**

5.1 Experiment Design . . . 19

5.2 An Experiment and Its Results . . . 20

6 Conclusions and Future Work **23**

7 Acknowledgments **24**

List of Figures

1 Information flow in the data mining life cycle . . . 2

2 Conceptual data model (ER) for VQLBCI . . . 10

3 A diagram to compare Entropy Gains . . . 11

4 Association Rules Diagram showing two example rules AB→C, and AD→E . . . 12

5 ER diagram for the search space of decision-tree learning algorithms and a form based interface for the manipulation of search parameters . . . 13

6 Search process . . . 14

7 Comparison of classification error for 10 data-sets. Five bars are shown for each problem. They represent C4.5, C4.5P, BF, CDP and CDP+ from left to right. . . . 21

8 Comparison of nodes generated for 10 data-sets. Five bars are shown for each problem. They represent C4.5, C4.5P, BF, CDP and CDP+ from left to right. . . . 21

9 Comparison of final decision tree size for 10 data-sets. Five bars are shown for each problem. They represent C4.5, C4.5P, BF, CDP and CDP+ from left to right. . . . 22