12-2006

Multivariate discretization of continuous valued attributes.

Ehab Ahmed El Sayed Ahmed 1978-
University of Louisville

MULTIVARIATE DISCRETIZATION OF CONTINUOUS VALUED ATTRIBUTES

By

Ehab Ahmed El Sayed Ahmed
B.S., October 6 University, 2001

A Thesis Submitted to the Faculty of the
Graduate School of the University of Louisville In Partial Fulfillment of the
Requirements For the Degree of

Master of Science in Computer Science

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky, USA

December 2006

MULTIVARIATE DISCRETIZATION OF CONTINUOUS VALUED ATTRIBUTES

By

Ehab Ahmed El Sayed Ahmed
B.S., October 6 University, 2001

A Thesis Approved on
December 28, 2006
By the following Thesis Committee:

Adel S. Elmaghraby, Co-Director

Ahmed Eman, Co-Director

Dar-jen Chang

Julius Wong, Mechanical Engineering

Khaled Wahba, RITI Director

DEDICATION

To my family, which gives me invaluable educational opportunities, a lot of effort to help me does this thesis.

ABSTRACT

WRITING ASSESSMENT AS SOCIAL ACTION

Ehab A.El Sayed Ahmed

December 28, 2006

The area of Knowledge discovery and data mining is growing rapidly. Feature Discretization is a crucial issue in Knowledge Discovery in Databases (KDD), or Data Mining because most data sets used in real world applications have features with continuously values. Discretization is performed as a preprocessing step of the data mining to make data mining techniques useful for these data sets.

This thesis addresses discretization issue by proposing a multivariate discretization (MVD) algorithm. It begins withal number of common discretization algorithms like Equal width discretization, Equal frequency discretization, Naïve; Entropy based discretization, Chi square discretization, and orthogonal hyper planes. After that comparing the results achieved by the multivariate discretization (MVD) algorithm with the accuracy results of other algorithms.

This thesis is divided into six chapters, covering a few common discretization algorithms and tests these algorithms on a real world datasets which varying in size and complexity, and shows how data visualization techniques will be effective in determining the degree of complexity of the given data set. We have examined the multivariate discretization (MVD) algorithm with the same data sets. After that we have classified

discrete data using artificial neural network single layer perceptron and multilayer perceptron with back propagation algorithm. We have trained the

Classifier using the training data set, and tested its accuracy using the testing data set. Our experiments lead to better accuracy results with some data sets and low accuracy results with other data sets, and this is subject to the degree of data complexity, then we have compared the accuracy results of multivariate discretization (MVD) algorithm with the results achieved by other discretization algorithms. We have found that multivariate discretization (MVD) algorithm produces good accuracy results in comparing with the other discretization algorithm.

TABLE OF CONTENTS

PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER I

INTRODUCTION TO KNOWLEDGE DISCOVERY AND DATA MINING

Nowadays each individual organization, business, family or institution can access a large quantity of data and information about itself and its environment. This data has the potential to predict the evolution of interesting variables or trends in the outside environment, but so far that potential has not been fully exploited. This is particularly true in the business field. There are two main problems. Information is scattered within different archive systems that are not connected with one another, producing an inefficient organization of the data. There is a lack of awareness about statistical tools and their potential for information elaboration. This interferes with the production of efficient and relevant data synthesis.

Two developments could help to overcome these problems. First, software and hardware continually, offer more power at lower cost, allowing organizations to collect and organize data in structures that give easier access and transfer. Second, methodological research, particularly in the field of computing and statistics, has recently led to the development of flexible and scalable procedures that can be used to analyze large data stores. These two developments have meant that data mining is rapidly spreading through many businesses as an important intelligence too. The information and knowledge gained from data mining tools can be used for applications ranging from business management to, production control and market analysis, and many others for backing up decisions.

Data Mining

Data mining refers to extracting or "mining" knowledge from large amounts of data. Giudici in 2003 [17] to understand the term it is useful to look at the literal translation of the word: to mine in English means to extract. The verb usually refers to mining operations that extract from the Earth her hidden, precious resources. The association of this word with data suggests an in-depth search to find additional information that previously went unnoticed in the mass of data available. From the viewpoint of scientific research, data mining is a relatively new discipline that has developed mainly from studies carried out in other disciplines such as computing, marketing, and statistics. Many of the methodologies used in data mining come from two branches of research, one developed in the machine learning community and the other developed in the statistical community, particularly in multivariate and computational statistics.

Machine learning is connected to computer science and artificial intelligence and is concerned with finding relations and regularities in data that can be translated into general truths. The aim of machine learning is the reproduction of the data-generating process, allowing analysts to generalize from the observed data to new, unobserved cases. The term Knowledge Discovery in Databases (KDD) was coined to describe all those methods that aimed to find relations and regularity among the observed data. Gradually the term KDD was expanded to describe the whole process of extrapolating information from a database, from the identification of the initial business aims to the application of the decision rules. The term "data mining" was used to describe the component of the KDD process where the learning algorithms were applied to the data.

This terminology was first formally put forward by Usama Fayyad at the First International Conference on Knowledge Discovery and Data Mining, held in Montreal in 1995 and still considered one of the main conferences on this topic. It was used to refer to a set of integrated analytical techniques divided into several phases with the aim of extrapolating previously unknown knowledge from massive sets of observed data that do not appear to have any obvious regularity or important relationships. As the term "data mining" slowly established itself, it became a synonym for the whole process of extrapolating knowledge. This is the meaning we shall use in this text. The previous definition omits one important aspect the ultimate aim of data mining. In data mining the aim is to obtain results that can be measured in terms of their relevance for the owner of the database-business advantage. Here are some complete definitions of data mining:

Fayyad in 1998, data mining is a step in the KDD process that, under acceptable computational efficiency limitations, enumerates structures (patterns or models) over the data.

Pyle in 1999, data mining is the process for transforming information into a form amenable to human cognition.

Kantardzic in 2001, data mining is the process of discovering various models, summaries and derived values from a given collection of data.

Emam in 2002, data mining is the process of discovering pattern / model by transforming the given collected data into understandable representation using different techniques based on the given dataset.

Giudici in 2003, data mining is the process of selection, exploration, and modeling of large quantities of data to discover regularities or relations that are at first unknown with the aim of obtaining clear and useful results for the owner of the database.

In a business context the utility of the result becomes a business result in itself. Therefore what distinguishes data mining from statistical analysis is not so much the amount of data we analyze or the methods we use but that we integrate what we know about the database, the means of analysis and the business knowledge. To apply a data mining methodology means following an integrated methodological process that involves translating the business needs into a problem which has to be analyzed, retrieving the database needed to carry out the analysis, and applying a statistical technique implemented in a computer algorithm with the final aim of achieving important results useful for taking a strategic decision. The strategic decision will itself create new measurement needs and consequently new business needs, setting off what has been called "the virtuous circle of knowledge" induced by data mining.

Data mining is not just about the use of a computer algorithm or a statistical technique; it is a process of business intelligence that can be used together with what is provided by information technology to support company decisions.

Data Mining Architecture

In industry, media, and in the database research milieu, the term data mining is becoming popular, so it must have a basic architecture that defines the components of the data mining process. Han et al. in 2001 [19] states that architecture of a data mining system may have the following major components:

- Database, data warehouse, or other information repository

This is one or set of databases, data warehouses, spreadsheets, or other kind of information repositories. Data cleaning and data integration techniques may be performed on the data.

- Database or data warehouse server

The database or data warehouse server is responsible for fetching the relevant data, based on the user's data mining request.

- Knowledge base

This is the domain knowledge that is used to guide the search, or evaluate the interestingness of resulting patterns. Such knowledge can include concept hierarchies, used to organize attributes or attribute values into different levels of abstraction. Knowledge such as user beliefs, which can be used to access a pattern's interestingness based on its unexpectedness, may also be included, other examples of domain knowledge are additional interestingness constraints or thresholds, and metadata (e.g., describing data from multiple heterogeneous sources).

- Data mining engine

This is essential to the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association, classification, cluster analysis, and evolution and deviation analysis.

- Pattern evaluation module

This component typically employs interestingness measures and interacts with the data mining modules so as to focus the search towards interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the pattern

evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used. For efficient data mining, it is highly recommended to push the evaluation of pattern interestingness as deep as possible into the mining process so as to confine the search to only the interesting patterns.

- Graphical user interface

This module communicates between users and the data mining system, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. In addition this component allows the user to browse database and data warehouse schemes or data structures, evaluate mind patterns, and visualize the patterns in different forms. Figure 1 the main components of a typical data mining system.

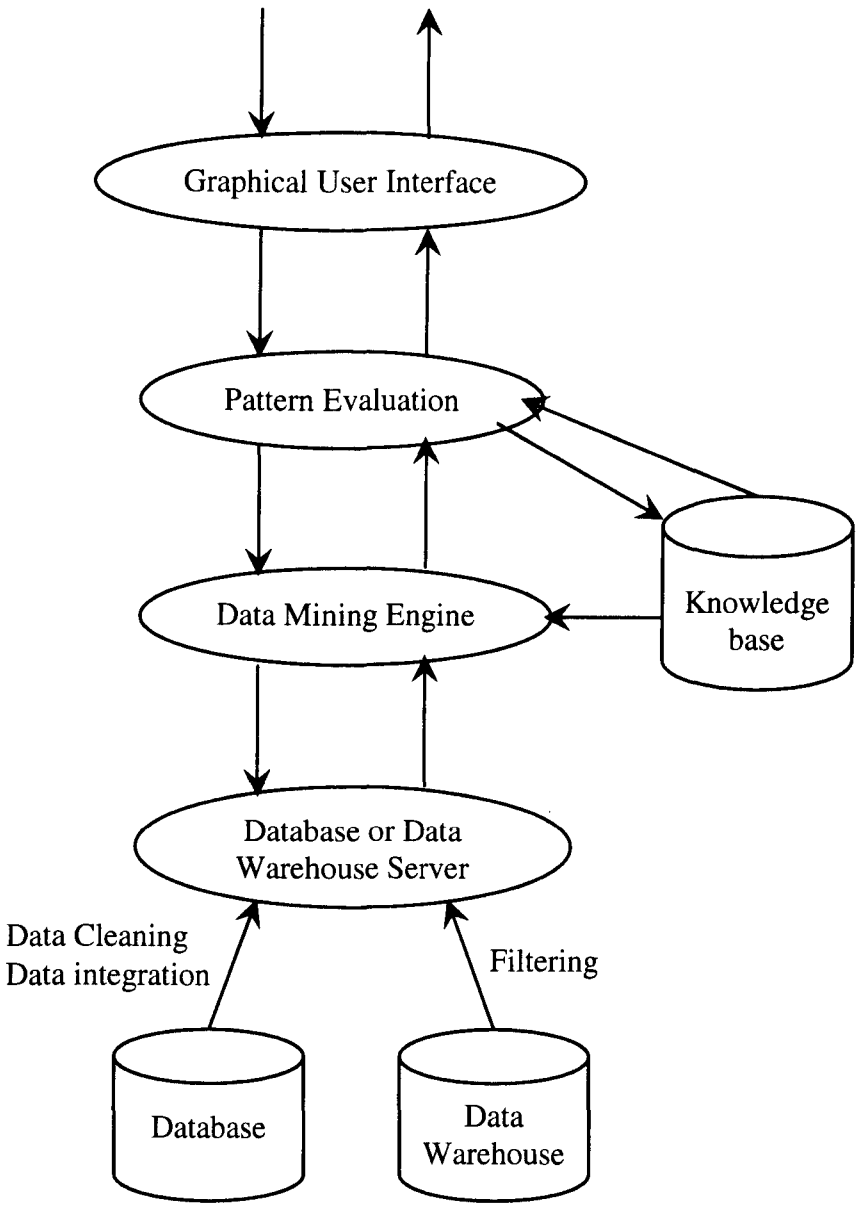

Figure 1: Architecture of a typical data mining system

While there may be many data mining systems on the market, not all of them can perform true data mining, data mining involves an integration of techniques from multiple disciplines such as a database technology, statistics, machine learning, high performance-computing, pattern recognition, neural networks, data visualization,

information retrieval, image and signal processing, and spatial data analysis. By performing data mining, interesting knowledge, regularities, or high-level information can be extracted from databases and viewed or browsed from different angles. The discovered knowledge can be applied to decision making, process control, information management, and query processing. Therefore, data mining is considered one of the most important frontiers in database systems and one of the most promising interdisciplinary developments in the information industry.

Data Mining Process

Giudici in 2003 [17] data mining is a series of activates from defining objectives to evaluating results. Here are the seven steps of data mining process:

- Definition of the objective for analysis

Definition of the objectives involves defining the aims of the analysis. It is not always easy to define the phenomenon we want to analyze. In fact, the company objectives that we are aiming for are usually clear, but the underlying problems can be difficult to translate into detailed objectives that need to be analyzed. A clear statement of the problem and the objectives to be achieved are the prerequisites for setting up the analysis correctly. This is certainly one of the most difficult parts of the process since what is established at this stage determines how the subsequent method is organized. Therefore the objectives must be clear and there must be no room for doubts or uncertainties.

- Organization of the data

Once the objectives of the analysis have been identified, it is necessary to select the data for the analysis. First of all it is necessary to identify the data sources. Usually

data is taken from internal sources that are cheaper and more reliable. This data also has the advantage of being the result of experiences and procedures of the company itself. The ideal data source is the company data warehouse, a storeroom of historical data that is no longer subject to changes and from which it is easy to extract topic databases, or data marts, of interest. If there is no data warehouse then the data marts must be created by overlapping the different sources of company data.

In general, the creation of data marts to be analyzed provides the fundamental input for the subsequent data analysis. It leads to a representation of the data, usually in a tabular form known as a data matrix that is based on the analytical needs and the previously established aims. Once a data matrix is available it is often necessary to carry out a preliminary cleaning of the data. In other words, a quality control is carried out on the available data, known as data cleansing. It is a formal process used to highlight any variables that exist but which are not suitable for analysis. It is also an important check on the contents of the variables and the possible presence of missing, or incorrect data. If any essential information is missing, it will then be necessary to review the phase that highlights the source.

Finally, it is often useful to set up an analysis on a subset or sample of the available data. This is because the quality of the information collected from the complete analysis across the whole available data mart is not always better than the information obtained from an investigation of the samples. In fact, in data mining the analyzed databases are often very large, so using a sample of the data reduces the analysis time. Working with samples allows us to check the models validity against the rest of the data, giving an important diagnostic tool. It also reduces the risk that the

statistical method might adapt to irregularities and loses its ability to generalize and forecast.

- Exploratory analysis of the data

Exploratory analysis of the data involves a preliminary exploratory analysis of the data, very similar to OLAP techniques. An initial evaluation of the data's importance can lead to a transformation of the original variables to better understand the phenomenon or it can lead to statistical methods based on satisfying specific initial hypotheses. Exploratory analysis can highlight any anomalous

Data-items that is different from the rest. These items data will not necessarily be eliminated because they might contain information that is important to achieve the objectives of the analysis. We think that an exploratory analysis of the data is essential because it allows the analyst to predict which statistical methods might be most appropriate in the next phase of the analysis. This choice must obviously bear in mind the quality of the data obtained from the previous phase. The exploratory analysis might also suggest the need for new extraction of data because the data collected is considered insufficient to achieve the set aims.

- Specification of statistical methods

There are various statistical methods that can be used and there are also many algorithms, so it is important to have a classification of the existing methods. The choice of method depends on the problem being studied or the type of data available. The data mining process is guided by the applications. For this reason the methods used can be classified according to the aim of the analysis. Then we can distinguish three main classes:

- Descriptive methods:

Aim to describe groups of data more briefly. They are also called symmetrical, unsupervised or indirect methods. Observations may be classified into groups not known beforehand (cluster analysis, Kohonen maps), variables may be connected among themselves according to links unknown beforehand (association methods, log-linear models, graphical models).

- Predictive methods

Aim to describe one or more of the variables in relation to all the others. They are also called asymmetrical, supervised or direct methods. This is done by looking for rules of classification or prediction based on the data. These rules help us to predict or classify the future result of one or more response or target variables in relation to what happens to the explanatory or input variables. The main methods of this type are those developed in the field of machine learning such as the neural networks (multilayer perceptrons) and decision trees but also classic statistical models such as linear and logistic regression models.

- Local methods

Aim to identify particular characteristics related to subset interests of the database; descriptive methods and predictive methods are global rather than local. Examples of local methods are association rules for analyzing transactional data.

From our discussion we think this classification is exhaustive, especially from a functional viewpoint. Each method can be used on its own right or as one stage in a multilayer analysis.

- Data analysis

Once the statistical methods have been specified, they must be translated into appropriate algorithms for computing calculations that help us synthesize the results we need from the available database. The wide range of specialized and non-specialized software for data mining means that for most standard applications it is not necessary to develop ad hoc algorithms. The algorithms that come with the software should be sufficient. Nevertheless, those managing the data mining process should have a sound knowledge of the different methods as well as the software solutions, so they can adapt the process to the specific needs of the company and interpret the results correctly when taking decisions.

- Evaluation of the statistical methods

To produce a final decision it is necessary to choose the best model of data analysis from the statistical methods available. Therefore the choice of the model and the final decision rule are based on a comparison of the results obtained with the different methods. This is an important diagnostic check on the validity of the specific statistical methods that are then applied to the available data. It is possible that none of the methods used permits the set of aims to be achieved satisfactorily. Then it will be necessary to go back and specify a new method that is more appropriate for the analysis.

When evaluating the performance of a specific method, as well as diagnostic measures of a statistical type, other things must be considered such as time constraints, resource constraints, data quality and data availability. In data mining it is rarely a good idea to use just one statistical method to analyze the data. Different methods have the potential to highlight different aspects, aspects which might otherwise have been ignored.

To choose the best final model it is necessary to apply and compare various techniques quickly and simply, to compare the results produced and then give a business evaluation of the different rules created.

- Implementation of the methods

Data mining is not just an analysis of the data; it is also the integration of the results into the decision process of the company. Business knowledge, the extraction of rules and their participation in the decision process allow us to move from the analytical phase to the production of a decision engine. Once the model has been chosen and tested with a data set, the classification rule can be applied to the whole reference population. For example we will be able to distinguish beforehand which customers will be more profitable or we can calibrate differentiated commercial policies for different target consumer groups, thereby increasing the profits of the company.

Having seen the benefits we can get from data mining, it is crucial to implement the process correctly to exploit its full potential. The inclusion of the data mining process in the company organization must be done gradually, setting out realistic aims and looking at the results along the way. The final aim is for data mining to be fully integrated with the other activities that are used to back up company decisions.

This process of integration can be divided into four phases:

- Strategic phase

In this first phase we study the business procedure being used in order to identify where data mining could give most benefits. The results at the end of this phase are the definition of the business objectives for a pilot data mining project and the definition of criteria to evaluate the project itself.

- Training phase

This phase allows us to evaluate the data mining activity more carefully. A pilot project is set up and the results are assessed using the objectives and the criteria established in the previous phase. The choice of the pilot project is a fundamental aspect. It must be simple and easy to use but important enough to create interest. If the pilot project is positive, there are two possible results: the preliminary evaluation of the utility of the different data mining techniques and the definition of a prototype data mining system.

- Creation phase

If the positive evaluation of the pilot project results in implementing a complete data mining system, it will then be necessary to establish a detailed plan to reorganize the business procedure to include the data mining activity. More specifically, it will be necessary to reorganize the business database with the possible creation of a data warehouse, to develop the previous data mining prototype until we have an initial operational version; and to allocate personnel and time to follow the project.

- Migration phase

At this stage all we need to do is prepare the organization appropriately so the data mining process can be successfully integrated. This means teaching likely users the potential of the new system and increasing their trust in the benefits it will bring. This means constantly evaluating (and communicating) the efficient results obtained from the data mining process.

For data mining to be considered a valid process within a company, it needs to involve at least three different people with strong communication and interactive skills:

1. Business experts, to set the objectives and interpret the results for data mining
2. Information technology experts, who know about the data and technologies needed.
3. Experts in statistical methods for the data analysis phase.

Data Mining Functionalities

Han et al. in 2001 [19] data mining functionalities are used to specify the kind of patterns to be found in data mining tasks. In general, data mining tasks can be classified into two categories: Descriptive and predictive. Descriptive mining tasks characterize the general properties of the data. Predictive mining tasks perform inference on the current data in order to make predictions.

Data mining systems should be able to discover patterns at various granularities (i.e., different levels of abstraction). Data mining systems should also allow users to specify hints to guide or focus the search for interesting patterns. Since some patterns may not hold for all of the data in the database, a measure of certainty is usually associated with each discovered pattern. Data mining functionalities, and the kinds of patterns they can discover, are as follows

1. Characterization and Discrimination

Data characterization is a summarization of the general characteristics or features of a target class of data. The data corresponding to the user-specified class are typically collected by a database query. For example, to study the characteristics of software products whose sales increased by 10% in the last year, the data related to such products can be collected by executing an SQL query. The output of a data characterization can be presented in various forms. Examples include pie charts, bar charts, curves, multidimensional data cubes, multidimensional tables, including cross-tabs. The resulting descriptions can also as generalized relations or in rule form (called characteristic rules).

Data discrimination is a comparison of the general features of target class data objects with the general features of objects from one or a set of contrasting classes. The target and contrasting classes can be specified by the user, and the corresponding data objects retrieved through database queries. For example, the user may like to compare the general features of software products whose sales increased by 10% in the last year with those whose sales decreased by at least 30% during the same period. The methods used for data discrimination are similar to those used for data characterization discrimination descriptions expressed in rule form are referred to as discriminant rules. The user should be able to manipulate the output for characteristic and discriminant descriptions.

2. Association Analysis

Association analysis is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data. Association analysis is widely used for market basket or transaction data analysis.

More formally, association rule are of the form $X \rightarrow Y$. The association rule $X \rightarrow Y$ is interpreted as records that satisfy the condition in X are also likely to satisfy the conditions in Y.

3. Classification and Prediction

Classification is the process of finding a set of models (or functions) that describe and distinguish data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known).

The derived model may be represented in various forms, such as classification (IF-THEN) rules, decision trees, mathematical formulae, or neural networks.

Classification can be used for predicting the class label of data objects. However, in many applications, users may wish to predict some missing or unavailable data values rather than class labels. This is usually the case when the predicated values are numerical data and is often specifically referred to as prediction.

Classification and prediction may need to be preceded by relevance analysis, which attempts to identify attributes that do not contribute to the classification or prediction process. These attributes can then be excluded.

4. Cluster Analysis

Unlike classification and prediction, which analyze class-labeled data objects, clustering analyzes data objects without consulting a known class label. The objects are clustered based on the principle of maximizing the interclass similarity and minimizing the interclass similarity. That is, clusters of objects are formed so that objects within a cluster have high similarity in comparison o one another, but are very dissimilar to objects in other clusters. Each cluster that is formed can be viewed as a class of objects, from which rules can be derived.

5. Outlier Analysis

Data may contain data objects that do not comply with the general behavior or model of the data. These data objects are outliers. Most data mining methods discard outliers as noise or expectations. However, in some applications such as fraud detection, they rare events can be more interesting than the more regularly occurring ones. The analysis of outlier data is referred to as outlier mining.

Outliers may be detected using statistical tests that assume a distribution or probability model for the data, or using distance measures where objects that are a substantial distance from any other cluster are considered outliers. Rather than using statistical or distance measures, deviation-based methods identify outliers by examining differences in the main characteristics of objects in a group.

6. Evolution Analysis

Data evolution analysis describes and models regularities or trends for objects whose behavior changes over time. Although this may include characterization, discrimination, association, classification, or clustering of time-related data, distinct features of such an analysis include time-series data analysis, sequence or periodicity pattern matching, and similarity-based data analysis.

Classification of Data Mining Systems

Han et al. in 2001 [19] data mining is an interdisciplinary field, the confluence of a set of disciplines as shown in figure 1.2, including database systems, statistics, machine learning, visualization, and information science. Moreover, depending on the data mining approach used, techniques from other disciplines may be applied, such as neural networks, fuzzy and/or rough set theory, knowledge representation, inductive logic programming, or high performance computing. Depending on the kind of data to be mined or on the given data mining application, the data mining system may also integrate techniques from spatial data analysis, information retrieval, pattern recognition, image analysis, signal processing, computer graphics, web technology, economics, business, bioinformatics, or psychology.

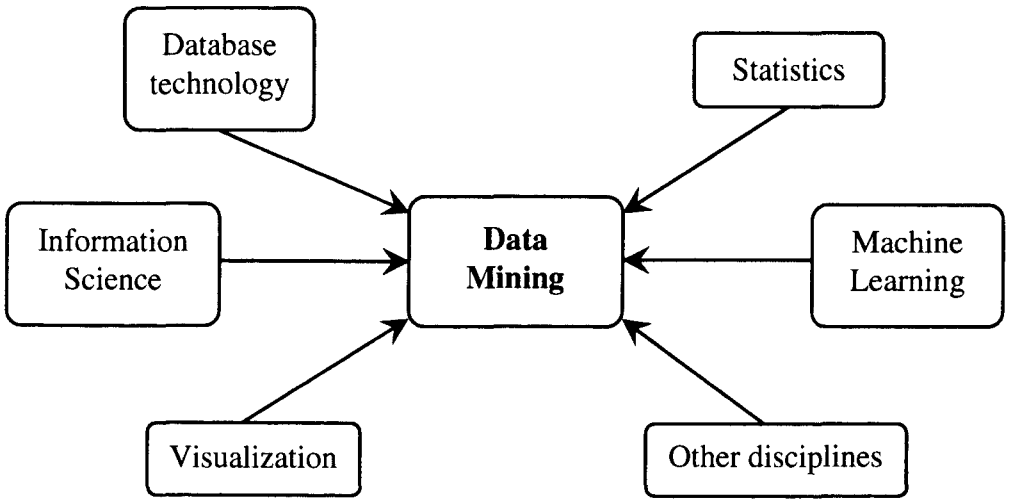

Figure 2: Data mining as a confluence of multiple disciplines

Because of the diversity of disciplines contributing to data mining, data mining research is expected to generate a large variety of data mining systems. Therefore, it is necessary to provide a clear classification of data mining systems. Such a classification may help potential users distinguish data mining systems and identify those that best match their needs. Data mining systems can be categorized according to various criteria, as follows:

- Classification according to the kind of databases mined

A data mining system can be classified according to the kinds of databases mined. Database systems themselves can be classified according to different criteria such as data models, or the types of data, or applications involved, each of which may require its own data mining techniques. Data mining systems can therefore be classified accordingly.

For instance, if classifying according to data models, we may have a relational, transactional, object-oriented, object-relational, or data warehouse mining system, if

classifying according to the special types of data handled, we may have a spatial, time-series, text, or multimedia data mining system, or a World Wide Web mining system.

- Classification according to the kinds of knowledge mined

Data mining systems can be categorized according to the kind of knowledge they mine, that is, based on data mining functionalities, such as characterization, discrimination, association, classification, clustering, outlier analysis, and evolution analysis. A comprehensive data mining system usually provides multiple and/or integrated data mining functionalities.

Moreover, data mining systems can be distinguished based on the granularity or levels of abstraction of the knowledge mined, including generalized knowledge (at high level of abstraction), primitive-level knowledge (at a raw data level), or knowledge at multiple levels (considering several levels of abstraction). An advanced data mining system should facilitate the discovery of knowledge at multiple levels of abstraction.

Data mining systems can also be categorized as those that mine data regularities (commonly occurring patterns) versus those that mine data irregularities (such as exceptions, or outliers). In general, concept description, association analysis, classification, prediction, and clustering mine data regularities, rejecting outliers as noise. These methods may also help to detect outliers.

- Classification according to the kinds of techniques utilized

Data mining systems can be categorized according to the underlying data mining techniques employed. These techniques can be described according to the degree of user interaction involved (e.g., autonomous systems, interactive exploratory systems,

query-driven systems) or the methods of data analysis employed (e.g., database-oriented, or data warehouse-oriented techniques, machine learning, statistics, visualization, pattern recognition, neural networks, and so on). A sophisticated data mining system will often adopt multiple data mining techniques or work out an effective, integrated technique that combines the merits of a few individual approaches.

- Classification according to the application adopted

Data mining systems can also be categorized according to the application they adopt. For example, there could be data mining systems tailored specifically for finance, telecommunications, DNA, stock markets, e-mail, and so on. Different applications often require the integration of application-specific methods. Therefore, a generic all-purpose data mining system may not fit domain-specific mining tasks.

Data Mining Profitable Applications

A wide range of companies have deployed successful applications of data mining. While early adopters of this technology have tended to be in information-intensive industries such as financial services and direct mail marketing, the technology is applicable to any company looking to leverage a large data warehouse to better manage their customer relationships. Two critical factors for success with data mining are: a large, well-integrated data warehouse and a well-defined understanding of the business process within which data mining are to be applied (such as customer prospecting, retention, campaign management, and so on).

Some successful application areas include:

- A pharmaceutical company can analyze its recent sales force activity and their

Result to improve targeting of high-value physicians and determine which marketing activities will have the greatest impact in the next few months. The data needs to include competitor market activity as well as information about the local health care systems. The results can be distributed to the sales force via a wide-area network that enables the representatives to review the recommendations from the perspective of the key attributes in the decision process. The ongoing, dynamic analysis of the data warehouse allows best practices from throughout the organization to be applied in specific sales situations.

