2008

A new approach of top-down induction of decision trees for knowledge discovery

Jun-Youl Lee
Iowa State University

A new approach of top-down induction of decision trees for knowledge discovery

by

Jun-Youl Lee

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Industrial Engineering

Program of Study Committee:
Sigurdur Olafsson, Major Professor
John Jackman
Sarah Ryan
Shashi K. Gadia
Prem G. Premkumar

Iowa State University

Ames, Iowa

2008

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Top-down induction of decision trees is the most popular technique for classification in the field of data mining and knowledge discovery. Quinlan developed the basic induction algorithm of decision trees, ID3 (1984), and extended to C4.5 (1993). There is a lot of research work for dealing with a single attribute decision-making node (so-called the *first-order* decision) of decision trees. Murphy and Pazzani (1991) addressed about multiple-attribute conditions at decision-making nodes. They show that higher order decision-making generates smaller decision trees and better accuracy. However, there always exist NP-complete combinations of multiple-attribute decision-makings.

We develop a new algorithm of *second-order* decision-tree inductions (SODI) for nominal attributes. The induction rules of first-order decision trees are combined by 'AND' logic only, but those of SODI consist of 'AND', 'OR', and 'OTHERWISE' logics. It generates more accurate results and smaller decision trees than any *first-order* decision tree inductions.

Quinlan used information gains via VC-dimension (Vapnik-Chevonenkis; Vapnik, 1995) for clustering the experimental values for each numerical attribute. However, many researchers have discovered the weakness of the use of VC-dim analysis. Bennett (1997) sophistically applies support vector machines (SVM) to decision tree induction. We suggest a heuristic algorithm (SVMM; SVM for Multi-category) that combines a TDIDT scheme with SVM. In this thesis it will be also addressed how to solve multiclass classification problems.

Our final goal for this thesis is IDSS (Induction of Decision Trees using SODI and SVMM). We will address how to combine SODI and SVMM for the construction of top-down induction of decision trees in order to minimize the generalized penalty cost.

CHAPTER 1. INTRODUCTION: DATA MINING AND KNOWLEDGE DISCOVERY

Data mining and knowledge discovery in databases one of fast growing and widely applying interdisciplinary fields such as statistics, databases, pattern recognition and learning machines, artificial intelligence, decision support system, data warehousing, optimization, visualization, and high performance and parallel computing. The recent attention is gradually increased because many people can inexpensively build and easily access their databases over Internet. Currently the success of database systems becomes important roles of many activities in education, science, business, politics, public services, and government.

With the widespread use of databases and the tremendous growth in their sizes, individuals and organizations are faced with the problem of making proper use of this data. A large database means a large body of information that is presumed to be valuable. The current interfaces between humans and commercial database systems, however, do not support more intelligent navigation, summarization, classification, association, or visualization of large databases. Providing these types of capabilities and more is the goal of the emerging research area of data mining and knowledge discovery in databases.

1. Introduction to data mining and knowledge discovery

Advances in data collection, storage, and distribution have motivated the necessary of computational tools and techniques for data analysis. Most of the information is in its raw form so called as data. If data is characterized as recorded facts, then information is the set of patterns, or expectations, that underlie the data. There is a huge amount of information locked up in databases, whose information is potentially important, but has not yet been discovered or articulated.

The term of "*data mining*" can be simply defined as the extraction processes of useful information from databases. According to Witten and Frank (1999), *data mining* is *the extraction of implicit, previously unknown, and potentially useful information from data*. The

basic idea is to build a computer program that automatically shifts through databases and seeks regularities or patterns fro them. However, it is not easy to build a computer program since many patterns or information in database may be not interesting or less important, the information may be not perfect (some record fields are missing), or some information can be unauthentic or dependent on accidental coincidences in particular dataset used. Algorithms need to be robust enough to cope with imperfect data and to extract regularities that are not exact but useful. Machine learning provides the technical basis of data mining. It is used to extract information from the raw data in databases.

The term of "*Knowledge Discovery in Databases*" (KDD) was mentioned at the first KDD workshop in 1989 (Piatetsky-Shapiro and Frawley, 1991) to emphasize that *knowledge* is the end product of a data-driven discovery. According to Fayyad *et al.* (1996b), *knowledge discovery in databases* is defined as the *overall process* of discovering useful *knowledge* from data while *data mining* refers to a particular *step* in this process. *Data mining* is the application of specific algorithms for extracting patterns from data. The additional steps in the *KDD* process, such as data preparation, data cleaning, incorporating appropriate prior knowledge, and proper interpretation of the results, are essential to ensure useful knowledge derived from the data.

Data is defined as a set of facts (or, cases in a database) and structure refers to either patterns or models. A pattern is an expression representing a prudent description of a subset of the data. A model is a representation of the source generating the data. The term of "process" implies that *KDD* is comprised of many steps that involve data preparation, search for patterns, knowledge evaluation, and refinement, all potentially repeated in multiple iterations.

In this thesis we define *data mining* as a process undertaken to *extract* any *content information* from large datasets *to provide a compact summary* of a dataset. Content information comes in many forms and can be applied in many ways. Unsupervised and supervised learning techniques are generally used to interpret the information contained in large datasets. An unsupervised learning algorithm aims to classify and identify interesting patterns in data, using techniques such as singular value decomposition or projection pursuit,

when input-output pairs are not available. On the other hand, supervised learning algorithms can be employed when robust training sets, characterizing the behavior of a particular system, are available. In both cases, feature analysis and clustering techniques can be used to reduce the dimensionality of the parameter space.

The utilization of data mining does not gradually required to learn such a high level of statistical education or disciplines. Therefore, the use of data mining becomes more easily deployed within various businesses in terms of both its application and its results. Data mining avoids the constraints of sample-based approaches. A framework of data mining, enabling software applications to be developed with simplified interfaces, increases the usability of these techniques.

Why is data mining necessary?

When one deals with a large body of data with complicated or professional information such as scientific, medical, or stock market measurements, it is typically very difficult for his interesting parts or fields to describe in a SQL query or a computer programming language such as C, C++, Visual Basic, etc. A more natural means of interacting with the database is to state the query by examples. In this case, the analyst would label a training set of cases of one class versus another and let the data mining system build a model for distinguishing one class from another. The system can then apply the extracted classifier to search the full database for events of interest.

Another major problem in databases is that it is principally difficult for human analysis to visualize and understand a large data set. Data can grow along two dimensions: the number of fields (dimensions or attributes) and the number of cases (the sample size or instances). Human analysis and visualization abilities do not scale to high-dimensions and massive volumes of data. A standard approach to dealing with high-dimensional data is to project it down to a low-dimensional space (by eliminating duplicated or redundant attributes or neglecting less important one) and attempt to build models in this simplified subspace.

As other purposes of data mining, one may be interested in the relationship or correlation between data fields of a record from a large size of databases. If one want find the relationship between one discrete data field (or decision field) and other nominal or numerical data fields, this problem becomes a classification problem. If the decision filed is numeric, the problem is called 'regression'. When one looks for the correlation between attributes (not the decision data field), the problem is also called 'association'. These all interests could not be accomplished by any conventional utilities (such as SQL and search engine), so that they become the reasons why data mining is necessary.

The scope of data mining

There are a lot of applications in data mining fields. Classification and association are most common problems in data mining for knowledge extraction and machine learning. Regression and classification are also important tools for estimation and prediction. Because human has very limited viewpoint of intuitive and visual understandability on problems with large dimension or huge size of databases, the visualization of data mining is recently emphasized in practices. Some special purposes of data mining are currently processed such as text mining or web mining (Zaiane and Han, 1998), for a new search technique in World Wide Web multimedia or texture mining for image processing, and spatial mining for the time-series analysis (Kim et al., 2000). Especially the text mining (Wallis and Nelson, 2001) is one of good approaches for natural language processing.

Figure 1 shows the scope of data mining fields. Many techniques or solutions for data mining and knowledge discovery in databases are very widely provided for classification, association, clustering and regression, search, optimization, etc. In detail top-down induction of decision trees (TDIDT; e.g., ID3 or C4.5: Quinlan, 1986, 1993), CART (CART: Breiman et al., 1984), fuzzy logic and artificial neural networks, or some statistical method are applicable for a classification problem.

For association, k-nearest neighbors and radial-based neural networks are well-known examples. Recently CMAR (Li et al., 2001) has been provided for a new association rules. For clustering, it is available to use self-organization map (SOM; Haese and Goodhill, 2001),

vector quantization (VQ; Gray, 1984), simulated annealing (SA; Brown and Huntley, 1991), genetic algorithm (GA), etc. For regression principal component analysis (PCA; Baudat and Anouar, 2000), or support vector machines for regression (Vapnik, 1995, 1998) can be used.

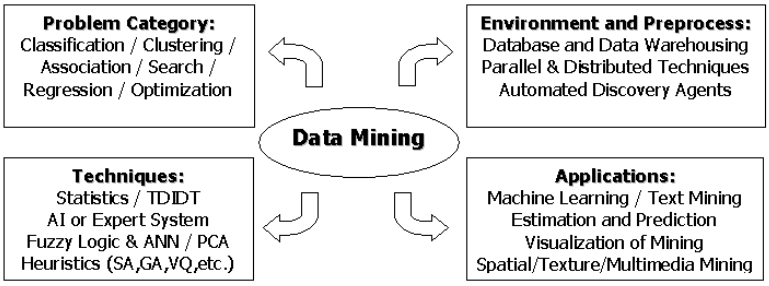


Figure 1. The scope of data mining

2. Introduction to classification problems in data mining

Classification is an essential data mining technique whereby database records, acting as *training samples*, are analyzed in order to produce a model of the given data (Fayyad et al., 1996a, 1996b; Piatetsky-Shapiro and Frawley, 1991). Each record is assumed to belong to a predefined class, as determined by one of the attributes, so-called *class attribute*. In spite of tremendous amount of research on classification, it is possible to categorize six methodologies of solutions for classification as

1. Mathematical Programming,
2. Statistical Approach using Hill Climbing Methods,
3. Linear Discriminant Trees,
4. Piecewise Linear Discriminant Trees,
5. Artificial Neural Networks, and
6. Variants of Top-Down Induction of Decision Trees (TDIDT)

Mathematical programming

Linear programming has been used for building adaptive classifiers since late 1960s (Ibaraki and Muroga, 1968). Given two possibly intersecting sets of points, Duda and Hart

(1973) proposed a linear programming formulation for finding the split whose distance from the misclassified points is minimized. Brown and Pittard (1993) also employed linear programming for finding optimal multivariate splits at classification tree nodes. Lin and Vitter (1992) built a model of zero-one integer programming for designing vector quantizers in order to utilize the classification. Bennett and Mangasarian (1992, 1994) introduced the use of both linear and quadratic-programming techniques to build machine-learning systems in general and decision trees in particular case of numerical attributes. More recently Bennett and Blue (1996, 1997) used support vector machines to optimize decision trees. Almost all the above papers attempt to minimize the distance of the misclassified points from the decision boundary.

Statistical approach using hill climbing methods

CART (Breiman et al., 1984) is one of is most well-known classification algorithm by the use of linear combinations of attributes. This algorithm uses heuristic *hill-climbing* and backward feature elimination to find good linear combinations at each node. Murthy et al. (1993, 1994a, 1994b) described significant extensions to CART using randomized techniques.

Linear discriminant trees

Several authors have considered the problem of constructing decision-tree-structured classifiers that have linear discriminants at each node (Duda and Hart, 1973; Fayyad et al., 1996b). Qing-Yun and Fu (1983) also described a method to build linear discriminant trees by using multivariate stepwise regression to optimize the structure of the decision tree as well as to choose subsets of features to be used in the linear discriminants. Todeshini and Marengo (1992) described a method for building linear discriminant classification trees, in which the user can decide at each node what classes need to be split.

Piecewise linear discriminant trees

Sklansky and Wassel (1981) suggested a procedure of training a linear split in order to minimize the probability of errors. Using this procedure, they developed a system to induce a piecewise linear classifier (Sklansky and Wassel, 1980). The final classifier

produced by their method was a piecewise linear decision surface, not a tree. Foroutan (1985) discovered that the retrial substitution error rate of optimized piecewise linear classifiers was nearly monotonic with respect to the number of features. Based on this result, Foroutan and Sklansky (1987) suggested an effective feature selection procedure for linear splits that uses zero-one integer programming.

Artificial neural networks

In the neural networks community, many researchers have recently considered hybrid structures between decision trees and artificial neural networks (Golea and Marchand, 1990; DAlché-Buc et al., 1994). Although these techniques were developed as neural networks of which structure could be automatically determined, their outcome can be interpreted as decision trees with nonlinear splits (Cios and Liu, 1992; Sirat and Nadal, 1990). It can be found in neural tree construction (i.e., a tree structure of nodal neural network systems) for the techniques which are very similar to those used in decision tree construction, such as information theory for splitting a decision tree. There exist other hybrid techniques between decision trees and neural networks (DAlché-Buc et al., 1994). Sethi (1990) described a method for converting a univariate decision tree into neural networks and then retraining them, resulting in tree structured entropy networks with sigmoid splits. Guo and Gelfand (1992) developed a construction method of decision trees with small multi-layered networks at each node by implementing nonlinear and multivariate splits (by using the small neural networks systems). Lee et al. (2000) presented the combination of a decision tree and artificial neural networks for the classification of magnetic flux leakage signals in the area of nondestructive evaluation applications.

Variants of top-down induction of decision trees (TDIDT)

Schuermann and Doster (1984) used polynomial splits at decision nodes for the construction of a decision tree. Heath et al. (1993a, 1993b) used simulated annealing to find the best oblique split at each tree node. Lubinsky (1994) attempted bivariate trees, trees in which some functions of two variables can be used as tests at internal nodes. Lubinsky considered the use of linear cuts, corner cuts and rectangular cuts, using ordered and unordered variables.

Literature reviews on top-down induction of decision trees

In recent years a number of classification techniques from both statistics and machine learning communities have been proposed (Fayyad *et al.,* 1996b; Quinlan, 1993; Weiss and Kulikowski, 1991). A well-known method of classification is the induction of decision trees (e.g., CART: Breiman et al*.,* 1984; C4.5: Quinlan, 1993). A decision tree is a top-down tree-structure consisting of internal nodes, leaf nodes, and branches. Each internal node represents a decision on a data attribute or a function of data attributes, and each outgoing branch corresponds to a possible outcome of the instance. Each leaf node represents a class. In order to classify an unlabeled data sample (a record in the database), the classifier tests the attribute values of the sample against the decision tree. A path is traced from the root to a leaf node, which holds the class predication for that sample. Decision trees can easily be converted into IF-THEN rules (Quinlan, 1993) and used for decision-making. The efficiency of existing decision tree algorithms (e.g., ID3: Quinlan, 1986; CART: Breiman et al., 1984), has been well established for relatively small data sets (Mooney et al*.,* 1989; Weiss and Kapouleas, 1989).

There are several discussions on top-down induction of decision trees (TDIDT) how to make it accurate, reliable, efficient, and valuable. Many studies on TDIDT have been performed to construct advanced structures of TDIDT in order to improve more accurate. First, it has been studied for the structure of decision node to be either *univariate* (a single attribute at each internal node) or *multivariate* (multiple attributes at each internal node). Second, there are several approaches of multiple decision trees to improve their adaptability and reliability. They built several decision trees by different sampling of training data. Then, they tried to find the best prediction of classification by voting the results of multiple trees. Third, self-rebuilding TDIDT by acquiring a new knowledge (classification sample) has been considered. In practice collected training data does not cover the real distribution of popularity of a classification problem. The classification of data mining is an empirical method to classify a phenomenon. Therefore, as much as growing the collection of training data samples, it is necessarily required to update a decision tree incrementally and efficiently. Forth, the scalability of classification instances has been widely considered for the efficiency

on a large database. There are a lot of interesting topics of classification in data mining area, such as building a decision tree incorporating relational databases, costs, and meta-knowledge (logical or functional combination of data attributes).

Univariate vs. multivariate

Decision trees most commonly are univariate. Multivariate decision trees can use splits that contain more than one attribute at each internal node. Although several methods have been developed in the literature for constructing multivariate trees, this body of work is not as well known as that on univariate trees. Several methods for the construction of decision trees with multivariate decision-makings have been presented (Murphy and Pazzani, 1991; Ali and Pazzani, 1995b). Murphy and Pazzani (1991) show the conceptual approach of the constructive induction of multivariate decision trees. They showed multivariate constructive induction systems have better performance than *univariate* systems. However, there are some problems: *NP-complete* for generating the combinations of multivariate decision-making and *pruning* (to be discussed later) methodology.

Most research of multivariate splits considered linear trees (Brodley and Utgoff, 1992, 1995; Murthy et al., 1994a, 1994b). These are trees which have tests based on a linear combination of the attributes at some internal nodes. The problem of finding an optimal linear split (optimal with respect to any of the feature evaluation measures) is more difficult than that of finding the optimal univariate split. Multivariate decision trees are often smaller and more accurate than univariate trees. However, the use of linear combinations of multiple attributes may be too hard to interpret the resulting trees. Bioch et al. (1997) showed that bivariate decision trees could take advantages of both univariate and multivariate trees. However, it was limited to apply the classification problem with numerical attributes only.

Multiple decision trees

A known weakness of decision tree construction is the variance of tree construction, especially when the samples are small and the features are many (Dietterich and Kong, 1995). Variance can be caused by random choice of training and pruning (to be discussed later) samples, by many equally good attributes only one of which can be chosen at a node,

due to cross validation or because of other reasons. There are several discussions on using a collection of decision trees, instead of just one, to reduce the variance in classification performance (Kwok and Carter, 1990; Buntine, 1992). The idea is to build a set of (correlated or uncorrelated) trees for the same training sample, and then combine their results. Multiple trees have been built using randomness (Heath et al., 1993a) or using different subsets of attributes for each tree (Shlien, 1990, 1992). Classification of decision trees have been combined using either simplistic voting methods (Heath et al., 1993a) or using statistical methods for combining evidence (Shlien, 1990). Murphy and Pazzani (1994) developed a decision forest consisting of all decision trees with the training data from a series of experiments. They presented the relationship between the size of a decision tree consistent with some training data and accuracy of the tree on test data. They show that smaller decision trees are more recommendable for simpler problem domain. However, for ore complex problems, slightly larger decision tree could be more recommendable in the viewpoint of prediction accuracy even thought its reliability is slightly less than the smallest decision tree.

Incremental decision trees

Fisher and Schlimmer (ID4, 1988) developed an incremental induction of decision trees. It was implemented for the reason that a new training example, which is incorrectly or improperly classified by the prebuilt decision tree, makes the decision tree reconstructed. So, ID4 algorithm is able to build decision trees incrementally: a decision tree can be updated when new instances become available. A non-incremental algorithm such as ID3 and C4.5 requires storing all historical data if the decision tree is necessarily updated by a new misclassifying instance. Utgoff (ID5 or IDL, 1989) suggests an advanced incremental algorithm that maintains statistics on the distributions of instances over attributes at each node in the tree in order to update the tree if necessary. When a new example is entered, then the effect of the training example on this distribution is computed, and the method checks if the tree must be revised by replacing the current node or by a different attributes.

Scalability of decision trees

One of the chief obstacles to effective data mining is the clumsiness of managing and analyzing data in very large files. In data mining applications, very large training sets of

millions of examples are common. Efficiency and scalability become issues of concern when these algorithms are applied to the mining of very large, real-world databases. Most decision tree algorithms have the restriction that the training instances should reside in computer main memory. Therefore, this restriction limits the scalability of such algorithms, where the decision tree construction can become inefficient due to swapping of the training samples in and out of computer cache memories. DuMouchel et al. (1999) introduced an algorithm named '*data squashing' that* substitutes a smaller dataset for the large one.

The induction of decision trees from very large training sets has been addressed by SLIQ (Mehta et al., 1996) and SPRINT (Shafer et al., 1996) decision tree algorithms. These propose presorting techniques on disk-resident data sets that are too large to fit in memory. While the scalability of SLIQ, however, is limited by the use of a memory-resident data structure, SPRINT removes all memory restrictions and hence can handle data sets that are too large for SLIQ. Unlike SLIQ and SPRINT, which operate on both raw- and low-level data, DBMiner (Kamber et al., 1997) has been proposed for efficiency and scalability issues by operating higher level of data descriptions.

Other interests in decision trees

Combining primitive decision rules:

Learning from interpretations has been growing interest in recent years. Primitive decision rules are easily generated by C4.5 (Quinlan, 1993). Some of decision rules are somewhat useless for applying in real practice if the occurrence possibility of those rules is very low. TILDE (Blockeel and De Raedt, 1998) is recent upgrade of Quinlan's C4.5 (1993). It employs logical queries, *first-order* upgrades of existing attribute-value descriptions, rather than just using attribute-value tests in nodes of a decision tree. It slightly complicates the classification process with *first-order* decision trees, because there are tremendous possible combinations of combining useless decision rules to make more usable.

Incorporating of meta-knowledge

Constructive induction algorithms create new complex attributes by combining existing attributes in ways that make the description of the concept easier. The fulfringe

constructive induction algorithm (Oliveria and Vincentelli, 1993) belongs to a family of *constructive induction algorithms* that identify patterns near the fringes of the decision tree and uses them to build new attributes.

Parallel or multi-relational queries in decision tree induction:

Extensibility, complexity of data types, and high performance of queries is one of the most important requirements of information systems supporting complex application domains such as geosciences, medicine, finance, and multimedia. Unfortunately, support for both the extensibility and the optimality of non-relational data in parallel query is recognized as open research problems because research in parallel query optimization has historically concentrated on relational join queries, such as development of efficient parallel joins and sorting algorithms and optimization of parallel join trees to maximize query performance. Shek et al. (1996) showed how database query processing and distributed object management techniques could be used to facilitate geoscientific data mining and analysis. They developed an extensible parallel geoscientific query processing system called '*Conquest'*, which concentrates on the features that make it especially suitable for geoscientific data modeling, parallel query processing, and heterogeneous distributed data access.

Traditional decision tree approaches have one major drawback: they cannot be employed to analyze relational databases containing multiple tables. Such databases can be used to describe objects with some internal structure, which may differ from one object to another. It implies any conventional or prepositional 'attribute-value' decision trees is not suitable for describing groups of such objects in terms of occurrence of a certain substructure. TILDE (Blockeel et al., 1998) has been applied to overcome this limitation of prepositional decision trees by using first order logic to represent decisions in the tree (so-called *first order logical decision tree*). However, TILDE assumes that objects are represented in first order logic rather than as collections of recodes in a relational database, and thus does not benefit from the opportunities for optimization that are provided by a relational representation. Knobbe et al. (1996) proposed an alternative approach that provides the means to induced decision trees multi-relational decision trees from structural information. In a multi-relational environment, producing such a complement is less trivial.

In relational databases an association between two tables describes the relationship between records in both tables. The nature of this relationship is characterized by the *multiplicity* of the association. The *multiplicity* of an association determines whether several records in one table relate to single or multiple records in the second table. They provided a generic algorithm for top-down induction of multi-relational decision trees within the multi-relational data-mining framework.

Open problems in TDIDT

Pruning

Pruning, the method most widely used for obtaining right sized trees, was proposed by Breiman et al. (1984). They suggested the following procedure: build the complete tree (a tree in which splitting no leaf node further will improve the accuracy on the training data) and then remove subtrees that are not contributing significantly towards generalization accuracy, based on a pruning set. It is argued that this method is better than stop-splitting rules, because it can compensate, to some extent, for the suboptimality of greedy tree induction. For instance, if there is very good node, 'a few levels below a not-so-good node', a stop-splitting rule will terminate tree growth at whereas pruning may give a high rating for, and retain the whole subtree at. Kim and Koehler (1994) analytically investigate the conditions under which pruning is beneficial for accuracy. Their result states pruning is more beneficial with increasing skewness in class distribution and/or increasing sample size.

Pruning set is a portion of the training data that is set aside exclusively for pruning alone. Use of a separate pruning set is a fairly common practice. Another method rather than cost complexity pruning is the reduced error pruning (Quinlan, 1987). This method, unlike cost complexity pruning (Breiman et al., 1984), does not build a sequence of trees and hence is claimed to be faster. The requirement for an independent pruning set may be difficult, especially when small training samples are involved.

Breiman et al. (1984) describe a cross validation procedure that avoids reserving part of training data for pruning, but has a large computational complexity. Crawford (1989) analyzed their cross validation procedure, and pointed out that it has a large variance,

especially for small training samples. He suggested a '0.632 *bootstrap method*' as an effective alternative. Gelfand et al. (1991) claimed that the cross validation method was both inefficient and possibly ineffective in finding the optimally pruned tree. They suggested an efficient iterative tree growing and pruning algorithm that is guaranteed to converge. Quinlan (1987, 1993) developed a pessimistic pruning unnecessarily required for a separate pruning set by using a statistical correlation test. Quinlan and Rivest (1989) used the minimum description length (MDL; Rissanen, 1989) for tree construction as well as for pruning.

Feature representation and extraction

Many techniques for data analysis can be regarded as seeking for a description of data in terms of elementary features. An advantage of a feature representation is that it reduces redundancy in the input patterns (Barlow, 1989). Furthermore, a description in terms of features can provide a lucid explanation of objects (input patterns), which can in addition be helpful in understanding the hidden data generating process.

Linear techniques for feature extraction are most widely applied in practice: e.g., principal component and factor analysis. Both these techniques give a meaningful representation of the data only if the data are Gaussian distributed around some low dimensional linear subspace. Some studies for non-Gaussian linear methods have been introduced such as independent component analysis (Bell and Sejnowski, 1995) and the sparse coding approach (Olshausen and Field, 1996). The significant advantages of linear methods are their speed and easy to understand (or, interpret) feature representation. However, the most important disadvantage of linear models is that they cannot describe multi-modal distributions. The most well-known and simplest method for finding multi-modal structure in the data is vector quantization (VQ) (Gray, 1984). The weakness of vector quantization, however, is its lack of a feature representation. To overcome this problem, more advanced nonlinear probability models have recently been promoted by several authors in the context of feature extraction (Sallans et al., 1998; Attias, 1999). In contrast to standard vector quantization, where a data point is explained in terms of a single code-vector, these models explain a data point in terms of a combination of elementary features.

Feature evaluation rules

Most techniques for attribute selection in decision trees are biased towards attributes with many values, and several *ad hoc* solutions to this problem have appeared in the machine learning literature. Statistical tests for the existence of an association with a prespecified significance level provide a well-found basis for addressing the problem. However, many statistical tests are computed from a chi-squared distribution, which is only a valid approximation to the actual distribution in the large sample case and this patently does not hold near the leaves of a decision tree. Frank and Witten (1998a, 1998b) suggested using a permutation test for attribute selection. They chose one such test for further exploration, and gave a novel two-stage method for applying it to select attributes in a decision tree.

Used for classification, decision trees are essentially probability estimators. Feature evaluation rules are heuristics whose aim is to produce as reliable probability estimates from training data as possible. A taxonomy, proposed by Ben-Bassat (1987), is helpful in easy to understand the large number of existing feature evaluation criteria. He divided feature evaluation rules into three categories:

- *Rules derived from information theory*: As an extension of Shannon's entropy (see Appendix B) there are several approaches for decision tree construction by maximizing global mutual information, i.e., by expanding tree nodes that contribute to the largest gain in average mutual information of the whole tree (Sethi and Savarayudu, 1982). Tree construction by locally optimizing information gain, which represents the reduction in entropy due to splitting each individual node, is explored in pattern recognition (Hanisch, 1990), in sequential fault diagnosis (Varshney et al., 1982), and in machine learning (Quinlan, 1986). Mingers (1987) suggested the G-statistic, an information theoretic measure that is a close approximation to distribution for tree construction as well as for a stopping rule. De Merckt (1993a, 1993b) suggested an attribute selection measure that combined geometric distance with information gain, and suggested that such measures are more appropriate for numeric attribute spaces.

- *Rules derived from distance measures*: "Distance" here refers to the distance between class probability distributions. Feature evaluation criteria are here used for measuring separability, divergence or discrimination between classes. A popular distance measure is the *Gini index* of diversity: it has been used for tree construction in statistics (Breiman et al., 1984), pattern recognition (Gelfand et al., 1991), sequential fault diagnosis (Pattipati and Alexandridis, 1990), etc. Breiman et al. (1984) pointed out the Gini index has difficulty when there are a relatively large number of classes (Murthy et al., 1994b). Taylor and Silverman (1993) pointed out the Gini index emphasizes equal sized offspring and purity of both children and, therefore, suggested a splitting criterion (or mean posterior improvement) that emphasizes exclusivity between offspring class subsets instead. Class separation-based metrics are also distance measures in machine learning (Fayyad and Irani, 1990).

- *Rules derived from dependence measures*: These measure the statistical dependence between two random variables. All dependence-based measures can be interpreted as belonging to one of the above two categories (Ben-Bassat, 1987).

There exist several attribute selection criteria that do not clearly belong to any category in Ben-Basset's taxonomy. For example, Talmon (1986) used a combination of mutual information and measures. First he measured the gain in average mutual information due to a new split, and, then, quantified the probability of which gain may be obtained by the split. The split that minimized mutual information was chosen by these methods. The main advantage of this statistic is that, unlike most of the other measures, its distribution is independent of the number of training instances.

Heath et al. (1993a, 1993b) used the simplest possible attribute selection criteria, based on the number of misclassified objects, for oblique decision tree induction. The measures were called max minority and sum minority, respectively denoting the maximum and the sum of the number of misclassified points on either side of a binary split. Max minority has the theoretical advantage that the depth of the tree constructed using this measure is at worst logarithmic in the number of examples. Lubinsky (1993, 1994) also used

the number of misclassified points as a splitting criterion, calling it inaccuracy. The performance of these measures does not seem to be in general as good as the information theory or distance-based measures, and additional tricks are needed to make these measures robust (Lubinsky, 1993; Murthy et al., 1994a).

Estimating probabilities

Decision trees have crisp decisions at leaf nodes. On the contrary, class probability trees assign a probability distribution for all classes at the terminal nodes. Breiman et al. 1984) proposed a method for building class probability trees. Buntine (1992) described Bayesian methods for building, smoothing and averaging class probability trees. Smyth et al. (1995) suggested an approach to refine the class probability estimates in a greedily induced decision tree using local kernel density estimates. Guur-Ali and Wallace (1993) described the assignment of probabilistic goodness to splits in a decision tree. Mogre et al. (1994) recommended a unified methodology for combining uncertainties associated with attributes into that of a given test, which can then be systematically propagated down the decision tree.

Incorporating costs

In most real-world domains, attributes can have costs of measurement, and objects can have misclassification costs. If the measurement of misclassification costs is not identical between different classes, decision tree algorithms need to be designed explicitly to prefer cheaper trees. Several attempts have been made to make tree construction cost-sensitive. These involve incorporating attribute measurement costs in machine learning (Tan, 1993), in pattern recognition (Morris and Kalles, 1994), and in statistics with incorporating misclassification costs. Methods to incorporate attribute measurement costs typically include a cost term by using prior probabilities or cost matrices into the feature evaluation criterion.

Tree quality measures

The fact that several trees can correctly represent the same data raises the question of how to decide that one tree is better than another. Several measures have been suggested to quantify tree quality. Fayyad and Irani (1990) argued that one could achieve performance improvement along other measures by concentrating on optimizing the number of leaf nodes.

Generalization accuracy is a common measure for quantifying the goodness of learning systems. The accuracy of the tree is computed using a testing set that is independent of the training set or using estimation techniques like cross-validation or bootstrap. Kononenko and Bratko (1991) pointed out comparisons on the basis of classification accuracy were unreliable, because different classifiers produced different types of estimates (e.g., some concluded 'yes-or-no' classifications, but some provided class probabilities) and accuracy values can vary with prior probabilities of the classes. They suggested an information-based matrix (so called as an *confusion matrix*) to evaluate classifiers from different methods.

Comparisons of multiple decision tree or rules

Given the large number of feature evaluation rules, a natural concern is to decide their relative effectiveness in constructing good trees. Evaluations in this direction, in statistics, pattern recognition and machine learning, have been predominantly empirical in nature, though there have been a few theoretical evaluations. A lot of studies have concluded that there are not much different performances between different measures for decision tree evaluation. Any strategy that results in superior generalization accuracy on some problems is bound to have inferior performance on some other problems.

Breiman et al. (1984) conjectured that decision tree design is rather insensitive to any one from a large class of splitting rules, and it is the stopping rule that is crucial. Mingers (1987) compared several attribute selection criteria, and concluded that tree quality does not seem to depend on the specific criterion used. He found random attribute selection criteria are even as good as measures like information gain (Quinlan, 1986). Several researchers pointed out that information gain is biased towards attributes with a large number of possible values. Quinlan (1993) suggested a gain ratio as a remedy for the bias of information gain. Kononenko (1995) pointed out that the "*Minimum Description Length*" based feature evaluation criteria have the least bias towards multi-valued attributes.

Comparisons of TDIDT with other exploration methods

There exist several alternatives to decision trees for data exploration, such as neural networks, nearest neighbor methods and regression analysis. Quinlan empirically compared decision trees to genetic classifiers (Quinlan, 1988) and to neural networks (Quinlan, 1993). Atlas et al. (1990) compared between multi-layered perceptrons and CART, and found that there is not much difference in accuracy. Talmon et al. (1994) compared classification trees and neural networks for analyzing electrocardiograms (ECG) and concluded that no technique is superior to the other. Brown et al. (1993) compared backpropagation neural networks with decision trees on three problems that are known to be multimodal. They showed there was not much difference between both methods. However, they mentioned that the computational efficiency of decision trees was better than neural network when multivariate splits were used, but the neural networks did better with feature selection. Feng et al. (1993) presented the comparison of several machine-learning methods including decision trees, neural networks and statistical classifiers. They concluded that no method seems uniformly superior to others, but machine-learning methods seemed to be superior for multi-modal distributions, and statistical methods are computationally the most efficient.

3. Model selection criteria for classification

For classification in data mining it is assumed that we have classified instances (sampling data) from which a model can be induced. We suggest looking for a good classification model in the sense of empirical risk minimum (ERM), which is defined as the statistical estimate of misclassification rates from training samples. The ERM inductive principle is typically used in a parametric setting where the model is specified first and then its parameters are estimated from the data. This approach works well only when the number of training samples is relatively large enough to the prespecified model complexity (or the degree of freedom).

Another important issue for model selection is the complexity of models. According to Occam's razor principle, there exists a trade-off relationship between model complexity and the accuracy of a model to fit the training data. Therefore, the selection of best- or high-

qualified model must be chosen by the consideration of both the model complexity and the model accuracy. Model complexity is usually controlled by a priori knowledge, however, by the Occam's razor principle, such a priori knowledge cannot assume a model of affixed complexity. In order words, even if the true parametric form of a model is known a priori, it should not be automatically used for predictive learning with the samples. The penalty of model complexity can be defined as the estimated prediction error (and may be its confidential limit if it is available), the size of models (i.e., the tree size or the "*Minimum Description Length (MDL)*" of a model; Kononenko, 1995), computational efforts, and so on.

In general, a model, which perfectly fits data, might be less desirable than others that partially fit the data. This is known as *an overfitting problem*. The problem of trading off the simplicity of a model with how well it fits the training data is a well-studied problem. In statistics this is known as the *bias-variance tradeoff* (Friedman, 1997). It is also known as *penalized likelihood* in Bayesian inference (Heckerman, 1997). In pattern recognition and machine learning, it is measured by the *minimum message length* (MML; Wallace and Patrick, 1993) or *minimum description length* (MDL; Rissanen, 1978) that determines the best model for a given data set by the minimum coding length of data and model combined. If a model fits the data exactly, the data need not be encoded and the cost is that of coding the model. Therefore, both the variance of model reliability and the bias of prediction accuracy should be considered as penalty factors of model selection criteria.

For describing a model selection principle, it is assumed that a flexible class (without limiting the number free parameters) of approximating functions $f(x,\omega)$, where x is the training sample data, ω is free parameters, which is in a set of abstract parameters, Ω. Let $R_{emp}(\omega)$ denote the usual empirical risk, which is usually describe by the training error, and $\phi[f(x,\omega)]$ represent the penalty of a model $f(x,\omega)$, which is a nonnegative function associated with each possible estimate $f(x,\omega)$. Then the penalized (or regularized) penalty $R_{pen}(\omega)$, as an objective function of the model selection, is defined as

$$R_{pen}(\omega) = R_{emp}(\omega) + \lambda \phi[f(x,\omega)], \tag{1.2}$$

where λ is a nonnative parameter to control the strength of the penalty relative to the term $R_{emp}(\omega)$. If λ is very large, then the result of minimizing $R_{pen}(\omega)$ does not depend on the training data. On the other hand, if λ is very small, the result of minimizing $R_{pen}(\omega)$ does not consider any prediction errors from further data, which do not belong to the training data, as well as the lack of easy to understand the final model.

In this thesis we select several empirical model approximating functions $f(x,\omega)$, such as ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), and PART (Frank and Witten, 1998a). Also, we defined $R_{emp}(\omega)$ as a relative score of training errors from the results of different approximating functions, and $\phi[f(x,\omega)]$ as a weighted sum of relative penalty scores, which consists of

I. the prediction errors of $f(y,\omega)$, where y is not overlapped with the training data,
II. the size of decision tree and the MDL of decision-making descriptions, and
III. the confidential limit for the prediction of $f(y,\omega)$ from a cross-validation.

Also, one can specify the above control parameter λ as the weight vector for each penalty category. Then, the objective function $R_{pen}(\omega)$ can be described by (relative) total score of each classification model $f(x,\omega)$. Each classification problem has a different objective, so that it has different weights. The cross-validation analysis was used for providing the sample mean and variance of misclassification errors for all resampling results.

4. Research objectives

For the successful data mining, a simple and reliable transparent system is very essential for many applications. The reliability of data mining systems is highly related to overcoming *overfitting* problems. The *overfitting* problem causes from unnecessary decision-makings built by a training set. That is, it occurs when some of decision-makings are too tightly described in the particular training examples to predict some of test data or uncollected data correctly. Some decision-makings may only fit for very few of the current training dataset. This situation is called as '*overfitting*'. One of the ways for resolving overfitting problems is *pruning* decision trees. The *pruning* of an internal node replaces the

subtree of the node by a leaf if the rest of the node is overfitting to the current training dataset. The more pruned machine learning system has the less difference of accuracy between training and testing results. Figurer 2 shows the overfitting problem while a model is being built from training data.

The policy of defining decision boundaries is the most essential to achieve both training accuracy and prediction accuracy. The shapes of decision boundaries are also highly related to the model complexity. Consider a decision tree with numerical attributes only. For C4.5 (Quinlan, 1993) the decision boundaries from univariate attributes are formed as a set of orthogonal partitions as shown in figure 3(a). Because of too much simplicity of this C4.5 decision boundary description, it has always a risk of overfitting problems. For an oblique decision tree (Murthy et al., 1994a), the decision boundaries form still a linear borders as shown in figure 3(b), but these borders are made by multiple decision variables, so that, with much smaller size of a decision tree than C4.5, the oblique decision tree can reduce the possibility of overfitting problems. However, it has still weakness when the actual decision boundary is nonlinear. In this case, support vector machines (SVM; Bennett, 1994-1997) are more suitable by building piecewise-linear or nonlinear decision boundaries as shown in figure 3(c). It is obviously trade-off relationship between the model complexity and the model accuracy (of not only training but also prediction). SVM has less possibility of overfitting problems than others, but the model description is more complex than others.

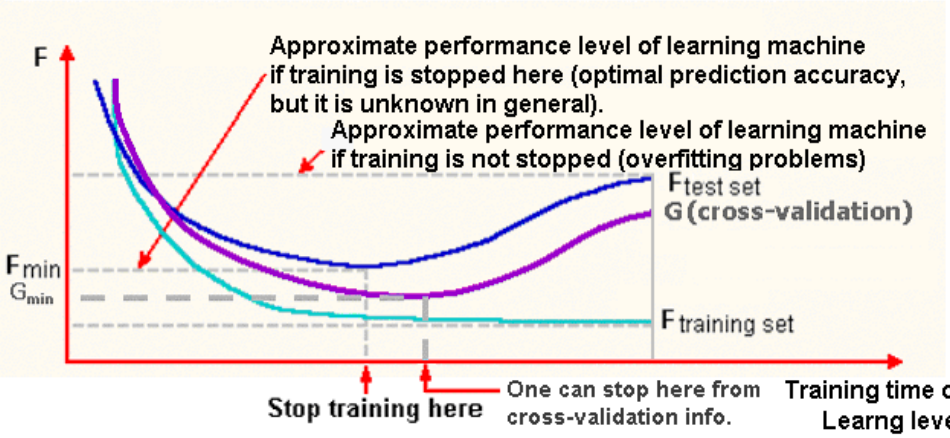


Figure 2. The overfitting problem

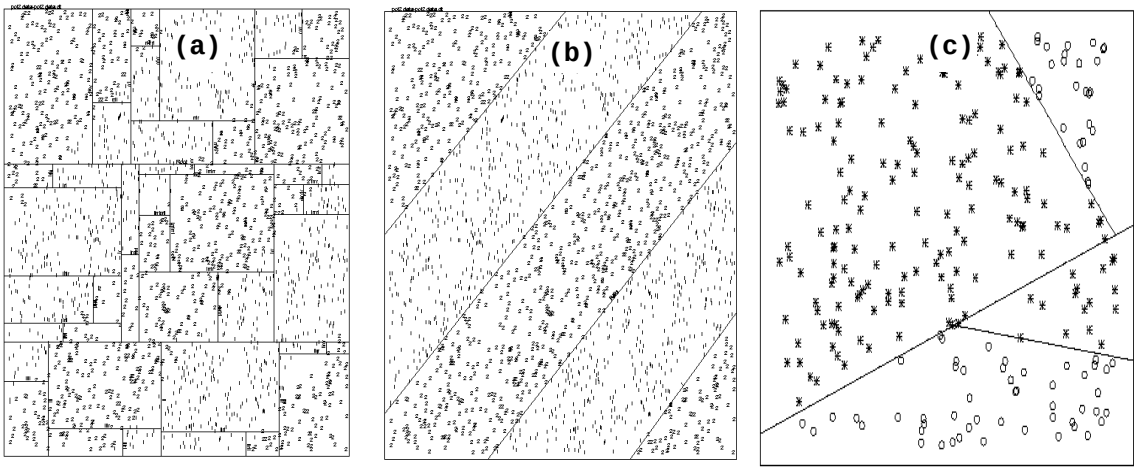


Figure 3. Making decision boundaries: (a) C4.5, (b) oblique decision tree, and (c) support vector machines (SVM)

To build a decision tree with nominal attributes in recent years, there is very little research for the consideration of nonlinear or multivariate decision boundary description. With this limited capability, it has always an overfitting risk.

In this thesis, new approaches of decision-tree construction have been developed to reduce the overfitting problems while the model complexity is not seriously increased. For only nominal attribute problems, a second-order decision-tree induction (SODI) has been developed. For only numerical attribute cases, a new algorithm of support vector machines for multi-category classification (SVMM) has been developed. For a general case of both nominal and numerical attributes, IDSS (Induction of Decision trees with SODI and SVMM) has been developed. In those applications the policy of the model selection is to minimize the penalized risk function in (1.2), which is the weighted sum of all penalty costs.

5. Thesis organization

Chapter 2 describes a new method of top-down induction of decision trees (TDIDT) for nominal attributes only, so called as 'Second-Order Decision-Tree Induction (SODI)'. '*Second-order*' decision-makings or a '*bivariate*' decision tree concept is employed to construct a decision tree. In this chapter, a method of how to eliminate some redundant nominal attributes is provided before applying the SODI algorithm. Both mathematical

proofs and empirical tests show the results of SODI tend to dominate other *univariate* decision trees, such as ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), and PART (Frank and Witten, 1998a, 1998b), for several real world problems in general. Also, we provided illustrative example for pruning of SODI, and experimental comparison between SODI and other methods.

Chapter 3 deals with classification problems consisting of numerical attributes only. We developed a new algorithm for top-down induction of decision trees using support vector machines (SVM). The first and second sections in this chapter introduced some literature reviews and brief summaries of support vector machines. In the third section, we proposed a new model of support vector machines for multi-category problems, a new method, so-called support vector machines for multi-category classification (SVMM) that combines TDIDT and SVM in order to take advantages from both methods. Also, we included two illustrative examples for easy to understand our algorithm. At the fourth section, we compared our SVMM with several conventional methods.

Chapter 4 describes how to combine the results of both chapter 2 and chapter3. A new system of TDIDT using the combination of both SODI and SVMM is called as IDSS (Induction of Decision trees with SODI and SVMM). Empirical tests show that this IDSS built more accurate decision trees that other decision trees for a Germany credit approval problem from data warehouse (see appendix C). Also, the performance of IDSS was evaluated with hierarchical artificial neural networks for the classification of magnetic flux leakage signals as an application of nondestructive test (Lee et al., 2000).

Finally at chapter 5, all works for new classification models described in the above chapters are summarized. Each of techniques performed reasonably with respect to both the prediction accuracy and reliability for several real world problems. A mathematical proof for finding redundant nominal attributes is also one of the contributions in this thesis. Appendix A shows some mathematical proofs of propositions or theorems described in this thesis. Appendix B provides the short introduction of the ‘measures of uncertainty’, well known in information theory. Finally, Appendix C provides the description of classification problems that we used in this thesis obtained from a data warehouse (Witten and Frank, 1999).

CHAPTER 2. SODI: SECOND-ORDER DECISION-TREE INDUCTION

Top-down induction of decision trees (TDIDT) has been significantly studied by a number of researchers. Breiman et al. (1984) proposed a regression tree for classification problems (CART; Classification and Regression Tress). ID3 and C4.5 are basic TDIDT algorithms introduced by Quinlan (1986, 1993) for inducing classification models. His research group recently developed C5.0 (or, See5.0, 1998) that has been improved for the efficient use of both memory and construction speed, manipulation of numerical attributes and missing values, more effective pruning process, and the interface with substantial databases. The majority of the algorithms that construct decision trees from examples use splitting heuristics that aim to minimize the size of the induced decision trees. Fisher and Schlimmer (ID4, 1988) developed an incremental induction of decision trees. It was implemented for the reason that a new training example, which was incorrectly classified by the previous decision tree, makes the tree reconstructed by storing all examples. This ID4 algorithm builds decision trees incrementally. A non-incremental algorithm such as ID3 and C4.5 requires storing all historical data if the decision tree is necessarily updated by a new misclassifying instance. Utgoff (ID5 or IDL, 1989) suggested an advanced incremental algorithm that maintains statistics on the distributions of instances over attributes at each node in the tree in order to update the tree if necessary. When a new example is entered, then the effect of the training example on this distribution is computed, and the method checks if the tree must be revised by replacing the current node or by a different attributes.

A lot of research has been studied to reveal the characteristics of the induction of decision trees. Murphy and Pazzani (1994) developed a decision forest consisting of all decision trees with the training data from a series of experiments. They presented the relationship between the size of a decision tree consistent with some training data and accuracy of the tree on test data. They showed that smaller decision trees are generally more recommendable for simple problems, but the average prediction accuracy of smaller decision trees is less consistent than slightly larger trees for many real problems. It means that slightly larger decision tree is more recommendable for complex problems than the smallest decision trees. Pazzani et al. (1994) showed the trading-off relationship between coverage and

accuracy for classification problems. The coverage is another description of pruning, which is highly related to the reliability of prediction errors. When coverage goes high, the prediction error becomes smaller. Therefore, there exists an optimal inductive decision tree to maximize a function of both accuracy and coverage.

Some research related to the disjunctive descriptions or multivariate combinations of nominal attributes has been studied (Ali and Pazzani, 1995b). However, most of them apply their multivariate decision-making process as a post-analysis after building a univariate decision tree. Murphy and Pazzani (1991) showed the conceptual approach of constructive induction of multivariate decision trees. They showed multivariate constructive induction has better performance than univariate systems. However, there was an NP-complete problem for generating the combinations of *multivariate* decision-makings. The meaning of Constructive induction algorithms is to create new complex attributes by combining existing attributes in ways that make the description of the concept easier. The fulfringe constructive induction algorithm (Oliveria and Vincentelli, 1993) belongs to a family of *constructive induction algorithms* that identify patterns near the fringes of the decision tree and uses them to build new attributes.

Learning from interpretations has been growing interest in recent years. Blockeel and De Raedt (1998) introduced the meaning of '*first-order logic*' and develop TILDE (Top-down Induction of Logical Decision Trees). The '*first-order logic*' is defined as simple logical combinations of attribute-value descriptions. TILDE was recent upgrade of Quinlan's predictive C4.5 algorithm. It employed logical queries, first-order upgrades of existing attribute-value descriptions, rather than just using attribute-value tests in nodes of a decision tree. This slightly complicates the classification process with *first-order* decision trees. Basically TILDE employs logical queries rather than just using attribute-value tests in nodes of a decision tree.

