2003

Scalable optimization-based feature selection with application to recommender systems

Jaekyung Yang
Iowa State University

Scalable optimization-based feature selection with application to recommender systems

by

Jaekyung Yang

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
Shashi K. Gadia
John Jackman
Douglas Jacobson
Sarah M. Ryan

Iowa State University

Ames, Iowa

2003

Graduate College

Iowa State University

This is to certify that the doctoral dissertation of

Jaekyung Yang

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

It has been an exceptional journey - one that I had never dreamed of. Passing through enjoyable but sometimes painful experiences, I could have been getting more encouraged and strengthened. I am indebted a lot of people for this dissertation, which could not have been written without their support, help, patience and love of family, friends and my advisor, Dr. Sigurdur Olafsson. I remember the first meeting with him. He gave me a chance to work with him without even little hesitation. Even when I came to grief by a miscarriage of my wife, he consoled us with sending a beautiful flowerpot. Those were deeply impressed experiences of mine from him. Whenever I was in troubles and difficulties, he always presented a solution to me with his smiling face. I deeply appreciate his generosity, encouragement, advice, mentoring, and research support throughout my doctoral studies. I will never forget you, Siggi.

I would like to express special thanks to my committee members – Dr. John Jackman, Dr. Sarah Ryan, Dr. Shashi Gadia and Dr. Doug Jacobson. Their thoughtful concern, encouragement, and valuable comments were truly fruitful and improved my dissertation.

Especially, I would like to thank my wife Taeyun for her understanding and love during the past few years. Her support and encouragement was in the end what made this dissertation and who I am today possible. My lovely daughter Seojin, you have been the source of joy and blessing. Your pure smile has got me forget agonies from a difficulty of the research. My parents and parents-in-law, brother Jae-Hoon and his wife Eun-Young, receive my deepest gratitude and love for their dedication and the many years of support during my studies that provided the foundation for this work. Especially my mother, without your hearty prayer with tears, I could not have done anything. I love and thank you with all my heart.

I acknowledge, foremost, my dependence on God. He ordered my steps in every aspect of this study and all of my life. I thank for the marvelous ways he has brought people into my life that have helped me achieve my goals. This dissertation is sincerely dedicated to them.

Thank you all.

ABSTRACT

Along with development of a variety of data mining techniques, numerous feature selection methods have been introduced to reduce dimensionality. This may improve scalability and make interpreting learning models easier. In this dissertation a new optimization based feature selection method using the nested partition (NP) approach is presented, including both basic analysis of the NP framework and numerical results on various experiment problems. The numerical results show how the optimal structure of the NP makes contributions on a feature selection process. Further, it is addressed how the new intelligent partitioning method obtains very high quality partition efficiently. The feature selection method is implemented as both a filter and a wrapper.

In addition, the scalability of the algorithm, which is the most significant issue in mining large databases, is also dealt with according to the instance dimension, the feature dimension, and new features adaptation. However, since the NP naturally handle the feature dimension effectively, the dissertation mostly focuses on scalability with respect to the instance dimension. In this research problem, two systematic approaches to improve scalability of instance dimension are presented, which both utilize random sampling. Through this study, a predicted best solution for the size of instance samples is presented using a two-stage version of the NP that also incorporates statistical selection, and a heuristic solution is as well presented in a new adaptive version of the algorithm. Numerical results report that those two approaches are effective for scalability improvement, and perform better than the generic NP method that uses a static sampling approach.

In order to have the NP feature selection method flexible for handling mixed type of features, feature quality evaluators are introduced to determine the order of partitioning with experiment results reporting which one performs better based on a data domain. Finally as a case study, a recommender system that can be effectively used in B2B (business to business) e-business systems is provided using classification, association rules and the new NP-based feature selection method. The systems create recommendation rules for Internet auction participation and auction bids. The new feature selection algorithm is used to build a simple recommendation model that can be easily explained.

1 INTRODUCTION

In a modern world, databases contain huge amount of data that includes useful information. But, businesses may have some difficulties to extract useful information from the massive data that resides in these databases, which has been a motivation for continuously increased research in a knowledge discovery area. This process of discovering useful information in large databases consists of numerous steps, which may include integration of data from legacy databases, manipulation of the data to account for missing and incorrect data, and induction of a model with a learning algorithm, which is then used to identify and implement actions to take within the enterprise. Traditionally data mining draws heavily on both statistics and artificial intelligence, but numerous problems in data mining and knowledge discovery can also be formulated as optimization problems [Basu, 1998; Bradley et al., 1999].

All data mining starts with a set of data or a training set, which consists of instances describing the values of certain features. These instances are then used to learn some target concept, and depending upon the nature of this concept different learning algorithms are applied. One of the most common concepts is classification [Quinlan, 1986], where a learning algorithm is used to induce a model that classifies any new instances into one of two or more given categories. The primary objective may be for this classification to be as accurate as possible, but accurate models are not necessarily useful or interesting and other measures such as simplicity and novelty are also important. In addition to classification, other common concepts to be learned include association rules, numerical prediction models, and natural clusters of the instances.

Apart from the inductive learning, an important problem in knowledge discovery is analyzing the relevance of the features, usually called feature or feature (both terms are used with same meaning) subset selection that can be used to get simplicity as well as possibly accuracy benefits for the inductive learning. In this dissertation we primarily focus on the analysis of the feature relevance where the data is nominal, that is, each feature can only take finitely many values. Specifically, we introduce a new optimization-based approach for feature selection. If the data is not nominal, a standard discretization technique can be applied

as a preprocessing step [Fayyad and Irani, 1993]. However, in order to overcome this limitation, we also introduce several feature quality evaluators to deal with mixed type of features. We also deal with scalability of the feature selection method that is newly introduced. Finally we apply the new feature selection method to create the recommender system, an Internet auction system to facilitate reverse logistics that Ryan, Min, Olafsson, Min, (2001) described. Finally, it is shown how the feature selection method can be used to improve performance and readability of the recommendation model.

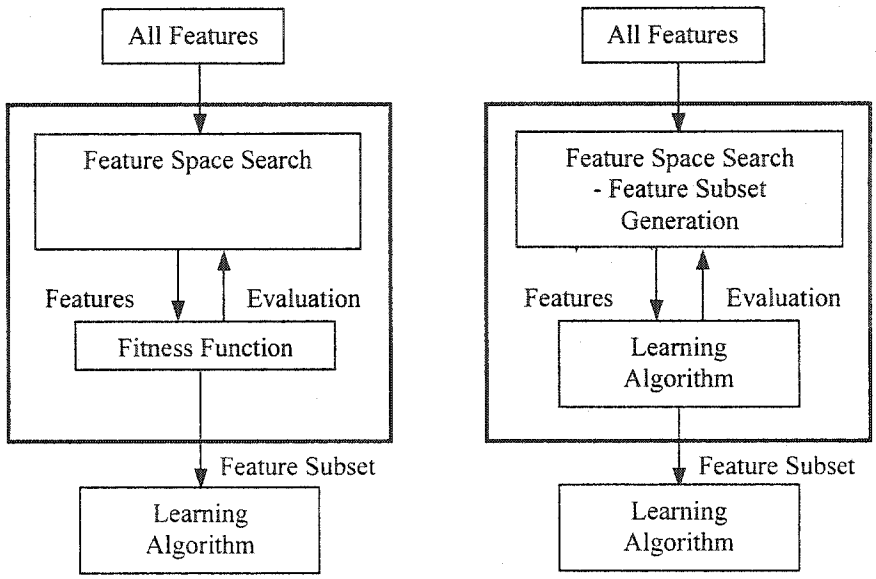
1.1 Feature Selection Method

Feature selection is an important data mining problem for numerous reasons. This involves a process for determining which features are relevant in that they predict or explain the data, and conversely which features are redundant or provide little information [Liu and Motoda, 1998]. Feature selection is commonly used as a preliminary step preceding a learning algorithm, which has numerous benefits. By eliminating many or even most of the features it becomes easier to train other learning methods, that is, computational time is reduced. Also, the resulting model may be simpler, which often makes it easier to interpret and thus more useful in practice. It is also often the case that simple models are often found be generalize better when applied for prediction. Thus, a model employing fewer features is likely to score higher on many interestingness measures and may even score higher in accuracy. Finally, discovering which features should be kept, that is identifying features that are relevant to the decision making, often provides valuable structural information and is therefore important in its own right.

The literature on feature relevance analysis is extensive within the machine learning and knowledge discovery communities, mostly under the name of feature selection. One way to classify the various algorithms is according to whether they evaluate one feature at a time and either include or eliminate this feature, or if an entire subset of features is evaluated together. According to such search schemes, most of the methods applied for this problem in the past can be categorized into sequential, random and exponential search based algorithms. This approach uses random search to explore the entire space of possible feature subsets, and

is thus similar to methods such as genetic algorithms and evolutionary search, although the search strategies themselves are quite different.

On the other hand these and other feature selection methods can be typically classified as either *filtering methods* that produce a ranking of all features before the learning algorithm is applied or *wrapper methods* that use the learning algorithm to evaluate subsets of features (See Figure 1.1). As a general rule, filtering methods are faster whereas wrapper methods usually produce subsets that results in more accurate models. We note that wrapper methods always fall into the latter category.



(a) Filter Approach (b) Wrapper Approach

Figure 1.1 Two approaches to feature selection on the learning algorithm dependence.

The feature selection problem is generally difficult to solve. The number of possible feature subsets is 2^n, where n is the number of features, and evaluating every possible subsets is therefore prohibitively expensive unless n is very small. Furthermore, in general there is no structure present that allows for efficient search through this large space, and a heuristic approach, that sacrifices optimality for efficiency, is typically applied in practice. Thus, most existing methods do not guarantee that the set of selected features is the optimal in any sense with a notable exception recently that applies mathematical programming to feature selection [Bradley, Mangasarian, and Street, 1998]. Furthermore, many of the widely used methods do

not allow the user to explicitly specify how many features should be considered. Some methods may thus have a bias towards sets with large number of features, and vice versa. Many of the methods have proven themselves valuable in practice, but not being able to make rigorous statements about the set of selected features without resorting to computationally expensive or otherwise restrictive methods is an apparent shortcoming.

In this dissertation, we propose the new feature selection methodology, which applies an optimization method called the nested partitions method [Shi and Olafsson, 2001] and its scalability. It is also addressed how the intelligent partitioning scheme that imposes a structure on the search space is incorporated in the new feature selection method to provide an efficient search.

1.2 Scalable Optimization-Based Feature Selection

Careful feature selection can improve the scalability of a data mining system as the induction is usually much faster with fewer features. Furthermore, feature selection also has an inherent value in that some structural knowledge may be obtained by selecting which features are important. In particular for applications where the last point is true, that is, we are really interested in which features are important, then considerable computation effort is justifiable for the feature selection process.

One situation where feature selection has intrinsic value is applications where it is important to not only obtain accurate predictions from a model but also to explain to a user how that prediction was obtained. In addition to the basic of new feature selection method, in this work we focus on the scalability of this approach in terms of its ability to handle increasing number of instances and increasing number of features [Liu and Setiono, 1998].

1.3 Mixed Type of Features and Recommender Systems

Our initial work will use the well known information gain [Quinlan, 1986] to evaluate feature quality. This has the limitation where a discretization process must be employed to evaluate numerical features. To overcome this limitation, we introduce two feature quality evaluators, namely correlation and ReliefF [Kononenko, 1994]. Those two feature evaluators are used in the NP feature selection algorithm to determine a partitioning order of features.

The numerical results are reported in terms of accuracy, size, and computational time based on three different kind of datasets, namely nominal, mixed and numeric. We investigate how the performance measures in the NP-Filter may be affected by the different feature quality evaluators with several learning algorithms through the numerical results.

As an application of the NP feature selection methods that is capable of handling mixed type of features, the recommender decision support for the auction system described above that can be effectively used in e-commerce systems between business to business is provided using classification and association rules. Through this case study we explore how recommendations in an e-auction system to users can be made intelligently and which benefits can be acquired by applying feature selection.

The remainder of this dissertation is organized as follows;

- Chapter 2

  This chapter surveys the literature related to the dissertation. Methods that have been proposed on feature selection, their scalability and recommender systems are discussed with some notion on relationship between those works and this dissertation.

- Chapter 3

  We discuss the basis for the new feature selection methodology, which is an optimization method called the nested partitions method. We report our numerical results in applying the method on well known classification problems. We also briefly evaluate the effectiveness and scalability of the new algorithms.

- Chapter 4

  In this chapter, we present systematic approaches to improve scalability of the proposed feature selection method. Furthermore, with the design of computing experiments about the tests that have not been presented in the previous chapter, various experiment results verify which approach is more efficient under some conditions.

- Chapter 5

  Several comments on limitation in the new methodology and alternatives to overcome the limitation are stated. Finally it is presented how the new feature selection method will be used for creating recommendation models.

- Chapter 6

  We conclude with a summary of the contribution of this dissertation and address some interesting directions for future research.

2 LITERATURE REVIEW

In the first section, we will review the literature on feature selection methods and will discuss scalability which is critical for avoiding complicated discovery processes on all the data. We also explore some of the current research on recommender systems.

2.1 Feature Selection Methods

A variety of feature selection methods have been proposed in the literature. The primal objective of feature selection is to find a possibly optimal or best feature subset by evaluating subsets of search space. In general, feature selection methods can be categorized into sequential search, random search, and exponential (exhaustive) search based algorithms according their search strategy [Doak, 1992]. Taking into account whether or not feature selection process embeds a learning algorithm, we can classify them into one of both filter and wrapper approaches [John, Kohavi and Pfleger, 1994].

2.1.1 Sequential Search Based Feature Selection

Various sequential search algorithms have been proposed. Sequential search algorithms sequentially add or remove features depending on a certain criteria using a hill climbing search strategy [Aha and Bankert, 1996].

The most well-known hill climbing search methods are forward selection and backward elimination methods. The forward selection starts with an empty feature set and adds/drops one feature at a time depending on whether or not it can improve the performance of the learner. On the other hand, the backward elimination starts with a complete set and remove one feature at a time using the same criteria as that of forward selection. Aha and Bankert (1996) examined variants of forward and backward feature selection. Koller and Sahami (1996;1997) employed forward selection and backward elimination to remove features according as a Markov blanket can be found within the set of remaining features, where a Markov blanket is a set of features that make the selected features conditionally independent of the remaining features and class feature.

Doak (1992) proposed bi-directional search for feature selection which does not add eliminated features and remove added features. Caruna and Freitag (1994) also proposed five hill greedy climbing procedures including three bi-directional search methods, FSS (Forward Stepwise Selection), BSE (Backward Stepwise Elimination) and BSE-SLASH. Both FSS and BSE start with empty and whole feature set respectively like others, but they can add or remove features at any step other than general forward and backward methods. Since they consider a best solution at a point, the solution can easily lead to local optimum. With the observation that it is not frequently for some learning algorithms (e.g. C4.5, ID3) to use many features, the BSE-SLASH starts with a complete feature set and removes features not used in learning procedures at every step. This search scheme allows a direct move to regions considering all features used in what is learned.

Kohavi (1994) and Kohavi and Frasca (1994) used a best first search [Ginsberg, 1993] to choose a set of relevant features maximizing the bootstrap accuracy estimate of the induced classifier at each state. Since the complexity of the search space is very expensive, $O(2^n)$, the search terminates when no improvement for a performance is found after k attempts [Raman, Ioerger, 2003]. Hall (1998) also employed a best first search in correlation-based filter algorithms.

The PRESET algorithm [Modrzejewski, 1993] uses rough sets theory to heuristically rank the features assuming a noise-free binary domain. A limitation to this approach is that it will fail to find the relevant features in case that features are highly correlated such as the parity problem, that is, the relevant features can not be found in the combinations of a few features [Liu and Setiono, 1998].

2.1.2 Random Search Based Feature Selection

Random search has a benefit that general heuristic based algorithms do not have, which can prevent the algorithm from easily falling into one local solution. Yang and Honavar (1998) used a genetic algorithm [Mitchell, 1997] to select a feature subset for neural network pattern classifiers (See Figure 2.1). The genetic operators, crossover and mutation, are very frequently used in genetic algorithms that represent individuals as binary strings. Given a string (e.g. 1111), the mutation operator changes 1 bit from 1 to 0, which represents that the

feature in the position of the bit 1 is deselected. Contrarily, change from 0 to 1 corresponds to a selected feature. Crossover exchanges, at a randomly selected position, sub-strings of two parent strings to produce two offspring. The genetic operators enable the algorithm to explore the feature subset space.

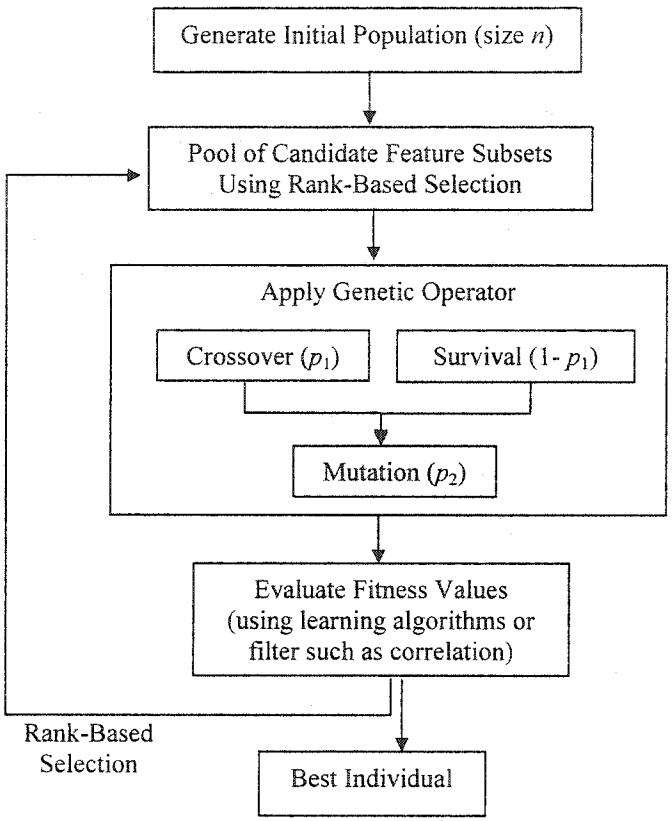


Figure 2.1 Feature selection procedures using a genetic algorithm [Yang and Honavar, 1998].

Feature selection using evolutionary search was proposed to search for all possible combinations of features and numbers of clusters with a standard K-means algorithm [Kim, Street, and Menczer, 2000]. Randomized (stochastic) search named random mutation hill climbing was proposed to select features and prototypes simultaneously for nearest neightbor classification where the prototypes are regarded as reference instances used in nearest neighbor computation [Skalak, 1994].

The Relief algorithm is an instance based filter that can handle numeric and nominal features but can not deal with more than 2 class problems [Kira and Rendell, 1992]. Kononenko (1994) proposed its variants solve the problems above on incomplete data set

(unknown value) and multiple classes. Liu and Setiono (1996a; 1996b) proposed LVF and LVW that are a modified version of Las Vegas algorithm [Brassard and Bratley]. Both LVF and LVW employ a random search to find relevant features.

2.1.3 Exponential and Optimal Search Based Feature Selection

All exhaustive search methods have an exponential time complexity which is impractical. This search can find an optimal solution, but contrarily all the optimal search methods do not require the exhaustive search. For example, if an evaluation measure is monotonic, the branch-and-bound algorithm can find an optimal solution [Naranda and Fukunaga, 1977]. The FOCUS algorithm starts with an empty feature set and performs exhaustive search until it finds a subset of as few features as possible called MIN-FEATURES that is in favor of consistent hypotheses on training examples [Almuallim and Dietterich, 1994]. It has limitation that the algorithm works on binary input features. The authors proposed three heuristics to make the search faster.

Bradley, Mangasarian and Street (1998) recently proposed feature selection methods applying mathematical programming. This mathematical model has a parametric objective function approximated by a sigmoid or by a concave exponential. The problem can be solved to achieve minimal number of features maintaining acceptable accuracy by generating a separating plane in a reduced feature space while minimizing the average distance of misclassified points to the plane. Koller and Sahami (1996) presented a theoretical model for optimal feature selection based on cross-entropy to minimize the amount of predictive information loss when removing features.