- A credit card company can leverage its vast warehouse of customer transaction

Data to identify customers most likely to be interested in new credit product, using a small test mailing, the attributes of customers with an affinity for the product can be identified. Recent projects have indicated more than a 20-fold decrease in costs for targeted mailing campaigns over conventional approaches.

- A diversified transportation company with a large direct sales force can apply

data mining to identify the best prospects for its services. Using data mining to analyze its own customer experience, this company can build a unique segmentation identifying the attributes of high-value prospects. Applying this segmentation to a general business database can yield a prioritized list of prospects by region.

- A large consumer package goods company can apply data mining to improve

Its sales process to retailers. Data from consumer panels, shipments, and competitor activity can be applied to understand the reasons for brand and store switching. Through this analysis, the manufacturer can select promotional strategies that best reach their target customer segments.

Each of these examples has a clear common ground. They leverage the knowledge about customers implicit in a data warehouse to reduce costs and improve the value of customer relationships. These organizations can now focus their efforts on the most important (profitable) customers and prospects, and design targeted marketing strategies to best reach them.

Challenges in Data Mining

Kohavi in 2000 [30] lists some challenges that face the evolution of data mining; here are some of these challenges:

- Make data mining models comprehensible to business users

Business users need to understand the results understand the results of data mining. Few data mining models are easy to understand and techniques need to be developed to explain or visualize existing ones or new models that are simple to understand with matching algorithms need to be derived. This is particularly hard for regression models. A related problem is that association algorithms usually derive too many rules (e.g., 100,000) and we need to find ways to highlight the "interesting" rules or families of associations.

- Make data transformations and model building accessible to business users

An important issue that has not been mentioned above is the need to translate user's questions into a data mining problem in relational format. This often requires writing SQL, Perl scripts, or small programs. Even defining what the desired transformations and features should be is a knowledge-intensive task requiring significant understanding of the tasks, the algorithms, and their capabilities.

- Scale algorithms to large volumes of data

Berry et al. in 2000 [3] estimated that the amount of text in the Library of Congress can be stored in about 17 terabytes of disk space. The package-level detail database used to track shipments at UPS is also 17 terabytes. Most data mining algorithms can handle a few gigabytes of data at best, so there are three to four orders of magnitude to grow before we can attack the largest databases that exist today. In addition, most algorithms learn in batch, but many applications require real-time learning.

- Close the loop: identify causality, suggest actions, and measure their effect

Discoveries may reveal correlations that are not causal. For example, human reading ability correlates with shoe size, but wearing larger shoes will not improve one's reading ability. (The correlation is explained by the fact that children have smaller shoe sizes and cannot read as well.) Controlled experiments and measurements of their effects can help pinpoint the causal relationships. One advantage of the online world is that experiments are easy to conduct: changing layout, emphasizing certain items, and offering cross-sells can all be easily done and their effect can be measured. For electronic commerce, the World Wide Web is a great laboratory for experiments, but our learning techniques need to improve to offer interventions and take them into account.

- Cope with privacy issues

Data mining holds the promise of reducing the amount of junk mail we receive by providing us with more targeted messages. However, data collection can also lead to neglects of the data, rising with many social and economic issues. This is doubly true in the online world where every page and every selection we make can be recorded.

The Concept of Knowledge Discovery

Risvik et al. in 1997 [52] notice that data are being collected and accumulated at dramatically high rates across a wide variety of fields. There is a need for computational tools to extract useful information from the tremendous amount of data that is rapidly growing. Theories and tools for such extraction are subject of the field of Knowledge discovery in Databases (KDD).

An informal definition of KDD is the methods and techniques used to make sense of the data. Most sampling processes return a huge amount of data, way to large to digest and interpret manually. The main goal of KDD is to reduce the amount of data, while preserving the knowledge that the original set of data represents.

The Knowledge Discovery Process

Fayyad et al. in 1996 [14] formalized the definition of the KDD process: Knowledge Discovery in Databases is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

Here data is a set of facts. (e.g., cases in a database), and pattern is an expression in some language describing a subset of the data or a model applicable to that subset. The term process implies that Knowledge Discovery in Databases (KDD) is comprised of many steps, which involve data preparation, search for patterns, knowledge evaluation, and refinement, all repeated in multiple iterations. By non-trivial we man that some search or inference is involved, i.e. it is not a straightforward computation of predefined quantities like computing the average value of a set of numbers. The discovered patterns should be valid for new data with some degree of certainty. We also want patterns to be novel (at least to the system and preferably to the user) and potentially useful, i.e. lead to

some benefit to the user/task. Finally, the patterns should be understandable, if not immediately, then after some post processing.

The above implies that quantitative measures for evaluating extracted patterns. In many cases, it is possible to define measures of certainty (e.g., estimated prediction accuracy on new data) or utility (e.g. gain, perhaps in dollars saved due to better predictions or speed-up in response time of a system) . Notations such as novelty and understandability are much more subjective. In certain contexts understandability can be estimated by simplicity (e.g., the number of bits to describe a pattern). An important nation, called interestingness, is usually taken as an overall measure of a pattern value, combining validity, novelty, usefulness, and simplicity. Interestingness functions can be manifested implicitly via an ordering placed by the Knowledge Discovery in Databases (KDD) system on the discovered patterns or models.

KDD process is the process of using the databases along with any required selection, preprocessing, subsampling, and transformation of it; to apply data mining methods (algorithms) to enumerate patterns from it; and to evaluate the products of data mining to identify the subset of the enumerated patterns deemed "knowledge". The knowledge discovery process is generally most successful if some understanding of the domain already exists, while at the same time there is still knowledge to be discovered from the data set being analyzed, Mannila et al. in 1996 [37]. Some understanding is usually required to preprocess data and interpret the output of the data mining step of the process. On the other hand, if the domain is already well understood, the output of the data mining process will most likely be knowledge that has been previously discovered.

The output of the knowledge discovery process can be on various formats, depending on the nature of the analyzed information. Examples of output formats are frequently occurring patterns, clustering of data object, or a set of rules. The process of Knowledge Discovery is a multi-step process, and it can be described as a pipeline refining raw data into knowledge. Figure 3 illustrates the different steps in the KDD process. From the figure we notice that Data Mining is the core of the Knowledge Discovery process (KDD).

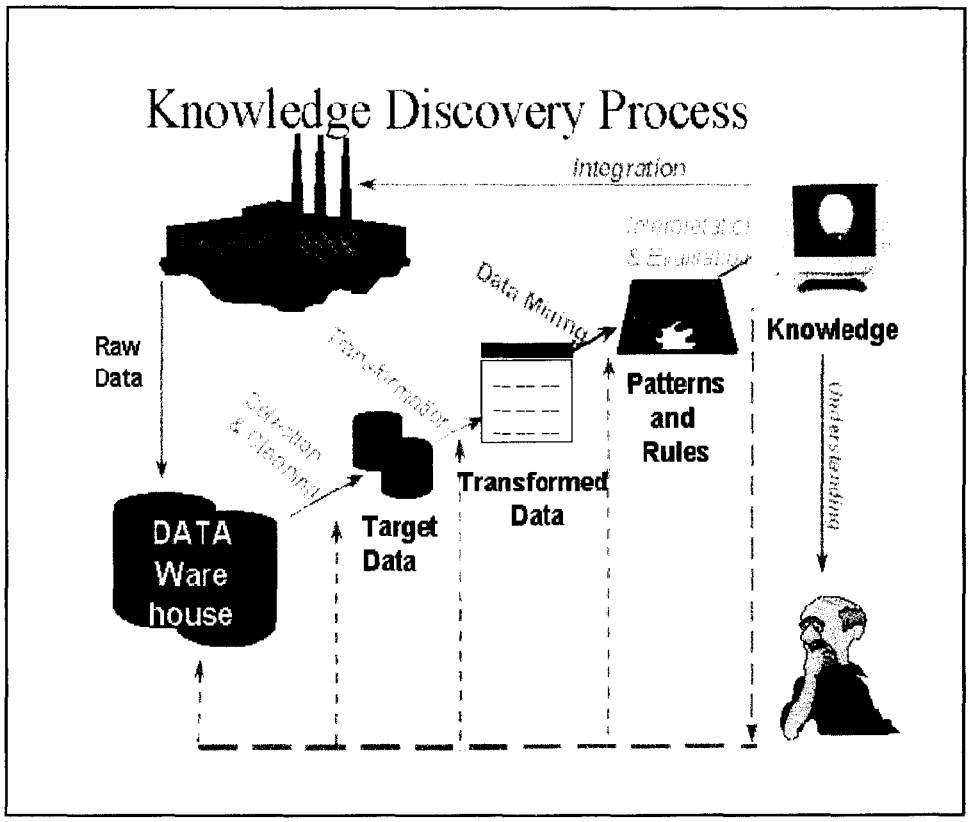

Figure 3: Overview of the steps comprising the KDD process

Fayyad et al. in 1996 [14] the knowledge discovery process is interactive and iterative with many decisions being made by the user, involving numerous steps, summarized as:

1. Developing an understanding of the application domain and the relevant prior knowledge, and identifying the goal of the KDD process from the customer's viewpoint.
2. Creating a target dataset: selecting a dataset, or focusing on a subset of variables or data samples, on which discovery is to be performed.
3. Data cleaning and preprocessing: basic operations such as the removal of noise or outliers if appropriate, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, accounting for time sequence information and known changes.
4. Data reduction and projection: finding useful features to represent the data depending on the goal of the task, using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representations for the data.
5. Matching the goals of the KDD process (step 1) to a particular data mining methods: e.g., summarization, classification, regression, clustering, etc.
6. Choosing the data mining algorithm(s): selecting method(s) to be used for searching for patterns in the data, this includes deciding which models and parameters may be appropriate (e.g., models for categorical data are different than models on vectors over the reals) and matching a particular data mining method

with the overall criteria of the KDD process (e.g., the end-user may be more interested in understanding the model than in its predictive capabilities)

7. Data mining: searching for patterns of interest in a particular representational form or a set of such representations: classification rules or trees, regression, clustering, and so forth. The user can significantly aid the data mining method by correctly performing the preceding steps.
8. Interpretation mined patterns, possibly return to any of steps 1-7 for further iteration. This step can also include visualization of the extracted patterns/models, or visualization of the data given the extracted models.
9. Consolidating discovered knowledge: incorporating this knowledge into another system for further action, or simply documenting it and reporting it to interested parties, this also includes checking for the resolving potential conflicts with previously believed (or extracted) knowledge.

Most previous work on KDD focused primarily on the data mining step. However, the other steps are equally if not more important for the successful application of KDD in practice. In the next section we now focus on the relation between the KDD process and data mining.

Knowledge Discovery and Data Mining

Fayyad et al. in 1996 [14] formalized the definition of the Knowledge Discovery in Databases (KDD):

Knowledge Discovery in Databases (KDD) is concerned with extracting useful information from databases. Data mining is a set of techniques used in an automated

approach to exhaustively explore and bring to the surface complex relationships in very large datasets.

Moxon in 1996 [39] the adopted approaches are discovery-based in which pattern-matching and other algorithms can look at numerous multidimensional data relations concurrently, highlighting those that are dominant or exceptional.

Data mining is usually viewed as a single step in the KDD process. Considering the steps at the level of feature selection, the summarized four basic steps in the process are:

- Data warehousing: observations from different sources are recorded as data.
- Pre-processing: which includes target data selection, data cleaning, Projection and reduction, data are selected for better use, given an application.
- Data mining: which includes model selection, a proper mining algorithm is chosen and rules are learned, here rules are a generic term for what is mined and discovered.
- Post-processing: which includes evaluation and interpretation and consolidation of the discovered information, rules are sifted or groups for their better use and understanding, since it is often that the number of rules is large, and some rules are more interesting than others. See Piatetsky-Shapiro in 1991 [46], Kamber et al. in 1996 [27], Silberschatz et al., in 1996 [54], Liu et al. in 1996 [32]. The four-step process is shown in Figure 4.

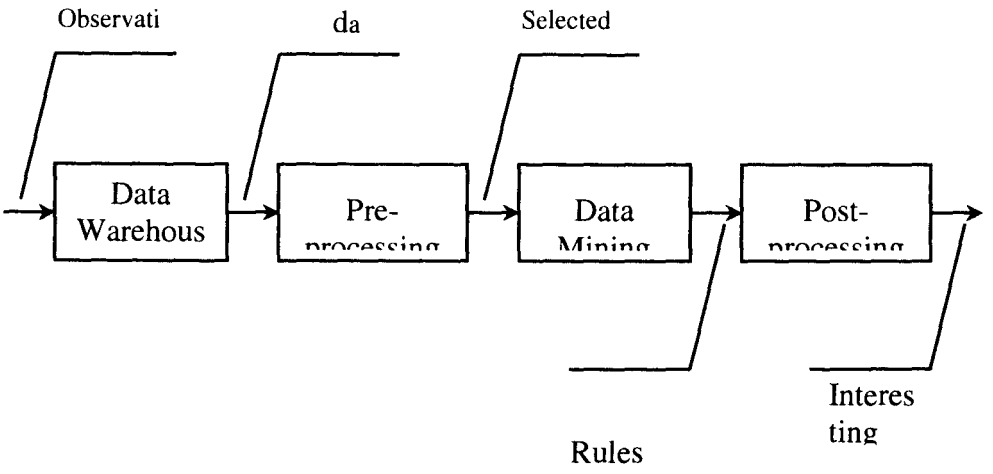

Figure 4: A general model of knowledge discovery and data mining (KDD)

Summary

In this chapter, we have introduced the term Data Mining and Knowledge Discovery in Database (KDD). Many scientists have defined the term Data Mining and Knowledge Discovery in database (KDD) like Fayyad. Data mining itself is a step of the KDD process.

Data mining has becoming popular and used in many field so it must have a predefined components that encompasses its major components, these components include data repository or data warehouse, knowledge base which used to guide the search, some modules which accomplish the data mining task, and finally we need a friendly user interface.

Data mining process consists of a series of activities from defining the objectives to evaluating results, these activities include defining of the analysis objective, organizing the data by identifying data sources, preliminary exploratory analysis of the data to have

better understanding of it, determining the statistical method that will be used in the mining process, it may be combine more than one method in a multilayer analysis, after that statistical methods must be translated into appropriate algorithms for computing calculations that help us synthesize the results, finally the evaluation of the statistical methods to choose the best data analysis model. In creation of valid data mining model we need to involve at least three different people with strong communication and interactive skills: business experts, information technology specialist, and experts for statistical methods for the data analysis phase.

Data mining has much functionality that used to specify the kind of patterns to be found. These functionalities include characterization and discrimination, association analysis, classification and prediction, cluster analysis, outlier analysis, and evolution analysis. Data mining is an interdisciplinary field, the confluence of a set of disciplines including database systems, statistics, machine learning, visualization, and information science. Because of the diversity of disciplines contributing to data mining, data mining research is expected to generate a large variety of data mining systems. Data mining systems can be categorized according to the kind of: databases or knowledge mined, the kinds of techniques utilized, and the kind of application adopted.

The evolution of data mining faces some challenges, some of these challenges are make models comprehensive, make data transformations and model building accessible to business users, scale algorithms to large volumes of data, close the loop, and finally cope with privacy issues.

The Knowledge Discovery in Databases (KDD) is a complicated task that comprises many steps; one of these steps is the data mining itself. The main goal of the

KDD process is to reduce the amount of data, while preserving the knowledge that the original set of data represents. The output of the knowledge discovery process can be on various formats, depending on the nature of the analyzed information. Examples of output formats are frequently occurring patterns, clustering of data object, or a set of rules. The process of knowledge discovery is a multi-step process, and it can be described as a pipeline refining raw data into knowledge, with many decisions being made by the user, these steps summarized as: developing an understanding of the application domain, creating a target dataset by focusing on a subset of variables or data samples on which discovery is to be performed, performing data cleaning and preprocessing: basic operations, data reduction and projection: finding useful features to represent the data and using dimensionality reduction to reduce the effective number of variables under consideration, matching the goals of the KDD process to a particular data mining methods: e.g., summarization, classification, regression, clustering, etc, choosing the data mining algorithm(s), data mining: searching for patterns of interest in a particular representational form, interpretation mined patterns, and finally consolidating discovered knowledge.

Data Mining is usually viewed as a single step in the KDD process. Considering the steps at the level of feature selection, the summarized four basic steps in the process are data warehousing, preprocessing, data mining, and post processing which represents the extracted pattern.

CHAPTER II

OVERVIEW OF DISCRETIZATION

Background

Many real-world data that will be applied to a data mining technique needs to some sort of preparation. Data preparation is an important step of the data mining process that influences that result and accuracy of the data mining technique. In practice, it has been generally found that data cleaning and preparation takes approximately eighty percent of the total engineering effort. Data preparation is, therefore, a crucial research topic. However, much work of the data mining techniques was built on the existence of quality data. That is, the input of the data mining technique is assumed to be nicely distributed, containing no missing or incorrect values where all feature are important. Data preparation has a great importance in many aspects, like that, it generates a dataset that is smaller than the original one, which can significantly improve the efficiency of data mining and decrease the computational time of the data mining technique, and generates quality data, which leads to quality patterns.

Zhang et al. in 2003 [64] data preparation itself is a complicated task that comprises several steps: (1) convert data from any format to a data mining ready format. (2) dealing with missing values. (3) data transformation: the data are transformed or consolidated into forms appropriate for mining. This can be done in several ways as: (a) smoothing. (b) aggregation. (c) normalization. (d) generalization. (e) feature composition.

(f) data reduction: responsible for reduce data comes from data warehouses which comes in huge amounts of data, by applying data Analysis and mining tools on this huge amount of data may take a very long time, making such analysis impractical or infeasible. Data reduction techniques can be applied to obtain a reduced representation of the dataset that is much smaller in volume, until now closely maintains the integrity of the original data. Strategies for data reduction include: (1) feature discretization: which is the main subject of the thesis, we will take it into details in our literature review, (2) sampling, (3) feature selection.

Discretization is a data processing procedure. It transforms an attribute (feature) from one type into another type. It divides the range of the attribute into intervals. Interval labels can then be used to replace actual data values, data can be divided to two different categories as: "quantitative" vs. "qualitative", "continuous" vs. "discrete", "ordinal" vs. "nominal", and "numeric" vs. "categorical". There are four common types of scales for data: nominal, ordinal, interval and ratio.

The classification applied to attribute types: qualitative attributes comprises two scale levels: nominal, ordinal levels, quantitative attributes comprises interval, either discrete or continuous, and ratio, either discrete or continuous. Discrete attributes makes no difficulties with discretization algorithms, but continuous attributes degrades the accuracy, and increase computational time of the discretization algorithms. Discretization algorithms convert continuous feature to a discrete one.

There exist diverse organizations to classify discretization algorithms. Different classification emphasizes different aspects of the distinctions among discretization algorithms. Typically, discretization algorithms can be classified into either primary or

composite. Primary methods accomplish discretization without reference to any other discretization method. Composite methods are built on top of a primary method. Primary methods can be classified as: (1) supervised vs. unsupervised, (2) univariate vs. multivariate, (3) parametric vs. non-parametric, (4) hierarchical vs. non-hierarchical, (5) global vs. local, (6) eager vs. lazy, (7) disjoint vs. non-disjoint, (8) dynamic vs. static.

Liu et al. in 1999 [33] argue a typical discretization process consists of four steps: (1) sorting the continuous values of the feature to be discretized, (2) evaluating the cut-point for splitting or adjacent intervals for merging, (3) according to some criterion, splitting or merging intervals of continuous values, and (4) finally stopping at some criteria. Any discretization algorithm must perform these four steps.

Assuming we have different discretization algorithms and each provides some kind of a discretized data. Which discretized data is the best? The result evaluation is a complex issue and depends on a users need in a particular application. It is complex because the evaluation can be done in many ways. We list three important dimensions: (1) the total numbers of intervals, the fewer the cut points (2) the number of inconsistencies caused by discretization must be less than number of inconsistencies in the original data, (3) predictive accuracy, we need at least three dimensions: simplicity, consistency, and accuracy. Some discretization algorithms cannot achieve the three departments.

There are several discretization algorithms available in the machine learning community. Liu et al. in 1999 [33] proposed a hierarchical framework for the discretization process that is systematic and expandable, and attempted to cover all existing methods. Each discretization method discretized a feature by either splitting the

interval of continuous values or by merging the adjacent intervals (level 1). Then they consider whether a method is supervised or unsupervised (level 2). They further group together methods that use similar discretization measures (level 3) e.g. binning and entropy. The supervised and unsupervised division determines different (non-overlapping) measures used. There may be variation in these measures Mantaras distance in Cerquides et al. in 1997 [7] which categorized here under entropy measure for their similarity.

In addition to discretization, there are numerous data reduction techniques. Sampling is a data reduction technique that is allows a large dataset to be represented by a much smaller random sample of the data. There are diverse sampling techniques like: (1) Simple Random Sample without Replacement (SWSWOR) on size n, where n is the size of the sample, (2) Simple Random Sample with Replacement (SRSWR) on size n, (3) cluster sample, (4) stratified sample.

Feature selection is also a data reduction technique which selects a subset of the original dataset by eliminating redundant and uninformative features. The process of feature selection extracts as much as information as possible from a given dataset while using the smallest number of features, this save significant computing time and often build models that generalize better to unseen points. It is important for several reasons, the fundamental one being arguably that noisy features can degrade the performance of most learning algorithms.

The selection criteria of a feature can be done from measuring perspective, there exist many measures that can be used to select features, such that information measures, distance measures, dependence measures, consistency measures, and accuracy measures.

Selecting features by using one of these measures will lead to reduce dimensionality and remove noise, improve the accuracy of the mining algorithm. Feature selection has been widely studied in the context of supervised learning; it has received comparatively very little attention in unsupervised learning or clustering.

There are several methods that have been proposed to discretize data in the machine learning community; we will look at some of these algorithms:

- Equal Width Discretization (EWD)

This algorithm discretizes data and has been applied as a means for producing nominal values from continuous ones. It involves sorting the observed values of a continuous feature and dividing the range of observed values for the variable into k equally sized bins, where k is a parameter supplied by the user. The variable x is observed to have values bounded by x_{min} and x_{max} then this method computes the bin width by using the equation: $\delta = x_{max} - x_{min} / k$

- Equal Frequency Discretization (EFD)

This algorithm divides the sorted values into k intervals so that each interval contains approximately the same number of training instances. Thus each interval contains n/k adjacent values. k is a user predefined parameter, note that the training with identical values must be placed in the same interval. In consequence it is not always possible to generate k-equal frequency intervals.

- A Naïve Discretization Algorithm

A naïve discretization algorithm sorts the objects according to the value of the attribute being considered, the algorithm works as follows: (1) select attribute $a \in A$, (2) set $Ca = \emptyset$, (3) sort values of the attribute, (4) for all adjacent pair of objects if $Va(i) \neq Va(i+1)$,

and d(i) ≠ d(i+1), create new cut c = (Va(i) + Va(i+1))/2, and Ca = Ca ∪ c, (5) repeat 2-4 for all continuous attributes a in A.

Where C is the class that the instance belongs to, and V is the value of the attribute being discretized.

- ChiMerge Discretization Algorithm

Kerber in 1992 [29] was first proposed ChiMerge method to provide a statistically justified heuristic method for supervised discretization. The ChiMerge algorithm consists of an initialization step which placing each observed real value into its own interval and proceeds by using the χ^2 test to determine when adjacent intervals should be merged. This bottom-up merging process is repeated until a stopping criterion is met. The formula for computing the χ^2 value is:

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} (A_{ij} - E_{ij})^2 / E_{ij}$$

Where: k = number of classes, A_{ij} = number of patterns in the interval,

R_i = number of patterns in the ith interval = $\sum_{j=1}^{K} A_{ij}$,

C_j = number of patterns in the $_j$th class = $\sum_{i=1}^{2} A_{ij}$,

N = total number of patterns = $\sum_{i=1}^{2} R_i$,

E_{ij} = expected frequency of A_{ij}; $E_{ij} = R_i * C_j / N$.

The value of the χ^2 threshold is determined by selecting a desired significance level α and then using a table or formula to obtain the corresponding χ^2 value.

- Chi2 Discretization Algorithm

Liu et al. in 1995 [35] used the ChiMerge algorithm as a basis for their Chi2 algorithm. He divides the algorithm into two phases. In the first phase begins with high significance level (sigLevel), e.g., 0.5, for all numeric attributes for discretization. Each attribute is sorted according to its values. Then the following is performed: (1) calculate the χ^2 value for every pair of adjacent intervals (at the beginning, each pattern is put into its own interval that contains only one value of an attribute), (2) merge the pair of adjacent intervals with the lowest χ^2 value. Merging continuous until all pairs of intervals have χ^2 values exceeding the parameter determined by sigLevel (initially, 0.5, its corresponding χ^2 value is 0.455 if the degree of freedom is 1, more below).

- Entropy-Based Discretization Algorithm

Fayyad, et al. in 1993 [12] proposed a recursive method that finds intervals that minimize the class information entropy. This supervised algorithm uses the class information entropy of candidate partitions to select bin boundaries for discretization. It first sorts all the examples by the attribute being discretized. The optimal boundary T* that minimizes the entropy is chosen to partition the range, and the algorithm is applied recursively to the sets to the left and right of T*. The resulting intervals will be closer to each other in the regions with high entropy, and far apart in the uniform regions where the entropy is low. A Minimum Description length (MDL) Criterion is applied to decide when to stop discretization. In a sense, supervised discretizers create a rough model that predicts the class label based on the intervals

- Orthogonal Hyperplanes

Skowron et al. in 1995 [55] proposed a method for discretization based on Rough Sets and Boolean Reasoning. Orthogonal hyperplanes can be seen as a special case of linear discriminations, used in pattern recognition. The algorithm is as follow:

(1) $P = \emptyset$. $L = \{U\}$; A_I = initial set of cuts on A, (2) for $c \in A_I$ do compute ($W_P(c)$), (3) choose c_{max} with the largest value $W_P(c_{max})$ among cuts from A_I and set

$P = P \cup \{c_{max}\}$; $A_I = A_I \setminus \{ c_{max} \}$, (4) for $X \in L$ do

If c_{max} cuts the set X into X_1, X_2 then

Remove X from L; Add to L the two sets X_1, X_2;

(4) If all set from L consists of objects from one decision class only then Stop else

Go to step 2.

P is the semi-minimal set of cuts.

Problem Statement

Many real world data are highly susceptible to noisy, inconsistent data due to the typically huge size. These data can be preprocessed in order to improve the quality of the data, and consequently, of the mining algorithm results. Data preparation is a fundamental stage in the data mining process which is a step in the KDD process. One needs to prepare quality data by preprocessing the raw data. In practice, it has been found that data cleaning and preparation takes approximately eighty percent of the total engineering effort. Data preparation is, therefore, a crucial research topic; much work in the field of data mining was built on the existence of quality data. That is, the input of the data mining algorithms is assumed to be nicely distributed, containing no missing or incorrect values where all features are important.

There are a number of data preprocessing techniques. Data cleaning can be applied to remove noise are correct inconsistencies in the data. Data integration merges data from multiple sources into a coherent data store, such as a data warehouse or data cube. Data transformation, such as normalization, and smoothing, may be applied. For example, normalization may improve the accuracy and efficiency of the mining algorithms involving distance measurements. Data reduction is very important step in the data preparation process where it reduces the size of the original dataset; real world datasets are coming in huge size which increase the computational time and degrade the accuracy of the mining algorithm. Data reduction can reduce the data size by eliminating redundant features, or aggregation, for instance. These data preprocessing techniques, when applied prior to the mining algorithm, can substantially improve the overall quality of the patterns mined and/or the time required for the actual mining. Data reduction contains two important steps which are feature selection, and discretization. Feature selection, and discretization are very important steps in the data preparation process. Feature selection works by choosing a subset of the original predictive variables by eliminating redundant and uninformative ones. By extracting as much as information as possible from a given dataset while using the smallest number of features, we can save significant computing time and often build models that generalize better to unseen points.

Data usually comes in a mixed format: discrete, and/or continuous. Discrete and continuous data are ordinal data types with order among the values, while nominal values do not possess any order amongst them. Discrete values are intervals in a continuous spectrum of values. While the number of continuous values for an attribute can be infinitely many, the number of discrete values is often few or finite. There is a need to

discretize continuous features by converting them to continuous ones. Widely used systems such as C4.5 and CART deploy various ways to avoid using continuous values directly. There are many other advantages of using discrete values over continuous ones. Discrete features are closer to a knowledge-level representation than continuous ones. Data can also be reduced and simplified through discretization. For both users and experts, discrete features are easier to understand, use, and explain. Discretization makes learning more accurate and faster. In general, obtained results using discrete features are usually more compact, shorter and more accurate than using continuous ones; hence the results can be more closely examined, compared, used and reused. In addition to the many advantages of having discrete data over continuous one, a suite of classification learning algorithms can only deal with discrete data. Discretization is a process of quantizing continuous attributes. The success of discretization can significantly extend the borders of many learning algorithms.

There exist a number of discretization algorithms in the research area of machine learning. Many real world data have numeric attributes with continuous values. The major task of discretization algorithms is to convert continuous value attributes into discrete values, and some of them extend to merging intervals until a known consistency level is reached. Discretization algorithms have been developed along different lines due to different needs: supervised vs. unsupervised, univariate vs. multivariate, parametric vs. non-parametric, hierarchical vs. non-hierarchical, global vs. local, eager vs. lazy, disjoint vs. non-disjoint, and dynamic vs. static. We have select several algorithms that have been tested many times and have a wide range of usage, The most common of them are equal width discretization (EWD), equal frequency discretization (EFD), Naïve, Entropy-based

discretization, Chi-square discretization algorithm, and orthogonal hyperplanes, but these algorithms are categorized as single value discretization algorithms, some of these algorithms have only been tested on small datasets with hundreds of instances. Since large datasets with high dimensional attribute spaces and huge numbers of instances are increasingly used in real-world applications.