Murphy and Pazzani (1991) introduced *m-of-n concepts* that are also known as Boolean threshold functions. The '*m-of-n concepts*' means all possible logical combinations of m attributes among n. They provided 'GS algorithm' (m-of-n concept construction) to compare *multivariate* decision trees to conventional ones. However, it was NP-complete to

generate all possible combinations of *multivariate* decision-making descriptions. Ali and Pazzani (1995a) developed HYDRA to learn concept descriptions consisting of rules with relational and attribute-value conditions. It built more complex decision-makings rather than single attribute-value descriptions to reduce the prediction errors. However, it was required to build after the construction of a decision tree.

Sebag (1995) introduced 'the *second-order* understandability', which is defined as 'the operational understandability' of knowledge or the ability to provide justifications for the results that it produces. The definition of 'understandable justification' is a subpart of a knowledge base, which is both understandable (in the sense of the *first-order* understandability, e.g., attribute-value descriptions) and suffices to classify the current examples.

A new algorithm for the *second-order* decision-tree induction (SODI) generates a top-down decision tree with the consideration of the *bivariate* correlation of nominal attributes simultaneously. The '*bivariate* combinations of nominal attributes' means that the decision-making or hypothesis can be described by some logical combinations of a pair of nominal attributes for each decision node. We define the decision-making of *bivariate* nominal attributes as a *second-order* decision-making.

SODI shows that the hypothesis description for each decision node becomes more complex, but the size of the decision tree is much less than any conventional *univariate* decision trees. It works effectively when some of decision attributes are correlated so that the joint distribution of the class attribute with these attributes is not linearly independent. The method for removing redundant attributes is mathematically proved before introducing a new algorithm. A numerical analysis from nine well-known classification problems is performed to compare SODI to other algorithms of *univariate* decision trees.

1. Statement of problem

In this chapter the classification problem is defined how to build a classification model from nominal attributes only in order to minimize a user-defined penalized risk. In this

chapter three conventional methods are introduced for the comparison of our new TDIDT method: ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), and PART (Frank and Witten, 1998a). These conventional methods use univariate descriptions of decision-making. However, this univariate description may be not suitable for general classification problems especially when decision-making at a real classification decision boundary requires describing multiple attributes.

The motivation of the research comes from the research of Murphy and Pazzani (1991). They consider all possible dimensions of nominal attributes, so that its application becomes NP-complete. Therefore, in this chapter, it has been applied for all pairs of nominal attributes to build a decision tree, so that it is able to escape from NP-complete as well as to improve the prediction accuracy.

The information of an attribute is expressed as the entropy of knowledge it contains. That is, the uncertainty of knowledge from an attribute is a function of the information of the attribute. The information gain from an attribute means the reduction of uncertainty when the attribute is employed for explaining the target attribute, or class attribute. The construction of TDIDT for maximizing the information gains is not favorable because every attribute has the range of its own values, and it is possible for an attribute that has the largest range of values to maximize the information gains. Then, it makes larger tree, so that it may generate overfitting problems. Therefore, we need another criteria or objective function for the TDIDT construction to maximize information gain and to minimize the description of knowledge simultaneously. The information gain ratio is one of most favorable criteria for building a decision tree. The definition of he information gain ratio will be introduced later.

In this chapter we limited the classification problem with nominal attributes only. The objectives of this chapter are as following:

1. providing mathematical proof for eliminating redundant nominal attributes,
2. introducing a new TDIDT construction algorithm (SODI) to minimize information gain ratio for each decision tree branch, and
3. evaluating several TDIDT models by the penalized risk shown in (1.2).

For evaluating several decision trees, the policy of scoring user-defined penalties, as described in the previous chapter, is the essential. For the comparison of SODI with other TDIDT models, the penalty was scored from 0 to 5, indicating the lower score as the higher penalty, and the higher score as the fewer penalties. From all TDIDT results of the sample classification problem, each penalty can be computed with a unique distribution of values. Then, one can compute the sample mean m and sample standard deviation σ. Then, the scoring policy is shown as the following table.

Table 1. The scoring policy for each penalty criterion: (m and σ are the sample mean and sample standard deviation of a penalty distribution from all TDIDT models, respectively).

Range of Penalty Values	Score
(-∞, m-1.5σ]	5
(m-1.5σ, m-0.5σ]	4
(m-0.5σ, m+0.5σ]	3
(m+0.5σ, m+1.5σ]	2
(m+1.5σ, m+3.0σ]	1
(m+3.0σ, ∞)	0

To motivate the SODI algorithm mathematically some terminology needs to be introduced. We let Y be a discrete random variable, with unknown probability density $p(Y)$, that represents the class attribute, and $A_1,A_2,...,A_N$ represents the other (decision) attributes. The values of these attributes are denoted with the corresponding lower case letter, e.g. $a_{i1}, a_{i2},..., a_{in_i}$ are the values of attribute A_i. Since the classification in data mining does not assumed any probability distribution, any probability function in this thesis should be defined empirically. Therefore a priori probability of a class c_j can be computed as

$$p(Y = c_i) = N(c_i) / \sum_{c_j \in Y} N(c_j), \tag{2.1}$$

where $N(c)$ is the number of training instances whose class is c.

We attempt to arrive at a number that will measure the amount of uncertainty. Let X be a discrete random variable taking a finite number of possible values $x_1, x_2,..., x_n$ with probabilities $p_1, p_2,..., p_n$ respectively such that $p_i \geq 0$ for $i = 1,2,...,n$, $\sum_{i=1}^{n} p_i = 1$. Let h be a function defined on the interval [0, 1] and $h(p)$ be interpreted as the uncertainty associated

with the event $X = x_i,\ i = 1,2,...,n$ or the information conveyed by revealing that X has taken on the value x_i in a given performance of the experiment. For each n, we shall define a function H of the n variables $p_1,p_2,\ldots,p_n$. The function $H(p_1, p_2,..., p_n)$ is to be interpreted as the average uncertainty associated with the event $\{X = x_i\}$, i=1,2,…n, given by

$$H(p_1, p_2,..., p_n) = \sum_{i=1}^{n} p_i h(p_i) = -\sum_{i=1}^{n} p_i \log_2 p_i. \tag{2.2}$$

Especially, the right-handed side of the above equation is called as *the measure of uncertainty* or *Shannon's entropy* (or, so-called *information entropy* in this thesis) according to the Information Theory [26]. The following terminology is used for further mathematical description in this thesis:

Terminology

Y ≡ *the random variable of classes, so called as class attribute.*

$A_1,A_2,...,A_N$ ≡ *the attributes for class Y.*

$a_{i1}, a_{i2},..., a_{in_i}$ ≡ *the values of attribute* A_i.

$p(Y)$ ≡ *the empirical probability density function (pdf) of classes of Y.*

$p(Y \mid A_i=a_{il})$ ≡ *the conditional empirical pdf of Y given attribute* $A_i = a_{il}$.

$p\{Y \mid (A_{i_1},\cdots,A_{i_m})\}$ ≡ *the conditional empirical pdf of Y given attributes* $(A_{i_1},A_{i_2},\cdots,A_{i_m})$.

$p_{A_i}(a_{ik})$ ≡ *the marginal empirical pdf of attribute* A_i *with respect to* $A_i = a_{ik}$.

$p_{A_i,A_j}(a_{ik}, a_{jl})$ ≡ *the joint empirical pdf of attributes* A_i *and* A_j *with respect to* $A_i=a_{ik}$ *and* $A_j=a_{jl}$.

$p_{A_j|A_i}(a_{jl} \mid a_{ik})$ ≡ *the conditional empirical pdf of attributes* A_j *on the condition of* A_i *with respect to* $A_i = a_{ik}$ *and* $A_j = a_{jl}$.

$H(Y)$ ≡ *the overall information entropy of classes without any attribute information. i.e.,*

$$H(Y) = -\sum_{y\in\{Y\}} p(y)\cdot\log_2 p(y). \tag{2.3}$$

$H(Y \mid A_i = a_{ik})$ ≡ *the information entropy of classes when an attribute* A_i *is provided to the specific value of* a_{ik}. *i.e.,*

$$H(Y \mid A_i = a_{ik}) = -\sum_{y \in \{Y\}} p(y \mid A_i = a_{ik}) \cdot \log_2 p(y \mid A_i = a_{ik}). \quad (2.4)$$

$H_{A_i}(Y)$ ≡ *the average information entropy of classes with the condition of attribute* A_i.

$$\textit{i.e.,}\ H_{A_i}(Y) = H(Y \mid A_i) = \sum_{k=1}^{n_i} H(Y \mid A_i = a_{ik}) p_{A_i}(a_{ik}). \quad (2.5)$$

$H\{Y \mid (A_i = a_{ik}, A_j = a_{jl})\}$ ≡ *the information entropy of classes when attributes* A_i *and* A_j *are provided to the specific values of* a_{ik} *and* a_{jl} *respectively. i.e.,*

$$H\{Y \mid (A_i = a_{ik}, A_j = a_{jl})\} = -\sum_{y \in \{Y\}} p\{y \mid (A_i = a_{ik}, A_j = a_{jl})\} \cdot \log_2 p\{y \mid (A_i = a_{ik}, A_j = a_{jl})\}. \quad (2.6)$$

$H_{A_i A_j}(Y)$ ≡ *the average information entropy of classes with the condition of attributes, both* A_i *and* A_j. *i.e.,*

$$H_{A_i A_j}(Y) = H(Y \mid A_i, A_j) = \sum_{k=1}^{n_i} \sum_{l=1}^{n_j} H\{Y \mid (A_i = a_{ik}, A_j = a_{jl})\} p_{A_i, A_j}(a_{ik}, a_{jl}). \quad (2.7)$$

$H(A_i, A_j)$ ≡ *the average information entropy of attributes* A_i *and* A_j. *i.e.,*

$$H(A_i, A_j) = -\sum_{k=1}^{n_i} \sum_{l=1}^{n_j} p_{A_i, A_j}(a_{ik}, a_{jl}) \cdot \log_2 p_{A_i, A_j}(a_{ik}, a_{jl}). \quad (2.8)$$

[COMMENT] The following equation represents the class information entropy of the conditional attribute of a previous conditional attributes is the same with the entropy of two simultaneous conditional attributes:

$$H_{A_j \mid A_i}(Y) = H\{(Y \mid A_j) \mid A_i\} = H\{(Y \mid A_i) \mid A_j\} = H_{A_i A_j}(Y). \quad (2.9)$$

We are now looking for the best description of decision attributes for the class attribute. In other words, the objective is to minimize the prediction error between the class attributes and the description of decision attributes. The following proposition is useful for understanding the characteristics of information theory, or entropy of knowledge. For the proof of this proposition, see appendix A.

PROPOSITION 2.1. PROPERTIES OF UNCERTAINTY (OR, ENTROPY) OF KNOWLEDGE

(1) $H(A_i, A_j) = H(A_i) + H(A_j \mid A_i)$. (2.10)

(2) A_i *and* A_j *are independent* $\Leftrightarrow H(A_i, A_j) = H(A_i) + H(A_j)$. (2.11)

(3) $H(A_i) \le H(A_i, A_j) \le H(A_i) + H(A_j)$. (2.12)

(4) $H_{A_i A_j}(Y) = H_{A_i}(Y) + H_{A_j|A_i}(Y) - H(Y)$. (2.13)

(5) $H_{A_1 A_2 \cdots A_n}(Y) = H_{A_1}(Y) + H_{A_2|A_1}(Y) + \cdots + H_{A_n|(A_1 A_2 \cdots A_{n-1})}(Y) - (n-1)H(Y)$. (2.14)

(6) A_i *and* A_j *are independent* $\Leftrightarrow H_{A_i A_j}(Y) = H_{A_i}(Y) + H_{A_j}(Y) - H(Y)$. (2.15)

(7) *Y is linearly dependent on* $\{A_1, A_2, \ldots, A_n\}$, *which are linearly independent each other, if and only if* $(n-1)H(Y) = H_{A_1}(Y) + H_{A_2}(Y) + \cdots + H_{A_n}(Y)$. (2.16)

Proposition 2.1(1) shows how to compute the information entropy of two correlated attributes, and 2.1(2) shows the total entropy of two independent attributes can be computed as the sum of individual information attributes. Proposition 2.1(3) shows the range of the information entropy for any pair of attributes. Proposition 2.1(4) and 2.1(5) respectively represent how to compute the information entropy of a class when two or multiple attributes are known. Especially Proposition 2.1(6) and 2.1(7) show how to compute the information entropy of a class when two or more independent attributes are known.

2. Elimination of redundant nominal attributes

For real world problem data is possibly collected redundantly. It makes some overhead to manipulate or extract some knowledge. Sometimes, discovering association rules among decision attributes is one of the most important research topics in data mining applications. As mentioned before, data is not perfectly provided to analyze. It is possible for the values of some attributes to be false or missing. If we can find some correlation among attributes, we can adjust some wrong data. The objective of this section is to provide a pre-process before the analysis of classification problems by detecting and eliminating redundant nominal attributes. The elimination of redundant attributes is equivalent to the dimension reduction of input space. For large dataset like the real world problem, the dimension reduction of input space makes the progress of analysis more efficient and simpler.

Liang *et al.* (1998) developed the "REVEAL" (REVerse Engineering ALgorithm) to find redundant attributes for a genetic-networks application [27]. They found the mutual information computational methods to maximize functional inference from large data sets such as human genomes, and applied the *mutual information* as defined as below. By systematically analyzing the mutual information between input states and out states, one is able to infer the sets of input elements covering each element or gene in the network. However, there is some weakness to apply: the number of combinations for all genes grows exponentially, and experimental data must include all 2^k numbers of {input, output} combinations to predict exactly the correct Boolean functions from k input elements. The *mutual information* is defined as follows:

DEFINITION 2.1. MUTUAL INFORMATION & MUTUAL INFORMATION MATRIX

The mutual information is defined as

$$M(A_i, A_j) = H(A_i) + H(A_j) - H(A_i, A_j), \tag{2.17}$$

where $H(A_i)$ is the Shannon's entropy of a dataset (or instances) when an attribute A_i is known, and $H(A_i, A_j)$ is the Shannon's entropy when both attributes A_i and A_j are known. Furthermore, the mutual information matrix is defined as

$$\mathbf{M} = [M(A_i, A_j)]_{n\times n} \text{ for all } i,j = 1,2,...,n, \tag{2.18}$$

where $M(A_i, A_i) = H(A_i)$.

The mutual information indicates the ability to predict the value of one variable based on the other. That is, it represents the reduction in uncertainty of an attribute due to the knowledge of the other. The following proposition describes the characteristics of the mutual information.

Proposition 2.2. PROPERTIES OF MUTUAL INFORMATION

(1) A_i *and* A_j *are independent* $\Leftrightarrow M(A_i, A_j) = 0$. (2.19)

(2) A_j *depends on* $A_i \Leftrightarrow M(A_i, A_j) = H(A_i)$. (2.20)

(3) $0 \le M(A_i, A_j) \le \min\{H(A_i), H(A_j)\}$. (2.21)

(4) *Y is linearly dependent on* $\{A_1, A_2, \ldots, A_n\}$*, which are linearly independent, if and only if* $H(Y) = M(Y, A_1) + M(Y, A_2) + \cdots + M(Y, A_n)$ (2.22)

Proposition 2.2(2) shows the mutual information of two linearly dependent attributes becomes an information entropy of a single attribute, and 2.2(3) shows the actual range of a mutual information value of two attributes. Proposition 2.2(4) represents the relationship between the class information entropy and the sum of mutual information for all independent attributes. The mutual information matrix was defined as a matrix consisting of all pairs of mutual information. All pairs of mutual information contain all possible of correlation among attributes. It turns out mutual information matrix can reveal all linearly dependent attributes with all combinations. The mutual information matrix ***M*** has the following properties:

Proposition 2.3. PROPERTIES OF MUTUAL INFORMATION MATRIX

(1) ***M*** *is symmetric, and the eigenvalues of* ***M*** *are real.*

(2) *If* $\sigma(\mathbf{M}) = \{\lambda_1, \lambda_2, \cdots, \lambda_N\}$, $p(t) = \prod_{i=1}^{N} (t - \lambda_i)$. (2.23)

(3) *There are m linearly dependent attributes in* $\{A_1, A_2, \ldots, A_N\}$.

$\Leftrightarrow p(0) = p^{(1)}(0) = \ldots = p^{(m-1)}(0) = 0,\ p^{(m)}(0) \neq 0 \quad (1 \le m \le N-1)$.

The proposition 2.3 is well known for the characteristics of a real symmetric matrix, and the property (3) gives the number of linearly dependent attributes. However, it is hard to analyze the characteristic polynomial for finding those dependent attributes. The following theorem provides more efficient approach to eliminate redundant nominal attributes.

THEOREM 2.4. ELIMINATION OF REDUNDANT NOMINAL ATTRIBUTES

There are m linearly dependent attributes if and only if the corresponding columns of the mutual information matrix are linearly dependent with nonnegative linear combinations.

PROOF

($\Rightarrow$) Suppose $S_{\mathbf{B}} = \{A_{i_1}, A_{i_2}, \ldots, A_{i_{N-m}}\}$ is the basis for the input space, and $S_{\mathbf{N}} = \{A_1, A_2, \ldots, A_N\} - S_{\mathbf{B}} = \{A_{j_1}, A_{j_2}, \ldots, A_{j_m}\}$ is the set of dependent attributes. Then, the original mutual information

matrix is equivalent to $\mathbf{M} = \begin{pmatrix} \mathbf{B} & \mathbf{N_B} \\ \mathbf{N_B^T} & \mathbf{N} \end{pmatrix}$, where **B** is the mutual information matrix of the basis, $S_{\mathbf{B}}$, and **N** is that of $S_{\mathbf{N}}$. Note that **B** and **N** are symmetric by the proposition 2.3(1). Suppose $\forall A_{j_k} (\in S_N)$ is linearly dependent on $S_B^{(k)} \equiv \{A_{q_1}, A_{q_2}, ..., A_{q_{s(k)}}\} \subset S_B$, where $s(k)$ is the number of independent attributes. Then, $H(A_{j_k} \mid S_B^{(k)}) = 0$, because $p\{A_{j_k} \mid (A_{q_1}, A_{q_2}, ..., A_{q_{s(k)}})\} = 0$ or 1. From the proposition 2.1(5),

$$\{s(k)-1\}H(A_{j_k}) = H(A_{j_k} \mid A_{q_1}) + H\{A_{j_k} \mid (A_{q_2} \mid A_{q_1})\} + \cdots + H[A_{j_k} \mid \{A_{q_{s(k)}} \mid (A_{q_1}, A_{q_2}, ..., A_{q_{s(k)-1}})\}]. \quad (2.24)$$

Because S_B is the basis of the input space, there exist $\omega_1, \omega_2, \ldots, \omega_{N-m}$ such that

$$H(A_{j_k} \mid A_{i_1}) + H\{A_{j_k} \mid (A_{i_2} \mid A_{i_1})\} + \cdots + H[A_{j_k} \mid \{A_{i_{N-m}} \mid (A_{i_1}, A_{i_2}, ..., A_{i_{N-m-1}})\}]$$
$$= \omega_1 H(A_{j_k} \mid A_{i_1}) + \omega_2 H(A_{j_k} \mid A_{i_2}) + \cdots + \omega_{N-m} H(A_{j_k} \mid A_{i_{N-m}}), \quad (2.25)$$

where $0 \le \omega_1, \omega_2, \ldots, \omega_{N-m} \le 1$.

Therefore, there exist nonnegative $w_1, w_2, \ldots, w_{N-m}$ such that

$$\{s(k)-1\}H(A_{j_k}) = w_1 H(A_{j_k} \mid A_{i_1}) + w_2 H(A_{j_k} \mid A_{i_2}) + \cdots + w_{N-m} H(A_{j_k} \mid A_{i_{N-m}}).$$
$$\Rightarrow \left(\sum_{p=1}^{N-m} w_p - s(k) + 1 \right) H(A_{j_k}) = \sum_{p=1}^{N-m} w_p M(A_{j_k}, A_{i_p}). \quad (2.26)$$

Note that $\sum_{p=1}^{N-m} w_p = \sum_{n=1}^{s(k)} w_{q_n} \le s(k)$. It must be true that $\sum_{p=1}^{N-m} w_p \ge s(k)-1$, because the right-handed side of (2.26) is nonnegative. If $\sum_{p=1}^{N-m} w_p = s(k)-1$, $M(A_{j_k}, A_{i_p}) = 0$ when $w_p \ne 0$. It implies A_{j_k} is linearly independent on A_{i_p}. Also, $w_p = 0$ if $A_{i_p} \notin S_B^{(k)}$. Therefore, A_{j_k} is linearly independent on S_B. It is the contradiction. Therefore, $\sum_{p=1}^{N-m} w_p > s(k)-1$. Then, (2.26) is equivalent to

$$H(A_{j_k}) = \sum_{p=1}^{N-m} w_p \left(\sum_{p=1}^{N-m} w_p - s(k) + 1 \right)^{-1} M(A_{j_k}, A_{i_p}) = \mathbf{e}_k^T \mathbf{N}_{\mathbf{B}}^T \mathbf{x}_k, \quad (2.27)$$

where

$$\mathbf{x}_k = \left(\sum_{p=1}^{N-m} w_p - s(k) + 1 \right)^{-1} [w_1 \quad w_2 \quad \cdots \quad w_{N-m}]^T. \quad (2.28)$$

Let $\forall A_{j_l} \in S_N$ for $l = 1,2,....,m$. Because A_{j_l} is linearly dependent on $S_{\mathbf{B}}$, there exists a linear function $L_l(\cdot)$, such that

$$A_{j_l} = L_l(A_{i_1}, A_{i_2},..., A_{i_{N-m}}). \tag{2.29}$$

Therefore, from proposition 2.1(5) and equations (2.24) and (2.29),

$$\begin{aligned} H(A_{j_k} \mid A_{j_l}) &= H\{A_{j_k} \mid L_l(A_{i_1}, A_{i_2},..., A_{i_{N-m}})\} \\ &= \sum_{n=1}^{N-m} \beta_{nl} H(A_{j_k} \mid A_{i_n}) - \{s(k)-1\} H(A_{j_k}) \end{aligned}, \tag{2.30}$$

where $s(k) = \left\lceil \sum_{n=1}^{N-m} \beta_{nl} \right\rceil^+$. (2.31)

Therefore, from equations (2.17), (2.27), and (2.30),

$$\begin{aligned} M(A_{j_k}, A_{j_l}) &= H(A_{j_k}) - H(A_{j_k} \mid A_{j_l}) \\ &= \left(s(k) - \sum_{n=1}^{N-m} \beta_{nl} \right) H(A_{j_k}) + \sum_{n=1}^{N-m} \beta_{nl} \left\{ H(A_{j_k}) - H(A_{j_k} \mid A_{i_n}) \right\} \\ &= \left(s(k) - \mathbf{1}^T \mathbf{b}_l \right) H(A_{j_k}) + \sum_{n=1}^{N-m} \beta_{nl} M(A_{j_k}, A_{i_n}) \; = \left(s(k) - \mathbf{1}^T \mathbf{b}_l \right) \mathbf{e}_k^T \mathbf{N}_{\mathbf{B}}^T \mathbf{x}_k + \mathbf{e}_k^T \mathbf{N}_{\mathbf{B}}^T \mathbf{b}_l \\ &= \mathbf{e}_k^T \mathbf{N}_{\mathbf{B}}^T \left\{ \left(s(k) - \mathbf{1}^T \mathbf{b}_l \right) \mathbf{x}_k + \mathbf{b}_l \right\}, \end{aligned} \tag{2.32}$$

where

$$\mathbf{b}_l = (\beta_{1l}, \beta_{21l},..., \beta_{N-m,l}) \geq \mathbf{0}. \tag{2.33}$$

If $k = l$, then

$$M(A_{j_k}, A_{j_l}) = M(A_{j_k}, A_{j_k}) = H(A_{j_k}) + H(A_{j_k}) - H(A_{j_k}, A_{j_k}) = H(A_{j_k}). \tag{2.34}$$

From (2.27), (2.32), and (2.34), for $k = l$,

$$\left(s(k) - \mathbf{1}^T \mathbf{b}_k \right) \mathbf{x}_k + \mathbf{b}_k = \mathbf{x}_k \Rightarrow \mathbf{x}_k = \frac{\mathbf{b}_k}{\mathbf{1}^T \mathbf{b}_k - s(k) + 1} \geq \mathbf{0}. \tag{2.35}$$

Therefore, (2.32) can be rewritten as

$$M(A_{j_k}, A_{j_l}) = \mathbf{e}_k^T \mathbf{N}_{\mathbf{B}}^T \left\{ \left(s(k) - \mathbf{1}^T \mathbf{b}_l \right) \mathbf{x}_k + \mathbf{b}_l \right\} = \mathbf{e}_k^T \mathbf{N}_{\mathbf{B}}^T \left\{ \frac{\left(s(k) - \mathbf{1}^T \mathbf{b}_l \right)}{\mathbf{1}^T \mathbf{b}_k - s(k) + 1} \mathbf{b}_k + \mathbf{b}_l \right\}$$

$$= \mathbf{e}_k^T \mathbf{N}_{\mathbf{B}}^T \left\{ \mathbf{x}_k + \mathbf{b}_l - \mathbf{b}_k \left(\frac{\mathbf{1}^T \mathbf{b}_l - s(k) + 1}{\mathbf{1}^T \mathbf{b}_k - s(k) + 1} \right) \right\}$$

$$= \mathbf{e}_k^T \mathbf{N}_\mathbf{B}^T \mathbf{x}_l, \tag{2.36}$$

where

$$\mathbf{x}_l = \mathbf{x}_k + \mathbf{b}_l - \mathbf{b}_k \left(\frac{\mathbf{1}^T \mathbf{b}_l - s(k) + 1}{\mathbf{1}^T \mathbf{b}_k - s(k) + 1} \right). \tag{2.37}$$

Then, $\mathbf{x}_l \geq \mathbf{0}$ because $\mathbf{x}_k \geq \mathbf{0}, \mathbf{1}^T \mathbf{b}_l \leq s(k)$, and $\mathbf{0} \leq \mathbf{b}_l \leq \mathbf{1}$ from equations (2.32) and (2.33).

From $M(A_{j_k}, A_{j_l}) = \mathbf{e}_k^T \mathbf{N} \mathbf{e}_l = \mathbf{e}_k^T \mathbf{N}_\mathbf{B}^T \mathbf{x}_l$, so that

$$\mathbf{e}_k^T \mathbf{N} = \mathbf{e}_k^T \mathbf{N}_\mathbf{B}^T [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_m] = \mathbf{e}_k^T \mathbf{N}_\mathbf{B}^T \mathbf{X}. \tag{2.38}$$

$$\therefore \mathbf{N} = \mathbf{N}_\mathbf{B}^\mathbf{T} \mathbf{X}. \tag{2.39}$$

From (2.27), (2.32), and the proposition 2.1(5),

$$\begin{aligned} H(A_{i_p} \mid A_{j_k}) &= H\{A_{i_p} \mid L_k(A_{i_1}, A_{i_2}, ..., A_{i_{N-m}})\} \\ &= \sum_{n=1}^{N-m} \alpha_n H(A_{i_p} \mid A_{i_n}) - (N-m-1) H(A_{i_p}) \quad \text{for } \exists \alpha_1, \alpha_2, ..., \alpha_{N-m} \\ &= \sum_{n \neq p} \alpha_n H(A_{i_p} \mid A_{i_n}) - (N-m-1-\alpha_p) H(A_{i_p}). \end{aligned} \tag{2.40}$$

$$\begin{aligned} M(A_{i_p}, A_{j_k}) &= H(A_{i_p}) - H(A_{i_p} \mid A_{j_k}) = (N-m-\alpha_p) H(A_{i_p}) - \sum_{n \neq p} \alpha_n H(A_{i_p} \mid A_{i_n}) \\ &= \left(N-m-\sum_{n=1}^{N-m} \alpha_p \right) H(A_{i_p}) + \sum_{n \neq p} \alpha_n \left\{ H(A_{i_p}) - H(A_{i_p} \mid A_{i_n}) \right\} \\ &= \left(N-m-\sum_{n=1}^{N-m} \alpha_p \right) H(A_{i_p}) + \sum_{n \neq p} \alpha_n M(A_{i_p}, A_{i_n}) \\ &= [\cdots \quad M(A_{i_p}, A_{i_{p-1}}) \quad H(A_{i_p}) \quad M(A_{i_p}, A_{i_{p+1}}) \quad \cdots] \cdot \mathbf{w}_k \\ &= \mathbf{e}_p^T \mathbf{B} \mathbf{w}_k, \end{aligned} \tag{2.41}$$

where

$$\mathbf{w}_k = \left[\alpha_1 \quad \cdots \quad \alpha_{p-1} \quad \left(N-m-\sum_{n=1}^{N-m} \alpha_p \right) \quad \alpha_{p+1} \quad \cdots \quad \alpha_{N-m} \right]^T \geq \mathbf{0}. \tag{2.42}$$

From $\mathbf{e}_p^T \mathbf{N}_\mathbf{B} \mathbf{e}_k = M(A_{i_p}, A_{j_k}) = \mathbf{e}_p^T \mathbf{B} \mathbf{w}_k$, so that $\mathbf{N}_\mathbf{B} \mathbf{e}_k = \mathbf{B} \mathbf{w}_k$. (2.43)

Let $\mathbf{W} = (\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_m)$. Then,

$$\therefore \mathbf{N_B} = \mathbf{BW}. \tag{2.44}$$

From (2.39) and (2.44),

$$\mathbf{N} = \mathbf{N_B^T X} = \mathbf{W^T B^T X} = \mathbf{W^T B X}. \tag{2.45}$$

Since $\mathbf{W^T BX} = \mathbf{N} = \mathbf{N^T} = \mathbf{X^T B^T W} = \mathbf{X^T BW}$, $\mathbf{X} = \mathbf{W}$. Therefore,

$$\mathbf{N} = \mathbf{W^T BW}. \tag{2.46}$$

From (2.44) and (2.46), it is obviously

$$\begin{pmatrix} \mathbf{N_B} \\ \mathbf{N} \end{pmatrix} = \begin{pmatrix} \mathbf{BW} \\ \mathbf{W^T BW} \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{W^T B} \end{pmatrix} \mathbf{W} = \begin{pmatrix} \mathbf{B} \\ (\mathbf{BW})^{\mathbf{T}} \end{pmatrix} \mathbf{W} = \begin{pmatrix} \mathbf{B} \\ \mathbf{N_B^T} \end{pmatrix} \mathbf{W}, \mathbf{W} \geq \mathbf{0}. \tag{2.47}$$

$\therefore$ Therefore, the corresponding columns of the mutual information matrix are linearly dependent with nonnegative linear combinations.

($\Leftarrow$) Suppose m columns of the mutual information matrix are linearly dependent on other columns. Then, there exists a nonzero vector **x** such that **Mx** = **0**. It implies **x** is an eigenvector for the zero eigenvalue. Because m columns of **M** are linearly dependent, there exist m orthogonal eigenvectors for the zero eigenvalue: i.e., $\mathbf{M}\begin{pmatrix} \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_m} \end{pmatrix} = \mathbf{MX} = \mathbf{0}$. It means that **M** is equivalent to $\begin{pmatrix} \mathbf{B} & \mathbf{N_B} \\ \mathbf{N_B^T} & \mathbf{N} \end{pmatrix}$, where *rank*(**M**) = *rank*(**B**) = *N*-*m*. So, there exists a nonzero matrix, $\mathbf{X} = (\mathbf{x_B}, \mathbf{x_N})^{\mathrm{T}}$, such that

$$\begin{pmatrix} \mathbf{B} & \mathbf{N_B} \\ \mathbf{N_B^T} & \mathbf{N} \end{pmatrix} \begin{pmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{pmatrix} = \begin{pmatrix} \mathbf{Bx_B} + \mathbf{N_B x_N} \\ \mathbf{N_B^T x_B} + \mathbf{N x_N} \end{pmatrix} = \mathbf{0}, \ \mathbf{x_N} \neq \mathbf{0}. \tag{2.48}$$

Therefore, $\mathbf{N} = \mathbf{N_B^T B^{-1} N_B}$ from $\mathbf{x_B} = -\mathbf{B^{-1} N_B x_N}$ and $\mathbf{N x_N} = -\mathbf{N_B^T x_B}$ ($\mathbf{x_N} \neq \mathbf{0}$).

Let $S_{\mathbf{B}} = \{A_{i_1}, A_{i_2}, ..., A_{i_{N-m}}\}$ and $S_{\mathbf{N}} = \{A_{j_1}, A_{j_2}, ..., A_{j_m}\}$ be the corresponding attribute sets of **B** and **N**, respectively. For any $A_{j_k} \in S_N$,

$$H(A_{j_k}) = \mathbf{e}_k^T \mathbf{N} \mathbf{e}_k = (\mathbf{N_B e}_k)^T (\mathbf{B^{-1} N_B e}_k) = (\mathbf{N_B})_k^T (\mathbf{B^{-1} N_B})_k, \tag{2.49}$$

where $(\mathbf{N_B})_k^T$ is the k$^{\text{th}}$ row of $\mathbf{N_B^T}$, and $(\mathbf{B^{-1} N_B})_k$ is the k$^{\text{th}}$ column of $(\mathbf{B^{-1} N_B})$.

$\because$ Note that $\mathbf{B^{-1} N_B} > 0$. Suppose $\mathbf{1^T} \cdot (\mathbf{B^{-1} N_B})_k < 1$. Then, the equation (2.49) is equivalent to

$$\left\{1 - \mathbf{1^T} \cdot (\mathbf{B^{-1} N_B})_k\right\} \cdot H(A_{j_k}) = -\sum_{p=1}^{N-m} \mathbf{e}_p^T (\mathbf{B^{-1} N_B})_k \, H(A_{j_k} \mid A_{i_p}) < 0. \tag{2.50}$$

It implies that there exists an attribute, A_{i_p}, so that $H(A_{j_k} \mid A_{i_p}) < 0$. It is contradicted.

$$\therefore \mathbf{1}^{\mathbf{T}} \cdot (\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k \geq 1. \tag{2.51}$$

(i) Suppose $(\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k = \mathbf{e}_p$ for some p. Then, (2.49) is equivalent to $H(A_{j_k}) = (\mathbf{N}_{\mathbf{B}})_k^T \mathbf{e}_p$ $= M(A_{j_k}, A_{i_p})$, so that $H(A_{j_k} \mid A_{i_p}) = 0$. It implies A_{j_k} is linearly dependent on A_{i_p}.

(ii) Suppose $(\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k \neq \mathbf{e}_p$ for all p, and $\mathbf{1}^{\mathbf{T}} \cdot (\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k = 1$. Then, (2.49) is equivalent to

$$H(A_{j_k}) = \left\{\mathbf{1}^{\mathbf{T}} \cdot (\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k\right\} H(A_{j_k}) - \sum_{p=1}^{N-m} x_{pk} H(A_{j_k} \mid A_{i_p}), \tag{2.52}$$

where $(\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k = \begin{pmatrix} x_{1k} & \cdots & x_{N-m,k} \end{pmatrix}^T$.

Because $H(\cdot)$ is a nonnegative function, $H(A_{j_k} \mid A_{i_p}) = 0$ when $x_{pk} \neq 0$. It implies that there exists a linear function such that $A_{j_k} = F(A_{i_p})$, when $x_{pk} \neq 0$. Suppose arbitrarily $x_{pk}, x_{qk} \neq 0$. Then, $A_{j_k} = F_p(A_{i_p}) = F_q(A_{i_q})$, so that A_{i_p} and A_{i_q} are linearly dependent. This is contradicted that $S_{\mathbf{B}}$ is the basis. So, if $\mathbf{1}^{\mathbf{T}} \cdot (\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k = 1$, $(\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k = \mathbf{e}_p$ for all p.

(iii) If $\mathbf{1}^{\mathbf{T}} \cdot (\mathbf{B}^{-1}\mathbf{N}_{\mathbf{B}})_k > 1$, the equation (2.49) is equivalent to

$$H(A_{j_k}) = (\mathbf{N}_{\mathbf{B}}\mathbf{e}_k)^T \mathbf{x}_k = \begin{pmatrix} M(A_{j_k}, A_{i_1}) \\ \vdots \\ M(A_{j_k}, A_{i_{N-m}}) \end{pmatrix}^T \mathbf{x}_k = \frac{\sum_{p=1}^{N-m} w_p M(A_{j_k}, A_{i_p})}{\sum_{p=1}^{N-m} w_p - s(k) + 1}, \tag{2.53}$$

$$\text{so that } x_{pk} = w_p \left(\sum_{p=1}^{N-m} w_p - s(k) + 1 \right)^{-1}, \text{ where } s(k) = \left\lceil \sum_{p=1}^{N-m} w_p \right\rceil^+. \tag{2.54}$$

Now, we claim that if

$$\left(\sum_{p=1}^{N-m} w_p - s(k) + 1 \right) x_{pk} = w_p, \tag{2.55}$$

then, $w = (w_1, w_2, \ldots, w_{N-m})$ is uniquely determined by $\mathbf{x}_k$. From the equation (2.55),

$$x_{pk} w_1 + \cdots + x_{pk} w_{p-1} + (x_{pk} - 1) w_p + x_{pk} w_{p+1} + \cdots + x_{pk} w_{N-m} = \{s(k) - 1\} x_{pk}. \tag{2.56}$$

For the matrix form,

$$\begin{pmatrix} x_{1,k}-1 & x_{1,k} & \cdots & x_{1,k} \\ x_{2,k} & x_{2,k}-1 & \cdots & x_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-m,k} & x_{N-m,k} & \cdots & x_{N-m,k}-1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{N-m} \end{pmatrix} \equiv \mathbf{Aw} = \{s(k)-1\}\mathbf{x}. \quad (2.57)$$

Then, A is nonsingular, because

$$A \sim \begin{pmatrix} x_{1,k}-1 & 1 & \cdots & 1 \\ x_{2,k} & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-m,k} & 0 & \cdots & -1 \end{pmatrix} \sim \begin{pmatrix} \sum_{p=1}^{N-m} x_{p,k}-1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{pmatrix}, \quad (2.58)$$

so $rank(\mathbf{A}) = N - m$. $\therefore \mathbf{w} = (N-m-1)\mathbf{A}^{-1}\mathbf{x} \Rightarrow \mathbf{w}$ is uniquely determined.

Therefore, according to (2.10) and (2.17), the equation (2.53) is equivalent to

$$\{s(k)-1\}H(A_{j_k}) = w_1 H(A_{j_k} \mid A_{i_1}) + w_2 H(A_{j_k} \mid A_{i_2}) + \cdots + w_{N-m} H(A_{j_k} \mid A_{i_{N-m}}). \quad (2.59)$$

According to (2.24) and (2.25), it implies that $H\{A_{j_k} \mid (A_{i_1}, A_{i_2}, ..., A_{i_{N-m}})\} = 0$, so that A_{j_k} is linearly dependent on $A_{i_1}, A_{i_2}, ..., A_{i_{N-m}}$.

$\therefore$ There are m linearly dependent attributes according to (i), (ii), and (iii).

□

The above theorem is very useful, because all redundant attributes are revealed when column operations are processed to determine the rank of the matrix. If a mutual information matrix is nonsingular, there is no redundant attribute. However, if a column of the matrix can be expressed by a linear combination of other attributes, it must be redundant. The above theorem implies that the size of rank of a mutual information matrix represents the minimum number of attributes that cover the range of data. This theorem is also able to apply feature selection: if there exist redundant attributes, the maximal description length of linearly dependent attributes can be removed for easy to develop an application of data mining. For easy to understand the meaning of this theorem, the three examples are as following:

Example 1) An attribute linearly depends on another attribute:

$$\text{Let } \mathbf{M} = [\mathbf{M_1} \quad \mathbf{M_2} \quad \mathbf{M_3}] = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 1 & 1 \\ 0.5 & 1 & 1 \end{bmatrix}. \text{ Then, } \mathrm{M}\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \mathbf{0}.$$

So, **M** has at least one linearly dependent attribute.

$\therefore p(t) = t(t^2 - 2.5t - 0.5)$, $p(0) = 0$, and $p'(t)\big|_{t=0} = 3t^2 - 5t + 0.5\big|_{t=0} \neq 0.$

Therefore, there is the only one linearly dependent attribute.

$[0 \quad -1 \quad 1] \cdot \mathbf{M}_3 = -M(A_2, A_3) + H(A_3) = 0$.

$\Leftrightarrow M(A_2, A_3) = H(A_2)$, so that A_3 is linearly dependent on A_2.

Example 2) an attribute linearly depends on others:

$$\text{Let } \mathbf{M} = [\mathbf{M_1} \quad \mathbf{M_2} \quad \mathbf{M_3}] = \begin{bmatrix} 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}. \text{ Then, } \mathrm{M}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \mathbf{0}.$$

So, **M** has at least one linearly dependent attribute. Note that A1 and A2 are linearly independent because $\mathrm{M}(A_1, A_2) = 0$. Then, $p(t) = t(t - 0.5)(t - 1.5)$, $\mathrm{p}(0) = 0$, and $\mathrm{p'(t)}\big|_{t=0} = 3t^2 - 4t + 0.75\big|_{t=0} \neq 0$. Therefore, there is the only one linearly dependent attribute.

$\therefore [1 \quad 1 \quad -1] \cdot \mathbf{M}_3 = M(A_1, A_3) + M(A_2, A_3) - H(A_3) = 0 \Leftrightarrow H(A_3) = M(A_1, A_3) + M(A_2, A_3)$

$\Leftrightarrow H(A_3) = H(A_3 \mid A_1) + H(A_3 \mid A_2)$, and also $M(A_1, A_2)=0$. $\therefore A_3$ is linearly dependent on both A_1 and A_2.

Example 3) two attributes are linearly dependent on others:

$$\text{Let } \mathbf{M} = \begin{pmatrix} \mathbf{B} & \mathbf{N_B} \\ \mathbf{N_B^T} & \mathbf{N} \end{pmatrix} = \left[\begin{array}{cc:cc} 1/2 & 1/4 & 1/2 & 5/8 \\ 1/4 & 1/2 & 1/4 & 1/2 \\ \hdashline 1/2 & 1/4 & 1/2 & 5/8 \\ 5/8 & 1/2 & 5/8 & 7/8 \end{array}\right] \Rightarrow \left\{ \begin{array}{l} \mathbf{B}^{-1} = \frac{1}{3}\begin{pmatrix} 8 & -4 \\ -4 & 8 \end{pmatrix} \\ \mathbf{N_B B}^{-1} = \begin{pmatrix} 1 & 1 \\ 0 & 0.5 \end{pmatrix} \end{array} \right\}.$$

Therefore, $\mathbf{N} = \mathbf{N_B^T N_B B}^{-1}$.

$$H(A_3) = \begin{pmatrix} M(A_3, A_1) \\ M(A_3, A_2) \end{pmatrix}^T \begin{pmatrix} 1 \\ 0 \end{pmatrix} = M(A_3, A_1) = H(A_3) - H(A_3 \mid A_1) \Leftrightarrow H(A_3 \mid A_1) = 0$$

$\Leftrightarrow A_3$ is linearly dependent on A_1.

$$H(A_4) = \begin{pmatrix} M(A_4, A_1) \\ M(A_4, A_2) \end{pmatrix}^T \begin{pmatrix} 1 \\ 0.5 \end{pmatrix} = M(A_4, A_1) + 0.5M(A_4, A_2)$$

$$\Leftrightarrow H(A_4) = 2H(A_4 \mid A_1) + H(A_4 \mid A_2).$$

If A_4 is linearly dependent on (A_1, A_2), $H(A_4) = H(A_4 \mid A_1) + H\{A_4 \mid (A_2 \mid A_1)\}$.

Now, we claim $H\{A_4 \mid (A_2 \mid A_1)\} = H(A_4 \mid A_1) + H(A_4 \mid A_2)$. From $M(A_4, A_1) = 5/8$,

$H(A_4 \mid A_1) = H(A_4) - M(A_4, A_1) = 1/4$. Similarly, $H(A_4 \mid A_2) = 3/8$.

$H\{A_4 \mid (A_2 \mid A_1)\} = H(A_4) - H(A_4 \mid A_1) = 7/8 - 1/4 = 5/8 = H(A_4 \mid A_1) + H(A_4 \mid A_2)$.

$\therefore A_4$ is linearly dependent on $\{A_1, A_2\}$.

3. SODI: a new algorithm of TDIDT with nominal attributes

In a recent paper, we developed the Second-Order Decision-tree Induction (SODI) algorithm, which generates a top-down decision tree with the consideration of the second-order decision-making of nominal attributes simultaneously (Lee and Olafsson, 2002). The SODI algorithm uses the information gain ratio suggested by Quinlan [6,7] as measure of the quality of attributes or combination of attributes.

Since $H(Y)$ represents the information entropy of classes without any attribute information, and similarly, $H_{A_i}(Y)$ is the average information entropy of classes when the attribute A_i is known, the information gain of the tree that branches at an attribute A_i is denoted with $G(A_i)$, defined as

$$G(A_i) = H(Y) - H_{A_i}(Y). \tag{2.60}$$

Let N_{ij} denote the subset of instances at the j^{th} internal node or end-leaf of the i^{th} tree, where $i=1,2,...,n,$ and $j=1,2,...k$. Let $p(N_{ij})$ denote the empirical probability of instances that are discovered at the j^{th} node of the i^{th} tree, i.e,

$$p(N_{ij}) = N_{ij} / \sum_j N_{ij}. \tag{2.61}$$

Let $S(A_i)$ denote the split entropy of a tree T that is branched by an attribute, A_i, i.e.,

$$S(A_i) = -\sum_{a_i=a_{i1}}^{a_{in_i}} p(A_i = a_i \mid T) \cdot \log_2 p(A_i = a_i \mid T)$$

$$= -\sum_{a_i=a_{i1}}^{a_{in_i}} \left(\frac{N(A_i = a_i)}{N_T} \right) \cdot \log_2 \left(\frac{N(A_i = a_i)}{N_T} \right), \tag{2.62}$$

where $N(A_i = a_i)$ represent the number of instances whose attribute A_i has the value a_i, and N_T is the total number of instances in this tree, so that

$$N_T = \sum_{a_i=a_{i1}}^{a_{in_i}} N(A_i = a_i) \tag{2.63}$$

Now, $G_R(A_i)$ denotes the information gain ratio of a tree due to A_i as following

$$G_R(A_i) = G(A_i) / S(A_i). \tag{2.64}$$

Consistent with the above notation, we let $G(A_i, A_j)$, $S(A_i, A_j)$, $G_R(A_i, A_j)$ denote the information gain, split entropy, and gain ratio, respectively, of a tree due to both A_i and A_j. Also, $H_{A_iA_j}(Y)$ denotes the average entropy of classes when both A_i and A_j are known. Finally, the incremental information gain $G(A_j | A_i)$ is defined as the information gain of class Y from an attribute A_j, where a subtree was branched by the previous attribute, A_i, i.e.,

$$G(A_j \mid A_i) = H_{A_i}(Y) - H_{A_j|A_i}(Y) = H_{A_i}(Y) - H_{A_iA_j}(Y). \tag{2.65}$$

The ID3 algorithm aims at quickly reducing entropy by selecting at each split node the attribute that has the highest information gain. Numerous other algorithms, such as the C4.5 algorithm that selects the attribute with the highest gain ratio, work in a similar fashion. The basic motivation behind the SODI algorithm is that such entropy reduction can be better achieved by sometimes using two attributes simultaneously in the decision node. In particular, the information gain of knowing two attributes is larger than or equal to the sum of the information gain of knowing each attribute independently, and the equality holds if and only if the attributes are independent. Thus, any two dependent attributes could reduce entropy faster if uses together, either conjunctively or disjunctively, in a split node. This is formalized in the following theorem.