2.1.4 Filter and Wrapper Approaches

As stated previously, feature selection methods can be categorized into filter and wrapper approaches, depending on whether or not the feature selection process is dependent of a learning algorithm.

The filter approach methods use a fitness function for evaluating feature subsets rather than using a learning algorithm. The most widely known existing algorithms that fall into the filter approach are FOCUS [Almuallim and Dietterich, 1994], Relief [Kira and Rendell,

1992] and its variants [Kononenko, 1994], CFS (Correlation Based Feature Selection) [Hall, 1998], and cross-entropy filter [Koller and Sahami, 1996]. Other algorithms except FOCUS stated in the previous subsection will be dealt with in main chapters later in detail.

On the other hand, the wrapper approaches use a learning algorithm in the feature selection process. The wrappers usually provide better accuracy but are computationally more expensive than the filters [Raman and Ioerger, 2002]. Kohavi and John (1997) used the best first search for a wrapper approach. Many algorithms stated previously such as genetic algorithm [Yang and Honavar, 1998] and LVW [Liu and Setiono; 1996b] incorporate with a learning algorithm for evaluating feature subsets.

2.2 Scalability for Data Mining

Scalability of feature selection can be considered in terms of its ability to handle large number of instances, large number of features, and increasing features [Liu and Setiono, 1998]. Even though the authors proposed how to make LVF, feature selection algorithm, scalable by random sampling, research on scalability in feature selection has been much less emphasized than that on scalability in learning algorithms. The one reason might be because feature selection already employs scalability concept.

Most techniques that have been proposed for scaling up learning algorithms can be categorized into "design a fast algorithm", "partition the data", and "use a relational representation" where representing data relationally helps scale-up mining due to more efficient data representation [Provost and Kolluri, 1999]. The greedy, divide-and-conquer approaches to building class descriptions are very well-known fast algorithms that had much success for the time complexity. The run time of ID3 [Quinlan, 1986] and C4.5 [Quinlan, 1993] increases linearly in the number of instances for non-numeric data sets. In practical problems, those algorithms take more time usually. However, pruning techniques can make the algorithms scaled up. In addition, many heuristic algorithms have been developed for scaling up by finding approximate solutions.

The data partitioning approach usually includes selecting an instance subset, selecting a feature subset, or processing subsets sequentially/concurrently. When it comes to scalability of instances, Catlett (1991) stated that sampling instances involves random sampling,

duplicate compaction that removes duplicated instances, and stratified sampling by selecting the minor classes more frequently in order to make the class values uniformly distributed. Here the problem on the appropriate sample size to maintain an acceptable accuracy arises. Toivonen (1996) discussed the sufficient sample sizes for finding association rules that are probably consistent with whole database and are no smaller than a predetermined sample size, based on thresholds on the probability of error and the size of the error. Provost, Jensen, and Oates (1999) studied progressive sampling methods for finding minimum number of instance samples, that starts with a small sample and takes progressively larger ones until the accuracy no longer improves. This sampling method takes samples based on the "sampling schedule", $S = \{n_0, n_1, n_2, \ldots, n_k\}$ where n_i is an integer that specifies the sample size. For $i < j$, $n_i < n_j$. The authors addressed that schedules where the n_i increases geometrically make an efficient sampling method and an adaptive sampling algorithm based on knowledge of convergence and actual run-time complexity. Kivinen and Mannila (1994) stated upper and lower bounds for the sample sizes needed for approximately verifying sets of universal statements by using a tuple relational calculus, given a relation M. Here universal statements mean sentences that express information about the general structure of the data in databases, simply relational calculus.

Domingo, Gavalda, and Watanabe (2000) studied how to reduce the dimensionality of the data set using adaptive random sampling. This includes a sequential sampling approach that takes different number of samples sequentially depending on whether there have been already a large enough number of samples. The authors insisted that theoretical bounds for sample sizes in the batch sampling approach (static sampling approach that calculates the size a priori) are overestimated for most situations since the bounds consider an worst case situation. The adaptive sampling in an on-line sequential fashion having a statistical stopping criterion can overcome the problem above, thus the algorithm called AdaSelect that the authors proposed can take very fewer samples in an worst case.

John and Langley (1996) also proposed an adaptive (dynamic) sampling method, comparing with a static method. The dynamic sampling uses knowledge about the behavior of the learning algorithm to calculate a sample size based on a criterion called "PCE (Probably Close Enough)" for evaluating a sampling strategy. The PCE implies that there is

only a little chance that the learning algorithm could perform better by using the entire database instead. The smallest sample size, n, is taken based on the following formula. $\Pr(\text{acc}(N) - \text{acc}(n) > \varepsilon) \leq \delta$, where acc($n$) is the accuracy of a learning algorithm with the sample size n, acc(N) refers to the accuracy with the entire database, ε means "close enough" specified by a customer, and δ means "probably".

Weiss and Provost (2001) considered scalability from a different view by investigating the effect of class distribution on learning when the sample size must be limited. It is addressed that the based on the fact that classifiers performs worse on the minority class than on the majority class, allocating half of the training examples to the minority class can lead to perform better then using a natural class distribution. Thus, the authors suggested that using a progressive (adaptive) sampling strategy [Provost, Jensen and Oates, 1999] depending on the added minority and majority class examples can make better performance for classification learning.

2.3 Recommender Systems

In general, recommendation systems can be categorized into two areas such as collaborative filtering (social filtering) methods and content based filtering methods. Terveen and Hill (2001) addressed more detailed groupings as follows; content-based recommenders, recommendation support systems, social data mining and collaborative filtering. The content based filtering methods use information about things or items and user profiles as features, for example, director and other staff lists, actor/actress, genre, plots, user reviews for a movie case, and age, gender, etc. for user profiles. Those methods make recommendations based on similarities between items to appropriate users. Karypis (2000) proposed an item-based recommendation algorithm. This algorithm first computes the similarity between the items, second combine these similarities, and finally computes similarity between a "basket" of items and a candidate recommender item and make recommendations.

Research of the recommender systems has mainly focused on the business to customer problems that deal with information from a customer about which products a user is interested in [Sarwar, Karypis, Konstan, Riedl, 2000]. This kind of recommender systems use usually a collaborative filtering which is one of the earliest and most successful

recommender systems [Resnick, Iacovou, Suchak, Berstorm, Riedl, 1994; Shardanand, Maes, 1995; Hill, Stead, Rosenstein, Furnas, 1995; Konstan, Miller, Maltz, Herlocker, Gordon, Riedl, 1997]. Those recommendation systems can be regarded as social filtering methods that use ratings of users on "things" (for example, movies, music titles, books, etc.) and make recommendations to users considering new candidate items by using nearest neighbor techniques. Collaborative filtering matches people having similar interests and make recommendations [Terveen and Hill, 2001]. Thus, the methods must have participants who experienced similar items before to produce recommendations. Since these methods do not usually consider the data explaining nature of things, that approach may not be affected by the properties of things and may be unbiased [Hill, Stead, Rosenstein, Furnas, 1995].

Learning algorithm based recommender systems other than nearest neighbor based clustering have also been proposed. Guttman, Maes and Moukas (1998) addressed the model about purchasing behavior pattern of consumers using classification, which includes recommendations on which products to buy or broker and from whom to buy or broker products. Basu, Hirch and Cohen (1998) proposed an inductive approach to recommendation by employing both social and content-based filter methods. The approach formulated recommendations as a classification problem.

Breese, Heckerman and Kadie (1998) proposed model-based statistical algorithms for collaborative filtering or recommender systems employing Bayesian network and Bayesian clustering. This approach first constructs a basic model for user preferences where predictions are inferred. In the Bayesian clustering model, users having a similar pattern are clustered together as classes with assumption that ratings of a user are independent. From the model, the number of classes and the parameters of the model can be learned. In the Bayesian network model, variables in the network are titles with values having the allowable ratings. The structure of the network encodes the dependencies between titles which can be learned from the data with the conditional probabilities.

Good, Shafer, Konstan, Borchers and Sarwar (1999) proposed the approach to combine collaborative filtering and personal information filtering agents to produce better recommendations than each individual agent does. Lin, Alverez and Ruiz (2000) developed the collaborative recommendation based on association rules. The algorithm adjusts the

minimum support in order that the number of rules lies in a specified range and provides rule based recommendations. Vucetic and Obradovic (2000) proposed a regression based approach that searches for relationships between items, builds simple linear models, and combines the models to provide recommendations for users.

2.4 Summary and Discussion

In this chapter, we addressed that many researchers have proposed a number of feature selection methods with various attempts in order to reduce the dimensionality of a feature so that induced learning models can be easily interpreted. In general, most feature selection algorithms do not provide solutions that are optimal as well as found in practically reasonable time, that is, it is rare to find methods to satisfy both the conflict objectives. Thus, it gives motivation for new research to overcome the dilemma.

In addition, we introduced research on recommender systems with a variety of different approach. The collaborative filtering based recommendation systems using nearest neighbor that are more appropriate for business to customer (BtoC) e-commerce systems have been more emphasized. However, it is expected that more recommender systems for business to business (BtoB) systems would be proposed.

3 SCALABLE NP BASED FEATURE SELECTION

3.1 Introduction

The basis of the NP (Nested Partition) method, the new feature selection algorithm applying the NP, and its scalability problems are presented in this chapter. The key to the success of this method is a partitioning scheme that imposes a structure on the search space. Thus, it becomes critical to intelligently take advantage of special structure, and we develop such methods for partitioning for feature selection and report on our numerical experience in applying the method on well known classification problems. In addition we evaluate the effectiveness by comparing the NP feature selection method with no feature selection, entropy filter based feature selection and other feature selection methods and scalability of the new algorithms.

We define the feature selection problem as an integer program (IP) and analyze the new methodology's complexity. The scalability of this new methodology is presented by analyzing the scalability of the method with respect to both the instance dimension and the feature dimension. Finally, we summarize and discuss the presented issues and results.

3.2 Optimization Foundation

The basis for our new feature selection methodology is a recently proposed optimization method called the nested partitions method, and we start by considering the key concepts of this approach.

3.2.1 The Nested Partitions Method

The nested partition method was developed in [Shi and Olafsson, 2000] for solving global optimization problems where the feasible region X is finite, that is, problems of the form:

$$\min_{x \in X} f(x) \tag{3.1}$$

where $f : X \to R$ if a real valued performance measure defined on the finite set X.

We note that although our initial discussion of the NP method is generic, what we are interested in is the application of this methodology to feature selection in classification problems, where a subset of features is used to predict the class of a given instance based on their feature values. In this context X is the space of all possible feature subsets, and $f(x)$ is a measure such as the accuracy of applying a given learning algorithm to the classification problem using the features in $x \in X$, that is, the percentage of training instances that are classified correctly. To distinguish this problem from the generic case we denote the feasible region, that is the space of all feature subsets, as A, and a particular solution or a feature subset in this space as $A \in A$.

The basic idea of the NP method is to systematically partition the feasible region into subsets and focus the computational effort in those subsets that are considered promising. The main components of the method are:

- **Partitioning**: at each iteration the feasible region is partitioned into subsets that cover the feasible region but concentrate the search in what is believed to be the most promising region.
- **Random Sampling**: to evaluate each of the subsets, a random sample of solutions are obtained from each subset and used to estimate the performance of the region as a whole.

This method uses partitioning to divide the design space into regions that can be analyzed individually and then aggregates the results from each region to determine how to continue the search, that is, how to concentrate the computational effort. In other words, the NP method adaptively samples from the entire space of possible feature subsets and concentrates the sampling effort by systematic partitioning of this space.

To implement the partitioning, the NP method maintains in the k-th iteration what is called the most promising region, that is, a subregion $X(k) \subseteq X$ that is considered the most likely to contain the best solution or feature subset. This most promising region is partitioned into a given number of subregions and what remains is aggregated into one region called the surrounding region. Thus, a disjoint collection of sets covering the entire feasible region is considered. The subregions and the surrounding region are sampled using random sampling,

and the sampling information used to determine which region should be the most promising region in the next iteration. If one of the subregions contains the best solution, this region is now selected as the new most promising region and is, in the next iteration, partitioned into smaller subregions. If the surrounding region contains the best solution this is taken as an indication that the last move might not have been the best move, so the algorithm backtracks to what was the most promising region in the previous iteration. This partitioning creates a tree of subsets that we refer to as the partitioning tree. The distance of the current promising region $X(k)$ from the top of the tree, which corresponds to the minimum number of iterations it takes to get to this region, we refer to as the depth of the region. Ones a maximum depth region is reached, that is a region that will not be partitioned further, the algorithm terminates. In the context of feature selection problem, this maximum depth will be equal to the number of features that are considered for either inclusion or exclusion from the selected set.

As opposed to purely heuristic optimization methods, the NP method guarantees that the optimum solution is eventually found [Shi and Olafsson, 2000]. Furthermore procedures have been developed that allows us to specify a probability, say 90% or 95%, and terminate the algorithm when the probability that a good solution has been selected exceeds this value. Here a good solution is defined as a solution that has a performance that is within certain distance, called the indifference zone, of the optimal performance. As the algorithm terminates at the maximum depth $d^{(max)}$, the algorithm is set up to assure that the first maximum depth region visited is a good region with the desired minimum probability. The key to this result is to guarantee in each iteration of the algorithm that the correct move is made with a minimum probability ψ, which can be calculated numerically based on the desired final probability Ψ of terminating correctly and the number of levels in the partitioning tree. To make the correct selection with probability at least ψ, it can be shown to be sufficient to obtain N samples from each region, where

$$N = \left\lceil \frac{h(\psi)^2 s^2}{\delta^2} \right\rceil . \tag{3.2}$$

Here δ is the indifference zone, $h(\psi)$ is a function of the desired probability ψ, and s^2 is an estimate of the sample variance in the region. Thus, up to a constant the amount of computational effort needed is proportional to the sample variance. The derivations of these results for the case where the performance has to be estimated using simulation can be found in [Olafsson, 2003], that also contains further analysis of the behavior of the algorithm. Equation (3.2) does provide us with an important motivation for the importance of high quality partitioning, which is the main topic of this proposal and is discussed further in the next subsection.

3.2.2 Importance of Partitioning

The selected partition imposes a structure on the feasible region. When it is done in such a way that good solution as clustered together in the same subsets, then those subsets are selected by the algorithm with relatively little effort. On the other end of the spectrum, if the optimal solution is surrounded by solutions of poor quality it is unlikely that the algorithm will move quickly towards those subsets. This can be made more rigorous using equation (3.2) above, which shows that the amount of computational effort is directly proportional to the variance and implies we should attempt to cluster together similar solutions.

Another concept that is useful for shedding light on the importance of partitioning is the overlap between subsets. Assume a subset $X_g \subset X$ contains the optimal solution whereas $X | X_g$ does not. What we are interested in is the amount of computational effort required to correctly select X_g. Define

$$r(X_g) = \frac{|X_g| - |x \in X_g : f(x) < \min_{y \in X | X_g} f(y)|}{|X_g|}. \tag{3.3}$$

The smaller the overlap, that is the larger the value of equation (3.3) above, the fewer samples are needed from each region. Furthermore the number $N(X_g)$ of uniformly selected samples required to select X_g with probability at least ψ, grows exponentially in the overlap [Olafsson and Shi, 2001]:

$$N(X_g) = \begin{cases} \left\lceil \dfrac{\log(1-\psi)}{\log(r(X_g))} \right\rceil, & r(X_g) > 0, \\ 1, & r(X_g) = 0. \end{cases} \tag{3.4}$$

From these theoretical results, we thus can formulate our goal with partitioning in two ways: (i) to partition such that the variance within each region is minimized, or (ii) to partition such that the overlap between the region that should be selected and the other regions is minimized.

Incorporating structure into the partitioning is critical for the efficiency of the method and the theoretical derivations that justify this can be found in [Olafsson, 2003] and [Olafsson and Shi, 2001]. However, it is equally important to provide practitioners with methodology that can be implemented as automatic or semi-automatic system where the user does not need to develop complex partitioning methods and this our focus here. Now assume that we have some n decision variables $x_1, x_2, \ldots, x_n$. Then a generic partitioning method is to set one of the variables to each of its possible values at each level. Each order of the decision variables creates a different partition implying $n!$ different partitions. Among those, the best partition is the one that has the least overlap between the region containing the global optimum and regions not containing the global optimum, or alternatively minimizes the sum of the variance in each region. Thus, the question of a good partition reduces to ranking the decision variables in order of importance in such a way that there is as little overlap as possible between regions containing the desired optimum and other regions.

3.3 Problem Formulation

In this section we formulate the general feature selection problem to be solved. The following notation will be used throughout:

T : Training data (instances).

m : Number of instances ($n = |T|$).

$A^{(ALL)}$: Set of all features.

n : Number of features ($n = |A^{(ALL)}|$).

a : A specific feature ($a \in A^{(ALL)}$).

f : Performance measure.

f^* : Optimal Performance.

The feature selection process involves selecting a 'good' subset $A \subset A^{(ALL)}$, and a key question is how to define what makes a good subset, that is, the performance measure of the optimization problem.

3.3.1 Performance Measures

The main component in formulating the feature selection problem is selecting a performance measure. Depending on how this is done, feature selection methods may be divided into two categories: wrappers and filters. Wrapper methods use the accuracy of the resulting predictive model. Thus, to evaluate a subset $A \subset A^{(ALL)}$ of features, a predictive model $M(A)$ is induced based on these features, and the accuracy of this model

$$f^{M}_{accuracy}(A) \tag{3.5}$$

estimated, usually using a statistical procedure such as cross-validation or bootstrapping. This is an expensive evaluation and only applies for supervised learning. Thus, wrappers can only be applied in limited number of situations. Filtering methods, on the other hand, select features before any other learning algorithm is applied (if one is to be applied to all). Thus, a different performance measure must be specified and one of two techniques is applied: (i) some measures evaluate the performance of individual features and based on this a given number of the best features are used in the selected subset, whereas (ii) other measures evaluate the performance of an entire subset of features. Some of the possible measures include those based on the concept of entropy and those based on the concept of correlation between features.

- Entropy-Based Measures: Individual features can be evaluated based on measures such as the information gain:

  $$f_{gain}(T,a) = I(T) - E(a), \tag{3.6}$$

  where T is the training data, $I(T)$ is the expected information needed to classify a given instance in the training data, and $E(a)$ is the entropy of the feature $a \in A$. The information gain has bias towards favoring features with large number of possible values. This can be addressed by using the gain ratio instead.

- Correlation-Based Measures: The correlation between features in a subset and with the class features (if present) can also be used to evaluate the quality of the subset. Intuitively, a good feature subset should correlate highly with the class feature, but have low correlation with each other. We can thus use the following performance function [Hall, 1998]:

$$f_{correlation}(A) = \frac{k\overline{\rho}_{ca}}{\sqrt{k + k(k-1)\overline{\rho}_{aa}}}, \tag{3.7}$$

  where k is the number of features in the set A, $\overline{\rho}_{ca}$ is the average correlation between the features in this set and the classification feature, and $\overline{\rho}_{aa}$ is the average correlation between features in the set A.

- Simplicity Measures: As one of the benefits of feature selection are smaller, simpler models that are easier to interpret, a measure can be introduced to evaluate this directly:

$$f_{simplicity}(A) = \frac{1}{|A|}, \tag{3.8}$$

When choosing a wrapper or filter, the general consideration is that wrappers will give better performance when used with a supervised learning method, whereas filters are usually much faster. The NP-framework can be implemented as either a wrapper or filter, resulting in the NP-Wrapper and NP-Filter algorithm, respectively [Olafsson and Yang, 2001].

3.3.2 Integer Programming Formulation

With the performance measures defined, we can now define the decision variables as

$$x_i = \begin{cases} 1 & \text{if the } i\text{th feature selected,} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

$i \in A^{(ALL)}$. The feature selection problem can then be formulated as the integer programming problem that maximizes a combination of the above performance measures subject to constraints

$$\sum_{i \in A^{(ALL)}} x_i \geq n_{\min}^{select}$$

$$\sum_{i \in A^{(ALL)}} x_i \leq n_{\max}^{select}$$

$$x_i = 0 \text{ or } 1, \ \forall i,$$

where $n_{\min}^{select}$ and $n_{\max}^{select}$ are the minimum and maximum number of features, respectively, in the selected set. We let A denote the entire feasible region.