In this thesis we develop a Multivariate Discretization (MVD) algorithm and testing it with a large data sets, showing the effect of the new technique on the efficiency and accuracy of data mining algorithm, and comparing the accuracy results with the other experimented discretization algorithms, we develop an automated and application independent system that takes the dataset as an input (we compare and test the system by selecting some of the our experimental data sets from the UCI machine learning repository by Blake, et al. in 1998 [4]). The data sets used through our experiments comes from different sources and varying in size and complexity. In addition, we use a data visualization tools that visualize data and help in determining the nature of the data if it is consistent or inconsistent and choose the suitable algorithm which gets the best accuracy results. Then run the discretization algorithms on the experimental data sets to convert continuous data values to discrete one. After that the data set will be classified using an artificial neural network classifier, the accuracy will be computed for each data discretized with different discretization algorithms in comparison with the multivariate discretization algorithm (MVD) to facilitate the user to choose the algorithm that gives the highest accuracy.

Case Study

In this section, we will give a case study that is used to test the accuracy of different discretization algorithms on a dataset to show which algorithm give high accuracy results. The sports player's data set contains two attributes which are weight and height of a player, the class attribute has four values (Judo players, Jockeys, Basket Ball players, Rugby players). The weight and height were measured on 30 players of each class. In total, there are 120 instances. The data set is divided to training and testing, the odd instances for training and even instances for testing. Figure 5 is a scatter plot of the sports player's data set which depicts that the given data set contains outliers.

In our case study, we apply the dataset to a four discretization algorithms as a preprocessing step for the data mining algorithm. These algorithms are Naïve, Entropy, ChiMerge, and Orthogonal Hyperplanes. We have used the Rosetta [50] and Orange [66] software. The results are presented in Table 1 as measures of correctly classified objects from the test data sets, using the different discretization algorithms.

Table 1

Summary of Results from Comparing Different Discretization Algorithms on a Sports Player's Data Set

Data Set/ Algorithm	Naïve Discretization	Entropy Discretization	ChiMerge Discretization	Orthogonal Hyperplanes
Sports Player's	78 %	97 %	89 %	86 %

We can observe that the results tend to favor the Entropy discretization algorithm. Also, the Orthogonal Hyperplanes algorithm seems to give pretty good results in comparison with Naïve algorithm. Finding the best confidence level for ChiMerge

discretization algorithm which produces highest accuracy is difficult, so it can be done by varying this parameter.

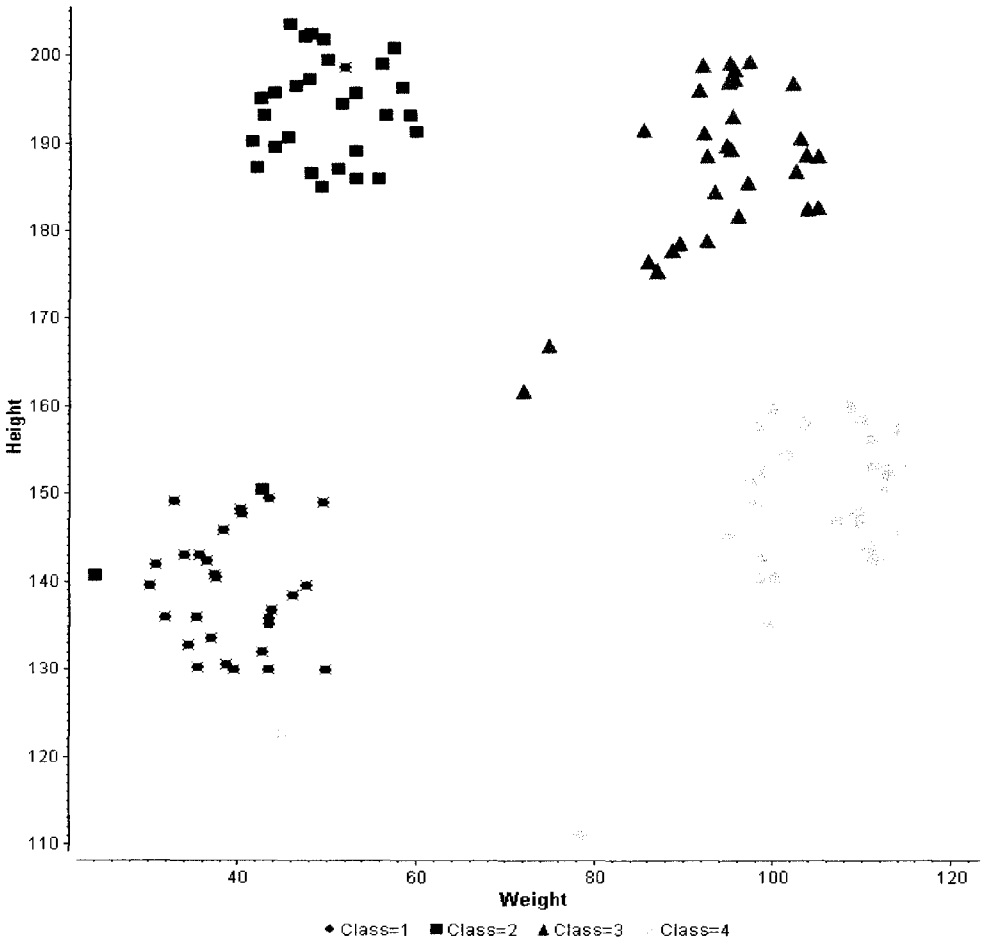

Figure 5: Scatter plot of sports player's data set

Summary

In this chapter, we have given a background of the discretization process. Discretization is a type of data reduction techniques which is part of data preparation process. Data preparation is an important step of the data mining process that influences that result and accuracy of the data mining technique Data reduction techniques can be applied to obtain a reduced representation of the dataset that is much smaller in volume

which maintains the integrity of the original data. Strategies for data reduction include feature discretization, sampling, and feature selection, we will focus only on feature discretization.

Discretization applied only on continuous features, it transforms continuous feature into discrete one. It divides the range of the attribute into intervals. Interval labels can then be used to replace actual data values. There are several discretization algorithms available in the machine learning community. We have chosen five popular algorithms that are Naïve, Entropy, ChiMerge, Orthogonal Hyperplanes, and Multivariate Discretization (MVD) algorithm; we will present these algorithms in details in our literature review. Another data reduction technique is sampling which allows a large dataset to be represented by a much smaller random sample of the data. Feature selection is also a data reduction technique which selects a subset of the original dataset by eliminating redundant and uninformative features.

Data preparation is a field of research in the machine learning community which has high impact on the mining algorithm, discretization is one of these techniques that will deal with continuous data and convert it to discrete one, and this will improve the accuracy of the mining algorithm. We will compare different discretization algorithms on standard numerical data sets.

We have given a case study on a sports player's data set which a measure of weight and height of 120 players, these players belong to 4 classes: Judo, Jockeys, Basket Ball, and Rugby players. We divided these samples into training and testing, odd samples for training and even samples for testing. We apply the data set to four algorithms and

test them with the test samples. We have observed that the Entropy discretization produces better accuracy comparing with the other algorithms.

CHAPTER III

LITERATURE REVIEW

Data Preparation

Data stored in real-world are far from perfect. it needs to be prepared before applying to the data mining technique. In the last few years, there has been significant advancement in data mining techniques. This advancement has not been matched with similar progress in data preparation. Therefore, there is now a strong need for new techniques and automated tools to be designed that can significantly assist us in preparing quality data, Zhang et al. in 2003 [64].

Importance of Data Preparation

In this section, we argue for the importance of data preparation at three aspects.

1. Real-world data may be incomplete, noisy, and inconsistent, which can disguise useful patterns. This is due to:
    - Incomplete data: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data.
    - Noisy data: containing noise or outliers.
    - Inconsistent data: containing inconsistencies in codes or names.
2. Data preparation generated a dataset smaller than the original one, which can significantly improve the efficiency of data mining. This task include:

- Selecting relevant data: attribute selection (filtering and wrapper methods), removing anomalies, or eliminating duplicate records.
- Reducing data: sampling or instance selection.

3. Data preparation generated quality data, which leads to quality patterns. For example, we can:

- Recover incomplete data: filling the values missed, or reducing ambiguity.
- Clean data: correcting errors, or removing outliers (unusual or exceptional values).
- Resolve data conflicts: using domain knowledge or expert decision to resolve discrepancy.

From the above three observations, it can be understood that data preprocessing, cleaning, and preparation is not a small task. Researchers and practitioners must intensify efforts to develop appropriate techniques for Data Preparation efficiently utilizing and managing the data, while data mining technology can support the data analysis applications within these organization, it must be possible to prepare quality data from the raw data to enable efficient and quality knowledge discovery from the data given. Thus, the development of data-preparation technologies and methodologies is both a challenging and critical task.

Data Preparation Steps

Zhang et al. in 2003 [64] defines data preparation as a complicated task that comprises many steps. The following are the steps of the data preparation process.

- Convert data to desired format

The purpose of this step is to prepare and convert data from any format to a data mining ready format.

- Dealing with Missing Data

Missing values is a serious problem. So, there are several ways to deal with missing values by ignoring this record or field, fill in the missing values manually, and use a global constant to fill in missing values, use the attribute mean to fill in the missing value, or use the most probable value to fill in the missing value.

- Data transformation:

In data transformation, the data are transformed or consolidated into forms appropriate for mining; data transformation can involve the following:

  - Smoothing

  This works to remove the noise from data. Such techniques include binning, clustering, and regression. Smoothing is a form of data cleaning.

  - Aggregation:

  Where summary or aggregation operations are applied to the data, For example, the daily sales data may be aggregated so as to compute monthly and annual total amounts.

  - Normalization

  Where the attribute data are scaled so as to fall within a small specified range, such as -1.0 to 1.0, or 0.0 to 1.0.

  - Generalization

In generalization, low-level or raw data are replaced by higher-level concepts through the use of concept hierarchies. For example, categorical attributes, like street, can be generalized to higher-level concepts like city or country. Similarly, values for numeric attributes, like age, may be mapped to higher-level concepts, like young, middle-aged, and senior.

- Data Reduction

Datasets that comes from data warehouses usually has huge amounts, by applying data analysis and mining tools on this huge amount of data may take a very long time, making such analysis impractical or infeasible. Data reduction techniques can be applied to obtain a reduced representation of the dataset that is much smaller in volume, yet closely maintains the integrity of the original data. Mining on the reduced data set should be more efficient. Strategies for data reduction include: (1) feature Discretization, (2) sampling, (3) feature selection.

We will take the three steps in more details in the next subsequent sections.

- Feature composition

It is common for multiple data fields in a data source to be transformed into just one data field in the data mining repository. Suppose, for example, that the source repository stores the following fields system version=2, system release=15. The data analyst may want to concatenate those two fields in a new field version release=2.15 in the data mining repository.

Discretization

Yang in 2003 [63] defines discretization as a data processing procedure. It transforms an attribute (feature) from one type into another type. It divides the range of the attribute into intervals. Interval labels can then be used to replace actual data values. In the large amount of existing literature that addresses discretization, there is considerable variation in the terminology used to describe these two data types, including "quantitative" vs. "qualitative", "continuous" vs. "discrete", "ordinal" vs. "nominal", and "numeric" vs. "categorical".

The objects of analysis could be persons, salaries, opinions, software entities and many others. These objects must be carefully presented in terms of their characteristics. These characteristics are the main variables of the problem and their choice greatly influences the results of the algorithm

There are two parallel ways to classify data into different types. Data can be classified into either qualitative or quantitative. Data can also be classified into different levels of measurement scales.

Qualitative vs. Quantitative Attributes

Attributes can be classified as either qualitative or quantitative. Qualitative attributes, also often referred to as categorical attributes, are attributes that can be placed into distinct categories, according to some characteristics. Qualitative attributes sometimes can be arrayed in a meaningful rank order. But no arithmetic operations can be applied to them. Examples of qualitative attributes are:

- Blood type of a person: A, B, AB, O.
- Sex of a fish: male, female.

- Student evaluation: fail, pass, good, excellent.
- Size of a bottle: Big, medium, small.

Quantitative attributes are numerical in nature. They can be ranked in order. They can also have meaningful arithmetic operations. Quantitative attributes can be further classified into two groups, discrete or continuous.

A discrete attribute assumes values that can be counted. The attribute cannot assume all values on the number line within its value range. Examples of discrete attributes are:

- Number of children in a family.
- Number of houses in a street.

A continuous attribute can assume all values on the number line with the value range. The values are obtained by measuring. Examples of continuous attributes are:

- Temperature.
- Weight of a person.

Levels of Measurement Scales

In addition to being classified as either qualitative or quantitative, attributes can also be classified by how they are categorized, counted or measured. This type of classification uses measurement scales, and four common types of scales are used: nominal, ordinal, interval and ratio.

The nominal level of measurement scales classifies data into mutually exclusive (non-overlapping), exhaustive categories in which no order or ranking can be imposed on the data. Examples of nominal attributes are:

- Blood type of a person: A, B, AB, O.

- Sex of a fish: male, female.

The ordinal level of a measurement scales classifies data into categories that can be ranked. However, the differences between the ranks cannot be calculated by arithmetic. Examples of the ordinal attributes are:

- Student evaluation: fail, pass, good, excellent.
- Size of a bottle: Big, medium, small.

It is meaningful to say that the student evaluation of pass ranks higher than that of fail. It is not meaningful in the same way to say that the blood type of A ranks higher than that of B.

The interval level of measurement scales ranks data and the differences between units of measure can be calculated by arithmetic. However, zero in the interval level of measurement does not mean "nil" or "nothing" as zero in arithmetic means. Examples of interval attributes are:

- IQ, whose values are yielded by a standardized psychological test, there is a meaningful difference of one point between an IQ of 109 and an IQ of 110. But IQ tests do not measure people who have no intelligence.
- Fahrenheit temperature, there a meaningful difference of one degree between each unit, such as 72 degrees and 73 degrees. But 0 degrees Fahrenheit does not mean any heat.

It is meaningful to say that the IQ of a person A is two points higher than that of person B. It is not meaningful in the same way to say that the tenderness of piece of beef A is two points higher than the tenderness of piece B.

The ratio level of measurement scales possesses all the characteristics of interval measurement, and there exists a zero that, the same as arithmetic zero, means "nil" or "nothing". In consequence, true ratios exist between different units of measure. Examples of ratio attributes are:

- Number of children in a family.
- Weight of a baby.

It is meaningful to say that the weight of a child A is twice that of child B. It is not meaningful in the same way to say that the IQ of person A is twice that of person B.

The nominal level is the lowest level of measurement scales. It is the least powerful in terms of including data information. The ordinal level is higher.

The interval level is even higher. The ratio level is the highest level. Any data conversion from a higher level of measurement scales to a lower level of measurement scales will lose information.

In summary, the following classification applies to attribute types:

1. qualitative attributes:
   - Nominal.
   - Ordinal.
2. quantitative attributes:
   - Interval, either discrete or continuous.
   - Ratio, either discrete or continuous.

We must note that some of the attribute types could belong two more than one category. For example, color is usually considered as nominal. However, we all know

that for each color there is a point on the spectrum line for it, making it an ordinal-scaled attribute.

We believe that "discretization" as it is usually applied in machine learning is best defined as the conversion of quantitative attributes to qualitative attributes. In consequence, we will refer to attributes as either quantitative or qualitative throughout this thesis.

Another term often used for describing discretization is "cut-point". When discretizing a quantitative attribute, a cut-point is a value of the attribute where an interval boundary is located by a discretization method.

Classification of Discretization Methods

From our literature we found that there exists a diverse classification to classify discretization algorithms. Different classification emphasizes different aspects of the distinctions among discretization algorithms.

Typically, discretization methods can be classified into either primary or composite. Primary methods accomplish discretization without reference to any other discretization method. Composite methods are built on top of a primary method. Primary methods can be classified as per the following classifications.

- Supervised vs. Unsupervised

  Dougherty et al. in 1995 [11] conducted an empirical evaluation of several algorithms for discretization of continuous attributes. They defined that algorithms that use the class information of the training instances to select discretization cut points are supervised. Algorithms that do not use the class information are unsupervised, and showed that discretization prior to induction

can sometimes significantly improve the accuracy of the induction algorithm. The global entropy-based discretization method seems to be the best choice of the discretization methods tested.

The entropy-discretized Naïve-Bayes improved so much, that its average performance slightly surpassed that of C4.5. C4.5's performance did not degrade if data were discretized in advance using the entropy discretization method, and in two cases even improved significantly.

Supervised discretization can be further characterized as error-based, entropy-based or statistics-based according to whether intervals are selected using metrics based on error on the training data, entropy of the intervals, or some statistical measure.

Equal-width discretization (EWD) and Equal-frequency discretization (EFD) are considered unsupervised discretization algorithms. Naïve, Entropy-based discretization, and Holte's are considered supervised discretization algorithms.

- Univariate vs. Multivariate

Algorithms that discretize each attribute in isolation at a time are univariate. Algorithms that take into consideration relationships among attributes during discretization are multivariate.

Divina et al. in 1999 [10] noticed that univariate discretization methods are not able to capture the interdependencies among attributes. For this reason globally discretizing numerical attributes with such methods yields the risk of

missing the necessary information in order to find the correct solution. Fayyad an Irani's algorithm is an example of univariate discretization algorithms.

Liu et al. 1999 [33] defined univariate and multivariate discretization algorithms, that univariate discretization quantify one continuous feature at a time while multivariate discretization considers simultaneously multiple features.

Bay in 2000 [2] presented a bottom up merging algorithm to discretize continuous variables based on multivariate discretization algorithm that finely partitions continuous variables and then merges adjacent intervals only if their instances have similar multivariate distributions. Merging allows them to mine an appropriate resolution to quantize the data. Their multivariate test ensures that only similar distributions are joined, and show patterns unlike univariate algorithms which destroy hidden patterns in data.

Experimental results on synthetic data indicate that the algorithm can detect high dimensional interactions between features and discretize the data appropriately. On real data the algorithm ran in time comparable to a popular univariate recursive approach and produced sensible discretization cut-points.
Most common discretization methods can be assumed univariate discretization algorithms.

Chao et al. in 2005 [8] proposed a multivariate interdependent discretization of continuous attributes (MIDCA) that incorporate the normalized relief and information measures to look for the best correlated attribute with respect to each continues-value attribute being discretized, and using the

discovered best correlated attribute as the interdependent attribute to carry out the multivariate discretization.

Monti†, et al. in 1999 [38] proposed a latent variable model as a tool to extend the scope of applicability of machine learning algorithms that handle discrete variables only.

- Parametric vs. Non-parametric

Parametric discretization algorithms require input from the user, such as the maximum number of discretized intervals. Non-parametric discretization algorithms only use information from data and do not need input from the user.

- Hierarchical vs. Non-hierarchical

Hierarchical discretization algorithms select cut points in an incremental process, forming an implicit hierarchy over the value range. The procedure can be split or (and) merge.

Kerber in 1992 [29] argued that in an accurate discretization, the relative class frequencies should be fairly consistent within an interval (otherwise the interval should be split to express this difference) but two adjacent intervals should not have similar relative class frequency (in that case the adjacent intervals should be merged into one). It tests the hypothesis that two adjacent intervals of a feature are independent of the class. If they are independent, they should be merged, otherwise they should remain separate. Split discretization initially has the whole value range as an interval, and then continues splitting it into sub-intervals until some threshold is met. Merge discretization initially puts each

value into an interval, and then continues merging adjacent intervals until some threshold is met.

Some discretization methods utilize both split and merge processes. For example, intervals are initially formed by splitting, and then a merge process is performed to post-process the formed intervals. Non-hierarchical discretization does not form any hierarchy during discretization. For example, many methods scan the ordered values only once, sequentially forming the intervals.

Liu et al. in 1999 [33] consider Equal-width discretization (EWD), Equal-frequency discretization (EFD), and Entropy-minimum description length principle (MDLP) discretization algorithm as hierarchical discretization algorithms, where ChiMerge and Chi2 are considered non-hierarchical discretization algorithms.

- Global vs. Local

Dougherty et al. in 1995 [11] proposed that global algorithms discretize with respect to the whole training data space. They perform discretization once only, using a single set of intervals throughout a single classification task. Local algorithms allow different sets of intervals to be formed for a single attribute, each set being applied in a different classification context. For example, different discretizations of a single attribute might be applied at different nodes of a decision tree, Quinlan in 1993 [48].

Dougherty et al. in 1995 [11], define that, local algorithms produce partitions that are applied to localized regions of the instance space (i.e. subset of instances)

Chmielewski et al. in 1994 [9] defines that methods of discretization restricted to a single continuous attributes will be called local while methods that simultaneously convert all continuous attributes will be called global. Global algorithms such as binning produce a mesh over the entire n-dimensional continuous instance space, where each feature is partitioned into regions independent of the other attributes. Global discretization algorithms use the entire instance space to discretize.

Liu et al. in 1999 [33] ID3, D2, and Entropy-minimum description length principle (MDLP) discretization algorithm are considered local discretization algorithms, while Zeta, ChiMerge, and Chi2 are considered global algorithms.

- Eager vs. Lazy

Eager methods perform discretization prior to classification time. Lazy methods perform discretization during the classification time.

Hsu et al. in 2000 [24] & Hsu et al. in 2003 [25] presented a lazy discretization method, which is derived directly from the properties of dirichlet distributions. This method waits until one or more test data are given to determine the cut points for each continuous variable. They showed that lazy method works equally well compared to well-known discretization methods. These results justify the selection of Dirichlet priors and verify their analysis. They also extend the method to allow a Naive Bayesian classifier to classify set and interval data.

- Disjoint vs. Non-disjoint.

Disjoint algorithms discretize the value range of the attribute under discretization into disjoint intervals. No intervals overlap. Non-disjoint algorithms discretize the value range into intervals that can overlap.

Yang et al. in 2002 [62] notice that non-disjoint discretization (NDD) algorithms forms overlapping for a numeric attribute x_i, always locating a value xi toward the middle of its corresponding interval. The idea behind NDD is that when substituting (a_i, b_i) for x_i, there is more distinguishing information about x_i, and thus the probability estimation is more reliable if x_i is in the middle of (a_i, b_i) than if xi is close to either the boundary. NDD bases its interval size strategy on Proportional k-Interval Discretization (PKID)

- Dynamic vs. Static

Liu et al. in 1998 [34] proposed that discretizing features while inducing a decision tree is called dynamic discretization algorithms, while discretizing it before tree induction is called static discretization algorithms.

Dougherty at al. in 1995 [11] produced a comparison between static and dynamic discretization algorithms, where defined static methods, such as binning, entropy-based partitioning, in Catlett 1991b [6], Fayyad et al. in 1993 [12], & Pfahringer in 1995 [45], and the IR algorithm by Holte in 1993 [23] perform one discretization pass of the data for each feature and determine the value of k for each feature independent of the other features. While dynamic methods conduct a search through the space of possible k values for all features simultaneously, thereby capturing interdependencies in feature discretization.

Quinlan in 1993 [48] defined that dynamic algorithms would discretize continuous values when a classifier is being built, such as in C4.5, while in the static algorithms discretization is done prior to the classification task.

Composite methods first choose some primary discretization methods to form the initial cut-points. They then focus on how to adjust these initial cut points to achieve certain goals. The classification of a composite method sometimes is flexible, depending on the classification of its primary method.

Discretization Process

Liu et al. in 1999 [33] clarified some terms used in different works followed by an abstract description of a typical discretization process.

Basic Terms

- Feature

"Feature" or "Attribute" or "Variable" refers to an aspect of the data. Usually before collecting data, features are specified or chosen. Features can be discrete, continuous, or nominal. Hereafter M stands for the number of features in the data.

- Instance

"Instance" or "Tuple" or "Record" or "Data Point" refers to a single collection refers to a single collection of feature values for all features. A set of instances makes a data set. Usually a data set is in a matrix form where a row corresponds to an instance and a column corresponds to a feature. Hereafter N is the number of instances in the data.

- Cut-point

The term "cut-point" refers to a real value within the range of continuous values that divides the range into two intervals, one interval is less than or equal to the cut-point and the other interval is greater than the cut-point. For example, a continuous interval [a, b] is partitioned into [a, c] and [c, b], where the value c is a cut-point. Cut-point is also known as split-point.

- Arity

The term "arity" in the discretization context means the number of intervals or partitions. Before discretization of a continuous feature, arity can be set to k-the number of partitions in the continuous features. The maximum number of cut-points is $k-_1$. Discretization process reduces the arity but there is a trade-off between arity and its effect on the accuracy of classification and other tasks. A higher arity can make the understanding of an attribute more difficult while a very low arity may affect predictive accuracy negatively.

A Typical Discretization Process

Liu et al. in 1999 [33] by "typical" we mean univariate discretization, a typical discretization process consists of four steps (seen in figure 3.1): (1) sorting the continuous values of the feature to be discretized, (2) evaluating the cut-point for splitting or adjacent intervals for merging, (3) according to some criterion, splitting or merging intervals of continuous values, and (4) finally stopping at some point in the following we will discuss these four steps in more detail.

- Sorting

The continuous value for a feature is sorted in either descending or ascending order. Sorting can be computationally very expensive if care is not taken in implementing it with discretization. It is important to speed up the discretization process by selecting suitable sorting algorithms. Many sorting algorithms can be found in classic data structures and algorithms books.

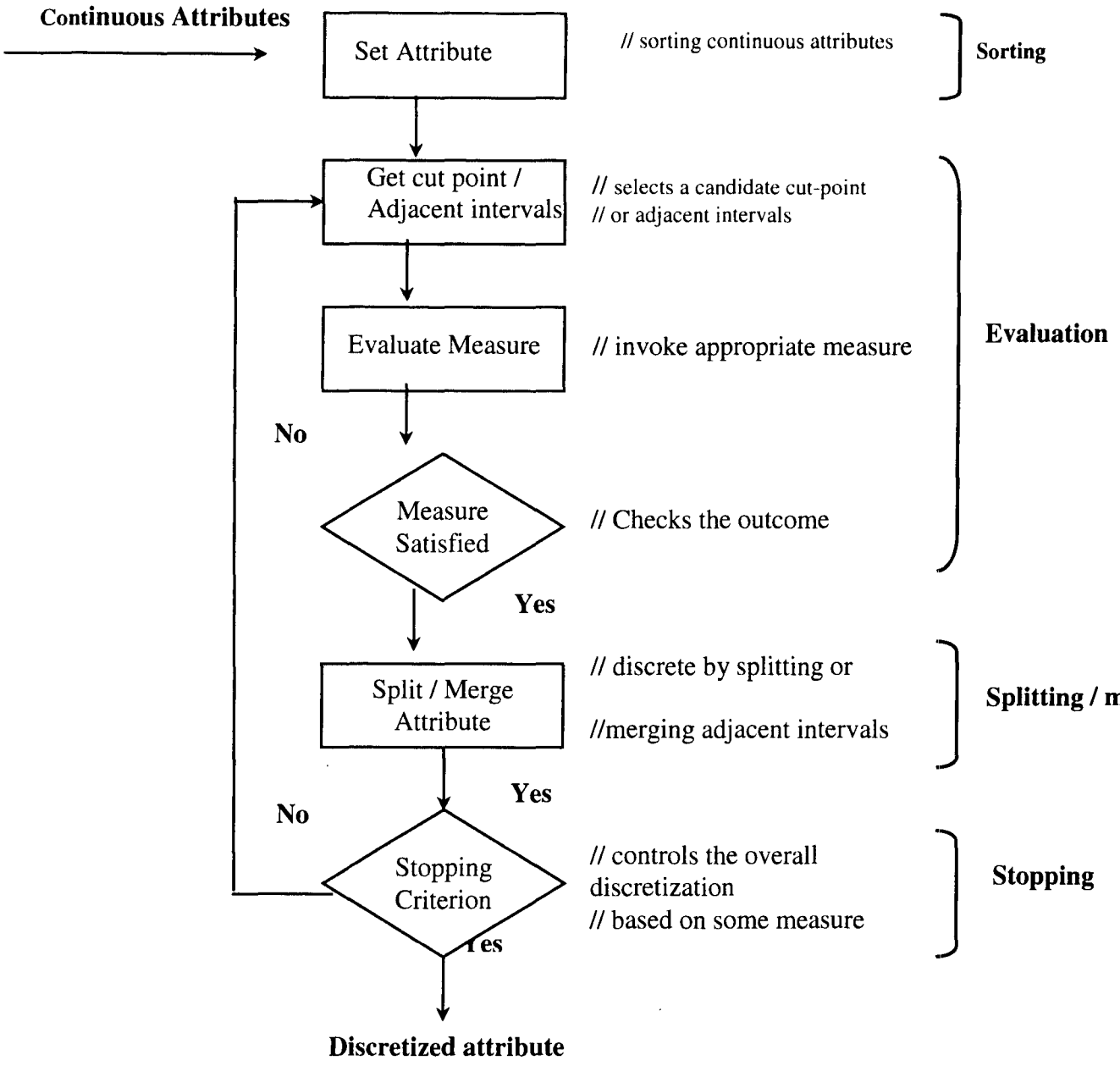

Figure 6: Discretization process

Among these "Quick-sort" is an efficient sorting algorithm with a time complexity of O(N log N).