THEOREM 2.5. PROPERTIES OF INFORMATION GAINS AND GAIN RATIO

a) *The average information entropy when two attributes are known is less than or equal to the entropy of knowing either one of the attributes*:

$$H_{A_iA_j}(Y) \le \min\{H_{A_i}(Y), H_{A_j}(Y)\}. \tag{2.66}$$

b) *The information gain of knowing two attributes is larger than or equal to the gain of knowing each of the thee attributes separately*:

$$\max\{G(A_i), G(A_j)\} \le G(A_i, A_j). \tag{2.67}$$

c) *The information gain of knowing two attributes can be calculated using the following relation*:

$$G(A_i, A_j) = G(A_i) + G(A_j \mid A_i) \le G(A_i) + G(A_j). \tag{2.68}$$

d) *Independent attributes can be characterized in terms of the following relationship between their joint information gain and split entropy*:

$$\begin{aligned} &A_i \text{ and } A_j \text{ are independent} \\ &\Leftrightarrow G(A_i, A_j) = G(A_i) + G(A_j), \quad S(A_i, A_j) = S(A_i) + S(A_j). \end{aligned} \tag{2.69}$$

e) *If the mutual information of knowing two attributes is zero, then the gain ratio of the two attributes is less than or equal to the larger of the two individual gain ratios*:

$$M(Y \mid A_i, Y \mid A_i) = 0 \text{ implies } G_R(A_i, A_j) \le \max\{G_R(A_i), G_R(A_j)\}. \tag{2.70}$$

PROOF

a) From the proposition 2.1(3), it is obviously true that

$$H\{(Y \mid A_j), A_i\} \le H(Y \mid A_j) + H(A_i) = H_{A_j}(Y) + H(A_i) \tag{a1}$$

Since A_i and A_j are linearly independent, from the proposition 2.1(1), the exact computation of $H\{(Y \mid A_j), A_i\}$ is

$$H\{(Y \mid A_j), A_i\} = H\{(Y \mid A_j) \mid A_i\} + H(A_i) = H_{A_i A_j}(Y) + H(A_i) \tag{a2}$$

From (a1) and (a2), $$H_{A_i A_j}(Y) \le H_{A_j}(Y). \tag{a3}$$

Similarly, $$H_{A_j A_i}(Y) \le H_{A_i}(Y). \tag{a4}$$

From (a3) and (a4), $$H_{A_i A_j}(Y) \le \min\{H_{A_i}(Y), H_{A_j}(Y)\}. \tag{a5}$$

b) By the definition of information gain from two arbitrary attributes, it is clear that

$$\begin{aligned} &G(A_i, A_j) = H(Y) - H_{A_i A_j}(Y) \ge H(Y) - \min\{H_{A_i}(Y), H_{A_j}(Y)\} \\ &= \max\{H(Y) - H_{A_i}(Y), H(Y) - H_{A_j}(Y)\} = \max\{G(A_i), G(A_j)\}. \end{aligned} \tag{b1}$$

c) From the definition of $G(A_j \mid A_i)$,

$$G(A_j \mid A_i) = H_{A_i}(Y) - H_{A_iA_j}(Y) = G(A_i, A_j) - G(A_i). \quad \text{(c1)}$$

$$\therefore G(A_i, A_j) = G(A_i) + G(A_j \mid A_i). \quad \text{(c2)}$$

If A_i and A_j are linearly independent, $H_{A_j|A_i}(Y) = H_{A_i}(Y) + H_{A_j}(Y) - H(Y)$, according to the proposition 2.1(4). Therefore, $G(A_j \mid A_i) = G(A_i)$ in this case. However, when A_j is linearly dependent on A_i, $G(A_j \mid A_i) = 0$, because $H_{A_j|A_i}(Y) = H_{A_j}(Y) = H_{A_i}(Y)$. From the characteristic property of information entropy of H(·) and G(·), the function G(·) has the maximum value when two attributes are linearly independent. In general, if A_i and A_j are correlated each other, $0 \le G(A_j \mid A_i) \le G(A_j)$. Therefore, finally

$$\therefore G(A_i, A_j) = G(A_i) + G(A_j \mid A_i) \le G(A_i) + G(A_j). \quad \text{(c4)}$$

d) From the proposition 2.1(4), if A_i and A_j are linearly independent, then

$$\begin{aligned} G(A_i, A_j) &= H(Y) - H_{A_iA_j}(Y) \\ &= H(Y) - \{H_{A_i}(Y) + H_{A_j}(Y) - H(Y)\} \\ &= G(A_i) + G(A_j). \end{aligned} \quad \text{(d1)}$$

From the definition of split information,

$$S(A_i, A_j) = \sum_{i=i_1}^{n_i} \sum_{j=j_1}^{n_j} p(A_i = a_i, A_j = a_j \mid T) \cdot \log_2 p(A_i = a_i, A_j = a_j \mid T)$$

$$= \sum_{i=i_1}^{n_i} \sum_{j=j_1}^{n_j} p(A_i = a_i \mid T) p(A_j = a_j \mid T) \cdot \log_2 p(A_i = a_i \mid T) p(A_j = a_j \mid T)$$

$$= \sum_{i=i_1}^{n_i} p(A_i = a_i \mid T) \cdot \log_2 p(A_i = a_i \mid T) \sum_{j=j_1}^{n_j} p(A_j = a_j \mid T)$$

$$+ \sum_{j=j_1}^{n_j} p(A_j = a_j \mid T) \cdot \log_2 p(A_j = a_j \mid T) \sum_{i=i_1}^{n_i} p(A_i = a_i \mid T)$$

$$= S(A_i) + S(A_j).$$

e) If $M(Y \mid A_i, Y \mid A_j) = 0$, $(Y \mid A_i)$ and $(Y \mid A_j)$ are independent. Therefore,

$$G(A_i, A_j) = G(A_i) + G(A_j), \quad S(A_i, A_j) = S(A_i) + S(A_j). \tag{e1}$$

Suppose $G_R(A_i) \geq G_R(A_j)$. Then, $G(A_i)/S(A_i) - G(A_j)/S(A_j) \geq 0$. Therefore,

$$\begin{aligned} G_R(A_i, A_j) - G(A_i) &= \frac{G(A_i) + G(A_j)}{S(A_i) + S(A_j)} - \frac{G(A_i)}{S(A_i)} \\ &= \frac{G(A_j)S(A_i) - G(A_j)S(A_j)}{S(A_i)\{S(A_i) + S(A_j)\}} \leq 0. \end{aligned} \tag{e2}$$

Similarly, if $G_R(A_i) \leq G_R(A_j)$, then $G_R(A_i, A_j) - G_R(A_j) \leq 0$. (e3)

From (e2) and (e3), $G_R(A_i, A_j) \leq \max\{G_R(A_i), G_R(A_j)\}$. (e4)

□

Theorem 1(a) implies that any pair of two attributes has less uncertainty than any of individual attributes. Similarly, Theorem 1(b) proves the information gain of a bi-attribute split is always larger than any of single attribute split. It also provides of a lower bound of the information of a bi-attribute split. On other hand, Theorem 1(c) provides an upper bound on the information of bi-attribute splits, and thus provides a weak condition for eliminating unnecessary pair-combinations among all attributes. Specifically, suppose there exist two smallest information gains from any single attribute. From these two attributes the joint information gain can be computed. If the sum of single information gains of a pair of other two attributes is larger than the joint information gain, then the computation of this joint information gain of that pair is not necessary. Theorem 1(d) shows the characteristics of a pair of independent attributes with respect to their information gain and split entropy. If $M(Y|A_i, Y|A_j) = 0$, the classification is independent on these two attributes. Theorem 1(e) shows that the information gain-ratio of two independent attributes is always less than any individual attribute. From this result the following corollary can be concluded.

COROLLARY 2.6. OPTIMAL CONDITION OF UNIVARIATE DECISION TREES

If all attributes affect the class attribute independently, the optimal solution is a univariate or first-order decision tree.

PROOF For any arbitrary attributes, A_i and A_j,

$$G_R(A_i, A_j) \le \max\{G_R(A_i), G_R(A_j)\}$$

because

$$M(Y \mid A_i, Y \mid A_j) = 0,$$

according to Theorem 2.5(e). That is, the gain ratio from any combination for arbitrary two attributes is no greater than the maximum gain ratio from single attributes. It is easily extended that a decision tree of any combination of multiple attributes has no greater information gain-ratio than that of single-attribute splits.

□

On the other hand, a second-order decision tree is possibly better than any single attribute decision trees if some attributes are correlated. There exists an optimal decision tree that consists of multi-attribute decision nodes. However, it is NP-complete to find an optimal solution by the searching all possible combinations (Murphy and Pazzani, 1991). Therefore, a second-order decision tree may be considered a heuristic approach to quickly get a good solution for the classification.

Suppose A_i is a node in the n^{th} depth of a first-order decision tree, and A_j is the only child node of A_i. If these consequent attributes are correlated to the classification, the number of branches of the joint condition, A_i and A_j, is less than the product of individual branches of these two attributes. By aggregating two correlated attributes the split entropy can be reduced even if the information gain remains the same. Therefore, the following corollary gives us a stronger motivation for the SODI algorithm.

COROLLARY 2.7. SUPERIORITY OF SECOND-ORDER DECISION TREES TO FIRST-ORDER ONES

If the class attribute is simultaneously correlated to a consequent attribute from a first-order decision tree, then there is a second-order decision tree that has better gain ratio than any other first-order decision tree.

PROOF From the first-order decision tree, if an attribute A_j succeeds an attribute A_i, the information gain of the second step is $G(A_j \mid A_i)$. Therefore,

$$G(A_i) + G(A_j \mid A_i) = \{H(Y) - H(Y \mid A_i)\} + \{H(Y \mid A_i) - H(Y \mid (A_j \mid A_i))\}$$

$$= G(A_i, A_j).$$

Also, the split information from A_i to A_j is $S(A_i)+S(A_j \mid A_i)$. Because the attributes are correlated, there exist some disjunctive logic combinations so that

$$S(A_i, A_j) < S(A_i) + S(A_j \mid A_i).$$

Therefore,

$$G_R(A_i, A_j) = G(A_i, A_j) / S(A_i, A_j)$$
$$> \big(G(A_i) + G(A_j \mid A_i)\big) / \big(S(A_i) + S(A_j \mid A_i)\big).$$

This implies the attribute (or, attributes) selection by gain ratio prefers a pair of attributes to any single attribute.

□

SODI construction algorithm

The new SODI algorithm does not only employ conjunctive expressions ('AND'), but also disjunctive expressions ('OR') to aggregate similar results from training instances. Also SODI adopts a new logical concept of 'OTHERWISE', which forces the aggregation of all trivial instances that are not included in any other logical conditions. The motivation for this is that it is important to aggregate trivial attributes that have very little information gain by the current split rules, so that the next split node may be introduced to obtain higher information gain.

More flexible logical description than 'AND' logic can reduce the number of decision rules or branches. The decision boundaries by conventional single attribute methods are orthogonal to each attribute, and intuitively it is clear that this requires more branches to approximate the ideal decision boundaries. In other words, adopting a pair of attributes can be a better approximation to describing nonlinear classification than conventional single attribute methods. This motivates the fact that SODI is able to improve the classification accuracy over single attribute decision trees.

To state a detailed description of the SODI algorithm, a few more terms and mathematical notations are required. We let T_i, denote the decision sub-tree of the i^{th} evolution, and L_{ij} be the j^{th} internal node or end-leaf of the i^{th} evolution of trees $i=1,2,...,n$,

$j=1,2,...k$. Consistent with our prior nation, $G(L_{nk})$ is the information gain of an end-leaf (L_{nk}) from a tree (T_n). i.e.,

$$G(L_{nk}) = -\sum_{c \in Y} P(Y = c \mid L_{nk}) \log_2 P(Y = c \mid L_{nk}). \tag{2.71}$$

and $G(T_n)$ is the average information gain of the decision tree, T_n. i.e.,

$$G(T_n) = \sum_{k=1}^{n_m} G(L_{nk}) P(L_{nk}). \tag{2.72}$$

Similarly, $S(T_n)$ is the split entropy of the decision tree T_n. i.e.,

$$S(T_n) = \sum_{k=1}^{n_m} p(L_{nk}) \log_2 p(L_{nk}). \tag{2.73}$$

and $G_R(T_n)$ is the gain ratio of the tree, i.e.,

$$G_R(T_n) = G(T_n) / S(T_n). \tag{2.74}$$

Let T_0 represent the whole tree with the root, N_0 denote the whole set of training instances, and $p(T_{n+1})$ denote the empirical probability that instances among N_{nk} belong to the subtree T_{n+1}. The information gain of constructive decision trees is recursively computed by:

$$\begin{aligned} G(T_n) &= \sum_{a_k \in A_p} G(T_{n+1}) p(T_{n+1}) \\ T_{n+1} &= T(N_{nk} \mid A_p = a_k) \text{ for } n = 0,1,2,..., \end{aligned} \tag{2.75}$$

where $T(N_{nk} \mid A_p = a_k)$ is a subtree of T_n branched at $A_p=a_k$, and $(N_{nk} \mid A_p = a_k)$ is the subset of instances with the value of $A_p=a_k$ among N_{nk}. Furthermore,

$$\begin{aligned} A_p &= \arg\max_{A \notin T_n} \left\{ \frac{G(N_{nk} \mid A)}{S(N_{nk} \mid A)} \right\}, \\ G\left(N_{nk} \mid A_p\right) &= \sum G(N_{nk} \mid A_p = a_k) \cdot p(N_{nk} \mid A_p = a_k), \\ S\left(N_{nk} \mid A_p\right) &= -\sum p(N_{nk} \mid A_p = a_k) \cdot \log_2 p(N_{nk} \mid A_p = a_k), \end{aligned} \tag{2.76}$$

where the probability $p(N_{nk} \mid A_p = a_k)$ can be computed empirically as follows:

$$p\left(N_k \middle| A_p = a_k\right) = \left| \left(N_{nk} \middle| A_p = a_k\right) \right| \Big/ \left| N_{nk} \right|. \tag{2.77}$$

There are two primary components to defining SODI. The first is the selection of an attribute or a pair of attributes for splitting, and the second is the pruning process that eliminates and combines branches while the tree is being constructed (i.e., pre-pruning or forward pruning). We start by describing the how attributes are selected and then discuss the pruning process.

```
Function SODI(R,S,DC)
   R: a set of attributes={A1,A2,…,AN};
   S: a set of training instances;
   C: a default class value;
Begin
If S is empty, return NULL;
   Let C be the dominant class index;
   If Pr(C)=Pr(DC), then C:=DC, else DC:=C;
   If H(R)<α, return a single node with class C;
   Sort by Gain Ratio: GR(A1)> GR(A2)>…> GR(AN);
   (Ai,Aj):=Find_Best_Pair(A1,A2,…,AN); // According to Theorem 1
   Let dj:=description of decision-making.
   Let Sj:=subset of S corresponding to dj.
   If GR(A1)> GR(Ai,Aj)
      {(dj,Sj)|j=1..m}:=SODI_Rules(S,A1);
      Let T be a tree with the root labeled A1;
      Rnew:=R-{A1};
   Else
      {(dj,Sj)|j=1..m}:=SODI_Rules(S,Ai,Aj);
      Let T be a tree with the root labeled (Ai,Aj);
      Rnew:=R-{Ai,Aj};
   End If
   For j=1 to m
      Add an arc labeled dj to T;
      Branch T from the arc with SODI(Rnew,C,DC);
      // SODI(Rnew,C,DC) is a recursive function.
   End For
   return T;
End.
```

Figure 4. SODI decision tree construction algorithm

Figure 4 shows the pseudo code for the SODI algorithm. As in any decision tree algorithm, the key issue is to select the order in which to use the attributes in the split nodes (see Section 2.1). To this end, SODI first constructs a list of attributed sorted according to their information gain. To find the best pair, the algorithm starts by considering the first two attributes from the list. If the gain ratio of the pair is higher than any of single attribute, the value becomes a lower bound for the gain ratio of all remaining pairs. The expected upper bound of the information gain ratio of a pair is the sum of information gains of these two

attributes divided by the maximum split information between those attributes. If it is lower than the current lower bound, this pair of attributes can be skipped. This process continues through the list of attributes until the best pair of attributes has been identified. In the worst case $n(n-1)/2$ information gain ration calculations are required to traverse the entire list. In practice, however, many candidate pairs can be eliminated by the bound of the best gain ratio, and finding the best pair is thus likely to take much fewer iterations.

In SODI, branches or decision arcs are aggregated while a decision tree is being constructed using a set of rules that we call the second-order logic descriptions (see figure 5). These rules can be thought of as a pre-pruning process, where the size of the tree is restricted as the tree is being constructed. The first three rules attempt to reduce the size of the tree by combining branches or decision arcs where there is no or little information gain in keeping all the branches. These three rules should be applied in order and if the first rule is satisfied there is no need to check the last two, and so forth. The fourth and final rule deals with small branches with little information gain and combines all such trivial branches into a single OTHERWISE branch. The details of the four pre-pruning rules are as follows:

Rule 1. Start by eliminating all arcs or branches where there is no or little reduction in entropy. Specifically, aggregate all decision arcs where the entropy after splitting relative to the entropy before splitting is less than some pre-specified constant $\alpha > 0$, that is

$$H(N_{nk} \mid T_n) / H(T_n) < \alpha \ . \tag{2.78}$$

Note that here N_{nk} is the subset of instances obtained by the k^{th} branch from T_n. Also note that the larger the constant α, the more aggressive the pruning, and vice versa. The default value of α in our implementation of SODI is set to $\alpha = 0.25$, which from numerical experience appears to give good performance.

The extreme case of this rule is $H(N_{nk} \mid T_n) = 0$, which means the instances in this subset have the same class and the decision node becomes a leaf node. This rule generalizes this concept to merging branches with almost pure classification.

Rule 2. If there are no more instances that satisfy Rule 1, then a majority dominance rule will be considered. Let $p(c_n \mid N_{nk})$ be the proportion of instances with class c_n in the N_{nk}. If some subsets of instances dominate a class c_n than any others, that is $p(c_n \mid N_{nk}) > \sum_{m \neq n} p(c_m \mid N_{nk})$, then these subsets can be merged.

Rule 3. If Rule 1 and Rule 2 do not apply, but the proportion of a class is significantly larger than any other classes (by some discrimination parameter, β), these subsets are still merged. The default value of β is 0.2 in the implementation of SODI.

Rule 4. The final rule combines small subsets that have negligible information gain. In particular, any subset where the size of the subset is very small (as a constant ε) and has gain ration smaller than α can be considered trivial and an overfitting problem may occur if such branches are included in the decision tree. Thus, all such branches are aggregated in an OTHERWISE branch, which could be split further in the next iteration using different attributes. The default value of ε is 3.

An illustrative example

In this section we illustrate the SODI algorithm described in Section 3.3 through a very simple classification problem with a class attribute Y that can take two values $Y \in \{X, O\}$, and four additional decision attributes A_1, A_2, A_3, and A_4. There are 25 instances in the training data set and those are shown in Table 2.

The global SODI parameters are set as α = 0.05, β = 0.10, and ε = 3. Figure 6(a) and 6(b) show the results of ID3 and SODI, respectively and both algorithms classify all of the training instances correctly (no training error). The tree size for ID3 is 22, the number of decision rules is 13, and the split entropy is 2.9778. On the other hand, the tree size for SODI is 11, the number of decision rules is 8, and the split entropy is 2.6970. Thus, SODI results in a much smaller decision tree and fewer decision rules.

```
Global Parameters:
   α: Approximation level (default:0.25);
   β: Discrimination level (default:0.2);
   ε: Minimum attractive number of instances (default: 3);
```

```
Function SODI_Rules(S,A,B)
   S: a set of training instances;
   A,B: decision attributes such that GR(A)>GR(B);
Begin
  If B is empty
      Let {dk=(ak)|ak∈A} be the mutual decision;
      Let Sk be the subset of S corresponding to dk.
  Else
      Let {dk=(ai,bj)|k=(i,j),ai∈A,bj∈B} be the mutual decision;
      Let Sk be the subset of S corresponding to dk.
  Endif
  Let Sp be the group of {Sk};
  Let dp be the condition of attributes,(A,B), corresponding to
Sp;
  For each class Ci:
   p:=1; dp:=FALSE; Sp:=∅;
   For rule_no = 1 to 3
      Repeat
        Find Sk such that it satisfies the Rule(rule_no)
        Sp := Sp ∪ Sk; dp := dp OR dk;
      Until Rule(rule_no) cannot satisfy all remained Sk;
      {(dj,Sj)|j=1..k} := Refine_Logics(Sp,dp); p:=p+k;
   End For
  End For
  For all ungrouped subsets,
    If (sizeOf(Sk)>ε) or (sizeOf(Sk)<=ε and G(Sk)/H(S)>α)
       p:=p+1; dp=(ai,bj) OR ak; Sp=Sk;
    Endif
  End For
  p:=p+1; dp:='OTHERWISE';
  Aggregate all ungrouped subsets {Sk} to Sp;
  return {(dj,Sj)|j=1..p};
End;
```

Figure 5. SODI Construction Rules

Table 2. A simple classification problem

A1	A2	A3	A4	Y	A1	A2	A3	A4	Y	A1	A2	A3	A4	Y
1	1	1	1	X	2	1	1	1	O	3	1	1	2	X
1	1	1	2	O	2	1	1	2	O	3	1	2	1	X
1	1	2	1	O	2	1	2	1	O	3	1	2	2	O
1	1	2	2	X	2	2	1	1	O	3	1	3	1	X
1	1	3	2	X	2	2	1	2	O	3	1	3	2	X
1	2	1	1	O	2	2	2	1	O	3	2	2	1	X
1	2	1	2	O	2	2	2	2	O	3	2	2	2	X
1	2	3	2	X						3	2	3	1	X
										3	1	1	1	O
										3	2	3	2	X

To understand how SODI achieves the reduction in tree size through the use of bi-attribute split nodes and a pre-pruning process (see Rule 1 – Rule 4 in Section 3.3), we consider the construction of the tree more closely. Start by noting that the information gain ratios of $A_1, A_2, A_3,$ and A_4 are 0.2486, 0.0015, 0.2274, and 0.0107, respectively. Therefore, A_1 and A_3 are the first two attributes on the ordered list of all attributes and are considered first. Also note that the gains of A_1 and A_3 are 0.3900 and 0.3532, respectively, and the gains of A_2 and A_4 are 0.0014 and 0.0107, respectively. Finally, the values of the split entropy for A_1 and A_3 are 1.5690 and 1.5535, respectively.

Representing the two highest gain ration attributes, the pair (A_1, A_3) is a candidate attribute pair for the first split node. The actual information gain of (A_1, A_3) is 0.5792, and its split entropy is 2.2774. Its gain ratio is 0.2543, which is bigger than 0.3900, the gain ratio of A_1, and the current lower bound of gain ratios for the first split is therefore taken as 0.2543. The next candidate is (A_1, A_4). The approximate upper bound of the gain ratio of (A_1, A_4) can be computed as follows:

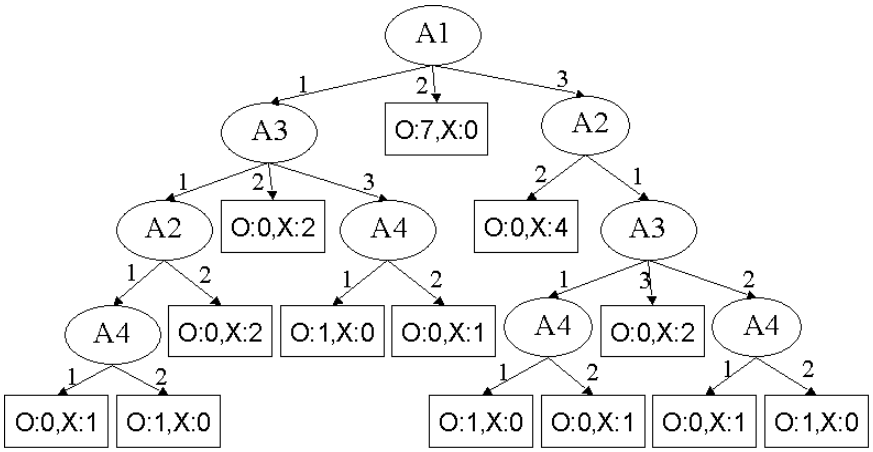
$$G_R(A_1, A_4) = \frac{G(A_1, A_4)}{S(A_1, A_4)} \le \frac{G(A_1) + G(A_4)}{\max\{S(A_1), S(A_4)\}} = \frac{0.2486 + 0.0107}{1.5690} = 0.1693$$

This is less than the current lower bound and the candidate (A_1, A_4) is thus rejected. Furthermore, similar calculation show that the upper bounds of gain ratios from any other

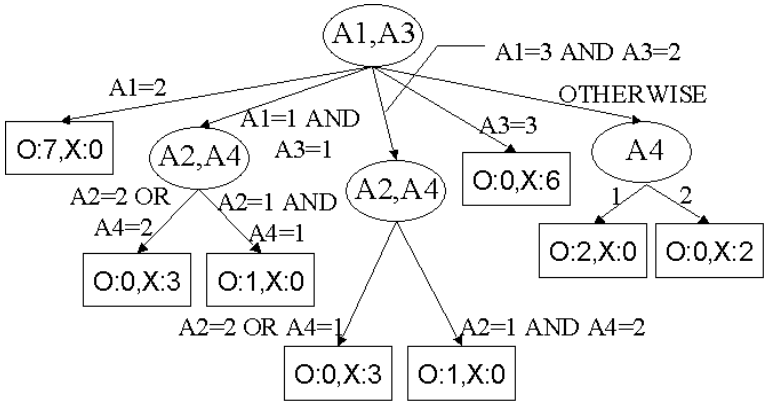
combinations are all less than the current lower bound. Therefore, the first decision node is selected to be (A_1, A_3).

Now when the split node has been selected, the next step is to consider all its possible values:

$$(A_1, A_3) = (1,1) \quad (A_1, A_3) = (2,1) \quad (A_1, A_3) = (3,1)$$
$$(A_1, A_3) = (1,2) \quad (A_1, A_3) = (2,2) \quad (A_1, A_3) = (3,2)$$
$$(A_1, A_3) = (1,3) \quad (A_1, A_3) = (2,3) \quad (A_1, A_3) = (3,3)$$



(a)



(b)

Figure 6. Numerical example for building TDIDT by (a) ID3 and (b) SODI.

At first glance this might indicate nine branches, but the pre-pruning rules must also be applied. First, lets consider the three cases where $A_1 = 2$. From Table 1 we observe that there is no instance where both $A_1 = 2$ and $A_3 = 3$. Furthermore, for the other two potential branches with $A_1 = 2$ both have all instances classified as $Y = O$. Therefore, $(A_1 = 2\ AND\ A_3 = 1)$ and $(A_1 = 2\ AND\ A_3 = 2)$ have zero entropy and can be combined by Rule 1. Thus, the three branches can be simplified to $A_1 = 2$ as shown in figure 6(b). Secondly, the decision arc $A_3 = 3$ from the decision node (A_1, A_3) is aggregated from $(A_1 = 1\ AND\ A_3 = 3)$ and $(A_1 = 3\ AND\ A_3 = 3)$ by the same reason. Thirdly, the sets of instances corresponding to $(A_1 = 1\ AND\ A_3 = 2)$ and $(A_1 = 3\ AND\ A_3 = 1)$ are both determined to be small as they have only two instances each, which is less that the threshold of ε = 3. They also have zero information gain and are thus combined in an OTHERWISE condition according to the Rule 4, which aggregates all small subsets of trivial unclassified instances. Thus, the nine potential branches become five branches as pre-pruning is applied.

From this example it is clear that the reason for why the SODI decision tree is appealing is two-fold: First, the use of bi-attribute splits allows for modeling of interactions between attributes. For example, 13 of the 25 instances are completely classified by the value of A_1 ($A_1 = 2$) or A_3 ($A_3 = 3$), but the remaining 12 instances require considering interactions. Second, the disjunctive and OTHERWISE logic allows for simplification of the tree. For example, two branches are combined into an OTHERWISE branch at the top level, that is then classified perfectly by A_4 at the next level. Also, for both of the split nodes involving A_2 and A_4 as a bi-attribute split, the use of a disjunctive OR branch allows us to combine what would otherwise be four branches into two. Thus, the combination of bi-attribute splits and extended logic descriptions makes for a simpler, easier to interpret, and potentially more useful tree.

Numerical analysis

The simple example in the previous section provides some intuition into why the SODI algorithm may perform well. In this section we present more extensive numerical

results from testing the SODI algorithm along with the comparisons with ID3 (Quinlan, 1986), C4.5 with pruning (Quinlan, 1993), and PART (Frank and Witten, 1998a). The ID3 and C4.5 algorithms are chosen for comparison as well known as both are widely used entropy-based decision tree algorithms. Also, Frank and Witten (1998a) developed the PART algorithm for inferring rules by repeatedly generating partial C4.5 decision trees, thus combining the two major paradigms for rule generation: creating rules from decision trees and the separate-and-conquer rule-learning technique

We analyzed eight classification problems that are widely used in the data mining literature (Witten and Frank, 1999):

1. Fitting contact lenses (5 attributes, 3 classes, 24 instances)
2. Balance scale weight & distance (4 attributes, 3 classes, 625 instances)
3. Breast cancer (9 attributes, 2 classes, 286 instances)
4. Chess end-game (36 attributes, 2 classes, 3196 instances)
5. 1984 United States Congressional voting (17 attributes, 2 classes, 435 instances)
6. Lymphography domain (17 attributes, 4 classes, 148 instances)
7. Mushroom records (22 attributes, 2 classes, 8124 instances)
8. Zoo classification (17 attributes, 7 classes, 101 instances)

The performance of the SODI algorithm compared to ID3, C4.5 and PART is shown in figure 7. Figure 7(a) shows the relative ratio of the sizes of the decision trees and figure 7(b) and 7(c) show the relative proportion of the training errors and the prediction errors for each problem, respectively. Finally, figure 7(d) shows the relative proportion of error gaps between training and testing results. The detailed values for the comparison of TDIDT models are shown in Table 3. Table 4 shows scoring result for user-defined penalty categories. However, the comparison is highly subject to user's preferences.

For problems 1, 2, and 4, the classification is fully described by the instances. Therefore, the objective of these problems is to find a simplest model of the classification within an acceptable error limit. For other problems, data sampling is processed to classify their objects. In these problems the estimated prediction error after the construction of

decision trees is more interest. Another interesting criterion for the decision tree selection is the gap between training errors and estimated prediction errors. Smaller gaps indicate a more reliable decision tree.

For the case of problem 1, SODI has the highest accuracy with a slightly larger TDIDT size than those of both C4.5 and PART. In the case of problem 2, ID3 empirically generates decision trees with the smallest training errors, but the estimated prediction errors are much higher than others. It implies ID3 tends to generate some overfitting problems. For the cases of problem 1, 2, and 4, SODI generates 58.3% smaller size of decision trees than ID3 and smaller training errors than either C4.5 or PART on average. SODI also compressed the decision trees about 67.8% of ID3 results on average as well as reduces the estimated prediction errors about 23% on the average of overall problems.

For the cases of both problem 3 and problem 5, C4.5 provided better solutions than any other methods with the consideration of the TDIDT size, the estimated prediction error, and the gap of errors. Even thought SODI provided slightly smaller prediction error than C4.5, the size of TDIDT is much larger than that of C4.5. It seems that SODI does not (without processing pruning steps) always dominate any other learning methods. For the case of problem 4, the results from C4.5, PART, and SODI have almost same performance. However, SODI has slightly better result than any other methods with the consideration of both training accuracy and the tree size. For the case of problem 6, SODI dominated any other methods except for the tree size. With an acceptably increased size of the tree, SODI provided the best accuracy with respect to both training and test. For the case of problem 7, SODI reduced the size of TDIDT more than any other methods. For the case of problem 8, any solution except for ID3 is acceptable, but PART built a slightly smaller decision tree construction than any other methods.

Figure 9 shows an example of TDIDT solutions of SODI. For the problem ‘Lymphography’, SODI generated a decision tree with 46 branches and 26 end leaves. The comparison of prediction accuracy of SODI with respect to C4.5 and PART is shown in (figure 10). The tables in (figure 10) are so-called ‘confusion’ matrices that show what number of instances for a class are correctly classified or misclassified simultaneously.

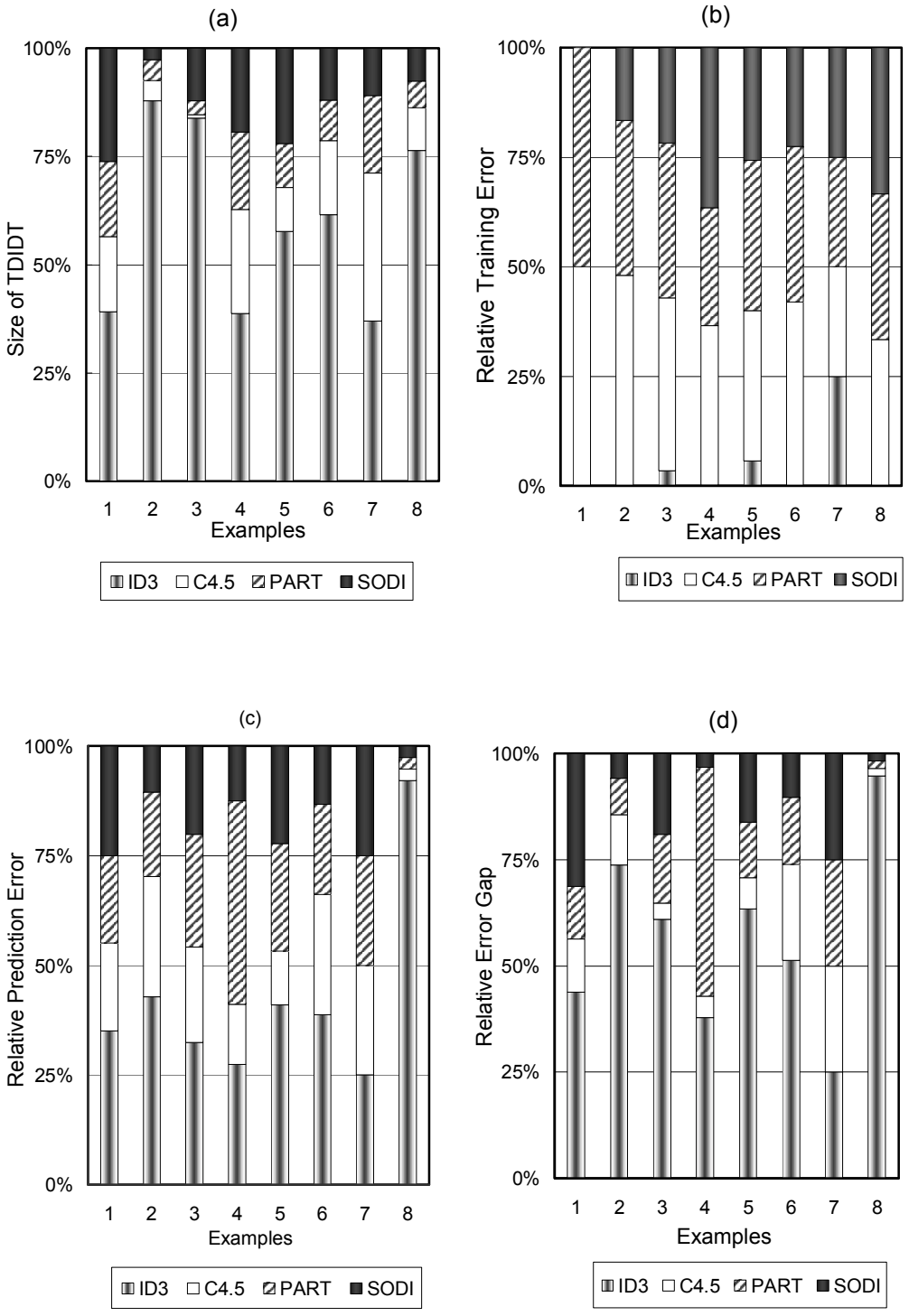


Figure 7. Performance evaluation of SODI compared with ID3, C4.5 and PART by (a) the proportion of number of decision rules, (b) training error, and (c) estimated prediction error, and (d) the gap between training error and prediction error. The smaller proportion is the better.

Table 3. Comparison of SODI with other univariate TDIDT algorithms

Problem Set	Method	Gain Ratio	N(leaves) / sizeOf(DT)	(a) Training Error (%)	(b) Prediction Error (%)	Gap (%) |(a)-(b)|	Std. Dev.[3] of Prediction Err
Contact Lenses	ID3	0.5476	9/15	0.00	29.17	29.17	9.73
	C4.5	0.6012	4/7	8.33	16.67	8.33	6.60
	PART	0.6012	4 rules	8.33	16.67	8.33	6.60
	SODI	0.6557	6/11	0.00	20.83	20.83	7.89
Balance Scale	ID3	0.1419	625/780	0.00	52.11	52.11	17.43
	C4.5	0.0813	33/41	24.96	33.33	8.37	12.22
	PART	0.1190	34 rules	18.40	23.47	6.07	8.38
	SODI	0.2741	19/23	8.64	12.80	4.16	4.57
Breast Cancer	ID3	0.1212	394/469	2.10	39.80	37.70	13.48
	C4.5	0.0907	4/6	24.13	26.53	2.40	9.86
	PART	0.0616	15 rules	21.68	31.63	9.95	11.89
	SODI	0.0919	57/80	13.29	24.67	11.83	8.25
Chess End-Game	ID3	0.2860	50/96	0.00	0.92	0.92	0.33
	C4.5	0.2880	31/59	0.34	0.46	0.12	0.15
	PART	0.2962	23 rules	0.25	1.56	1.31	0.58
	SODI	0.2943	25/47	0.34	0.42	0.08	0.14
1984 USA Voting	ID3	0.3121	34/69	0.46	6.76	6.30	2.20
	C4.5	0.5854	6/11	2.76	2.03	0.73	0.76
	PART	0.4266	6 rules	2.76	4.05	1.29	1.43
	SODI	0.5608	13/21	2.07	3.68	1.61	1.26
Lymphography	ID3	0.2463	72/94	0.00	33.33	33.33	11.11
	C4.5	0.2871	20/30	8.78	23.53	14.75	7.74
	PART	0.3420	11 rules	7.43	17.65	10.22	6.30
	SODI	0.3184	29/46	2.03	7.43	5.40	2.48
Mushroom	ID3	0.3005	27/37	0.00	0.00	0.00	0.0
	C4.5	0.3090	25/30	0.00	0.00	0.00	0.0
	PART	0.4058	13 rules	0.00	0.00	0.00	0.0
	SODI	0.5607	8/11	0.00	0.00	0.00	0.0
Classifying Zoo	ID3	0.3635	100/101	0.00	100.00	100.00	0.00
	C4.5	0.9342	13/21	0.99	2.86	1.87	0.10
	PART	0.9587	8 rules	0.99	2.86	1.87	0.10
	SODI	0.9342	10/15	0.99	2.86	1.87	0.10

3 Std. Dev. = sample standard deviation from cross validation results for each learning model.

Table 4. Comparison of SODI with other univariate TDIDT algorithms (scoring§ results from Table 3)

Problem Set	Method	N(leaves) / sizeOf(DT)	(a) Train. Err.	(b) Pred. Err.	Gap betw/ |(a)-(b)|	Std. Dev.ℨ of Pred. Err.	Average Score
Contact Lenses	ID3	3	5	-	-	-	4.00
	C4.5	5	3	-	-	-	4.00
	PART	5	3	-	-	-	4.00
	SODI	4	5	-	-	-	**4.50**
Balance Scale	ID3	0	5	-	-	-	2.50
	C4.5	3	3	-	-	-	3.00
	PART	3	4	-	-	-	3.50
	SODI	5	4	-	-	-	**4.50**
Breast Cancer	ID3	0	-	3	3	3	2.25
	C4.5	5	-	5	5	5	**5.00**
	PART	4	-	4	4	4	4.00
	SODI	3	-	5	4	5	4.25
Chess End-Game	ID3	3	5	-	-	-	4.00
	C4.5	4	3	-	-	-	3.50
	PART	5	4	-	-	-	**4.50**
	SODI	5	3	-	-	-	4.00
1984 USA Voting	ID3	3	-	3	3	3	3.00
	C4.5	5	-	5	5	5	**5.00**
	PART	5	-	4	4	4	4.25
	SODI	4	-	4	4	4	4.00
Lymphography	ID3	3	-	3	3	3	3.00
	C4.5	4	-	4	4	4	4.00
	PART	5	-	4	4	4	4.25
	SODI	4	-	5	5	5	**4.75**
Mushroom	ID3	3	-	5	5	5	4.50
	C4.5	3	-	5	5	5	4.50
	PART	4	-	5	5	5	4.75
	SODI	5	-	5	5	5	**5.00**
Classifying Zoo	ID3	0	-	0	0	5	1.25
	C4.5	3	-	5	5	4	4.25
	PART	5	-	5	5	4	**4.75**
	SODI	4	-	5	5	4	4.50

§Every criterion has been assigned to the same weight for each problem. The policy of both selecting criteria and making their scores for each problem is subject to a user's preference.

ℨ Std. Dev. = sample standard deviation from cross validation results for each learning model.

SODI Decision Tree for Problem, 'Lymphography'

```
lym_nodes_dimin = 1
|  changes_in_node = lac_central: malign_lymph (23)/ metastases (2)
|  changes_in_node = lac_margin
|  |  (block_of_affere, extravasates) = (no, no)
|  |  |  lymphatics = arched
|  |  |  |  (changes_in_lym, defect_in_node)
|  |  |  |    = {(oval, lac_margin), (round, lacunar)}: malign_lymph (4)
|  |  |  |  (changes_in_lym, defect_in_node)
|  |  |  |    = {(oval, lac_central), (oval, lacunar), (round, lac_margin)}: metastases (5)
|  |  |  |  (changes_in_lym, defect_in_node) = OTHERWISE: N/D (Not Defined)
|  |  |  lymphatics = deformed: metastases (5)
|  |  |  lymphatics = displaced: malign_lymph (1)
|  |  |  lymphatics = normal: N/D (Not Defined)
|  |  (block_of_affere, extravasates) = {(no, yes), (yes, no)} : malign_lymph (25)
|  |  (block_of_affere, extravasates) = (yes, yes)
|  |  |  early_uptake_in = no: metastases (14)
|  |  |  early_uptake_in = yes
|  |  |  |  bl_of_lymph_c = yes: metastases (8)
|  |  |  |  bl_of_lymph_c = no
|  |  |  |  |  no_of_nodes_in = {1, 2}: metastases (8)
|  |  |  |  |  no_of_nodes_in = {3, 4}
|  |  |  |  |  |  changes_in_stru = {grainy}: metastases (2)
|  |  |  |  |  |  changes_in_stru = {diluted, stripped, faint}: malign_lymph (3)
|  |  |  |  |  |  changes_in_stru = OTHERWISE: N/D (Not Defined)
|  |  |  |  |  no_of_nodes_in = OTHERWISE: N/D (Not Defined)
|  changes_in_node = lacunar
|  |  exclusion_of_no = no: metastases (9) / malign_lymph (1)
|  |  exclusion_of_no = yes
|  |  |  special_forms = chalices
|  |  |  |  changes_in_lym = oval: malign_lymph (3)
|  |  |  |  changes_in_lym = round: metastases (2)
|  |  |  |  changes_in_lym = OTHERWISE: N/D (Not Defined)
|  |  |  special_forms = no
|  |  |  |  dislocation_of = no: malign_lymph (1)
|  |  |  |  dislocation_of = yes: metastases (2)
|  |  |  special_forms = vesicles: malign_lymph (18) / metastases (1)
|  changes_in_node = no
|  |  dislocation_of = yes
|  |  |  early_uptake_in = yes: malign_lymph (2)
|  |  |  early_uptake_in = no: metastases (1)
|  |  dislocation_of = no: normal (2)
lym_nodes_dimin = 2
|  (regeneration_of, early_uptake_in) = (yes, yes): N/D (Not Defined)
|  (regeneration_of, early_uptake_in) = (yes, no): fibrosis (1)
|  (regeneration_of, early_uptake_in) = (no, yes): malign_lymph (1)
|  (regeneration_of, early_uptake_in) = (no, no): metastases (1)
lym_nodes_dimin = 3: fibrosis (3)
```

Figure 8. The SODI classification for the problem 'Lymphography'.

C 4.5 Decision Tree: Estimated prediction accuracy 76.5%

Actual Class \ Classified As	normal	metastases	malign_lymph	fibrosis
normal	0	1	1	0
metastases	1	69	11	0
malign_lymph	1	14	46	0
fibrosis	1	1	2	1

PART Decision Rules: Estimated prediction accuracy 88.3%

Actual Class \ Classified As	normal	metastases	malign_lymph	fibrosis
normal	0	1	1	0
metastases	0	71	9	1
malign_lymph	0	10	51	0
fibrosis	0	0	0	4

SODI Decision Tree: Estimated prediction accuracy 92.6%

Actual Class \ Classified As	normal	metastases	malign_lymph	fibrosis
normal	2	0	0	0
metastases	0	75	5	1
malign_lymph	0	4	57	0
fibrosis	0	1	0	3

Figure 9. Confusion matrices for estimated prediction errors of the problem 'Lymphography' by C4.5, PART, and SODI, respectively.

Concluding remarks

We compared SODI to ID3, pruning C4.5, and PART. SODI generated a decision tree with almost better information gain ratio than other methods. ID3 built decision trees with the smallest training errors, but met serious overfitting problems. PART generated the smallest number of decision rules on average. However, with slightly larger sizes of decision trees, SODI can reduce both training errors and estimated prediction errors. For the TDIDT method of SODI, the decision-makings or decision branches are more complex than other methods, but the sizes of trees are effectively reduced.

A user-defined scheme for setting weights of penalties in the generalized risk minimization is highly subject to the user's preference. Table 4 showed an example of schemes of penalty weights. Test problems 1, 2, and 4 have more interest in the training accuracy and the smaller size of decision trees (or concise model description) than any other factors because the given training instances could cover all cases of classification problems. For other problems, since the training instances are random sampling, the prediction accuracy is more important than the training accuracy. As a measurement of overfitting the gap of error between training and testing is also essential factor for model selection. With these considerations, two different schemes of model selection were reviewed as shown in figure 4. For five cases among eight problems SODI generated better solutions. For two cases of problems C4.5 built better solutions than any others, but PART provided a better solution for only one problem.

4. Pruning algorithm of SODI

The goal of this section is to present methods to prune the unnecessary nodes of the resulting decision tree by SODI. The problem is actually of model selection - the model size parameter is the tree size and the trade off is with the accuracy of the tree on the given sample set. The pruning of an internal node is replacing the subtree of the node by a leaf as stopping classification process within the tolerance of training errors. Basically the pruning scheme can be classified as pre-pruning or post-pruning. The pre-pruning method has a stooping criterion when the size of instances at internal node is too small, or the number of

instances that belong to minor class is too small. SODI presented in the previous section is one of pre-pruning learning machine. On the other hand, Post-pruning methods remove one or more subtrees and replace them by a leaf or one branch of that subtree after a decision has been built. The disadvantage of pre-pruning methods is that decision tree growth may be prematurely stopped by its own stopping criterion when the sample size of training instances is relative small.

All the pruning methods considered in this section belong to post-pruning. There are seven pruning methods in general as following (Mansour, 1997; Windeatt and Ardeshir, 2001):

Minimum error pruning

This method was introduced by Niblett and Bratko (1986), and uses Laplace probability estimates to improve the performance of ID3 (Quinlan, 1986) in noisy domains. Cestnik and Bratko (1991) have changed this algorithm by using more general Bayesian approach to estimating probabilities that they called *m-probability estimation*, whose *m* implies the degree of tree pruning. Their suggestion was that the parameter could be adjusted to match properties of learning domain such as noise.

Error-based pruning

Quinlan (1993) developed the error based pruning method for use in C4.5. This pruning method does not need a separate pruning set, but uses an estimate of expected error rate. Examples covered by a leaf from a tree are considered to be a statistical sample from which it is possible to calculate confidence for the posterior probability of misclassification. The assumption is derived from the error in this sample follows a binomial distribution. A default confidence level of 25% is suggested, and the upper limit of confidence is multiplied by the number of cases that are covered by a leaf to determine the number of predicted errors for that leaf.

Critical value pruning

This critical value pruning was proposed by Mingers (1987), and operates with a variety of node selection measures. The idea is to set a threshold, the critical value, which

defines the level at which pruning takes place. An internal node is only pruned if the associated selection measures for the node and all its children do not exceed the critical value.

Cost-complexity pruning

This pruning was developed for CART (Breiman et al., 1984), and produces a sequence of trees by pruning those branches that give lowest increase in error rate per leaf over the training set. In order to select the best tree in sequence, either cross-validation on training set or a separate pruning set was employed. The selected tree is the smallest tree with error rate less than either minimum observed error rate or minimum observed error rate plus one standard error.

Reduce error pruning

The method of reduced error pruning arbitrarily splits the sample set into a training set and a test set at first. The training set is used to build the decision tree. And the test set is used to decide whether a node of the tree should be pruned or not. First, we determine whether it is pruned or not for every internal node of the tree. We set the label of the pruned node to be the majority class among the examples in the training set that reach the pruned node. Then, for each node, the test set is used for computing the number of examples to reach the pruned node and the number of errors caused from the label of the pruned node.

The algorithm basically checks if pruning for each internal node improves the generalization ability of the tree. Specifically, we compare the computed number of errors over the test set of the tree pruned at a pruned node to the number of errors over the test set of all the subtree rooted at the node, which is the sum of the errors of all the leaves of the subtree. If the number of classification errors over the test set of a node as a pruned leaf is significantly smaller than the error of the subtree rooted at the original node, then we choose to prune the node. On the other hand, if the number of classification errors over the test set of a node as a pruned leaf is significantly larger than that of the subtree rooted at the node, then we choose not to prune. Otherwise, i.e., there is no significant difference in the number of errors, we can choose arbitrarily whether it should be pruned or not.

Structural risk minimization

In this approach a dynamic programming is applied for finding for every k the best decision tree T_k (with minimum number of errors) of size k that is a pruned tree of a given decision tree T. Obviously, the number of training errors decreases as k is increased, however, it may meet the change of overfitting problems.