3.4 Feature Selection Methodology

As a method for solving the feature selection problem, the NP method has numerous attractive properties and advantages over previously proposed methods. Depending on the method that is used to evaluate sample subsets, it can be implemented as either a filter or a wrapper algorithm, but always searches through the space of feature subsets by evaluating the entire subsets. On the other hand, it also incorporates methods that evaluate individual features into the partitioning to impose a structure that speeds the search. Its eventual convergence is assured and once terminated rigorous statements can be made.

On the other hand, as it is computationally intensive compared to simple filters, this new methodology is particularly appealing when the feature selection is a critical task, such as when it is applied for obtaining structural information rather than as a preliminary step preceding a learning algorithm.

3.4.1 Intelligent Partitioning

The importance of partitioning as a means for imposing a structure on the search space is motivated above, and we now discuss an intelligent partitioning strategy when solving feature selection problems. For feature selection we let the decision variables (3.9) determine whether a feature is included in the set of selected feature or not.

Thus given a current set $A(k)$ of potential feature subsets, partition the set into two disjoint subsets

$$A_1(k) = \{A \in A(k) : a \in A\}, \tag{3.10}$$

$$A_2(k) = \{A \in A(k) : a \notin A\}. \tag{3.11}$$

Hence, a partition is defined by a sequence of feature a_1, a_2, …, a_n, which determines the order in which the features are either included or excluded (see Figure 3.1). According to the goals of a good partition, the order of the features should be selected so that the features that best separate good feature subsets from poor sets should be selected first. In other words, if there is a feature that must be included in any high quality feature subset, or vice versa a feature that should not be included, it is advantageous to select this feature early.

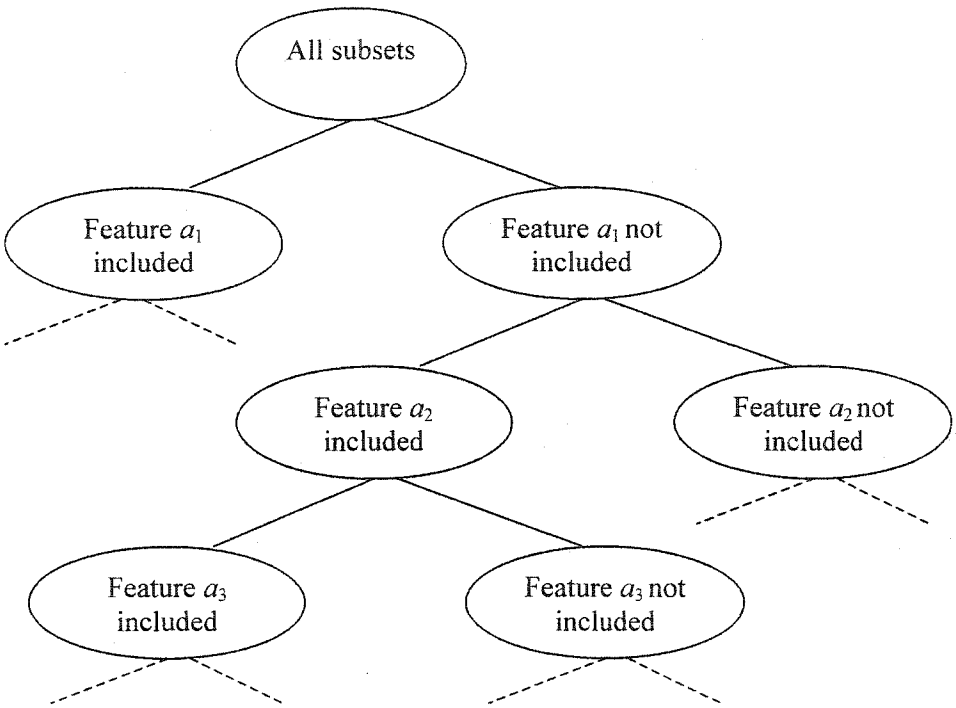


Figure 3.1 Partitioning tree

There are a number of strategies that have been developed to measure the importance of features, including the *information gain* which is obtained by knowing the value of each of the features [Quinlan, 1986], which is the method that we utilize here.

Suppose a training set T of m instances contains $s_{ij}(a)$ instances where feature a is set to its j-th value and the instance is classified as the i-th class. The total number of instances where a is set to the j-th value is then $S_j(a) = \sum_{i=1}^{c} s_{ij}(a)$, where c is the total number of classes. For this training set, the expected information that is needed to classify a given instance is given by

$$I(T) = -\sum_{i=1}^{c} p_i \log_2(p_i), \tag{3.12}$$

where $p_i = \frac{\sum_j s_{ij}(a)}{m}$ is the fraction of instances that belong to the i-th class. The information gain of a feature is the expected amount by which (3.12) is reduced if the value of the feature was known. It is calculated based on the *entropy* of the feature

$$E(a) = \sum_{j=1}^{v} q_j(a) \cdot I_j(a) \tag{3.13}$$

where v is the number of distinct values that feature a can take, and $q_j = \frac{S_j(a)}{m}$, the relative frequency of the j-th value in the training set, is the weight when the information function when a is set to its j-th value:

$$I_j(a) = -\sum_{i=1}^{c} p_{ij} \log_2(p_{ij}), \tag{3.14}$$

where $p_{ij} = \frac{s_{ij}(a)}{S_j(a)}$ is the proportion of instances with j-th value of feature a that belong to the i-th class. Then the *information gain* of feature a is $Gain(T,a) = I(T) - E(a)$ stated at (3.6). That is, the expected reduction in entropy that would occur if we know the value of feature a. Note that the feature with the highest information gain has the lowest entropy value.

The maximum information gain, or equivalently the minimum entropy, determines a ranking of the features. Thus, we select

$$\begin{aligned} a_1 &= \arg\min_{a \in A^{(ALL)}} E(a), \\ a_2 &= \arg\min_{a \in A^{(ALL)} \setminus \{a_1\}} E(a), \\ &\vdots \\ a_n &= \arg\min_{a \in A^{(AKK)} \setminus \{a_1, \ldots, a_{n-1}\}} E(a). \end{aligned}$$

where $A^{(ALL)}$ denotes the set of all features. The feature order $a_1, a_2, \ldots a_n$ defines a partition for the NP method that we call the *entropy partition*. We note that we chose to consider entropy to define the partition due to past success of using this measure for feature selection.

However, any other method for evaluating the value of individual features could be used in a similar manner.

3.4.2 Obtaining and Evaluating Feature Subsets

In our description of the NP method above we identified two primary components of the method: partitioning and sampling. The former is addressed using the entropy partition, and the latter can be improved by incorporating similar ideas.

There are two basic issues to be resolved when it comes to sampling, namely how to select samples from a given region, and how many samples to obtain. The amount of sampling can be solved by using formula (3.2) that has the benefit of guaranteeing the performance of the algorithm. How these samples are obtain, however, can be done either generically using uniform sampling, or by incorporating structure that assigns different probability to different feature subsets. The aim of the latter would be to select good feature subsets with higher probability and thus more quickly obtain a good estimate of the quality of each region. The idea of information gain can again be used. As features with high information gain are believed to be more useful, it is intuitively appealing to select those features with higher probability than features with low information gain. We thus propose the following approach for determining if a feature should be included in a sample. Assume that we are at the k-th iteration and as before let $a_1, a_2, \ldots a_n$ denote the selected sequence of features. As before, we let $d(k)$ denote the position or depth of $A(k)$ in the partition tree of the current most promising region. Recall that this implies that the first $d(k)$ features have either been fixed as included or exclude in the current set of features. We then sample according to the following probabilities:

$$Prob[\text{Select feature } a_i] = \frac{Gain(T, a_i)}{K \cdot \max\limits_{h \in \{d(k)+1,\ldots,n\}} Gain(T, a_h)}, \tag{3.15}$$

for $i = d(k)+1, d(k)+2, \ldots, n$. Here $Gain(T,a)$ is the information gain of each feature calculated according to equation (3.6) above and $K > 1$ is scaling constant. Note that all or none of the features can be included in the sample and the higher the information gain the more likely it is that a feature is included in a sample feature set. Furthermore, by selecting K

> 1 there is a positive probability to select feature subsets that do not include the feature with the highest information gain, and the expected number of features included is inversely proportional to the value of K.

Finally, once sample feature subsets have been obtained, these subsets must be evaluated. As is discussed in the introduction, how this is done defines whether the algorithm is a filter or a wrapper approach. An interesting property of the NP framework is that it can be implemented according to either approach and we thus consider two alternatives:

- (NP-Wrapper) We can use the learning algorithm itself to measure the performance of a set, that is, the set performance function becomes

$$f(A) = \text{Accuracy}(A), \tag{3.16}$$

  where the accuracy depends on which learning algorithm is going to be applied. This is what is commonly referred to as the wrapper approach and we refer to the NP algorithm that used equation (3.16) as the *NP-Wrapper*.
- (NP-Filter) We can also use the correlation (3.7) among features to measure the performance of each feature set. The basic idea here is that good feature sets should correlate highly with the class feature, but have low correlation with each other. We note that this is a filter approach and hence we refer to it as the *NP-Filter*.

We note that any learning algorithm can be used with a NP-Wrapper, and methods other than the correlation measure (3.7) that similarly evaluate feature subsets can be used for different variants of the NP-Filter.

3.4.3 Convergence

The key to the convergence of the NP method is the probability by which a region is selected correctly in each iteration. A sufficient condition for asymptotic convergence is that this probability of correct selection is bigger than one half, and to guarantee that a minimum probability is obtained, Olafsson (2003) proposed using a two-stage sampling procedure that determines how much random sampling effort, $N(\psi,\delta)$, is needed from each region to guarantee correct selection with probability ψ within an indifference zone $\delta > 0$. The two-stage sampling also allows us to further analyze the convergence of the algorithm and

develop statements concerning the quality of the solution once maximum depth is reached. In particular, an expression can be derived for the probability of having found sufficiently good solution the first time maximum depth is reached:

$$Prob\{|\, f(\mathsf{A}(k)) - f^{*}\,| \leq \delta\} \geq \Psi, \tag{3.17}$$

Where $\delta > 0$ is an indifference zone, that is a performance value difference that is considered insignificant, and

$$\Psi = \frac{\psi^{n}}{(1-\psi)^{n} + \psi^{n}}, \tag{3.18}$$

where ψ is the user selected minimum probability by which a correct selection is made in each iteration, and n is as before the total number of features. Sometimes it may be beneficial to stop the algorithm early, that is, before the maximum depth, n, of the partitioning tree is reached. Thus, we can specify a stopping depth $d_{stop}(n) \leq n$, define the objective function on sets of feature subsets as

$$f(\mathsf{A}(k)) = \max_{a \in \mathsf{A}(k)} f(a), \tag{3.19}$$

and equation (3.17) holds with Ψ replaced with

$$\Psi' = \frac{\psi^{d_{stop}(n)}}{(1-\psi)^{d_{stop}(n)} + \psi^{d_{stop}(n)}}. \tag{3.20}$$

The two-stage sampling defines on of the two key components of the NP method. The other is partitioning, and as shown in Section 3.3.1, partitioning for the feature selection problem reduces to determining an order for the features and then the subregions correspond to either including a feature or not including a feature (see Figure 3.1). Thus, assuming that the current most promising region is some subset $A(k) \subset A$ of the entire feasible region, then this subset is partitioned by fixing the next feature a in the order, that is, the subsets, $A_1(k)$ and $A_2(k)$ (3.10), (3.11) respectively. The surrounding region is simply $A_3(k) = A \backslash A(k)$. Each of these three regions is then sampled as discussed above and based on these samples the next most promising region is selected. In theory, the features can be selected in an arbitrary order, but in section 3.4.1 it is shown that an intelligent partitioning where features are ordered according to their information gain (3.6) performs significantly better, and this

partitioning is used in all of the numerical experiments below. Making the above discussion more specific, a NP-Filter algorithm is stated in section 3.4.4.

3.4.4 NP-Filter Algorithm

With all of the components of the NP for feature selection in place, we are in a position to state the proposed feature selection algorithms completely. The following algorithm can be used to implement the filter approach. Note that it uses a fixed number of n_0 samples to evaluate each region, starts with the set A of all possible feature subsets as the most promising region, and terminates when the depth of the most promising region has reached maximum, that is, it is a singleton. We also let A^* be the best feature subset found and f^* be the corresponding performance value, which is calculated according to equation (3.7) above.

NP-Filter

Step 0. Select the constant $K > 1$ for scaling the sampling distribution, and n_0, the number of sample points. Evaluate the entropy value of each feature and let $a_1, a_2, \ldots a_n$ be the corresponding order of features:

$$a_1 = \arg\min_{a \in A^{(ALL)}} E(a),$$
$$a_2 = \arg\min_{a \in A^{(ALL)} \setminus \{a_1\}} E(a),$$
$$\vdots$$
$$a_n = \arg \min_{a \in A^{(AKK)} \setminus \{a_1, \ldots, a_{n-1}\}} E(a).$$

Set $\mathsf{A}(0) = A$, $k = 0$, and $d(0) = 0$. Let $A^* = \{\}$ and set $f^* = \infty$.

Step 1. Partition $A(k)$ into two subregions and aggregate what remains into one surrounding region:

$$A_1(k) = \{A \in \mathsf{A}(k) : a_{d(k)} \in A\},$$
$$A_2(k) = \{A \in \mathsf{A}(k) : a_{d(k)} \notin A\},$$
$$A_3(k) = A \setminus A(k),$$

Step 2. From each of the three regions, independently obtain n_0 sample sets $A_1^j, A_2^j, \ldots, A_{n_0}^j$, $j = 1,2,3$, according the distribution

$$Prob[\text{Select feature } a_i] = \frac{Gain(T, a_i)}{K \cdot \max\limits_{h \in \{d(k)+1,\ldots,n\}} Gain(T, a_h)}, \text{ for } l = d(k)+1, d(k)+2, \ldots, n.$$

Here $Gain(T, a)$ is calculated according to equation (3.6).

Step 3. Obtain the best sample set from each region

$$A_{best}^j = \arg \min_{i=1,2,\ldots,n_0} f(A_i^j),$$

and $f(\cdot)$ is defined according to equation (3.7).

Step 4. Select the next most promising region based on the sample results

(a) If $f(A_{best}^1) < \min\{f(A_{best}^2), f(A_{best}^3)\}$, let $A(k+1) = A_1(k)$, $d(k+1) = d(k) + 1$.

If $f(A_{best}^1) < f^*$, let $f^* = f(A_{best}^1)$ and $A^* = A_{best}^1$.

(b) If $f(A_{best}^2) < \min\{f(A_{best}^1), f(A_{best}^3)\}$, let $A(k+1) = A_2(k)$, $d(k+1) = d(k) + 1$.

If $f(A_{best}^2) < f^*$, let $f^* = f(A_{best}^2)$ and $A^* = A_{best}^2$.

(c) Otherwise, let $A(k+1) = s(A_k)$ where $s(A(k))$ is the superregion of $A(k)$, and $d(k+1) = d(k) - 1$.

If $f(A_{best}^3) < f^*$, let $f^* = f(A_{best}^3)$ and $A^* = A_{best}^3$.

Step 5. If $d(k+1) = n$, stop and return A^* as the best subset. Otherwise, let $k = k + 1$ and go back to Step 1.

We emphasize that due to the possible backtracking in Step 4(c), the number of iterations taken by the algorithm is random, and as we will show in the numerical results, depends on the quality of the partition. Except for slight modifications to Step 3, the NP-Wrapper is implemented in an identical fashion. We do therefore not present that implementation in detail, but rather illustrate this algorithm via an example.

3.4.5 Complexity

As the NP-Filter is a randomized algorithm with non-deterministic stopping time, analyzing complexity requires some care. Notice that the algorithm has two loops that may

depend on the number of features (n) and the number of instances (m). Let's start with the inner loop. Randomly selecting a feature subset normally requires one pass through the set of features and is thus $O(n)$. If the number of such sample subsets is fixed, that is, $N(\psi,\delta)$ is constant, then the entire loop is $O(m)$. However, if the more advanced two-stage sampling is used, the number $N(\psi,\delta)$ of samples is random and depends on the structure of the space of all feature subsets. In general one can expect this to depend on both n and m, and for a specific problem have some expected value $\mu(n,m) = E[N(\psi,\delta,n,m)]$. Thus the complexity of the inner loop is $O(\mu(n,m)\, n)$. In the outer loop, a sample of instances $\nu(m)$ must first be obtained with complexity $O(\nu(m))$. Finally, the number of iterations in the outer loop is stochastic because of possible backtracking. On the other hand, if there is no backtracking it takes some $d_{stop}(n)$ number of steps where $d_{stop}(n) \leq n$, and the complexity of the entire algorithm is $O(d_{stop}(n)(\nu(m) + \mu(n,m)\, n))$. The functions $d_{stop}(n)$ and $\nu(m)$ are entirely up to the designer of the algorithm, and one suggested strategy is to let both be square-root functions. We could also simply have fixed number of sample feature subset, that is, $\mu(n,m)$ would be equal to a constant. Then the complexity reduces to $O(\sqrt{n}(\sqrt{m}+n))$, but since the variable number of sample subsets and backtracking have been dropped, this is now simply a heuristic with no performance guarantees.

3.4.6 NP-Wrapper Example

To illustrate the mechanism of the new algorithm we apply the NP-Wrapper using Naive Bayes as learning algorithm to the simple weather classification problem illustrated in Table 3.1. There are 4 features (*Outlook, Temperature, Humidity, and Windy*) that can be used to predict a class feature that can take two values: play or no play. To determine the order in which features are selected for partitioning, we calculate the entropy of each feature according to equation (3.13) and get:

$$E(Outlook) = 0.693,$$
$$E(Temperature) = 0.911,$$
$$E(Humidity) = 0.788,$$
$$E(Windy) = 0.892.$$

Table 3.1 Data for simple weather example

Outlook	*Temperature*	*Humidity*	*Windy*	*Class*
Sunny	Hot	High	False	No play
Sunny	Hot	High	True	No play
Overcast	Hot	High	False	Play
Rain	Mild	High	False	Play
Rain	Cool	Normal	False	Play
Rain	Cool	Normal	True	No play
Overcast	Cool	Normal	True	Play
Sunny	Mild	High	False	No play
Sunny	Cool	Normal	False	Play
Rain	Mild	Normal	False	Play
Sunny	Mild	Normal	True	Play
Overcast	Mild	High	True	Play
Overcast	Hot	Normal	False	Play
Rain	Mild	High	True	No play

The expected information of the training set T is $I(T) = 0.94$, so *Outlook* has the highest information gain $Gain(T, Outlook) = 0.247$, followed by *Humidity* with $Gain(T, Humidity) = 0.152$ and *Windy* with $Gain(T, Windy) = 0.048$, and finally *Temperature* has the smallest information gain of $Gain(T, Temperature) = 0.029$. The resulting partitioning tree is shown in Figure 2, which also shows the final feature set for each maximum depth region, and the corresponding accuracy value when Naive Bayes is used as learning algorithm. As this problem is quite small we have chosen the maximum depth equal to the total number of features. The accuracy values in Figure 3.2, as well as those use by the NP-Wrapper, are calculated using 10-fold cross-validation.

First, note that the intelligent partitioning indeed imposes a useful structure on the space of feature subset that can be exploited by the random search. As outline above, we can measure this in two ways: using the variability of the accuracy or the percentage overlap between the set containing the global optimum and other sets. In particular, note that feature subsets with similar accuracy tend to be clustered together and the sample estimates of the best accuracy in each region will therefore have low variability. An extreme case is the set defined by all feature subsets containing *Outlook* but not containing *Humidity*, where every feature subset has the same accuracy and hence the variability is zero. Similarly, the overlap

is small and in the extreme case, the set containing *Outlook* and *Humidity*, which includes the optimum, has no overlap with other subsets. Due to this imposed structure it can be expected that the random search quickly moves towards the optimal feature subset of {*Outlook, Humidity*}.