Another way to improve efficiency is to avoid sorting a feature's values repeatedly. If sorting is done once and for all at the beginning of discretization, it

is a global treatment and can be applied when the entire instance space is used for discretization. If sorting is done at each iteration of a process, it is a local treatment in which only a region of entire instance space is considered for discretization.

- Choosing a cut-point

After sorting, the next step in the discretization process is to find the best "cut-point" to spit a range of continuous values or the best pair of adjacent intervals to merge. One typical evaluation function is to determine the correlation of a split or a merge with the class label. There are numerous evaluation functions found in the literature such as entropy measures and statistical measures. More about these evaluation functions and their various applications are discussed in the following sections.

- Splitting/merging

As we know, in a top-down approach, intervals are split while for a bottom-up approach intervals are merged. For splitting it is required to evaluate "cut-points" and to choose the best one and split the range of continuous values into two partitions. Discretization continues with each part (increased by one) until a stopping criterion is satisfied. Similarly for merging, adjacent intervals are evaluated to find the best pair of intervals to merge in each iteration. Discretization continues with the reduced number (decreased by one) of intervals until the stopping criterion is satisfied.

- Stopping criteria

A stopping criterion specifies when to stop the discretization process. When to stop the discretization process. It is usually governed by a trade-off between lower arity with a better understanding but less accuracy and a higher arity with a poorer understanding but higher accuracy. We may consider k to be an upper bound for the arity of the resulting discretization. In practice the upper bound k is set much less than N, assuming there is no repetition of continuous value for a feature. A stopping criterion can be very simple such as fixing the number of intervals at the beginning or a more complex one like evaluating a function. We describe different stopping criteria in the next section.

Result Evaluation for Discretization

Assuming we have many methods Liu et al. in 1999 [33], and each provides some kind of discretized data, which discretized data is the best? This seemingly simple question cannot be easily dealt with a simple answer. This is because the result evaluation is a complex issue and depends on a users need in a particular application. It is complex because the evaluation can be done in many ways. We list three important dimensions:

- The total numbers of intervals-intuitively, the fewer the cut-points, the better the discretization result; but there is a limit imposed by the data representation. This leads to the next dimension.
- The number of inconsistencies, caused by discretization, it should not be much higher than the number of inconsistencies of the original data before discretization. If the ultimate goal is to generate a classifier from the data, we should consider yet another perspective.

- Predictive accuracy, how discretization helps improve accuracy. In short, we need at least three dimensions: simplicity, consistency, and accuracy. Ideally, the best discretization result can score highest in all three departments. In reality, it may not be achievable, or necessary to achieve these goals.

Simplicity is defined by the total number of cut-points. Accuracy is usually obtained by running classifier in cross validation mode. Consistency is defined by having the least inconsistency count which is calculated in three steps: (in the following description a pattern is a set of values for a feature set while an instance is a pattern with a class label) (1) two instances are considered inconsistent if they are the same in their attribute values except for their class labels, for example, we have an inconsistency if there are two instances (0 1, a) and (0 1, ¯a) class label is separated by "," because of different classes a and ¯a. (2) the inconsistency count for a pattern is the number of times it appears in the data minus the largest number of class label: for example, let us assume there are n instances that match the pattern, among them, c1 instances belong to label1, c2 to label2, and c3 to label3 where c1 + c2 + c3 = n. If c3 is the largest among the three, the inconsistency count is (n - c3). (3) The total inconsistency count is the sum of all the inconsistency counts for all possible patterns of a feature subset.

Discretization Framework

There are several discretization algorithms available in the literature, these methods can be categorized in several dimensions as discussed earlier. We restate them here: supervised vs. unsupervised, univariate vs. multivariate, parametric vs. non-parametric, hierarchical vs. non-hierarchical, global vs. local, eager vs. lazy, disjoint vs. non-disjoint, and dynamic vs. static. One can construct different combinations of these

dimensions to group the methods. But arbitrary combinations will not help in advancing the research of discretization. Liu et al. in 1999 [33] wish to create a hierarchical framework that is systematic and expandable, and attempts to cover all existing methods. Each discretization method found in the literature discretized a feature by either splitting the interval of continuous values or by merging the adjacent intervals. Both splitting and merging categories can further be grouped as supervised or unsupervised depending on whether class information is used. To repeat, supervised discretization methods use the available class information while the unsupervised methods do not.

Liu et al. in 1999 [33] proposed a hierarchical framework in figure 3.2. We describe different discretization measures according to two approaches: splitting and merging (level 1). We then consider whether a method is supervised or unsupervised (level 2). We further group together methods that use similar discretization measures (level 3) e.g. binning and entropy. As is suggested in figure 3.2, the supervised and unsupervised division determines different (non-overlapping) measures used. Hence this conceptually useful division will not be discussed in detail below. We may find variations of these measures like Mantaras distance in Cerquides et al. in 1997 [7] which we categorize under entropy measure for their similarity.

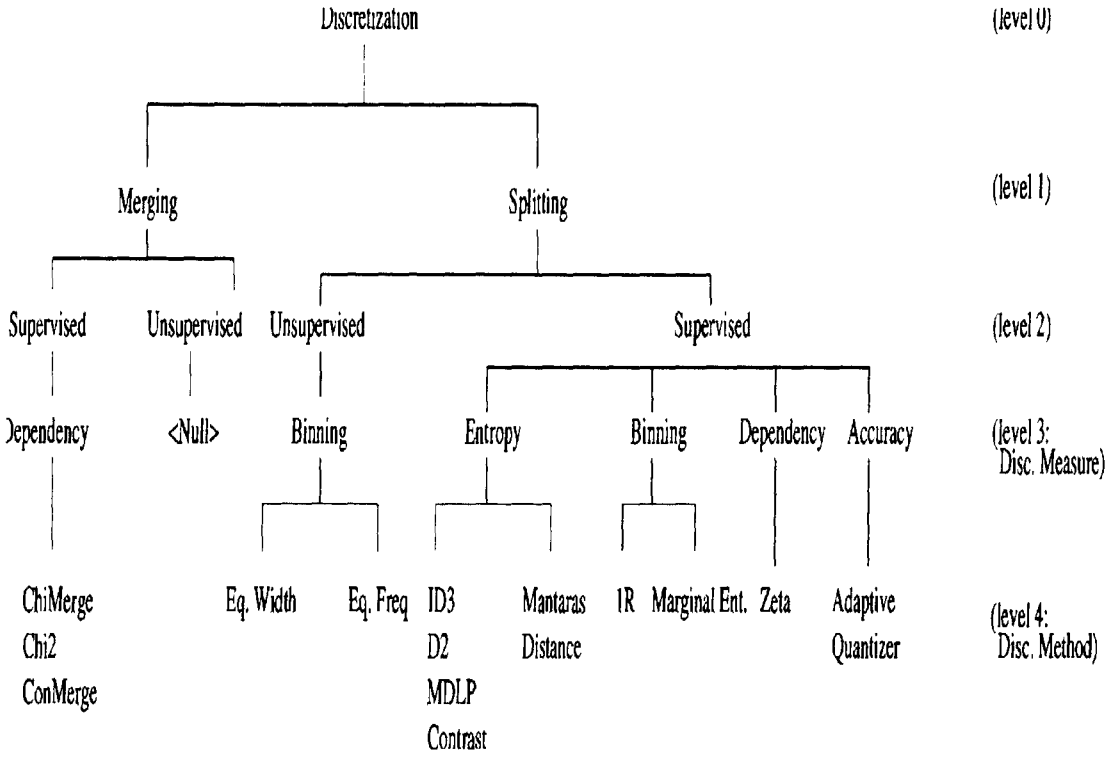

Figure 7: A hierarchical framework for discretization methods

Sampling

Sampling used as a data reduction technique since it allows a large data set to be represented by a much smaller random sample of the data. Suppose that a large data set, D, contains N records, the possible sampling techniques for D is: by Han et al. in 2001 [19]. There sampling techniques are as follows:

Simple random sample without replacement (SWSWOR) of size n.

- This is created by drawing n of the N records from D (n < N), where the probability of drawing any records in D is 1/N, that is all records are equal likely.

Simple random sample with replacement (SRSWR) on size n

This is similar to SRSWOR; except that each time a record is drawn from D. it is recorded and then replaced. That is, after a record is drawn, it is placed back in D so that it may be drawn again.

- Cluster sample

If the records in D are grouped into M mutually disjoint clusters, then a Simple Random Sampling (SRS) of m clusters can be obtained, where m < M. for example records in a database are usually retrieved a page at a time, so that each page can be considered a cluster. A reduct data representation can be obtained by applying, say, SRSWOR to the pages, resulting in a cluster sample of the records.

- Stratified sample

If D is divided into mutually disjoint parts called strata, a stratified sample of D is generated by obtaining an SRS at each stratum. This helps to ensure a representative sample, especially when the data are skewed. For example, a stratified sample may be obtained from customer data, where a stratum is created for each customer each group. In this way, the age group having the smallest of customers will be sure to represent.

An advantage of sampling for data reduction is that the cost of obtaining a sample is proportional to the size of the sample, n, as opposed to N, the data set size. Hence, sampling complexity is potentially sublinear to the size of the data.

Feature Selection

Feature selection is the process of choosing a subset of the original predictive variables by eliminating redundant and uninformative ones. By extracting as much as information as possible from a given dataset while using the smallest number of features,

we can save significant computing time and often build models that generalize better to unseen points. Further, it is often the case that finding a predictive subset of input variables is an important problem in its own right.

Feature selection is important for several reasons, the fundamental one being arguably that noisy features can degrade the performance of most learning algorithms. In supervised learning, it is known that feature selection can improve the performance of classifiers learned from limited amounts of data, it leads to more economical classifiers and in many cases, it may lead to interpretable models. It is particularly important for datasets with large numbers of features; e.g., classification problems in molecular biology may involve thousands of features, and a web-page can be represented by thousands of different key-terms. Appearance-based image classification methods may use each pixel as a feature, thus easily involving thousands of features.

The selection criteria of a feature can be done from measuring perspective, there exist many measures that can be used to select features, such that information measures, distance measures, dependence measures, consistency measures, and accuracy measures. Selecting features by using one of these measures will lead to reduce dimensionality and remove noise, improve the accuracy of the classifier.

Feature selection has been widely studied in the context of supervised learning, where the ultimate goal is to select features that can achieve the highest accuracy on unseen data. It has received comparatively very little attention in unsupervised learning or clustering. One important reason is that it is not at all clear how to assess the relevance of a subset of features without resorting to class labels. The problem is made even more challenging when the number of clusters is unknown, since the optimal number of

clusters and the optimal feature subset are inter-related. Methods based on variance (such as principal components analysis) need not select good features for clustering, as features with large variance can be independent of the intrinsic grouping of the data. Feature selection can be carried out by an Expectation Maximization (EM) algorithm.

Review of Discretization Methods

There are several methods that have been proposed to discretize data in the machine learning community; in this section we will present a comprehensive literature review on existing discretization methods in the research area of machine learning.

Many real-world data tackled by machine learning algorithms involve quantitative data. However, Kerber in 1992 [29], & Dougherty et al. in 1995 [11], Kohavi et al. in 1996 [31] proposed that there exist many machine learning algorithms that are more oriented to handle qualitative attributes than quantitative attributes. Catlett in 1991b [6], Kerber in 1992 [29], Richard et al. in 1995 [49], Frank et al. in 1999 [16].

Even for algorithms that can directly deal with quantitative attributes, learning is often less efficient and less effective for quantitative data than for qualitative data. Since larger and larger datasets are becoming routinely available for most modern research, the learning efficiency is of particular importance. Thus discretization has attracted much attention.

Over the years, many discretization algorithms have been proposed and tested to show that discretization helps improve the performance of learning and helps understand the learning result. In this section we will look into several methods for discretizing data, with the main focus on the most common discretization methods.

Equal Width Discretization (EWD) Algorithm

Dougherty et al. in 1995 [11], Risvik in 1997 [52] when discretizing a quantitative attribute equal width discretization is the simplest approach to discretization, this method discretize data and has been applied as a means for producing nominal values from continuous ones. It involves sorting the observed values of a continuous feature and dividing the range of observed values for the variable into k equally sized bins, where k is a parameter supplied by the user. If a variable x is observed to have values bounded by x_{min} and x_{max} then this method computes the bin width.

$$\delta = \frac{x_{max} - x_{min}}{k} \qquad \text{(eq. 1)}$$

And constructs in boundaries, or thresholds, at $x_{min} + i\delta$ where $i = 1 \ldots k-1$. The method is applied to each continuous feature independently. It makes no use of instance class information whatsoever and is thus an unsupervised discretization method.

Risvik in 1997 [52] concluded that the Equal Width Discretization (EWD) assumes that the underlying data fits somewhat well into a uniform distribution. It is very sensible to outliers, and can perform extremely bad under some conditions.

Equal Frequency Discretization (EFD) Algorithm

Yang et al. in 2002 [62] Equal Frequency Discretization (EFD) algorithm divides the sorted values into k intervals so that each interval contains approximately the same number of training instances. Thus each interval contains n/k (possibly duplicated) adjacent values. K is a user predefined parameter. Note that the training with identical values must be placed in the same interval. In consequence it is not always possible to generate k-equal frequency intervals.

Both Equal Width Discretization (EWD) and Equal Frequency Discretization (EFD) potentially suffer much attribute information loss since k is determined without reference to the properties of the training data. But although they may be considered simplistic, both the two methods are unsupervised discretization methods and will not take the decision attributes into consideration, therefore classification maybe difficult in several cases.

Hsu, et al. in 2000 [24], Hsu et al. in 2003 [25] get the results that both Equal Width Discretization (EWD) and Equal Frequency Discretization (EFD) are often used for Naïve-Bayes classifiers because their simplicity and effectiveness.

Yang in 2003 [63] proposed that both Equal Width Discretization (EWD) and Equal Frequency Discretization (EFD) fix the number of intervals to be produced (decided by the user predefined parameter k). When the training dataset is very small, intervals tend to have small frequency and thus tend to incur high discretization variance. When the training data size is very large, intervals tend to have large frequency and thus tend to incur very high discretization bias. Thus we anticipate that they do not control either discretization bias or discretization variance well.

A Naïve Discretization Algorithm

Risvik in 1997 [52] proposed a very simple discretization algorithm. Talking the decision into consideration, then sorting the objects according to the value of the attribute being considered, we can "walk" along the axis of the attribute, creating a new interval whenever the decision value is changing. This algorithm yields a set of all required cuts to keep the consistency level. (Will be defined precisely later in next section) in the information system constant.

1. Select attribute a $\in$ A.
2. Set Ca = Ø.
3. Sort values of the attribute.
4. For all adjacent pair of objects if Va(i) $\neq$ Va(i+1), and d(i) $\neq$ d(i+1), create new cut c = (Va(i) + Va(i+1))/2, and Ca = Ca $\cup$ c.
5. Repeat 2-4 for all continuous attributes a in A.

This algorithm is bounded by the sorting step, therefore the computational complexity is O(k N log N), where k is the number of continuous attributes, and N is the number of objects in the decision system.

ChiMerge Discretization Algorithm

Kerber in 1992 [29] was first proposed ChiMerge method to provide a statistically justified heuristic method for supervised discretization. The ChiMerge algorithm consists of an initialization step (i.e., placing each observed real value into its own interval) and proceeds by using the χ^2 test to determine when adjacent intervals should be merged. This bottom-up merging process is repeated until a stopping criterion (set manually) is met. Here, the χ^2 test is a statistical measure used to test the hypothesis that two discrete attributes are statistically independent. Applied to the discretization problem, it tests the hypothesis that a class attribute is independent of the two adjacent intervals an example belongs to. If the conclusion of the χ^2 is that the class is independent of the intervals, then the intervals should be merged. On the other hand, if the χ^2 test concludes that they a rent independent, it indicates that the difference in relative class frequencies is statistically significant, and therefore the intervals should be remain separate. The formula for computing the χ^2 value is:

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \qquad \text{(eq. 2)}$$

Where:

k = number of classes,

A_{ij} = number of patterns in the interval,

R_i = number of patterns in the ith interval = $\sum_{j=1}^{K} A_{ij}$,

C_j = number of patterns in the $_j$th class = $\sum_{i=1}^{2} A_{ij}$,

N = total number of patterns = $\sum_{i=1}^{2} R_i$,

E_{ij} = expected frequency of A_{ij}; $E_{ij} = R_i * C_j / N$.

The value of the χ^2 threshold is determined by selecting a desired significance level α and then using a table or formula to obtain the corresponding χ^2 value. Obtaining the χ^2 value also requires specifying the number of degrees of freedom, which is one less than the number of classes. For example, when there are three classes, the degree of freedom is 2, the χ^2 value at $\alpha = 0.1$ level is 4.6. The meaning of this threshold is that among the cases where the class and attribute are independent, there is a 90 percent probability that the computed χ^2 value will be less than 4.6. The two adjacent intervals with the lowest χ^2 values will merge together. This step is repeated continuously until all χ^2 value exceeding a predefined threshold.

Immediate complexity is $O(N^2)$, but with some optimization, the complexity can be reduced to O(N log N). N is the number of examples in the dataset being discretized. ChiMerge algorithm classified as a supervised method, but also can be classified as a dynamic method.

Chi2 Discretization Algorithm

Liu et al. in 1995 [35] used the ChiMerge algorithm as a basis for their Chi2 algorithm. Specifying a proper value for α in many cases can be difficult. It would therefore be ideal if α can be determined from the data itself. Tay et al. in 2002 [57] notice that the Chi2 algorithm enhanced the ChiMerge algorithm in that the value of α was calculated based on the training data itself. The calculated value of α differed from attribute to attribute, so that it would only continue merging intervals on those attributes which needed it.

Liu et al. in 1995 [35] divides the algorithm into two phases. In the first phase begins with high significance level (sigLevel), e.g., 0.5, for all numeric attributes for discretization. Each attribute is sorted according to its values. Then the following is performed:

1. Calculate the χ^2 value (as in equation (2)) for every pair of adjacent intervals (at the beginning, each pattern is put into its own interval that contains only one value of an attribute).
2. Merge the pair of adjacent intervals with the lowest χ^2 value. Merging continuous until all pairs of intervals have χ^2 values exceeding the parameter determined by sigLevel (initially, 0.5, its corresponding χ^2 value is 0.455 if the degree of freedom is 1, more below).

The above process is repeated with a decreased sigLevel until an inconsistency rate, δ is exceeded in the discretized data. Phase 1 is, as a matter of fact, a generalized version of ChiMerge of Kerber [29]. Instead of specifying a χ^2 threshold, Chi2 wraps up ChiMerge with a loop that increment the χ^2 threshold (decrementing sigLevel). A

consistency checking is also introduced as a stopping criterion in order to guarantee that the discretized dataset accurately represents the original one. With these two new features, Chi2 automatically determines a proper χ^2 threshold that keeps the fidelity of the original data.

The phase 2 is a finer process of phase 1, starting with sigLevel0 determined in phase 1, each attribute i is associated with a sigLevel[i], and takes turns for merging, consistency checking is conducted after each attribute's merging. If the inconsistency rate is not exceeded, sigLevel[i] is decremented for attribute i's next round of merging, otherwise attribute i will not be involved in further merging. This process is continued until no attribute's value can be merged. At the end of phase 2, is an attribute is merged to only one value; it simply means that this attribute is not relevant in representing the original data set. As a result when discretization ends, feature selection is accomplished.

The Chi2 Algorithm Phases:

Phase 1:

```
Set α = 0.5;
do while (InConCheck (data) < δ)
{ For each numeric attribute
  {   Sort (attribute, data);              /* Sort data on Attribute*/
      Ch-sq-init (att, data);
      do    {Ch-sq-calculation (att, data);}
            While (Merge(data))
  }
```

```
    α0 = α; α = DecreSigLevel(α);
  }
```

Phase 2:

```
  Set all sigLvL[i] = α0 for attribute I;
  do until no attribute can be merged
  { for each mergable attribute i
    {   Sort(attribute, data);                    /* Sort data on Attribute*\
        Ch-sq-init (att, data);
        do      {Ch-sq-calculation (att, data);}
                while (Merge(data))
        If (InConCheck(data) < δ)
                SigLvl[i] = DecreSigLevel(sigLvL[i]);
        Else attribute i is not mergable;
    }
  }
```

Where:

InConCheck(): calculation of inconsistency rate, it is calculated as follows:

1. Two instances are considered inconsistent if they match, except for their class label.
2. for all matching instances (without considering their class labels), the inconsistency count is the number of the instances minus the largest number of the instances of the class labels, for example, there are n matching instances

among them, c_1 instances belong to $label_1$, c_2 to $label_2$, and c_3 to $label_3$ where $c_1 + c_2 + c_3 = n$. if c_3 is the largest among the three, the inconsistency count is $(n - c_3)$.

3. The inconsistency rate is the sum of all the inconsistency counts divided by the total number of the instances.

DecreSigLevel(): decreasing the significance level by one level;

Merge(): returning true or false depending on whether the concerned attribute is merged or not;

Chi-sq-init(): calculation of the A_{ij}, R_i, C_j, N, E_{ij}, k for the calculation of χ^2 value;

Chi-sq-calculation(): calculation of χ^2value corresponding to (1).

The first phase of this algorithm can be regarded as a generalization of the ChiMerge algorithm. Instead of a predefined significance level α, the Chi2 algorithm provided a wrapping that automatically incremented the threshold (decreasing the significance level α). Consistency checking was utilized as a stopping criterion. These enhancements ensured that the Chi2 algorithm automatically determined a proper threshold while still keeping the fidelity of the original data.

The second phase refines the intervals. If any of the attributes consisting of any of the intervals can be further merged without increasing the inconsistency of the training data above the given limit, then the merging phase was carried out. While the first phase worked on a global significance level α, the second phase used separate local significance levels for each attribute, Risvik in 1997 [52].

Entropy-Based Discretization Algorithm

An information-based measure called entropy can be used to recursively partition the values of a numeric attribute, resulting in a hierarchical discretization. Discretization based on entropy criteria have been discussed in several papers, and will be investigated in this section.

Risvik in 1997 [52] defines the entropy as a measure of the degree of randomness of the distribution of instances over positive and negative classes.

Dougherty et al. in 1995 [11] notice that there are a number of entropy-based methods have recently come to the forefront of work on discretization. Chiu have proposed a hierarchical discretization method based on maximizing the Shannon entropy over the discretized space. This method uses a hill climbing search to find a suitable initial partition of the continuous space into k bins along each axis and then re-applies this method to particular intervals to obtain finer intervals. This method has been applied primarily to an information synthesis task yet it bears strong similarities to work in discretization by machine learning researches.

Catlett in 1991b [6] has explored the use of entropy-based discretization in decision tree domains as a means of achieving an impressive increase in the speed of induction on very large data sets with many continuous features. His 2-D discretizer uses several conditions as criteria for stopping the recursive formation of partitions for each attribute: a minimum number of samples in one partition, a maximum number of partitions, and a minimum information gain.

Fayyad et al. in 1993 [12] use a recursive entropy minimization heuristic for discretization and couple this with a Minimum Description Length (MDL) Criterion by

Rissanen in 1986 [51] to control the number of intervals produced over the continuous space. In the original paper, this method was applied locally at each node during tree generation. The method was found to be quite promising as a global discretization method by Ting in 1994 [58] and in this paper the method is used for global discretization.

Pfahringer in 1995 [45] uses entropy to select a large number of candidate split-points and employs a best-first search with a Minimum Description Length (MDL) heuristic to determine a good discretization.

Catlett in 1991b [6], & Fayyad, et al. in 1993 [12] presented a discretization algorithm for discretizing continuous attributes based on a minimal entropy heuristic. This supervised algorithm uses the class information entropy of candidate partitions to select bin boundaries for discretization. It first sorts all the examples by the attribute being discretized. Our notation closely follows the notation of Fayyad and Irani. If we given a set of instances S, a feature A, and a partition boundary T, the class information entropy of the partition induced by T, denoted E(A;T;S) is given by:

$$E(A,T;S) = \frac{|S1|}{|S|} \text{ Ent } (S1) + \frac{|S2|}{|S|} \text{ Ent } (S2), \qquad \text{(eq. 3)}$$

For a given feature A, the boundary T_{min} which minimizes the entropy function over all possible partition boundaries is selected as a binary discretization boundary. This method can be applied recursively to both of the partitions induced by T_{min} until some stopping condition is achieved, thus creating multiple intervals on the feature A.

Fayyad and Irani make use of the Minimal Description Length Principle (MDLP) to determine stopping criteria for their recursive discretization strategy. Recursive partitioning with a set of values S stops iff

$$\text{Gain}(A,T;S) < \frac{\lg(N-1)}{N} + \frac{\Delta(A,T;S)}{N}, \qquad \text{(eq. 4)}$$

Where N is the number of instances in the set S,

$$\text{Gain}(A,T;S) = \text{Ent}(S) - E(E,A;S); \qquad \text{(eq. 5)}$$

$$\Delta(A,T;S) = \log_2(3^k - 2) - [k * \text{Ent}(S) - k_1 *_ \qquad \text{(eq. 6)}$$

$$\text{Ent}(S1) - k2 * \text{Ent}(S2)],$$

And k_i is the number of class labels represented in the set S_i. Since the partition along each branch of the recursive discretization are evaluated independently using this criteria, some areas in the continuous spaces will be partitioned very finely whereas others (which have relatively low entropy) will be partitioned coarsely.

Risvik in 1997 [52] in entropy-based discretization procedure, Minimum Description Length Procedure (MDLP) is employed as follows:

Divide the system into a sender and receiver both with identical copies of the set of training cases. However, the receiver lacks the classification information. The sender must transmit this missing information to the receiver. This is done by transmitting a theory along with exceptions to this theory. The MDL principle states that the optimum division between a simple theory with a large number of exceptions and a complex theory with a small number of exceptions is the case when the number of bits required to encode both the theory and the exceptions are minimized. The MDL principle is used as a

stopping criterion only. When splitting an interval increases the number of bits needed to encode the theory and exceptions the recursive splitting procedure is ended.

Entropy-based discretization can reduce data size. Unlike the other method mentioned, entropy-based discretization uses class information. This makes it more likely that the interval boundaries are defined to occur in places that may help improve accuracy

Orthogonal Hyperplanes Discretization Algorithm

Pawlak in 1991 [41] introduced Rough Sets as a tool to deal with uncertain or vague knowledge in AI applications. Rough Sets can be treated as a tool for data table analysis.

Skowron et al. in 1995 [55] proposed a method for discretization based on Rough Sets and Boolean Reasoning.

Orthogonal hyperplanes can be seen as a special case of linear discriminations, used in pattern recognition.

Defining the problem of discretization of real value attributes is done as follows in this approach.

Let $A = (U, A \cup \{d\})$ be a decision table where $U = \{x_1, x_2, \ldots, x_n\}$ assuming

$V_a = [I_a, r_a) \subset R$ for any $a \in A$ where R is the set of real numbers.

Assume now that the A is a consistent decision table.

Let P_a be a partition of V_a (for $a \in A$) into subintervals, i.e.

$$P_a = \{[\overset{0}{P_a} = I_a, \overset{1}{P_a}), [\overset{1}{P_a}, \overset{2}{P_a}), \ldots, [\overset{k}{P_a}, \overset{k+1}{P_a} = r_a)\}$$

Where

$$V_a = [\overset{0}{P_a} = I_a, \overset{1}{P_a}) \cup [\overset{1}{P_a}, \overset{2}{P_a}) \cup \ldots \cup [\overset{k}{P_a}, \overset{k+1}{P_a} = r_a) \qquad \text{(eq. 7)}$$

And

$$P_a^0 < P_a^1 < P_a^2 < \ldots < P_a^k < P_a^{k+1}$$

And P_a is uniquely defined by the set of cuts on V_a: $\{P_a^1, P_a^2, \ldots, P_a^k\}$ (empty if card $(P_a) = 1$). The set of cuts on V_a defined by P_a can be identified by P_a. a family $P = \{P_a: a \in V_a\}$ of partitions on A can be expressed on the form

$$\bigcup_{a \in A} \{a\} \times P_a \qquad \text{(eq. 8)}$$

Any $(a, v) \in Pa$ will also be called a cut n Va.

Then the family $P = \{P_a: a \in V_a\}$ defines from $A = (U, A \cup \{d\})$ a new decision table

$$A^P = (U, A^P \cup \{d\}) \qquad \text{(eq. 9)}$$

Where $A^P = \{a^P : a \in A\}$ and $a^P(x) = i \iff a(x) \in [Pa^i, Pa^{i+1})$ for any $x \in U$ and $i \in \{0,\ldots,k\}$. The table A^P is called the P-quantization of A.

Any partition P defines a valuation val_p of Boolean variables of the form $p(a,k)$ by $val_p(p(a,k)) =$ true iff there exists a cut $(a, p_a) \in P$ satisfying . $V_a^k \leq P_a < V_a^{k+1}$. Instead of $val_p(p(a,k))$ we will also write $P \models p(a,k)$.

Defining the initial set of cuts:

$$A_1 = \bigcup_{a \in A} \{ C_i^a = \frac{V_i^a 0 V_{i+1}^a}{2}, 1 \leq i \leq \text{card}(A) - 1 \} \qquad \text{(eq. 10)}$$

For a given subset $X \subseteq U$, and a given cut $c_m^a \in A_1$ we further define

- $W^X(c_m^a)$ = number of pairs of objects from X discerned by c_m^a.