For the testing part, the number of errors will be decreased for a while k is increased, but it becomes increased as k increased further. A model selection policy with structural risk minimization will find the best T_k. We can alternatively use cross validation to test each T_k on an independent test set, and choose among the set of 'best' trees the tree with the least number of errors.

Figure 10 shows the pseudo code of this pruning method with structural risk minimization (Mansour, 1997). T0 and T1 represent the left and right child subtrees of T, respectively. The function of `root`(T) is the root node of the tree T. We also define `makeTree`(`root`(T)*;* T0*;* T1) to be the tree formed by making the subtrees T0 and T1 the left and right child of the root node `root`(T). For every node v of T, `Errors`(v) is the number of classification errors on the sample set of v as a leaf. Now that we have a way to compute for every k the best decision tree T_k of size k, we have to choose one of them as the best hypothesis. It is actually a question of model selection (k being the model size), and we can use the method of structural risk minimization (SRM, or generalized risk minimization as mentioned in chapter 1).

Pessimistic pruning

We now present a pruning method that does not require a separate test set or high time complexity. The idea is to make a single pass from the leaves of the decision tree up towards the root, and at every internal node make a decision whether to prune or not. The decision is based on a comparison of the error at the node (if pruned) to the error of the node subtree.

```
[s,P] = pruning_test(k,T)
   k=number of errors as an INPUT;
   T=the original subtree as an INPUT;
   s=size of a new pruned tree as an OUTPUT;
   P=the pruned tree from the original tree T as an OUTPUT;
Begin
If isLeaf(T)
   If Errors(T) ≤ k
      s = 1;
   Else
      s =infinity;
      P = T;
      return (s, P);
   End If
   If Errors(root(T)) ≤ k
      s = 1
      P = root(T)
      return (s, P)
   End If
   For i = 1 to k
      // Let T0 be the left child (subtree) of T
      // Let T1 be the right child (subtree) of T
      [s^0_i,P^0_i] = pruning_test(i,T0);
      [s^1_i,P^1_i] = pruning_test(k-i,T1);
   End For
   I = arg min_i {s^0_i + s^1_i};
   s = s^0_I + s^1_I + 1;
   // makeTree(...) is a classification algorithm,
   // such as ID3, C4.5, PART, SODI, etc.
   P = makeTree(root(T),P^0_I,P^1_I);
   return (s, P);
Endif
End.
```

Figure 10. Pseudo code of this pruning method with structural risk minimization

Since the true error can only be approximated we make a pessimistic approximation. Let n_v be the number of instances that arrive to a node v, and T_v be the subtree rooted at v. Let e_1 be the number of errors at the subtree n_v, and e_2 be the number of errors if v is replaced by a (pruned) leaf. For processing a pessimistic pruning, one must provide a threshold for pruning criteria between 0 and 1 (usually a small number). We will then decide to prune only

if the estimated error of v (when it is replaced by a leaf) is smaller than the (pessimistically) estimated error of the subtree T_v, i.e.,

$$\frac{e_2}{n_v} < \frac{e_1}{n_v} + \theta . \tag{2.80}$$

Reduced error pruning of SODI

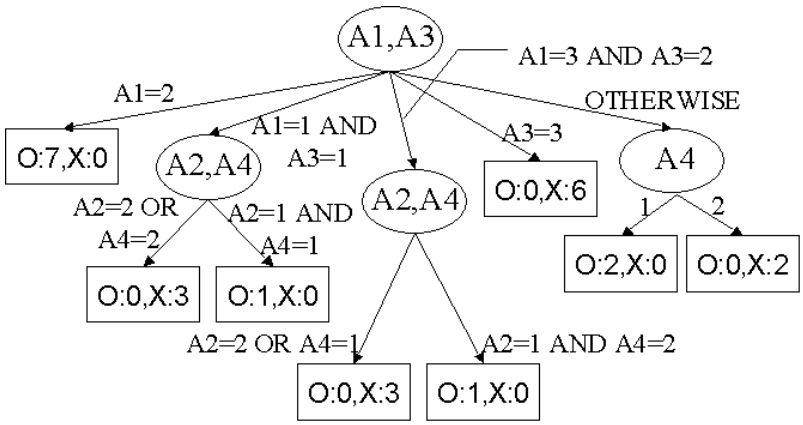
In this thesis we adopted the "reduced error pruning" scheme for SODI to reduce the change of overfitting problems. For processing reduced error pruning, the table 2 for the previous illustrative example should be randomly clustered as a training set and test set. We used this example just for the illustrating of SODI algorithm under the reduced error pruning process with the risk of this pruning method that may derive an anomalous classification model when sample size is extremely small. Therefore, we applied the cross-validation tests with eight different combinations of 22 training data and 3 testing data (total 25 instances)[1]. Figure 11(a) and 11(b) show the results of SODI without reduced error pruning applied (just denoting SODI) and SODI with the pruning applied (denoting pSODI), respectively. The tree size for SODI was 11, and the gain ratio was 0.3704 without having any misclassification. One the other hand, tree size for pSODI is now 5, the split entropy is 1.6329, and the information gain is 0.6179, so that the gain ratio is 0.3784. For the case of this pSODI, there are two misclassifications: classified as 'O' where the original was 'X' at (A1=1, A2=1, A3= 1, A4=1) and classified as 'X' where the original was 'O' at (A1=3, A2=1, A3=2, A4=2).

The notation "(2,*) OR (3,2)" from the internal node (A1,A3) shown in figure 11(b) represents "(A1=2) OR (A1=3 AND A3=2)". The notation "*" means wild card, so that A3 can take any value. Similarly "(1,1) OR (*,3)" from (A1,A3) means "(A1=1 AND A3=1) OR (A3=3)". The notation "O/W" from the internal node (A1,A3) to (A4) represents any other cases except for the previously mentioned two cases (O/W is the abbreviation of OTHERWISE). That is, the values of (A1,A3) for this branch are '(A1=1 AND A3=2) OR (A1=3 AND A3=1)'. The total training errors of pSODI is 8 %, and the average prediction error from cross validation is 1.16 %. Furthermore, the empirical standard deviation of

[1] The combination of the last cross-validation test was 21 training data and 4 testing data.

prediction errors is 2.02 %. On the other hand, the training error of SODI is 0 %, and the average prediction error from cross validation is also 1.16 % (because of very small sample size it was the same as pSODI). The only benefit for a pruned SODI is very easy to understand (see the comparison of those decision tree sizes).

(a)



(b)

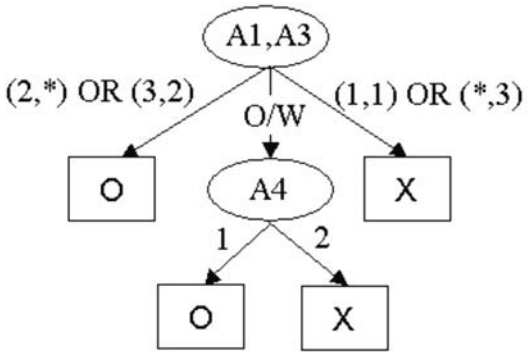


Figure 11. Numerical example for building TDIDT by (a) SODI without pruning and (b) pSODI with pruning

Numerical analysis

In this section we tested the same classification problems presented in the previous section. For evaluating the performance of pruned SODI (pSODI), the following three candidates are selected: C4.5, PART, and (original) SODI. Figure 12(a) shows the relative ratio of the sizes of the decision trees and figure 12(b) shows obtained information gain ratio

for each method. Figure 12(c) shows a comparison of the training errors and figure 12(d) shows a comparison of the prediction errors. The detailed numbers for both of these, as well as the gain ratio for each model and the sample standard deviation of prediction errors computed from cross validation results, are shown in Table 5. Table 6 shows scoring results for user-defined penalty categories defined as same as the previous section.

For the case of problem 1, SODI has the highest accuracy with a slightly larger TDIDT size than any others as shown in Table 5. However, in the viewpoint of overfitting problems, SODI may be not favorable (when we assumed the collected instances only random samples). The final solution of pSODI for problem 1 was concluded as the same as C4.5. For the case of problem 2, SODI and pSODI have better performance than other two methods as shown in Table 6. For the case of problem 3 (Breast Cancer), C4.5 and pSODI has almost same performance, but pSODI has slightly smaller description of decision tree.

For the viewpoint of estimated prediction accuracy, all methods performed almost similar as shown in figure 12(c). However, as shown in figure 12(d), the pruned SODI (pSODI) had the smallest error gaps between training and testing results. It was the reason that this pruned SODI processed not only pre-pruning but also post-pruning. Also, this pruned SODI created the smallest decision trees almost for every problem as shown in figure 12(a). Therefore, this pruned SODI is in general competitively acceptable for model selection.

For the classification of both ‘Balance Scale’ and ‘Lymphography’, the pruned SODI performed extraordinary compared to any others. It may be caused from the high association between their attributes, so that bivariate decision-makings of SODI can describe unknown real decision boundary more precisely.

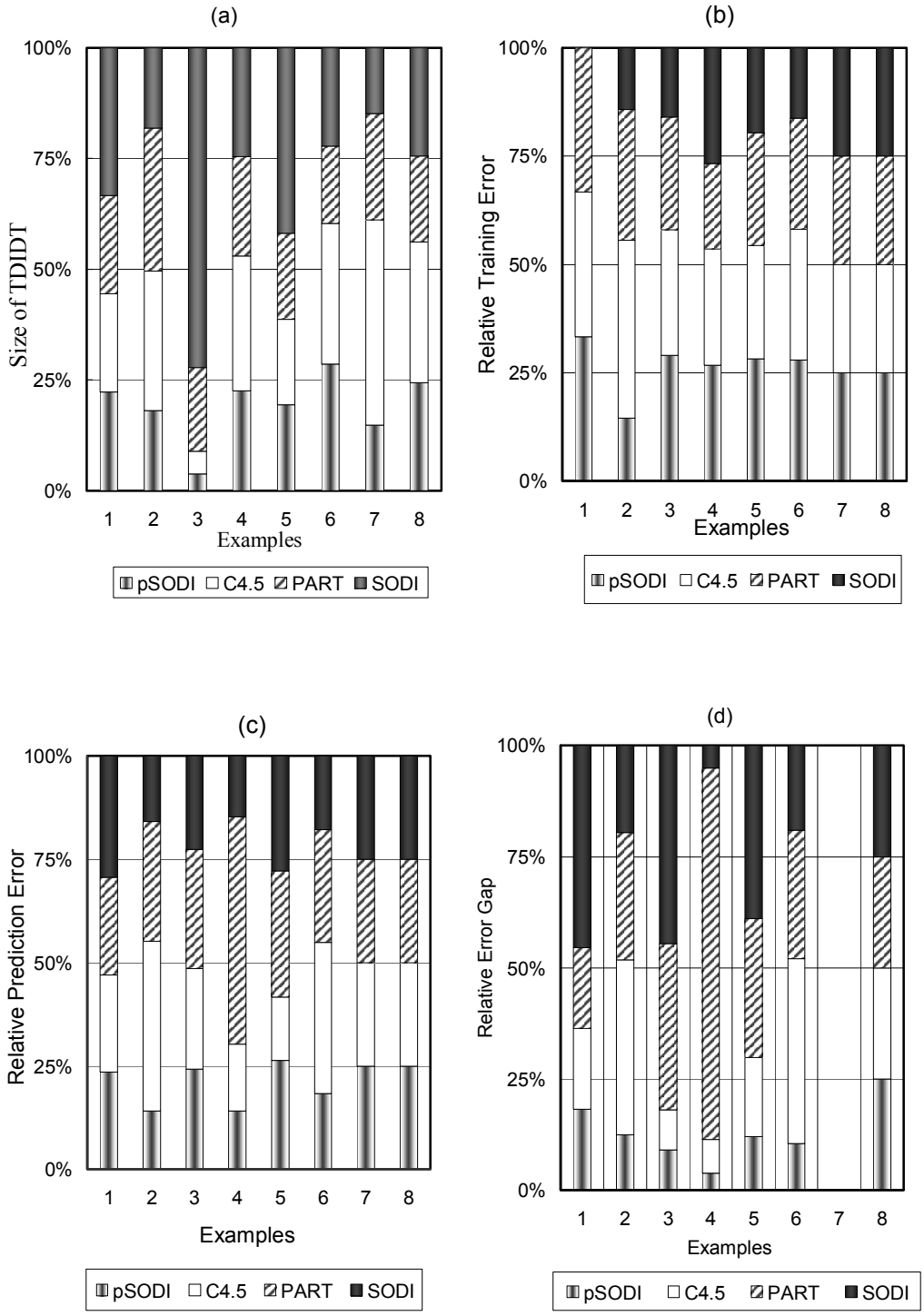


Figure 12. Performance evaluation of pruned SODI comparing with C4.5/PART/SODI without pruning: (a) the proportion of number of decision rules, (b) training error, and (c) estimated prediction error, and (d) the gap between training error and prediction error. The smaller proportion is the better.

Table 5. Comparison of pruned SODI with other univariate TDIDT algorithms

Problem Set	Method	Gain Ratio	N(leaves) / sizeOf(DT)	(a) Training Error (%)	(b) Prediction Error (%)	Gap (%) |(a)-(b)|	Std. Dev.[3] of Prediction Err
Contact Lenses	C4.5	0.6012	4/7	8.33	16.67	8.33	6.60
	PART	0.6012	4 rules	8.33	16.67	8.33	6.60
	SODI	0.6557	6/11	0.00	20.83	20.83	7.89
	pSODI	0.6012	4/7	8.33	16.67	8.33	6.60
Balance Scale	C4.5	0.0813	33/41	24.96	33.33	8.37	12.22
	PART	0.1190	34 rules	18.40	23.47	6.07	8.38
	SODI	0.2741	19/23	8.64	12.80	4.16	4.57
	pSODI	0.2741	19/23	8.64	12.80	4.16	4.57
Breast Cancer	C4.5	0.0907	4/6	24.13	26.53	2.40	9.86
	PART	0.0616	15 rules	21.68	31.63	9.95	11.89
	SODI	0.0919	57/80	13.29	24.67	11.83	8.25
	pSODI	0.0911	3/4	24.13	26.53	2.40	9.86
Chess End-Game	C4.5	0.2880	31/59	0.34	0.46	0.12	0.15
	PART	0.2962	23 rules	0.25	1.56	1.31	0.58
	SODI	0.2943	25/47	0.34	0.42	0.08	0.14
	pSODI	0.3014	23/40	0.34	0.40	0.06	0.14
1984 USA Voting	C4.5	0.5854	6/11	2.76	2.03	0.73	0.76
	PART	0.4266	6 rules	2.76	4.05	1.29	1.43
	SODI	0.5608	13/21	2.07	3.68	1.61	1.26
	pSODI	0.5884	6/9	2.99	3.49	0.50	1.24
Lymphography	C4.5	0.2871	20/30	8.78	23.53	14.75	7.74
	PART	0.3420	11 rules	7.43	17.65	10.22	6.30
	SODI	0.3184	29/46	2.03	7.43	5.40	2.48
	pSODI	0.3243	18/28	8.11	8.83	0.72	2.88
Mushroom	C4.5	0.3090	25/30	0.00	0.00	0.00	0.0
	PART	0.4058	13 rules	0.00	0.00	0.00	0.0
	SODI	0.5607	8/11	0.00	0.00	0.00	0.0
	pSODI	0.5607	8/11	0.00	0.00	0.00	0.0
Classifying Zoo	C4.5	0.9342	13/21	0.99	2.86	1.87	0.10
	PART	0.9587	8 rules	0.99	2.86	1.87	0.10
	SODI	0.9342	10/15	0.99	2.86	1.87	0.10
	pSODI	0.9342	10/15	0.99	2.86	1.87	0.10

3 Std. Dev. = sample standard deviation from cross validation results for each learning model.

Table 6. Comparison of pruned SODI with other univariate TDIDT algorithms (scoring§ results from Table 5)

Problem Set	Method	N(leaves) / SizeOf(DT)	(a) Train. Err.	(b) Pred. Err.	Gap betw/ |(a)-(b)|	Std. Dev.ℨ of Pred. Err.	Average Score
Contact Lenses	C4.5	5	3	-	-	-	4.00
	PART	5	3	-	-	-	4.00
	SODI	4	5	-	-	-	**4.50**
	pSODI	5	3	-	-	-	4.00
Balance Scale	C4.5	3	3	-	-	-	3.00
	PART	3	4	-	-	-	3.50
	SODI	5	5	-	-	-	**5.00**
	pSODI	5	5	-	-	-	**5.00**
Breast Cancer	C4.5	5	-	4	5	4	4.50
	PART	4	-	3	3	3	3.25
	SODI	3	-	5	3	5	4.00
	pSODI	5	-	4	5	4	**4.50**
Chess End-Game	C4.5	3	3	-	-	-	3.00
	PART	5	5	-	-	-	**5.00**
	SODI	4	3	-	-	-	3.50
	pSODI	5	3	-	-	-	4.00
1984 USA Voting	**C4.5**	5	-	5	5	5	**5.00**
	PART	5	-	3	3	3	3.50
	SODI	3	-	4	3	4	3.50
	pSODI	5	-	4	5	4	4.50
Lymphography	C4.5	4	-	2	2	2	2.25
	PART	5	-	3	3	3	3.50
	SODI	3	-	5	4	5	4.25
	pSODI	4	-	5	5	5	**4.75**
Mushroom	C4.5	3	-	5	5	5	4.50
	PART	4	-	5	5	5	4.75
	SODI	5	-	5	5	5	**5.00**
	pSODI	5	-	5	5	5	**5.00**
Classifying Zoo	C4.5	3	-	5	5	5	4.50
	PART	5	-	5	5	5	**5.00**
	SODI	4	-	5	5	5	4.75
	pSODI	4	-	5	5	5	4.75

§Every criterion has been assigned to the same weight for each problem. The policy of both selecting criteria and making their scores for each problem is subject to a user's preference.

ℨ Std. Dev. = sample standard deviation from cross validation results for each learning model.

Concluding remarks

We compared pruned SODI to C4.5, PART, and SODI without pruning. The pruned SODI generated a decision tree with almost better information gain ratio than other methods. The pruned SODI generated the smallest size of decision trees on average as well as the smallest error gap between training and testing (or cross validation). It implies this pruning SODI has more capable for removing overfitting problems in many cases.

For each problem different schemes of model selection was reviewed. For four cases among eight problems the pruning SODI (pSODI) generated better solutions with respect to the model selection by user-defined generalized risk minimization. For two cases of problems PART built better solutions than any others, but C4.5 provided a better solution for only one problem (at the breast cancer problem, the scores of C4.5 and pruning SODI are same, but the size of decision tree form the SODI is slightly smaller than C4.5).

5. Summary

At the first section of this chapter the information entropy (or Shannon's entropy; see appendix B for more detailed properties of it) in the information theory has been introduced. Also, the concept of mutual information and its properties are described. With these properties the technique of eliminating redundant nominal attributes has been verified at the second section. This can be extended for feature selection and feature cleaning. For examples, suppose three attributes are linearly dependent on each other, and there is only one is redundant. Then, one can compute the gain ratio for each attribute. From this information, the redundant attribute that has the smallest gain ratio should be removed for further data mining. Also, it can be applied for our SODI. Since SODI requires computing all pairs of two attributes at the worst case, removing redundant attribute will promise more efficient computation.

At the third section, SODI for a new TDIDT algorithm has been introduced for any classification problems with nominal attributes only. In general SODI performed better quality of classification results than other univariate TDIDT methods. Without pruning

process of SODI it may meet some overfitting problems. Therefore, at the next section, a pruning SODI has been introduced and evaluated. With pruning process of SODI, it generated the smallest decision trees for almost every problem, as well as provided the stable prediction accuracy.

Now, we want to consider the classification with numerical attributes. The classification problems in the data mining are widely studied. Especially it is tremendous research works for the classification with numerical attributes without assuming any distribution of numerical attributes. The support vector machines are one of the most famous research fields in this category of problems. The next chapter will discuss this in details.

CHAPTER 3. SVMM: SUPPORT VECTOR MACHINES FOR MULTI-CATEGORY

In this chapter we consider a classification problem with numerical attributes only. The policy of defining decision boundaries for a classification is the most essential to achieve both training accuracy and prediction accuracy. The shapes of decision boundaries are also highly related to the model complexity. For C4.5 (Quinlan, 1993) the decision boundaries from univariate attributes are formed as a set of orthogonal partitions for each decision attribute axis. For an oblique decision tree (Murthy et al., 1994a), the decision boundaries form still linear hyperplanes constructed by multiple attributes. Therefore, the oblique decision tree can reduce in general more overfitting problems than C4.5. However, it has still a serious problem: the decision boundaries from this oblique TDIDT are parallel to each other. If the decision boundaries are not parallel, even nonlinear, we need more systematic techniques for satisfying classification quality. In this case, support vector machines (SVM; Bennett, 1994-1997) are more suitable by building piecewise-linear or nonlinear decision boundaries. It is obviously trade-off relationship between the model complexity and the model accuracy (of not only training but also prediction). SVM have less possibility of overfitting problems than others, but the model description is more complicated than others. In this chapter we focus on the use of SVM for how to describe piecewise linear decision boundaries for multiple classes, and for how to apply for TDIDT to improve both prediction accuracy and its stability.

SVM is one of best tools for the self-constructive classification problems in machine learning and data mining. SVM yields an extremely fast result due to its simple algorithm for generating a linear or nonlinear classifier that merely requires the solution of a single instance. SVM have been applied for wide fields of studies (Burges, 1998), such as isolated handwritten digit recognition (Burges and Vapnik, 1996), object recognition (Blanz et al., 1996), speaker identification (Schmidt, 1996), text categorization (Joachims, 1997), time series prediction tests for regression (Müller et al., 1997), the Boston housing problem for a regression estimation (Drucker et al., 1997), and so on.

Now suppose that a machine learns to the mapping $\mathbf{x}_i \mapsto y_i$, or $\mathbf{x} \mapsto f(\mathbf{x}, \alpha)$, where the functions $f(\mathbf{x}, \alpha)$ are unrevealed knowledge functions by the adjustable parameter α. Suppose the machine must be deterministic such that, for a given input $\mathbf{x}$ and the choice of α, it will always provide the same output $f(\mathbf{x}, \alpha)$. Then, we can describe an objective function for a classification problem with the expectation of test errors for a trained SVM as fallows

$$R(\alpha) = \int \frac{1}{2} |y - f(\mathbf{x}, \alpha)| p(\mathbf{x}, y) d\mathbf{x} dy \,, \tag{3.1}$$

where $p(\mathbf{x}, y)$ represents the probability density at or the priori knowledge of an example $\{\mathbf{x}, y\}$. The quantity $R(\alpha)$ is called the expected risk (Burges, 1998). In general, since $p(\mathbf{x}, y)$ is unknown, we need to estimate an upper bound of this expected risk. The *empirical risk* (Burges, 1998) is defined to be the sample mean error rate on the finite size (n) of a training set as follows:

$$R_{emp}(\alpha) = \frac{1}{2n} \sum_{i=1}^{n} |y_i - f(\mathbf{x}_i, \alpha)| \,. \tag{3.2}$$

Now take any significant level ρ such that $0 \le \rho \le 1$. Vapnik (1995) showed that the expected risk has be upper bound with (1-ρ) probability as follows:

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\left(\frac{h + h \log(2n/h) - \log(\rho/4)}{n} \right)}, \tag{3.3}$$

where h is a nonnegative integer so-called as the "Vapnik-Chervonekis (VC) dimension", which is a measure of the notion of capacity (or, the ability of the machine to learn any training set without error). The second term in equation (3.3) is called the "VC confidence". There are significantly meaningful information with this upper bound of $R(\alpha)$: (1) it is unnecessary to assume any priori probability function, and (2) we can easily obtain the upper bound of the expected risk when the value h is determined by the confidential limit (1-ρ). Therefore, for given several different learning machines $f(\mathbf{x}, \alpha)$ and a fixed confidential limit (1-ρ), we can easily find the lowest upper bound on the actual risk by taking the machine that minimizes the right-handed side of equation (3.3).

1. Introduction to support vector machines (SVM)

Linearly separable SVM

Suppose that there are only two classes such that a learning machine $\mathbf{x} \mapsto f(\mathbf{x}, \alpha)$ can be established by a give training data $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i \in \mathbf{R}^d$ and $y_i \in \{-1,+1\}$ for i=1,2,…, n with d attributes of $\boldsymbol{x}_i$. Then, there exists a hyperplane to separate the positive class examples from the negative ones because this problem is linearly separable. The point $\mathbf{x}$, which lie on the hyperplane, satisfy $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is a normal vector to the hyperplane (the thick centerline in the figure) as shown in Figurer 13. Let all training data satisfy the following conditions:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \quad \text{for } y_i = +1, \text{ some } i \in \{1,2,...,n\}, \tag{3.4}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \text{for } y_i = -1, \text{ some } i \in \{1,2,...,n\}. \tag{3.5}$$

Consequently

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \text{for all } i = 1,2,...,n. \tag{3.6}$$

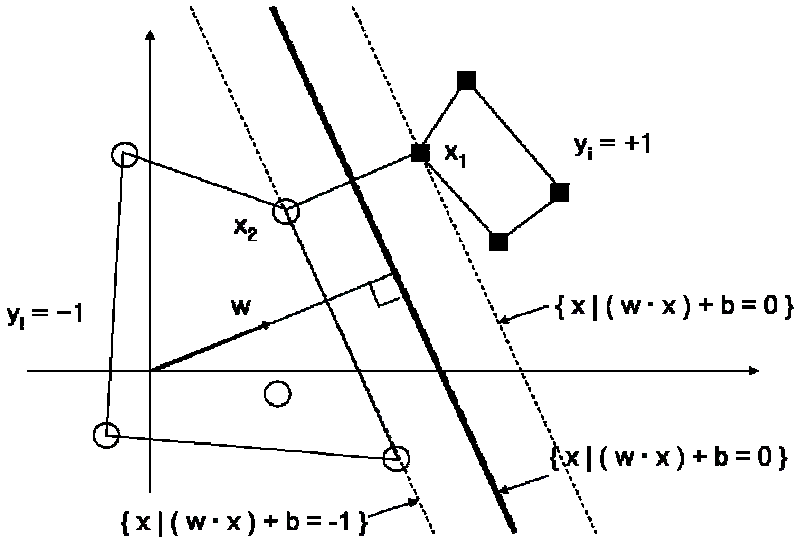


Figure 13. Example of a simple linear SVM from separable training data[2]: The white boxes and black circles indicate two different class data. Especially white circle and black box, which lie on two separable marginal lines, are called the "support vectors". The thick centerline is the actual decision boundary for this classification (Smola et al., 1999).

[2] This figure was provided by Smola et al. (1999).

Now consider the points which lie on the hyperplane $H_1 : \mathbf{w} \cdot \mathbf{x}_i + b = +1$ with a normal vector **w**. Then, the normal distance from the origin is $|1-b|/\|\mathbf{w}\|$. Similarly, the points for which the equality in (3.5) holds with the same normal vector **w**, depart as much as $|-1-b|/\|\mathbf{w}\|$ from the origin. So, the normal distance between these two hyperplane is $2/\|\mathbf{w}\|$. Therefore, the objective for this problem is to find the pair of hyperplanes such that they make the maximum margin of $2/\|\mathbf{w}\|$. It is then equivalent to minimize $\|\mathbf{w}\|$ subject to the constraints (3.6). That is, the primal quadratic problem can be established as follows:

$$\text{minimize} \frac{\|\mathbf{w}\|^2}{2} \text{ subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq +1 \quad \text{for all } i \in \{1,2,...,n\}. \tag{3.7}$$

Now consider a Lagrange formulation of the primal problem (3.7). Let positive α_i ($i = 1,2,...l$) be Lagrange multipliers for each inequality constraint in (3.6). Then, the relaxed Lagrange function is formulated as

$$L_P = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{n} \alpha_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) + \sum_{i=1}^{n} \alpha_i. \tag{3.8}$$

The Karush-Kuhn-Tucker (KKT) conditions for this Lagrange function is as follows (Smola et al., 1999):

$$\sum_{i=1}^{n} \alpha_i y_i x_{ij} = w_j \quad \text{for all } j = 1,2,...d. \tag{3.9}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0. \tag{3.10}$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \text{for all } i = 1,2,...,n. \tag{3.11}$$

$$\alpha_i \geq 0 \quad \text{for all } i = 1,2,...,n. \tag{3.12}$$

$$\alpha_i \{y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\} = 0 \quad \text{for all } i = 1,2,...,n. \tag{3.13}$$

Especially the equation (3.13) is called the *KKT complementary slackness*. If α_i is not zero, then the sample point $\mathbf{x}_i$ must satisfy the equality in equation (3.6). Thus, this point $\mathbf{x}_i$ becomes a support vector. Since the primal problem in (3.7) has a quadratic convex function with a convex feasible region which is constructed by linear constraints, the above KKT

conditions are necessary and sufficient for **w**, b, and α to be an optimal solution (Fletcher, 1987). Therefore, solving the SVM primal problem is equivalent to finding the KKT conditions. While **w** is explicitly determined by the training data as shown in (3.9), b can be easily computed by using the *KKT complementary slackness* such as $b = y_i - \mathbf{w}_i \cdot \mathbf{x}_i$, where α_i is not zero. By substituting equations (3.9) and (3.10) into (3.8) the primal SVM problem is equivalent to the following dual SVM problem:

$$\text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} (y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j) \alpha_i \alpha_j \tag{3.14}$$

subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1,2,...,n. \tag{3.15}$$

This dual problem becomes now the standard quadratic programming problem with unknown decision variables, α_i $(i = 1,2,...,n)$. Since the algorithm for a quadratic programming is out of our interest, we omit the introduction for how to solve this problem. After finding the optimal solution of α_i^* $(i = 1,2,...,n)$, the hyperplane decision function $d(\mathbf{x})$ can be written as

$$d(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i^* (\mathbf{x} \cdot \mathbf{x}_i) + b \right). \tag{3.16}$$

Not separable training data by a linear SVM

Suppose that some training data cannot be separable by the above linear SVM. Then, we need to allow for some tolerances or relaxed constraints for these data. Let tolerance variables (or, positive slack variables) for relaxing the equations (3.4) and (3.5) in order to allow some limited level of misclassifications (Cortes and Vapnik, 1995) as follows:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 - \varepsilon_i \quad \text{for } y_i = +1, \text{ some } i \in \{1,2,...,n\}, \tag{3.17}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 + \varepsilon_i \quad \text{for } y_i = -1, \text{ some } i \in \{1,2,...,n\}. \tag{3.18}$$

Consequently

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \varepsilon_i \quad \text{for all } i = 1,2,...,n, \tag{3.18}$$

where

$$\varepsilon_i \geq 0 \quad \text{for all } i = 1,2,...,n. \tag{3.19}$$

Here, $\sum_i \varepsilon_i$ represents an upper bound on the number of training errors. Assigning a large amount of penalty cost C as the upper bound of training errors $\sum_i \varepsilon_i$, we can construct a primal quadratic problem, whose objective is to minimize a generalized (or penalized) risk function, as follows:

$$\text{(P') minimize} \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{n} \varepsilon_i \tag{3.20}$$

subject to

$$\begin{aligned} & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \varepsilon_i \quad \text{for all } i = 1,2,...,n. \\ & \varepsilon_i \geq 0 \quad \text{for all } i = 1,2,...,n. \end{aligned} \tag{3.21}$$

Now consider a Lagrange formulation of the above primal problem. Let positive α_i and μ_i $(i = 1,2,...n)$ be Lagrange multipliers for each inequality constraint in both (3.20) and (3.21), so that a relaxed Lagrange function can be formulated as

$$L_P = \frac{\|\mathbf{w}\|^2}{2} C\sum_{i=1}^{n} \varepsilon_i - \sum_{i=1}^{n} \alpha_i y_i(\mathbf{w}\cdot\mathbf{x}_i + b) + \sum_{i=1}^{n} \alpha_i(1-\varepsilon_i) - \sum_{i=1}^{n} \alpha_i \mu_i. \tag{3.22}$$

The Karush-Kuhn-Tucker (KKT) conditions for this Lagrange function is then as follows (Smola et al., 1999):

$$\sum_{i=1}^{n} \alpha_i y_i x_{ij} = w_j \quad \text{for all } j = 1,2,...d\,. \tag{3.23}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0\,. \tag{3.24}$$

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \varepsilon_i \quad \text{for all } i = 1,2,...,n\,. \tag{3.25}$$

$$\alpha_i + \mu_i = C \quad \text{for all } i = 1,2,...,n\,. \tag{3.26}$$

$$\varepsilon_i \geq 0, \alpha_i \geq 0, \mu_i \geq 0 \quad \text{for all } i = 1,2,...,n\,. \tag{3.27}$$

$$\alpha_i\{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \varepsilon_i\} = 0 \quad \text{for all } i = 1,2,...,n\,. \tag{3.28}$$

$$\varepsilon_i \mu_i = 0 \quad \text{for all } i = 1,2,...,n \tag{3.29}$$

The equations (3.28) and (3.29) are called the *KKT complementary slackness*. Note that equation (3.26) combined with (3.29) shows $\varepsilon_i = 0$ if $\alpha_i < C$. The above KKT conditions

are necessary and sufficient for **w**, b, ε, α, and μ to be an optimal solution (Fletcher, 1987). Therefore, solving the SVM primal problem is equivalent to finding the KKT conditions. By substituting (3.23) and (3.24) into (3.22) the primal SVM problem (P') is equivalent to the following dual problem (D'):

$$\text{(D')}\ \text{maximize} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} (y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j) \alpha_i \alpha_j \tag{3.30}$$

subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C, \quad i = 1,2,...,n\,. \tag{3.31}$$

This dual problem becomes now a standard quadratic programming problem with unknown decision variables, α_i $(i = 1,2,...,n)$ with the upper bound C (a user-defined penalty cost for misclassification).

2. Extension of SVM for classification problems with 3 or more classes

The extension from the two-class problem to K (> 2) classes is an important question for the support vector machine research. There are several approaches to extend, but in this thesis, we only introduce three ways of extension: 'one class versus all' method, pairwise classification approach (Kreβel, 1997), and mathematical programming methods (Bennett, 1994-1997).

'One class versus all'

Here, the binary decision functions for K classes with d attributes can be written as

$$f_k : \mathbf{R}^d \to \{\pm 1\} \quad \begin{cases} +1 & \text{far all sampls in class k} \\ -1 & \text{otherwise.} \end{cases} \tag{3.32}$$

For a two-dimensional problem with three classes (notated by '+', '*', and '×') the resulting three boundaries of linear support vector machines are shown in figure 14(a). In the middle of the pictures in figure 14(a) all decision functions result in (-1) since this area does not belong to any class. Other three triangle areas have the same situation. Therefore, the tie-breaking rule is necessary to determine the classification for this area: The classification

result can be determined by a discriminator$\in \mathbf{R}^K$, where the index of the largest component is chosen as class decision. This approach is called 'winner-takes-all' method in figure 14(b).

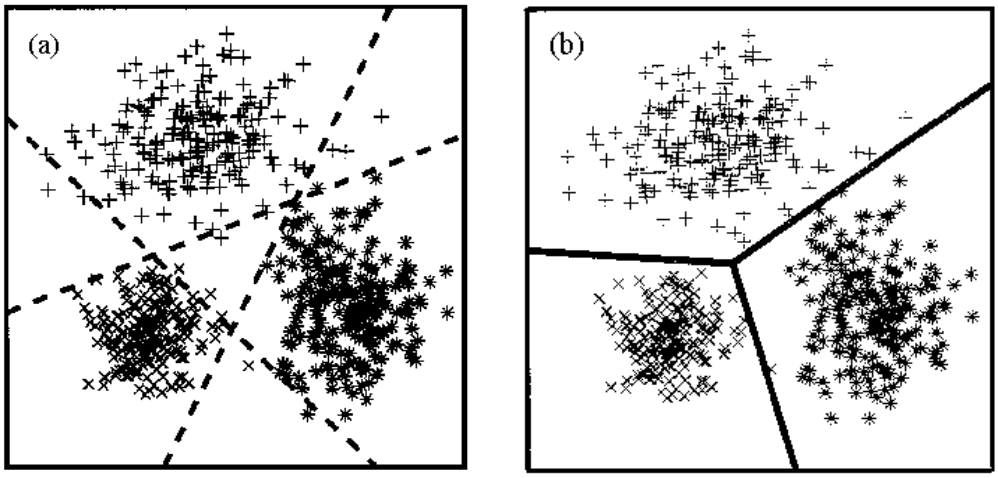


Figure 14. Three class example for 'one class versus all'[3]: (a) resulting three decision boundaries and (b) tie-breaking by 'winner-takes-all' from three decision boundaries. At the final status, a training sample for class (×) has been misclassified by the final decision boundaries.

Pairwise classification

For the pairwise classification a decision function f_{kl} is introduced for each pair (k, l) of classes. Since the pairwise approach is symmetric, $f_{kl} = -f_{lk}$ as follows:

$$f_{kl} : \mathbf{R}^d \rightarrow \{\pm 1\} \quad \begin{cases} +1 & \text{far all samples in class } k \\ -1 & \text{far all samples in class } l \end{cases} \tag{3.33}$$

where d is the number of attributes. In this approach it is required to compute $K(K\text{-}1)/2$ times SVM calculations as shown in figure 15(a) . The decision can be derived by summing up the according pairwise decision functions: $f_k = \sum_l f_{kl}$. If there are no ties, the maximum value of f_k is (K-1). That is, the winner class can get exactly (K-1) positive votes as shown in figure 15(b). Each point in the tie region was assigned to the 'closest' class, using the real input to the decision function in (3.33). For example in figure 15(a), there are 9 line segments from this algorithm: three (+1) votes, 3 (-1) votes, and three (-3) votes as shown in the small

[3] This figure was provided by Kreβel (1997) in the book "Advances in Kernel Methods", edited by Schölkopf et al. (2001).

triangle at the middle region. Therefore, in this classification problem, only three positive decision boundaries will be remained as shown in figure 15(b). This algorithm provides more spacious margins in the near of borders than the 'one class versus all' method. However, it requires (K-1)/2 times of calculations rather than the other.

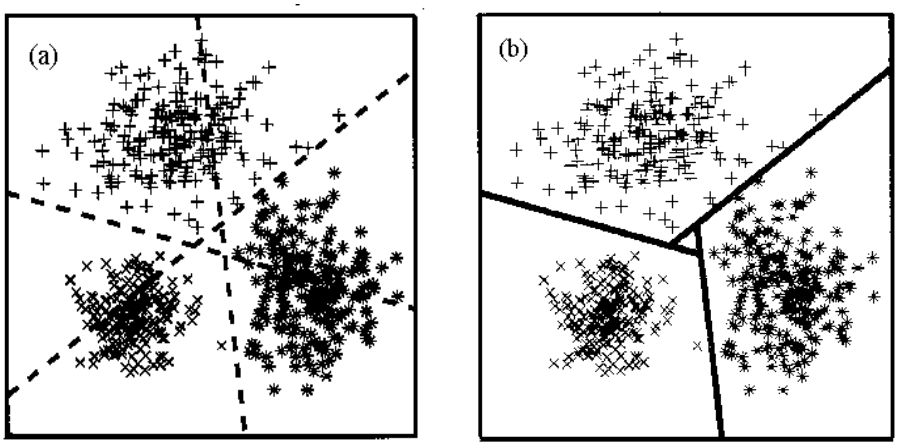


Figure 15. Three class example for pairwise classification[4]: (a) resulting three decision boundaries, and (b) the tie-breaking rules does not required for this problem. At the middle triangle area, the classification was remained as unclassified status.

Mathematical programming

Let N be the number of attributes. Let $\mathbf{A}^j$ be a set of training data that belong to class j (j=1,2,…,K), and m_j be the cardinality of $\mathbf{A}^j$, i.e., the number of training data belong to $\mathbf{A}^j$. Then, the dimension of is $m_j \times N$. The i^{th} row of $\mathbf{A}^j$ (i^{th} training data) is denoted $\mathbf{A}_i^j$. Let e denote a vector of ones of the appropriate dimension. Then, we describe a set of constraints such as $\mathbf{w} \cdot \mathbf{A}_i^j \geq \gamma + 1, \; i = 1,2,..., m_i$ as $\mathbf{A}^j \mathbf{w} \geq (\gamma + 1)\mathbf{e}$ (here, $\gamma = -b$ in equation (3.6)).

Bennett and Mangasarian (1992) illustrated how to minimize the average magnitude of the misclassification errors in the construction of the following robust linear programming (RLP) for two class problems:

[4] This figure was provided by Kreβel (1997) in the book "Advances in Kernel Methods", edited by Schölkopf et al. (2001).

$$\begin{aligned} \text{minimize} \quad & \lambda\|\mathbf{w}\|_1 + (1-\lambda)\left(\frac{\mathbf{e}^T\mathbf{y}_1}{m_1} + \frac{\mathbf{e}^T\mathbf{y}_2}{m_1}\right) \\ \text{subject to} \quad & \mathbf{A}^1\mathbf{w} - (\gamma+1)\mathbf{e} + \mathbf{y}_1 \geq 0, \\ & -\mathbf{A}^2\mathbf{w} + (\gamma+1)\mathbf{e} + \mathbf{y}_2 \geq 0, \\ & \mathbf{y}_1 \geq 0, \quad \mathbf{y}_2 \geq 0, \end{aligned} \tag{3.34}$$

where m_1 and m_2 are the cardinality of $\mathbf{A}^1$ and $\mathbf{A}^2$ respectively, and $\lambda \in (0, 1)$ is related to a misclassification cost from the objective function. Here $\|\mathbf{w}\|_1$ is defined the 1-norm as

$$\|\mathbf{w}\|_1 = \sum_{j=1}^{N} |w_j|. \tag{3.35}$$

From (3.34) $\mathbf{y}_1$ and $\mathbf{y}_2$ represent the distances between the current decision borderline and misclassified training data, which belong to $\mathbf{A}^1$ and $\mathbf{A}^2$ respectively.

The advantage of RLP model over SVM is a linear programming. RLP is transformed to a linear programming of SVM, so that it spent much less computational cost (Bennett and Mangasarian, 1992). RLP minimizes both the average distance of the misclassified points from the relaxed supporting planes and the maximum number of classification errors at the same time. By introducing the variable $\mathbf{s}$ such that $|\mathbf{w}| \leq \mathbf{s}$ from (3.35), the RLP in (3.34) can be rewritten as (Bredensteiner and Bennett, 1999)

$$\begin{aligned} \text{minimize} \quad & \lambda\mathbf{e}^T\mathbf{s} + (1-\lambda)\left(\frac{\mathbf{e}^T\mathbf{y}_1}{m_1} + \frac{\mathbf{e}^T\mathbf{y}_2}{m_1}\right) \\ \text{subject to} \quad & \mathbf{A}^1\mathbf{w} - \gamma\mathbf{e} + \mathbf{y}_1 \geq 0, \\ & -\mathbf{A}^2\mathbf{w} + \gamma\mathbf{e} + \mathbf{y}_2 \geq 0, \\ & -\mathbf{s} \leq \mathbf{w} \leq \mathbf{s}, \\ & \mathbf{y}_1 \geq 0, \quad \mathbf{y}_2 \geq 0, \quad \mathbf{s} \geq 0. \end{aligned} \tag{3.36}$$

It is more computationally efficient to solve the dual RLP problems as follows

$$\begin{aligned} &\text{minimize} \quad \mathbf{e}^T\mathbf{u} + \mathbf{e}^T\mathbf{v} \\ &\text{subject to} \quad -\lambda\mathbf{e} \le \mathbf{u}^T\mathbf{A}^1 - \mathbf{v}^T\mathbf{A}^1 \le \lambda\mathbf{e}, \\ &\qquad \mathbf{e}^T\mathbf{u} - \mathbf{e}^T\mathbf{v} = 0, \\ &\qquad 0 \le \mathbf{u} \le (1-\lambda)/m_1, \\ &\qquad 0 \le \mathbf{v} \le (1-\lambda)/m_2, \end{aligned} \tag{3.37}$$

where **u** and **v** are the vector of dual variables of the first and the second constraints in (3.36), respectively.

In multi-category classification a piecewise-linear separator is used to discriminate between K (> 2) classes of m^i (i=1,2,…,K) training data. A discriminate function to separate one class from the remaining (K-1) classes can be constructed as

$$\mathbf{A}^i\mathbf{w}^i - \gamma^i\mathbf{e} > \mathbf{A}^i\mathbf{w}^j - \gamma^j\mathbf{e}, \quad i, j = 1,2,..., K,\ j \ne i. \tag{3.38}$$

To classify a new instance **x**, compute $f_i(\mathbf{x}) = \mathbf{x}^T\mathbf{w}^i - \gamma^i$ for i=1,2,…,K. If there exists only one instance of $f_i(\mathbf{x}) > 0$ for any index of i, then clearly the instance belongs to the class i. If there are multiple instances of $f_i(\mathbf{x}) > 0$ or every instance belongs to $f_i(\mathbf{x}) \le 0$ for i=1,2,…,K, then the class identification of such an instance is ambiguous. Therefore, the general rule is that the class of an instance $\boldsymbol{x}$ is determined from $(\mathbf{w}^i, \gamma^i)$, i=1,2,…,K, by finding i such that

$$f_i(\mathbf{x}) = \mathbf{x}^T\mathbf{w}^i - \gamma^i \tag{3.39}$$

is maximized (Bredensteiner and Bennett, 1999). The inequality in (3.38) can be used as a definition of piecewise-linear separability (Bredensteiner and Bennett, 1999). Figure 16 shows an example for piecewise-linearly separable problem with three classes.

Note that RLP in (3.36) can be extended to a multi-category classification model by constructing K two-class discriminants depending on the 1-norm desired for margin control. This method is denoted as k-RLP (Bredensteiner and Bennett, 1999). To obtain more accurate classification rules from the training set from the decision function (3.39), the following inequalities must be satisfied as follows:

$$\mathbf{A}^i(\mathbf{w}^i - \mathbf{w}^j) - (\gamma^i - \gamma^j)\mathbf{e} \ge \mathbf{e}, \quad i, j = 1,2,..., K,\ j \ne i. \tag{3.40}$$

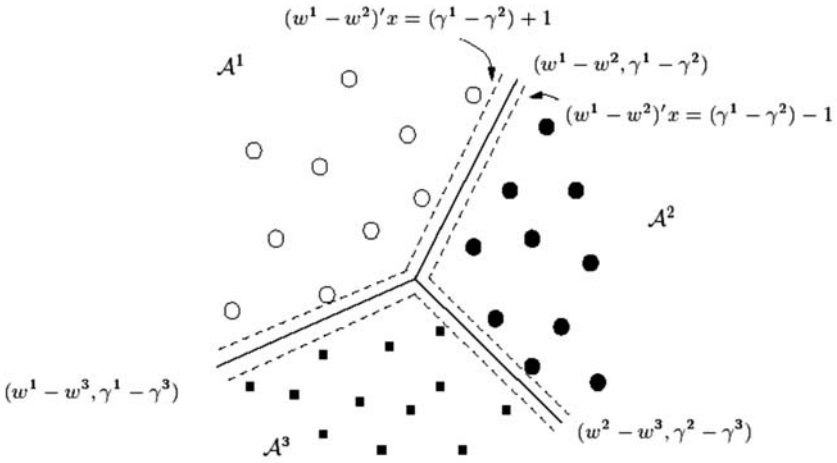


Figure 16. Example for a piecewise-linearly separable problem with three classes[5]: The dashed lines represent the margins for each piecewise-linear separating decision borderline.

The M-RLP[6] method (Bennett and Mangasarian, 1993, 1994) was introduced to find $(\mathbf{w}^i, \gamma^i)$, i=1,2,…,K, satisfying the inequality (3.40). In the two-class case, M-RLP becomes the original RLP in (3.36). The M-RLP was established as follows

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{K} \sum_{\substack{j=1 \\ j \neq i}}^{K} \frac{\mathbf{e}^T \mathbf{z}}{m^i} \\ &\text{subject to } \mathbf{z}^{ij} \geq -\mathbf{A}^i(\mathbf{w}^i - \mathbf{w}^j) + (\gamma^i - \gamma^j)\mathbf{e} + \mathbf{e}, \\ &\mathbf{z}^{ij} \geq 0, \quad i, j = 1,2,...,K,\ j \neq i, \end{aligned} \tag{3.41}$$

where $\mathbf{z}^{ij} \in \mathbf{R}^{m^i}$, and m^i is the cardinality of a training set $\mathbf{A}^i$ for class i. The computational efforts for this M-RLP are very inexpensive because it is also linear. However, there is no constraint to maximize the margin between two classes in this M-RLP, so that it may result in very narrow margin at the optimality. In M-RLP (3.41), if the optimal objective value is zero, then the data set is piecewise-linearly separable. If the data set is not piecewise linearly separable, the positive values of the variables $\mathbf{z}^{ij}$ are proportional to the magnitude of the misclassified points from the plane

$$(\mathbf{w}^i - \mathbf{w}^j) \cdot \mathbf{x} = (\gamma^i - \gamma^j) + 1. \tag{3.42}$$

[5] This figure was provided by Bredensteiner and Bennett (1999).