We now illustrate a few iterations of the algorithm. We initialize the algorithm by setting $A(0) = A$, and then partition and sample as follows: The most promising region $A(0)$ is partitioned into two subsets depending on if we include or exclude the feature with the highest information gain, namely *Outlook*:

$$A_1(0) = \{A \in A(0) : Outlook \in A\},$$

$$A_2(0) = \{A \in A(0) : Outlook \notin A\},$$

Next we obtain samples from each one of those regions, according to the distribution (3.15) with $K = 1.2$, which takes the information gain into account. For example, the probabilities that each of the three remaining features is included in a sample from $A_1(0)$ are given as follows:

$$Prob[\text{Sample includes } Temperature] = \frac{0.029}{1.2 \cdot 0.152} = 0.16$$

$$Prob[\text{Sample includes } Humidity] = \frac{0.152}{1.2 \cdot 0.152} = 0.83$$

$$Prob[\text{Sample includes } Windy] = \frac{0.048}{1.2 \cdot 0.152} = 0.26$$

Thus, for each sample obtained it has the following distribution

$$Prob[\text{Selects } \{Outlook, Temperature\}] = 0.16 \cdot (1 - 0.83) \cdot (1 - 0.26) = 0.02$$

$$Prob[\text{Selects } \{Outlook, Humidity\}] = (1 - 0.16) \cdot 0.83 \cdot (1 - 0.26) = 0.52$$

$$Prob[\text{Selects } \{Outlook, Temperature, Humidity\}] = 0.16 \cdot 0.83 \cdot (1 - 0.26) = 0.10$$

$$\vdots \qquad \vdots \qquad \vdots$$

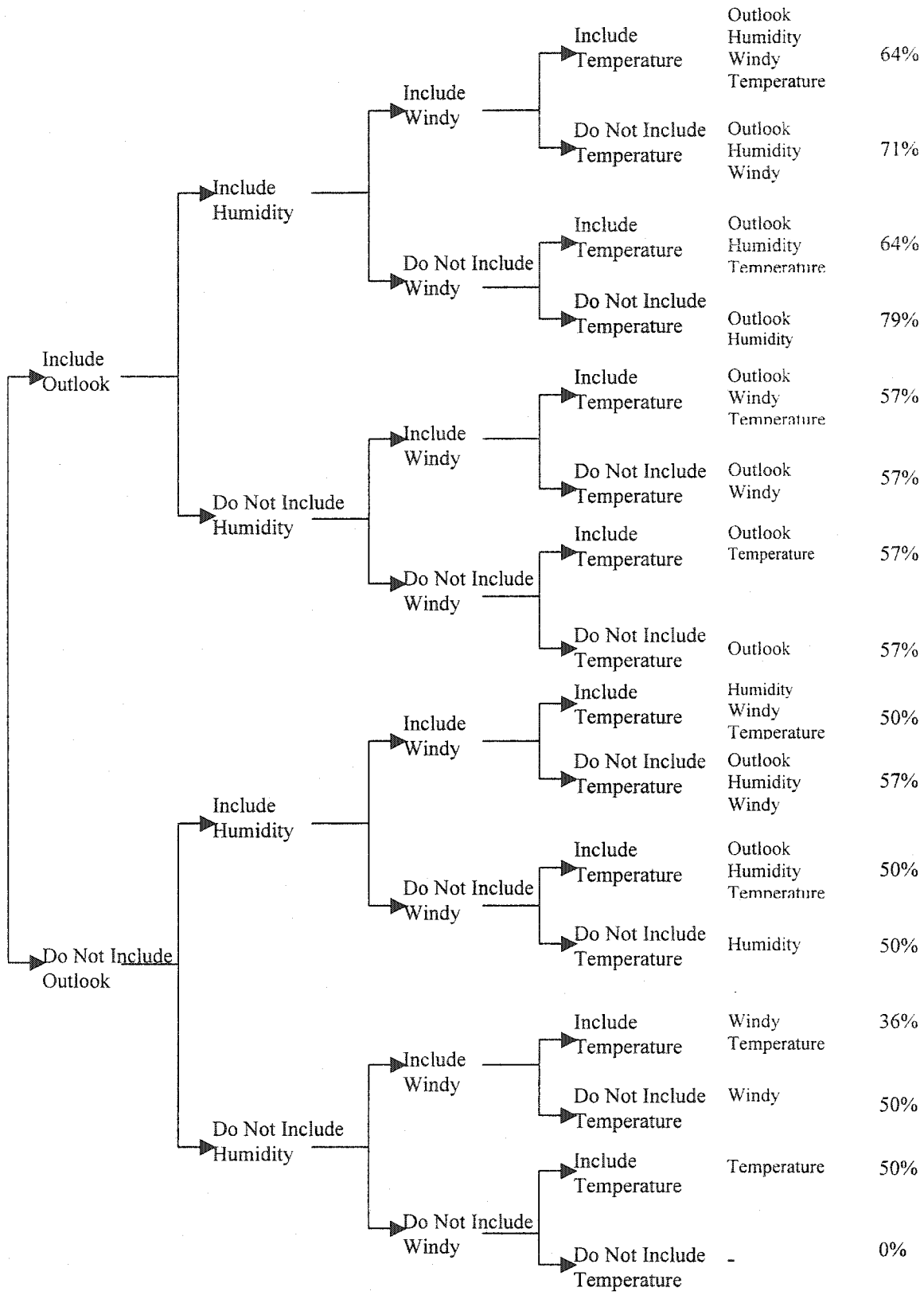


Figure 3.2 Partitioning tree for weather example

Thus, each of the eight feature subsets has a positive probability of being randomly selected, but the probability depends on the perceived information gain of the features in the set.

Say that we select one sample from each region, {*Outlook, Windy, Temperature*} from $A_1(0)$ and {*Humidity, Windy*} from $A_2(0)$. Since

$$f(\{Outlook, Windy, Temperature\}) = 57 = f(\{Humidity, Windy\})$$

a tie must be broken. Given the goals of feature selection, we adopt the rule to break ties by favoring the smaller set, that is {*Humidity, Windy*}, so in the next iteration $A(1) = A_2(0)$ and this new most promising region is partitioned into two subregion and what remains is aggregated into one set:

$$A_1(1) = \{A \in A : Outlook \notin A, Humidity \in A\},$$

$$A_2(1) = \{A \in A : Outlook \notin A, Humidity \notin A\},$$

$$A_3(1) = A \setminus A(1).$$

A quick glance at Figure 3.2 reveals that this move takes the search away from the optimal solution but by maintaining the surrounding region $A_3(1)$ the algorithm is able to recover. Thus, we obtain one sample from each region, say, {*Humidity, Windy*} from $A_1(1)$, {*Windy*} from $A_2(1)$, and {*Outlook, Humidity*} from $A_3(1)$. As the sample from $A_3(1)$ has the best accuracy, the algorithm backtracks and sets $A(2) = A$.

We are now back to where we started and this time we are likely to select different samples, say {*Outlook, Humidity, Temperature*} from $A_1(2)$ and {*Humidity, Windy*} from $A_2(2)$. This time around

$$f(\{Outlook, Humidity, Temperature\}) = 64 > 57 = f(\{Humidity, Windy\})$$

and the first subset is selected as the most promising region, that is, $A(3) = A_1(2)$. It is partitioned into two subregion and what remains is aggregated as before:

$$A_1(3) = \{A \in A : Outlook \notin A, Humidity \in A\},$$

$$A_2(3) = \{A \in A : Outlook \notin A, Humidity \notin A\},$$

$$A_3(3) = A \setminus A(3).$$

Now note that regardless of which samples are selected from these three regions, the sample from $A_1(3)$ will have the highest accuracy, so $A(4) = A_1(3)$. The new most promising region is partitioned into two subregions and what remains is aggregated into one set:

$A_1(4) = \{\{Outlook, Humidity, Windy, Temperature\}, \{Outlook, Humidity, Windy\}\}$,

$A_2(4) = \{\{\{Outlook, Humidity, Temperature\}, \{Outlook, Humidity\}\}$,

$A_3(3) = A \setminus A(3)$.

Depending on the sampling, either $A_1(4)$ or $A_2(4)$ may be selected, but since there is no overlap between $A_3(4)$ and the `good' region $A_2(4)$, backtracking will not be warranted by the sampling.

From these first four iterations, it is clear that the sequence of most promising regions moves towards the optimum with high probability and has the potential to recover from wrong moves via backtracking. As the algorithm progresses the sampling, and thus the computational effort, is concentrated where good feature subsets are likely to be found. Ones the maximum depth is reached, that is, the current most promising region is a singleton, the algorithm stops. Although simple, this example thus illustrates many key aspects of the NP-Wrapper, including how it converges, how the computational effort is focused in most promising regions, the value of backtracking, and the paramount importance of an intelligent partitioning strategy. In the next section we evaluate both the NP-Wrapper and NP-Filter numerically.

3.5 Evaluation of the NP-Wrapper and NP-Filter

In this section we present numerical results for tests of the NP-Wrapper and NP-Filter when used to precede two classification algorithms, namely the Naïve Bayes algorithm and the C4.5 decision tree induction algorithm. These tests were conducted on a Dell OptiPlex GX200 with Intel Pentium III 662 MHz and Memory 64 Mb. The code is written in Java using the *Weka* machine learning software library (Witten et al., 1999) for implementation of the learning algorithms themselves. We used five data sets from the *UCI Repository of*

machine learning databases (Blake and Merz, 1998). The characteristics of these data sets are shown in Table 3.2, from which we note that the sizes range from 148 to 3196 instances and from 9 to 69 features. As both the NP-Filter and NP-Wrapper are randomized algorithms, we run five replications for each experiment and report both the average and the standard error.

Table 3.2: Characteristics of the tested data sets

Data Set	Instances	Features
lymph	148	18
vote	435	16
audiology	226	69
cancer	286	9
kr-vs-kp	3196	36

3.5.1 Value of Feature Selection

Our first set of experiments addresses the effectiveness of feature selection using the NP-Filter and NP-Wrapper for the selected data sets. As noted before, both Naïve Bayes and C4.5 are used to induce classification models with the selected features. We measure the effectiveness along two dimensions. First we consider the accuracy of the models induced after feature selection compared to the corresponding models without feature selection, and second we consider how many features are eliminated, that is, how much smaller the models become when feature selection is employed.

Table 3.3: Accuracy of Naive Bayes with and without feature selection

Data Set	NFS		NPF		NPW	
	Accuracy	Size	Accuracy	Size	Accuracy	Size
lymph	85.1	18	85.4±1.0	10.6±2.1	86.2±0.8	9.2±0.8
Vote	90.1	16	93.2±1.0	6.8±1.1	95.8±0.4	3.0±1.4
audiology	71.2	69	71.2±1.5	27.4±3.2	75.0±2.3	23.0±3.9
cancer	73.4	9	73.8±0.4	5.8±0.8	75.7±0.2	3.6±0.9
kr-vs-kp	88.0	36	90.8±2.1	11.6±1.5	94.4±0.3	14.2±3.8

The results for the Naïve Bayes classification method are shown in Table 3.3. Columns 2-3 show the results for no feature selection (NFS), columns 4-5 show results for the NP-Filter (NPF), and finally columns 6-7 for the NP-Wrapper (NPW). First looking at the accuracy we note that it actually improves or is no worse when we use feature selection, and

the models where classification is preceded by a NP-Wrapper have the highest accuracy. Indeed, there is an average of 1.5% improvement in accuracy when we precede a Naïve Bayes method with NP-Filter, and 4.7% when it is preceded with a NP-Wrapper feature selection. As discussed in the introduction, such improvement in accuracy may or may not occur when feature selection is employed. In particular, the performance of Naïve Bayes is known to be degraded by redundant features and it appears that those are effectively eliminated by both feature selection algorithms. What we do, however, always expect from a feature selection procedure is a significant reduction in the number of features, resulting in simpler and easier to explain models. Table 3.3 demonstrates this reduction. For example, when the NP-Filter is used, the 69 features of the 'audiology' data set are reduced to an average of 27.4 features, and when the NP-Wrapper is used, they are reduced to an average 23.0 features. This is a significant simplification of the models. Across all the data set there is an average 52% reduction in number of features when we use the NP-Filter and an average of 64% fewer features when we use the NP-Wrapper. We note that the NP-Wrapper performs better on both the accuracy and simplicity measures.

Table 3.4: Accuracy of C4.5 with an without feature selection

Data Set	NFS		NPF		NPW	
	Accuracy	Size	Accuracy	Size	Accuracy	Size
lymph	78.4	18	78.1±1.4	9.6±0.8	82.2±0.4	7.6±1.7
vote	96.5	16	95.7±0.2	6.0±2.0	96.6±0.6	3.8±1.6
audiology	77.4	69	75.0±1.6	25.0±4.3	79.9±1.1	14.0±2.7
cancer	75.5	9	73.7±0.2	5.2±0.5	76.2±0.6	2.4±0.9
kr-vs-kp	99.1	36	94.0±1.1	11.6±1.3	96.5±0.9	17.0±2.7

We repeat the same experiments for the C4.5 decision tree induction algorithm, with the results reported in Table 3.4 according to the same format as before. Here the accuracy of the model is actually degraded somewhat when the NP-Filter is used (average of 2.4%), but using the NP-Wrapper still results in higher accuracy models than using all of the features for all but one of the data sets (average of 1.3%). The reduction in the number of features, however, is even higher than before, with an average of 57% and 68% reduction for the NP-Filter and NP-Wrapper, respectively.

3.5.2 Comparison with Simple Entropy Filter

The section demonstrates the value of using the NP-Filter and NP-Wrapper for feature selection, as this results in smaller but often higher accuracy models. However, the question arises if this performance is primarily due to properties of the NP method or because it incorporates an information gain ranking of features that is known to perform quite well in practice. Indeed, the selection of this approach to define the intelligent partitioning is based on the expectation that a reasonable ranking of features can be obtained by considering their information gain.

Table 3.5: Accuracy of Naive Bayes with early termination of feature selection

Data Set	Depth	NPF		NPW		EF	
		Accuracy	Size	Accuracy	Size	Accurac	Size
lymph	Max	84.7±1.6	11.6±1.1	86.9±1.9	9.6±0.6	81.8	11
vote		93.0±0.5	6.8±0.8	95.0±0.6	4.8±1.3	89.9	10
audiology		69.7±0.7	28.6±3.3	73.6±1.1	30.0±3.0	75.2	43
cancer		73.7±0.3	5.4±0.6	75.5±0.5	2.8±0.8	74.1	6
kr-vs-kp		90.7±1.2	11.6±0.9	94.1±0.4	13.0±1.6	88.1	23
lymph	Avg	83.9±0.7	10.6±0.9	86.4±0.3	11.2±1.1	81.8	7
vote		93.2±0.4	8.2±1.1	94.7±0.8	3.8±0.8	92.4	6
audiology		71.0±2.7	24.8±3.5	74.8±2.8	26.6±4.3	73.5	26
cancer		73.6±0.3	4.8±0.5	75.4±0.8	3.0±1.7	72.7	3
kr-vs-kp		90.8±2.8	10.0±2.1	93.2±0.7	12.4±1.5	89.9	14
lymph	Min	85.7±0.7	11.8±1.6	86.0±1.8	9.4±0.9	80.4	2
vote		99.0±1.4	5.4±2.5	94.3±0.5	5.8±0.5	95.6	2
audiology		70.5±3.1	11.6±1.8	73.5±2.5	14.0±4.3	67.7	9
cancer		73.7±0.4	4.6±0.6	74.3±0.9	4.2±1.6	72.0	1
kr-vs-kp		90.7±1.3	7.6±0.9	94.2±0.1	7.4±1.7	86.7	5

To evaluate the contribution of the NP method versus that of simply using the information gain ranking, we compare the performance of the NP-Filter and NP-Wrapper with a filter that we refer to as the Entropy-Filter (EF). This filter simply selects the features with the highest information gain to be included in the model. Since the number of features used by the solutions found by the NP-Filter and NP-Wrapper is not fixed, a comparison of models with the exact same number of features is not possible. Furthermore, by fixing the number of features to be selected the Entropy-Filter only considers this many features, whereas by terminating the NP algorithms at maximum depth it is assured that every feature

is considered. Thus, for a fairer comparison we change the stopping criterion of the NP algorithms to that we terminate when a certain depth is reached, that is, after the given number of features has been considered for inclusion in the set.

Table 3.6: Accuracy of C4.5 with early termination of feature selection

Data Set	Depth	NPF		NPW		EF	
		Accuracy	Size	Accuracy	Size	Accurac	Size
lymph	Max	78.1±0.8	12.0±0.7	82.2±1.0	8.4±2.3	76.4	9
vote		95.5±0.2	6.4±1.8	96.4±0.7	6.0±2.6	95.6	8
audiology		76.4±0.9	28.6±3.5	79.6±1.7	27.4±4.2	77.9	36
cancer		73.4±0.0	6.0±0.0	75.9±0.0	4.2±0.5	73.8	5
kr-vs-kp		91.6±3.6	10.0±0.7	97.1±0.3	17.2±2.8	97.1	19
lymph	Avg	79.1±1.4	10.6±1.1	81.1±0.5	9.0±0.7	78.4	6
vote		96.6±0.2	8.2±1.6	96.3±0.5	7.2±1.9	95.6	5
audiology		74.6±2.9	27.2±2.2	79.5±1.6	30.4±2.5	77.9	22
cancer		73.8±0.0	5.0±0.0	75.9±0.0	2.6±0.9	71.7	3
kr-vs-kp		91.9±4.2	11.8±2.1	97.1±0.3	1.8±0.8	96.5	12
lymph	Min	78.7±0.6	9.0±0.7	81.1±0.5	9.8±1.9	73.7	2
vote		95.5±0.3	6.2±1.9	96.3±0.5	7.2±1.9	95.6	2
audiology		73.2±2.1	13.0±3.1	77.2±1.5	16.4±2.4	69.9	8
cancer		72.9±1.2	4.6±0.6	74.1±1.1	3.2±1.3	69.6	1
kr-vs-kp		87.2±6.2	6.6±0.9	96.2±0.9	13.2±3.1	90.4	4

The same number of features is then used by the Entropy-Filter. As the data sets have varying number of features but we want a common testing procedure, we consider using approximately 60%, 40%, and 15% of the features in three different experiments. The results for Naïve Bayes are shown in Table 3.5. The first set of results for each data set uses 60% of features (Max), the second set 40% (Avg), and the third 15% (Min). From these results we see that the simple Entropy-Filter actually performs quite well, but on the average both the NP-Filter and NP-Wrapper perform significantly better with respect to accuracy. The average accuracy of the NP-Filter is better for all of the problems except the 'audiology' test set, with the average improvement ranging from 1.0% to 4.2%. The average accuracy of the NP-Wrapper is better for all of the problems, and in 13 out of 15 experiments the average accuracy of the NP-Wrapper is better than for both of the other methods. The average improvement for each problem ranges from 2.2% for the 'vote' data set to 6.3% for the

'lymph' and 'kr-vs-kp' data sets, and there is no apparent pattern with respect to the size of the problems.

The same results for the C4.5 decision tree algorithm are shown in Table 3.6. The results here are similar, except that the NP-Filter performs relatively worse, with an average improvement for only three out of the five problems, namely for the 'lymph', 'vote', and 'cancer' data sets. Again, the NP-Wrapper has the best performance for 13 out of 15 experiments and the average improvement over the Entropy-Filter ranges from only 1% for the 'vote' data set to 7% for the 'lymph' data set.

Table 3.7: Average speed using Naive Bayes (milliseconds)

Data Set	Depth	NFS	NPF	NPW	EF
lymph	Full	114	5313	6113	N/A
	Max	N/A	3954	3833	120
	Avg	N/A	2557	2628	120
	Min	N/A	1190	853	112
vote	Full	164	11541	13341	N/A
	Max	N/A	8096	8711	180
	Avg	N/A	5416	5350	178
	Min	N/A	1791	1915	154
audiology	Full	370	124205	127769	N/A
	Max	N/A	88125	77452	322
	Avg	N/A	52583	43256	297
	Min	N/A	16049	16387	252
cancer	Full	119	2390	2888	N/A
	Max	N/A	1679	1987	124
	Avg	N/A	1232	1064	124
	Min	N/A	573	431	126
kr-vs-kp	Full	886	410682	515299	N/A
	Max	N/A	272552	315520	1294
	Avg	N/A	160109	192314	1298
	Min	N/A	50278	63697	1262

The improved accuracy obtained by using the NP algorithms, and especially the NP-Wrapper, does of course come at a price, which is increased computational time. In Table 3.7 and Table 3.8 we report the amount of computational time (in millseconds) used by each of the algorithms for all of the experiments reported above. For each of the data sets, the first line reports the time used for the experiments reported in Table 3.3 and Table 3.4 for no

feature selection (NFS), the NP-Filter (NPF) and NP-Wrapper (NPW). The next three lines report the time used for the experiments reported in Table 3.5 and Table 3.6 for the NP-Filter (NPF), NP-Wrapper (NPW), and Entropy-Filter (EF).