- $W_p(c) = W^{X1}(c) + W^{X2}(c) + \ldots + W^{Xm}(c)$, $X_1, \ldots, X_n$ is the equivalence classes of $IND(A^B)$, indiscernibility (IND) denotes the set of all equivalence classes in the relation $IND(A^B)$.

Based on these definitions Son, et al., in 1997, [54] further proposes an approximate Algorithm:

1. $P = \emptyset$. $L = \{U\}$; A_1 = initial set of cuts on A.
2. For $c \in A_1$ do compute $(W_P(c))$;
3. Choose c_{max} with the largest value $W_P(c_{max})$ among cuts from A_1 and set $P = P \cup \{c_{max}\}$; $A_1 = A_1 \setminus \{ c_{max} \}$;
4. For $X \in L$ do

   If c_{max} cuts the set X into X_1, X_2 then

   Remove X from L; Add to L the two sets X_1, X_2;
5. If all set from L consists of objects from one decision class only then Stop else Go to step 2.

P is the semi-minimal set of cuts.

The algorithm needs time of order $O(kb(|P| \log n))$.

Multivariate Discretization (MVD) Algorithm

Many of the available discretization algorithms search for the best discretization of each continuous variable individually, an approach that refers to as univariate discretization, it does not considering the interdependent relationship between other attributes, ideally the discretization of a continuous variable should be carried out so as to minimize the loss of information that the given variable may contain about other

variables in the domain of interest, an approach that we refer to as multivariate discretization.

Monti† et al. in 1999 [38], describe a new method for multivariate discretization based on the use of a latent variable model. The method is proposed as a tool to extend the scope of applicability of machine learning algorithms that handle discrete variables only. The method is based on clustering: it looks for sub populations of data points which are closest according to their probability conditioned on a cluster variable probably defined. The data points included in each cluster then determine the partition of the value range of the continuous variables. More specifically, following a Bayesian approach, the joint probability distribution over the observed variables is modeled by means of a finite mixture (FM). The induced FM model defines a latent cluster variable, as well as its conditional distribution given the observed variables. The cluster variable and its conditional distribution are then used to drive the discretization of the observed continuous variables.

Chao et al. in 2005 [8], proposed a new multivariate discretization method called Multivariate Interdependent Discretization for Continuous Attributes - MIDCA. This method incorporates the normalized relief and information measures to look for the best correlated attribute with respect to each continuous-valued attribute being discretized, and using the discovered best correlated attribute as the interdependent attribute to carry out the multivariate discretization. They believe that a good multivariate discretization scheme for continuous-valued attributes should rely highly on their perfect correlated attributes respectively. Among an attribute space, each attribute should have at least one most relevant attribute that may be different from others. Their novel multivariate

discretization algorithm can minimize the uncertainty between the interdependent attribute and the continuous-valued attribute being discretized and at the same time maximize their correlation. Such a method can be used as a pre-processing step for the learning algorithms. Their empirical results demonstrate a comparison of performance between MIDCA and various discretization methods for two decision tree algorithms ID3 and C4.5 on twelve real-life data sets from UCI repository.

Bay in 2000 [2] presented a bottom up merging algorithm to discretize continuous variables based on considering the effects on all variables in the analysis and that two regions X and Y should be in the same region after discretization if the instances in those regions have similar multivariate distribution (Fx ~ Fy) across all variables and combinations of variables. He discussed "how continuous values should be handled?" By discretizing these continuous values attributes into disjoint regions and then apply the set mining algorithm. Their experiments indicate that the approach is feasible and, that it does not destroy hidden patterns and that it generates meaningful intervals.

Past work on discretization has usually been done in a classification context where the goal is to maximize predictive accuracy. In knowledge discovery we often analyze the data in an exploratory fashion where the emphasis is not on predictive accuracy but rather on finding previously unknown and insightful patterns in the data. Thus we feel that the criteria for choosing intervals should be different from this predictive context as follows:

- The discretized intervals should not hide patterns

The intervals must be choosing carefully, or you will miss potential discoveries. We refer to this as the resolution problem. Additionally, if the intervals are determined by

examining features in isolation then with discretization we may destroy interactions that occur between several features.

- The intervals should be semantically meaningful

The intervals we choose must make sense to a human expert. Their approach is to finely partition each continuous attribute into n basic regions and then to iteratively merge adjacent intervals only when the instances in those intervals have similar distributions. That is, given intervals X and Y we merge them if (Fx ~ Fy). They use a multivariate test of differences to check this. Merging allows us to deal with the resolution problem and it automatically determines the number of intervals. The multivariate test means that merging only cells with similar distributions so hidden patterns are not destroyed and the regions are coherent.

In this multivariate discretization algorithm they use an alternate test of differences based on Contrast Set miners such as STUCCO. STUCCO attempts to find large differences between two probability distributions based on observational data. For example, given census data if we compare PhD and Bachelor's degree holders, STUCCO would return differences between their distributions such as: P(occupation = sales | PhD) = 2.7 %, while P(occupation = sales | Bachelor) = 15.8 %.

The mining objectives of STUCCO can be stated as follows: Given two groups of instances G1 and G2, and all conjunctions of attribute value pairs C (contrast sets) such that:

$$P(C \mid G1) \neq P(C \mid G2) \qquad \text{(eq 11)}$$

$$|\text{support}(C|G1) - \text{support}(C|G2)| \geq \delta \qquad \text{(eq. 12)}$$

Support is a frequency measurement and is the percentage of examples where C is true for the given group. Equation (11) is a significance criterion and ensures that the differences could not be explained by fluctuations in random sampling. They test this by using a chi-square test which must reject independence of group membership and probability of C. Equation (12) is a size criterion and estimates how big the difference is between two distributions. We require the minimum difference in support to be greater than δ. STUCCO finds these contrast sets using search. It uses a set enumeration tree to organize the search and it uses many of the techniques such as dynamic ordering of search operators, candidate groups and support bounds in conjunction with pruning rules geared for finding support differences. STUCCO also carefully controls error caused by multiple hypothesis testing (i.e., false positives).

STUCCO requires specifying δ which represents how big a difference they are willing to tolerate between two distributions. This allows controlling the merging process: small δ means more intervals and large δ means fewer intervals. They set δ adaptively according to the support of X and Y so that the difference between the two cells must be larger than a fixed percentage of the entire dataset.

The algorithm for Multivariate Discretization (MVD) is as follow:

1. Finely partition all continuous attributes into n basic intervals with equal width of frequency discretization.
2. Select two adjacent intervals X and Y that have the minimum combined support and do not have a known discretization boundary between them as candidates for merging.

3. If Fx ~ Fy then merge X and Y into a single interval. Otherwise place a discretization boundary between them.
4. If there are no eligible intervals stop. Otherwise go to 2.

The algorithm run efficiently, and has the ability to properly discretize data with hidden patterns in high dimensional data by defining a problem called Parity R+I with a high sensitivity to hidden patterns.

Summary

In this chapter, we have reviewed the data preparation which is one step of the data mining process. Data preparation itself is a complicated task that includes many steps, it has high importance for many reasons that: real world data may be incomplete, noisy, and inconsistent. Data preparation also generates a smaller and high quality data set than the original one.

Data preparation task started with converting raw data set to a data mining ready format, dealing with missing data, perform some transformation on data like smoothing or data reduction which is important, data reduction include: feature discretization, sampling, and feature selection. Finally feature composition is applied to transform multiple fields to one field. It is not necessary for any data set to apply all these steps of data preparation; this depends on the type of data set used and quality degree.

The term discretization means that transforming continuous features to discrete one, it works only on continuous numeric (quantitative) data. In the large amount of literature that addresses discretization there is considerable variation in the terminology used to describe these two data types, including "quantitative" vs. "qualitative", "continuous" vs. "discrete", "ordinal" vs. "nominal", and "numeric" vs. "categorical"

Besides, attributes can also be classified by its measurement scales, and four common types of scales are used: nominal, ordinal, interval and ratio.

There exists a diverse classification to classify discretization methods; it can be classified as primary or composite methods. Primary methods can be classified as per the following classifications: supervised vs. unsupervised, univariate vs. multivariate, parametric vs. non-parametric, hierarchical vs. non-hierarchical, global vs. local, eager vs. lazy, disjoint vs. non-disjoint, and dynamic vs. static.

A typical (univariate) discretization process consists the following four steps: sorting the continuous values of the feature to be discretized, evaluating the cut-point for splitting or adjacent intervals for merging, according to some criterion, splitting or merging intervals of continuous values, and finally stopping at some point.

There are three important dimensions to evaluate the results of a discretization method; these dimensions are simplicity, consistency, and accuracy. We need the discretization method to score high result in all three aspects.

Discretization methods can be categorized in several dimensions. We describe different discretization measures according to two approaches: splitting and merging (level 1). We then consider whether a method is supervised or unsupervised (level 2). We further group together methods that use similar discretization measures (level 3) e.g. binning and entropy, the supervised and unsupervised division determines different (non-overlapping) measures used.

Sampling is a data reduction technique that is used allows a large data set to be represented by a much smaller random sample of the data. There are a number of sampling techniques like: Simple random sample without replacement (SWSWOR) of

size n, simple random sample with replacement (SRSWR) on size n, cluster sample, and stratified sample. An advantage of sampling is that the cost of obtaining a sample is proportional to the size of the sample n, as opposite to the data set size.

Feature selection is the process of choosing a subset of the original predictive variable by eliminating redundant and uninformative ones. By extracting as much as information as possible from a given dataset while using the smallest number of features, we can save significant computing time and often build models that generalize better to unseen points. Feature selection is important for several reasons, the fundamental one being arguably that noisy features can degrade the performance of most learning algorithms. The selection criteria of a feature can be done from measuring perspective, there exist many measures that can be used to select features, such that information measures, distance measures, dependence measures, consistency measures, and accuracy measures.

There are several methods that have been proposed to discretize data in the machine learning community. We have look into several methods for discretizing data, with the main focus on the most common discretization methods. These discretization methods are: equal width discretization (EWD), equal frequency discretization (EFD), naïve discretization, chimerge discretization, chi_2 discretization, entropy-based discretization, orthogonal hyperplanes, and multivariate discretization (MVD).

CHAPTER IV

PROPOSED TECHNIQUES

Introduction to Multivariate Discretization

Primitive data can be discrete, continuous or nominal. Nominal type merely lists the elements without any structure; whereas discrete and continuous data have an order can be compared. Discrete data differs from continuous that it has a finite number of values. Discretization, digitization or quantization maps a continuous interval into one discrete value, the idea being that the projection preserves important distinctions. If all that matters, e.g., is a real value's sign, it could be digitized to {0, 1}, 0 for negative, 1 otherwise. A data set has data dimension of attributes or features each holding a single type of values across all data instances. If attributes are the data columns, instances are the rows and their number is the data size. If one of the attributes is the class to be predicted, we are dealing with supervised data, versus unsupervised. The data description vocabulary carries over to the discretization algorithms. If an algorithm discretizing an attribute takes into account the class, it is supervised.

Most common univariate methods discretize one attribute at a time, whereas multivariate methods consider interactions between attributes in the process. Discretization is global if performed on the whole data set, versus local if only part of the data is used, e.g., a subset of instances. There are many advantages of discretized data. Discrete features are closer to a knowledge level representation than continuous ones.

Data can be reduced and simplified, so it is easier to understand, use, and explain. Discretization can make learning more accurate and faster and the resulting Hypotheses (decision trees, induction rules) more compact, shorter, hence can be more efficiently examined, compared and used. Some learning algorithms can only deal with discrete data.

Discretizing one feature at a time is computationally less demanding, though limiting. First, some variables are only important together, e.g., if the predicted class = sign (xy), only discretizing x and y in tandem can discover their significance, each alone can be inferred as not related to the class and even discarded. Second, especially in noisy/stochastic data, a nonrandom feature may be only slightly above randomness, so can still test as non-significant. Only a grouping a number of such features can reveal their above random nature. Those considerations are crucial since discretization is information loosing transformation, if the discretization algorithm cannot spot regularity, it will discretize sub optimally, possibly corrupting the features or omitting them as irrelevant. Besides, data mining applications proliferate beyond the each feature alone informative data. To exploit such data and the capabilities of advanced mining algorithms, the data preprocessing including discretization, needs to be equally adequate. Voting ensembles also pose demands. When data is used to train a number of imperfect classifiers, which together yield the final hypothesis, the aim of discretization should not be so much to perfect individual feature cut-points, but to ensure that the features as a whole carry as much information as possible to be recaptured by the ensemble.

When discretization goes beyond the fixed interval length/frequency approach, it needs to measure guiding it through the search for salient features and their cut-points.

Liu et al. in 1999 [33] the search strategy can itself have different implantations. However there are many score functions like Shannon conditional entropy, Chi-square statistics, accuracy, and number of discrete values.

The Proposed System

The system was designed to compare different discretization algorithms; we have used several datasets with a large percentage of continuous attributes, testing will be performed to evaluate the performance of the different discretization algorithms described in chapter 3. The pipeline used to perform the data analysis is illustrated in Figure 8.

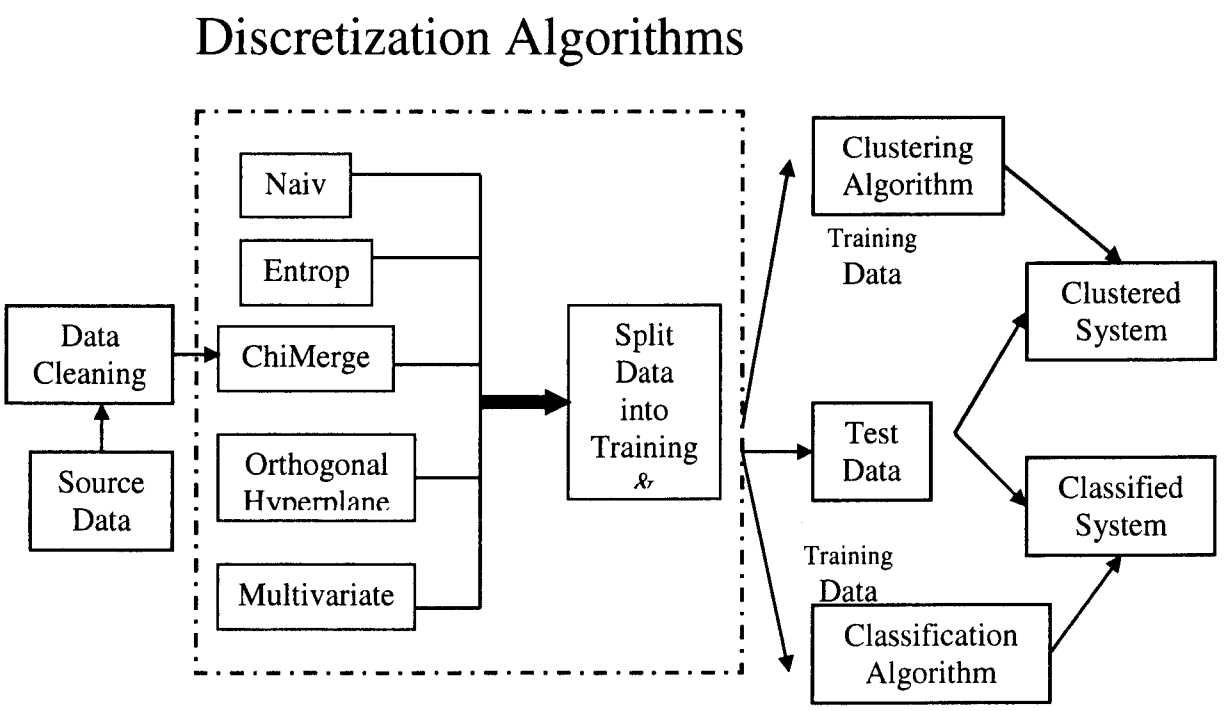

Figure 8: KDD pipeline of the proposed system

- Source Data

The data will be accepted from a data source, the data sources may vary in size and complexity from flat files to a multi dimensional data models. Data warehouses also can be useful data sources. It based on a multidimensional data model. This model views data in the form of data cubes which allows data to be modeled and viewed in multiple dimensions. In our proposed system we will accept two types of data files which are comma delimited text files (.txt) files, and Microsoft Data Base Access (.mdb) files

- Data Cleaning

Real world's data are highly susceptible to noisy, missing, and inconsistent data due to their huge size. These data need to some sort of data preprocessing before usage to improve accuracy and efficiency of specific algorithm to get the perfect usage of these data and to extract useful patterns. There are a number of data preprocessing techniques. Han et al. in 2001 [19] data cleaning is one of them that can be applied to remove noise and correct inconsistencies in the data.

Data cleaning routines work to clean the data by filling in missing values, smoothing noisy data, identifying or remove outliers, and resolving inconsistencies. Dirty data can cause confusion for the mining procedure, resulting in unreliable output.

- Discretization Algorithms

Several methods have been proposed to discretize data as a preprocessing step for the data mining process. In our system we have chose five different discretization algorithms. Some of these discretization algorithms have widely used in the machine learning community. These methods are belonging to different categories of

discretization algorithms. We have chosen five discretization algorithms of different types. These algorithms are Naïve, Entropy, ChiMerge, Orthogonal Hyperplanes, and Multivariate Discretization (MVD) algorithms.

- Splitting Data into Training & Testing

Data sets will be divided into training and testing sets; the software will create two files one for training and the other for testing data sets, according to the type of input file, the two file will created one for odd instances that will be used as a training data set and the other will be used as a testing data set, if the user open a text file the software will create two text files for with training and testing datasets and if the user open Ms Access table the software will create also two text files one for training (odd records), and one for testing (even records).

- Clustering Algorithm

After data set has been discretized and divided into training and testing data sets, a clustering algorithm will be applied on training data set to group data into classes or clusters so that objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters. Clustering is an example of unsupervised learning. There are several clustering algorithms organized into different categories, but we will use one suitable to numerical data sets like k-means, and k-medoids algorithms. After data has been clustered, a model will be generated. This model will be tested using testing data set to estimate the accuracy of this model.

- Classification Algorithm

Data classification is a two-step process, in the first step a model is built describing a predetermined set of data classes or concepts. The model is constructed by analyzing

data sets described by attributes, each object is assumed to belong to a predefined class, as determined by one of the attributes, called the class label. In this stage data objects are analyzed to build the model from the training data set. There are several classification algorithms that we can use like decision tree induction, Naïve Bayesian classifier, and Back propagation. In later stage this model will be tested using tested data set to estimate its prediction accuracy. This is known as supervised learning.

- Clustered System

After a clustering model has been generated, it will be used for clustering a new data sample that will be joined to a certain cluster. The quality of clustering algorithm can be measured based on a measure of dissimilarity of objects, which can be computed for several data types, many of these clustering algorithms developed especially for numerical data. Test data will be used to estimate the accuracy of the clustering model, and finally the system will be used to join a new data samples to a specific cluster (or group).

- Classified System

After a classification model has been generated, it will be used for classification. First, the predictive accuracy of the model (or classifier) is estimated. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. For each test sample, the known class label is compared with the learned model's class prediction for that sample. As a result the system can work properly and classify any new data sample.

The Implementation of Multivariate Discretization (MVD) Process

Our implementation of the Multivariate Discretization (MVD) process is shown in Figure 9. By using the word multivariate, we mean the implementation of the discretization process by quantifying multiple continuous attributes (features) at the same time. Figure 9 depicts the steps of the (MVD) discretization process. The discretization process consists of the following steps:

- Get Continuous Data

A filtering operation will be performed on data set to choose only the continuous value attributes in case of that data set may contain mixed type of attributes. Some data sets may contain categorical attributes that we must ignore.

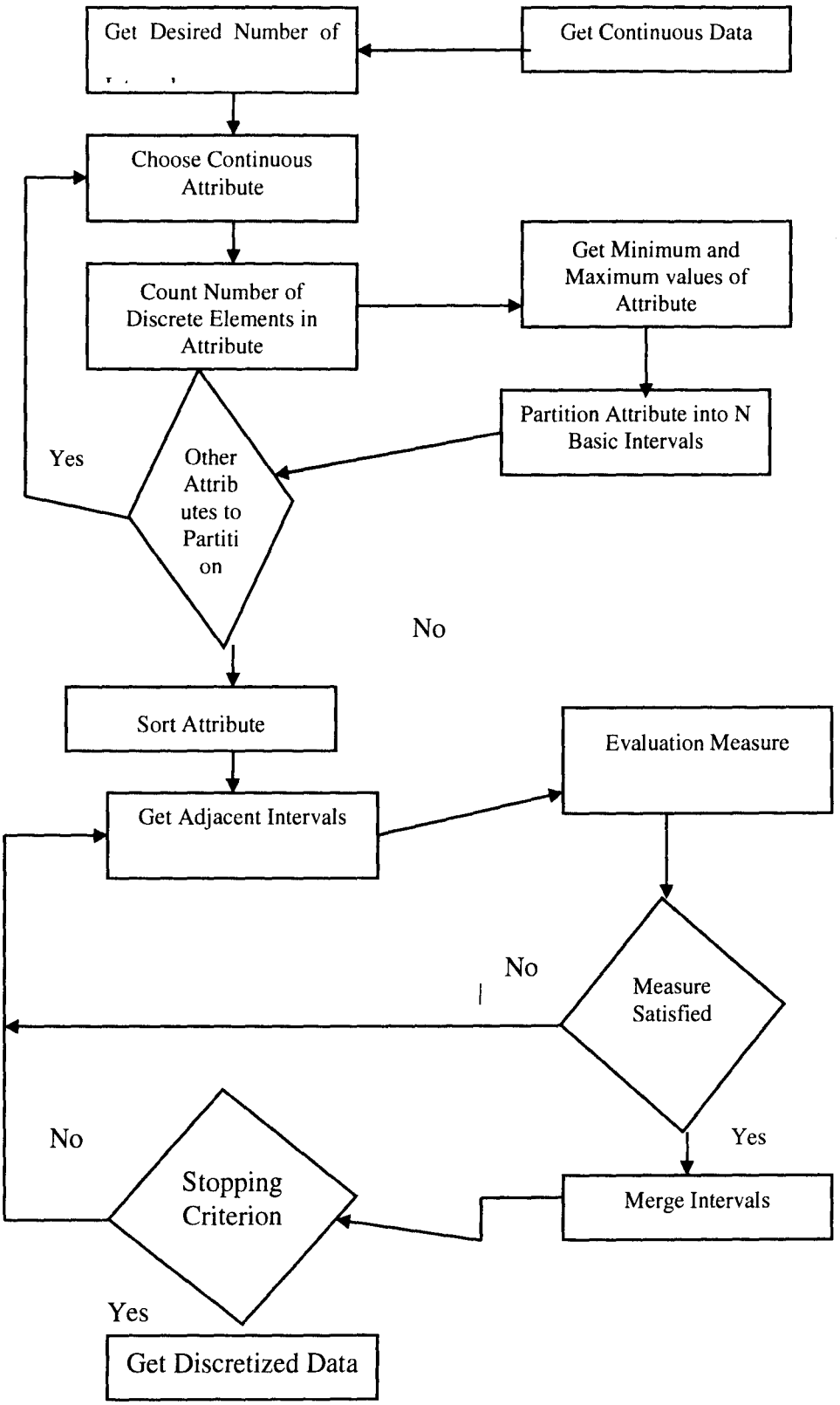

Figure 9: The Implementation of MVD process

- Get Desired Number of Intervals

The user will be prompted to enter the desired number of intervals; if the user does not enter a number of the intervals the system will use the default value:

Max (M/3C, min (C=1, M)), where C is the number of classes in the input set, and M is the Number of distinct values for each attribute.

Dougherty et al. in 1995 [11] set number of desired intervals (number of bins) = 10 and a default value = max {1, 2.log ℓ}, where ι is the number of distinct observed values for each attribute.

- Choose Continuous Attribute

The continuous attribute will be selected to perform the discretization process on it, in other word, to convert these continuous values into discrete values.

- Count Number of Discrete Elements in Attribute

In this step we start to work on the continuous attribute by counting the number of distinct values in the selected attribute (feature).

- Get Minimum and Maximum Values of Attribute

After sorting the continuous attributes in an ascending order, every attributes will be searched to get its maximum and minimum values it. The minimum and maximum values in and number of intervals will determine the interval width.

- Partition Attribute into N Basic Intervals

Each attribute will be partitioned (divided) into number of intervals determined by the user, or if the user does not enter a value, the default value will be computed by the program. These intervals have the same width

- Sort Attribute

The continuous values of an attribute are sorted in an ascending order. Sorting is very important because it is gives an ordered interval labels which candidates for merging. Sorting can be computationally very expensive if care is not taken in implementing it with discretization. It is important to speed up the discretization process by selecting suitable sorting algorithms. Many sorting algorithms can be found in classic data structures and algorithms books. Among these "Quick-sort" is an efficient sorting algorithm. We will sort attributes once and for all the attributes at the beginning of the discretization to improve the efficiency of the algorithm.

- Get Adjacent Intervals

The continuous values are now divided into intervals; the next step in the discretization process is to find the best pair of adjacent intervals that have minimum combined support, and do not have a known discretization boundary between them as candidates for merging.

- Evaluation Measure

There are numerous evaluation functions found in the literature such as entropy measures, and statistical measures. The evaluation measure in Multivariate Discretization (MVD) algorithm for two adjacent intervals is that, they must have a minimum combined support, and unknown discretization boundary, they are eligible to merge, otherwise MVD search for other adjacent intervals in the whole set.

- Merge Intervals

In our bottom-up approach intervals are merged. For merging, adjacent intervals are evaluated to find the best pair of intervals to merge in each iteration. Discretization

continuous with the reduced number (decreased by one) of intervals until the stopping criterion is satisfied.

- Stopping Criterion

A stopping criterion specifies when to stop the discretization process. It is usually governed by a trade-off between lower number of intervals with a better understanding but less accuracy and a higher number of intervals with a poorer understanding but higher accuracy. A stopping criterion can be very simple such as fixing the number of intervals at the beginning or a more complex one like evaluating a function. The MVD requires that the minimum difference in support is greater than δ.

- Get Discretized Data

After merging stopped according to a stopping criterion, now this data set has discrete values which can be categorized into a classification and ready for used by classification or clustering algorithms.

Data Visualization

Data visualization has developed in several directions: theoretical, methodological, and in new application areas. Advances include the development of computer graphics, hardware, and better approaches to visualizing large data sets. Fayyad et al. in 2001 [15] data visualization is a growing area of research and applications, In this section we will define and present types of data to be visualized, and the most up to date data visualization techniques for data mining.

Introduction

The term visualization means the graphical representation of information. The point of data visualization is to let the user understand what is going on. Since data

mining usually involves extracting "hidden" information from a database, this understanding process can get somewhat complicated. In most standard database operations nearly everything the user sees is something that they knew existed in the database already. A report showing the breakdown of sales by product and region is straightforward for the user to understand because they intuitively know that this kind of information already exists in the database. If the company sells different products in different regions of the county, there is no problem translating a display of this information into a relevant understanding of the business process.

Data mining, on the other hand, extracts information from a database that the user did not already know about. Useful relationships between variables that are non-intuitive are the jewels that data mining hopes to locate. Since the user does not know beforehand what the data mining process has discovered, it is a much bigger leap to take the output of the system and translate it into an actionable solution to a business problem. Since there are usually many ways to graphically represent information (or data sets), the visualizations that are used should be chosen to maximize the value to the viewer. This requires that we understand the viewer's needs and design the visualization with that end-user in mind. The patterns that discovered must be presenting in a specific visual form. This visual form should perceivable, understandable, and interpretable in frames of a domain theory associated with data used for data mining.

Data Type to Be Visualized

Not all of the data can be visualized, in other words, there must be characteristics of data that can be visualized. Ward in 1997 [60] determine 16 characteristics of data that can be visualized: (1) Numeric, Symbolic (or mix). (2) Scalar, vector, or complex structure. (3)

Various units. (4) Discrete or continuous. (5) Spatial, quality, category, temporal, relational, structural. (6) Accurate or approximate. (7) Dense or sparse. (8) Ordered or non-ordered. (9) Disjoint or overlapping. (10) Binary, enumerated, multilevel. (11) Independent or dependent. (12) Multidimensional. (13) Single or multiple sets. (14) May have similarity or distance metric. (15) May have intuitive graphical representation (e.g. temperature with color). (16) Has semantics which may be crucial in graphical consideration.

Keim in 2002 [28] describes that, in data visualization the data usually consists of a large number of records each consisting of a number of variables or dimensions. Each record corresponds to an observation, measurement, transaction, etc. Examples are customer properties, e-commerce transactions, and physical experiments. The number of attributes can differ from data set to data set: One particular physical experiment, for example, can be described by five variables, while other may need hundreds of variables. We call the number of variables the dimensionality of the data set. Data sets may be one-dimensional, two-dimensional, and multidimensional or may have more complex data types such as text/hypertext or hierarchies/graphs. Sometimes, a distinction is made between dense (or grid) dimensions and the dimensions which may have arbitrary values. Depending on the number of dimensions with arbitrary values the data is sometimes also called univariate, bivariate, etc.

- One Dimensional Data

One dimensional data usually has one dense dimension. A typical example of one dimensional data is temporal data. Note that with each point of time, one or multiple

data values may be associated. An example is time series of stock prices or the time series data.

- Two Dimensional data

Two-dimensional data has two distinct dimensions. A typical example is geographical data where the two distinct dimensions are longitude and latitude. X-Y plots are a typical method for showing two dimensional data and maps are a special type of x-y plots for showing two-dimensional geographical data. Examples are the geographical maps used in Polaris and in MGV. Although it seems easy to deal with temporal or geographic data, caution is advised. If the number of records to be visualized is large, temporal axes and maps get quickly glutted and may not help to understand the data.