[6] 'M-; implies here the multicategory.

At the optimal, the margin between classes must be maximized. It will guarantee better accuracy of prediction. It means that the more margin space we have, the less occurrence of overfitting problems. Suppose there are two arbitrary classes i and j such that the margins of their decision borderline are respectively

$$\mathbf{A}^i(\mathbf{w}^i - \mathbf{w}^j) \geq (\gamma^i - \gamma^j)\mathbf{e} + \mathbf{e} \text{ and } \mathbf{A}^j(\mathbf{w}^i - \mathbf{w}^j) \leq (\gamma^i - \gamma^j)\mathbf{e} - \mathbf{e}. \quad (3.43)$$

The distance between two margins in (3.44) is $2/\left\|\mathbf{w}^i - \mathbf{w}^j\right\|$. Therefore, the model M-RLP can be rebuilt to minimize $\left\|\mathbf{w}^i - \mathbf{w}^j\right\|$ as well as the regularization term $\left\|\mathbf{w}^i\right\|$ as follows.

$$\begin{aligned} &\text{(M - SVM) minimize} \quad (1-\lambda)\sum_{i=1}^{K}\sum_{\substack{j=1 \\ j\neq i}}^{K}\frac{\mathbf{e}^T\mathbf{z}}{m^i} + \frac{\lambda}{2}\left[\sum_{i=1}^{K}\sum_{j=1}^{i-1}\left\|\mathbf{w}^i - \mathbf{w}^j\right\|^2 + \sum_{i=1}^{K}\left\|\mathbf{w}^i\right\|^2\right] \\ &\text{subject to } \mathbf{z}^{ij} \geq -\mathbf{A}^i(\mathbf{w}^i - \mathbf{w}^j) + (\gamma^i - \gamma^j)\mathbf{e} + \mathbf{e},\ \mathbf{z}^{ij} \geq 0, \quad i, j = 1,2,...,K,\ j \neq i \end{aligned} \quad (3.44)$$

The original multi-category RLP (M-RLP) for K classes constructed a piecewise-linear discriminant using a single linear programming model (RLP). The k-RLP method provided accurate and efficient results on the piecewise-linear separable datasets. The benefit of M-RLP is that this can be formulated only one model for a multi-category problem while k-RLP requires as many models as classes. Also, the M-SVM model was proposed for eliminating overfitting problems more. On the piecewise linearly inseparable dataset, the polynomial and piecewise-polynomial classifiers has been investigated in an improvement over the M-SVM method (Bredensteiner and Bennett, 1999).

3. A new application of TDIDT with SVM

The objective function of the M-RLP model in (3.41) represents the average distance of misclassified training data from their decision borders. Intuitively it may be not a good measure when the distance of a misclassified data is extremely larger than others. If a training data is abnormally outlier, the misclassified data will lead poor decision boundary. If we change this objective function to the number of misclassified data, we can be achieved more desirable decision boundary conditions. The following model, so-called M-RIP (Robust mixed Integer Programming for Multi-category), can be rewritten from (3.41) as follows

$$\text{minimize} \quad (1-\lambda)\sum_{i=1}^{K}\sum_{\substack{j=1\\ j\neq i}}^{K}\frac{\mathbf{e}^T\mathbf{y}^{ij}}{Km^i}+\frac{\lambda}{2}\mathbf{e}^T\mathbf{s}$$

$$\text{subject to } \mathrm{M}\mathbf{y}^{ij} \geq -\mathbf{A}^i(\mathbf{w}^i-\mathbf{w}^j)+(\gamma^i-\gamma^j)\mathbf{e}+\mathbf{e}, \tag{3.45}$$

$$-\mathbf{s} \leq \mathbf{w}^i \leq \mathbf{s}, \quad i=1,2,...,K,$$

$$\mathbf{y}^{ij} \in \{0,1\}, \quad i,j=1,2,...,K,\ j\neq i,$$

where M is an extremely large number (so-called *Big-M method*). Using the mathematical programming modeling language AMPL[7] (Fourer et al., 1993), this M-RIP (a mixed-integer programming model) was solved for our experimental problems that will be mentioned at the next section. The first term of the objective function of (3.45) represents the average error rate of misclassification over all classes. If the right-handed side of the first constraint at the l^{th} training data in (3.45) is correctly classified, $\mathbf{y}_l^{ij}$ becomes zero at the optimal. However, if it is misclassified, $\mathbf{y}_l^{ij}$ becomes unity at the optimal so that the left-handed side $\mathrm{M}\mathbf{y}_l^{ij}$ can be larger than the right-handed side. Figure 17 shows an illustrative example for the comparison of results between M-RLP and M-RIP. The total number of misclassified errors for M-RLP and M-RIP are 8 and 3, respectively.

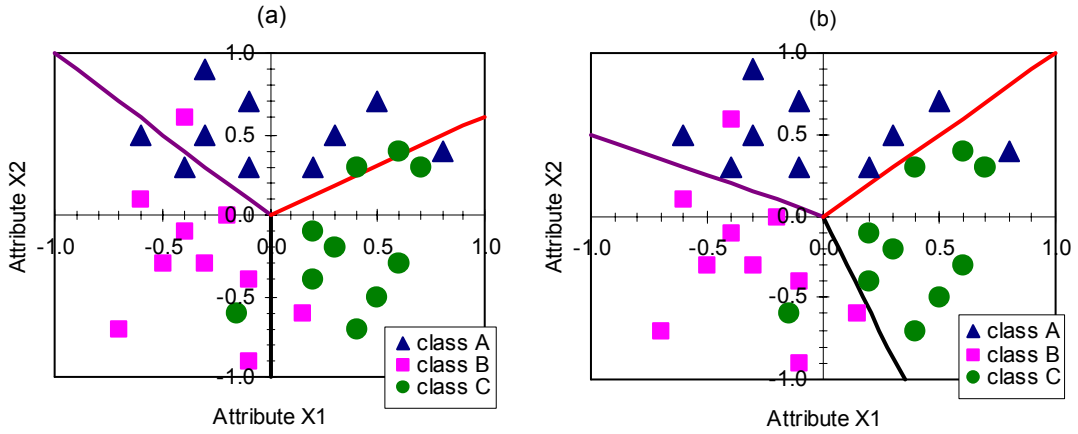


Figure 17. Illustrative example for the comparison of M-RLP and M-RIP: Three thick lines indicate the classification borders (a) from M-RLP, and (b) from M-RIP. The total numbers of misclassified data for M-RLP and M-RIP are respectively 8 and 3 among total 30 training data.

[7] The AMPL code is available on request from the author at "http://www.ampl.com/".

Bennett (1996) and Wu et al. (1998) suggested that the combination of TDIDT and SVM can describe nonlinear decision boundaries to piecewise linear boundaries. Figure 15 shows an illustrative example of how to construct piecewise linear decision boundaries or segments to classify two classes. Here, any numerical space can be described by some finite number of subspaces, so that these subspaces can transform as nominal attributes enable to apply other TDIDT methods. For example, we can transform new nominal attributes X, Y, and Z (see Figure 18) such that $X = \{a_1, a_2\}$, $Y = \{b_1, b_2\}$, $Z = \{c_1, c_2\}$ where,

$$\begin{aligned} a_1 &= \{\mathbf{x} \mid \mathbf{x}^T\mathbf{w}^0 \geq b^0\}, \quad a_2 = \{\mathbf{x} \mid \mathbf{x}^T\mathbf{w}^0 < b^0\} \\ b_1 &= \{\mathbf{x} \mid \mathbf{x}^T\mathbf{w}^1 \geq b^1\}, \quad b_2 = \{\mathbf{x} \mid \mathbf{x}^T\mathbf{w}^1 < b^1\}, \\ c_1 &= \{\mathbf{x} \mid \mathbf{x}^T\mathbf{w}^2 \geq b^2\}, \quad c_2 = \{\mathbf{x} \mid \mathbf{x}^T\mathbf{w}^2 < b^2\}. \end{aligned} \tag{3.46}$$

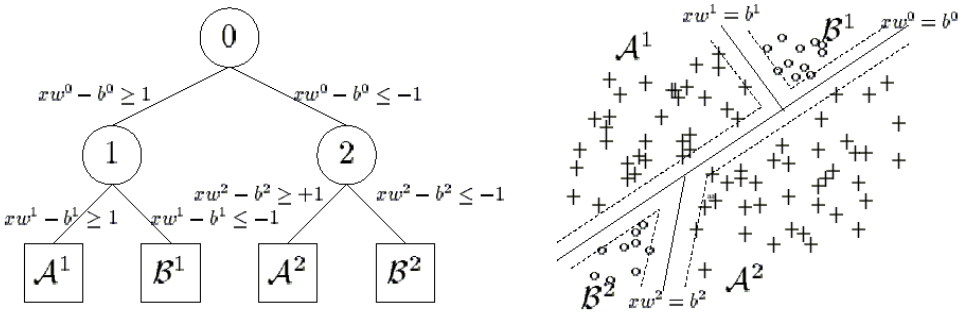


Figure 18. Example of how to convert two-dimensional numerical space to clustered subspaces[8]: $\boldsymbol{A}^1$, $\boldsymbol{A}^2$, $\boldsymbol{B}^1$, and $\boldsymbol{B}^2$ can be nominal attributes for the training sample $\mathbf{x}$.

With these attributes any instances for each subspace in (3.46) can be described as three values of nominal attributes (as shown in Figure 18) as follows:

$$\begin{aligned} &\text{if any } \mathbf{x} \in \mathbf{A}^1, \text{ then } X \in a_1, Y \in b_1, \text{ and } Z \in c_1 \cup c_2; \\ &\text{if any } \mathbf{x} \in \mathbf{A}^2, \text{ then } X \in a_2, Y \in b_1 \cup b_2, \text{ and } Z = c_1; \\ &\text{if any } \mathbf{x} \in \mathbf{B}^1, \text{ then } X \in a_1, Y \in b_2, \text{ and } Z = c_1 \cup c_2; \\ &\text{if any } \mathbf{x} \in \mathbf{B}^2, \text{ then } X \in a_2, Y \in b_1 \cup b_2, \text{ and } Z \in c_2. \end{aligned} \tag{3.47}$$

With the transformation rules in (3.47), the decision tree in Figure 18 can be redrawn as shown in Figure 19. This transformation has very important advantages as follows: (1) any

[8] This figure 18 was provided by Bennett (1996).

nonlinear decision boundaries for the classification can be described as piecewise-linear segments, (2) transforming numerical variables to a nominal attribute allows to apply many other TDIDT methods such as C4.5, PART, SODI, AdaBoost, etc., and (3) it is able to compare information gain or gain ratio between the original nominal attribute and newly converted nominal attributes (subspaces of numerical space).

In the real situation the construction of a classification is common for very huge dimension of a numerical space. Applying the whole numerical attributes for SVM may not preferable since it is not easy to understand as well as it cannot distinguish more important attributes for the classification from quite unnecessary attributes. For the construction of a new algorithm for a decision tree as shown as figure 18, we have two policies: (1) applying a general SVM in (3.30) and (3.31) for a two-class problem, and (2) applying our suggested model M-RIP in (3.45) for a multi-category problem more than two classes. We denote this TDIDT construction algorithm as DT-SVM (see figure 20).

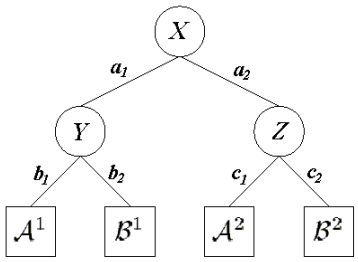


Figure 19. A new decision tree described by three nominal attributes transformed from the two-dimensional numerical space shown in (figure 18).

We attempted to construct a new approach that combines the techniques of C4.5 and SVM in order both to build smaller description of model complexity and to improve prediction accuracy as shown in figure 19. We denote this new algorithm as SVMM (support vector machines for multi-category). It also contains transformation of numerical data to nominal data.

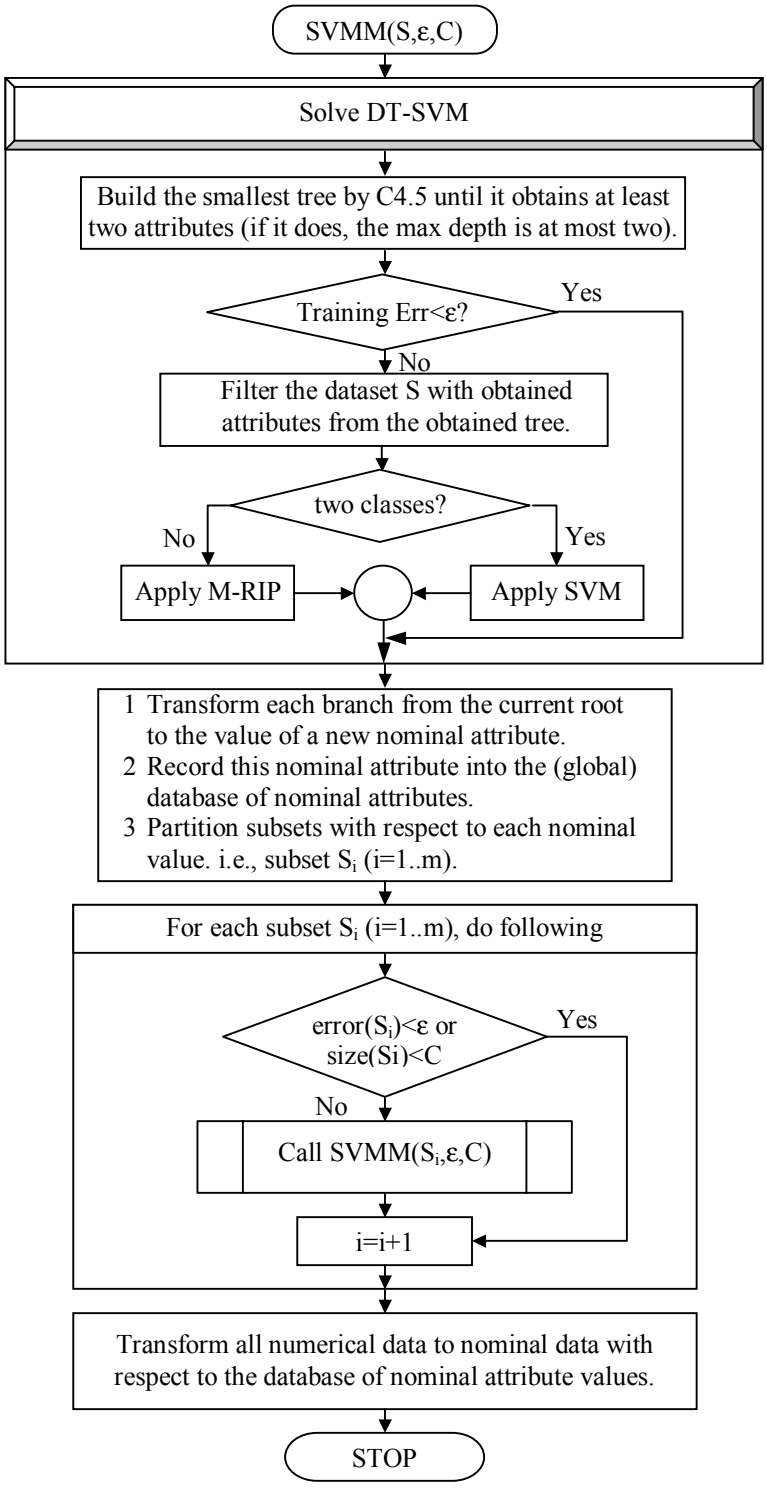


Figure 20. Flowchart for a new TDIDT algorithm, SVMM

Illustrative examples for SVMM

For easy to understand how the new algorithm works, we introduce the 'Iris Plants Database' (Marshell, 1988; see Appendix C for more details) that is a classification problem with four numerical attributes and three classes. Figure 21(a) shows the distribution of three classes on the two dimensional space, 'petal length' and 'petal width'. At the beginning one must set up the minimum training error (ε; the default sets 1%) and the minimum size of training subset at the end leaf (C; the default is 1% of the size of a dataset) for stopping rule.

As a heuristic approach, we only build a temporary decision tree from a simplified C4.5 method that constructs the smallest tree with al least two numerical attributes. The maximum number of obtained numerical attributes from the smallest tree is 3 when the depth of tree is two and each internal node has different attributes. With these selected attributes, we can build a new decision tree by using DT-SVM as shown in 18. Figure 21(b) shows the smallest decision tree of the iris problem by using C4.5. As the candidates of DT-SVM, 'pedal length' and 'pedal width' we found as the best two attributes.

We filtered the original problem to a temporary problem with only these two attributes. Since the number of classes is three, we adopted M-RIP for solving this problem. Figure 22 shows the result. From the first solution of an initial M-RIP model, there were three subspaces found for classification as follows:

$$\begin{aligned} S_1 &= \{(x,y) \mid 0.489x + 0.511y - 1.579 \le 0\}, \\ S_2 &= \{(x,y) \mid 0.489x + 0.511y - 1.579 > 0 \; and \; 0.422x + 0.578y - 3.043 \le 0\}, \\ S_3 &= \{(x,y) \mid 0.422x + 0.578y - 3.043 > 0\}, \end{aligned} \quad (3.48)$$

where x is the 'petal length', and y is the 'petal width'. In the subspace S_1, the class 'setosa' was perfectly classified. In the subspace S_2, 47 points of versilcolor were correctly classified, but 3 points of virginica were misclassified. In the subspace S_3, 47 points of virginica were correctly classified, but 3 points of versicolor were misclassified.

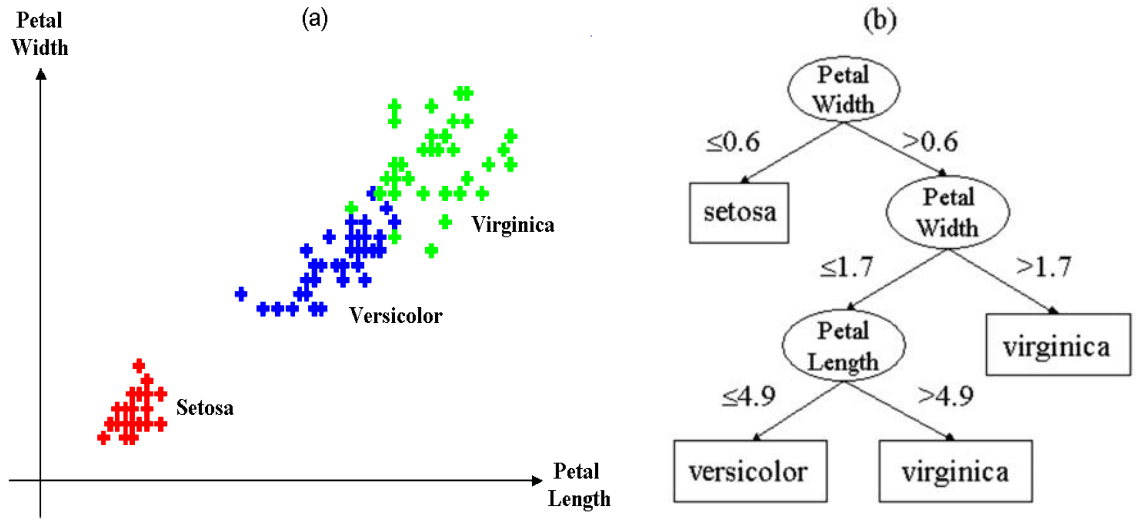


Figure 21. An illustrative example of the 'Iris Plant' database: (a) the distribution of three classes (setosa, versilcolor, and viginica) on the two dimensional space, 'petal length' and 'petal width', and (b) the result of a decision tree construction up to depth 2 by the simplified C4.5 method.

According to our DT-SVM algorithm (misclassification error of both S_2 and S_3 =6% > ε), the subsets S_2 and S_3 were required to branch more. A new nominal attribute (arbitrary named A) is then

$$\begin{aligned} &A = \{a_1, a_2, a_3\}, \text{ where} \\ &a_1 = (0.489x + 0.511y - 1.579 \le 0), \\ &a_2 = ((0.489x + 0.511y - 1.579 > 0) \text{ AND } (0.422x + 0.578y - 3.043 \le 0)), \text{ and} \\ &a_3 = (0.422x + 0.578y - 3.043 > 0). \end{aligned} \tag{3.49}$$

When we apply M-RIP for the subset S_2, we found the only one decision function $f(x, y) = 0.5x + 0.5y - 3.225$, where x is the 'petal length', and y is the 'petal width'. If $f(x, y)$ is positive, the data point is classified as 'virginica'; otherwise, it is 'versicolor'. On the other hand, the subset S_3 was found as linearly inseparable, so that there was no further improvement of training accuracy. Therefore, a new nominal attribute is as follows:

$$\begin{aligned} &B = \{b_1, b_2\}, \text{ where} \\ &b_1 = (0.5x + 0.5y \le 3.225), \text{and} \\ &b_2 = (0.5x + 0.5y > 3.225). \end{aligned} \tag{3.50}$$

At the final solution of DT-SVM, we have four end leaves for this classification, and the decision rules for these end leaves are as follows:

$$\begin{aligned} &\text{Rule 1}: A = a_1; \\ &\text{Rule 2}: A = a_2 \text{ AND } B = b_1; \\ &\text{Rule 3}: A = a_2 \text{ AND } B = b_2; \\ &\text{Rule 4}: A = a_3; \end{aligned} \tag{3.51}$$

where the nominal attributes A and B are defined in (3.49) and (3.50), respectively. Since these subsets are all classified with the stopping rules, our SVMM algorithm stopped here. Figure 22 shows all decision boundaries generated by the DT-SVM method.

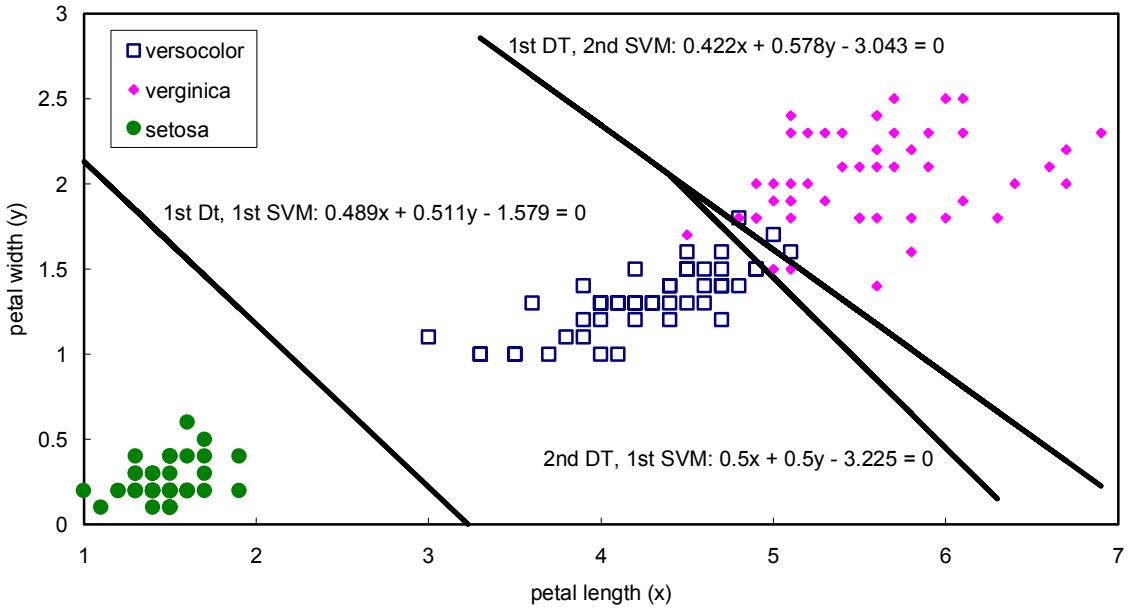


Figure 22. An illustrative example for the TDIDT construction by DT-SVM: the 'iris plant' problem was classified by two depths of decision trees. The total accuracy for training is 2.67% (there are 4 misclassified points among 150 points). At the final solution of DT-SVM, we have four end leaves for this classification.

Another illustrative example is the 'Ionosphere' classification problem from Johns Hopkins University that collected the radar data in Goose Bay, Labrador (see Appendix C for more details). The 'Ionosphere' database consists of 34 numerical attributes for a two-class problem. The numerical attributes were indexed from 'a01' to 'a34', and the class was

described as either ‘b’ or ‘g’. At the beginning, we set up SVMM parameter ε and C as 0.05 and 10, respectively.

At the first step, by applying the simplified C4.5 method, we found best two attributes ‘a01’ and ‘a05’. With these two attributes, the original dataset was classified by SVM with the decision border

$$D_0(a01,a05) = 2.74616\ a01 + 2.67567\ a05 - 3.67644 \quad (3.52)$$

Then, the original dataset has been partitioned two subsets S1 and S2 such that S1 was collected by all instances that has non-positve values of $D_0(a01,a05)$, and S2 consisted of the remaining instances.

For the subset S1, there are three important attributes to classify this subset: ‘a01’, ‘a03’, and ‘a05’. When we applied C4.5, it was almost perfectly classified with depth 2 as follows (total 101 instances are in S1):

1. If a05 <= 0.0409, classified as ‘b’ (67 instances).
2. If a05 > 0.0409 and a01 <= 0, classified as ‘b’ (19 instances).
3. If a05 > 0.0409 and a01 > 0 and a03 <= 0.10135, classified as ‘b’ (5 instances).
4. If a05 > 0.0409 and a01 > 0 and a03 >0.10135, classified as ‘g’ (10 instances and 1 misclassified data).

Since this training error is much less than ε, we stopped partitioning S1 any further.

For the subset S2 we found three attributes within the tree depth 2 according the C4.5 method: ‘a03’, ‘a08’, and ‘a27’. With these three attributes, the subset S2 was classified by with the decision border

$$D_1(a03,a08,a27) = 1.22880\ a03 + 0.95557\ a08 - 0.72741\ a27 + 0.21113. \quad (3.53)$$

Then, S2 has been partitioned two subsets S21 and S22 such that S21 was collected by all instances that have non-positve values of $D_1(a03,a08,a27)$, and S22 consisted of the remaining instances. In the subset S21, among 17 instances, 15 points are classified as ‘b’ and there is only two points of ‘g’. Within the error limit ε, the subset S1 was stopped here. For subset S22 we found three important attributes of ‘a16’, ‘a21’, and ‘a27’. With these attributes SVM constructed the following decision boundary:

$$D_2(a16,a26,a27) = -1.26991\ a16 + 0.05526\ a21 - 0.98858\ a27 + 1.87212 \quad (3.54)$$

Then it partitioned two subsets: S221 for negative D_2(a16,a26,a27), and S222 for positive D_2(a16,a26,a27). In S221, 5 instances are class 'b' among 6 points. On the contrary, in S222, there are 213 data that are classified as 'g' among total 227 instances. Since the training error (14/227=0.0617) is higher than ε, further loop has been processed. However, this subset was turned out linearly inseparable by any SVM. So, we terminated the algorithm SVMM here. The final solution for this 'Ionosphere' problem is as follows

D_0(a01,a05)=2.74616 a01 + 2.67567 a05 – 3.67644
| S1={D_0(a01,a05) <=0}
| | a05 <= 0.0409: class 'b' (67.0)
| | a05 > 0.0409
| | | a01 <= 0: class 'b' (19.0)
| | | a01 > 0
| | | | a03 <= 0.10135: class 'b' (5.0)
| | | | a03 > 0.10135: class 'g' (10.0/1.0)
| S2={D_0(a01,a05)>0}: D_1(a03,a08,a27)=1.2288 a03 + 0.9556 a08 – 0.7274 a27 + 0.2111
| | S21={D_1(a03,a08,a27)<=0} : class 'b' (17.0/2.0)
| | S22={D_1(a03,a08,a27)>0}:D_2(a16,a26,a27)=–1.270 a16 + 0.055 a21 – 0.989 a27 + 1.872
| | | S221={D_2(a16,a26,a27)<=0}: class 'b' (6.0/1.0)
| | | S222={D_2(a16,a26,a27)>0}: class 'g' (227.0/14.0)

Training accuracy 94.9%

		Classified as	
		'b'	'g'
Actual class	'b'	111	15
	'g'	3	222

Estimated prediction accuracy 93.7%

		Classified as	
		'b'	'g'
Actual class	'b'	99	17
	'g'	5	220

Figure 23. The final decision tree for 'Ionosphere' is constructed by SVMM. The table shows the confusion matrices for both training and 10-fold cross validation results.

4. Experimental results

In this section we present more extensive numerical results for testing the SVMM algorithm along with the comparisons with C4.5, PART, SVM (two-class problems only) and M-RIP (as mentioned in (3.46)).

We analyzed six classification problems that are widely used in the data mining literature (see Witten and Frank, 1999) as follows:

1. Breast cancer (9 attributes, 2 classes, 699 instances)
2. Pima Indians Diabetes (13 attributes, 2 classes, 768 instances)
3. Heart Statlog (13 attributes, 2 classes, 270 instances)
4. Iris Plants (4 attributes, 3 classes, 150 instances)
5. Ionosphere (34 attributes, 2 classes, 351 instances)
6. Sonar (60 attributes, 2 classes, 208 instances)

The performance of the SVM algorithm was evaluated by the comparisons with C4.5, PART, SVM, and M-RIP as shown in table 7. The first column and second column in Table 7 indicates the testing problems and TDIDT methods, respectively. The third column in Table 7 shows both the number of end leaves (i.e., final decision rules) and the number of attributes that used in the model. They are most interesting measures for model complexity. The forth and fifth columns show the training errors and estimated prediction (via cross-validation) with percentage, respectively. The sixth column is the absolute gap between the training error and prediction error. Finally, the seventh column shows the standard deviation of estimated prediction errors computed from the sampling of 10-fold cross-validation.

Table 8 shows scoring result for user-defined penalty categories mentioned at Chapter 1. The scoring policy has been also mentioned at the beginning of the previous chapter (it is very subject to user's preference, and so the final result may not different if the scoring policy is changed). The average value of gaps between training errors and estimated prediction errors is smaller at SVM than at any other methods. However, SVMM could describe the classification problems with smaller number of attributes than SVM.

Table 7. Comparison of SVMM with other classification methods

Problem Set	Method	N(leaves) / N(attribs)	(a) Training Error (%)	(b) Prediction Error (%)	Gap (%) |(a)-(b)|	Std. Dev.[3] of Prediction Err
Breast Cancer (2 classes)	C4.5	16 / 9	1.57	5.43	3.86	1.82
	PART	10 / 8	1.57	5.15	3.58	1.76
	SVM	2 / 9	2.86	3.29	0.43	1.11
	M-RIP	2 / 9	3.00	3.43	0.43	1.15
	SVMM	5 / 4	3.15	3.74	0.59	1.29
Pima Indians Diabetes (2 classes)	C4.5	22 / 6	15.63	25.52	9.90	8.86
	PART	13 / 7	18.75	26.17	7.42	8.92
	SVM	2 / 8	22.53	23.57	1.04	7.85
	M-RIP	2 / 8	22.14	23.57	1.43	7.85
	SVMM	12 / 2	20.18	22.53	2.53	7.58
Heart Statlog (2 classes)	C4.5	18 / 11	8.52	18.52	10.00	6.33
	PART	24 / 10	5.56	19.26	13.70	6.87
	SVM	2 / 13	14.81	17.04	2.23	5.68
	M-RIP	2 / 13	12.96	16.30	3.34	5.42
	SVMM	9 / 6	11.48	16.30	4.82	5.42
Iris Plants (3 classes)	C4.5	5 / 2	2.00	4.67	2.67	1.56
	PART	3 / 2	2.67	4.00	1.33	1.33
	SVM	3 / 4	0.67	3.33	2.67	1.11
	M-RIP	3 / 4	2.67	4.00	1.33	1.33
	SVMM	4 / 2	2.67	4.00	1.33	1.33
Ionosphere (2 classes)	C4.5	18 / 13	0.28	8.55	8.27	3.20
	PART	10 / 14	0.28	8.26	7.98	3.06
	SVM	2 / 34	8.55	11.40	2.85	3.96
	M-RIP	2 / 34	7.69	10.83	3.14	4.01
	SVMM	7 / 7	5.13	6.27	1.14	1.05
Sonar (2 classes)	C4.5	18 / 13	1.92	28.85	26.93	10.20
	PART	8 / 15	0.96	19.71	18.75	7.23
	SVM	2 / 60	12.50	24.04	11.54	8.01
	M-RIP	2 / 60	12.02	22.12	10.10	7.37
	SVMM	13 / 13	5.29	23.56	18.27	7.85

3 Std. Dev. = sample standard deviation from cross validation results for each learning model.

Table 8. Comparison of SVMM with other classification methods (scoring§ results from Table 7)

Problem Set	Method	Model Complexityℨ	Prediction Error	Error gap betw/ training and testing	Average Score
Breast Cancer (2 classes)	C4.5	1	2	2	1.67
	PART	2	2	2	2.00
	SVM	4	4	5	4.33
	M-RIP	4	4	5	4.33
	SVMM	5	5	5	**5.00**
Pima Indians Diabetes (2 classes)	C4.5	1	3	1	1.67
	PART	3	2	2	2.33
	SVM	5	4	5	**4.67**
	M-RIP	5	4	5	**4.67**
	SVMM	4	5	4	4.33
Heart Statlog (2 classes)	C4.5	3	3	2	2.77
	PART	2	2	1	1.67
	SVM	5	4	5	4.67
	M-RIP	5	5	5	**5.00**
	SVMM	5	5	4	4.67
Iris Plants (3 classes)	C4.5	3	3	3	3.00
	PART	5	4	5	**4.67**
	SVM	3	5	3	4.33
	M-RIP	3	4	5	4.00
	SVMM	4	4	5	4.33
Ionosphere (2 classes)	C4.5	3	4	1	2.67
	PART	4	4	1	3.00
	SVM	2	2	4	2.67
	M-RIP	2	3	3	2.67
	SVMM	5	5	5	**5.00**
Sonar (2 classes)	C4.5	4	1	1	2.00
	PART	5	5	3	**4.33**
	SVM	1	3	5	3.00
	M-RIP	1	4	5	3.33
	SVMM	5	3	3	3.67

§ Every criterion has been assigned to the same weight for each problem. The policy of both selecting criteria and making their scores for each problem is subject to a user's preference.

ℨ The model complexity has been measured by the sum of the numbers of both end leaves and used attributes.

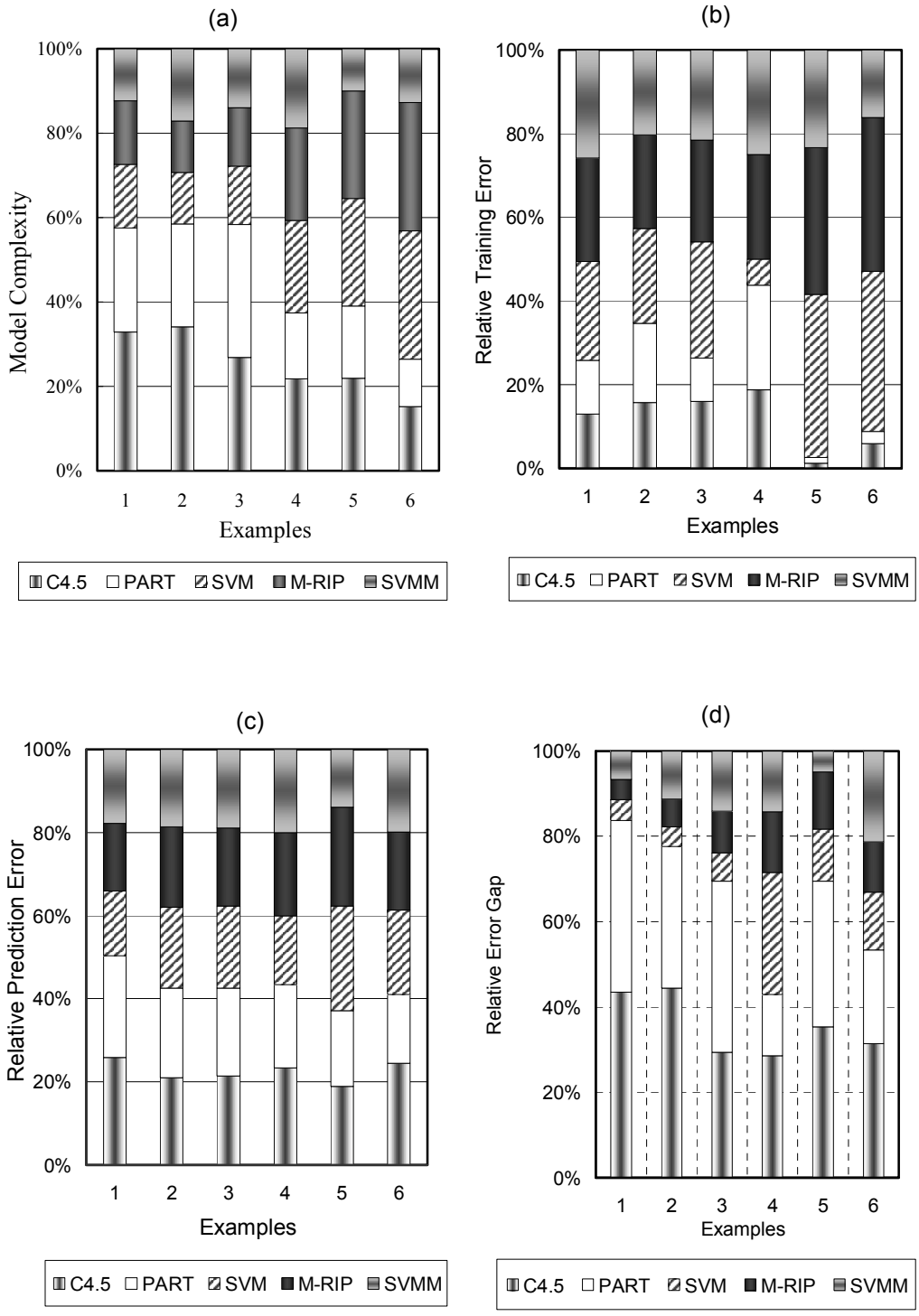


Figure 24. Performance evaluation of SVMM comparing with C4.5/PART/SVM/M-RIP: (a) the proportion of number of decision rules, (b) training error, and (c) estimated prediction error, and (d) the gap between training error and prediction error

Figure 24(a) shows the relative ratio of the number of end leaves or decision rules of the decision trees. Figure 24(b) and 24(c) shows the comparison of the training errors and the estimated prediction errors. Finally, figure 24(d) shows the error gap between training and testing results. As shown in figure 24(c) the estimated prediction errors for each problem as had no big differences between methods. For the case of 'Ionosphere' problem, SVMM performed outstanding rather than any other methods. SVMM generated higher accuracy of both training and cross-validation testing results. On average, however, a general SVM or M-RIP generated more pruned decision schemes (i.e., the gaps between training and cross-validation for over all problems are relatively smaller than any other methods on average). However, for the viewpoint of understandability, these methods may be not preferable since the visualization of classification results is very difficult caused by their extremely high dimension. However, for SVMM as a combination between C4.5 and SVM, the classification descriptions for numerical attributes are limited to use at most 3 attributes for each decision node (or, internal node). It means that we are able to visualize the classification boundary on screen easily.

Both C4.5 and PART created highest training accuracy of classification on average, but they had poor prediction accuracy at the same time. It implies these methods may meet higher change of overfitting problems. If means also that we cannot trust the classification results without any huge size of training dataset. However, SVM improved better prediction accuracy than both C4.5 and PART as shown in figure 24(c).

The winners by the user-defined scoring policy from Table 8 are almost uniformly distributed over classification methods except for C4.5 (C4.5 scored lowest ranks for all problems). In general, SVMM performed better than any others when a classification is very complicated or numerical attributes are highly corrected each other. SVMM always be in the first or second rank for each problem as shown in Table 8. It implies SVMM is generally recommendable method for solving any classification problem with numerical attributes.

Also, the M-RIP as mentioned before performed as well as the conventional SVM. Because the M-RIP model is mixed integer programming problem, computational efforts between SVM and M-RIP does not have significant difference in experience.

5. Summary

The benefits of SVMM with respect to either C4.5 or PART came from how to describe the decision boundaries. C4.5 treats the classification boundaries as axis-orthogonal (if we assume any numerical attribute as an axis), so that every decision areas may be shaped as rectangle, cubic, or hyper-cubic geometries. However, SVMM can generate more flexible convex subspace consisting of more flexible two- or three-dimensional convex subspace from some subsets of numerical attributes for the decision tree branch. This flexibility can guarantee the improvement of model complexity as well as overfitting problems.

There is a trade-off relationship between model complexity and prediction accuracy in general data mining problems. For example, the problem of either 'Ionosphere' or 'Sonar' requires very huge number of attributes to determine the classification boundary, even though it provided smaller error gaps between training and cross-validation than either C4.5 or PART. The SVMM method took the advantages between C4.5 and the general SVM. From a benefit of C4.5, SVMM can identify which attributes are more important than any others for the purpose of easy to understand. From a benefit of SVM, SVMM can describe the decision boundary more flexible ways than C4.5. With the combination of C4.5 and SVM, SVMM can describe the decision boundaries with piecewise linear shapes. Therefore, SVMM has more chance to solve linearly inseparable problems determined by SVM.

Also, SVMM provide the function of transformation that has very important advantages as follows: (1) any nonlinear decision boundaries for the classification can be described as piecewise-linear segments, (2) transforming numerical variables to a nominal attribute allows to apply many other TDIDT methods such as C4.5, PART, SODI, AdaBoost, etc., and (3) it is able to compare information gain or gain ratio between the original nominal attribute and newly converted nominal attributes (subspaces of numerical space). It means that, with SVMM, it is able to compare numerical attributes with nominal attributes by the measurement of information gains or gain ratio. In the next chapter, we will discuss more details how to combine SODI and SVMM in order to solve more complicated and nonlinear problems.

CHAPTER 4. IDSS: A NEW TDIDT CLASSIFICATION ALGOIRTHM

For the successful classification, it is very important to explore a simple, accurate and reliable learning system. However, it is well known to exist a trade-off relationship between model complexity and prediction accuracy in general classification problems. In the research area of TDIDT classification, the model complexity is usually highly related to the size of a decision tree and its decision descriptions. If a construction algorithm of decision trees is possible to extremely reduce the size of trees by using little more complex decision descriptions rather than any other conventional methods, such as C4.5 (Quinlan, 1993) or ADTree (alternating decision tree learning algorithm; Freund and Mason, 1999), it is possible to reduce the overall model complexity with respect to the minimum length description (MDL; Rissanen, 1989). Both SODI and SVMM are the examples for this case. These methods use two (both SODI and SVM) or three (SVMM only) combination of attributes for describing decision-makings that are more complex than C4.5 or SVM. However, from the results of chapter 2 and chapter 3, both SODI and SVMM reduced the model complexity more than the univariate TDIDT method, C4.5, as well as improved the estimated prediction accuracy. Furthermore, with little more complex than SVM, SVMM improved both training and prediction accuracy in commonly complex problems. Especially when the number of numerical attributes is too many, SVMM could reduce the model complexity more than SVM as shown in chapter 3.

The reliability of data mining systems is highly related to overcoming *overfitting* problems. When one build a classification decision tree in order to enhance the training accuracy, he may make a mistake if he develop the tree in details. It is very possible for such a decision tree to create unproved decision rules too much. That is, some decision-makings may only fit for very few cases of the current training dataset. This situation is called as '*overfitting*'. The problem of trading off the simplicity of a model with how well it fits the training data is a well-studied problem. In statistics this is known as the *bias-variance tradeoff* (Friedman, 1997). The most preferable measure of overfitting problems is to compute the estimated error gap between training and testing. For some statistic analysis of the estimating the prediction accuracy, or for the case of lack of sampling of classification

examples, the cross-validation is recommendable, even though it costs more computationally. Because of fast growing computational advances, the computational effort is not big issues in TDIDT field any more (TDIDT was turned out to require polynomial-time computational effort).

Suppose a model 'A' construct a decision tree with 95% training accuracy, but the estimated prediction accuracy is 70%. Suppose also a model 'B' builds a tree with 90% training accuracy and obtains 85% of estimated prediction accuracy. Then, which model is the better? This question is about the reliability of decision-making models. Of cause we must consider the variance of estimated prediction accuracy in the viewpoint of statistical comparison. Probability the measurement of model reliability can be described by the weighted sum of both the absolute difference between the accuracy of training and the average accuracy of prediction from cross-validation and the standard deviation of prediction accuracy. That is, if the difference between training and testing accuracy and the sample variance of prediction accuracy If the measurement is smaller than any others, the model is the currently best for the reliability. Even though the difference between training and testing is small, if the variance of the estimated prediction accuracy is too high, this model may be not good for trust. With another sampling, the result can be differently produced. The opposite case also has the same situation. Even if the variance is quite small, but the difference of accuracies is too much, then, we can predict that the model shows overfitting problems constantly.

To build a decision tree with nominal attributes in recent years, there is very little research for the consideration of nonlinear or multivariate decision boundary description. With this limited capability, it has always an overfitting risk. In this thesis, new approaches of decision-tree construction have been developed in order to reduce the overfitting problems while the model complexity is not seriously increased. We presented the second-order decision-tree induction (SODI) for only nominal attribute problems and the support vector machines for multi-category (SVMM) for only numerical attribute problems. Now, in this chapter, we introduce a new algorithm of TDIDT, IDSS (Induction of Decision trees with SODI and SVMM) for both nominal and numerical attributes by the combination of SODI

and SVMM. For easy to understand and for the evaluation of IDSS, we provide two examples that are 'Germany credit approval' problem and 'the classification problem of MFL signals' as an application of nondestructive evaluation.

1. IDSS: induction of decision trees using SODI and SVMM

The policy of defining decision boundaries is the most essential for the classification problem to achieve both training accuracy and prediction accuracy. The shapes of decision boundaries are also highly related to the model complexity. It is obviously trade-off relationship between the model complexity and the model accuracy (of not only training but also prediction). Univariate decision trees such as C4.5 and ADTree build less complex models than multivariate decision trees. On the other hand, the multivariate decision tree can improve both training and prediction accuracy rather than the univariate one. However, these extreme cases may not the optimal. Since 'fining the best descriptions of multivariate decision-makings' is known to be NP-complete, and multivariate ways may be not preferable for easy to understand their users. We suggested two methods, SODI and SVMM, by limiting the maximum combination of attributes up to two or three. With this heuristic condition, we found well performance from several examples.

SODI for a new TDIDT algorithm has been introduced for any classification problems with nominal attributes only. In general SODI performs higher quality of classification results than other univariate TDIDT methods. Without pruning process of SODI it may meet some overfitting problems. However, the pruned SODI generated the smallest size of decision trees on average as well as the smallest error gap between training and cross validation. It implies this pruning SODI has more capable for removing overfitting problems in many cases.

SVMM for the classification with numerical attributes ahs been developed and taken the advantage of both C4.5 and PART. C4.5 could describe nonlinear boundary more than SVM, but the classification boundaries generated C4.5 is always orthogonal to attributes. If the real decision boundaries are oblique, C4.5 will generate either so may overfitting problems or very low accurate decision-makings. On the contrary, SVM can handle this

oblique boundaries as well as more linearly flexible decision boundaries. However, if cannot distinguish which attributes are more important than others while C4.5 can do. Also, for easy to understand or for the visualization of the decision rules, SVM may be not recommendable when the number of attributes is too huge (more than 3 attribute cannot visualize the decision boundaries). SVMM can generate, however, more flexible convex subspace consisting of more flexible two- or three-dimensional convex subspace from some subsets of numerical attributes. As the advantages of both C4.5 and SVM, SVMM can describe nonlinear decision boundaries with piecewise linear segments, find the order of most important attributes from the top of decision tree to end leaves. This flexibility of SVMM can guarantee the improvement of both model complexity and prediction accuracy as well as the removal of overfitting problems more.

We have been developed a new algorithm of TDIDT, IDSS (Induction of Decision trees with SODI and SVMM) for any attributes consisting of SODI and SVMM. Figure 25 shows the flowchart of IDSS algorithm. IDSS is processed recursively until all classification conditions for each end-node are satisfied.

For starting IDSS, one must prepare the dataset S, the information of nominal attributes R (the 'R' means here the remaining nominal attributes), the stopping criteria α for SODI expansion, the tolerance limit ε for accepting pruned C4.5 tree, and the minimum size of dataset C for branching subtrees.