Table 3.8: Average speed using C4.5 (milliseconds)

Data Set	Depth	NFS	NPF	NPW	EF
lymph	Full	304	5476	28929	N/A
	Max	N/A	4210	14188	270
	Avg	N/A	2774	9514	276
	Min	N/A	1402	2912	240
vote	Full	392	11685	41708	N/A
	Max	N/A	8238	22178	356
	Avg	N/A	5588	13890	302
	Min	N/A	2017	5454	222
audiology	Full	1076	127100	327725	N/A
	Max	N/A	92225	172884	845
	Avg	N/A	54246	111969	709
	Min	N/A	16622	32048	488
cancer	Full	373	2650	8142	N/A
	Max	N/A	1985	4877	303
	Avg	N/A	1538	2940	284
	Min	N/A	801	1031	188
kr-vs-kp	Full	4558	430171	1836724	N/A
	Max	N/A	276450	614206	3509
	Avg	N/A	189066	382934	2794
	Min	N/A	54348	101643	1772

From these results we see that using the Entropy-Filter takes the least amount of time, followed by using no feature selection at all. Thus, even though using the Entropy-Filter adds a step to the process, this is more that compensated for by the faster induction of the classification model that occurs when fewer features are employed. The NP-Wrapper takes by far the most amount of computation time and the NP-Filter falls between the NP-Wrapper and no feature selection. Thus, as stated before, the NP algorithmic approach is primarily appropriate when significant computational time can be devoted to obtain high quality feature subsets that are to be used on their own for structural information or can be used repeatedly for classification or other learning. If speed is the primary concern, the much simpler Entropy-Filter is superior.

3.5.3 Importance of Intelligent Partitioning

From the last subsection we know that the high accuracy obtained using the NP-Filter and NP-Wrapper is not completely explained by the use of an information gain ranking. Thus, conversely, we can ask how much of these good results is due to the generic NP framework itself and how much can be contributed to the intelligent partitioning scheme developed in this dissertation. To address this, we compare NP algorithms using the intelligent partitioning to NP algorithms using all other possible ways of partitioning. Since a partition is defined by the order in which features are either included or not, this implies considering all possible orders of the features.

Table 3.9: Accuracy of intelligent partitioning in NP-Wrapper using Naive Bayes

Data Set		Accuracy		
		Intelligent	Best	Worst
vote	1	90.8±0.3	91.0	86.4
	2	95.9±0.0	95.9	94.3
	3	89.0±0.0	89.0	87.4
	4	90.0±0.3	90.1	85.1
	5	95.6±0.0	95.6	92.0
cancer	1	75.9±0.0	75.9	72.7
	2	75.7±0.0	75.9	72.7
	3	75.8±0.2	75.9	73.1
	4	72.8±0.3	73.1	70.6
	5	75.9±0.0	75.9	72.0

In particular, we use a complete enumeration of all partitions to find the best and worst one, and compare those to the intelligent partitioning. Since the number of ways in which the features can be ordered is $n!$ where n is the number of features, a study of even the smallest test problem would involve considering $9! = 362880$ different partitions. This is prohibitively time consuming and we thus modify our data sets so that we first draw a sample of 7 features and then apply the NP algorithms. Note that we still have to evaluate 5040 different partitions. To assure the sampling does not introduce a bias we repeat the process five times, each time drawing an independent sample of 7 features. For those experiments we restrict ourselves to the 'vote' and 'cancer' data sets and we only consider the NP-Wrapper with

Naïve Bayes classification. Results for other configurations are similar and are omitted here for brevity.

Table 3.10: Speed of intelligent partitioning in NP-Wrapper using Naive Bayes

Data Set		Computation Time		
		Intelligent	Slowest	Fastest
vote	1	3812	20370	2814
	2	3515	19408	2153
	3	3371	35080	2784
	4	3690	18046	2774
	5	3433	17375	3094
cancer	1	2740	14050	1382
	2	2624	16013	1372
	3	2664	20390	1342
	4	4969	25226	1362
	5	2642	6920	1372

In Table 3.9 the prediction accuracy of the models using intelligent partition, and the best and worst partition found using enumeration are reported. We note that the accuracy found using the intelligent partition is very close to the optimal. In particular, for half of the problems the intelligent partitioning always results in the same accuracy as the optimal partition, and for the other half the performance is within one standard deviation. On the other hand, we note that partitioning poorly results in feature subsets that have significantly lower accuracy but even for the worst possible partition the NP method is still able to obtain fairly high quality subsets.

In addition to the accuracy, we also compare the computational time used by the NP-Wrapper if different partitioning schemes are used, These results are reported in Table 3.10 and we see that again using the intelligent partitioning results in performance that is fairly close to the optimal, although this time there is more different than with respect to accuracy. In particular, the intelligent partitioning takes, on the average, 36% and 75% longer than the best, for the 'vote' and 'cancer' data sets, respectively. On the other hand, the worst partition takes, on the average, 6.24 and 5.32 times longer than the intelligent partitioning, for the 'vote' and 'cancer' data sets, respectively. Thus, we can conclude that the NP-Filter is capable of compensating fairly well for poor partitions in terms of obtaining accurate models,

but this occurs at the expense of using very long computation time. The intuitive reason for this is that any NP algorithm can compensate for mistakes, that moves in the wrong direction, by backtracking, but frequent backtracking is time consuming and will slow the search significantly. We conclude that a good partition is important with respect to both obtaining high accuracy models and in the time it takes to find the appropriate feature subsets, and of the two the latter is by far the most significant. Finally, we note that the difficult of obtaining the optimal partition is in general equal to solving the problem itself. However, our results show that very high quality partition can be obtained efficiently with the new intelligent partitioning method.

3.6 Comparison with Other Feature Selection Methods

Genetic algorithms (GA) are similar to the NP method in that they use a randomized search strategy to explore the set of alternatives, in this case all possible subsets of features. They have also been shown to perform well for the feature selection [Yang and Honovar, 1998]. Genetic algorithms thus provide a useful benchmark for comparing the performance of the new methodology.

These GA comparisons use Naive Bayes as the classifier. The maximum depth for the NP-Filter is taken to be the minimum depth as explained in Section 3.5.2 above, and as before 5 replications are run for each algorithm. The NP-based algorithms and the GA algorithms are allowed to run for the same amount of time and the GA parameters were selected for best overall performance. Thus, NP and GA are compared only in terms of solution quality, that is model accuracy, and the size of the selected subsets.

The first experiments compare the NP-Filter with GA search that uses the same evaluation criterion, that is, a corresponding GA-Filter. The results are reported in Table 3.11. The accuracy of the sets obtained by the two algorithms appears to be quite similar. Although the average accuracy obtained by NP-Filter is strictly better for all five of the test sets, the difference is only statistically significant for the `vote' data. The difference in performance may be explained by the fact that the NP-Filter tends to select slightly larger feature subsets for all but one of the data sets.

Table 3.11 Comparison of NP and GA Filters.

Data Set	NPF		GA Filter	
	Accuracy	Size	Accuracy	Size
lymph	85.7±0.7	11.8±1.6	84.3±1.3	9.8±1.1
vote	99.0±1.4	5.4±2.5	94.3±1.3	3.8±1.3
audiology	70.5±3.1	11.6±1.8	69.6±0.9	9.0±2.3
cancer	73.7±0.4	4.6±0.6	73.6±0.2	5.6±1.1
kr-vs-kp	90.7±1.3	7.6±0.9	89.8±1.4	4.8±1.3

A similar comparison with the NP-Filter and a corresponding GA-Wrapper is reported in Table 3.12, and the results are similar to the filter results. The NP-Wrapper has higher accuracy for three out of the five data sets, the GA-Wrapper is better for one, and the two are tied for the `cancer' data set. However, none of these differences are statistically significant.

Table 3.12 Comparison of NP and GA Wrappers.

Data Set	NPF		GA Filter	
	Accuracy	Size	Accuracy	Size
lymph	86.0±1.8	9.4±0.9	84.9±1.6	12.0±0.7
vote	94.3±0.5	5.8±0.5	95.1±0.8	5.6±1.5
audiology	73.5±2.5	14.0±4.3	72.0±1.7	38.0±9.3
cancer	74.3±0.9	4.2±1.6	74.3±0.6	5.0±1.4
kr-vs-kp	94.2±0.1	7.4±1.7	92.4±0.7	19.3±3.1

We conclude that the new methodology is a promising alternative and is certainly competitive to other methods such as GA that produce high quality feature subsets. However, we do not expect the NP-based methods to outperform GA for every data set.

3.7 Scalability of Feature Selection

In this section, we discuss the scalability of this approach in terms of its ability to handle increasing number of instances and increasing number of features.

3.7.1 Instance Dimension

As data mining is applied to every larger databases it becomes critical for any data mining method to be able to effectively deal with very large number of data objects or instances [Liu and Setiono, 1998]. One way of accomplishing this is to base the learning not

on the entire database, but on a randomly selected subset as in the NP-Filter, where in each iteration $A(k)$ is such a subset. This has been effectively used in data mining before, for example in the CLARANS clustering algorithm [Ng and Han, 1994]. The use of sampling of instances in data mining to improve scalability can be also found in many researchers' works. However, by using random sampling there is a danger in introducing a new bias into the learning and it is therefore essential to explicitly account for the noise introduced by sampling.

The NP method was originally conceived for simulation-based optimization and is therefore naturally consistent with using performance estimates that are noisy due to sampling. Indeed, in the NP-Filter, a new set $A(k)$ of instances is sampled in each iteration in such a way that this set is independent of the previous sets: $A(0)$, $A(1)$, ..., $A(k - 1)$. Thus, if the new instances indicate an erroneous decision has been made the backtracking feature of the NP method enables the algorithm to make corrections, thus correcting the potential bias.

The question still remains as of how large of a portion of the database is needed by the NP method. In particular, as the proportion is decreased and more backtracking is required then as some point the computational inefficiencies of backtracking will outweigh the savings obtained by using fewer instances. In particular, the expected complexity of the algorithm may increase (see Section 3.4.5 above).

To evaluate these questions empirically, we apply the NP-Filter three well known data sets that are described in Table 3.2. We evaluate the estimated accuracy as well as the computation time when either 2%, 5%, 10%, 20%, 40%, 80%, or 100% of the instances is used by the NP-Filter. For example, when testing the 'vote' data with 20%, we set $\nu(435) = 0.2 \cdot 435 = 87$ instances. Other parameters are set as follows. The sampling effort is constant $N(\psi, \delta) = 5$ so ψ and δ need not be specified, the stopping depth is maximum depth $d_{stop}(n) = n$, the order $a_{[1]}, a_{[2]}, \ldots, a_{[n]}$ is determined by the information gain.

The results are reported as average and estimated standard deviation over five replications, and are shown in Table 3.13. First note that the desired speedups in the algorithm are indeed achieved. By using 10% of the instances rather than 100% of the instance, the computing time is reduced by 71%, 13%, 39%, 93%, and 28% for the five data sets, respectively. Due to very high variance, however, we cannot say that this difference is

significant for the 'audiology' data set. Even though we accomplished the speedup, the accuracy (performance) was not significantly sacrificed for the all data sets. In the case of the 'kr-vs-kp' data set, the accuracy even increased. These are encouraging results. These results illustrate that sampling of instances is a reasonable way to improve the scalability of the NP-Filter with respect to large number of instances. However, the effectiveness of this approach will in general depend on the particular data set being analyzed.

Table 3.13. Effect of using fraction of instance space

Data Set	Fraction	Accuracy	Speed (millisec)	Backtracking
vote	100%	93.5±0.4	2820±93	0.0±0.0
	80%	92.8±0.6	2766±224	0.0±0.0
	40%	92.2±0.5	1694±352	0.0±0.0
	20%	92.6±1.3	1065±174	0.6±0.5
	10%	92.4±1.0	816±167	1.6±2.2
	5%	91.9±1.7	947±515	13.2±18.5
	2%	92.6±1.1	1314±728	90.4±66.7
audiology	100%	69.7±1.9	41105±3255	0.0±0.0
	80%	70.2±1.9	58230±18616	78.8±66.8
	40%	70.2±2.3	38462±3451	108.6±13.3
	20%	70.5±1.0	37280±26368	235.0±214.6
	10%	69.2±2.4	35840±14563	371.0±182.2
	5%	69.6±1.9	37025±14612	566.2±279.1
cancer	100%	73.2±0.6	795±83	0.0±0.0
	80%	73.6±0.3	793±26	0.8±1.3
	40%	73.0±0.8	647±142	1.8±1.5
	20%	73.3±0.8	640±140	3.8±4.1
	10%	72.6±1.2	486±89	7.4±3.4
	5%	73.1±0.4	947±456	78.4±48.8
kr-vs-kp	100%	87.9±5.7	107467±8287	0.0±0.0
	80%	87.3±7.3	87687±12209	0.0±0.0
	40%	89.8±3.1	47741±4359	0.0±0.0
	20%	86.1±4.4	19384±2727	0.4±0.9
	10%	91.1±3.6	11482±2074	0.2±0.4
	5%	89.0±1.2	7246±809	1.8±3.0
	2%	88.6±2.3	7742±1892	25.8±15.6
lymph	100%	83.3±1.2	1734±38	0.0±0.0
	80%	84.2±1.3	2289±545	0.0±0.0
	40%	84.3±2.0	1512±344	2.6±3.8
	20%	84.7±1.0	1013±182	2.0±1.6
	10%	84.5±1.1	1248±385	30.0±16.0
	5%	83.1±2.1	28104±21543	2655±2122.0

One more result we should note is that for all of the data sets except 'kr-vs-kp', the variability of the performance increases significantly. For example when looking at the 'audiology' data set, although the average performance and the estimated standard deviation

do not change significantly (69.7% versus 69.6%) when using only 5% of the instances because of good features of the NP method, the estimated standard deviation of performances in an iteration goes up substantially as the sampling rate of the instances goes down, that results in high number of backtrackings (0.0 versus 566.2). For all data sets above, when the proportion of instances is very low, the variability of the performances gets bigger. This is to be expected as using fewer instances corresponds to the performance estimates used by the algorithm being more noisy. This issue will be dealt with in Chapter 4 in detail.

3.7.2 Feature Dimension

The ability of a feature selection method to effectively scale up when the number of features increases is very important, especially when the primary purpose of using feature selection is dimensionality reduction.

The NP has a very natural mechanism to handle large number of features, namely the depth of partitioning tree. Table 3.5 ~ Table 3.8 show that even though we terminate the algorithm early, the accuracy is not sacrificed while the computational time is significantly reduced as described in the previous section. This issue will be also addressed in the Chapter 4 in detail

3.7.3 Adapting to New Features

Another important scalability property is the ability of the data mining algorithm to effectively adapt to the introduction of new features. That is, assuming that a solution has already been obtained and a new feature is introduced (for example with a new product being added to an auction system), then can the problem be solved faster than by simply starting from scratch.

In the NP-Filter, the simplest approach would be to simply add the new feature to the set of features that have not yet been fixed by the partitioning and select it as soon as warranted. This approach seems highly scalable. There is insignificant overhead and minimal interruption for the algorithm. There are, on the other hand, a couple of potential drawbacks to this approach. First, when a new feature is added the performance of existing subregions may change and thus, the current most promising region may not be the one that should have been selected. This should not be a serious drawback, however, as backtracking should

automatically correct any such errors. Secondly, it may be that the new feature should have been selected higher up in the tree. Again, the effects of this are not clear and backtracking always has the capability of moving up to the appropriate spot in the tree and rectifying the problem. However, these corrections may cause significant amount of backtracking that increases the computation time.

Following this discussion, we evaluate three options for dynamically adding a new feature to an existing solution: (i) Prune the tree up to the level at which the feature would have been fixed (according to its entropy value), and start over from there; (ii) add the new feature at the lowest depth without interrupting already fixed features; and (iii) start the algorithm over in which case the feature will be fixed as all others according to its information gain. Note that the last option 'Lowest Depth' requires no disruption to the current partitioning tree. Numerical results for three data sets adapted from the 'vote' data set, illustrating these three strategies are reported in Table 3.14. In each of these sets one random feature is held back, the NP-Filter applied to the remaining set and the random feature then added. As before, we assume that the selected feature subsets are to be used by the Naïve Bayes algorithm and the reported accuracy is the estimated accuracy in this case. The parameters for this problem are as follows;

$$
\begin{aligned}
n &= 435 \\
m &= 16 \\
N(\psi,\delta) &= 5 \\
d_{(stop)}(n) &= 16 \\
v(n) &= 435.
\end{aligned}
$$

All numbers are reported as average and estimated standard deviation from five replications. From these results it is clear that in terms of accuracy all strategies are indistinguishable. Thus, a selection between strategies can be made exclusively based on computing efficiency. The hypothesis is that the proposed strategies can improve the amount of computing time needed from the benchmark of simply starting all the calculations over then a new feature must be accounted for.

Table 3.14: Dynamically adding a new feature

Data Set	Strategy	Accuracy	Speed
Set 1	Start Over	92.7±0.4	12396±151
	Correct Depth	93.0±1.3	4204±88
	Last Depth	92.8±1.1	1312±24
Set 2	Start Over	93.4±0.9	12625±286
	Correct Depth	93.3±0.8	5678±134
	Last Depth	93.0±0.8	1308±21
Set 3	Start Over	93.7±1.4	12207±94
	Correct Depth	93.2±0.7	7336±94
	Last Depth	93.4±1.7	1304±33

Indeed, the data supports that both strategies are capable of such improvements. The strategy of using part of the current tree and inserting the new feature in its proper order according to its entropy value reduce the computing time by 66%, 55%, and 40%, respectively. On the other hand, by simply adding the new feature at maximum depth, the computing time can be improved by an order of magnitude starting over. With computing times of 1312, 1308, and 1304, this approach also has the most predictable computing time. Thus, we conclude that implementing this strategy will greatly enhancing the scalability of the algorithm when applied in dynamic environments.

3.8 Summary and Discussion

We have developed a new optimization based approach to feature selection that can be implemented as both a filter and a wrapper. The new approach falls within an optimization framework that in previous work has been shown to have desirable convergence properties, such as asymptotic convergence and guarantee of being with a certain distance of the optimum with a given probability after a finite time stopping criterion is satisfied. Our numerical results show that the new method performs quite well on several test problems.

Using a new optimization-based feature selection methodology called the NP-Filter, we have also demonstrated its scalability. Numerical results show that using sampling is potentially a very effective way to deal with large number of instances as the NP-Filter can use backtracking to correct any bias that may arise and random sampling. Finally, due to the partitioning that fixes one feature at a time as either being included or not, new features can

be dynamically accounted for with computing time that is an order of magnitude faster than starting over.

However, some potential for the scalability of the NP feature selection method can be found in this chapter. As stated previously in the section, we showed that the NP feature selection method has a potential to handle large number of instances by using static random sampling. However, some questions remain about systematic ways for improving scalability using random sampling, which is the focus of Chapter 4.

4 SYSTEMATIC APPROACHES TO SCALABLE FEATURE SELECTION

4.1 Introduction

The ability of a feature selection method to effectively scale up when data mining is applied to every larger databases is very important. In this chapter we evaluate the accuracy and computational time of the NP-Filter as functions of both the number of features and number of instances using both synthetic data sets and realistic data sets. Numerical results are presented to show that using sampling is a very effective way to deal with the large number of instances as the NP-Filter uses backtracking to correct any bias that may arise, It is reported that the NP-Filter may be vulnerable to a rapid increase of the computation time for the increasing number of features. However, as stated in Chapter 3, the feature dimension scalability of NP feature selection methods can be achieved by controlling the depth. Even use of very small depth (e.g. 15%) does not lose an acceptable accuracy level with significant reduction on the computation time. Strictly speaking on the scalability for a large number of instances, even though random sampling is a good strategy for scalable feature selection on the instance dimension, questions on if an optimal solution for a size of instance samples can be found and how many samples are required for minimizing the computation time of the algorithms need to be more investigated.