- Multi Dimensional Data

Many data sets consist of more than three attributes and therefore, they do not allow a simple visualization as 2-dimensional or 3-dimensional plots. Examples of multidimensional (or multivariate) data are tables from relational databases, which often have tens to hundreds of columns (or attributes). Since there is no simple mapping of the attributes to the two dimensions of the screen, more sophisticated visualization techniques are needed. An example of a technique which allows the visualization of multidimensional data is the Parallel Coordinate Technique, which is also used in the Scalable Framework. Parallel Coordinates display each multidimensional data item as a polygonal line which intersects the horizontal dimension axes at the position corresponding to the data value for the corresponding dimension.

- Text & Hypertext

Not all data types can be described in terms of dimensionality. In the age of the World Wide Web, one important data type is text and hypertext as well as multimedia web page contents. These data types differ in that they can not be easily described by numbers and therefore, most of the standard visualization techniques can not be applied. In most cases, first a transformation of the data into description vectors is necessary before visualization techniques can be used. An example for a simple transformation is word counting which is often combined with a principal component analysis or multidimensional scaling.

- Hierarchies & Graphs

Data records often have some relationship to other pieces of information. Graphs are widely used to represent such interdependencies. A graph consists of set of objects, called nodes, and connections between these objects, called edges. Examples are the e-mail interrelationships among people, their shopping behavior, and the file structure of the hard disk or the hyperlinks in the World Wide Web. There are a number of specific visualization techniques that deal with hierarchical and graphical data

- Algorithms & Software

Another class of data is algorithms & software. Coping with large software projects is a challenge. The goal of visualization is to support software development by helping to understand algorithms, e.g. by showing the flow of information in a program, to enhance the understanding of written code, e.g. by representing the structure of thousands of source code lines as graphs, and to support the programmer in debugging the code, i.e. by visualizing errors.

According to Ward [60] characterization (or classification) of data that can be visualized, we can classify the data sets that we will use in experimentation as: (Numeric, continuous, relational, approximate, dense, non-ordered, overlapping, enumerated, dependent, Multidimensional, multiple sets, have distance metric, some of them have intuitive graphical representation, has semantics in crucial representation). Also for organization done by Keim [28], the type of data sets that we will use in experimentation is multidimensional, and text.

Conditions For a Good Data Visualization System

There exist numerous data visualization systems available, but how we can judge (or choose) between these data visualization systems, Tufte in 2001 [59] states the basic conditions that must offered by a good data visualization system:

- Show the data.
- Induce the viewer to think about the substance, not the methodology.
- Avoid distorting what the data says.
- Present many numbers in a small space.
- Make large data sets coherent.
- Encourage the eye to compare different pieces of data.
- Reveal the data at several levels of detail (broad overview to fine structure).
- Serve a reasonably clear purpose: description, exploration, or decoration.
- Be closely integrated with the statistical and verbal descriptions of the data set.
- Gives the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.
- Nearly always multivariate.

- Requires telling the truth about the data.

Data Visualization Techniques

Keim in 2002 [28] states that there are a large number of visualization techniques which can be used for visualizing the data. In addition to standard 2D/3D techniques such as x-y (x-y-z) plots, bar charts, line graphs, etc., there are a number of more sophisticated visualization techniques. The classes correspond to basic visualization principles which may be combined in order to implement a specific visualization system.

- Geometrically-transformed Displays

Geometrically transformed displays techniques aim at finding "interesting" transformations of multidimensional data sets. The class of geometric display techniques includes techniques from exploratory statistics such as scatter plot matrices, and techniques which can be subsumed under the term "projection pursuit". Other geometric projection techniques include Prosection Views, Hyper slice, and the well-known Parallel Coordinates visualization technique. The parallel coordinate technique maps the k-dimensional space onto the two display dimensions by using k equidistant axes which are parallel to one of the display axes. The axes correspond to the dimensions 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 dimensions as seen in Figure 10.

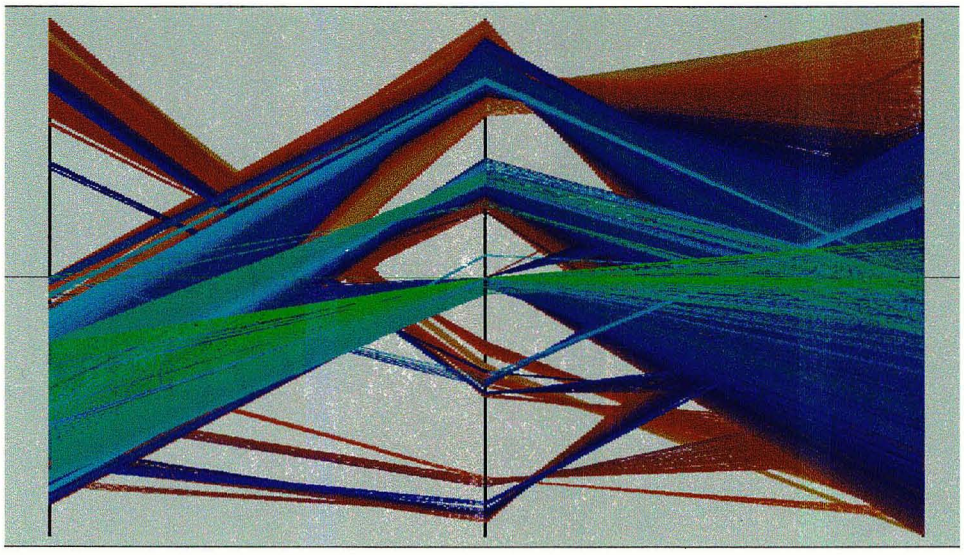

Figure 10: Parallel coordinate visualization © IEEE

- Iconic Displays

Another class of visual data exploration techniques is the iconic display techniques. The idea is to map the attribute values of a multidimensional data item to the features of an icon. Icons can be arbitrarily defined: They may be little faces, needle icons as used in MGV, star icons, stick figure icons, color icons, and tile bars. The visualization is generated by mapping the attribute values of each data record to the features of the icons. In case of the stick figure technique, for example, two dimensions are mapped to the display dimensions and the remaining dimensions are mapped to the angles and/or limb length of the stick figure icon. If the data items are relatively dense with respect to the two display dimensions, the resulting visualization presents texture patterns that vary according to the characteristics of the data and are therefore detectable by preattentive perception.

- Dense Pixel Displays

The basic idea of dense basic technique is to map each dimension value to a colored pixel and group the pixels belonging to each dimension into adjacent areas. Since in

general dense pixel displays use one pixel per data value, the techniques allow the visualization of the largest amount of data possible on current displays (up to about 1.000.000 data values). If each data value is represented by one pixel, the main question is how to arrange the pixels on the screen. Dense pixel techniques use different arrangements for different purposes. By arranging the pixels in an appropriate way, the resulting visualization provides detailed information on local correlations, dependencies, and hot spots.

Well-known examples are the recursive pattern technique and the circle segments technique. The recursive pattern technique is based on a generic recursive back-and forth arrangement of the pixels and is particular aimed at representing datasets with a natural order according to one attribute (e.g. time series data). The user may specify parameters for each recursion level, and thereby controls the arrangement of the pixels to form semantically meaningful substructures. The base element on each recursion level is a pattern of height h_i und width w_i as specified by the user. First, the elements correspond to single pixels which are arranged within a rectangle of height h_1 and width w_1 from left to right, then below backwards from right to left, then again forward from left to right, and so on.

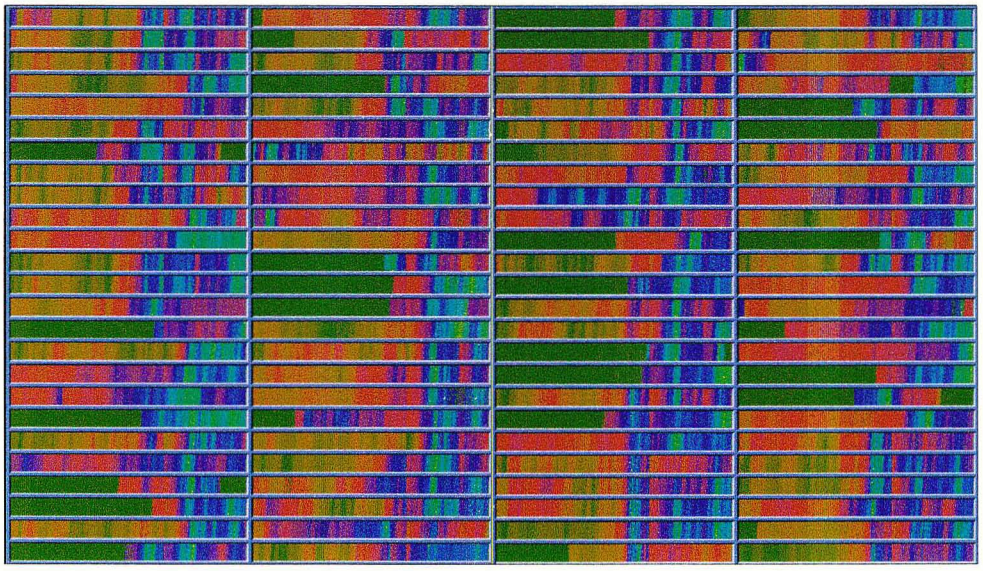

Figure 11: Dense Pixel Display: Recursive Pattern Technique © IEEE

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 pattern resulting from the level (i–1) arrangements. In Figure 11, an example recursive pattern visualization of financial data is shown. The visualization shows twenty years (January 1974 - April 1995) of daily prices of the 100 stocks contained in the Frankfurt Stock Index (FAZ). The idea of the circle segments technique is to represent the data in a circle which is divided into segments, one for each attribute. Within the segments each attribute value is again visualized by a single colored pixel. The arrangement of the pixels starts at the center of the circle and continues to the outside by plotting on a line orthogonal to the segment halving line in a back and forth manner. The rational of this approach is that close to the center all attributes are close to each other enhancing the visual comparison of their values. Figure 12 shows an example circle segment visualization of the same data (50 stocks) as shown in Figure 11.

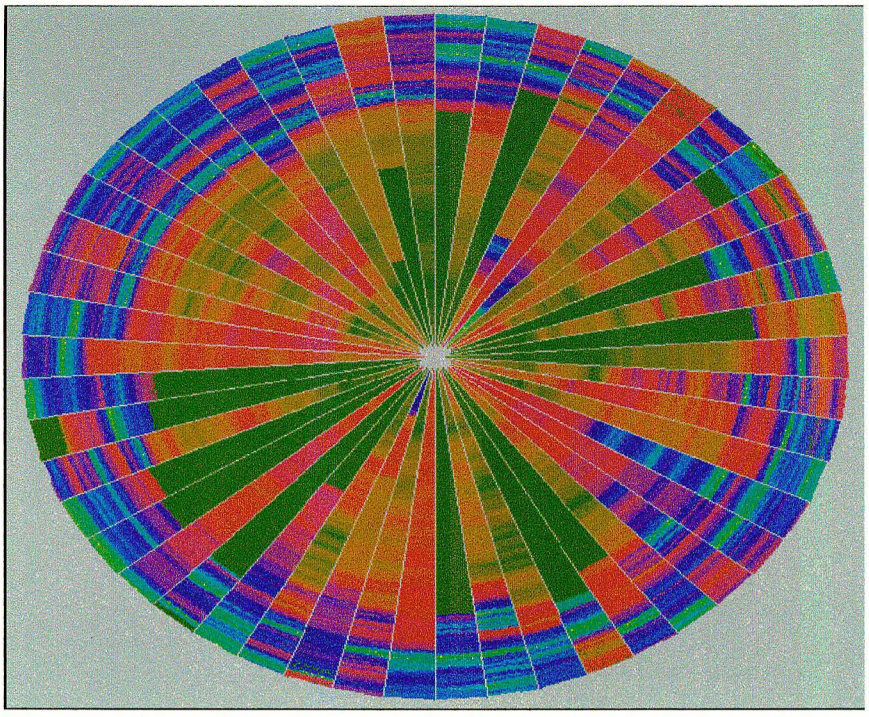

Figure 12: Dense Pixel Display: Circle Segments Technique © IEEE

- Stacked Displays

Stacked display techniques are tailored to present data partitioned in a hierarchical fashion. In case of multidimensional data, the data dimensions to be used for partitioning the data and building the hierarchy have to be selected appropriately. An example of a stacked display technique is Dimensional Stacking. The basic idea is to embed one coordinate system inside another coordinate system, i.e. two attributes form the outer coordinate system, and two other attributes are embedded into the outer coordinate system, and so on. The display is generated by dividing the outermost level coordinate systems into rectangular cells and within the cells the next two attributes are used to span the second level coordinate system. This process may be repeated one more time. The usefulness of the resulting visualization largely depends on the data distribution of

the outer coordinates and therefore the dimensions which are used for defining the outer coordinate system have to be selected carefully. A rule of thumb is to choose the most important dimensions first.

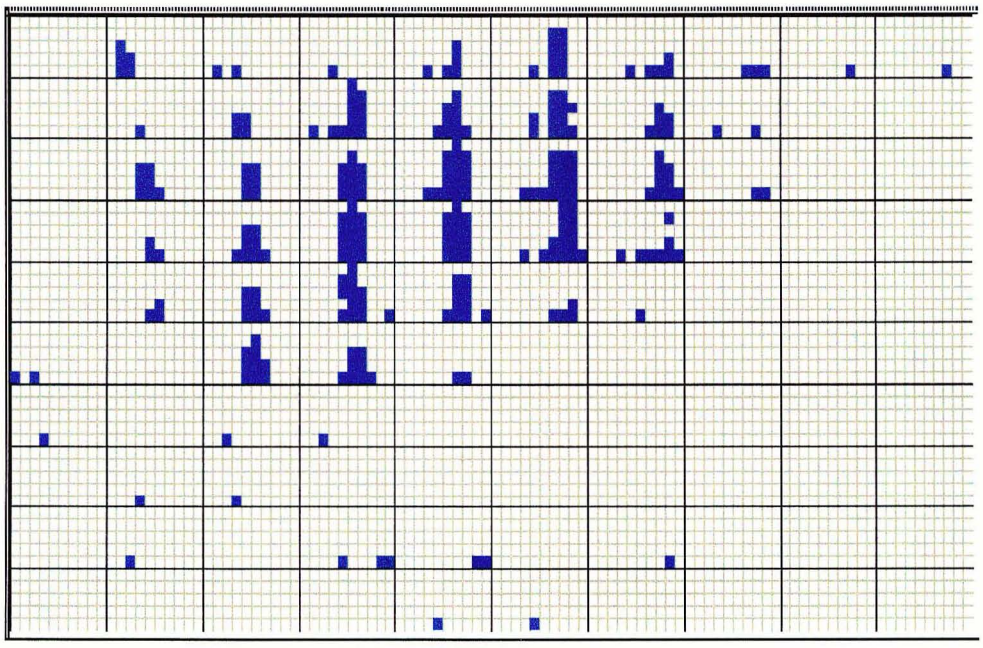

Figure 13: Dimensional Stacking Visualization of Oil Mining Data © IEEE

A dimensional stacking visualization of oil mining data with longitude and latitude mapped to the outer x and y axes, as well as ore grade and depth mapped to the inner x and y axes is shown in Figure 13.

Focus on Software

Wong in 1999 [61] defines Visualization of Data Mining (VDM) as an interactive and iterative process and requires software that should be reconfigurable, general, and widely usable. Hibino in 1999 [21] the observations of Pyle (1999) on the importance and time consumption of data preprocessing in KDD process apply also to visual exploration of data. In an expert user study, reformatting and transforming data were evaluated to be at least as important functions as the tasks of exploring and presenting the data.

Consequently, enabling visual data mining require a lot of effort in integrated software systems. Polaris and GGobi are recent examples of visualization systems for data exploration in general. They support various multivariate interactive visualization techniques, provide programming interfaces, specialized formal visualization languages, and are compatible with different databases and data types.

Grinstein et al. in 2002 [18] categorize MATLAB and S-PLUS as a general purpose statistical and technical computation environments that also provide a collection of different, interactive visualization primitives.

Kohonen Self Organized Maps (SOM) is a nonlinear visualization system that projects data down to two dimensions while trying to preserve the distances between all initially given n-dimensional samples, this type of visualization could be also classified as dimensionality reduction visualization systems. Besides, there exist numerous visualization software available on the internet that can be used in diverse applications to visualize data sets.

Most visualization examples in this thesis were created using Orange [66] and Silicon Graphics MineSet [67] that implement various data preprocessing, and projection methods in addition to visualization.

Examples

In this section we present examples that demonstrate some visualization methods on standard data sets.

- Sports Players Data Set

The sports player's dataset contains two continuous attributes which are weight and height of a player, the class attribute has four values (Judo players, Jockeys, Basket Ball

players, Rugby players). The weight and height were measured on 30 players of each class. In total, there are 120 samples. The data set is used throughout the experimentation analysis in chapter five.

Figure 14 shows a scatter plot of the two attributes weight and height. A scatter plot is one of the common visualization methods of multidimensional data sets; the class of a data item is shown using shape and color. One can see that the classes is linearly separated, but with some noise (or outliers).

- Iris Data Set

The iris data set is often used in such illustrations of visualization and machine learning. The data contains sepal and petal lengths and widths measured from 150 Iris flowers that belong to three specific types called Iris setosa, virginica and versicolor. Fifty plants stem from each class. Two features (Petal Length and Petal Width) are relevant ones.

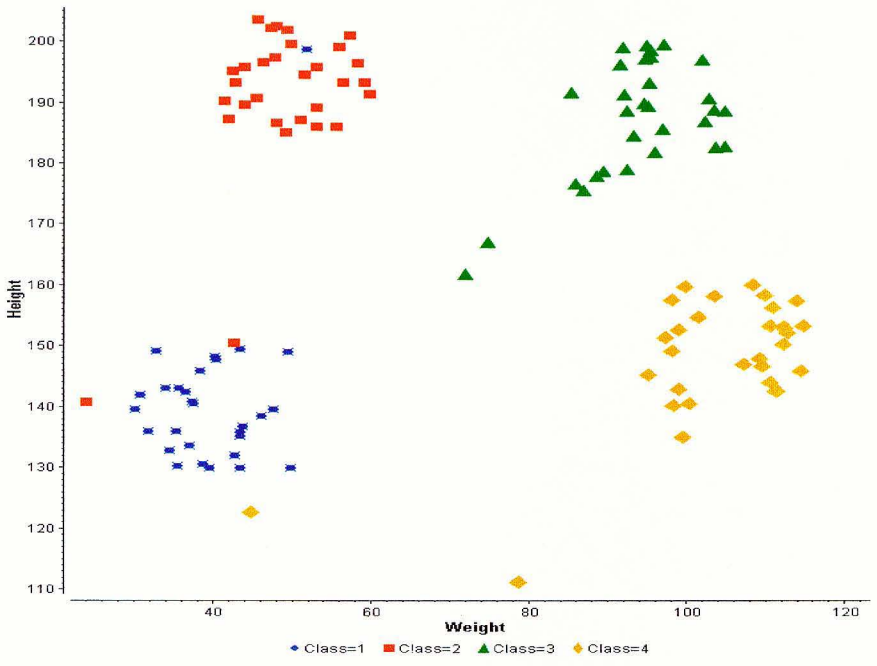

Figure 14: Scatter plot of sports player's data set

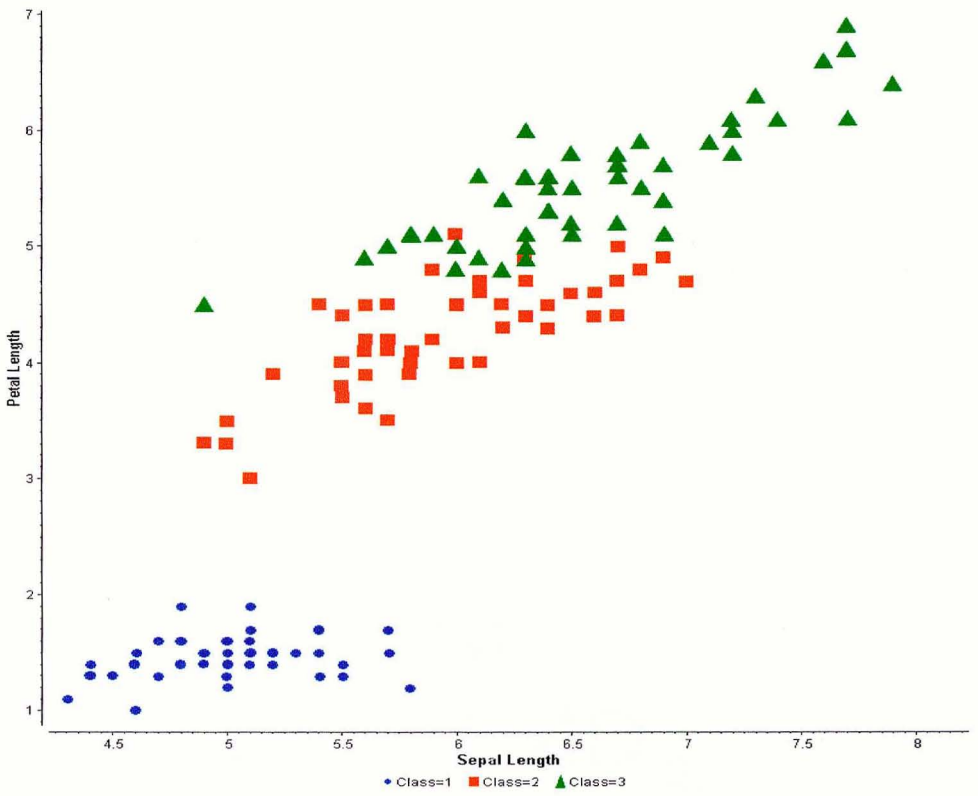

Figure 15: Scatter plot of iris data set

Figure 15 shows a scatter plot of iris data set, the class of data item is shown using shape and color, we can see that most samples of two classes iris-versicolor and iris-virginica interfered with each other.

Another way of visualizing multidimensional data sets is the parallel coordinate plot is shown in Figure 16. In the parallel coordinate plot, one can read the attribute values of a particular item quite accurately, if it is not shadowed by other items. Visual overlapping may be a severe problem in this visualization.

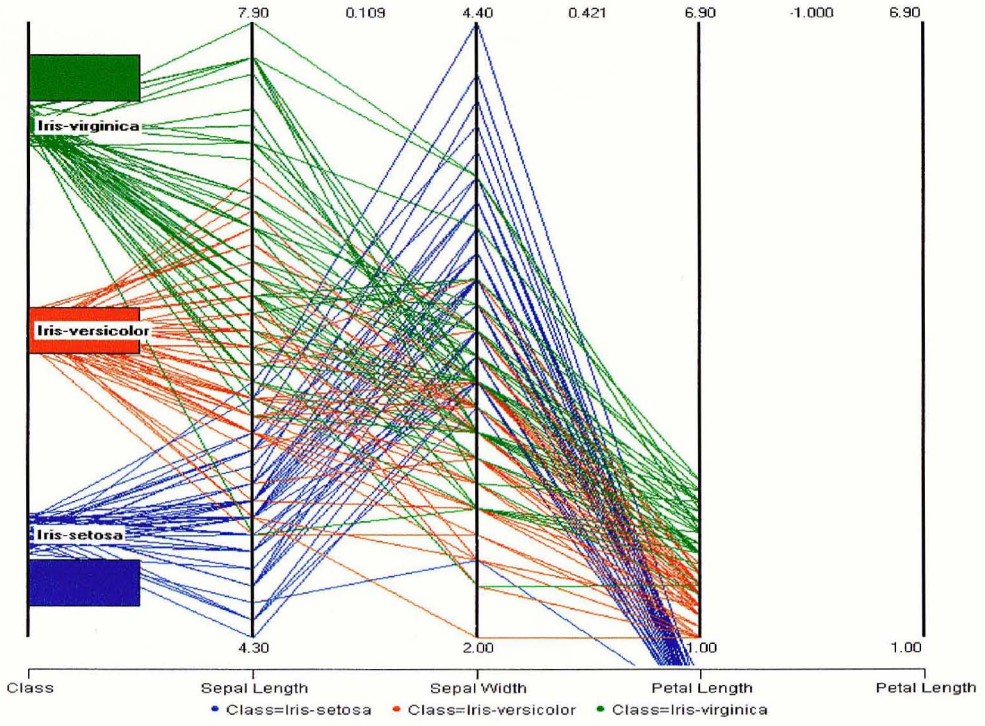

Figure 16: Parallel Coordinate plot of iris data set

Summary

In this chapter, we have produced the Multivariate Discretization (MVD) algorithm, and the proposed system for discretization that includes many discretization algorithms as a preprocessing step for the classification or clustering system. Training data set used in the training of the discretization algorithm and testing data set used to test the algorithm. We have describe our implementation of MVD process, which contains different preprocessing steps for the algorithm

The exploration of large data sets is important but difficult problem. Data visualization techniques may help to solve the problem. Data visualization is an effective tool to improve data analysis. We can use data visualization as a statistical tool to explore data to determine its level of consistency, to aid the user to understand the nature of the data, and finally to choose the best discretization technique (the technique that will

produce highest accuracy with this data). After that data is ready for classification or clustering.

Data sets may be one-dimensional, two-dimensional, and multidimensional or may have more complex data types such as text/hypertext or hierarchies/graphs. There are some conditions for determining the quality of a data visualization system.

There are a large number of visualization techniques which can be used for visualizing the data. Besides common techniques like x-y plots, bar charts, line graphs, etc. there are a number of more sophisticated visualization techniques. Visualization software allows the users to explore the data in different ways and at different levels of abstraction with highly interactive tools from Polaris to SOM. We have given examples of scatter plots, parallel coordinates on standard data sets to demonstrate how these techniques will help in providing the viewer with a qualitative understanding of the data.

CHAPTER V

EXPERIMENTAL EVALUATION

Motivations

There are several discretization algorithms available in the machine learning community; these algorithms can be classified to different categories, but the base function of these algorithms is to convert continuous data collected from different sources to discrete data. Not all of these algorithms can be effective with all data sources, in other words, some algorithms produces higher accuracy results with some data types and low accuracy results with others. We have selected some algorithms that belong to different discretization categories for comparing the performance of each algorithm with others. These algorithms are: Naïve, Entropy, Chimerge, Orthogonal Hyperplanes, and Multivariate discretization (MVD) algorithms.

In this chapter we will present and explain multivariate discretization (MVD) main modules. After that we will run the MVD program on experimental data sets, showing the effect of discretization on different data sets by using visualization techniques, and comparing the results obtained from running continuous data sets on these algorithms. The MVD program split discrete data file (output of discretization algorithm) into training and testing data sets by talking the odd instances for training and even instances for testing the classification algorithm.

To classify data objects, we have used artificial neural network–single layer perceptron classifier, and multilayer perceptron classifier, then computing the accuracy obtained from training and testing data sets.

Data Sets

We have run our experiments on five natural data sets all with quantitative attributes; we have chose four data sets from UCI machine learning repository (Blake, et al. in 1998 [4]) for studying the performance of each discretization method. These data sets are:

- Cars

The data set contains measures of cars. The data set contains 14 continuous attributes (Price, City MPG, High Way MPG, # Cylinders, Engine Size, Horse Power, RPM (@ max HP), Revs/min, Fuel Tank Capacity, Passenger Capacity, Length, Width, Wheel Base, Weight). The class attribute has 5 values (Small, MidSize, Compact, Large, Sporty). The data set contains 500 samples, 100 samples for each class.

- Glass-Identification

The data set includes substances of a glass to determine whether the glass is a type of float or not. This data set contains 10 attributes including an ID number, all attributes are continuous expect ID number, the other attributes are the substances of a glass (refractive index, Sodium, Magnesium, Aluminum, Silicon, Potassium, Calcium, Barium, Iron). The data set contains 214 samples glasses: 163 samples windows glasses and 51 samples non-window glasses. The class attribute has values form 1 to 7.

- Heart

The heart (Cleveland) data set contains several biological and medical attributes like: age, sex, blood pressure, heart rate, etc. it contains 14 attributes including class attribute to classify objects into 5 different classes from (1 to 5) indicating a diagnosis of heart condition. The data set contains 303 samples, distributed as: (164 samples for class 1), (55 samples for class 2), (36 samples for class 3), (35 samples for class 4), (and 13 samples for class 5).

- Iris

The data set contains measures of iris flower. There are 4 continuous attributes. The class attribute has 3 values (Setosa, Versicolor, and Virginica). The length and breadth of both petal and sepal were measured on 50 flowers of each class. In total, there are 150 instances.

- Liver-Disorders

The data set is a blood test of liver disorders that may arise from excessive alcohol consumption. The data set contains 7 attributes including selector which is the class attribute. The data set contains 345 samples. The class value is either (1 or 2) distributed as145 samples (1) and 200 samples (2).

These data sets vary extensively in the number of instances and the dimension of the attribute space. Table 2 concludes each data set, including the number of instances (Size), number of numerical attributes (Num), and classes (Class).

Table 2

Experimental Data Sets

Data Set	Size	Num	Class
Cars	500	14	5
Glass-Identification	214	9	7
Heart	303	7	5
Iris	150	4	3
Liver-Disorders	345	6	2

Modules of Multivariate Discretization (MVD) Program

The multivariate discretization (MVD) program contains different modules, as shown in figure 5.1. These modules are:

- Open Text File/Access Database

This module handles opening comma delimited text files or access database tables, then retrieves data from it, and stores data values in the specified memory locations.