Then, IDSS looks for the best nominal attributes by using SODI method as shown at the left top in figure 25. The algorithm is almost same as figure 4 (the original SODI algorithm) except for the recursive function by itself. The function 'Best_Pair(A_1,A_2,…,A_n)' is the same with 'Find_Best_Pair(A_1,A_2,…,A_n)' in figure 4, which looks for the best pair among the remained nominal attributes. Then, the SODI internal module will decide which attribute(s) should be chosen by the measurement of information gain ratio between the best single attribute and the best pair. After applying the best set of attributes, the dataset can be temporarily partitioned and be able to compute the information gain ratio.

From the SVMM module as shown at the right top in figure 25, a pruned C4.5 decision tree is temporarily built by numerical attributes only. This is the same as the original SVMM in figure 20. If the pruned decision tree satisfies the training accuracy limit, the current dataset will accept this C4.5 tree. Otherwise, with two or three numerical attributes, either the conventional SVM or the M-RIP method is processed. By applying the results of decision functions the dataset S is temporarily partitioned and computes the information gain ratio. Up to now, we found best two candidates of both nominal and numerical attributes.

Now, we seek the current best attributes between nominal and numerical attribute by comparing their information gain ratio. If the nominal set is better than the numerical, then we accept the split of S resulted by these nominal attributes, and update the set of remaining nominal attributes, R, excluding these current best attributes. Otherwise, we accept the result from the best numerical attributes. Then, for each subset S_i, we recursively apply this IDSS module.

This branch algorithm of IDSS is a depth-first method. Once an initial subset is found, then the sunsets of this set will be searched at first, and so on. It will be processed until a branch is terminated as an end leaf and, then, look forward to the next branch. After branching all the subtrees, a reduced error pruning can be applied by the user's options.

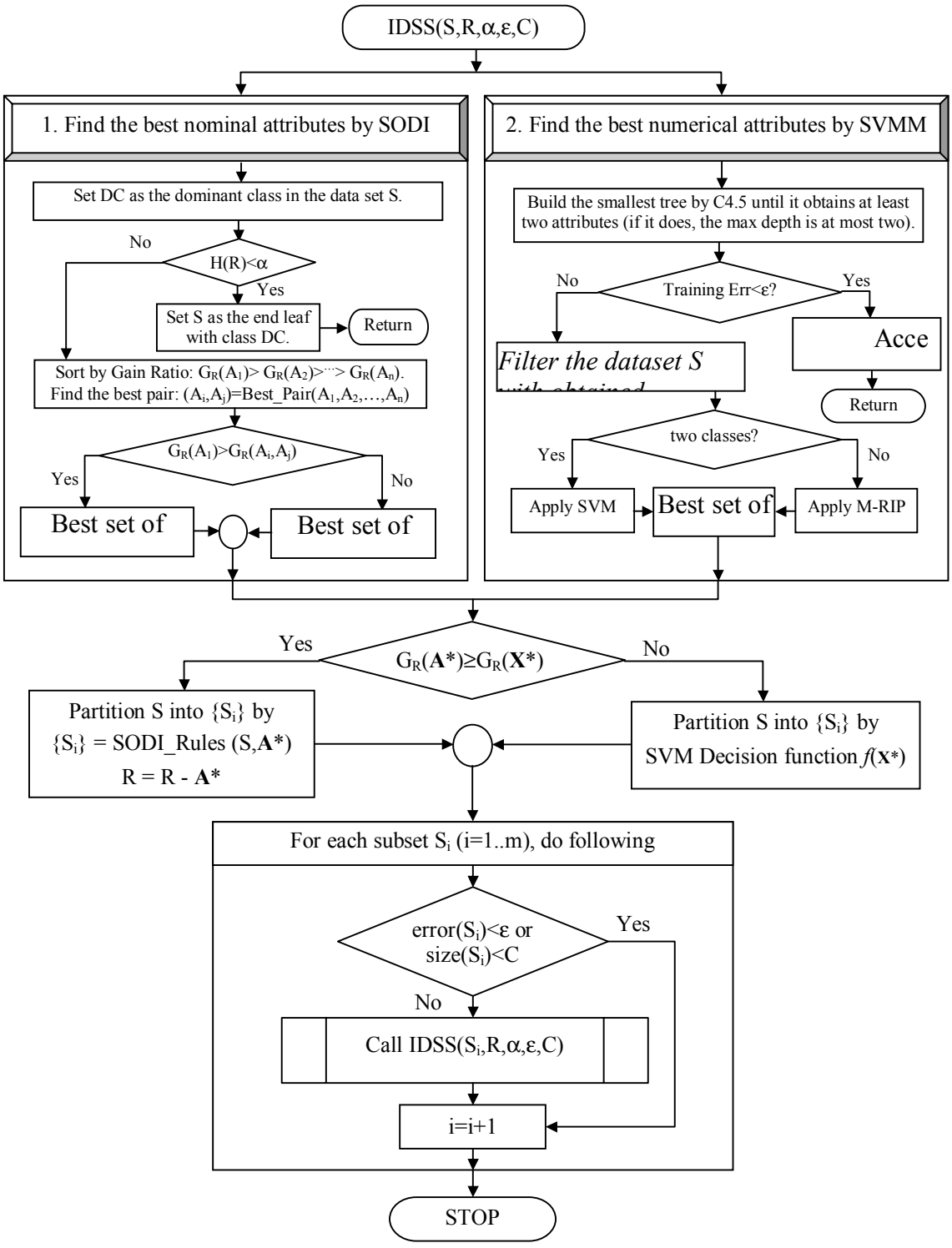


Figure 25. Flowchart for a new TDIDT algorithm, IDSS.

2. Case study A: German credit approval problem

The application presented here was undertaken for a German bank, its products being for example checking accounts, credit cards and investment schemes. Depending on the type of product the target group may differ considerably. Offering useful and affordable products to its customers is not only a question of the bank's marketing expenses but also of its credibility. We employed the German credit dataset for explain how to build and evaluate IDSS with respect to other conventional methods (see Appendix C for their citations). Table 9 shows the training data distribution for all attributes including class.

Table 9. The summary of German credit approval database

	Attribute Name	Type	Distinct	Distribution
1	Checking status	Nominal	4	A:.274, B:.269, C:.063, D:.394
2	Duration (in month)	Numerical	33	Mean=20.90; St.Dev= 12.06 [4, 72]
3	Credit history	Nominal	5	A:.040, B:.049, C:.530, D:.088, E:.293
4	Purpose	Nominal	10	A:.234, B:.103, C:.181, D:.280, E:.012 F:.050, G:.000, H:.009, I:.097, J:.012
5	Credit amount	Numerical	921	Mean=3,271; St.Dev=2,822 [250, 18424]
6	Savings status (account/bonds)	Nominal	5	A:.603, B:.103, C:.063, D:.048, E:.183
7	Present employment since	Nominal	5	A:.062, B:.172, C:.339, D:.174, E:253
8	Installment commitment	Numerical	4	Mean=2.973; St.Dev=1.119 [1, 4]
9	Personal status (including sex)	Nominal	4	A:.050, B:.310, C:.548, D:.092, E:.000
10	Other parties (debtors/guarantors)	Nominal	3	A:.907, B:.041, C:.052
11	Present residence since	Numerical	4	Mean=2.845; St.Dev=1.104 [1, 4]
12	Property magnitude	Nominal	4	A:.282, B:.232, C:.332, D:.154
13	Age (in years)	Numerical	53	Mean=35.546; St.Dev=11.375 [19, 75]
14	Other payment plans	Nominal	3	A:.139, B:.047, C:.814
15	Housing	Nominal	3	A:.179, B:.713, C:.108
16	Number of existing credits at this bank	Numerical	4	Mean=1.407; St.Dev=0.578 [1, 4]
17	Job	Nominal	4	A:.022,B:.200, C:.630, D:.148
18	Number of dependents	Numerical	2	Num=1: 0.845, Num=2: 0.155
19	Own telephone	Nominal	2	A:.596, B:.404
20	Foreign worker	Nominal	2	A:.963, B:.037
	Class = {good, bad}	Nominal	2	good: 0.700, bad: 0.300

The index of 'A,B,C…' in the right column means hear the indices of each attribute values.

The German credit dataset contains information on 1000 loan instances, and each instances is described by 20 attributes, which consist of 14 nominal attributes, 4 discrete numerical attributes, and 3 continuous numerical attributes (see Table 9). A classification assigned to each instance whether a loan applicant is a good or bad credit risk. In the data set, there are a total of 700 cases of good applicants and 300 cases of bad applicants. Since the attribute, 'Number of dependents', has only two values even if it is numeric, we converted this as a nominal attribute valued as '1=single' or '2=multiple'.

In most real-world domains, attributes can have costs of measurement, and objects can have misclassification costs. If the measurement of misclassification costs is not identical between different classes, decision tree algorithms need to be designed explicitly to prefer cheaper trees. Methods to incorporate attribute measurement costs typically include a cost term by using prior probabilities or cost matrix into the feature evaluation criterion. In German credit approval, there are two types of misclassification: classifying a customer as a 'good' credit applicant when he/she is 'bad' (so-call Type-I error), and classifying a customer as a 'bad' credit applicant when he/she is 'good' (so-call Type-II error). We arbitrarily set up misclassification costs such that the former costs relatively as much as 5 for each case, and the latter costs relatively as much as 1 for each case.

In this section we introduce how to build a decision tree by using the IDSS algorithm as shown in figure 25. For the comparison of IDSS, we used 10-fold cross-validation tests with the following conventional classification methods:

1. Naïve-Bayes statistical classification (John and Langley, 1995)
2. C4.5 (Quinlan, 1993)
3. PART (Frank and Witten, 1998a, 1998b)
4. JRip (implementation of RIPPER rule learner; Cohen, 1995)
5. SMO (Support vector machines; Platt, 1998)

For starting IDSS, we need to find two types of best attributes: best nominal attributes from SODI evaluation and best numerical attributes from SVMM evaluation over all dataset. For nominal attributes only, SODI found '*Checking status*' has better gain ratio than any

other pairs of nominal attributes. For numerical attributes only, SVMM found the best set of numerical attributes are '*Duration*', '*Age*', and '*Present residence since*'. The decision boundary is expressed by

$$2.71\times10^{-5}\ duration + 3.61\times10^{-4}\ residence\_since + 4.48\times10^{-6}\ age - 1.00. \quad (4.1)$$

However, this decision boundary did not improve information gain ratio as much as SODI did. Therefore, at the initial step, IDSS chose the attribute '*Checking status*' for the initial tree split. According SODI_Rules(original dataset, *Checking status*), the attribute values 'checking<0' and '0<=checking<200' are clustered (Rule No. 4: too small information gain). Therefore, at the initial split, IDSS branched three subset, says S1, S2, and S3, which were branched by the decision, ('checking<0' or '0<=checking<200'), ('checking>=200'), and ('no checking'), respectively.

For the subset S1, SODI in the second recursive loop of IDSS found that the pair of '*Property magnitude*' and '*Savings status*' is the best selection. It resulted in 120 Type-I errors and 84 Type-II errors, so that the total cost paid 684. SVMM for the subset S1 found the best set of numerical attributes as ('*Duration*', '*Installment commitment*'), which has the decision boundary

$$DC(S1) = 3.96\times10^{-3}\ duration + 0.130\ installment\_commitment - 0.420. \quad (4.2)$$

With this decision function, SVMM found 68 Type-I errors and 177 Type-II errors, so that the total cost is 517, which is less than the result of SODI. Since we use cost factors, our information gain ratio to compare the results between SODI and SVMM has been changed by applying the cost weight factor. With this criterion, it also turned out SVMM superior to SODI in this step. Now, S1 separated by the subsets S11 (DC(S1)<0) and S12 (DC(S1)≥0).

For the subset S11 in the superset S1, SODI found the best pair of attributes, '*Job*' and '*Purpose'*. According to SODI rules as shown in figure 5, SODI branched four internal nodes and two end leaves as follows:

1. SODI Rule 1: Clustered ('unskilled', 'life insurance') and ('unskilled', 'real estate') to classify as 'good' (33 instances are 'good', and 5 cases are 'bad'), and finalized.
2. SODI Rule 1: Clustered ('high qualified', 'life insurance') and ('unskilled', 'unknown') to classify as 'bad' (4 instances are all 'bad' credits), and finalized

3. SODI Rule 2: Separated ('skilled', 'car') to classify as 'good' (34 instances are 'good' credits, and 13 instances are 'bad' credits), but more works required
4. SODI Rule 3: Clustered ('high qualified', 'unknown'), ('skilled', 'unknown'), and ('skilled, 'real estate') to classify as 'good' (32 instances are 'good' credits, and 16 instances are 'bad' credits), but more works required
5. SODI Rule 3: Separated ('high qualified', 'car') to classify as 'bad' (3 instances are 'good' credits, and 6 instances are 'bad' credits), but more works required
6. SODI Rule 4: Clustered ('high qualified', 'real estate'), ('skilled', 'life insurance'), (all 'Job'='unemployed'), and ('unskilled', 'car') for uncompleted classification (24 instances are 'good' credits, and 24 instances are 'bad' credits)

From SVMM for the subset S11, there is no improvement by using support vector machines. Therefore, the subset S11 chose the result of SODI with the above 6 branches. Since the first two cases above the list are turned out as end leaves, we only branched at four internal nodes, and named the corresponding subsets as S113, S114, S115, and S116, respectively.

For the subset S113, SVMM performed better than SODI and was terminated by the pruned C4.5 decision tree with three attributes, '*Credit amount*', '*Installment commitment*', and '*Age*'. The total cost for the subset S113 is 21 (four Type-I errors and one Type-II errors). For the subset S115, SVMM perfectly classified by the decision function

$$DC(S115) = -0.133\ Duration + 1.599\ Residence\_since - 0.600. \quad (4.3)$$

For the cases of both subsets S114 and S116, the default costs are respectively 32 and 24 when we classify all cases in this branch as 'bad' so that there are only Type-II errors. When we applied SODI and SVMM for this subset, both methods paid more than these default costs. That means further branch will pay more. Therefore, we added a new rule for the case of 'cost-sensitive' problems:

IDSS Rule No. 1: If either SODI or SVMM pays more cost, then stop branching.

Therefore, these subsets were determined as end leaves with the class 'bad'.

Now, the IDSS recursive loops returned up to the subset S12 ($DC(S1)\geq 0$). In this subset, SVMM found a decision border that had better performance than SODI as follows

$$DC(S12) = 0.00117\ Duration + 0.541\ Residence\_since - 1.385. \quad (4.4)$$

From this split, S12 was separated by two subsets, S121 ($DC(S12)<0$) and S122 ($DC(S12)\geq 0$). For the subset S121, IDSS found the performance of SVMM was better than that of SODI. The subset S121 was branched by the attribute '*Present residence since*' with the condition '*Present residence since* ≤ 1(year)' or '*Present residence since* > 1(year)' to S1211 and S1212. In the subset S1211, there is no further improvement. Even though the number of 'good' credit applicant in S1211 is larger than that of 'bad' credit applicant, the total cost (65) for Type-I errors when the subset was treated as 'good' was much higher than that (32) of Type-II errors when the subset was treated as 'bad'. Therefore, this subset was turned out the end leaf with the class 'bad'. For the subset S1212, SODI found better spilt-conditions than SVMM by selecting the attribute '*Savings status*'. Further processing of IDSS from the subset S1212, it was found that the attribute '*Credit history*' contributed to the classification improvement. For the subset S122, SODI found the best combination of nominal attributes, '*Credit history*' and '*Other parties*', which performed better than SVMM. It clustered four conditions as follows

1. SODI Rule 1: Clustered ('*Credit history*=no credit/all paid'), ('all paid', 'none'), ('all paid', 'guarantor'), ('existing', 'co applicant') and ('delayed previously', 'guarantor') to classify as 'bad' (28 instances are 'bad', and 4 instances are 'good'), and finalized.
2. SODI Rule 1: Clustered ('all paid', 'co applicant'), ('existing', 'guarantor'), and ('delayed previously', 'co applicant') to classify as 'good' (only 1 instance is 'bad' among 12 instances), and finalized
3. SODI Rule 2: Clustered ('*Credit history*=critical/other existing credit'), which were dominated by class 'good' (41 instances are 'good' credits, and 17 instances are 'bad' credits), but more works required (Denote this subset as S1223).
4. SODI Rule 4: Clustered ('existing', 'none') and ('delayed previously', 'none') with the logic of 'OTHERWISE', and more works required (60 instances are 'bad', and 51 instances are 'good' denoting this subset as S1224).

In the subset S1223, SVMM found better performance than SODI by using the following decision function:

$$DC(S1223) = 0.04554\ Duration + 0.22388\ Existing\_credits - 1.60907. \quad (4.5)$$

This decision function separated S1223 into S12231 (in the case of DC(1223)<0) and S12232 (in the case of DC(1223)≥0). The subset 12231 has been finalized by SODI using the attributes '*Purpose*' and '*Employment*'. The subset 12232 has been also finalized by SVMM by using the numerical attributes '*Duration*' and '*Age*'. For the subset 1224, SODI found the better combination of attributes than SVMM did. See figure 26 for more detail branches.

Now, IDSS came back to the subset S2 ('checking>=200'). SVMM could not improve the classification in the subset, but SODI found that the pair of attributes '*Property magnitude*' and '*Present employment since*' could improve the penalty cost of both Type-I and Type-II errors. The decision ('*Property magnitude*=life insurance' or '*employment*>=7') classified 25 'good' credit applicants and misclassified 2 'bad' credit applicants (2 Type-I errors). The decision {('*Property magnitude*=car' and '*employment*= more than 1 year but less than 7 years ') OR ('*Property magnitude*=unknown' and '*employment*= unemployed')} determined to classify the 'good' credit approval with just one Type-I error among 19 instances. The decision 'OTHERWISE', which means all other combinations of '*Property magnitude*' and '*employment*' values, was turned out to be 'bad' credit approval with 11 Type-II errors among 22 cases. Therefore, the subset S2 paid for total cost 26.

For the subset S3, SODI found the best pair of attributes, '*Purpose*' and '*Other payment plans*', which performed better than SVMM. If the value of {'*Purpose*', '*Other payment plans*'} for each instance in S3 was ('new car', 'bank') or ('new car', 'stores') or ('education', 'bank') or ('education', 'stores') or ('business', 'bank') or ('business', 'stores'), then it was classified as a 'bad' credit applicant (total 11 Type-II errors among 27 cases). Otherwise, SODI classified all other instances as 'good' credit applicants (total 30 Type-I errors among 367 instances). So, the subset S3 paid for total 161 cost by applying SODI. There is no further improvement of the classification. Therefore, the recursive IDSS algorithm stopped here. The final solution of the German credit approval problems by using IDSS is as shown in Figure 26. The performance of IDSS has been evaluated with respect to the conventional methods such as Näive-Bayes, C4.5, PART, JRip, and SMO as shown in Table 10. Figure 27 also shows the graphs of performance evaluation results.

```
(0<check_status OR 0<=check_status<200):DC(S1)= 0.00397 * duration + 0.130 * installment_commitment - 0.420
| S11=(DC(S1)<0)
| | (job, purpose)=(unskilled, life insurance) OR (unskilled, real estate): good (38.0/5.0)
| | (job, purpose)=(high qualif, life insurance) OR (unskilled, no known): bad (4.0)
| | (job, purpose)=(skilled, car)
| | | credit_amount <= 1372: bad (4.0)
| | | credit_amount > 1372
| | | | installment_commitment <= 1: good (16.0/2.0)
| | | | installment_commitment > 1
| | | | | age <= 24: bad (6.0/1.0)
| | | | | age > 24: good (21.0/2.0)
| | (job,purpose)=(high qualif, no known) OR (skilled, no known) OR (skilled, real estate): bad (48.0/32.0)
| | (job,purpose)=(high qualif, car): DC(S115)=-0.133 duration + 1.599 residence_since - 0.600
| | | S1151=(DS(S115)<0): good (3.0)
| | | S1152=(DS(S115)>=0): bad (6.0)
| | (high qualif, real estate) OR (skilled, life insurance) OR (unemp/unskilled, *)
| | | OR (unskilled, car): bad (48.0/24.0)
| S12=(DC(S1)>=0):DC(S12) = 0.001171 * duration + 0.54148 * residence_since - 1.38531
| | S121=(DC(S12)<0):DC(S121) = residence_since - 1.5
| | | (DC(S121)<0) : bad (45.0/32.0)
| | | (DC(S121)>=0)
| | | | savings_status = <100: bad (56.0/18.0)
| | | | savings_status = 100<=X<500
| | | | | credit_history = no credits/all paid: bad (0.0)
| | | | | credit_history = all paid: good (1.0)
| | | | | credit_history = existing paid: bad (9.0/2.0)
| | | | | credit_history = delayed previously: good (2.0)
| | | | | credit_history = critical/other existing credit: good (1.0)
| | | | savings_status = 500<=X<1000: bad (1.0)
| | | | savings_status = >=1000: good (6.0/1.0)
| | | | savings_status = no known savings: bad (15.0/9.0)
| | S122=(DC(S12)>=0)
| | | (credit history, other_parties)= (no credit/all paid, *) OR (all paid, none) OR (all paid, guarantor)
| | | | OR (existing, co applicant) OR (delayed previously, guarantor): bad (32.0/4.0)
| | | (credit history, other_parties)= (all paid, co applicant) OR (existing, guarantor)
| | | | OR (delayed previously, co applicant): good (12.0/1.0)
| | | (credit history, other_parties)= (critical/other existing credit, *):
| | | | >>> DC(S1223) =  0.04554 * duration + 0.22388 * existing_credits - 1.60907
| | | | S12231=(DC(S1223)<0)
| | | | | purpose = new car
| | | | | | employment = unemployed OR <1 OR >=7: bad (8.0/3.0)
| | | | | | employment = 1<=X<4 OR 4<=X<7: good (5.0)
| | | | | purpose = used car OR domestic appliance OR other OR repairs OR : good (7.0)
| | | | | purpose = furniture/equipment OR radio/tv: good (18.0/3.0)
| | | | | purpose = education OR business OR vacation OR retraining: bad (5.0/3.0)
| | | | S12232=(DC(S1223)>=0)
| | | | | age <= 27: bad (5.0/1.0)
| | | | | age > 27
| | | | | | duration <= 45: good (7.0)
| | | | | | duration > 45: bad (3.0)
| | | (credit history, other_parties)= OTHERWISE): S1224
| | | | (employment, job)=(unemployed,*) OR (<1,*) OR (*,unemp/unskilled): bad (30.0/8.0)
| | | | (employment, job)=(1<=X<4, high qualify) OR (4<=X<7, unskilled) OR (4<=X<7, high qualify)
| | | | | : good (8.0/1.0)
| | | | (employment, job)=(>=7, high qualify): bad (4.0)
| | | | (employment, job)=(1<=X<4, unskilled): bad (11.0/6.0)
| | | | (employment, job)=(1<=X<4, skilled) OR (4<=X<7, skilled)
| | | | | credit_amount <= 3622
| | | | | credit_amount <= 888
| | | | | | credit_amount <= 652: good (2.0)
| | | | | | credit_amount > 652: bad (4.0/1.0)
| | | | | credit_amount > 888: good (17.0/2.0)
| | | | | credit_amount > 3622: bad (4.0)
| | | | (employment, job)=(>=7, unskilled) OR (>=7, skilled): bad (32.0/12.0)
(check_status>=200)
| (property_magnitude,employment)=(life insurance, *) OR (*, >=7):good (27.0/2.0)
| (property_magnitude,employment)=(car, 1<=X<4) OR (car, 4<=X<7) OR (no known, unemployed): good (19.0/1.0)
| (property_magnitude,employment)=OTHERWISE: bad (22.0/11.0)
(no checking)
| (purpose, other_payment_plans)=(used_car,*) OR (furniture, *) OR (radio/tv,*) OR (domestic equipment,*)
| | OR (repair, *) OR (vacation, *) OR (retraining, *) OR (other, *) OR (*, none): good (367.0/30.0)
| (purpose, other_payment_plans)=OTHERWISE: bad (27.0/11.0)
```

Figure 26. IDSS Classification of the German credit approval problem.

Table 10. Performance evaluation of IDSS vs. Naïve-Bayes, C4.5, PART, JRip, and SMO

Classification Methods	Training (A)				10-fold Cross Validation (B)				Cost difference between (A) & (B)
	Type-I Error	Type-II Error	Misclassi-fication	Total Cost	Type-I Error	Type-II Error	Misclassi-fication	Total Cost	
Naïve-Bayes	139	89	22.8 %	784	147	93	24.0 %	828	40
Pruning C4.5	176	38	21.4 %	918	201	73	27.4 %	1078	160
PART	56	47	10.3 %	327	153	129	28.2 %	894	567
JRip	109	78	26.9 %	1033	200	92	29.2 %	1092	59
SMO	142	74	21.6 %	784	159	91	25.0 %	886	102
IDSS	50	178	22.8 %	428	68	189	25.7%	529	101

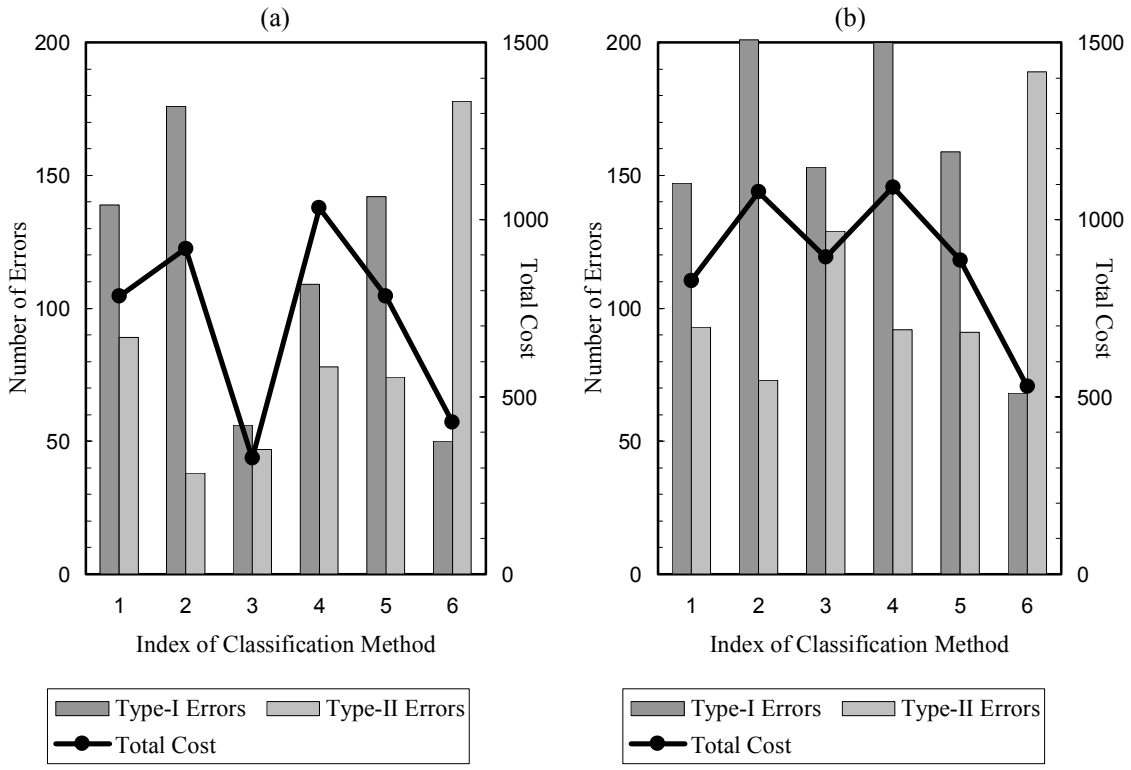


Figure 27. Performance evaluation of classification methods ([1] Naïve-Bayes, [2] C4.5, [3] PART, [4] JRip (Implementation of Ripper), [5] SMO (support vector machines), and [6] IDSS (Induction of Decision Tress with SODI and SVMM)): Both Type-I and Type-II errors as well as the total cost were drawn (a) from training results and (b) from 10-fold cross-validation results.

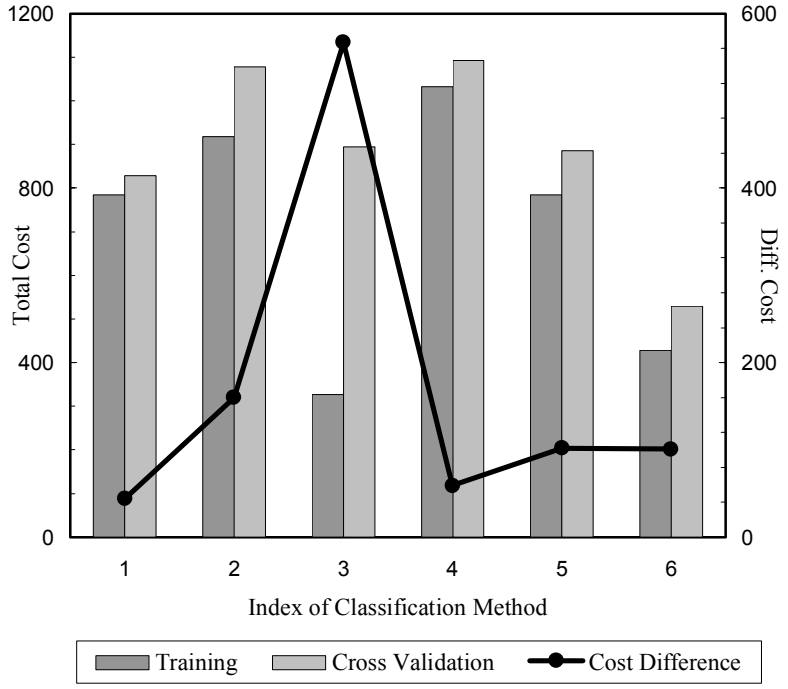


Figure 28. Comparison of cost evaluation between training and cross validation results for each classification method: [1] Naïve-Bayes, [2] C4.5, [3] PART, [4] JRip (Implementation of Ripper), [5] SMO (support vector machines), and [6] IDSS (Induction of Decision Tress with SODI and SVMM).

As shown in figure 28, IDSS provided the smallest cost from cross validation tests. It also provided relatively smaller gap of costs between training and cross validation results. With nonlinear or piecewise linear decision boundary description of both nominal and numerical attributes, we achieved an inexpensive classification model by using IDSS. Also, the error gap between training and cross validation testing in IDSS is only 2.9%, which is the third smallest gap and much smaller than the average of error gaps from all models. Other methods seemed not to have the consideration of cost factor while they were building a decision tree. Especially C4.5 and JRip generated much more expensive decision trees than others. According to Table 10, PART seemed not to have a pruning result since the expected cost has been extraordinarily increased more than any other method. We did not count on the model complexity in this section, but usually IDSS has more complex model descriptions than any others. This is one of most important counterparts in classification. This is what we are expected results since IDSS has been attempted to describe unknown decision boundaries more approximately. With this concept, several classification problems have been improved.

3. Case study B: classification of magnetic flux leakage (MFL) signals

Most countries employ a network of buried or aboveground pipelines to transport natural gas from production sites or dockyard terminals to consumer locations. The desire to maintain a reliable supply as well as safety considerations makes it necessary to inspect these pipelines periodically for damage caused by corrosion and other factors. This is typically accomplished by using a device called a pig, as shown at the left-handed side in figure 29, which is launched at one end of a line section and retrieved at the other end. The pig, which moves due to pressure exerted by the gas, contains all the necessary sensors, excitation sources and storage media to record information relating to the condition of the pipe as it moves along.

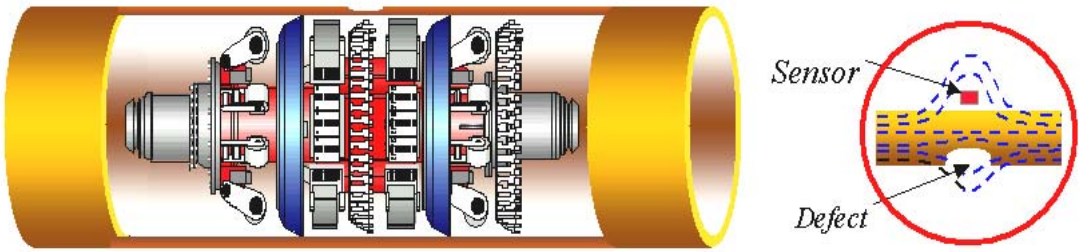


Figure 29. The flaw detection pig for gas pipeline inspection: Around a defect magnetic flux leakage signals are appeared as shown at the right-handed side of the above pictures.

Magnetic flux leakage (MFL) methods are widely employed for the nondestructive evaluation (NDE) of gas pipelines as shown at the right-handed side in figure 29. The inspection typically generates over 25 GB of compressed data for every 100 mile of pipeline inspected. The manual analysis of the large amount of data produced during the inspection can be both time-consuming and expensive. The gas industry is keenly interested in automating the interpretation process. Key advantages of the automation process include improvement in the accuracy, speed and consistency of interpretation. The magnetic leakage fields around defects are measured using a circumferential array of sensors. The sensor signals are appropriately sampled and stored for off-line analysis. Figure 30 shows a typical of magnetic flux leakage (MFL) around a defect.

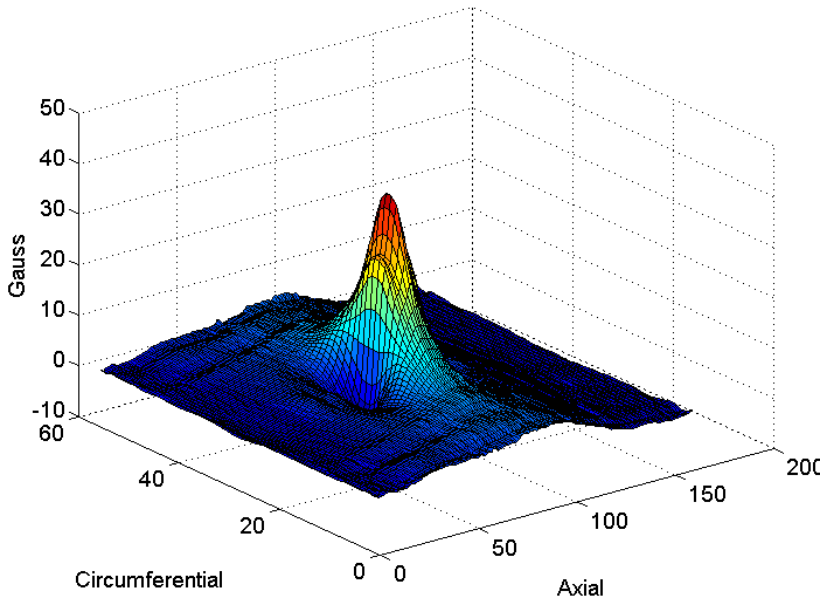


Figure 30. Typical magnetic flux leakage signals acquired during gas pipeline inspection.

An indication consists of any signal in the MFL data caused by benign pipeline artifacts (valves, welds, tees, flanges, etc.) or defects (small, medium or large metal corrosion, flaw on welding area, etc.). The indication extraction process involves the determination of the location and the size of MFL signals in the data. Indication extraction (feature extraction or attribute representation) serves a dual purpose; first, it provides significant data compression, since only signals detected in this process are used for further analysis, and second, it allows for a means to organize a database of signal properties for use by a classification network or tree. Many techniques for data analysis can be regarded as seeking for a description of data in terms of elementary features. An advantage of a feature representation is that it reduces redundancy in the input patterns (Barlow, 1989). Furthermore, a description in terms of features can provide a lucid explanation of objects (input patterns), which can in addition be helpful in understanding the hidden data generating process.

We compute twenty different attributes for each indication for building the MFL database as shown in Table 11. They included various physical and statistical properties of the MFL signal indications. For instance, the first parameter used in feature representation is the geometric shape of the indication. This determines whether the feature covers the

complete circumference of the pipe or not. This simple and always distinguishable parameter discriminates between features like welds, flanges and valves, which always encompass the complete circumference of the pipe, and defects, which are usually smaller in size. We partitioned the original classification problems into six different subset of independent classification problem according to a priori knowledge from MFL signal characteristics for signal classification. This categorization came from C4.5 deductive inference based on an extensive MFL database.

Table 11. Feature representation of acquired indications for MFL signal classification

A1	Feature shape (geometry) in 2D projection (nominal)	Vertical / Elliptical
A2	Amplitude at the center of an indication object (MFL pattern)	Up / Down / Both
A3	Axial size in inch	Numeric
A4	Circumferential size in degree	Numeric
A5	Exists different size of indication object in axial direction	Yes / No
A6	The size of area of MFL signals higher than background	Numerical
A7	The size of area of MFL signals lower than background	Numerical
A8	Average of positive values of MFL signals	Numerical
A9	Average of negative values of MFL signals	Numerical
A10	The 'peak-to-peak' value of signals	Numerical
A11	The peak value at the center of signals	Numerical
A12	Maximum value of MFL signals	Numerical
A13	Clock position of the center of signals	Numerical
A14	Orientation of the feature (0-360 degree)	Numerical
A15	Number of vertical shapes of signals (Integer)	Numerical
A16	Background signal difference for pipeline transition (changes)	Numerical
A17	2nd Moment parameter 1 $\varphi 1 = (m_{20} + m_{02}) / m_{00}$	Numerical
A18	2nd Moment parameter 2 $\varphi 2 = \sqrt{(m_{20} - m_{02})^2 + 4m_{11}} / m_{00}$	Numerical
A19	3rd Moment parameter 3 $\varphi 3 = \sqrt{(m_{30} - 3m_{12})^2 + (3m_{21} - m_{03})^2} / m_{00}^{3/2}$	Numerical

A20	3rd Moment parameter 4 $\varphi 4 = \sqrt{(m_{30} + m_{12})^2 + (m_{21} + m_{03})^2} \,/\, m_{00}^{3/2}$	Numerical

Lee et al. (2000) introduced a hierarchical multi-layered perceptron (HMLP) neural network, as shown in Figure 31, which is constructed by the combination of a predefined hierarchical structure of a decision tree and multi-layered perceptron as the learning module for each end-node of this decision tree. Figure 32 shows how to find best attributes for each category, and Table 12 shows the final maximum possible combination of attributes for each category. The variable X(i) in Table 12 indicates the input at i^{th} input node of a sub-neural network system in the hierarchical structure of multi-layered perceptron (HMLP). For example, X(2) in the category I column in the table indicates that the 2nd input node in the neural network corresponding to category I contains the circumferential width of the feature.

The classification analysis is performed in various stages. For the classification of MFL signals in this section, we consider to apply three methods as follows:

1. A single multi-layered perceptron (SMLP) with twenty attributes.
2. A hierarchical structure of six different multi-layered perceptrons (HMLP) with ten independent attributes according to a priori knowledge from the characteristics of MFL signal.
3. IDSS (Induction of Decision trees with SODI and SVMM) with twenty attributes.

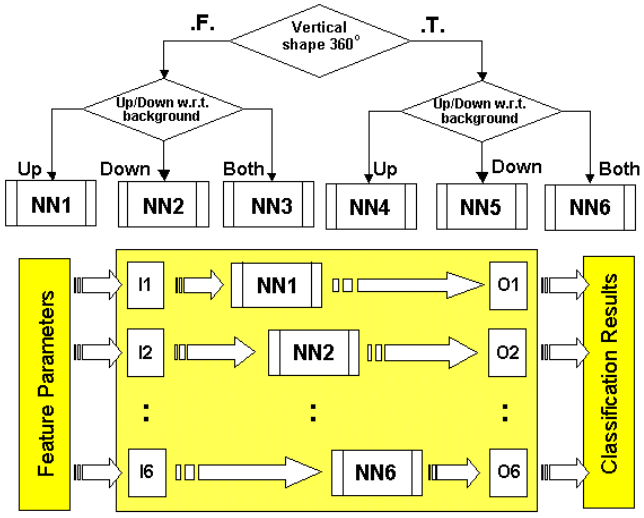
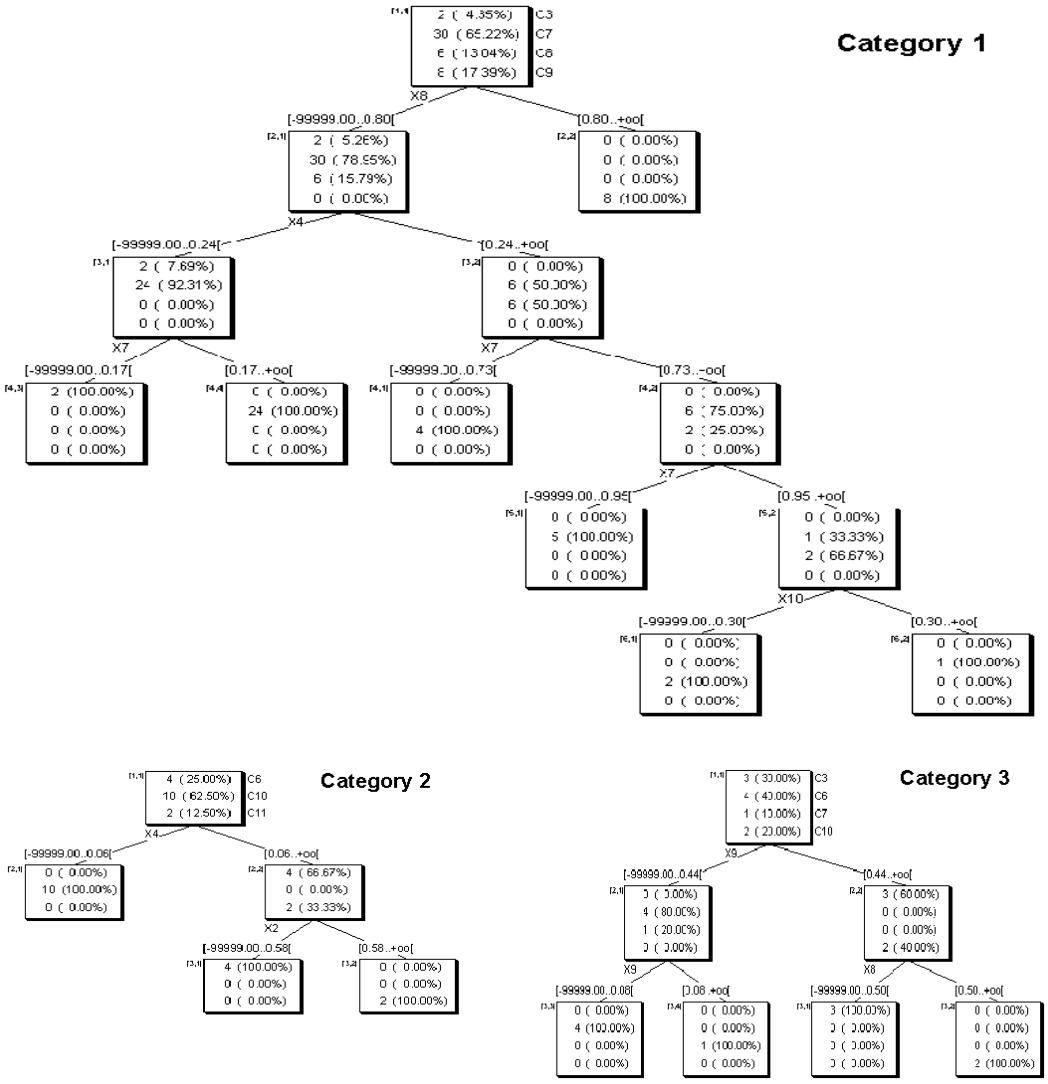


Figure 31. Architecture of the HMLP (hierarchical multi-layered perceptrons) classification neural network



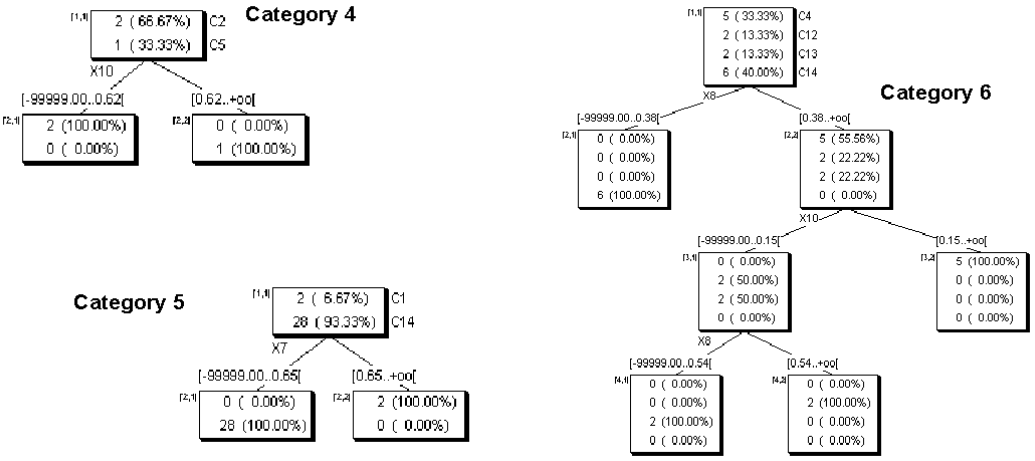


Figure 32. Attribute evaluation for hierarchical multi-layered perceptrons (HMLP).

Table 12. Feature representation scheme for HMLP

Feature Parameters	Category I	Category II	Category III	Category IV	Category V	Category VI
[1] Geometry	0	0	0	1	1	1
[2] MFL patterns	1/2	-1/2	0	1/2	-1/2	0
[3] Axial Length	X(1)	X(1)	X(1)	X(1)	X(1)	X(1)
[4] Circum. Width	X(2)	X(2)	X(2)	1	1	1
[5] Diff(axis-length)	0	0	0	X(2) (0/1)	(0/1)	X(2) (0/1)
[6] Area (+) signals	X(3)	0	X(3)	X(3)	0	X(3)
[7] Area (-) signals	0	X(3)	X(4)	0	X(3)	X(4)
[8] Average of (+)	X(4)	0	X(5)	X(4)	0	X(5)
[9] Average of (-)	0	X(4)	X(6)	0	X(4)	X(6)
[10] Peak-to-Peak	X(5)	X(5)	X(7)	*	*	*
[11] Peak (center)	X(6)	0	0	0	0	0
[12] Max. Signal	X(7)	0	0	1	1	1
[13] Clock position	0	X(6)	X(8)	0	0	0
[14] Orientation	0	0	0	*	X(5)	X(7)
[15] N(Vert.Shapes)	0	0	0	X(5)	0	*
[16] Bkgrnd Change	0	0	0	X(6)	X(6)	X(8)
[17] Moment. φ1	X(8)	X(7)	X(8)	X(7)	X(7)	X(8)
[18] Moment. φ2	X(8)	X(8)	*	X(8)	X(8)	*
[19] Moment. φ3	*	X(8)	*	X(8)	X(8)	*

[20] Moment. φ4	X(10)	X(10)	X(10)	X(10)	X(10)	X(10)

Another consideration of this MFL signal classification is to consider the minimization of both Type-I error and Type-II error. Type-I error is defined as the probability that a defect is classified as a non-defect class (i.e., the probability of failure to detect a defect). On the other hand, Type-II error is defined as the probability that a non-defect feature is classified as a defect class (i.e., the probability of false alarm). Because of very huge number of indications in MFL signal database, a human operator or expert was not able to verify all classified indications after obtaining automatic classification results. However, he was able to check all indications that are classified as defects. If a classification model misclassifies a defect, the human operator cannot protect a serious problem such as gas leakage under the ground or explosion of gas in future. Therefore, the Type-I error is more important than Type-II error. If a system classify all features as a defect class, the Type I error is zero, but there are too many false alarm occurred. That means the human operator may classify all indication manually rather than depend on the classification system. The best solution of this classification is to find the minimum Type-II error while the Type-I error keep zero. However, generally speaking, it is almost impossible to keep Type-I error be zero. Most acceptable way for this problem is to define cost matrix as mention in the previous section. Initially we set up the cost factors for Type-I error and Type-II error as 0.9 and 0.1, respectively. We collected the dataset of MFL signals obtained by real experiments (Lee et al., 2000), which consists of 14 classes and 20 attributes as shown Table 13. According to a priori knowledge of MFL signals, these six categories already partitioned subsets of classes.

Table 13. Data collection for the MFL signal classification with 4 types of defects and 10 types of artifacts

Classes	**SMLP or IDSS**	Category I	Category II	Category III	Category IV	Category V	Category VI
1. Anchor	16					16	
2. Abnormal Weld	22				22		
3. Pipeline Bend	24	8		16			
4. Flaw on Weld[3R]	32					7	25

5. Flange	17				17		
6. Metal Close Proximity	108		54	54			
7. Small Metal Loss[ℜ]	82	74		8			
8. Medium Metal Loss[ℜ]	76	76					
9. Large Metal Loss[ℜ]	48	48					
10. Spring Weld	13		7	6			
11. Tap	34		21	13			
12. TRHL[ℑ]	24						24
13. TRLH[Ψ]	24						24
14. Weld	36					28	8
Number of classes in group		4	3	5	2	3	4

[ℑ] TRHL: Pipeline Transition (from thin pipeline to thick pipeline)
[Ψ] TRLH: Pipeline Transition (from thick pipeline to thin pipeline)
[ℜ] These classes are defect types in which we are interested, and any others are artifacts (total 532 instances).