Based on the those results, we develop two systematic approaches to apply random sampling to improve the scalability of the NP-Filter in terms of its ability to handle the increasing number of instances. All iterations in the nested partitions (NP) method employ random sampling approach that takes samples of instances instead using all training data set. A random sampling procedure could use the fixed number of samples in the Static NP or a different number of samples in each iteration according to some legitimate criteria, which is called the Adaptive NP and addressed in a later section. Here the main issue is how many random samples should be used. Small samples may reduce computational time to calculate performances of a feature subset but increase performance variability of the subsets that may require more random samples. Therefore, it is very important to find an appropriate amount

of samples to improve scalability without sacrificing the accuracy of a finally selected feature subset.

Most prior research on scalability can be broadly grouped into two categories, namely efficient search and data partitioning as described in the literature review chapter. For a fast search, heuristic based algorithms were proposed with approximate solutions. Even though some optimization based algorithms were proposed, they had to employ approximation for compromising expensive time complexity to find an optimal solution. On one hand, instead of focusing on the algorithm search, the data partitioning approach was introduced by dividing the data horizontally (instance sampling) or vertically (feature selection). We use the random sampling approach for rendering the feature selection process scalable that has been frequently used in many research articles.

The remainder of this chapter is organized as follows. Firstly scalability issue on the NP-Filter using synthetic data set and real data set is addressed. Then both analytical and heuristic approaches are presented to find a solution for sample sizes in two different ways. Numerical results are also presented to show how the methods affect the performance in terms of mainly speed and accuracy. Finally empirical comparisons including the Static NP-Filter are reported to show which strategy works well to improve scalability.

4.2 Scalability of NP-Filter

In this section we consider the scalability of the new methodology, the NP-Filter. In particular, we evaluate the accuracy and computational time as functions of both the number of features and number of instances. Ideally, a highly scalable algorithm would achieve linear growth in the computational time while maintaining the acceptable accuracy level.

4.2.1 Scalability Using Synthetic Data Sets

For testifying which factor affects the scalability of the NP method we use synthetically generated test data where both the number of instances and number of features are control parameters. In particular, we generate test sets with 50, 100, 200, 400, and 800 instances, and 50,100, 200, 400, and 800 features using the following approach. To create a single instance i, a value for the class feature Y_i is generated according to a uniform distribution over the interval [-3,3], The value for each of the other features X_{ij} is then generated according to :

$$X_{ij} = \rho_j Y_i + (|\rho_j| - 1) \cdot Z_j \tag{4.1}$$

where ρ_j is the amount of correlation between feature j and the class feature, and Z_j is drawn from a unit normal distribution, j = 1,2,...,n, i = 1,2,...,m. For each of the test problems, 10% of the features are highly correlated with $|\rho_j| \geq 0.9$, 40% have correlation $0.3 \leq |\rho_j| < 0.9$, and 50% of the features do not correlate highly with the class feature, that is $|\rho_j| < 0.3$. The NP-Filter, followed by Naive Bayes classification model induction, is run five times for each of those test sets.

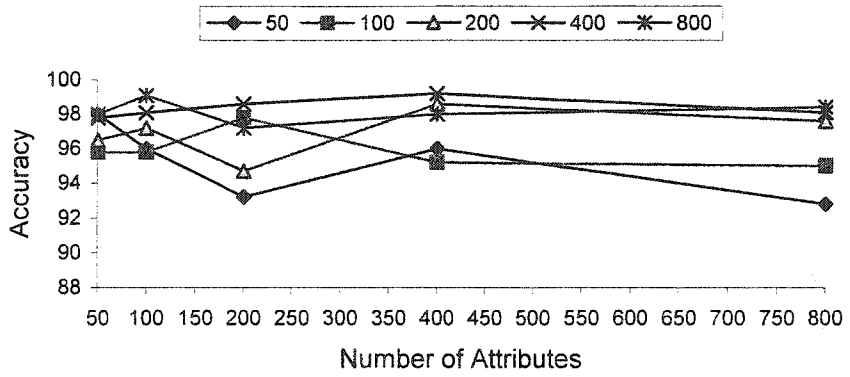


Figure 4.1. Accuracy as a function of features for the five instance settings.

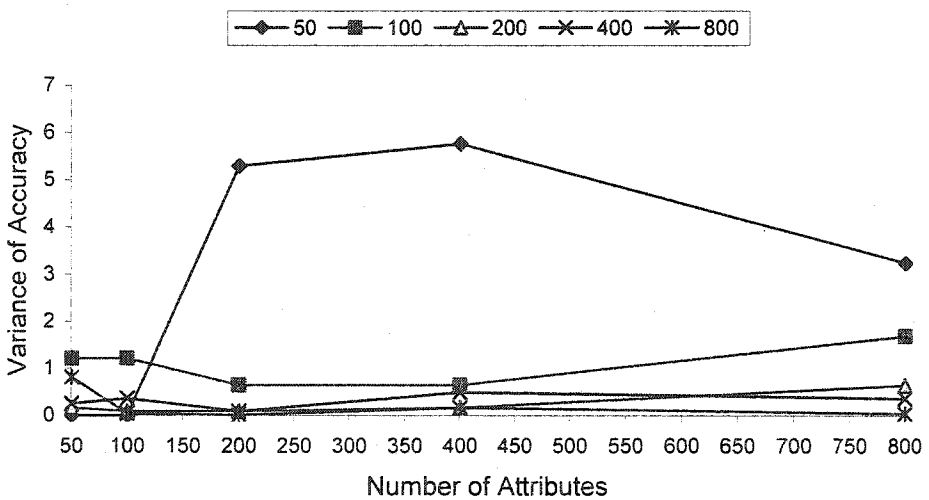


Figure 4.2 Variance of accuracy as a function of features for the five instance settings.

Let's first consider the scalability with respect to the number of features. Figure 4.1 and Figure 4.2 show the accuracy and its variance as a function of number of features for the five instance settings (50 to 800 instances). From these results we conclude that there is no significant change in the accuracy obtained as the number of features grows. The plotted variances of accuracies also imply that significant differences do not happen most of the time.

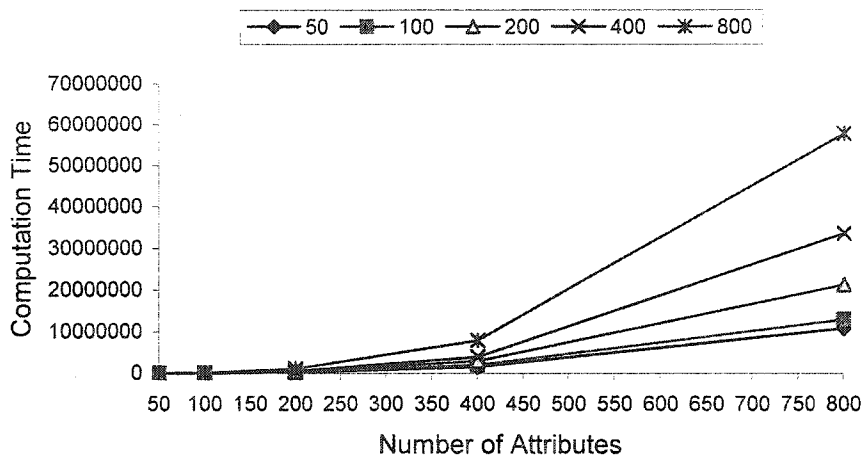


Figure 4.3. Computation time as a function of features for the five instance settings.

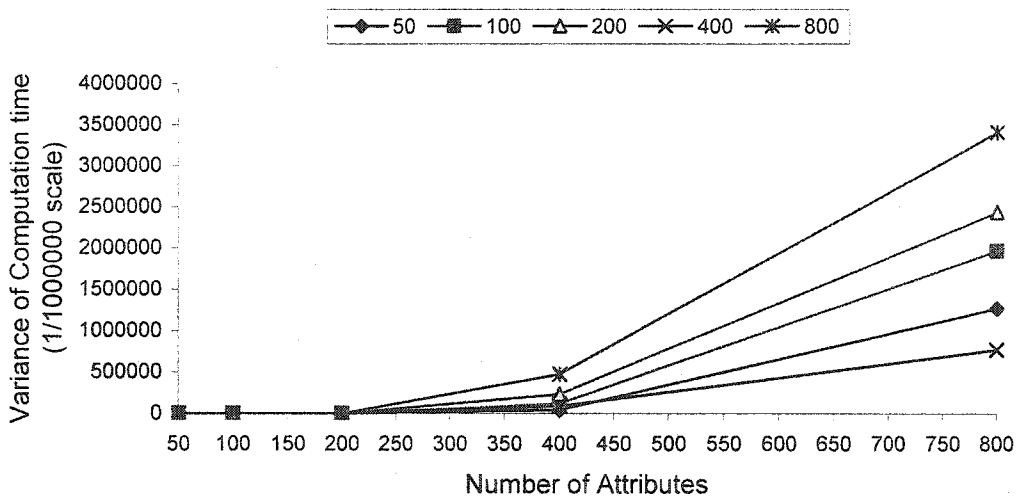


Figure 4.4 Variance of speed as a function of features for the five instance settings.

The results for computational time shown in Figure 4.3 and Figure 4.4 report that the time and its variance grow rapidly as the number of features increases as clearly shown in 400 and 800 instance settings, and indeed it appears to demonstrate exponential growth. Thus, although quality is not lost as the problem size increases, the time it takes to achieve this quality increases quickly and the NP-Filter is therefore somewhat lacking in terms of scalability with respect to the number of features.

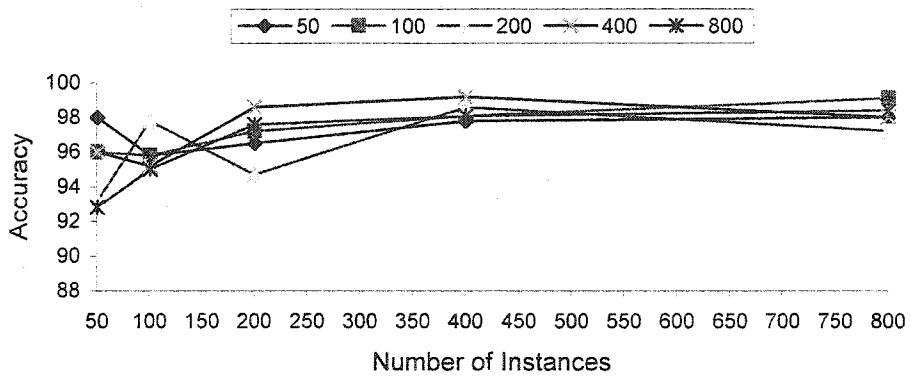


Figure 4.5. Accuracy as a function of instances for the five feature settings.

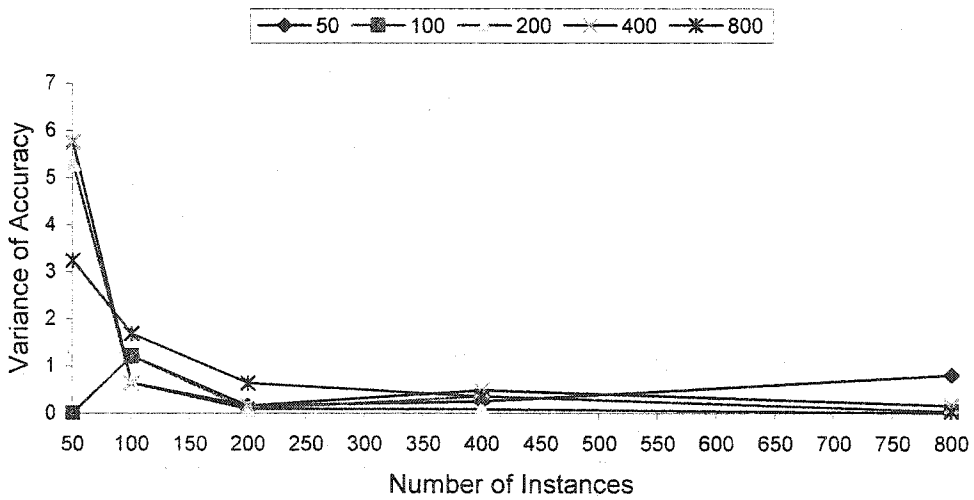


Figure 4.6 Variance of accuracy as a function of instance for the five features settings.

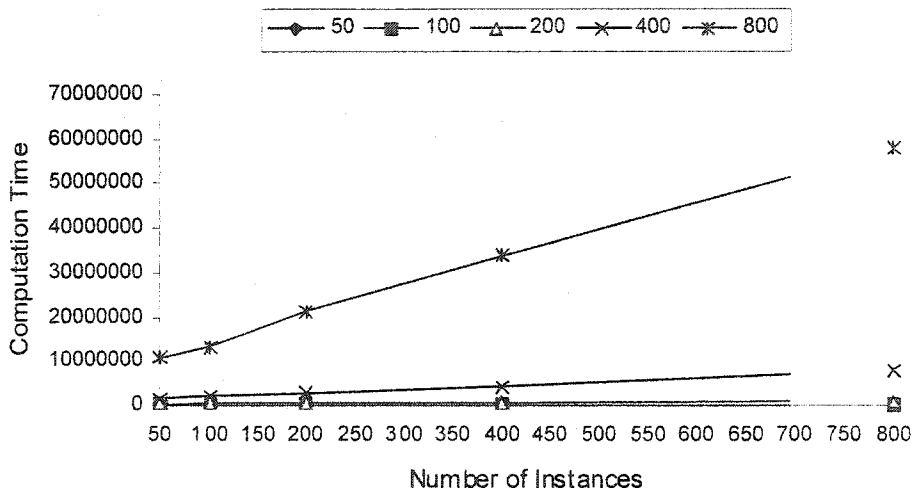


Figure 4.7. Computation time as a function of instances for the five feature settings.

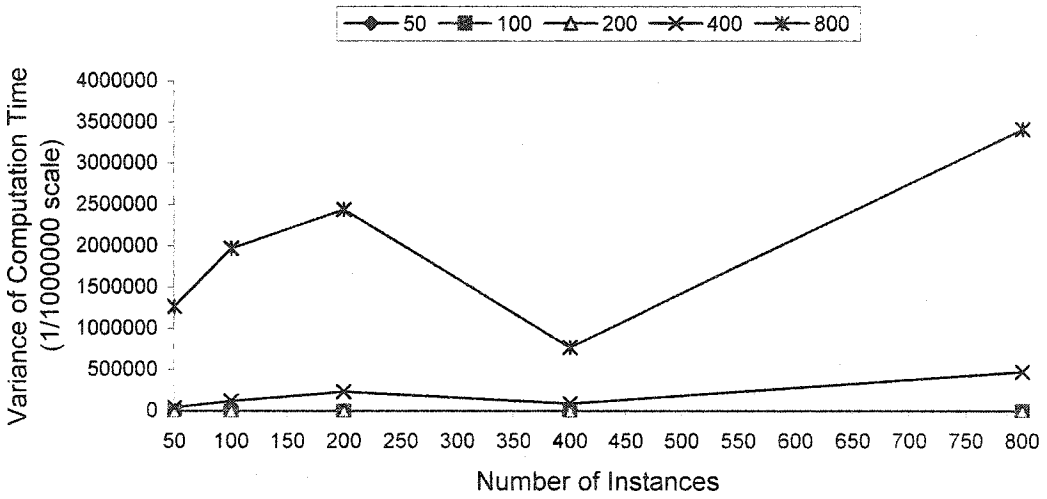


Figure 4.8 Variance of computation time as a function of instances for the five feature settings.

Looking at the scalability of the NP-Filter as a function of number of instances, Figure 4.5 and Figure 4.6 report the accuracy and its variance of accuracy obtained as a function of the number of instances. An interesting observation from these figures are that the solution quality actually improves as the problem size increases, which is not entirely surprising as more instances imply that more data is available to induce a model with high accuracy. Now turning to the computational time shown in Figure 4.7 required to achieve this accuracy, As opposed to the rapid growth in computational time seen when the number of features

increases, the time here grows only linearly, thus implying that the NP-Filter is scalable with respect to the number of instances.

In the NP method, a new set of instances is sampled in each iteration in such a way that this set is independent of the previous set. Thus, if the new instances indicate that an erroneous decision has been made, the backtracking feature of the NP method enables the algorithm to make corrections, thus correcting the potential bias. The question still remains as of how large of a portion of the database is needed by the NP method. As the proportion is decreased, more backtracking is required because at some point the computational inefficiencies of backtracking will outweigh the savings obtained by using fewer instances.

4.2.2 Performance Variances for Instance Sampling Rates

The NP-Filter corrects mistakes made due to noisy performance estimates by backtracking when the error is discovered, so we would expect to see more backtracking when fewer instances are used. This is indeed supported by the data in Table 3.13, as the average number of backtracking moves increases for each of the data sets. Even though it is expected that the speed would become slow as the proportion of instances decreases, the real time increases at the very lower proportion point after it decreases for a while. Excessive backtracking may slow down the NP-Filter. The main reason for excessive backtrackings is that performance variance increases as the number of instances used decreases. The following tables report the results that also illustrate that sampling of instances is a reasonable way to improve the scalability of the NP-Filter with respect to large number of instances.

Performance variance would be affected by both instance and feature variability. In order to show the relationship between performance variances and only instance sampling rates with excluding the feature variability, the following test configuration is needed. For the test, the NP-Filter with modified data sets that contain 7 randomly selected features and 1 class feature is used. In each iteration, enumerated feature data sets ($127 = 2^7 - 1$) are evaluated by the correlation filter and the best one is selected as a current optimal subset (solution). At the end of the last iteration, the variance of performances of best solutions is calculated to determine if there is a relationship between performance variances, and each case experiment is replicated 5 times. It is expected that the variance is zero when the 100%

instances are used and variances with lower portion of instances increase. If we use 100% instances, the algorithm always selects an optimal solution from the enumerated data sets in all iteration, which results in zero variance of performances. On the other hand, if we sample fewer instances, the selected best one at each iteration would be different. Thus the variance would not be zero.

Table 4.1 Performance variances for sampling rates of instances.

Sample Rates	100%	80%	60%	40%	20%	10%	5%	2%
vote1	0.0	1.4	4.0	5.0	8.1	17.3	27.5	N/A
vote2	0.0	6.6	9.1	16.4	28.0	38.3	41.9	N/A
audiology1	0.0	1.5	4.2	6.1	16.9	33.4	48.8	94.3
audiology2	0.0	1.2	1.9	5.3	14.9	25.6	58.7	91.1
audiology3	0.0	0.9	2.4	4.9	10.8	28.7	58.2	185.4
cancer1	0.0	0.7	2.1	4.3	19.2	49.1	109.7	N/A
cancer2	0.0	0.5	1.8	4.5	14.7	52.7	104.7	N/A
cancer3	0.0	0.6	1.4	3.0	13.9	53.4	150.8	N/A
kr-vs-kp1	0.0	0.2	0.4	0.9	2.8	5.9	8.9	14.9
kr-vs-kp2	0.0	0.1	0.1	0.2	0.5	1.2	2.7	6.3
kr-vs-kp3	0.0	0.1	0.1	0.2	0.5	1.2	2.7	9.5
lymph1	0.0	1.0	5.4	10.6	15.4	21.4	25.6	58.8
lymph2	0.0	1.7	4.4	7.8	15.8	26.9	29.2	67.8
lymph3	0.0	1.4	1.5	4.3	11.9	30.0	37.2	78.1

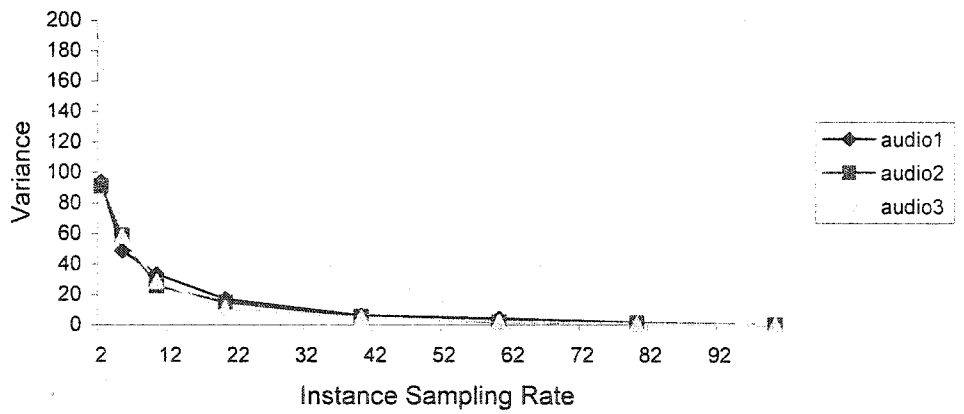


Figure 4.9 Performance variances for instance sampling rates.