- View Continuous Data

After retrieving data from text file/access database table, continuous data are used to populate data grid to give the user a view to the data before implementing discretization algorithm.

- Discretize Continuous Data

This module implements the multivariate discretization (MVD) algorithm; the output of discretization module is discrete data and cut points of each attribute stored in a multidimensional array.

- Create Discrete File

The MVD program creates a new text file with the same name of the opened file that contained continuous data, and stores the discretized data in this new file.

- View Discrete Data

The program opened the discrete file created in the last step and populate data grid with discrete data to give the user a view of the discrete data.

- Split Discrete File

The MVD program creates a new text file which has the same name of the opened file and saves discrete data to it, after that it splits data into training and testing. The output of the program is a three text files, the first one contains the discretized data and has the same name of the original opened file. The other two text files are named training and testing proceeded with the opened file name regardless of the type of the opened file if it is text file or access database table, training file contains odd instances and testing file contains even instances of the discrete file. Training file is used for training the classification algorithm and the testing file is used for testing the accuracy of the classification algorithm.

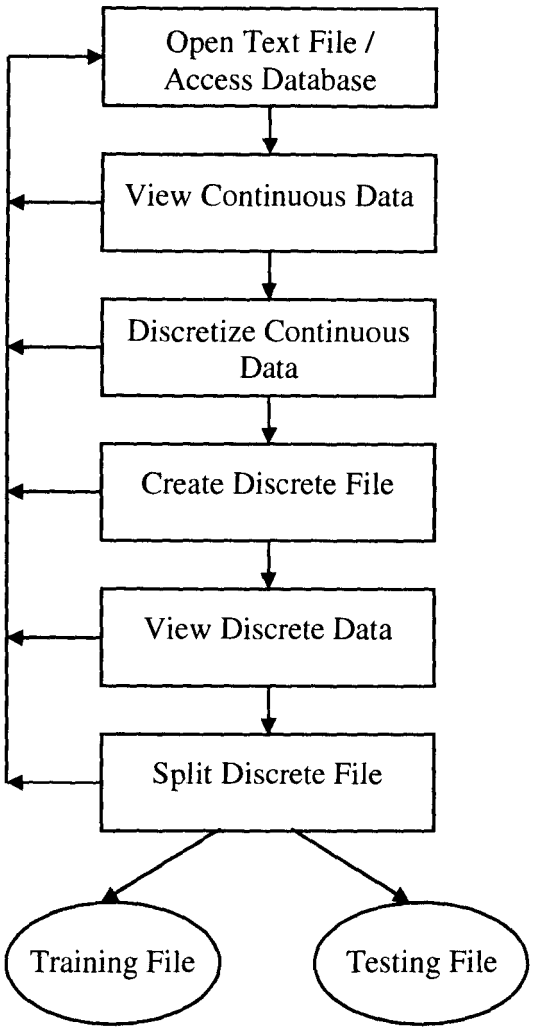

Figure 17: Main modules of multivariate discretization (MVD) program

Results and Analysis

In this section we will show the results obtained from the multivariate discretization (MVD) program on several data sets, and shows how visualization techniques will help us to determine data complexity, in other words if this data are linearly separable or not. We will use training data set obtained from the multivariate discretization (MVD) program to train a classifier using neural networks architecture, and

test the accuracy of the classifier using testing data set. We will create a single layer perceptron classifiers, and multilayer perceptron classifiers and shows the accuracy obtained from both architectures. Finally we will compare the accuracy results with accuracy results obtained from the other discretization algorithms on different data sets.

The Effect of Discretization

The discretization process has great benefits on data; one of these benefits is that it shows the significant clusters in the data set. Each interval group many data objects on a small region that belongs to the same class. Some data sets may contain outliers, these outliers expose region that doesn't belong this interval. A visualization techniques comes here with a great advantage, in which it facilitate to us to show the clusters generated by discretizing continuous data. Figure 18 depicts a scatter plot of iris data set.

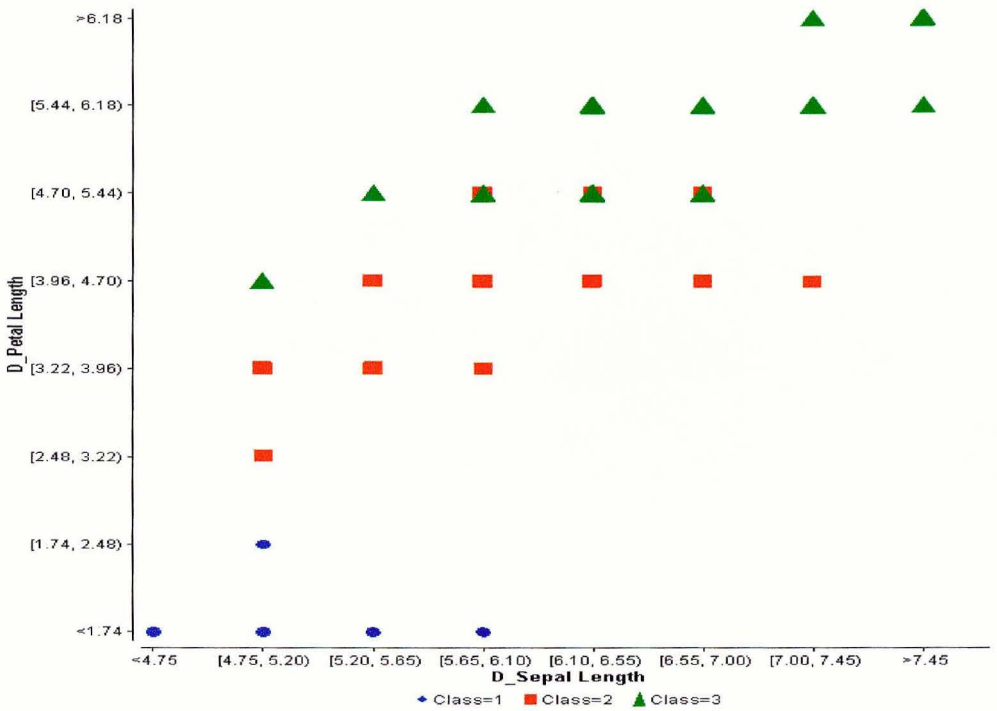

Figure 18: The effect of discretization on continuous data

By analyzing Figure 18, we can find that data nearly grouped to three regions, these three regions equal to the three predefined classes. In addition, we can observe the presence of outliers or inconsistencies in data. We need only two cuts to separate these regions, and these cuts will provide us with the three clusters. Figure 19 depicts the most significant cuts; we can show two horizontal cuts are sufficient to separate data objects.

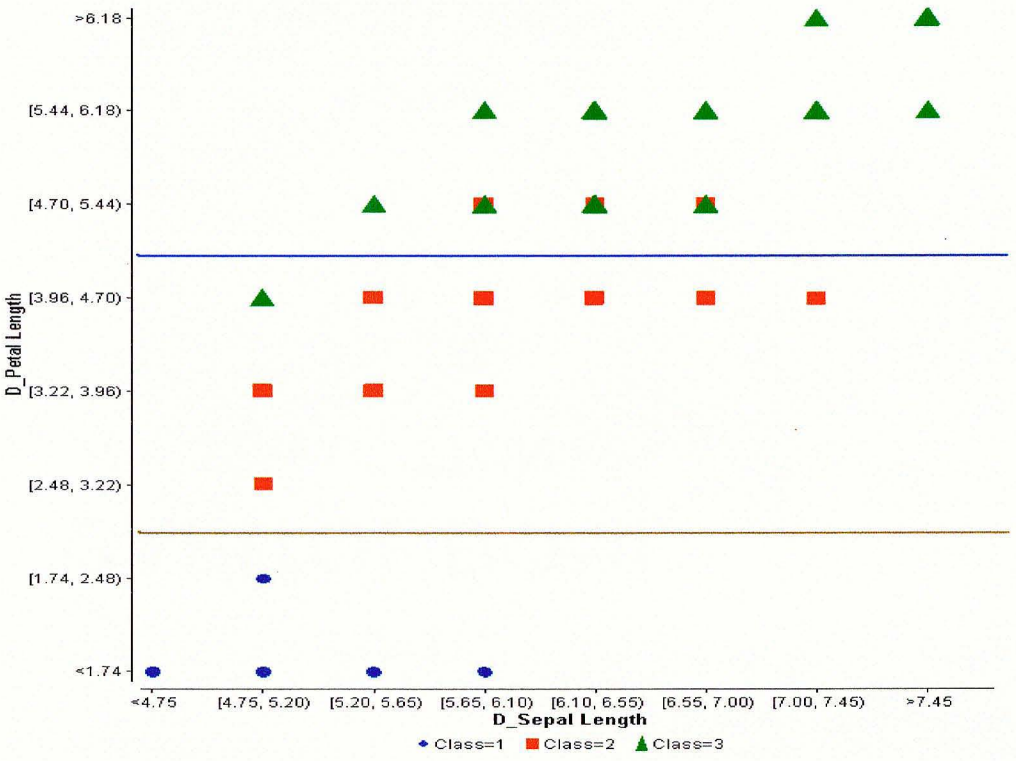

Figure 19: Clusters separated by two cuts

Artificial Neural Networks - Single Layer Perceptron Classifiers

In 1958 Rosenblatt invented the perceptron algorithm to be a simplified model of the biological neuron system. The model consists of a linear combiner followed by an activation function which is hard limiter. The weighted sum of the inputs is applied to the hard limiter, which produces an output equal to +1 if its input is positive and -1 if it is negative. Figure 20 is a general model of the perceptron algorithm for classification problems that have n input values. Where X_1, X_2, X_3,..., X_n, are the input features and

W_1, W_2, W_3,..., W_n, are the weights of the artificial neuron. A single layer perceptron can only separate linearly separable patterns.

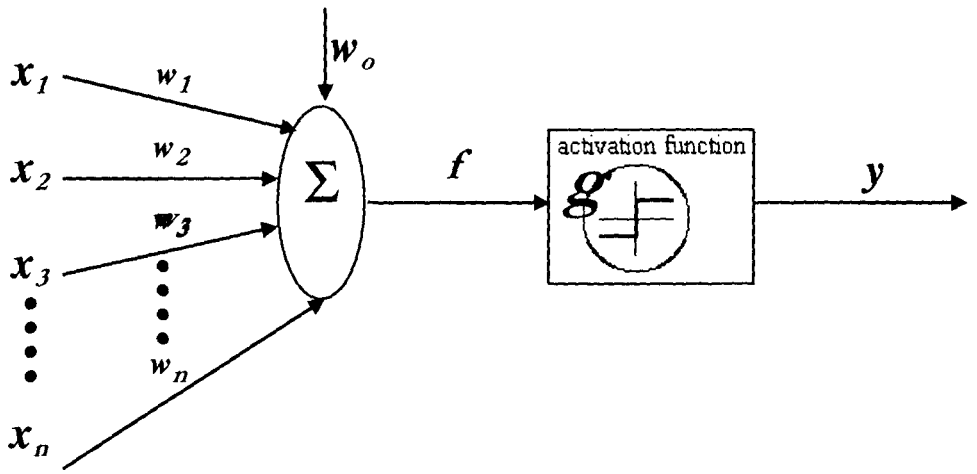

Figure 20: A general model of the perceptron algorithm

The perceptron training algorithm consists of four steps, it aims to find the right classifier that will classify data samples, and the weights could be computed automatically. The steps are as follow:

Step 1: Initialization

Set initial weights W_1, W_2, W_3,..., W_n and threshold θ randomly.

Step 2: Activation

Activate the perceptron by applying inputs $X_1(p)$, $X_2(p)$, $X_3(p)$, ..., $X_n(p)$ and desired output $Y_d(p)$. Calculate the actual output at iteration p=1.

$$Y(p) = \text{step}\left[\sum_{i=1}^{n} X_i(p)\, W_i(p) - \theta\right],$$

Where n is the number of perceptron inputs, and step is a step activation function.

Step 3: Weight Training

Update the weights of the perceptron

$$W_i(p + 1) = W_i(p) + \Delta W_i(p),$$

Where $\Delta W_i(p)$ is the weight correction at iteration p. the weight correction is computed by the delta rule:

$$\Delta W_i(p) = \alpha \times X_i(p) \times e(p)$$

Step 4: Iteration

Increase iteration p by one, go back to step 2 and repeat the process until convergence.

The training process is repeated until achieve convergence, and the accurate weights are learned successfully. After that testing samples applied to the network to be classified correctly.

The perceptron can be modeled as a single layer feedforward networks. This type of networks comprises of two layers, namely the input layer and the output layer. The input layer neurons receive the input signals and the output layer neurons receive the output signals as shown in Figure 21. The synaptic links carrying the weights connect every input neuron to the output neuron not vice versa. Such a network is said to be feedforward in type or acyclic in nature.

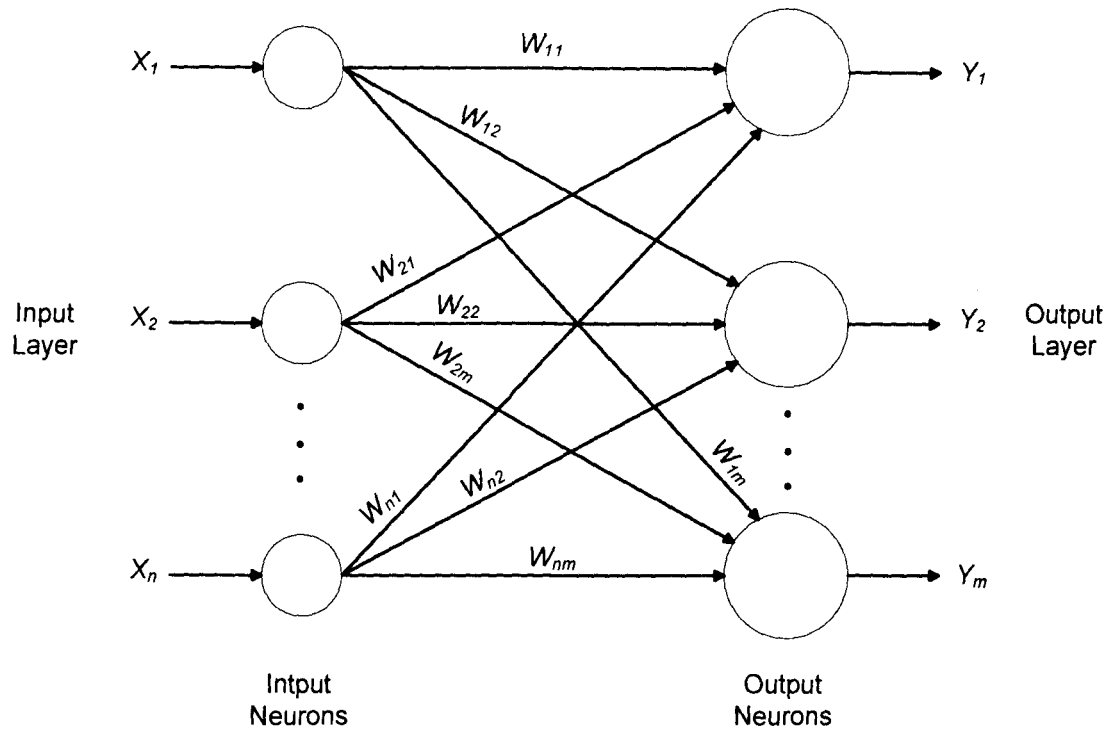

Figure 21: Single layer feedforward network

Despite two layers, the network is termed single layer since it is the output layer, alone which forms computation. The input layer merely transmits the signals to the output layer. Hence, the single layer feedforward network. Figure 21 illustrates an example of a single layer feedforward network.

Learning methods of neural networks can be can be broadly classified into three two types: supervised, and unsupervised. In supervised learning a training set of input patterns (instances) is presented to the network. The output classes are known, and the network computes its output pattern, if there is an error, or in other words a difference between actual and desired output patterns, the weights are adjusted to reduce the error. The perceptron algorithm is an example of supervised learning classifiers.

Artificial Neural Networks - Multilayer Perceptron Classifiers

In the previous section we have seen the single layer perceptron that is contains only two layers namely input layer and output layer. These two layers are not sufficient for the majority of classification problems. We need computational layers that help the classifier to get better performance and produce high accuracy results. In this section we will introduce the multilayer perceptron or multilayer feedforward neural network which is the most efficient neural network architecture for classification problems.

This multilayer perceptron, as its name is made up of multiple layers. Thus, architectures of this class besides processing an input and an output layer also have one or more intermediary layers called hidden layers. The hidden layer aids in performing useful intermediary computations before directing the input to the output layer. The input layer neurons are linked to the hidden layer neuron and the weights on these links are referred to as input-hidden layer weights. Again the hidden layer neurons are linked to the output layer neurons and the corresponding weights referred to as hidden-output layer weights. Figure 22 illustrates a multilayer perceptron network model with one hidden layer.

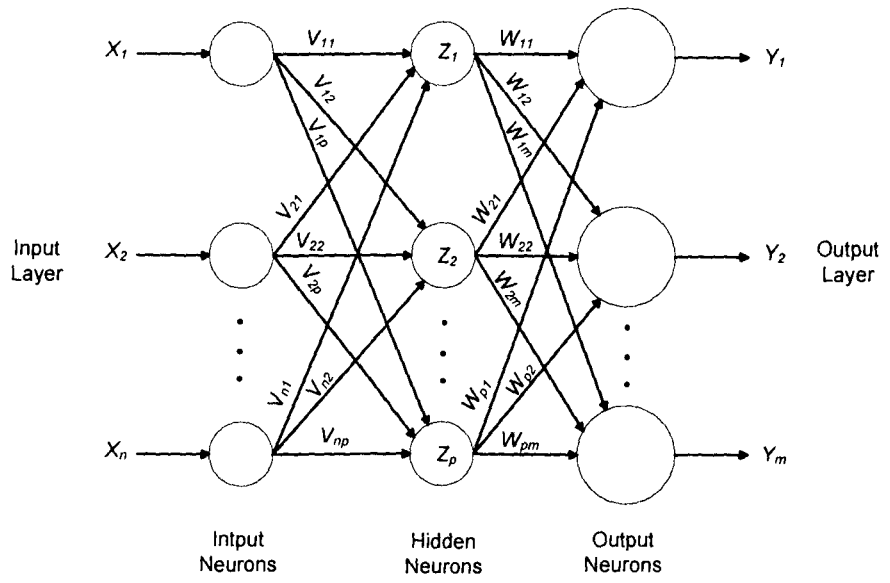

Figure 22: Multilayer perceptron network

When designing a multilayer perceptron networks, we should determine the network topology by determining the number of neurons in the input layer, number of neurons in the output layer, and number of neurons in the hidden layer(s). The number of neurons in the input layer is equal to the number of features in the training data set, and number of neurons in the output layer is equal to the number of classes. The number of hidden layers and number of neurons in each layer is determined by the degree of complexity of the classification problem and trials with more than one topology where the best topology is that which give the highest training accuracy. Actually from heuristics we can say that two hidden layers are sufficient to solve any classification problem.

Multilayer perceptron networks solve the XOR problem by using a combination of perceptrons. On examination, it is clear that this arrangement of perceptrons in layers will be unable to learn. Each neuron in the structure takes the weighted sum of inputs, thresholds it and outputs either (1 or 0) because it uses a hard-limit activation function or

using symmetric hard-limit function which outputs (1 or -1). For the perceptrons in the first layer, the inputs comes from the actual inputs of the problem , while for the perceptrons in the second layer the inputs are outputs of the first layer. The perceptrons on the second layer do not know which of the real inputs from the first layer was (1 or 0).

The actual inputs are effectively masked off from the output layer by the intermediate layer. The two states of neuron being (1 or 0) do not give us any indication of the scale by which we have to adjust the weights. The hard-hitting threshold functions remove the information that is needed if the network is to successfully learn. Hence, the network is unable to determine which of the inputs should be increased and which one should not, so it is not unable to learn to produce a better solution next time.

The solution is to use a nonlinearly (smoothed) function that has a sloping region in the middle that will give us some information on the inputs, we will be able to determine when we need to strengthen or weaken the relevant weights, and the network will be able to learn as required. There are two types that can be used as an activation functions. These functions are: log-sigmoid transfer function and tan-sigmoid transfer function. The sigmoid function is a nonlinear function and helps in modeling linearly inseparable classification problems. This function is very useful when using with the back-propagation algorithm. Figure 23 depicts the tan-sigmoid transfer function and log-sigmoid transfer function.

Learning in multilayer perceptron proceeds the same way as for a perceptron. In single layer perceptron there is only one weight for each input and only one output, but in multilayer perceptron, there are many weights, each of which contributes to more than one output. All the weights that connects the input layer and the hidden layer does not

updated during the learning process, so the back-propagation technique is then used to perform the updating process.

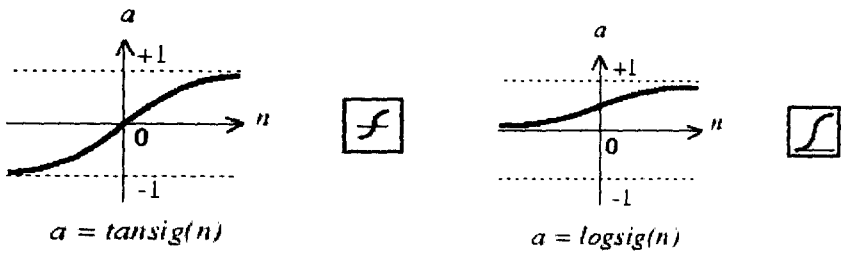

Figure 23: Activation functions of multilayer perceptron with back-propagation

The back-propagation network is a multilayer perceptron network that has three or four layers. These layers are fully connected, that is every neuron in each layer is connected to every other neuron in the adjacent forward layer.

In back-propagation network, the learning process is performed in two phases: First, a training input pattern is presented to the network input layer. The network then propagates the input pattern from layer to layer until the output pattern is generated by the output layer. If the pattern is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated.

As with any other neural networks, a back-propagation is determined by the connections between neurons, the activation functions used by the neurons, and the learning law that specifies the procedure for adjusting weights. The back-propagation algorithm contains the following steps:

Step 1: Initialization

Haykin in 1994 [20] set all the weights and threshold levels of the network to random numbers uniformly distributed inside a small range:

$$\left[-\frac{2.4}{F_i} \quad + \frac{2.4}{F_i} \right],$$

Where F_i is the total number of inputs of neuron i in the network. The weight initialization is done on a neuron-by-neuron basis.

Step 2: Activation

Activate the back-propagation neural network by applying inputs $X_1(p)$, $X_2(p)$, $X_3(p)$, ..., $X_n(p)$ and desired outputs $Y_{d,1}(p)$, $Y_{d,2}(p)$, $Y_{d,3}(p)$, ..., $Y_{d,n}(p)$.

(a) Calculate the actual outputs of the neurons in the hidden layers:

$$Y_j(p) = \text{sigmoid}\left[\sum_{i=1}^{n} X_i(p)\, W_{ij}(p) - \theta_j\right],$$

Where n is the number of inputs of neurons j in the hidden layer, and sigmoid is the sigmoid activation function.

(b) Calculate the actual outputs of neurons in the output layer:

$$Y_k(p) = \text{sigmoid}\left[\sum_{k=1}^{m} X_{jk}(p)\, W_{jk}(p) - \theta_k\right],$$

Where m is the number of inputs of neuron k in the output layer.

Step 3: Weight Training

Update the weights in the back-propagation network propagating backward the errors associated with output neurons.

(a) Calculate the error gradient for the neurons in the output layer:

$$\delta_k = Y_k(p) \times [1 - Y_k(p) \times e_k(p)]$$

Where

$$e_k(p) = Y_{d,k}(p)\ Y_k(p)$$

Calculate the weight corrections:

$$\Delta W_{jk}(p) = \alpha \times Y_i(p) \times \delta_k(p)$$

Update the weights at the output neurons:

$$W_{jk}(p + 1) = W_{jk}(p) + \Delta W_{jk}(p)$$

(b) Calculate the error gradient for the neurons in the hidden layer:

$$\delta_j(p) = Y_j(p) \times [1 - Y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) \times W_{jk}(p)$$

Calculate the weight corrections:

$$\Delta W_{ij}(p) = \alpha \times X_i(p) \times \delta_i(p)$$

Update the weights at the hidden neurons:

$$W_{ij}(p + 1) = W_{ij}(p) + \Delta W_{ij}(p)$$

Step 4: Iteration

Increase iteration p by one; go back to step 2 and repeat the process until the selected error criterion is satisfied.

One of the most effective means to accelerate the convergence of back-propagation learning is to adjust the learning rate parameter during training. The small learning rate parameter α cause's small changes to the weight in the network from one iteration to the next iteration, and thus leads to increase convergence time and smoothed learning curve. On the other hand, if the learning rate parameter α is made larger to speed up the training process, the resulting larger changes in the weights may cause instability and, as a result, the network may become oscillatory.

Jacobs in 1988 [26] proposed two heuristics to accelerate the convergence and yet avoid the danger of instability, these two heuristics are:

- If the change of the sum of squared errors has the same algebraic sign for several consequent epochs, then the learning rate parameter α should be increased.
- If the algebraic sign of the change of the sum of squared errors alternates for several consequent epochs, then the learning rate parameter α should be decreased.

Adapting the learning rate requires some changes in the back-propagation algorithm; first, the network outputs and errors are calculated from the initial learning rate parameter. If the sum of squared errors is at the current epoch exceeds the previous value by more than a predefined ratio (typically 1.04), the learning rate parameter is decreased (typically by multiplying by 0.7) and new weights and thresholds are calculated. However, if the error is less than the previous one, the learning rate is increased (typically by multiplying by 1.05).

Learning rate adaptation can be used together with learning with momentum, where it improves the performance of a multilayer back-propagation network and minimizes the chance that the network can become oscillatory.

The success of a multilayer network is to find the optimum network architecture that gives the highest accuracy. Any given classification problem determines the number of nodes in the hidden layer which have a great impact on the classifier accuracy. It has been experimentally proved that the learning process could be significantly faster when the number of hidden neurons is equal to the number of input patterns, and we suggest that the trial-and-error method is better for selecting number of hidden nodes.

In multilayer network the most important thing is to achieve better accuracy results with testing data set not training data set, because increasing number of hidden nodes with training data sets it gets better accuracy results, but with testing data gets low

accuracy results. By increasing number of hidden nodes, the network performance on testing data set improves significantly.

In the next section we will make practical experiments on several discretized data sets using different topologies, and shows how the number of hidden layers and number of hidden nodes has a great impact on the accuracy results of the network.

Practical Experiments

In this section we will present and discuss results achieved by running single layer perceptron, and multilayer perceptron classifier on each data set. We have designed a network with different architectures for each classification problem in order to find the optimum weights vector that classifies data samples with the least number of errors. There are two data sets training and testing. We have learned the classifier using the training data set, and tested its accuracy using testing data set.

In our experiment we start with a single layer perceptron which is the basic network topology that contains the output layer and shows the error rate. If the error rate is above the accepted value, we use a multilayer perceptron with back-propagation by adding a hidden layer with a number of neurons, and if the error rate is still high we continue increase number of neurons in the hidden layer(s) until number of errors is acceptable. The code will be running twice, because every time weights initialized randomly, this means that learning accuracy may change at each time the code running and gets high training accuracy and low testing accuracy. So we have chosen the topology that gets highest testing accuracy.

The stopping criterion is the point by which the system stops processing the data. We can not guarantee a hundred percent convergences with the classification problems. So we will use some common stopping criteria like:

- Stop after a certain number of runs (epochs) through all the training data.
- Stop when the total sum of the error reaches some low level.

Both criterions will be used through the classification problems.

Cars Classification Problem

In our classification problem, cars data set contains 500 samples divided equally to 250 training samples and 250 testing samples. The data set contains 14 input attributes and a class attributes which classified to 5 classes. Our target is to find the optimum classifier that classifies data into 5 classes as described in section 5.2.

The network topologies for this classification problem starting with input layer that contains 14 neurons that are the number of input attributes, output layer that contains 5 neurons that are the number of output classes, and adding hidden layer with different number of neurons according to problem complexity.

The code has been run twice for training and testing data sets, the training and testing accuracies and number of errors for each experiment on each topology is recorded in table 5.2.

Table 3

Results of Single & Multi-layer Perceptron Classifiers on Cars Data Set

Network Topology				Experiment 1				Experiment 2			
				Training Accuracy		Testing Accuracy		Training Accuracy		Testing Accuracy	
Input layer	Hidden Layer1	Hidden Layer2	Output Layer	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors
14	---	---	5	74.8 %	63	59.2 %	102	75.6 %	61	58.8 %	103
14	10	---	5	86.4 %	34	66.4 %	84	90.8 %	23	73.8 %	67
14	15	---	5	92.4 %	19	72.4 %	69	92.8 %	18	71.2 %	72
14	250	---	5	89.6 %	26	69.2 %	77	88 %	30	69.6 %	76
14	10	15	5	94.4 %	14	77.6 %	56	87.6 %	31	63.6 %	91
14	15	15	5	94.4 %	14	78 %	55	93.6 %	16	76 %	60
14	50	50	5	94.8 %	13	76 %	60	93.2 %	17	76.4 %	59

From Table 3 we can conclude that the cars classification problem is linearly inseparable and complex. The accuracy obtained from running a network twice is different, and this indicates that the randomly initialized weights have a high impact on the learning and testing accuracy. The learning process takes more time as the number of neurons increases.