Table 14. Performance evaluation of IDSS comparing with both SMLP and HMLP

Evaluation Factors	Type-I Error		Type-II Error		Misclassification		Cost[ℜ] for Type-I & Type-II
	Average	St. Dev.[ℑ]	Average	St. Dev.[ℑ]	Average	St. Dev.[ℑ]	
BPNN	5.6 %	1.2 %	12.0 %	2.6 %	31.4 %	12.3 %	6.3 ± 2.7 %
HMLP	0.9 %	0.3 %	2.8 %	0.6 %	6.0 %	2.4 %	1.1 ± 0.7 %
IDSS	1.1 %	0.4 %	2.4 %	0.6 %	4.3 %	1.8 %	1.3 ± 0.8 %

[ℑ] 'St. Dev.' means here the standard deviation from 10-fold cross-validation sampling

[ℜ] The average cost for both Type-I error and Type-II error was computed by the weighted sum with weights 0.9 and 0.1, respectively. The ranges of these average costs were calculated with 95% confidential limit.

The classification results obtained from the two neural networks (SMLP and HMLP) and our IDSS are shown in Table 14. It showed that either HMLP or IDSS reduced both Type-I and Type-II errors considerably smaller than SMLP neural networks when we set up cost factors for Type-I and Type-II errors as 0.9 and 0.1, respectively. For the comparison of IDSS with HMLP, it is very hard to say which one is better performed. However, HMLP has the critical assumption that it is require a priori knowledge for making category clustering. Therefore, this assumption may not easy to apply for general classification problems. The basic idea of HMLP was how much we can improve the prediction accuracy when we

combine a TDIDT method like C4.5 (for providing both highly qualified clustering schemes and better understandability) and artificial neural networks (for more flexible nonlinear decision boundary descriptions). Therefore, it was not exactly verified for overfitting problems. That is, the hierarchical structure of HMLP may be changed by a new collection of dataset. However, IDSS does not require any priori knowledge for classification dataset. Therefore, IDSS is more available to solve any classification problems than HMLP. Figure 33 shows more detailed classification results from cross-validation test for each method.

We changed the cost factors (or cost matrix) such that the cost ratio for Type-I and Type-II errors is to be 0.7:03, 05:0.5, or 0.3:0.7. Figure 34(a) shows the sensitivity analysis of Type-I error for each classification method, and figure 34(b) shows the sensitivity analysis of Type-II error, respectively: i.e., figure 34(a) shows how much percent of defects we may miss, and figure (b) tells us how much percent of false alarm is in data classified as defect.

SMLP

Actual Class \ Classified As	1	2	3	4	5	6	7	8	9	10	11	12	13	14	sum
1	4			12											16
2		6		16											22
3			13	4		3	4								24
4		8		22				1	1						32
5		2		1	14										17
6						89	9								108
7			16				56	8	2						82
8			6				6	63	6						76
9								6	42						48
10			6							28					13
11											34				34
12								2				4		18	24
13								4					4	16	24
14				6										30	36

HMLP

Actual Class \ Classified As	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Sum
1	16														16
2		22													22
3			17				5			2					24
4				29										3	32
5					17										17
6						100	2			3	3				108
7						1	77	2	2						82
8			1				8	64	3						76
9								2	46						48
10						1				12					13
11				3			1				30				34
12												23		1	24

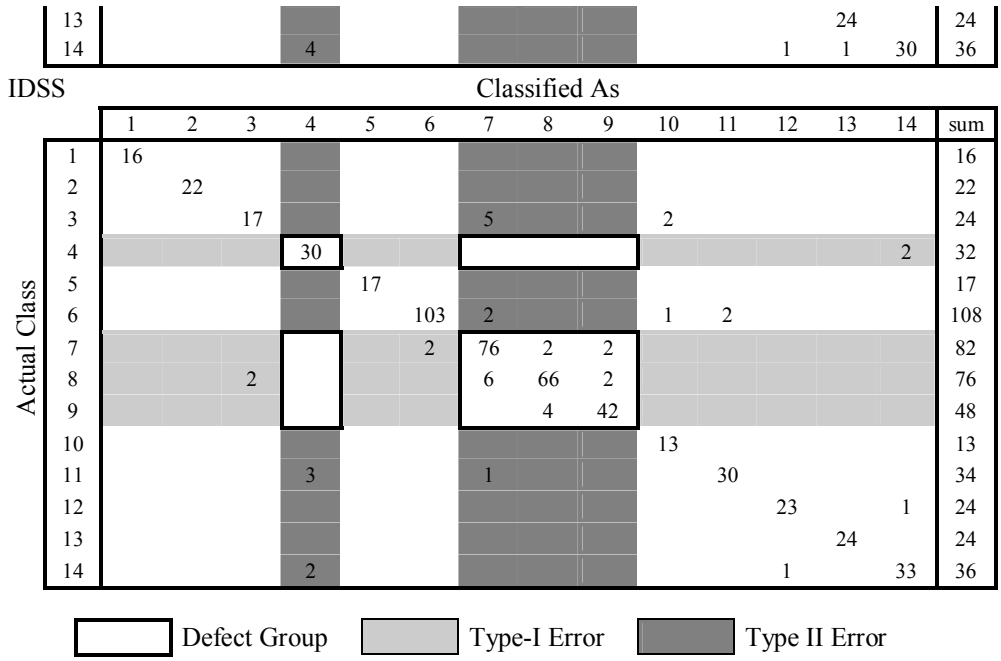


Figure 33. Cross validation test for SMLP, HMLP and IDSS

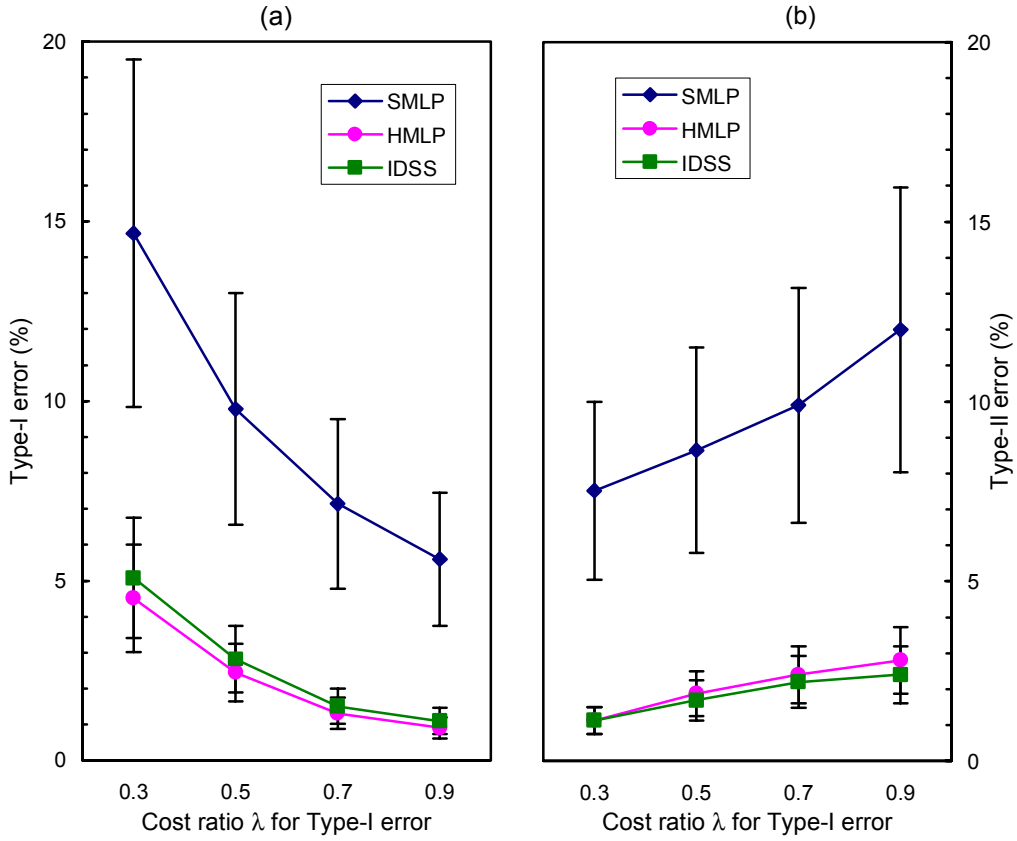


Figure 34. Sensitivity analysis of cost factor for MFL classification: (a) varying Type-I errors for each method with the cost ratio λ, and (b) varying Type-II errors for each method with the cost ratio (1-λ)

Conclusions

It showed that either HMLP or IDSS reduced both Type-I and Type-II errors considerably smaller than SMLP neural networks when we set up cost factors for Type-I and Type-II errors. As shown in figure 34, the performance between HMLP and IDSS did not significantly different. For the viewpoint of computational efforts, IDSS was much cheaper than HMLP since HMLP required certain times of weight matrix multiplications to find the minimum mean square root of weight errors, but IDSS did not require such computation of matrix multiplications. Furthermore, HMLP required a priori knowledge for making category clustering before applying sub-systems of MLP. However, this assumption may be not available for any classification problem. Therefore, IDSS is more recommendable for applying general classification problem rather than multi-layered perceptrons like SMLP as shown in figure 34.

4. Summary

The algorithm of IDSS has been introduced in this chapter. We did not compare the model complexity of IDSS wit other methods in this chapter, but showed how IDSS could improve the total cost when one set up any cost factors. One of most important ideas in IDSS is how much IDSS can approximately describe unknown decision boundaries. For nominal attributes, IDSS can describe second-order of decision-makings by using SODI algorithm. Similarly for numerical attributes, IDSS can describe unknown decision boundaries piecewise linearly or linearly oblique by using SVMM as mentioned in chapter 3. This is what the convectional decision trees such as C4.5 and JRip could not do. Also, IDSS shows which attributes are more important than others according to the advantage of TDIDT algorithms that any support vector machines cannot achieve. It also has better visualization support than support vector machines since IDSS strictly limit the maximum number of numerical attribute for each decision node up to three. Therefore, the decision boundary functions can be displayed on 2D or 3D spatial domains.

IDSS provided the smallest cost from cross validation tests in the German credit approval problem. It also provided relatively smaller gap of costs between training and cross validation results. With nonlinear or piecewise linear decision boundary description of both nominal and numerical attributes, we achieved the most inexpensive classification model by using IDSS. Especially C4.5 and JRip generated much more expensive decision trees than others. According to Table 10, PART seemed not to have a pruning result since the expected cost has been extraordinarily increased more than any other method.

It also showed how much IDSS could improve the classification opportunity cost with respect to any conventional (artificial) neural networks or hybrid (hierarchical) neural networks for the classification of MFL signals acquired from natural gas pipelines. Either HMLP (Hierarchical Multi-Layered Perceptron; Lee et al., 2000) or IDSS reduced both Type-I and Type-II errors considerably smaller than just conventional single neural networks with respect particular cost factors. We found that the performance between HMLP and IDSS did not significantly different, but IDSS provided computationally cheaper costs than HMLP did since HMLP required certain times of weight matrix multiplications to find the minimum mean square root of weight errors. Furthermore, HMLP requires a priori knowledge for making category clustering before applying sub-systems of MLP. However, this condition may be not available for any time to solve a classification problem.

In general IDSS can generate better accuracy of estimate prediction than conventional C4.5, especially when unknown decision boundaries may be very complicated or nonlinear. There is general rule: nonlinear problems must be solved by nonlinear solvers. C4.5 is a kind of linear solvers that only consider univariate decision-makings. If a problem is much more complex than univariate descriptions, then C4.5 may result in overfitting problems or unsatisfactory prediction accuracy. In this case IDSS has more possibility to improve both overfitting problems and prediction accuracy. That is, the classification result from IDSS is more reliable and better predictable than that from C4.5.

CHAPTER 5. CONCLUSION AND DISCUSSION

In this chapter we summarize all the work of this thesis and present some important issues for further research. We developed three classification methods: SODI (Second Order Decision-tree Induction), SVMM (Support Vector Machines for Multi-category), and IDSS (Induction of Decision trees using SODI and SVMM). Each method has its own limitations and characteristics of applications, and each of those is summarized in the next section.

1. Thesis summary

In the first chapter we introduced the definition of data mining and knowledge discovery in databases and described the important issues and research fields in data mining. The methods of classification problems in data mining area have been introduced in this thesis, too. Also, we introduced important issues for classification problems. We also addressed the concept of model complexity and overfitting problems. The reliability of classification models is highly related to overcoming *overfitting* problems. The *overfitting* problem causes from unnecessary decision-makings built by a training set. If a model fits the data exactly, it is almost impossible to predict unexpected or untrained test data correctly. Therefore, both the variance of model reliability and the bias of prediction accuracy should be considered as penalty factors of model selection criteria.

The model complexity in classification research area can be describe by how many essential attributes can describe unknown decision boundaries by which way, i.e., by linear, by nonlinear, by kernel functions, or by piecewise linear segments. For example, C4.5 (Quinlan, 1993) describes unknown decision boundaries formed by a set of orthogonal partitions with univariate attributes (attribute-value manner). For an oblique decision tree (Murthy et al., 1994a), the decision boundaries can be described by linear borders with multiple decision variables. However, It still has very important weakness: it builds all oblique decision borders to be parallel. How many cases of classification problems have the decision boundary must be parallel? So, the construction of a decision tree using support vector machines (Bennett, 1994-1997) is more reasonable to apply general classification

problems whose unknown decision borders are described by piecewise-linear segments. It also could not distinguish more important parameters or attributes from probably unnecessary ones.

To build a decision tree with nominal attributes in recent years, there is very little research for the consideration of nonlinear or multivariate decision boundary description. With this limited capability, it has always an overfitting risk. In this thesis, three new methods of TDIDT (top-down induction of decision trees) have been introduced to reduce the overfitting problems while the model complexity is not seriously increased. For only nominal attribute cases, a second-order decision-tree induction method (SODI) has been developed, and, for numerical cases, a new algorithm of support vector machines for multi-category classification (SVMM) was described. For a general case of both nominal and numerical attributes, IDSS (Induction of Decision trees with SODI and SVMM) was introduced. In both applications the policy of the model selection is to minimize the penalized (or, generalized) risk function, which is the weighted sum of all penalty costs.

In the first section of the second chapter the information entropy, or Shannon's entropy, in information theory, was introduced. Also, the concept of mutual information and its properties was described. Based on these properties techniques for eliminating redundant nominal attributes was developed and verified in the second section. This approach can be extended for feature selection and feature cleaning. For example, suppose three attributes are linearly dependent on each other, and one of them is redundant. Then, one can compute the gain ratio for each attribute, and using this information, the redundant attribute that has the smallest gain ratio should be removed for further data mining. Furthermore, it can be applied for our SODI algorithm. Since SODI requires computing all pairs of two attributes at the worst case, removing redundant attribute promises more efficient computation.

In the third section of the second chapter, a new TDIDT algorithm called SODI was introduced for any classification problems with nominal attributes only. In general SODI obtains high quality of classification results compared to other univariate TDIDT methods, such as C4.5 and PART. However, without pruning, SODI may generate some overfitting problems. Therefore, a pruning SODI was introduced and evaluated. With pruning, SODI

generated the smallest decision trees for almost every problem, as well as more stable prediction accuracy. Using SODI the hypothesis description (or decision-makings) for each decision node becomes more complex, but the size of a decision tree by the SODI method is much smaller than any conventional univariate decision trees. It works effectively when some of decision attributes are significantly correlated so that the joint distribution of the class attribute with these attributes is not linearly independent. A numerical analysis from nine well-known classification problems was performed to compare SODI to other algorithms of univariate decision trees.

The major limitation of SODI, namely only being applicable for nominal attributes only, is addressed by the SVMM algorithm. The benefits of SVMM with respect to either C4.5 or PART came from how to describe the decision boundaries. C4.5 treats the classification boundaries as axis-orthogonal (if we assume any numerical attribute as an axis), so that every decision areas may be shaped as rectangle, cubic, or hyper-cubic geometries. However, SVMM can generate more flexible convex subspace consisting of two- or three-dimensional convex subspace from some subsets of numerical attributes for the decision tree branch. This flexibility can result in the improvement of model complexity as well as resolution of overfitting problems.

There is a trade-off between model complexity and prediction accuracy for most data mining and classification problems. For example, the problem of either 'Ionosphere' or 'Sonar' problem requires a huge number of attributes to determine the classification boundary, even though it provided smaller error gaps between training and cross-validation than either C4.5 or PART. The SVMM method took some advantages between C4.5 and conventional SVM. As a benefit of C4.5, SVMM can identify which attributes are more important than any others for purposes of easy to understand. As a benefit of SVM, it can describe unknown decision boundaries more flexibly than C4.5. With the combination of C4.5 and SVM, SVMM can describe the decision boundaries with piecewise linear segments. Therefore, SVMM has more chance to solve linearly inseparable problems determined by SVM. Also, SVMM provided a transformation function for the following important issues: (1) any nonlinear decision boundaries for the classification can be described as piecewise-

linear segments, (2) transforming numerical variables to a nominal attribute allows to apply many other TDIDT methods such as C4.5, PART, SODI, AdaBoost, etc., and (3) it is able to compare information gain ratio between the original nominal attribute and newly converted nominal attributes (subspaces of numerical space). It means that, with SVMM, it is able to combine numerical attributes with nominal attributes by the measurement of information gain ratio.

The algorithm of IDSS was introduced in chapter 4. We did not compare the model complexity of IDSS with other methods in this chapter, but we showed how IDSS could improve the total cost when one set up arbitrary cost factors. One of most important ideas in IDSS is how much IDSS can approximately describe unknown decision boundaries. For nominal attributes, IDSS can describe second-order of decision-makings by using SODI algorithm. Similarly for numerical attributes, IDSS can describe those decision boundaries by piecewise linear segments or linearly oblique borders by using SVMM.

IDSS also provided which attributes were more important than others as an advantage of TDIDT algorithms that any conventional support vector machines could not achieve. It also has better visualization opportunity than support vector machines since IDSS strictly limit the maximum number of numerical attributes for each decision node up to three, so that the decision boundaries can be displayed on 2D or 3D space.

In the fourth chapter, IDSS has been compared not only conventional TDIDT algorithms of decision trees or decision rules, such as C4.5, JRip, and PART, but also mathematical optimization methods, such as support vector machines and artificial neural networks. According to the evaluation, IDSS performs better than other conventional methods with respect to the classification cost factors (for example, both Type-I and Type-II errors).

2. Discussion

We did not mention about meta-knowledge (knowledge of knowledge) or any combinations of classification algorithms in this thesis. AdaBoost, which classifies for boosting a classifier using AdaBoost-M1 method (Freund and Schapire, 1996), is an example of meta-knowledge. AdaBoost-M1 builds several solutions of a classification problem, and classifies any instances by voting the results from all models. It is possible to compare IDSS with several combinations of AdaBoost-M1. Also, it is very possible to evaluate several experimental examples for the comparison of IDSS with respect to other methods, but we did not process it in this thesis.

We also did not present the pruning of IDSS in this thesis. It will be one of the future works. Also, we are considering about visualization of multiple attributes limited up to 3D space. The visualization in data mining become more important because it is more attractive for potential customers to pay attentions what we want to do. According to the IDSS algorithm, we did not consider on the use of both nominal and numerical attributes at the decision node (or internal node) at the same time. If it is possible that any nominal attributes can be ordered, then we can apply nominal attributes into support vector machines or SVMM. We found there are a lot of ordered nominal attributes in the German credit approval problems. On the other hand, it is possible to segment the spatial space of numerical attributes into subsets of partial convex space of them in order to transform numerical values to the values of newly created nominal attributes. Then, we can apply the SODI method for purposes of combining the original nominal and numerical attributes. However, too early transform of numerical attributes to nominal ones may be lost more chance of the improvement of prediction accuracy because, while a decision tree is constructed, predefined or transformed nominal attributes from numerical space may be not important or lost their flexibility. In general, with more appropriate conversion of class description attributes, one can achieve more simple and accurate prediction results.

For general classification problems we suggested to use our IDSS method, but I did not mention that our method can be always better than any others. For the classification

problem that has unknown but potentially very simple decision boundaries, either C4.5 or PART may perform better than IDSS. It implies that any linear problem can be better solved by linear solvers than any nonlinear solvers. Since one cannot expect how difficult a classification problem is, we recommend the use of multiple classification methods. Very fast computational advances in recent years allow us to use several methods without paying extra expensive.

BIBLIOGRAPHY

Ali, K. M. and M. J. Pazzani (1995a), "HYDRA: A nose-tolerant relational concept learning algorithm," *International Joint Conference on Artificial Intelligence*, Chambery, France

Ali, K. M. and M. J. Pazzani (1995b), "Reducing the small disjuncts problems by learning probabilistic concept descriptions," *Computational Learning Theory and Natural Learning Systems*, Vol. 3, pp.183-199

Les Atlas, R. Cole, Y. Muthuswamy, A. Lipman, J. Connor, D. Park, M. El-Sharkawi, and R. J. Marks Ii (1990), "A performance comparison of trained multilayer perceptrons and trained classification trees," *Proceedings of the IEEE*, **78**(10), pp.1614-1619.

Attias, H. (1999), ''Independent factor analysis,'' *Neural Computation*, 11(4), pp.803-851.

Barlow, H. B. (1989), ''Unsupervised learning,'' *Neural Computation*, 1(3), pp.295-311.

Baudat, G. and F. Anouar, (2000) "Generalized Discriminant Analysis Using a Kernel Approach", *Neural Computation* 12, 2385-2404

Bell, A. J. and T. J. Sejnowski, (1995), ''An information-maximization approach to blind separation and blind deconvolution,'' *Neural Computation*, 7(6), pp.1129-1159.

Ben-Bassat, M. (1987), "Use of distance measures, information measures and error bounds on feature evaluation," *Classification, Pattern Recognition and Reduction of Dimensionality*, *Volume 2 of Handbook of Statistics*, Krishnaiah and K. Paruchuri, R. Krishnaiah and L. N. Kanal, eds, North-Holland Pub., Amsterdam, pp. 773-791.

Bennett, K. P (1994), "Global tree optimization: a non-greedy decision tree algorithm," *Computing Science and Statistics*, Vol. 26, pp.156-160

Bennett, K. P (1992), "Decision tree construction via linear programming," *Proceedings of the fourth Midwest Artificial Intelligence and Cognitive Science Society Conference*, Utica, IL, pp.97-101

Bennett, K. P, and J. A. Blue (1997), *A support vector approach to decision trees*, R.P.I Math Report No. 97-100, Rensselaer Polytechnic Institute, Troy, NY 12180

Bennett, K. P, and J. A. Blue (1996), *Optimal decision trees*, R.P.I Math Report No. 214, Rensselaer Polytechnic Institute, Troy, NY 12180

Bennett, K. P., and O. L. Mangasarian (1994), "Multi-category Discrimination via Linear Programming," *Optimization Methods and Software*, Vol. 3, pp.29-39

Bennett, K. P., and O. L. Mangasarian (1992), "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets," *Optimization Methods and Software*, Vol. 1, pp.23-34

Bioch, J. C., O. van der Meer, and R. Potharst, "Bivariate Decision Trees", pp. 232-243 of J. Komorowski, J. Zytkow, eds. *Principles of Data Mining and Knowledge Discovery*, *Lecture Notes in Artificial Intelligence 1263*, Springer Verlag, 1997.

V. Blanz, B. Schölkopf, H. Bülthoff, C. J. Berges, V. Vapnik (1996), and T. Vetter, "Comparison of view-based object recognition algorithms using realistic 3D models," *Artificial Neural Networks: ICANN'96*, Berlin, Vol. 1112, pp. 251-256.

Blockeel H., L. De Raedt, and L. Ramon (1998), "Top-down induction of first order logical decision trees," *Artificial Intelligence*, Vol. 101, pp. 258-297

J. Bredensteiner and Bennett, K. P (1999), *Computational Optimizations and Applications*, Vol. 12, pp. 53-79.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984), *Classification and Regression Trees*, Wadsworth International Group

Brodley, C. E. and P. E. Utgoff (1995) "Multivariate Decision Trees," *Machine Learning*, Vol. 19, pp.45-77.

Brodley C. E., P. E. Utgoff (1992), *Multivariate versus univariate decision trees*, Coins Technical Report 92-8, Amherst, MA, University of Massachusetts, Department of Computer and Information Science

Brown, D. E. and C. L. Pittard (1993), "Classification Trees With Optimal Multivariate Splits," *In Proceedings Of The International Conference On Systems, Man and Cybernetics*, Vol. 3, pp. 475-477, Le Touquet, France, October 1993

D. E. Brown, V. Corruble, and C. L. Pittard (1993), "A comparison of decision tree classifiers with backpropagation neural networks for multimodal classification problems," Pattern Recognition, **26**(6), pp. 953-961.

Brown, D. E. and C. L. Huntley (1991), "A Practical Application of Simulated Annealing to Clustering," *Technical report LPC-TR-91-003*, University of Virginia, 1991

Buntine, W. (1992), "Learning classification trees," Statistics and Computing, Vol. 2, 63--73.

C. J. Berges (1996), "Simplified support vector decision rules," *Proceedings of the thirteenth International Conference on Machine Learning*, Bari, Italy, pp. 71-77.

B. Cestnik and I. Bratko (1991), "On estimating probabilities in tree pruning," *Machine Learning – EWSL-91. European Working Session on Learning Proceedings*, pp. 138-150. *Ed.*, Kodratoff, Spinger-Verlag.

Cios, K. J. and N. Liu (1992) "A machine learning method for generation of a neural network architecture: A continuous ID3 algorithm," *IEEE Transactions on Neural Networks*, 3(2), pp.280--291.

W. W. Cohen (1995), "Fast Effective Rule Induction," *Proceedings of the twelfth International Conference on Machine Learning*., Lake Taho, CA, Morgan Kauffman.

Crawford, S. L. (1989), "Extensions to the CART algorithm, "International Journal of Man-Machine Studies, 31(2), pp.197-217

DAlché-Buc, F., D. Zwierski, and J.-P. Nadal (1994), "Trio Learning: A new strategy for building hybrid neural trees," *International Journal of Neural Systems*, 5(4) pp.259-274.

Dietterich, T. G. and E. B. Kong (1995), "Machine learning bias, statistical bias and statistical variance of decision tree algorithms," *Machine Learning: Proceedings of the 12th International Conference*, Tahoe City, CA, 1995

H. Drucker, C. J. Burges, L. Kaufmann, A. Smola, and V. Vapnik (1997), Support vector regression machines," *Advances in Neural Information Processing Systems*, Vol. 9, pp.155-161.

Duda, R. O. and P. E. Hart (1973), *Pattern Classification and Scene Analysis,* John Wiley

DuMouchel, W., Volinsky C, Johnson T, Cortes C, Pregibon D (1999) Squashing flat files flatter. In *Proceedings of the Fifth ACM Conference onKnowledge Discovery and Data Mining*,6-15.

U. Fayyad, S. G. Djorgovski, and N. Weir (1996a), "Automating the analysis and cataloging of sky surveys," *Advances in Knowledge Discovery and Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Ed., pp. 471–493. AAAI/MIT Press.

U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (1996b). *Advances in Knowledge Discovery and Data Mining*, MIT Press, Cambridge, MA, 1996

U. Fayyad and K. B. Irani (1990), "What should be minimized in a decision tree," In AAAI-90: PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, Vol. 2, pp. 749-754.

C. Feng, A. Sutherland, R. King, S. Muggleton, And R. Henery (1993), "Comparison of machine learning classifiers to statistics and neural networks," AI&STATS-93: PRELIMINARY PAPERS OF THE FOURTH INTERNATIONAL WORKSHOP ON ARTIFICIAL INTELLIGENCE AND STATISTICS, Ft. Lauderdale, FL, pp. 41-52.

Fisher, D. H. & J. Schlimmer (1988), "Concept simplification and prediction accuracy," *Proceedings of the Fifth International Conference on Machine Learning*, pp.22-28, Ann Arbor, MI: Morgan Kaufmann

Fisher, R. A. (1936), "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, Vol. 7, No. 2, pp.179-188.

R. Fletcher (1987), "Practical methods of optimization," *John Wiley and Sons, 2nd Ed.*

Foroutan, I. (1985), "Feature Selection For Piecewise Linear Classifiers," Ph.D Thesis, University of California, Irvine, CA.

Foroutan, I. and J. Sklansky (1987), "Feature Selection for Automatic Classification of Non-Gaussian Data," *IEEE Transactions on Systems, Man nnd Cybernetics*, 17(2) pp.187-198.

Forsyth, R. S., D. D. Clarke, and R. L. Wright (1994) Overfitting revisited: an information-theoretic approach to simplifying discrimination trees. Journal of Experimental and Theoretical Artificial Intelligence, 6(3):289--302, July--September 1994.

Frank E. and Witten I. H. (1998a), "Using a permutation test for attribute selection in decision trees," *Proc International Conference on Machine Learning,* Madison, Wisconsin, Morgan Kaufmann, pp 152-160.

Frank, E. and I. H. Witten (1998b), "Generating accurate rule sets without global optimization," In Shavlik, J., editor, *Machine Learning: Proceedings of the Fifteenth International Conference*, Morgan Kaufmann Publishers, San Francisco, CA

Freund, Y., Mason, L. (1999), "The alternating decision tree learning algorithm," *Proceeding of the 16th International Conference on Machine Learning*, Bled, Slovenia, pp. 124-133.

Y. Freund and R. Schapire (1996), "Experiments with a new boosting algorithm," *Proceedings of International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 148-156.

Friedman, J. H (1997), "On bias, variance, 0/1-loss, and the curse-of-dimensionality," *Data Mining and Knowledge Discovery*, Vol. 1, No. 1.

Gelfand, S. B., C. S. Ravishankar, and E. J. Delp (1991), "An iterative growing and pruning algorithm for classification tree design," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 13(2), pp. 163-174.

Gray, R. M. (1984), ''Vector quantization,'' *IEEE ASSP Magazine*, pp. 4–29.

Guur-Ali, O. and W. A. Wallace, (1993), "Induction of rules subject to a quality constraint: Probabilistic inductive learning," *IEEE Transactions on Knowldge and Data Engineering*, 5(6), pp. 979-984.

Golea, M. and M. Marchand (1990), "A growth algorithm for neural network decision trees," *EuroPhysics Letters*, 12(3), pp.205--210.

Guo, H. and S. B. Gelfand (1992), "Classification trees with neural network feature extraction," *IEEE Transactions on Neural Networks*, 3(6), pp. 923--933.

Haese, K. and G. J. Goodhill (2001), "Auto-SOM: Recursive Parameter Estimation for Guidance of Self-Organizing Feature Maps," *Neural Computation* 13, 595–619

Hanisch, W. (1990), "Design and optimization of a hierarchical classifier," *Journal of New Generation Computer Systems*, 3(2), pp. 159-173.

Heath, D., S. Kasif, and S. Salzberg (1993a), "k-DT: A multi-tree learning method," *Proceedings of the 2nd International Workshop on Multistrategy Learning*, pp. 138-149., Harpers Ferry, WV, 1993. George Mason University.

Heath, D., S. Kasif, and S. Salzberg (1993b), "Learning oblique decision trees," *IJCAI-93: Proceedings Of The Thirteenth International Joint Conference On Artificial Intelligence*, Vol. II, Chambery, France, 1993, Morgan Kaufmann, San Mateo, CA, Vol. 160, pp. 1002-1007.

D. Heckerman, "Bayesian networks for data mining," *Data Mining and Knowledge Discovery*, 1(1), 1997.

Ibaraki, T., and S. Muroga, "Adaptive Linear Classifiers By Linear Programming," *Technical Report 284*, Computer Science, University Of Illinois, Urbana-Champaign, 1968

T. Joachims (1997), "Text categorization with support vector machines," *Techical Report, LS VIII Number 23*, University of Dortmund.

G. H. John and P. Langley (1995), "Estimating Continuous Distributions in Bayesian Classifiers," *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Mateo, pp. 338-345.

Kamber, M., L. Winstone, W. Gong, S. Cheng, and J. Han (1997), "Generalization and Decision Tree Induction: Efficient Classification in Data Mining," *Proc. of 1997 Int'l Workshop on Research Issues on Data Engineering* (*RIDE'97*), Birmingham, England, 1997, pp. 111-120.

Kim, E. D., J. M. W. Lam, and J. Han (2000), "AIM: Approximate Intelligent Matching for Time Series Data", *2000 Int. Conf. on Data Wareshouse and Knowledge Discovery* (*DaWaK'00*), Greenwich, U.K., 2000.

Kim, H. and G. J. Koehler (1994), "An investigation on the conditions of pruning an induced decision tree," *European Journal of Operational Research*, 77(1), p.82, August 1994

Knobbe, A. J., A. Siebes, and D.M.G. Van der Wallen (1999), "Multi-Relational Decision Tree Induction" *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 378-383. Springer-Verlag, 1999.

Igor Kononenko (1995), "On biases in estimating multi-valued attributes," IJCAI-95: *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada, pp. 1034-1040.

Igor Kononenko And Ivan Bratko (1991), "Information based evaluation criterion for classifier's performance," *Machine Learning*, 6(1), pp. 67-80.

Kwok, S.W. and C. Carter (1990), "Multiple decision trees," In R.D. Schachter, T.S. Levitt, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, Vol. 4, pp. 327--335. Elsevier Science, Amsterdam

J.-Y. Lee, M. Afzal, S. Udpa, L. Udpa, and P. Massopust (2000), "Hierarchical rule based classification of MFL signals obtained from gas pipeline inspection," *IJCNN'00: International Joint Conference on Neural Networks*, Vol. 5, pp. 5071-5078.

Li, W., J. Han, and J. Pei (2001), "CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules," *Proc. 2001 Int. Conf. on Data Mining* (*ICDM'01*), San Jose, CA, 2001.

Liang, S., S. Fuhrman, R. Somogyi (1998), "REVEAL: A general reverse engineering algorithm for inference of genetic network architecture," *Pacific symposium on Biocomputing*, Vol. 3, pp.18-29

Lin, J.-H., and J. S. Vitter, "Nearly Optimal Vector Quantization via Linear Programming," In J. A. Storer and M. Cohn, Editors, Dcc 1992, *Data Compression Conference*, pp. 22-31, Los Alamitos, CA, March 24th-27th 1992, IEEE Computer Society Press

Lubinsky, D. (1994), "Classification trees with bivariate splits," *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks and Complex Problem-Solving Technologies*, 4(3) pp.283-296. 1994

Lubinsky, D. (1993), "Algorithmic speedups in growing classification trees by using an additive split criterion," *AI & Statistics-93: AI & Stats-93: Preliminary Papers of the Fourth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 1993, *Society for AI and Statistics*, pp. 435-444.

Y. Mansour (1997), "Decision tree pruning," *Machine Learning Foundations*, Lecture 9

Mehta, M., R. Agrawal, and J. Rissanen (1996), "SLIQ: A fast scalable classifier for data mining," *Proc. 1996 Intl. Conf. on Extending Database Technology* (*EDBT'96*), Avignon, France, 1996.

Van de Merckt, T. (1993a), "Decision trees in numerical attribute spaces," *Proceedings of the thirteenth International Joint Conference of Artificial Intelligence*, pp.1016-1021, Chambery, France

Van De Merckt, T. (1993b), "Decision trees in numerical attribute spaces," IJCAI-93: PROCEEDINGS OF THE THIRTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, Vol 2, Chambery, France, Morgan Kaufmann, pp. 1016-1021.

Mingers, J. (1987), "Expert systems - rule induction with statistical data," *Journal of the Operational Research Society*, 38(1), pp. 39-47.

Mogre, A., R. Mclaren, J. Keller, and R. Krishnapuram (1994), "Uncertainty management for rule-based systems with application to image analysis," *IEEE Transactions on Systems, Man and Cybernetics*, **24**(3), pp. 470-481.

Mooney, R., J. Shavlik, G. Towell, and A. Grove (1989), "An experimental comparison of symbolic and connectionist learning algorithms," In *Proc. 11th Intl. Joint Conf. on Artificial Intelligence*, pp. 775–787., Detroit, MI, 1989. Morgan Kaufmann.

D. T. Morris and D. Kalles (1994), "Decision trees and domain knowledge in pattern recognition," *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid systems, Machine Intelligence and Pattern Recognition*, Vol. 16, pp. 25-36

K. R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik (1997), "Predicting time series with support vector machines," *Proceedings of International Conference on Artificial Neural Networks*, p. 999.

Murphy, P. M., and M. Pazzani (1994), "Exploring the decision forest: An empirical investigation of Occam's Razor in decision tree induction," *Journal of Artificial Intelligence Research*, Vol. 1., pp.257-275

Murphy, P. M., and M. Pazzani (1991), "ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision trees," *Proceedings of the Eighth International Workshop of Machine Learning*

Murthy, S. K., S. Kasif, and S. Salzberg (1994a), "A system for induction of oblique decision trees," *Journal of Artificial Intelligence Research*, Vol.2, No.1, pp.1-32

Murthy, S. K., S. Kasif, and S. Salzberg (1994b), "A System for Induction of Oblique Decision Trees, " *Journal of Artificial Intelligence Research*, Vol. 2, pp. 1-33.

Murthy, S. K., S. Kasif, S. Salzberg, and R. Beigel (1993), "OCL: Randomized Induction of Oblique Decision Trees," *AAAI-93: Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, July 1993, AAAI Press, pp. 322-327.

T. Niblett and I. Bratko (1986), "Learning decision rules in noisy domains," *Expert Systems*, **86**, Cambridge Universoty Press.

Olshausen, B. A. and D. J. Field (1996), ''Emergence of simple-cell receptive field properties by learning a sparse code for natural images,'' *Nature*, Vol. 381, pp.607–609.Quinlan, J. R. (1987), "Simplifying decision trees," *International Journal of Man-Machine Studies*, Vol. 27, pp. 221-234

Pattipati, K. R. and M. G. Alexandridis (1990), "Application of heuristic search and information theory to sequential fault diagnosis," *IEEE Transactions on Systems, Man and Cybernetics*, 20(4), pp. 872-887.

Pazzani, M., P. Murphy, K. Ali, and D. Schulenburg (1994), "Trading off coverage for accuracy in forecasts: Applications to clinical data analysis," *AAAI Symposium on AI in Medicine*, pp. 106-110, Stanford, CA

Piatetsky-Shapiro, G., and W. J. Frawley (1991), *Knowledge Discovery in Databases*, AAAI/MIT Press

J. Platt (1998), "Fast Training of Support Vector Machines using Sequential Minimal Optimization," *Advances in Kernel Methods - Support Vector Learning*, A. Smola, B. Schölkopf, and C. Burges, eds., MIT Press.

Qing-Yun, S. and K.-G. Fu (1983), "A Method for the Design of Binary Tree Classifiers," *Pattern Recognition*, Vol. 16, pp.593-603.

J. R. Quinlan (1993), C4.5: Programs for machine learning, *Vol. 29, pp.5-44, Morgan Kaufmann*

J. R. Quinlan (1988), "An empirical comparison of genetic and decision tree classifiers, FIFTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, pp. 135-141, Ann Arbor, MI, Morgan Kaufmann.

J. R. Quinlan (1987), "Simplifying decision trees," *International Journal of Man-machine Studies*, 27, pp.221-234.

J. R. Quinlan (1986), *"Induction of decision trees,"* Machine learning, *Vol. 1, pp.81-106, edited by Jude W. Shavlik and Thomas G. Dietterich, Morgan Kaufmann*

J. R. Quinlan and R. L. Rivest (1989), "Inferring decision trees using the minimum description length principle. Information and Computation," 80(3):227--248, March 1989.

J. Rissanen (1989), *Stochastic Complexity in Statistica Enquiry*, World Scientific

J. Rissanen (1978), "Modeling by shortest data description," *Automatica*, pp. 465–471.

Schuermann, J. and W. Doster (1984) "A decision-theoretic approach in hierarchical classifier design," *Pattern Recognition*, Vol. 17, pp.359-369.

M. Sebag (1995) "Second order understandability of disjunctive version spaces," *IJCAI-95: Workshop on Machine Learning & Comprehensibility*, *Report LRI*, Universit'e ParisSud.

Sethi, I. K. (1990), "Entropy nets: From decision trees to neural networks," *Proceedings of IEEE*, 78(10).

Sethi, I. K. and G. P. R. Sarvarayudu (1982), "Hierarchical classifier design using mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4), pp. 441-445.

Smyth, P., A. Gray, and U. M. Fayyad (1995), "Retrofitting decision tree classifiers using kernel density estimation," *ML-95: Machine Learning: Proceedings of the Twelfth International Conference*, Tahoe City, CA, 1995, Morgan Kaufmann., San Mateo, CA. J. Schlimmer, editor.

Shafer, J. , R. Agrawal, and M. Mehta, (1996), "SPRINT: a scalable parallel classifier for data mining," *Proc. 22nd Intl. Conf. Very Large Data Bases (VLDB)*, pp. 544–555, Mumbai (Bombay), India, 1996.

Shek, E. C., R. R. Muntz, E. Mesrobian, and K. Ng (1996), "Scalable exploratory data mining of distributed geoscientific data," *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 32-37.

Shlien, S. (1990), "Multiple binary decision tree classifiers," *Pattern Recognition*, 23(7), pp.757-763.

Shlien, S. (1992), "Nonparametric classification using matched binary decision trees," *Pattern Recognition Letters*, 13(2) pp. 83-88.

Sirat, J. A. and J.-P. Nadal (1990), "Neural trees: A new tool for classification," *Network Computation in Neural Systems*, 1(4) pp.423-438.

Sklansky, J. and G. N. Wassel (1981), *Pattern Classifiers and Trainable Machines*, Springer-Verlag, New York

Sklansky, J. and L. Michelotti (1980), "Locally Trained Piecewise Linear Classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(2), pp.101—111.

A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans (1999), *Advances in Large Margin Classifiers*, The MIT Press, Cambridge, MA, London, England.

Jan L. Talmon, Willem R. M. Dassen, and Vincent Karthaus (1994), "Neural nets and classification trees: A comparison in the domain of ECG analysis," *Pattern Recognition*

in Practice IV: Multiple paradigms, Comparative studies and hybrid systems, also, *Machine Intelligence and Pattern Recognition*, Vol. 16. pp. 415—423.

Talmon, J. L. (1986), "A multiclass nonparametric partitioning algorithm," *Pattern Recognition Letters*, Vol. 4, pp. 31-38.

M. Tan (1993), "Cost-sensitive learning of classification knowledge and its applications in Robotics," *Machine Learning*, Vop.13, pp.7-33.

P. C. Taylor and B. W. Silverman (1993), "Block diagrams and splitting criteria for classification trees," *Statistics and Computing*, 3(4), pp. 147-161.

Todeshini, R. and E. Marengo (1992), "Linear Discriminant Classification Tree: A User-Driven Multicriteria Classification Method," *Chemometrics and Intelligent Laboratory Systems*, 16, pp. 25--35.

Utgoff, P. E. (1989), "Incremental induction of decision trees," *Machine Learning*, Vol.4, pp.161-186

Varshney, P. K., C. R. P. Hartmann, and J. M. De Faria Jr. (1982), "Applications of information theory to sequential fault diagnosis," *IEEE Transactions on Computers*, 31(2), pp.164-170.

Vapnik, V. (1998), *Statistical Learning Theory*, Wiley, New York

Vapnik, V. (1995), *The Nature of Statistical Learning Theory*, Springer Verlag, New York

C. S. Wallace and J. D. Patrick (1993), "Coding decision trees," *Machine Learning*, 11(1), pp. 7-22.

Wallis, S., and G. Nelson (2001), "Knowledge Discovery in Grammatically Analysed Corpora", *Data Mining and Knowledge Discovery*, 5, pp.305–335.

Weiss, S. M. and C. A. Kulikowski (1991), "Computer Systems that Learn: Classification and Prediction Methods from Statistics Neural Nets," *Machine Learning, and Expert Systems*, Morgan Kaufman

Weiss, S. M. and I. Kapouleas (1989), "An empirical comparison of pattern recognition, neural nets, and machine learning classification methods," In *Proc. 11th Intl. Joint Conf. on Artificial Intelligence*, pp. 781–787., Detroit, MI, 1989. Morgan Kaufmann.

T. Windeatt and G. Ardeshir (2001), "Boosting unpruned and pruned decision trees," *Applied Information: Proceedings of the IASTED International Symposia*, pp. 66-71.

Witten, I. H. and E. Frank. (1999), *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, pp.1-36. Morgan Kaufmann

D. H. Wu, L. Auslender, and K. P. Bennett (1998), "On Support Vector Decision Trees for Database Marketing," R.P.I Math Report No. 98-100, Rensselaer Polytechnic Institute, Troy, NY.

O. Zaiane and J. Han (1998), "WebML: Querying the World-Wide Web for Resources and Knowledge," *Proc. (CIKM'98) Int'l Workshop on Web Information and Data Management* (*WIDM'98*), Bethesda, Maryland, Nov. 1998, pp. 9-12.

APPENDIX A: PROOF OF LEMMAS

Proof of Proposition 2.1.

(1) From $p(y) = p(y \mid A_i = a_{ik}, A_j = a_{jl}) \cdot p_{A_i,A_j}(a_{ik}, a_{jl})$,

$$H(A_i, A_j) = -\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} p_{A_i,A_j}(a_{ik}, a_{jl}) \cdot \log_2 p_{A_i,A_j}(a_{ik}, a_{jl})$$

$$= -\sum_{k=1}^{n_i}\sum_{l=1}^{n_j} p_{A_i}(a_{ik}) p_{A_j|A_i}(a_{jl} \mid a_{ik}) \cdot \left\{\log_2 p_{A_i}(a_{ik}) + \log_2 p_{A_j|A_i}(a_{jl} \mid a_{ik})\right\}$$

$$= -\sum_{k=1}^{n_i} p_{A_i}(a_{ik}) \log_2 p_{A_i}(a_{ik}) \sum_{l=1}^{n_j} p_{A_j|A_i}(a_{jl} \mid a_{ik})$$

$$-\sum_{k=1}^{n_i}\left[\sum_{l=1}^{n_j} p_{A_j|A_i}(a_{jl} \mid a_{ik}) \cdot \log_2 p_{A_j|A_i}(a_{jl} \mid a_{ik})\right] \cdot p_{A_i}(a_{ik})$$

$$= H(A_i) + \sum_{k=1}^{n_i} H(A_j \mid A_i = a_{ik}) p_{A_i}(a_{ik}) = H(A_i) + H(A_j \mid A_i).$$

(2) ($\Rightarrow$) From $p(A_j = a_{jl} \mid A_i = a_{ik}) = p(A_j = a_{jl})$,

$$H(A_j \mid A_i) = -\sum_{k=1}^{n_i}\left[\sum_{l=1}^{n_j} p_{A_j|A_i}(a_{jl} \mid a_{ik}) \cdot \log_2 p_{A_j|A_i}(a_{jl} \mid a_{ik})\right] \cdot p_{A_i}(a_{ik})$$

$$= -\sum_{k=1}^{n_i}\left[\sum_{l=1}^{n_j} p_{A_j}(a_{jl}) \cdot \log_2 p_{A_j}(a_{jl})\right] \cdot p_{A_i}(a_{ik}) = H(A_j).$$

($\Leftarrow$) From $H(A_i, A_j) = H(A_i) + H(A_j)$, $H(A_j \mid A_i) = H(A_j)$.

$\Leftrightarrow p_{A_i}(a_{jl}) p_{A_j|A_i}(a_{jl} \mid a_{ik}) \cdot \log_2 p_{A_j|A_i}(a_{jl} \mid a_{ik}) = p_{A_j}(a_{jl}) \cdot \log_2 p_{A_j}(a_{jl})$

for any $\{p_{A_i}(\cdot)\}$ and $\{p_{A_j}(\cdot)\}$

$\Leftrightarrow \log_2 \{p_{A_j|A_i}(a_{jl} \mid a_{ik}) / p_{A_j}(a_{jl})\} = 0$ for any $\{p_{A_i}(\cdot)\}$ and $\{p_{A_j}(\cdot)\}$

$\Leftrightarrow p_{A_j|A_i}(a_{jl} \mid a_{ik}) = p_{A_j}(a_{jl})$ for any $\{p_{A_i}(\cdot)\}$ and $\{p_{A_j}(\cdot)\}$

$\Leftrightarrow A_i$ and A_j are independant.

(3) If A_j is dependent on A_i, then $p_{A_j|A_i}(a_{jl} \mid a_{ik}) = 0$ or 1. i.e., $H(A_i, A_j) = 0$. If A_j is independent on A_i, then $p_{A_j|A_i}(a_{jl} \mid a_{ik}) = p_{A_j}(a_{jl}) \Rightarrow H(A_j \mid A_i) = H(A_j)$. Therefore, it is always true that $0 \le H(A_j \mid A_i) \le H(A_j)$. It is equivalent to $H(A_i) \le H(A_j \mid A_i) \le H(A_i) + H(A_j)$.

Similarly, $H(A_j) \le H(A_j \mid A_i) \le H(A_i) + H(A_j)$.