As shown the Table 4.1, as the sampling rate decreases, the variance of performances increases exponentially in most cases. The increasing rate of variance is very high at the very

low sampling rate of instances in most data sets. For example, the variances at 80% and 2% sampling rates are 0.9 and 185.4 in one of modified 'audiology' data sets.

From the results above, an importance question arises. What is the optimal proportion point of instances that minimizes the computational time of the NP-Filter? That question is addressed in the following sections. However, the effectiveness of this approach in general depends on the particular data set being analyzed.

4.3 Analytical Approach

In this section we consider an analytical approach to improve the scalability of the new methodology using the results on the relationship between performance variances and instance sampling rates. In order to make a feature selection process scalable, random sampling for instances can be a good strategy as shown previously. Small samples can reduce computational time of the algorithm, but there is a trade-off since that small data set may produce undesirable accuracy of a finally acquired feature subset.

4.3.1 NP/Rinott-Filter

In this section, the NP/Rinott-Filter algorithm is stated, which is an enhanced version of NP-Filter. The NP/Rinott-Filter, which is derived from the NP/Rinott [Olafsson, 2003], employs Rinott's two-stage ranking-and-selection procedure to ensure that the correct best sample set is selected with a correct selection probability. It was originally developed for simulation-based optimization by combining the benefits of global random search and statistical selection, and then may be considered an iterative ranking-and-selection algorithm. Based on the NP-Filter described in the previous chapter, we let N_j denote the number of sample sets in $A_j(k), j = 1, 2, 3.$, j-th subregion in the k-th iteration and $X_{ij} = f(A_i^j)$, where A_i^j and $f(\cdot)$ are defined according to equation (3.7) be the sample performance of i-th set in the j-th region. The two-stage ranking-and-selection procedure takes n_0 samples first, and then determine the total number N_j samples required from the j-th region using that information. This number N_j is selected to be sufficiently large so that the selection for correct subregion is made with probability at least P^* and an indifference zone $\varepsilon > 0$. Based

on the NP/Rinott-Filter method, the predicted best sampling rate of instances is derived analytically, considering the selection probability and indifference zone.

NP/Rinott-Filter

Step 0 ~ *Step* 2. Same as NP-Filter

Step 2 – 1. Let i the number of sample sets in each region.

If $i = n_0$ continue to *Step* 3. Otherwise let $i = i + 1$ and go back to *Step* 2.

Step 2 – 2. Calculate the first-stage sample means and variance

$$\bar{X}_j^{(1)}(k) = \frac{1}{n_0}\sum_{i=1}^{n_0} X_{ij}(k), \tag{4.2}$$

and

$$S_j^2(k) = \frac{\sum_{i=1}^{n_0}\left[X_{ij}(k) - X_j^{(1)}(k)\right]^2}{n_0 - 1}, \text{ for } j = 1, 2, 3. \tag{4.3}$$

Step 2 – 3. Compute the total sample size

$$N_j(k) = \max\left\{n_0 + 1, \left\lceil \frac{h^2 S_j^2(k)}{\varepsilon^2} \right\rceil\right\} \tag{4.4}$$

where ε is the indifference zone and h is a constant that is determined by n_0 and the minimum selection probability P^* of correct selection [Rinott, 1978].

Step 2 – 4. Obtain $N_j(k) - n_0$ more samples in each region.

Step 3 ~ *Step* 5. Same as NP-Filter.

Note: When obtaining the best sample set in each region, the newly created sample sets must be considered.

As described in the previous chapter, the NP/Rinott method guarantees to find an optimal solution probabilistically with the two-stage sampling efforts [Olafsson, 2003]. If a variance of performances in one region in the first sampling stage is large, it would be very liable to need the second sampling stage for enhancing possibility to find a best solution. Thus, reducing the performance variance can be a key point to find a best proportion of instances. Figure 4.10, graphical version of the results of Table 3.13 in Chapter 3, shows that

10% of instances is a best sampling rate of vote data set if we use static sampling method in every iteration. After the computational time decreases for a while as the instance sampling rate decreases, the time increases at the very lower proportion points, 5% and below due to backtrackings.

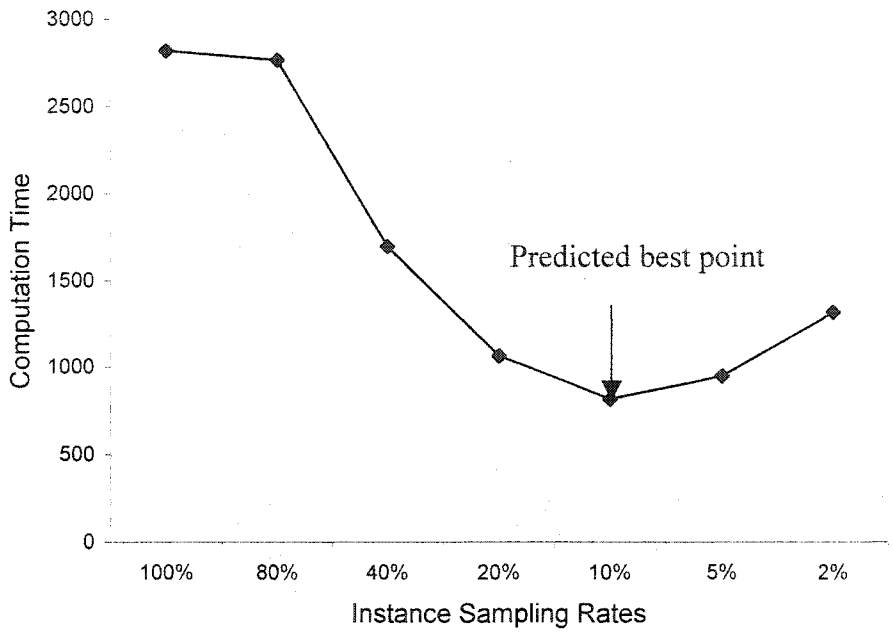


Figure 4.10 Computational time for instance sampling rates (data set 'vote').

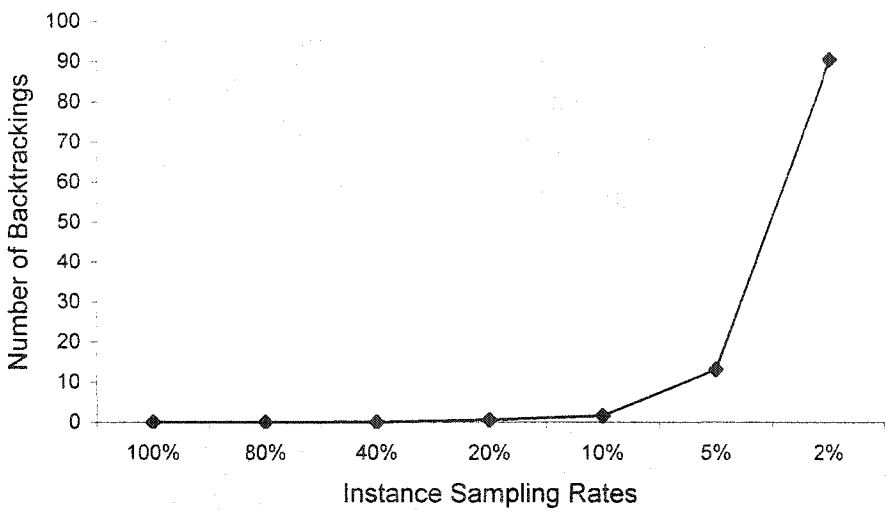


Figure 4.11 Number of backtrackings on instance sampling rates.

As shown in Figure 4.11, it is plotted that number of backtrackings abruptly increases at the 5% and 2% sampling rate points that cause the excessive computational works. Abrupt increase of backtrackings implies that the sampling rate should be made before the steep increase point of backtrackings, which matches the result of the minimum point of computational time in Figure 4.10. Those two figures provide the motivation for how to find the best solution.

4.3.2 Formulation

The best proportion of instances for minimizing the computational time of the NP/Rinott-Filter algorithm is derived in this section. First we define an optimization problem to find such a solution but partially employing heuristics and describe the reason why the heuristics should be used. To state those clearly, the following notations for expressing them are stated.

Notation:

T : Total computational time, $T = T_1 + T_2 + \ldots T_k + \ldots + T_K$, for $k = 1, 2, \ldots, K$

K : Total number of iteration,

$N = N_j(k)$: Number of sample feature sets at each iteration k,

I : Number of sample instances,

n : Total number of features,

m: Total number of instances,

R : Sample proportion of instances (sampling rate), $R = I / m$,

P^* : Probability of correct selection,

ε : Indifference zone.

If we define the total expected time of NP/Rinott algorithm as $E[T]$, $E[T]$ can be straightforwardly stated a product form of the total expected number of iterations of the algorithm and the expected time of iteration k.

$$E[T] = E_{d^*}[K] \cdot E[T_k \mid K] \tag{4.5}$$

Fortunately, we already know that the total expected number of iterations of the NP/Rinott algorithm [Olafsson, 2003] is given as follows:

$$E_{d^*}[K] = \frac{d^*}{2P^* - 1} + \frac{(1-P^*)^{2d^*+1}}{(P^*)^{2d^*}(2P^*-1)^2}\left(1 - \left(\frac{P^*}{1-P^*}\right)^{d^*}\right),$$

which can be approximately bounded by $E_{d^*}[K] \le \frac{d^*}{2P^* - 1}$.

Therefore, we are only interested in the expected time of iteration k that can be affected by the performance variability and sampling rate of instance as stated in the previous sections. The expected time of iteration k, $E[T_k \mid K]$, can be stated as a product form of the expected number of feature sets and the expected computational time of each feature sample set in the k-th iteration.

$$E[T_k \mid K] = E[N_k \mid K] \cdot E[T_k \mid N_k] \tag{4.6}$$

Unfortunately it is difficult or impossible to find an optimal solution of (4.6) because $E[N_k \mid K]$ is a function of $E[S^2(k)]$ as stated in (4.4), and $E[S^2(k)]$ does not have an analytically explicit relationship with the instance sampling rate (or number of instances). However, alternatively, we may find a solution using the trade-off between $E[N_k \mid K]$ and $E[T_k \mid N_k]$ that was noted at the end of the section 3.7.1. Since from the previous section it is known that the variability can be represented as the number of instances and $E[N_k \mid K]$ would increase as the number of instances decreases while $E[T_k \mid N_k]$ would decrease as shown in Figure 4.12. Therefore, rather than trying to solve (4.6), we propose to solve the following problem.

$$\text{Minimize } \lambda \cdot E[N_k \mid K] + (1-\lambda) \cdot E[T_k \mid N_k] \tag{4.7}$$

Our objective is to minimize both $E[N_k \mid K]$ and $E[T_k \mid N_k]$ simultaneously. Since $E[N_k \mid K]$ is a decreasing function, but $E[T_k \mid N_k]$ on the other hand is an increasing

function, and the units of both expected number and time are different, we create the objective function as stated in (4.7) by using an weight λ which should be determined by the experimenter. Furthermore, since as we pointed out earlier, $E[N_k \mid K]$ does not have an explicit analytical form, we replace $E[N_k \mid K]$ by a function of $E[S^2(k)]$ that should be heuristically found from the relationship with the instance sampling rate. Now we derive a solution for the problem above based on the following assumptions.

Assumption 1. An instance is uniformly sampled, that is $R \sim U(0, 1)$.

Assumption 2. The expected calculation time of each feature sample is directly proportional to the number of instances.

The number of features contained in the sample set may affect the calculation time. However, we discard the consideration on the number of features for simplicity. Thus, the number of instances only affects the calculation time of each feature sample set in (4.8) and the calculation time of each instance is a unit constant time.

$$E[T_k \mid N_k] = c_0 \cdot E[I] = c_0 \cdot I \tag{4.8}$$

Assumption 3. The constant n_0 in the equation (4.4) is chosen sufficiently small, i.e. $n_0 \leq N_j(k)$ so that we always take $N_j(k) = \frac{h^2 S^2(k)}{\varepsilon^2}$ of samples from each region in one iteration.

Now we need to know the expected number of feature sample sets from each region. For handling this problem, we need to consider the second stage sampling mechanism of NP/Rinott. Since h and ε are constants in (4.4), only $S^2(k)$ is a random variable. Thus, the expected number of feature sample sets in each iteration,

$$E[N_k \mid K] = \frac{h^2}{\varepsilon^2} E\left[S^2(k)\right] \tag{4.9}$$

From the fact that if the number of instance samples decreases, the performance variability of feature sample sets increases as stated in the previous section, thus, we know that the expected number of feature sample sets increases in (4.9).

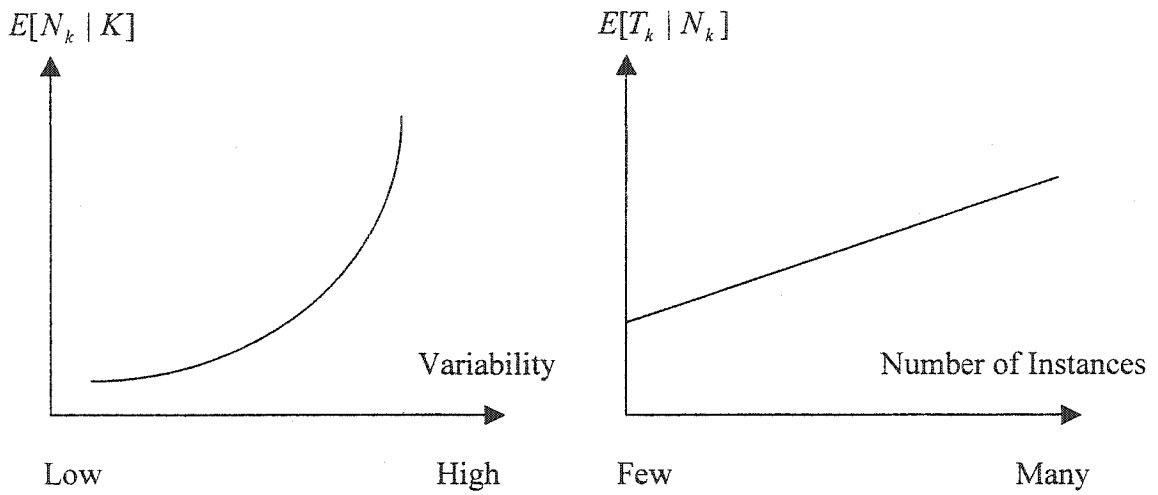


Figure 4.12 The expected number of feature sets and the expected calculation time of each feature set on variabilities and number of instances.

As noted earlier, it is unlikely that we find the distribution function of $S^2(k)$ analytically since $S^2(k)$ is a function of sample performance which does not have an analytically explicit form in terms of m. Thus, we find the probability distribution function empirically from tests that can determine the relation between $S^2(k)$ and I. The test is designed to calculate performance variances for various sampling rates of instances. There are two terms that can affect the variance, which are the number of feature sample sets, N_k and the sampling rate of instances, R. Thus, in order to know the relation between $S^2(k)$ and R only, we generate enumerated feature sample sets for one region at each iteration, calculate the performances, pick up one best performance, and finally calculate the variance of the best performances at each iteration. The test results are reported in Table 4.1. As shown in Figure 4.9, there is an exponential relationship between performance variance and instance sampling rate.

Assumption 4. The relationship between performance variance and instance sampling rate is exponentially distributed. $E[S^2(k)] = c_1 e^{-c_2 \cdot R}$ for $c_1 > 0$, $c_2 > 0$.

Based on the (4.7) and the assumptions, the restated problem is as follows.

$$\text{Min Cost}(R) = \lambda \cdot \frac{h^2}{\varepsilon^2} \cdot c_1 e^{-c_2 \cdot R} + (1-\lambda) \cdot c_0 \cdot m \cdot R$$

$$\text{Subject to} \quad \begin{array}{l} 0 < \lambda < 1,\ 0 < R \le 1, \\ h, \varepsilon, c_0, c_1, c_2 > 0. \end{array}$$

If we take the first derivative,

$$\frac{dCost}{dR} = -\frac{\lambda \cdot h^2}{\varepsilon^2} \cdot c_1 \cdot c_2 \cdot e^{-c_2 \cdot R} + (1-\lambda) \cdot c_0 \cdot m = 0$$

we can derive the best solution in (4.10).

$$R^* = -\frac{1}{c_2} \cdot \ln\left[\frac{(1-\lambda) \cdot c_0 \cdot \varepsilon^2 \cdot m}{\lambda \cdot h^2 \cdot c_1 \cdot c_2}\right] \tag{4.10}$$

Since $\frac{d^2 Cost}{dR^2} = \frac{\lambda \cdot h^2}{\varepsilon^2} \cdot c_1 \cdot c_2{}^2 \cdot e^{-c_2 \cdot R} > 0$, R^* is the minimal point. The value of λ can be chosen by an experimenter according to the preference. As λ grows closely to 1, R^* becomes larger to reduce the performance variance, while R^* becomes smaller as λ decreases to 0. The indifference zone, ε and selection probability, P^* that determines the value of h should be also determined by the experimenter. If ε is small and P^* is large, the sampling rate would be large. Otherwise it would be small. However, there is the interval that the value of λ have to be in because I is less than or equals to m, that is R is greater than 0 or less than 1.

$$-1 \le \frac{1}{c_2} \cdot \ln\left[\frac{(1-\lambda) \cdot c_0 \cdot \varepsilon^2 \cdot m}{\lambda \cdot h^2 \cdot c_1 \cdot c_2}\right] \le 0$$

$$\lambda \ge \frac{c_0 \cdot \varepsilon^2 \cdot m}{(c_0 \cdot \varepsilon^2 \cdot m + h^2 \cdot c_1 \cdot c_2)} \quad or \quad \lambda \le \frac{c_0 \cdot \varepsilon^2 \cdot m}{(c_0 \cdot \varepsilon^2 \cdot m + h^2 \cdot c_1 \cdot c_2 \cdot e^{-c_2})}$$

The analytical formula on the optimal instance sampling rate in the NP/Rinott-Filter is evaluated and compared with other approaches in the later sections.

4.3.3 Evaluation of Analytical Solution

By intuition, we could guess that small R^* would reduce the computational time of the feature selection process. However, since the small value of R^* can cause a large variance of performances that leads to the excessive computation time. Thus now we evaluate the formula (4.10) numerically with constants that can be determined experimentally and

estimated using a nonlinear programming solver in the case of the c_1 and c_2. Then the predictive best solution, R^* is evaluated in the NP/Rinott-Filter with a static sampling approach, that is we use a fixed number of instances, $R^* \times m$, in every iteration.

Since the relationship between performance variance and instance sampling rate is exponentially distributed, it has a nonlinear form with constants c_1 and c_2. Those constants are investigated by the least square method in LINGO, which is an well-known software. Using the modified data sets for each 5 data set in Table 4.1, we estimate the values of the constants that are reported in Table 4.2.

Table 4.2 Estimated constants from the data.

Data set	$\hat{c}_1$	$\hat{c}_2$
vote	36.8	6.8
audiology	360.4	33.8
cancer	402.4	19.7
kr-vs-kp	17.6	11.0
lymph	94.7	13.3

Other constants are chosen by the experimenter, and λ, ε, P^* are set as follows. The constant λ has three different settings, 0.25, 0.50 and 0.75, each of which represents which factor should be more importantly considered for the minimization. For example, the value, 0.25 implies that we minimize more intensively the expected computation time of each feature sample set. On the other hand, the value 0.75 implies that we concentrate on the minimization of the expected number of feature sample sets. The constant ε, an indifference zone on the optimal solution is set to 0.01 and 0.05, and P^* is set to 0.75, 0.90 and 0.95, which is experimentally believed that those values provide desirable results in the NP/Rinott algorithm. Note that the value h is a derived value based on the n_0 and P^*, and c_0 can have an arbitrary value depending on the experimenter. In this test, it is set to the same value as c_1 for simple calculation.