There is a gap between the training and testing accuracy for most topologies, and the best testing accuracy is achieved by using a topology with two hidden layers, each one has 15 neurons. It is also noticeable that increasing number of neurons in hidden layer(s) does not guarantee increasing the training and testing accuracy.

Glass-Identification Problem

The glass data set has 9 attributes that determines the type of a glass, in addition to the class attribute. The data set contains 214 samples divided equally into training and testing data sets. The class attribute determines either a glass is a type of float or not. The class distribution is 163 samples windows glasses and 51 samples non-window glasses; it has values from 1 to 6.

In our experiment, we have design different network architectures in order to achieve a better training and testing accuracies, we starting with single layer perceptron network namely, output layer that contains 6 neurons. After that we have used a multilayer perceptron network by adding first hidden layer with different number of neurons in the hidden layer. We added a second hidden layer with different number of neurons. Table 4 summarizes results achieved by running different network architectures on glass data set.

In this classification problem, we have used different network topologies; the best testing accuracy is 66.4 % with 36 classification errors by using one hidden layer with 107 neurons. In general, there is no big difference between training and testing accuracies.

Table 4

Results of Single & Multi-layer Perceptron Classifiers on Glass-Identification Data Set

Network Topology				Experiment 1				Experiment 2			
				Training Accuracy		Testing Accuracy		Training Accuracy		Testing Accuracy	
Input layer	Hidden Layer1	Hidden Layer2	Output Layer	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors
9	---	---	6	37.4 %	67	37.4 %	67	40.2 %	64	34.6 %	70
9	6	---	6	35.5 %	69	35.5 %	69	35.5 %	69	35.5 %	69
9	12	---	6	55.2 %	48	56.1 %	47	58 %	45	62.6 %	40
9	24	---	6	61.7 %	41	58.9 %	44	64.5 %	38	57.1 %	46
9	48	---	6	63.6 %	39	64.5 %	38	63.6 %	39	62.6 %	40
9	107	---	6	72 %	30	36.6 %	39	71.1 %	31	66.4 %	36
9	6	12	6	54.3 %	49	62.6 %	40	56.1 %	47	58.9 %	44
9	12	12	6	63.6 %	39	57.1 %	46	35.5 %	69	35.5 %	69
9	24	24	6	63.6 %	39	62.6 %	40	57.9 %	45	57 %	46

The accuracy results obtained is not high in comparison with results obtained in other classification problems, and this is due to two reasons, the first is the existence of

missing values in data set. Missing values has a great impact on the resulting accuracy. The second reason is that classes has not the same distribution, in other words, there are 163 samples windows glasses, and only 51 samples non-window glasses.

Heart Classification Problem

The heart (Cleveland) data set contains 13 biological and medical attributes, in addition to the class attribute that indicating the diagnosis of heart condition. The data set contains only 5 continuous attributes. The class attribute has values from 1 to 5. The data set contains 303 samples, distributed as: (164 samples for class 1), (55 samples for class 2), (36 samples for class 3), (35 samples for class 4), (and 13 samples for class 5).

In heart classification problem, we have designed several network topologies to achieve the best training and testing accuracies. Table 5 summarizes results obtained by running different network architectures on heart data set.

Table 5

Results of Single & Multi-layer Perceptron Classifiers on Heart Data Set

Network Topology				Experiment 1				Experiment 2			
				Training Accuracy		Testing Accuracy		Training Accuracy		Testing Accuracy	
Input layer	Hidden Layer1	Hidden Layer2	Output Layer	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors
13	---	---	5	54.3 %	69	53.6 %	70	54.3 %	69	53.6 %	70
13	5	---	5	54.3 %	69	54.3 %	69	53.6 %	70	53.6 %	70
13	10	---	5	54.3 %	69	53.6 %	70	54.3 %	69	53.6 %	70
13	25	---	5	53.3 %	69	53.6 %	70	66.2 %	51	55.6 %	67
13	50	---	5	63.6 %	55	57 %	65	61.6 %	58	56.3 %	66
13	75	---	5	68.2 %	48	59.6 %	61	64.9 %	53	58.3 %	63
13	100	---	5	84.1 %	24	57.6 %	64	62.3 %	57	60.9 %	59
13	151	---	5	54.3 %	69	53.6 %	70	89.4 %	16	57.6 %	54
13	10	15	5	54.3 %	69	53.6 %	70	54.3 %	69	53.6 %	70
13	25	25	5	55.6 %	67	51.7 %	73	58.3 %	63	56.3 %	66
13	50	50	5	62.9 %	53	55 %	68	58.3 %	63	59.6 %	61
13	75	75	5	82.8 %	26	60.8 %	60	63.6 %	55	58.3 %	63
13	100	125	5	72.9 %	41	60.9 %	59	74.2 %	39	58.3 %	63

In our experiments, we have started with a single layer perceptron classifier then using a multilayer perceptron classifier by adding the first hidden layer with increased number of neurons. After that we add a second hidden layer, and using different topologies with different number of neurons in each hidden layer. The best testing accuracy is 60.9 % with 59 classification errors achieved by using a topology with one hidden layer with 100 neurons, and also with a topology that has two hidden, the first hidden layer has 100 neurons, and the second hidden layer has 125 neurons.

From Table 5, we can observe that the testing accuracy in most experiments is low, that is because data set contains missing values, and class distribution is not equal among all samples.

Iris Classification Problem

The Iris data set contains 150 samples divided equally into training and testing data sets. The data set contains 4 attributes and a class attribute classified into 3 classes. So the input layer of the network contains 4 neurons and the output layer contains always 3 neurons.

In our experiment, we have built several network topologies in order to find the best topology that achieves better training and testing accuracies. We have started with a single layer perceptron namely the output layer with 3 neurons, and adding a hidden layer with simple number of neurons, and continually add neurons to the hidden layer until network convergence.

From table 6 we can conclude that, better testing accuracy is 98.7 with 1 classification error achieved by using one hidden layer with 6 neurons, we do not need to use a second hidden layer. There is no big difference between the training and testing

accuracies. The learning process is quite fast because the network topology is not complex and the size of the training data set is not large. Is it known that as the network topology used gets more complex, the learning process takes more time.

Table 6

Results of Single & Multi-layer Perceptron Classifiers on Iris Data Set

Network Topology				Experiment 1				Experiment 2			
				Training Accuracy		Testing Accuracy		Training Accuracy		Testing Accuracy	
Input layer	Hidden Layer1	Hidden Layer2	Output Layer	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors
4	---	---	3	98.7 %	1	97.3 %	2	98.7 %	1	93.3 %	5
4	3	---	3	100 %	0	79.3 %	2	97.3 %	2	94.6 %	4
4	6	---	3	98.6 %	1	93.3 %	5	100 %	0	98.7 %	1
4	75	---	3	100 %	0	96 %	3	100 %	0	94.7 %	4

Liver-Disorders Classification problem

The liver data set is a blood test of liver disorders that may arise from excessive alcohol consumption. The data set contains 6 attributes, in addition to the class attribute, which has only two values (1 or 2). The data set contains 345 samples divided into training and testing data sets. A full explanation of liver data set founded in section 5.2.

In our experiments, we have built several network topologies in order to find the best topology that achieves better training and testing accuracies. We have started with a single layer perceptron classifier, then using a multilayer perceptron classifier by adding the first hidden layer with increased number of neurons. After that we add a second hidden layer, and using different topologies with different number of neurons in each hidden layer.

Table 7

Results of Single & Multi-layer Perceptron Classifiers on Liver-Disorders Data Set

Network Topology				Experiment 1				Experiment 2			
				Training Accuracy		Testing Accuracy		Training Accuracy		Testing Accuracy	
Input layer	Hidden Layer1	Hidden Layer2	Output Layer	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors	Accuracy Rate (%)	Num of Errors
6	---	---	2	68.6 %	54	62.2 %	65	68.1 %	55	66.3 %	58
6	2	---	2	58.2 %	72	58.2 %	72	58.2 %	72	58.2 %	72
6	6	---	2	67.4 %	56	64 %	62	68.6 %	54	62.2 %	65
6	12	---	2	73.8 %	45	67.4 %	56	72.1 %	48	69.2 %	53
6	24	---	2	81.4 %	32	69.2 %	53	76.2 %	41	69.8 %	52
6	48	---	2	86.1 %	24	63.4 %	63	84.3 %	27	62.8 %	64
6	64	---	2	84.9 %	26	63.4 %	63	87.8 %	21	61.1 %	67
6	128	---	2	90.7 %	16	58.7 %	71	89 %	19	59.3 %	70
6	172	---	2	91.9 %	14	58.2 %	72	88.4 %	20	58.7 %	71
6	12	24	2	77.3 %	39	69.1 %	53	77.9 %	38	65.1 %	60
6	24	48	2	82.6 %	30	61.6 %	66	78.5 %	37	65.1 %	60
6	64	64	2	89.5 %	19	57.6 %	73	89.5 %	18	61.1 %	67
6	64	98	2	89.5 %	18	61.1 %	67	89 %	19	58.2 %	72

Table 7 summarizes results obtained by running training and testing data sets using different network topologies. The best testing accuracy is 58.9 % with 62 classification errors, achieved by using a topology with one hidden layer that has 50 neurons.

From Table 7 we can observe that, for each experiment, the training and testing accuracies are closed to each other. The liver data set is complex and linearly inseparable, so that the training and testing accuracy in all experiments is low.

Comparing Results of Discretization Algorithms

In this chapter, we have made practical experiments on a number of data sets. These data sets passed from different phases, starting with a data cleaning process. After that data are examined to choose continuous value attributes to be converted to discrete one by applying it to a discretization algorithm. Discrete data set is divided to two sets,

namely training and testing data sets. The training data set is used to learn the classifier, and testing data set is used to test the accuracy of the classifier.

In our experiments, we have chosen different types of discretization algorithms. These algorithms are: Naïve discretization, Entropy discretization, ChiMerge discretization, Orthogonal Hyperplanes discretization, and Multivariate Discretization (MVD) algorithms. Each discretization algorithm is used to discretize each data set. After discretizing data set, these data is divided into training and testing sets.

In this section, we will discuss results obtained by applying each data set on a classification algorithm. The testing accuracy of each data set on the different discretization algorithms are presented in Table 8.

Table 8

Summary of Results from Comparing Different Discretization Algorithms on Experimental Data Set

Data Set	Size	Num	Class	Majority Accuracy (%)				
				Naïve Discretization	Entropy Discretization	ChiMerge Discretization	Orthogonal Hyperplanes	Multivariate Discretization
Glass	214	9	7	60 %	70 %	52 %	63 %	66.4 %
Heart	303	7	5	44 %	59 %	44 %	67 %	60.9 %
Iris	150	4	3	76 %	97 %	92 %	93 %	98.7 %

We can observe that, the entropy discretization algorithm gives the best accuracy result with the glass data set, the multivariate discretization (MVD) also gives a good result and near to the entropy discretization.

The orthogonal hyperplanes achieves the best accuracy result with heart data set among all discretization algorithms. There is a big gap between the accuracy results

obtained from different algorithms, and multivariate discretization (MVD) is the closest one to the orthogonal hyperplanes algorithm.

Multivariate discretization (MVD) and Entropy discretization algorithms give the best accuracy results with iris data set, and naïve discretization algorithm gives the lowest accuracy result in comparing to other algorithms.

In general, we can say that, not all discretization algorithms will produce a better accuracy results with all data sets. Some discretization algorithms may produce better accuracy results with some data sets and a low accuracy results with other data sets. Through the experiments done in this thesis, we can observe that, the selection of the best discretization algorithm that will produce good performance on a specific data set depends on trying more than one algorithm to be able to choose the right one.

The supervised learning methods are slightly better than the unsupervised methods, although a good discretization method tends to significantly increase the performance of the Naïve-Bayesian classifier for continuous attributes.

Summary

In this chapter, we have focused on evaluating our proposed Multivariate discretization algorithm against other common discretization algorithms: Naïve, Entropy, Chimerge, and Orthogonal Hyperplanes. Using a real world data sets that varying in size and complexity. We have given a detailed explanation of each data set.

The multivariate discretization (MVD) program contains four main modules. These modules perform the main functionality of the algorithm. These modules are: opening text file or access data base table, viewing continuous data, perform discretization process, and create discrete files. The program creates three text files: a

discrete file with the same number of instances as the continuous file, and other two files one contains odd instances for training and the other contains even instances for testing.

We have made practical experiments on five real world data sets that varying in size and complexity in order to measure the performance of each discretization algorithm. These practical experiments are done in two steps. The first step is to examine and review each data set to choose the continuous value attributes. These continuous attributes are selected to be discretized using multivariate discretization (MVD) algorithm and other discretization algorithms. The output of the discretization algorithm will be divided into two sets one for training and the other for testing, by talking odd instances for training, and even instances for testing. The second step is to classify data samples using a classifier. We have chosen the artificial neural network-Single layer perceptron, and multilayer perceptron with back-propagation algorithm. We have started our experiments on each data set with a single layer perceptron and then using a multilayer perceptron by adding the first hidden layer with simple number of neurons, and increase number of neurons as long as error rate is not acceptable. After that we add the second hidden layer and continually increase number of neurons to achieve the best training and testing accuracies. In general, we have used different network topologies in order to achieve network convergence.

Through our experiments, we have found that the training and testing accuracies are subject to the complexity of the given classification problem, in other words, is the given data is linearly separable or inseparable. Not all discretization algorithms produce good results with all types of data set, we must try more than one discretization algorithm

on the same data set in order to choose the algorithm that produces the best accuracy results.

We have compared the results obtained with multivariate discretization (MVD) with other discretization algorithms, and we have found that it produces good results in comparing with other discretization algorithms.

CHAPTER VI

CONCLUSION AND FUTURE WORK

Conclusion

In this thesis we have compared different discretization algorithms and measure its accuracy on many data sets. The kernel issues of this thesis are to concentrate on the importance of discretization and how it can improve the accuracy of the learning algorithm. Since many learning algorithms can not handle continuous values, discretization solves this problem by converting continuous data type to nominal data type.

In chapter two we have given a comprehensive background of the data preparation process which is a complicated process that encompasses many steps. Data preparation is an important step of the data mining process that influences the result and accuracy of the data mining technique. In practice, it has been generally founded that data cleaning and preparation takes approximately eighty percent of the total engineering effort. Discretization is one of the data reduction techniques which are part of the data preparation process.

In chapter three we have reviewed the importance of data preparation at different aspects, from which real-world data is incomplete. Data preparation generates a reduced version of the original data set, from this version we can extract quality pattern. Data preparation is a complicated task which comprises many steps. We have focused on data

reduction techniques which one of them is discretization. Discretization is a data processing procedure that transfers one type of data into another type of data. In the existing literature, there is a considerable variation in the terminology that used to address each of these data types. Discretization methods falls into either primary or composite, primary methods accomplish discretization without reference to any other discretization method. Composite methods are built on top of a primary method. A typical discretization process consists of four steps: sorting values, getting cut point, splitting or merging according to some criterion, and finally stopping at some point. Discretization methods can be putted in a hierarchical framework according to splitting and merging approaches. In our literature we found that there exist several discretization methods that have been proposed to discretize continuous data in the research area of machine learning that is belonging to different categories. Feature selection and sampling are also important feature reduction which can be used simultaneously with discretization. Many discretization algorithms have been proposed and tested to show that discretization helps improve the performance of learning and helps understand the learning result. We have selected some of these algorithms for experiment as: Naïve, Entropy, ChiMerge, Orthogonal Hyperplanes, and multivariate discretization (MVD) algorithm.

In chapter four we have introduced our proposed system for discretization that encompasses several discretization algorithms that belong to different categories. Data sets will be passed from a data cleaning step before applying to the discretization algorithm. After data has been discretized and saved in a file, the data set will be divided into training and testing. Training data can be applied to a classification or clustering

algorithm for learning, testing data will be used to test the accuracy of the classified or clustered system.

We have introduced an implementation for multivariate discretization (MVD) algorithm. By using the word multivariate, we mean the implementation of the discretization process by quantifying multiple continuous attributes at the same time. The merging process is the core of the discretization process that influences the quality of the discretization algorithm where two adjacent intervals merged only when the instances in those intervals have similar distributions.

We have used data visualization as a preliminary investigation step for determining level of consistency of data, in other words, is the data set contains outliers. This will help choosing the appropriate discretization algorithm. Another benefit of visualization is that after data converted to discrete one, data visualization will show extracted patterns from data, and how much these patterns meaningful.

In chapter five we have implemented the multivariate discretization (MVD) algorithm on real world data sets. These data sets represent application with hundreds of instances. The mvd program converts each continuous data type to a discrete one, and divides discrete data into two sets by taking odd instances as training data set and even instances as testing data set. We have classified data samples using artificial neural network-single layer perceptron, and multilayer perceptron with back-propagation algorithm. We have run our experimental data sets on a single layer perceptron and multilayer perceptron. We have designed different topologies using multilayer perceptron, show the training and testing accuracies of each topology in order to find the network topology that achieves the best training and testing accuracies. Some data sets

give better accuracy results and the other give low accuracy results, and this is subject to the complexity of the classification problem.

We have compared the results obtained with multivariate discretization (MVD) with other discretization algorithms, and we have found that it produces good results in comparing with other discretization algorithms.

Future Work

The area of discretization is an area in which research in different fields can be applied. Several techniques from pattern recognition, statistics, and other disciplines can be used in developing and improving discretization process. This is a large area of research with a lot of work to be done.

Two factors can affect the discretization accuracy. These factors are the decision boundary and the error tolerance. Both depend on the probability distribution of the training data, therefore the more we know about the data distribution, the better our discretization can be.

Discretization algorithms can not be applied to all types of data, they produce high accuracy results with some data sets and low accuracy with others, thus discretization process can be done by trying the same data with more than one discretization algorithm and choose the algorithm which produces the highest accuracy.

Several experiments can be carried out with different measures for entropy of a data set, possibly improving the results of the Entropy based discretization algorithm.

The multivariate discretization (MVD) algorithm is a good one and needs more work to produce good results with most data sets. We will continue improving all the proposed system to be a data independent system which can discretize all types of data.

For all the algorithms presented in this thesis, their implementation can be optimized with regards to speed.

REFERENCES

[1] Bay, S.D. (1999). *The UCI KDD archive*. Department of information and computer science, University of California, Irvine. At http://www.kdd.ics.uci.edu.

[2] Bay, S.D. (2000) *Multivariate discretization of continuous variables for set mining*. In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[3] Berry, M.J., & Linoff, G.S. (2000). *Mastering Data Mining*. John Wiley & Sons, Inc.

[4] Blake, C. L, & Merz, C. J. (1998). *UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/mlrepository.html]*. Department of Information and Computer Science, University of California, Irvine.

[5] Cantú-Paz, E. (1998). *Supervised and Unsupervised Discretization Methods for Evolutionary Algorithms*. Lawrence Livermore national Laboratory.

[6] Catlett, J. (1991*b*). *On changing continuous attributes into ordered discrete attributes*, in Y. Kodratoff,ed., Proceedings of the European Working Session on Learning, Berlin, Germany: Springer Verlag.

[7] Cerquides, J., & Mantaras, R.L. (1997). *Proposal and empirical comparison of parallelizable distance-based discretization method.* In KDD 1997: Third International Conference on Knowledge Discovery and Data Mining.

[8] Chao, S., & Li, Y.P. (2005). *Multivariate interdependent discretization in discovering the best correlated attribute*. Faculty of science and technology, University of Macau, China. WIT Press.

[9] Chmielewski, M.R, & Grzymala-Busse, J.W. (1994). *Global discretization of Continuous attributes as preprocessing for machine learning*. In third international workshop on Rough Sets and Soft Computing.

[10] Divina, F., Keijzer, M., & Marchiori, E. (1999). *Evolutionary Concept Learning with Constraints for Numerical Attributes.* Department of Computer Science, Vrije University. Amsterdam.

[11] Dougherty, J., Kohavi, R., & Sahami, M. (1995). *Supervised and unsupervised discretization of continuous features.* In Proceedings of the 12th International Conference on Machine Learning.

[12] Fayyad, U. M, & Irani, K.B. (1993) *Multi-interval discretization of continuous-valued attributes for classification learning.* In proceedings of the Thirteenth international joint Conference on Artificial Intelligence. Morgan Kaufmann.

[13] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). *From Data Mining to Knowledge Discovery:* An Overview, In Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, U., *Advances in Knowledge Discovery and Data Mining.* AAAI Press/MIT Press.

[14] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). *Knowledge Discovery and Data Mining: Towards a Unifying Model.* Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, August 2-4. AAAI Press.

[15] Fayyad, U., Grinstein, G., & Wierse, A. (2001). *Information Visualization In Data Mining and Knowledge Discovery.* 300 p.

[16] Frank, E., & Witten, I.H. (1999). *Making better use of discretization.* In proceedings of the 16th International Conference on Machine Learning. Morgan Kaufmann publishers.

[17] Giudici, P. (2003). *Applied Data Mining.* John Wiley & Sons, Ltd. ISBN (0-470-84678-X).

[18] Grinstein, G.G, & Ward, M.O. (2002). *Introduction to Data Visualization.* In Fayyad et al. (2002), chapter 1, pages 21-45.

[19] Han, J., & Kamber, M., (2001). *Data Mining: Concepts and Techniques.* Simon Fraser University. Morgan Kaufmann publishers. ISBN (1-55860-489-8).

[20] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* Macmillan College Publishing Company, New York.

[21] Hibino, S.L. (1999). *A Task-Oriented View of Information Visualization.* In Proceedings of Computer-Human Interaction (CH199). Pages 178-179 ACM.

[22] Ho, K.M., & Scott, P.D., (1997). *Zeta: A global method for discretization of continuous variables.* In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining.

[23] Holte, R.C. (1993). *Very simple classification rules perform well on most commonly used datasets.* Machine Learning 11.

[24] Hsu, C.-N., Huang, H.-J., & Wong, T.-T. (2000). *Why discretization works for Naïve Bayesian classifiers.* In Proceedings of the 17th International Conference on Machine Learning.

[25] Hsu, C.-N., Huang, H.-J., & Wong, T.-T. (2003). *Implications of the Dirchlet assumption for discretization of continuous variables in naïve Bayesian classifiers. Machine Learning.* In press.

[26] Jacobs, R.A. (1988). *Increased Rates of Convergence through Learning Rate Adaptation.* Neural Networks1, pp.295-307.

[27] Kamber, M., & Shinghal, R. (1996). *Evaluating the interestingness of characteristic rules.* In proceedings of the second international conference on data mining (KDD-96), AAAI Press.

[28] Keim, D. (2002). *Information Visualization and Visual Data Mining.* IEEE transactions on visualization and computer graphics, vol. 7, no. 1.

[29] Kerber, R. (1992). *ChiMerge: Discretization for numeric attributes.* In National Conference on Artificial Intelligence (1992), AAAI Press/ The MIT Press.

[30] Kohavi, R. (2000). *Data Mining and Visualization.* In National Academy of Engineering (NAE), US Frontiers of Engineering.

[31] Kohavi, R., & Sahami, M. (1996). *Error-based and entropy-based discretization of continuous features.* In proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining.

[32] Liu, B., & Hsu, W. (1996). *Post-analysis of learned rules.* In proceedings of the Thirteenth National Conference on Artificial Intelligence. AAAI Press / The MIT Press.

[33] Liu, H., Hussain, F., Tan, C.L., & Dash, M. (1999). *Discretization: An Enabling Technique.* Technical report, School of Computing, National University in Singapore, Singapore. Kluwer Academic Publishers.

[34] Liu, H., & Motoda, H. (1998). *FEATURE SELECTION for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers. ISBN (0-7923-8198-X).

[35] Liu, H., & Setiono, R. (1995). *Chi2: Feature Selection and Discretization of Numeric Attributes.* In proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence, November 5-8. J.F. Vassilopoulos (Ed.). Herndon, Virginia, IEEE Computer Society.

[36] Li, J., Cercone, N. (2003). *Empirical Analysis on the Geriatric Care Data Set Using Rough Sets Theory.*

[37] Mannila, H., & Toivonen, H. (1996). *Discovered generalized eposodes using minimal occurrences*. In 2nd International Conference on Knowledge Discovery and Data Mining. Portland, Oregon.

[38] Monti†, S., & Cooper†‡, F. G. (1999). *A latent variable model for multivariate discretization*. University of Pittsburgh, Pa – 15260.

[39] Moxon, B. (1996). *Defining Data Mining. DBMS ONLINE, DBMS Data warehouse supplement.*

[40] Murphy, P.M. (1997). *UCI Repository of Machine Learning Databases and Domain Theories*. At http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.

[41] Pawlak, Z. (1991). *Rough Sets: Theoretical aspects of Reasoning about data.* Dordrecht: Kluwer.

[42] Pazzani, M. J. (1995). *An iterative improvement approach for the discretization of numeric attributes in Bayesian classifiers*. In proceedings of the 1st international conference of Knowledge Discovery and Data Mining.

[43] Periklis, A. (2002). *Data Clustering Techniques*. Qualifying Oral Examination Paper. Department Computer Science, University of Toronto.

[44] Perner, P., & Trautzsch, S. (2000). *A comparison of Multi-Interval Discretization Methods for Decision Tree Learning*. Institute of Computer Vision and Applied Computer Sciences e.V.

[45] Pfahringer, B. (1995). *Compression-based discretization of continuous attributes.* In A. Prieditis & S. Russell, eds. Proceedings of Twelfth international Conference on Machine Learning. Morgan Kaufmann.

[46] Piatetsky-Shapiro, G. (1991). *Discovery, analysis, and presentation of strong rules*. In Piatetsky-Shapiro, G. and Frawley, W.J., editors, *Knowledge Discovery in Databases*. AAAI / The MIT Press.

[47] Pyle, D. (1999). *Data Preparation for Data Mining*. Morgan Kaufmann Publishers.

[48] Quinlan, J.R. (1993). C4.5: *Programs for machine learning*. Morgan Kaufmann Publishers.

[49] Richard, M., & Rossotto, M. (1995). *Class-driven statistical discretization of continuous attributes (extended abstract)*. In European Conference on Machine Learning.

[50] Øhrn, A., & Komorowski, J. (1997). *ROSETTA – A Rough Set Toolkit for Analysis of Data.* In Proceedings of the 3rd International Joint Conference on Information Sciences, Durham, NC, USA. Mar. 1 -5. Vol. 3, pp. 403 – 407.

[51] Rissanen, J. (1986). *Stochastic complexity and modeling.* Ann. Statist 14, 1080 – 1100.

[52] Risvik, M, K. (1997). *Discretization of Numerical Attributes: Preprocessing for Machine Learning.* N-7034 Trondheim, Norway.

[53] RØed, G. (1999). *Knowledge Extraction from Process Data: A Rough Set Approach to Data Mining on Time Series. Master's thesis.* IDI NTNU.

[54] Silberschatz, A., & Tuzhilin, A. (1996). *What makes patterns interesting in knowledge discovery systems.* IEEE Trans. on Knowledge and Data Engineering.

[55] Skowron. A., & Son, N.H. (1995). *Quantization of real value attributes: Rough Set and Boolean Reasoning approach.* In Proceedings of the second International Joint Conference on Information Sciences, Wrightsville Beach, NC, USA.

[56] Son, N.H., & Hoa, N.S. (1997). *Some efficient algorithms for Rough Set methods.*

[57] Tay, E.H, F., & Shen, L. (2002). *A Modified Chi2 Algorithm for Discretization.* Department of Mechanical Engineering, National University of Singapore. IEEE Transactions on Knowledge and Data Engineering, VOL. 14, NO.3, MAY/JUNE 2002.

[58] Ting, K.M. (1994). *Discretization of continuous-valued attributes and instance-based learning.* Technical Report 491, University of Sydney.

[59] Tufte, R.E. (2001). *The Visual Display of Quantitative information.* Graphics Press, 2nd Edition, ISBN 0961392142.

[60] Ward, M. (1997). *Overview of Data Visualization.* WPI CS Department.

[61] Wong, P.C. (1999). *Visual Data Mining*, IEEE Computer Graphics and Applications, 19(5):20 – 21.

[62] Yang, Y., & Webb, G.I. (2002). *A Comparative Study of Discretization Methods for Naïve-Bayes Classifiers.* In proceedings of PKAW 2002, The 2002 Pacific Rim Knowledge Acquisition Workshop, Tokyo, Japan.

[63] Yang, Y. (2003). *Discretization for Naïve-Bayes Learning.* A thesis submitted to the degree of Doctor of Philosophy to the school of Computer Science and Software Engineering of Monash University.

[64] Zhang, S., Zhang, C., & Yang, Q. (2003). *Data Preparation for Data Mining*. Applied Artificial Intelligence. Taylor & Francis.

[65] Zemake, S., & Rams *, M (2003). *Multivariate Feature Coupling and Discretization*. Institute of Mathematics, Polish Academy of Sciences, Poland.

[66] See [http://magix.fri.uni-lj.si/orange].

[67] See [http://www.sgi.com/software/mineset/].

CURRICULUM VITAE

NAME: Ehab Ahmed El Sayed Ahmed

ADDRESS: 23 Mahmood Hassenin St, Khalousy, Shoubra.

DOB: Shoubra, Cairo – October 03, 1978

EDUCATION & TRANING:

B.S., Computer Science & Information Systems
October 6 University
1997 -2001