(4) $H_{A_iA_j}(Y) = H(Y \mid A_i, A_j) = H(Y, A_i, A_j) - H(A_i, A_j)$

$$= H(A_i, A_j \mid Y) - H(A_i) - H(A_j \mid A_i) + H(Y)$$

$$= \{H(A_i \mid Y) - H(A_i)\} + [H\{(A_j \mid A_i) \mid Y\} - H(A_j \mid A_i)\} + H(Y)$$

$$= \{H(Y \mid A_i) - H(Y)\} + [H\{Y \mid (A_j \mid A_i)\} - H(Y)] + H(Y)$$

$$= H_{A_i}(Y) + H_{A_j|A_i}(Y) - H(Y).$$

(5) Substituting $A_i = (A_1, A_2)$ and $A_j = A_3$ from (4),

$$H_{A_1A_2A_3}(Y) = H_{(A_1A_2)}(Y) + H_{A_3|(A_1A_2)}(Y) - H(Y)$$

$$= H_{A_1}(Y) + H_{A_2|A_1}(Y) + H_{A_3|(A_1A_2)}(Y) - 2H(Y).$$

Suppose that, for $n = k$,

$$H_{A_1A_2\cdots A_k}(Y) = H_{A_1}(Y) + H_{A_2|A_1}(Y) + \cdots + H_{A_k|(A_1A_2\cdots A_{k-1})}(Y) - (k-1)H(Y).$$

For n = k + 1,

$$H_{A_1A_2\cdots A_{k+1}}(Y) = H_{(A_1A_2\cdots A_k)}(Y) + H_{A_{k+1}|(A_1A_2\cdots A_k)}(Y) - H(Y)$$

$$= H_{A_1}(Y) + H_{A_2|A_1}(Y) + \cdots + H_{A_k|(A_1A_2\cdots A_{k-1})}(Y) + H_{A_{k+1}|(A_1A_2\cdots A_k)}(Y) - k\,H(Y).$$

∴ By the mathematical induction,

$$H_{A_1A_2\cdots A_n}(Y) = H_{A_1}(Y) + H_{A_2|A_1}(Y) + \cdots + H_{A_n|(A_1A_2\cdots A_{n-1})}(Y) - (n-1)H(Y).$$

(6) A_i and A_j are independent

$\Leftrightarrow H((A_i, A_j) \mid Y) = H(A_i \mid Y, A_j \mid Y) = H(A_i \mid Y) + H(A_j \mid Y).$

$H_{A_iA_j}(Y) = H(Y \mid A_i, A_j) = H(Y, A_i, A_j) - H(A_i, A_j)$

$$= H(A_i, A_j \mid Y) - H(A_i) - H(A_j) + H(Y)$$

$$= \{H(A_i \mid Y) - H(A_i)\} + \{H(A_j \mid Y) - H(A_j)\} + H(Y)$$

$$= \{H(Y \mid A_i) - H(Y)\} + \{H(Y \mid A_j) - H(Y)\} + H(Y)$$

$$= H(Y \mid A_i) + H(Y \mid A_j) - H(Y) = H_{A_i}(Y) + H_{A_j}(Y) - H(Y).$$

(7) Y is linearly dependent on $\{A_1, A_2, \ldots, A_n\}$, and $\{A_1, A_2, \ldots, A_n\}$ are linearly independent.

$\Leftrightarrow H(Y \mid A_1, A_2, \ldots, A_n) = 0$, and

$H(A_1, A_2, \ldots, A_n \mid Y) = H(A_1 \mid Y) + H(A_2 \mid Y) + \cdots + H(A_n \mid Y)$. Therefore,

$$H(Y \mid A_1, A_2, \ldots, A_n) = H(A_1, A_2, \ldots, A_n \mid Y) + H(Y)$$

$$= \sum_{i=1}^{n} H(A_i \mid Y) + H(Y) = \sum_{i=1}^{n} \{H(Y \mid A_i) - H(Y)\} + H(Y) = \sum_{i=1}^{n} H_{A_i}(Y) + (n-1)H(Y)$$

$$= 0.$$

$$\therefore (n-1)H(Y) = H_{A_1}(Y) + H_{A_2}(Y) + \cdots + H_{A_n}(Y).$$

□

Proof of Proposition 2.2.

(1) A_i and A_j are independent. $\Leftrightarrow H(A_i) + H(A_j) = H(A_i, A_j) \Leftrightarrow M(A_i, A_j) = 0$.

(2) A_j depends on A_i. $\Leftrightarrow H(A_i \mid A_j) = 0 \Leftrightarrow M(A_i, A_j) = H(A_i)$.

(3) $0 \le H(A_j \mid A_i) \le H(A_j)$.

$\Leftrightarrow \max\{H(A_i), H(A_j)\} \le H(A_i, A_j) \le H(A_i) + H(A_j)$.

$\Leftrightarrow 0 \le H(A_i) + H(A_j) - H(A_i, A_j) \le H(A_i) + H(A_j) - \max\{H(A_i), H(A_j)\}$.

$\Leftrightarrow 0 \le M(A_i, A_j) \le \min\{H(A_i), H(A_j)\}$.

(4) From lemma 3.1.1(7), we claim that $(n-1)H(Y) = H_{A_1}(Y) + H_{A_2}(Y) + \cdots + H_{A_n}(Y)$ is equivalent to $H(Y) = M(Y, A_1) + M(Y, A_2) + \cdots + M(Y, A_n)$.

$(n-1)H(Y) = H_{A_1}(Y) + H_{A_2}(Y) + \cdots + H_{A_n}(Y)$

$$\Leftrightarrow H(Y)=\sum_{i=1}^{n}\{H(Y)-H(Y\mid A_i)\}=\sum_{i=1}^{n}\{H(Y)+H(A_i)-H(Y,A_i)\}=\sum_{i=1}^{n}M(Y,A_i).$$

□

Proof of Proposition 2.3.

(1) Since $M(A_i,A_j)=M(A_j,A_i)\ for\ \forall i\ and\ j$, **M** is symmetric. Suppose λ is an arbitrary complex eigenvalue of **M**, and **x** is the corresponding eigenvector. Let $\mathbf{x}^*$ is the complex conjugate of **x**. Then,

$$\overline{\mathbf{x}^*\mathbf{M}\mathbf{x}}=\mathbf{x}^*\mathbf{M}^{\mathrm{T}}\mathbf{x}=\mathbf{x}^*\mathbf{M}\mathbf{x}=\mathbf{x}^*\lambda\mathbf{x}=\lambda\|\mathbf{x}\|^2.\ \text{Since}\ \overline{\mathbf{x}^*\mathbf{M}\mathbf{x}}=\overline{\mathbf{x}^*\lambda\mathbf{x}}=\overline{\lambda}\|\mathbf{x}\|^2,\ \lambda=\overline{\lambda}.$$

It means that λ is real.

(2) Suppose λ_i is an arbitrary eigenvalue of **M**, and **x** is the corresponding eigenvector. So, $\mathbf{M}\mathbf{x}=\lambda_i\mathbf{x}$, where $\mathbf{x}\neq\mathbf{0}\Leftrightarrow\det(\mathbf{M}-\lambda_i\mathbf{I})=0\Leftrightarrow p(\lambda_i)=0\ for\ any\ \lambda_i\in\sigma(\mathbf{M})$. Therefore,

$$p(t)=\prod_{i=1}^{N}(t-\lambda_i).$$

(3) There are m linearly dependent attributes in $\{A_1, A_2, ..., A_N\}$.

$\Leftrightarrow$ There exist $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m$ such that $\mathbf{M}\mathbf{x}_i=\mathbf{0}$ $(i=1,2,...,m)$.

$\Leftrightarrow$ m eigenvalues of $\sigma(\mathbf{M})$ are zero.

$\Leftrightarrow$ $p(t)=\lambda^m\prod_{i=1}^{N-m}(t-\lambda_i),$ where $\lambda_i\neq 0\ for\ \forall i=1,2,...,N\text{-}m.$

Let $p(t)=\lambda^m g(t),$ where $g(t)=\prod_{i=1}^{N-m}(t-\lambda_i),$ so that $g(0)\neq 0.$ Then,

$$p^{(m-1)}(t)=m!g(t)+\binom{m-1}{1}\frac{m!}{2!}\lambda^2 g^{(1)}(t)+\binom{m-1}{2}\frac{m!}{3!}\lambda^3 g^{(2)}(t)+\cdots+\lambda^m g^{(m-1)}(t),$$

$$p^{(m)}(t)=m!g(t)+\binom{m}{1}\frac{m!}{1!}\lambda g^{(1)}(t)+\binom{m}{2}\frac{m!}{2!}\lambda^2 g^{(2)}(t)+\cdots+\lambda^m g^{(m)}(t).$$

$$\therefore p(0)=p^{(1)}(0)=\ldots=p^{(m-1)}(0)=0,\ p^{(m)}(0)\neq 0\quad (1\le m\le N-1).$$

□

APPENDIX B: MEASURES OF UNCERTAINTY

Shannon's Entropy

Let X be a discrete random variable taking a finite number of possible values $x_1,x_2,\ldots,x_n$ with probabilities $p_1,p_2,\ldots,p_n$ respectively such that $p_i \geq 0, i = 1,2,...,n, \sum_{i=1}^{n} p_i = 1$. We attempt to arrive at a number that will measure the amount of uncertainty. Let h be a function defined on the interval (0, 1] and $h(p)$ be interpreted as the uncertainty associated with the event $X = x_i, \; i = 1,2,...,n$ or the information conveyed by revealing that X has taken on the value x_i in a given performance of the experiment. For each n, we shall define a function H_n of the n variables $p_1,p_2,\ldots,p_n$. The function $H_n(p_1, p_2,..., p_n)$ is to be interpreted as the average uncertainty associated with the event $\{X = x_i\}, i = 1,2,...,n$ given by

$$H_n(p_1, p_2,..., p_n) = \sum_{i=1}^{n} p_i h(p_i) \tag{B.1}$$

Thus $H_n(p_1, p_2,..., p_n)$ is the average uncertainty removed by revealing the value of X. For simplicity we shall denote

$$\Delta_n = \left\{ P = (p_1, p_2,..., p_n) : p_i \geq 0, \sum_{i=1}^{n} p_i = 1 \right\}. \tag{B.2}$$

We shall now present some axiomatic characterizations of the measure of uncertainty $H_n(p_1, p_2,..., p_n)$ to arrive at its exact expression. For that, let X and Y be two independent experiments with n and m values respectively. Let $P = (p_1, p_2,..., p_n) \in \Delta_n$ be a probability distribution associated with X and $Q = (q_1, q_2,..., q_m) \in \Delta_m$ be a probability distribution associated with Y. This lead us to write that

$$H_{nm}(P * Q) = H_n(P) + H_m(Q), \tag{B.3}$$

for all $P = (p_1, p_2,..., p_n) \in \Delta_n$, $Q = (q_1, q_2,..., q_m) \in \Delta_m$ and

$$P*Q=(p_1q_1,...,p_1q_m,p_2q_1,...,p_2q_m,...,p_nq_1,...,p_nq_m)\in\Delta_{nm}. \tag{B.4}$$

Replacing $p_ih(p_i)$ by $f(p_i),\forall i=1,2,...,n,$ we get $H_n(P)=\sum_{i=1}^{n}f(p_i).$

Lemma B.1. Let $f:[0,1]\to\Re$ be a continuous function satisfying

$$\sum_{i=1}^{n}\sum_{j=1}^{m}f(p_i,q_j)=\sum_{i=1}^{n}f(p_i)+\sum_{j=1}^{m}f(q_j), \tag{B.5}$$

for all $p_i\geq 0, q_j\geq 0, \sum_{i=1}^{n}p_i=\sum_{j=1}^{m}q_j=1$. Then

$$f(p)=-Cp\log_b p, C>0, b>1, \tag{B.6}$$

for all $p\in[0,1]$ with $0\log_b 0=0.$

The proof is provided by Chaundy and McLeod (1960). Based on (B.3), (B.6), and Lemma B.1 we present the following theorem:

Theorem B.1. Let $H_n:\Delta_n\to\Re\,(n>1)$ be a function satisfying (B.3) and (B.6), where f is real valued continuous function defined over [0,1]. Then H_n is given by

$$H_n(p_1,p_2,...,p_n)=-C\sum_{i=1}^{n}p_i\log_b p_i, \tag{B.7}$$

where $C>0, b>1$ with $0\log_b 0=0.$

Alternatively the measure (B.7) can be characterized as follows (Shannon,1948; Feinstein, 1958).

Theorem B.2. Let $H_n:\Delta_n\to\Re\,(n>1)$ be a function satisfying the following axioms:

(i) $H_2(p,1-p)$ is a continuous function of $p\in[0,1]$.

(ii) $H_n(p_1,p_2,...,p_n)$ be a symmetric function of its arguments.

(iii) $$H_n(p_1,p_2,...,p_n)=H_{n-1}(p_1+p_2,p_3,...,p_n)+(p_1+p_2)H_2\left(\frac{p_1}{p_1+p_2},\frac{p_2}{p_1+p_2}\right),$$

$$p_1+p_2>0.$$

Then $H_n(p_1,p_2,...,p_n)$ is given by (B.7).

A third way to characterize the measure (B.7) is as follows (Aczél and Daróczy, 1975).

Theorem B.3. Let $H_n:\Delta_n\to\Re\,(n>1)$ be a function satisfying the following axioms:

(i) $H_n(p_1,p_2,...,p_n)$ is a continuous and symmetric function with respect to its arguments.

(ii) $H_{n+1}(p_1,p_2,...,p_n,0)=H_n(p_1,p_2,...,p_n)$.

(iii) $$H_n(p_1,p_2,...,p_n)\le H_n(\frac{1}{n},...,\frac{1}{n})$$

(iv) For $q_{kj}\ge 0,\sum_{k=1}^{n}\sum_{j=1}^{m}q_{kj}=1,\ p_k=\sum_{j=1}^{n}q_{kj},\forall k=1,2,...,n,$ we have

$$H_{nm}(q_{11},...,q_{1m},q_{21},...,q_{2m},...,q_{n1},...,q_{nm})$$
$$=H_n(p_1,p_2,...,p_n)+\sum_{k=1}^{n}p_kH_m\left(\frac{q_{k1}}{p_k},...,\frac{q_{km}}{p_k}\right),\ p_k>0,\forall k.$$

Then $H_n(p_1,p_2,...,p_n)$ is given by (B.7).

The following is a different way to characterize the measure (B.7). It is based on the functional equation famous as *fundamental equation of information.*

Theorem B.4. Let $H_n:\Delta_n\to\Re\,(n>1)$ be a function satisfying

$$H_n(p_1,p_2,...,p_n)=\sum_{t=2}^{n}(p_1+\cdots+p_t)\psi\left(\frac{p_i}{p_1+\cdots+p_t}\right),\qquad(B.8)$$

where ψ satisfies the following functional equation

$$\psi(p)+(1-p)\psi\left(\frac{q}{1-p}\right)=\psi(q)+(1-q)\psi\left(\frac{p}{1-q}\right),\ p,q\in[0,1],\ p+q\le 1, \tag{B.9}$$

with $K\le\psi(0)\le 0$ for all $p\in[0,1]$. Then $H_n(p_1,p_2,...,p_n)$ is given by (B.7).

For simplicity, let us take $b = 2$ in (B.7). If we put the restriction $H_2(0.5, 0.5)=1$ in the above theorems with take $b = 2$, we get $C = 1$. This yields

$$H_n(p_1,p_2,...,p_n)=-\sum_{t=2}^{n}p_i\log_2 p_i. \tag{B.10}$$

The expression (B.10) is famous as *Shannon's entropy* or *measure of uncertainty*.

Properties of Shannon's Entropy

The measure of uncertainty $H_n(p_1,p_2,...,p_n)$ given by (B.10) satisfies many interesting properties. For simplicity, we shall take $H(p_1,p_2,...,p_n)$ or $H(P)$ instead of $H_n(p_1,p_2,...,p_n)$. Unless otherwise specified, it is understood that $P=(p_1,p_2,...,p_n)\in\Delta_n$, $Q=(q_1,q_2,...,q_m)\in\Delta_m$, $P*Q=(p_1q_1,...,p_1q_m,p_2q_1,...,p_2q_m,...,p_nq_1,...,p_nq_m)\in\Delta_{nm}$, $V=(v_{11},...,v_{1m},v_{21},...,v_{2m},\ ...,v_{n1},...,v_{nm})\in\Delta_{nm}$, and $W_i=(w_{i1},w_{i2},...,w_{im})\in\Delta_m, \forall i=1,2,...,n$.

It is also understood that $0\log_b 0=0$ for any $b > 0$.

Property B.1. According to the above theorems, the general properties of $H(P)$ can be summarized as follows:

(i) (Nonnegativity) $H(P)\ge 0$ with equality iff $P = P^0$, where $P^0=(0,0,...,1^{(i)},0,...,0)\in\Delta_n$.

(ii) (Continuity) $H(P)$ is a continuous function of P.

(iii) (Symmetry) $H(P)$ is a symmetric function of its arguments, i.e., $H(p_1,p_2,...,p_n)=H(p_{r(1)},p_{r(2)},...,p_{r(n)})$ where r is any permutation from 1 to n.

(iv) (Expansible) $H(p_1, p_2, ..., p_n, 0) = H(p_1, p_2, ..., p_n)$

(v) (Decisive) $H(1,0) = H(0,1) = 0$.

(Normality) H(0.5,0.5) = 1

(Sum Representation) $H(P) = \sum_{i=1}^{n} f(p_i)$ where $f(p) = -p \log_2 p, 0 \le p \le 1$.

(Recursivity) $$H(p_1, p_2, ..., p_n) = H(p_1 + p_2, p_3, ..., p_n) + (p_1 + p_2) H\left(\frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2}\right),$$

$p_1 + p_2 > 0, n > 2.$

(vi) (Additivity) $H(p_1 q_1, ..., p_1 q_m, p_2 q_1, ..., p_2 q_m, ..., p_n q_1, ..., p_n q_m) = H(p_1, p_2, ..., p_n) +$ $H(q_1, q_2, ..., q_m)$, i.e., $H(P * Q) = H(P) + H(Q)$.

(vii) (Strongly Additive) $H(p_1 w_{11}, ..., p_1 w_{1m}, p_2 w_{21}, ..., p_2 w_{2m}, ..., p_n w_{n1}, ..., p_n w_{nm})$

$$= H(p_1, p_2, ..., p_n) + \sum_{i=1}^{n} p_i H(w_{i1}, w_{i2}, ..., w_{im}).$$

(viii) (Grouping) $H(p_1, p_2, ..., p_n) = H(p_1 + \cdots + p_t + p_{t+1} + \cdots + p_n)$

$$+ \sum_{k=1}^{t} p_k H\left(\frac{p_1}{\sum_{j=1}^{t} p_j}, ..., \frac{p_t}{\sum_{j=1}^{t} p_j}\right) + \sum_{k=t+1}^{n} p_k H\left(\frac{p_{t+1}}{\sum_{j=t+1}^{n} p_j}, ..., \frac{p_n}{\sum_{j=t+1}^{n} p_j}\right).$$

(ix) (Generalized Grouping) $H(p_1, ..., p_{s_1}, p_{s_1+1}, ..., p_{s_2}, ..., p_{s_{m-1}+1}, ..., p_{s_m})$

$$= H(p_1 + \cdots + p_{s_1}, p_{s_1+1} + \cdots + p_{s_2}, ..., p_{s_{m-1}+1} + \cdots + p_{s_m})$$

$$+ \sum_{i=1}^{m} (p_{s_{i-1}+1} + \cdots + p_{s_i}) H\left(p_{s_{i-1}+1} \Big/ \sum_{j=s_{i-1}+1}^{s_i} p_j, ..., p_{s_i} \Big/ \sum_{j=s_{i-1}+1}^{s_i} p_j\right).$$

Property B.2. (Binary-Entropic) Let $\psi(p) = H(p, 1-p), \quad 0 \le p \le 1.$ Then

(i) $\psi(p) = \psi(1-p).$ Therefore, $\psi(1) = \psi(0)$.

(ii) $\psi(0.5)=1$.

(iii) $\psi(p)+(1-p)\psi\left(\frac{q}{1-p}\right)=\psi(q)+(1-q)\psi\left(\frac{p}{1-q}\right),\ p,q\in[0,1],\ p+q\le 1.$

(iv) There exists a K such that $\psi(p)\le K$.

(v) The function $f: p\to\psi(p)$ is non-decreasing on the interval (0, 0.5].

(vi) For every $p_0\in(0,1),\ \lim_{q\to 0^+} H(p_0,q)=h(p_0)$.

(vii) $\lim_{p\to 0^+}\psi(p)=0.$

(viii) $H(p_1,p_2,...,p_n)=\sum_{t=2}^{n}(p_1+\cdots+p_t)\psi\left(p_t/(p_1+\cdots+p_t)\right).$

Property B.3. (Shannon-Gibbs Inequality) For $\forall\ P=(p_1,p_2,...,p_n)\in\Delta_n$ & $Q=(q_1,q_2,...,q_n)\in\Delta_n$, we have

$$-\sum_{i=1}^{n}p_i\log_2 p_i\le-\sum_{i=1}^{n}p_i\log_2 q_i, \tag{B.11}$$

with equality iff $p_i=q_i,\ \forall i$, or iff $P=Q$.

Property B.4. (Maximality) $H(p_1,p_2,...,p_n)$ is maximum when all the probabilities are equal, i.e.,

$$H(p_1,p_2,...,p_n)\le H(\frac{1}{n},...,\frac{1}{n}), \tag{B.12}$$

with equality iff $pi=\frac{1}{n},\ \forall i=1,2,...,n.$

Property B.5. (Uniform distribution) Let $\phi(n)=H\left(\frac{1}{n},\frac{1}{n},...,\frac{1}{n}\right),\ n>1,\ n\in\aleph.$ Then,

(i) $H\left(\frac{n-n_1}{n},\frac{n_1}{n}\right)=-\frac{n_1}{n}\{\phi(n_1)-\phi(n)\}-\frac{n-n_1}{n}\{\phi(n-n_1)-\phi(n)\},\ 0\cdot\phi(0)=0,\ 1\le n_1\le n.$

(ii) $\phi(n) \le \phi(n+1)$ as well as $n\phi(n) \le (n+1)\phi(n+1).$

(iii) $$\lim_{n\to\infty}\left[\phi(n+1) - \frac{n+1}{n}\phi(n)\right] = 0.$$

Property B.6. (Sub-additivity)

$$H(v_{11},...,v_{1m},v_{21},...,v_{2m},...,v_{n1},...,v_{nm}) \le H\left(\sum_{i=1}^{n} v_{i1}, \sum_{i=1}^{n} v_{i2},..., \sum_{i=1}^{n} v_{im}\right) + H\left(\sum_{j=1}^{m} v_{1j}, \sum_{j=1}^{m} v_{2j},..., \sum_{j=1}^{m} v_{nj}\right).$$

Property B.7. (Independence Inequality)

If $p_i = \sum_{j=1}^{m} v_{ij}, \forall\, i = 1,2,...,n,$ and $q_j = \sum_{i=1}^{n} v_{ij}, \forall\, j = 1,2,...,m,$ then

$$H(v_{11},...,v_{1m},v_{21},...,v_{2m},...,v_{n1},...,v_{nm}) \le H(p_1q_1,...,p_1q_m,p_2q_1,...,p_2q_m,...,p_nq_1,...,p_nq_m).$$

Property B.8. (Concavity) $H(P)$ is a concave function of P in Δ_n.

Property B.9. Let $Q = (q_1, q_2,..., q_n) \in \Delta_n$ be a probability distribution such that $q_1 \ge q_2 \ge ... \ge q_n$. Let us define $P = (p_1, p_2,..., p_n) \in \Delta_n$ such that $p_1 = q_1 - \Delta_u$, $p_2 = q_2 + \Delta_u$, and $p_i = q_j$, $\forall\, i = 3,4,...,n$. Then $H(Q) \ge H(P)$.

Property B.10. (Difference among two entropies)

If there exists θ such that $\sum_{i=1}^{n} |p_i - q_i| \le \theta \le \frac{1}{2},$ then $|H(P) - H(Q)| \le -\theta \log_2 \frac{\theta}{n}$ for all $P, Q \in \Delta_n$.

Property B.11. (Bounds on $H(P)$) For $1 \le \sigma \le n$, we have

(i) $$H\left(\sum_{i=1}^{\sigma} p_i, 1 - \sum_{i=1}^{\sigma} p_i\right) \le H(p_1, p_2,..., p_n).$$

(ii) $$H(p_1, p_2,..., p_n) \le H\left(\underbrace{\sum_{i=1}^{\sigma}\frac{p_i}{\sigma},..., \sum_{i=1}^{\sigma}\frac{p_i}{\sigma}}_{\sigma-times}, p_{\sigma+1},..., p_n\right).$$

Property B.12. (Relative to maximum probability) Let $p_{\max} = \max\{p_1,...,p_n\}$. Then

(i) $H(p_{\max}, 1-p_{\max}) \le H(P)$.

(ii) $$H(P) \le H\left(\underbrace{\frac{1-p_{\max}}{n-1},...,\frac{1-p_{\max}}{n-1}}_{(n-1)-times}, p_{\max}\right).$$

(iii) $H(kp_{\max}, 1-kp_{\max}) + kp_{\max}\log_2 k \le H(P)$, where k is a positive integer satisfying $\frac{1}{k+1} \le p_{\max} \le \frac{1}{k}$

(iv) $1-p_{\max} \le 0.5H(P), \quad p_{\max} \ge 0.5.$

Property B.13. Let $(p,1-p) \in \Delta_2$ and $(u,1-u) \in \Delta_2$ be two probability distributions. If $u > \max\{p,1-p\}$, then $\psi(u) < \psi(p)$, where ψ is as given in (B.8).

Property B.14.

(i) If $\{H_n\}$ is recursive (property B.1(viii)) for n = 3 and symmetric (property B.1(iii)) for n = 3, then it is symmetric (property B.1 (iii)) for n = 2 and decisive (property B.1(v)).

(ii) If $\{H_n\}$ is recursive (property B.1(viii)), symmetric (property B.1(iii)) for n = 3, then it is symmetric (property B.1(iii)) and expansible (property B.1(iv)).

(iii) If $\{H_n\}$ is recursive (property B.1(viii)) and symmetric (property B.1(iii)) for n = 3, then it is also strongly additive (property B.1(x)).

(iv) If $\{H_n\}$ is expansible (property B.1(iv)) and sub-additive (property B.6) for $n = m$, then it is nonnegative (property B.1(i)).

(v) If $\{H_n\}$ is branching of the form $H(p_1,p_2,...,p_n) - H(p_1+p_2,p_3,...,p_n) = \phi(p_1,p_2)$, where $\phi : \{(x,y) \mid x \in [0,1], y \in [0,1], x+y \le 1\} \to \Re$, then

$$H(p_1, p_2, ..., p_n) = H(p_1 + \cdots + p_{n-1}, p_n) + \sum_{k=2}^{n-1} \phi(p_1 + \cdots + p_{k-1}, p_k).$$

(vi) If $\{H_n\}$ is recursive (property B.1.(viii)), and $\psi(p) = H(p, 1-p), 0 \le p \le 1$, then

$$H(p_1, p_2, ..., p_n) = \sum_{t=2}^{n} H(p_1 + \cdots + p_t)\psi\left(\frac{p_t}{p_1 + \cdots + p_t}\right), \quad \text{with} \quad 0\psi\left(\frac{0}{0}\right) = 0.$$

(vii) If $\{H_n\}$ is normalized (property 1.6), symmetric (for n=3) (property B.1(iii)) and recursive (for n=3) (property B.1(viii)), then the function $\psi(x) = H(1-x, x), x \in [0,1]$ satisfies the properties B.3(i)-(iv).

(viii) Binary entropy properties given by B.3(i)-(iv) implies that $\{H_n\}$ is symmetric (property B.1(iii)), normalized (property B.1(vi)), expansible (property B.1(iv)), decisive (property B.1(v)), recursive (property B.1(viii)), strongly additive (property B.1(ix)) and additive (property B.1(x)).

[References]

1. ACZÉL, J.D. and Z. DARÓCZY (1975), *On Measures of Information and Their Generalizations"*, Academic Press, New York
2. CHAUNDY, T.W. and J.B. McLEOD (1960), "On a Functional Equation", *Proc. Edin. Math. Soc. Edin. Math. Notes*, **43**, 7-8.
3. FEINSTEIN, F. (1958), *"Foundations of Information Theory"*, McGraw Hill, New York.
4. MATHAI, A.M. and P.N. RATHIE (1975), *"Basic Concepts in Information Theory and Statistics"*, Wiley Eastern Ltd., New Delhi.
5. SHANNON, C.E. (1948), "A Mathematical Theory of Communication", *Bell Syst. Tech. J.*, **27**, 379-423, 623-656.

APPENDIX C: EXPERIMENTAL ANALYSIS EXAMPLES

A) Sample classification problems for nominal attributes only

Example 1) database for fitting contact lenses

1. Title: Database for fitting contact lenses
2. Source Information:
   a) Cendrowska, J (1987), "PRISM: An algorithm for inducing modular rules," International Journal of Man-Machine Studies, Vol. 27, pp.349-370.
   b) Donor: Benoit Julien (Julien@ce.cmu.edu)
   c) Date: 1st August 1990

3 Attribute Information (5 Attributes):

age	{young, pre-presbyopic, presbyopic}
spectacle-prescrip	{myope, hypermetrope}
astigmatism	{no, yes}
tear-prod-rate	{reduced, normal}
contact-lenses	{soft, hard, none}

4 Class Distribution (Number of Instances: 24):

hard contact lenses: 4; soft contact lenses: 5; no contact lenses: 15.

Example 2) balance scale weight & distance database

1. Title: Balance Scale Weight & Distance Database
2. Source Information:
   a) Siegler, R. S. (1976), "Three aspects of cognitive development," *Cognitive Psychology*, Vol. 8, pp.481-520.
   b) Donor: Tim Hume (hume@ics.uci.edu)
   c) Relevant Information:

      This data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the

right distance. The correct way to find the class is the greater of (left-distance * left-weight) and (right-distance * right-weight). If they are equal, it is balanced.

4. Attribute Information (4 Attributes):

left-weight	{ 1, 2, 3, 4, 5 }	right-weight	{ 1, 2, 3, 4, 5 }
left-distance	{ 1, 2, 3, 4, 5 }	right-distance	{ 1, 2, 3, 4, 5 }

5. Class Distribution (Number of Instances: 625):

L (Left): 288; B (Balanced): 49; R (Right): 288.

Example 3) breast cancer data

Citation request: *This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data. Please include this citation if you plan to use this database.*

1. Title: Breast cancer data (Michalski has used this)
2. Source Information:
   a) Zwitter, M. and M. Soklic, Institute of Oncology University Medical Center, Ljubljana, Yugoslavia
   b) Donors: Ming Tan and Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)
3. Attribute Information (9 Attributes):

Age	{10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99}
Menopause	{lt40, ge40, premeno}
tumor-size	{0-4,5-9,10-14,15-19,20-24,25-29,30-34,35-39,40-44,45-49,50-54,55-59}
inv-nodes	{0-2,3-5,6-8,9-11,12-14,15-17,18-20,21-23,24-26,27-29,30-32,33-35,36-39}
node-caps	{yes, no}
deg-malig	{1, 2, 3}
breast	{left, right}
breast-quad	{left-up, left-low, right-up, right-low, central}
irradiat	{yes, no}

4. Class Distribution (Number of Instances: 286):

no-recurrence-events: 201 instances;

recurrence-events: 85 instances

Example 4) chess end-game

1. Title: Chess End-Game: King+Rook versus King+Pawn on A7 (KRKPA7).
2. Source Information:
   a) Database originally generated and described by Alen Shapiro.
   b) Donor/Coder: Rob Holte (holte@uottawa.bitnet). The database was supplied to Holte by Peter Clark of the Turing Institute in Glasgow (pete@turing.ac.uk).
3. Attribute Information (36 Attributes):
   33 Attributes that have the value of 'True' or 'False':
   bkblk, bknwy, bkon8, bkona, bkspr, bkxbq, bkxcr, bkxwp, blxwp, bxqsq, cntxt, dsopp, hdchk, mulch, qxmsq, r2ar8, reskd, reskr, rimmx, rkxwp, rxmsq, simpl, skach, skewr, skrxp, spcop, stlmt, thrsk, wkcti, wkna8, wknck, wkovl, wkpos
   Others: dwipd = {'g', 'l'}, katri = {'b', 'n', 'w'}, wtoeg = {'n', 't', 'f'}
5. Class Distribution (Number of Instances: 3196):
   White can win in 1669 of the positions (52%).
   White cannot win in 1527 of the positions (48%).

Example 5) 1984 United States Congressional voting records database

1. Title: 1984 United States Congressional Voting Records Database
2. Source Information:
   a) Source: Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc., Washington, D.C., 1985.
   b) Donor: Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)
3. Attribute Information (17 Attributes): all Boolean valued = {YES, NO}
   handicapped-infants, water-project-cost-sharing, adoption-of-the-budget-resolution, physician-fee-freeze, el-Salvador-aid, religious-groups-in-schools, anti-satellite-test-ban, aid-to-Nicaraguan-contras, MX-missile, immigration, synfuels-corporation-cutback, education-spending, superfund-right-to-sue, crime, duty-free-exports, export-administration-act-south-Africa
4. Class Distribution (Number of Instances: 435):
   267 democrats (45.2 %)

168 republicans (54.8 %)

Example 6) lymphography domain

Citation request: *This lymphography domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data. Please include this citation if you plan to use this database.*

1. Title: Lymphography Domain

2. Sources:

a) Zwitter, M. and M. Soklic, Institute of Oncology University Medical Center, Ljubljana, Yugoslavia

b) Donors: Igor Kononenko, University E. Kardelj, Trzaska 25, 61000 Ljubljana, Bojan Cestnik, Jozef Stefan Institute, Jamova 39, 61000 Ljubljana, Yugoslavia

3. Attribute Information (17 Attributes):

9 attributes that are Boolean valued:

block_of_affere, bl_of_lymph_c, bl_of_lymph_s, by_pass, extravasates, regeneration_of, early_uptake_in, dislocation_of, exclusion_of_no,

Others:

lymphatics	{ normal, arched, deformed, displaced}
changes_in_lym	{ bean, oval, round}
defect_in_node	{ no, lacunar, lac_margin, lac_central}
changes_in_node	{ no, lacunar, lac_margin, lac_central}
changes_in_stru	{ no, grainy, drop_like, coarse, diluted, reticular, stripped, faint}
special_forms	{ no, chalices, vesicles}
lym_nodes_dimin	{ 1, 2, 3 }
lym_nodes_enlar	{ 1, 2, 3, 4 }
no_of_nodes_in	{ 1, 2, 3, 4, 5, 6, 7, 8 }

4. Class Distribution (Number of Instances: 148):

normal: 2; metastases: 81; malign_lymph: 61; fibrosis: 4.

Example 7) mushroom database

1. Title: Mushroom Database

2. Sources:

a) Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf

b) Donor: Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)

3. Attribute Information (22 Attributes):

cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s

cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s

cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y

bruises: true=t, false=f

odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s

gill-attachment: attached=a, descending=d, free=f, notched=n

gill-spacing: close=c, crowded=w, distant=d

gill-size: broad=b, narrow=n

gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y

stalk-shape: enlarging=e, tapering=t

stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?

stalk-surface-above-ring: ibrous=f, scaly=y, silky=k, smooth=s

stalk-surface-below-ring: ibrous=f, scaly=y, silky=k, smooth=s

stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y

stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y

veil-type: partial=p, universal=u

veil-color: brown=n, orange=o, white=w, yellow=y

ring-number: none=n, one=o, two=t

ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z

spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y

population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y

habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

5. Class Distribution (Number of Instances: 8124):

e = edible: 4208 (51.8%); p = poisonous: 3916 (48.2%)

Example 8) zoo database

1. Title: Zoo database
2. Source Information
   a) Creator: Richard Forsyth (1994)
   b) Donor: Richard S. Forsyth, 8 Grosvenor Ave., Mapperley Park, Nottingham NG3 5DX 0602-621676
3. Attribute Information (17 Attributes):
   1 nominal attribute: animal name = Unique for each instance
   1 numeric attribute: legs = { 0, 2, 4, 5, 6, 8 }
   15 attrbitues that have Boolean values:
   hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, tail, domestic, catsize.
4. Class Distribution (Number of Instances: 101):

mammal (41):	aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mink, mole, mongoose, opossum, oryx, platypus, polecat, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sealion, squirrel, vampire, vole, wallaby, wolf
bird (20):	chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren
reptile (5):	pitviper, seasnake, slowworm, tortoise, tuatara
fish(13):	bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna

amphibian (4): frog, frog, newt, toad

insect (8): flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp

invertebrate(10): clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm

B) Sample classification problems for numerical attributes only

Example 1) Wisconsin breast cancer

1. Title: Breast Cancer
2. Source Information
   a) Creator: Rich Maclin and Mark Craven. Computer Science Department, University of Minnesota, Duluth and Biostatistics and Medical Informatics Department, University of Wisconsin, Madison.
2. Attribute Information (9 Attributes, two-class problem):

Clump_Thickness	integer [1,10]
Cell_Size_Uniformity	integer [1,10]
Cell_Shape_Uniformity	integer [1,10]
Marginal_Adhesion	integer [1,10]
Single_Epi_Cell_Size	integer [1,10]
Bare_Nuclei	integer [1,10]
Bland_Chromatin	integer [1,10]
Normal_Nucleoli	integer [1,10]
Mitoses	integer [1,10]
Class	{ benign, malignant}

Example 2) Pima Indians diabetes

1. Title: Pima Indians Diabetes Database
2. Sources:
   (a) Original owners: National Institute of Diabetes and Digestive and Kidney Diseases
   (b) Donor of database: Vincent Sigillito (vgs@aplcen.apl.jhu.edu)

(c) The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). The population lives near Phoenix, Arizona, USA.

3. Number of Instances: 768
4. For Each Attribute: (all numeric-valued)
   1. Number of times pregnant
   2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
   3. Diastolic blood pressure (mm Hg)
   4. Triceps skin fold thickness (mm)
   5. 2-Hour serum insulin (mu U/ml)
   6. Body mass index (weight in kg/(height in m)^2)
   7. Diabetes pedigree function
   8. Age (years)
5. Class variable (0 or 1)
6. Missing Attribute Values: None

Example 3) heart statlog

1. Title: Heart Statlog

   This database contains 13 attributes (which have been extracted from a larger set of 75)
2. Attribute Information:
   1. age
   2. sex
   3. chest pain type (4 values)
   4. resting blood pressure
   5. serum cholestoral in mg/dl
   6. fasting blood sugar > 120 mg/dl
   7. resting electrocardiographic results (values 0,1,2)
   8. maximum heart rate achieved
   9. exercise induced angina
   10. oldpeak = ST depression induced by exercise relative to rest

11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

3. Attributes types

Real: 1,4,5,8,10,12

Ordered:11,

Binary: 2,6,9

Nominal:7,3,13

4. No missing values and 270 observations

Example 4) Iris plants

1. Title: Iris Plants Database
2. Sources:

(a) Creator: R.A. Fisher

(b) Donor: Michael Marshall (MARSHALLPLU@io.arc.nasa.gov)

3. Relevant Information:

This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper (1936) is a classic in the field and is referenced frequently to this day (see Duda and Hart, 1993, for example). The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

4. Number of Instances: 150 (50 in each of three classes)
5. Number of Attributes: 4 numeric, predictive attributes

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

6. class: Iris Setosa / Iris Versicolour / Iris Virginica
7. Missing Attribute Values: None

Example 5) ionosphere

1. Title: Johns Hopkins University Ionosphere database
2. Source Information:
   (a) Donor: Vince Sigillito (vgs@aplcen.apl.jhu.edu)
   (b) Source: Space Physics Group, Applied Physics Laboratory, Johns Hopkins University, Johns Hopkins Road, Laurel, MD 20723
3. Relevant Information:
   This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. See the paper for more details. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere.
4. Number of Instances: 351
5. Number of Attributes: 34 (All 34 predictor attributes are continuous)
6. Class = good or bad
7. Missing Values: None

Example 6) sonar

1. Title: Sonar, Mines vs. Rocks
2. (a) Source: The data set was contributed to the benchmark collection by Terry Sejnowski. The data set was developed in collaboration with R. Paul Gorman of Allied-Signal Aerospace Technology Center.
   (b) Maintainer: Scott E. Fahlman
3. Problem Description: The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occurs later in time, since these frequencies are

transmitted later during the chirp. The numbers in the labels are in increasing order of aspect angle, but they do not encode the angle directly.

It was observed that this random division of the sample set led to rather uneven performance. A few of the splits gave poor results, presumably because the test set contains some samples from aspect angles that are under-represented in the corresponding training set. This motivated Gorman and Sejnowski to devise a different set of experiments in which an attempt was made to balance the training and test sets so that each would have a representative number of samples from all aspect angles. Since detailed aspect angle information was not present in the database of samples, the 208 samples were first divided into clusters, using a 60-dimensional Euclidian metric.

4. Number of attributes: 60 (all are real valued from 0 to 1)
5. Class = Rock or Mine
6. Number of instances = 208

C) Sample classification problems for both nominal and numerical attributes

Example 1) description of the German credit dataset

1. Title: German Credit data
2. Source Information

   Creator & Donor: Hans Hofmann, Institut für Statistik und Ökonometrie Universität, Hamburg FB Wirtschaftswissenschaften, Von-Melle-Park 5, 2000 Hamburg 13
3. Attribute Information (20 Attributes):
4. Class Distribution (Number of Instances: 1000):

   'good' status: 700 instances

   'bad' status: 300 instances

5. Cost Matrix for misclassification:

   Unit cost for misclassification of 'good' credit as 'bad' = 5

   Unit cost for misclassification of 'bad' credit as 'good' = 1
6. Information of Attributes: Number of Attributes: 20 (7 numerical, 13 nominal)

Attribute 1: (nominal) Status of existing checking account (X)

A11: X < 0 DM

A12: 0 <= X < 200 DM

A13: X>= 200 DM / salary assignments for at least 1 year

A14: no checking account

Attribute 2: (numerical) Duration in month

Attribute 3: (nominal) Credit history

A30: no credits taken/all credits paid back duly

A31: all credits at this bank paid back duly

A32: existing credits paid back duly till now

A33: delay in paying off in the past

A34: critical account/other credits existing (not at this bank)

Attribute 4: (nominal) Purpose

A40: car (new)

A41: car (used)

A42: furniture/equipment

A43: radio/television

A44: domestic appliances

A45: repairs

A46: education

A47: (vacation - does not exist?)

A48: retraining

A49: business

A4a: others

Attribute 5: (numerical) Credit amount

Attribute 6: (nominal) Savings account/bonds (Y)

A61: Y < 100 DM

A62: 100 <= Y < 500 DM

A63: 500 <= Y < 1000 DM

A64: Y>=1000 DM

A65: unknown/ no savings account

Attribute 7: (nominal) Period (Z) of employment since

A71: unemployed

A72: Z < 1 year

A73: 1 <= Z < 4 years

A74: 4 <= Z < 7 years

A75: Z>=7 years

Attribute 8: (numerical) Installment rate in percentage of disposable income

Attribute 9: (nominal) Personal status and sex

A91: male : divorced/separated

A92: female: divorced/separated/married

A93: male : single

A94: male : married/widowed

A95: female: single

Attribute 10: (nominal) Other debtors / guarantors

A101: none

A102: co-applicant

A103: guarantor

Attribute 11: (numerical) Present residence since

Attribute 12: (nominal) Property

A121: real estate

A122: building society savings agreement/life insurance

A123: car or other, not in attribute 6

A124: unknown / no property

Attribute 13: (numerical) Age in years

Attribute 14: (nominal) Other installment plans

A141: bank

A142: stores

A143: none

Attribute 15: (nominal) Housing

A151: rent

A152: own

A153: for free

Attribute 16: (numerical) Number of existing credits at this bank

Attribute 17: (nominal) Job

A171: unemployed/unskilled - non-resident

A172: unskilled - resident

A173: skilled employee/official

A174: management/self-employed/highly qualified employee/officer

Attribute 18: (numerical) Number of people being liable to provide maintenance for

Attribute 19: (nominal) Telephone

A191: none

A192: yes, registered under the customers name

Attribute 20: (nominal) foreign worker

A201: yes

A202: no

Example 2) classification of MFL signals for gas pipeline inspection

The dataset of MFL signals has been acquired by the real experiment from gas pipeline transmission (Lee et al., 2000) supported by Gas Research Institute. Figure C1 shows a sample of MFL signals (it is a defect MFL signal).

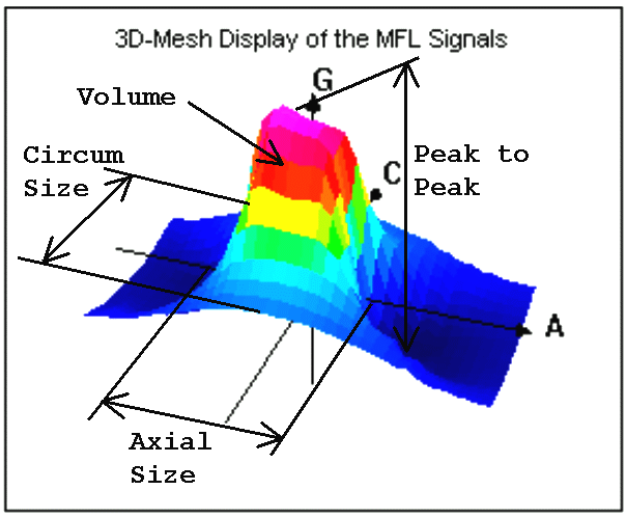


Figure C1. Feature representation of MFL signals.

Since the size of MFL signals are all different, just using pattern recognition does not classify these MFL signals. Therefore, appropriate attribute or feature representation is essential for MFL signal classification in this NDE application. The following attributes are the representation of MFL signals.

A1	Feature shape (geometry) in 2D projection (nominal)	Vertical / Elliptical
A2	Amplitude at the center of an indication object (MFL pattern)	Up / Down / Both
A3	Axial size in inch	Numeric
A4	Circumferential size in degree	Numeric
A5	Exists different size of indication object in axial direction	Yes / No
A6	The size of area of MFL signals higher than background	Numerical
A7	The size of area of MFL signals lower than background	Numerical
A8	Average of positive values of MFL signals	Numerical
A9	Average of negative values of MFL signals	Numerical
A10	The 'peak-to-peak' value of signals	Numerical
A11	The peak value at the center of signals	Numerical
A12	Maximum value of MFL signals	Numerical
A13	Clock position of the center of signals	Numerical
A14	Orientation of the feature (0-360 degree)	Numerical
A15	Number of vertical shapes of signals (Integer)	Numerical
A16	Background signal difference for pipeline transition (changes)	Numerical
A17	2nd Moment parameter 1 $\varphi 1 = (m_{20} + m_{02}) / m_{00}$	Numerical
A18	2nd Moment parameter 2 $\varphi 2 = \sqrt{(m_{20} - m_{02})^2 + 4m_{11}} / m_{00}$	Numerical
A19	3rd Moment parameter 3 $\varphi 3 = \sqrt{(m_{30} - 3m_{12})^2 + (3m_{21} - m_{03})^2} / m_{00}^{3/2}$	Numerical
A20	3rd Moment parameter 4 $\varphi 4 = \sqrt{(m_{30} + m_{12})^2 + (m_{21} + m_{03})^2} / m_{00}^{3/2}$	Numerical

The attributes A17 to A20 are called as invariant moment terms against the changes of size and skewness of MFL signals (see Lee et al., 2000). It was computed by the value of MFL signals in 2D images.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Professor Sigurdur Olafsson for their assistance in the preparation of this manuscript. Especially I thank Dr. David Jiles, Dr. Satish Udpa, and Dr. Lalita Udpa, who sincerely supported me for such a long time until I accomplish my works. Thanks also to the members of the school council for their valuable input. Also I specially thank to Dr. Seung-Jae Lee who supported and helped me for my works like his ones.

I sincerely thank to Dr. Sigurdur Olafsson, who is my major advisor, for instructing me on the scholar's conscience as well as academic areas. Also, I wish to thank to both Ms. Lori Bushore and Ms. Lisa Meyer for helping me graduate from abroad.

Personally I thank my wife, Minkyoung Song, and my daughter, Lucia, for believing me without any doubt or complaint, so that I could keep studying my own research. I also wish to appreciate our parents' belief in my long abroad studies. I always have a lot of debts for my family and friends. According to the oriental philosophy, parents, teachers, and lords are identical. I also remembered my advisor, Prof. Seung-Kown Kim, at Korea University, Seoul, South Korea. I don't know how I have to express how much I thank him.

In my memory, there were a lot of sincere friends at Iowa State University, and they are all going on the way of their success. I wish that every one with whom I got familiar will be happier and happier every day.