In order to reflect various settings on the numerical tests, 18 combination problems for each data set were tested, that is, 18 different instance sampling rates were generated for the test using the equation (4.10). The NP/Rinott-Filter algorithm ran 5 times with full depth and

Naïve Bayes classifier to calculate accuracy of the finally selected feature set. The Table 4.3 ~ Table 4.5 refer to the averaged accuracy, computational time and number of backtrackings with standard deviations respectively.

Table 4.3 Accuracies of NP/Rinott-Filter on various R^*.

			$\lambda = 0.25$		$\lambda = 0.50$		$\lambda = 0.75$	
Data	ε	P^*	Sample Rates (%)	Accuracy	Sample Rates (%)	Accuracy	Sample Rates (%)	Accuracy
vote	0.01	0.75	27	92.7±1.9	32	93.9±1.7	38	92.8±0.5
		0.90	31	93.2±1.0	37	93.0±1.1	42	93.6±0.3
		0.95	34	92.5±0.6	39	93.7±1.4	45	93.9±1.2
	0.05	0.75	10	91.7±0.9	16	93.2±1.4	21	92.5±1.2
		0.90	15	93.0±0.8	21	92.1±1.6	26	93.0±1.2
		0.95	18	92.2±1.0	23	92.6±0.3	29	92.8±1.4
audiology	0.01	0.75	38	70.7±1.2	44	69.7±2.2	49	69.4±2.9
		0.90	43	72.3±1.2	48	69.4±1.8	54	71.5±2.1
		0.95	45	68.8±1.8	51	71.5±1.8	56	69.3±2.8
	0.05	0.75	22	69.7±2.9	27	69.8±2.0	33	71.3±1.8
		0.90	27	71.3±2.1	32	70.8±1.7	38	72.1±1.8
		0.95	29	70.4±2.1	35	69.6±2.7	40	67.9±2.9
cancer	0.01	0.75	34	73.3±1.1	40	73.2±0.8	45	72.9±2.1
		0.90	39	73.0±0.7	45	73.4±0.6	50	73.5±0.3
		0.95	41	73.4±0.5	47	73.6±0.2	53	74.0±0.4
	0.05	0.75	18	73.3±0.5	24	73.5±0.6	29	73.2±0.7
		0.90	23	73.1±0.4	28	73.0±1.6	34	73.6±0.2
		0.95	25	73.7±0.4	31	73.4±0.9	36	73.1±0.6
kr-vs-kp	0.01	0.75	19	87.3±5.7	25	87.1±6.0	30	86.5±7.8
		0.90	24	89.2±5.6	29	84.9±6.4	35	91.0±2.3
		0.95	26	89.8±1.2	32	89.2±1.9	37	88.9±0.7
	0.05	0.75	3	89.0±0.5	8	88.3±1.0	14	89.8±1.9
		0.90	7	89.6±0.9	13	90.5±2.8	19	90.1±1.1
		0.95	10	89.8±2.3	15	89.6±4.0	21	87.8±3.5
lymph	0.01	0.75	28	84.6±1.0	34	83.9±1.1	39	85.1±1.1
		0.90	33	84.9±0.8	38	84.1±0.8	44	84.2±0.6
		0.95	35	84.1±1.0	41	83.7±1.2	47	83.0±1.2
	0.05	0.75	12	83.5±2.0	17	83.7±1.9	23	84.4±1.4
		0.90	17	84.3±0.9	22	83.5±1.9	28	84.1±1.5
		0.95	19	85.3±0.9	25	83.5±0.6	30	85.0±1.0

As reported in Table 4.3, even though 18 different instance sampling rates were evaluated according to constant setting, there would be no significant difference in accuracies since the Rinott's two stage sampling efforts support problems caused from the fewer

number of instances. However, the computational time of data sets containing the fewer number of instances takes longer than that of larger size data sets. Small number of instances cause easily large differences on performances, which would results in additional feature sample sets in the second sampling stage, more calculation works for those additional sets, and possibly the more backtrackings in the algorithm. For example, in 'vote' data set, we hardly find any significant difference in accuracies for three sampling rates, 27%, 31%,

Table 4.4 Computation time of NP/Rinott-Filter on various R^*.

			$\lambda = 0.25$		$\lambda = 0.50$		$\lambda = 0.75$	
Data	ε	P^*	Sample Rates	Speed	Sample Rates	Speed	Sample Rates	Speed
vote	0.01	0.75	27	5660±1525	32	5485±2185	38	5282±1351
		0.90	31	12089±7129	37	14047±3906	42	15556±3314
		0.95	34	22452±3111	39	26347±6966	45	28743±10879
	0.05	0.75	10	857±61	16	799±119	21	1027±152
		0.90	15	913±81	21	1237±248	26	1209±145
		0.95	18	1329±241	23	1350±93	29	1616±244
audiology	0.01	0.75	38	162094±111523	44	114892±32789	49	157615±93509
		0.90	43	364322±138453	48	462902±320380	54	423618±136529
		0.95	45	631091±476810	51	571414±232038	56	377947±177592
	0.05	0.75	22	31972±10584	27	28872±6644	33	37100±18299
		0.90	27	44318±9151	32	48325±38379	38	46513±16672
		0.95	29	74852±52994	35	46025±17890	40	59431±24863
cancer	0.01	0.75	34	665±71	40	711±149	45	815±165
		0.90	39	963±358	45	1125±651	50	1163±289
		0.95	41	1488±401	47	1333±293	53	1502±433
	0.05	0.75	18	481±86	24	434±25	29	473±29
		0.90	23	478±94	28	476±34	34	542±35
		0.95	25	474±60	31	511±86	36	516±34
kr-vs-kp	0.01	0.75	19	27407±11133	25	32336±5715	30	40726±6399
		0.90	24	61012±23175	29	74957±24444	35	72227±2929
		0.95	26	78313±7547	32	98408±3051	37	130727±12512
	0.05	0.75	3	5189±537	8	7400±889	14	14242±2055
		0.90	7	7551±1323	13	11793±1186	19	16758±1158
		0.95	10	9517±1278	15	14985±2787	21	19747±2289
lymph	0.01	0.75	28	4348±885	34	3699±563	39	4011±932
		0.90	33	9770±1960	38	8017±1760	44	9535±2355
		0.95	35	13148±2108	41	12706±3216	47	15021±1564
	0.05	0.75	12	1395±468	17	1125±141	23	1053±71
		0.90	17	1346±463	22	1241±100	28	1212±101
		0.95	19	1476±181	25	1592±615	30	1320±66

and 34% while the computational time, 5660, 12089, and 22452 (milliseconds), for those cases is significantly different respectively as shown in Table 4.4. For other data sets, we can find very similar results.

Table 4.5 Number of backtracks of NP/Rinott-Filter on various R^*.

			$\lambda = 0.25$		$\lambda = 0.50$		$\lambda = 0.75$	
Data	ε	P^*	Sample Rates	Backtracks	Sample Rates	Backtracks	Sample Rates	Backtracks
vote	0.01	0.75	27	2.4±3.6	32	1.6±2.5	38	0.0±0.0
		0.90	31	0.0±0.0	37	0.0±0.0	42	0.2±0.4
		0.95	34	0.0±0.0	39	0.6±1.3	45	0.0±0.0
	0.05	0.75	10	0.6±0.5	16	0.2±0.4	21	0.4±0.9
		0.90	15	1.2±1.3	21	0.0±0.0	26	0.2±0.4
		0.95	18	0.2±0.4	23	0.8±1.3	29	0.2±0.4
audiology	0.01	0.75	38	149.8±136.3	44	87.6±49.2	49	153.2±131.5
		0.90	43	175.8±99.0	48	230.6±113.1	54	161.6±73.6
		0.95	45	148.0±179.6	51	116.0±89.8	56	77.0±60.5
	0.05	0.75	22	176.8±55.0	27	128.8±27.0	33	137.8±79.5
		0.90	27	179.8±48.2	32	181.6±191.5	38	143.8±75.5
		0.95	29	285.6±224.2	35	145.6±57.6	40	171.2±115.5
cancer	0.01	0.75	34	0.8±1.3	40	0.2±0.4	45	4.8±7.5
		0.90	39	5.6±7.2	45	0.6±0.9	50	3.2±5.5
		0.95	41	8.4±14.4	47	0.4±0.9	53	0.8±1.3
	0.05	0.75	18	8.2±4.4	24	2.4±3.0	29	2.6±3.6
		0.90	23	4.6±3.2	28	3.6±5.0	34	1.2±2.2
		0.95	25	4.4±3.8	31	3.6±3.8	36	1.0±1.7
kr-vs-kp	0.01	0.75	19	2.0±4.5	25	0.0±0.0	30	1.2±2.2
		0.90	24	1.0±2.2	29	1.2±2.2	35	0.0±0.0
		0.95	26	0.0±0.0	32	0.0±0.0	37	0.0±0.0
	0.05	0.75	3	2.2±3.3	8	0.4±0.9	14	0.0±0.0
		0.90	7	0.2±0.4	13	0.8±1.8	19	0.4±0.9
		0.95	10	0.4±0.5	15	2.2±4.9	21	1.2±2.7
lymph	0.01	0.75	28	6.4±4.8	34	2.2±1.8	39	3.0±2.9
		0.90	33	4.2±1.5	38	1.4±1.7	44	1.6±1.8
		0.95	35	2.8±3.3	41	1.6±3.0	47	3.0±5.7
	0.05	0.75	12	29.6±19.6	17	10.2±4.0	23	2.0±2.8
		0.90	17	11.8±10.5	22	10.8±6.3	28	2.6±2.5
		0.95	19	8.0±6.9	25	9.6±9.7	30	3.6±2.9

Simply, we can explain the computation time (or scalability) of the NP/Rinott-Filter using three major factors under the condition such that it has an equal depth, which are the number of instances, performance variance, and the number of backtrackings. If we employ a

broad structure of the algorithm, the computation time of the outer loop of the algorithm can be explained as the number of backtrackings while the performance variance is related to the computation time of the inner loop. Those two factors, the number of backtrackings and the performance variance are strongly related to the number of instances as described earlier in that the small number of instances may lead to a large variance of performances and possibly large number of backtrackings. If we have a small performance variance enough not to require any additional feature sample sets in the second stage and the algorithm partitions correctly in all iteration, only the number of instances would affect the computational time. But in real the number of backtrackings and performance variance are also very critical factors for the scalability as reported in Table 4.5. Let's consider the 'audiology' data set problems, both the instance sampling rates, 54% and 56% report significantly different computation time, 423618 and 377947 milliseconds, even though the two sample rates are slightly different. The backtrackings occurs more frequently (161.6 versus 77.0) so that it takes significantly longer in the former case. From the result, it is inferred that the fewer instances may bring such frequent backtrackings. On the other hand, in 'cancer' data set, even though both 23% and 25%, even smaller difference between the two sampling rates are used, the computation time at 23% is somewhat smaller than that of 25% (e.g. 478 versus 474). In this case, the number of backtrackings is not so different (4.6 versus 4.4). Thus we can say that large performance variances caused by fewer instances cause the excessive computation time. Therefore, in a word, we can conclude that the total computation time of the NP/Rinott-Filter depends on the number of instances, the number of backtrackings, and the performance variance as expected.

4.4 Dynamic Sampling Approach

In this section, we investigate a new approach to dynamically find a good instance sampling rate. This approach provides an alternative to the two-stage NP/Rinott-Filter. As described in the previous sections, the number of backtrackings is one of the critical factors determining the computation time (speed) of the feature selection process. Backtracking is necessary to correct mistakes, but many backtrackings directly affect the excessive computation time. In this algorithm, we consider only how to prevent backtrackings from

occurring. If performance variability is low, which implies large amount of instances or a high instance sampling rate is used, it is obvious that we can reduce the number of backtrackings. On the other hand, a high performance variance that can be regarded as a low instance sampling rate easily causes frequent occurrences of backtrackings. However, if a sufficient number of instances statically in every iteration of the algorithm to reduce the number of backtrackings, it would often require an overestimated amount of instances for even a small size of feature sets. Thus, if we change the number of instances (or instance sampling rates) adaptively based on the occurrence of backtrackings, that is, we use larger samples for the frequent occurrences of backtrackings and smaller samples for the rare occurrences, we can reduce the computation time of the algorithm. Based on this idea, we developed the heuristic algorithm named Adaptive NP-Filter.

4.4.1 Adaptive NP-Filter

This Adaptive NP-Filter uses same framework as the NP-Filter as well as dynamical sampling works to find a good instance sampling rate. Specifically if the percentage of backtrackings in last several iteration is greater than that of backtrackings in all iteration up to current iteration, we increase the sampling rate because that many backtrackings are undesirable. Otherwise the number of samples is reduced by the reason where we do not want unnecessary instances.

Notation:

N	: Number of iterations, $N = 1, 2, 3, \ldots,$
last N	: Number of iterations performed last,
n	: depth (full depth is same as the number of features),
b	: Number of backtrackings for last N iterations ,
B	: Total number of backtrackings up to current iteration,
R	: Instance sampling rate,
Δ_n	: Step size on instance sampling rate,
c	: Constant,

Adaptive NP-Filter

Initialize $N = 0$, last $N = 0$, $b = 0$, $B = 0$, $n = 0$, R, Δ_n, c, set by an experimenter.

Loop

All procedures in one iteration are same as NP-Filter.

If b / last $N >= c \cdot \frac{B}{N}$, then $R = R + \Delta_n / n$,

Otherwise, $R = R - \Delta_n / n$.

Until n = the number of features (full depth is reached).

It would be better that the values for last N and c should be carefully chosen by the experimenter after some pilot tests because large last N and c would make the algorithm's response for instance sampling rate changes insensitive. On the other hand, small values of the constants would make it very sensitive. As the depth approaches into the end, smaller amount of features in a set is needed, which means that the sampling rate converges to some point. Thus it is required that smaller amount of changes (Δ_n / n) on the sampling rate should be applied as the depth of the algorithm increases.

4.4.2 Evaluation of Adaptive NP-Filter

The Adaptive NP-Filter is evaluated numerically on the same test problems before. For this test, several parameters and constants were set as stated in Table 4.6. In order to test several situation, 5% and 10% of features of original data sets as two last N, 1.0 and 1.2 as two constant c, and 5% to 80% initial sampling rates of instances were used. For a step size, ∇ is set to 0.2 which is believed by experimenter to be reasonable. The instance sampling rate is dynamically changed 0.01 to 0.99 based on the decision criteria until the full depth is reached. A moving average of the instance sampling rates is calculated for the rates of the last 5 iterations, which is chosen arbitrarily. Since the instance sampling rates fluctuate up and down frequently, it is hard to see a trend of the dynamic rates. Furthermore, even though the rate converges to some point finally, it is hard to accept that the point represents a really

converged point because it could fluctuate even right before the final iteration. Thus, the moving average may show more reasonable convergence for the sampling rate although it is a little rough. As a classifier, Naïve Bayes was used and for each case, the same tests were repeated 5 times.

Table 4.6 Test configuration of Adaptive NP-Filter.

Data set	Number of Features	last N		Initial R (%)
		5%	10%	
vote	16	1	2	5
audiology	69	3	7	10
cancer	9	N/A	1	20
kr-vs-kp	36	2	4	40
lymph	18	1	2	80

The following figures, Figure 4.13 ~ Figure 4.16, show moving averages of last 5 iteration for instance sampling rates adaptively changed by the algorithm as the iteration proceeds. Each figure contains four different initial instance sampling rates, 5%, 20%, 40%, and 80%, representing two data sets, 'vote' and 'kr-vs-kp'. The results for other data sets are presented in Appendix B.

As shown in Figure 4.13 and Figure 4.14, the moving average curve looks sensitive since small c and last N make more frequent decisions for changing the instance sampling rate and likely to change the rate. When it comes to the initial sampling rate, large initial sampling rate, for example 80% initial sampling rate, can easily make the algorithm terminated with rapid reduction of the rate since many instances hardly cause backtrackings. If we use larger step size enough to allow the algorithm to find a good solution or more iterations, it would converge to a lower rate point rapidly. On the other hand, small initial sampling rate, for example 5% initial sampling rate, shows a pattern to increase at the early phase, decrease for a while, and finally fluctuate with converging to a good instance sampling rate in most cases. It is shown that most cases in figures have a tendency that the instance sampling rates finally converge to a desired solution according as a low rate is better for scalability even though some problems are hard to be terminated early.

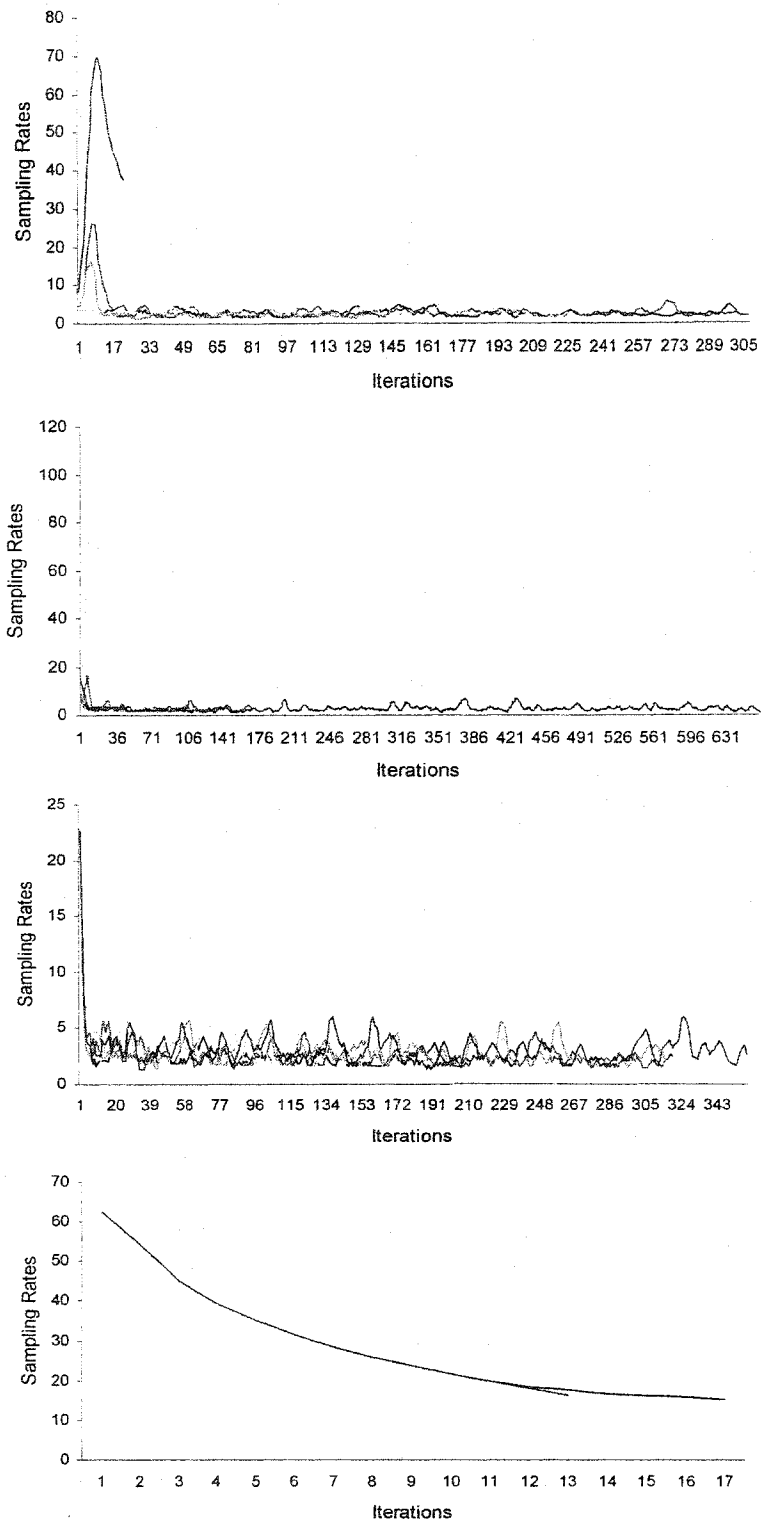


Figure 4.13 Moving averages of instance sampling rates of 'vote' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 1$, and $k = 5$ (from the top).

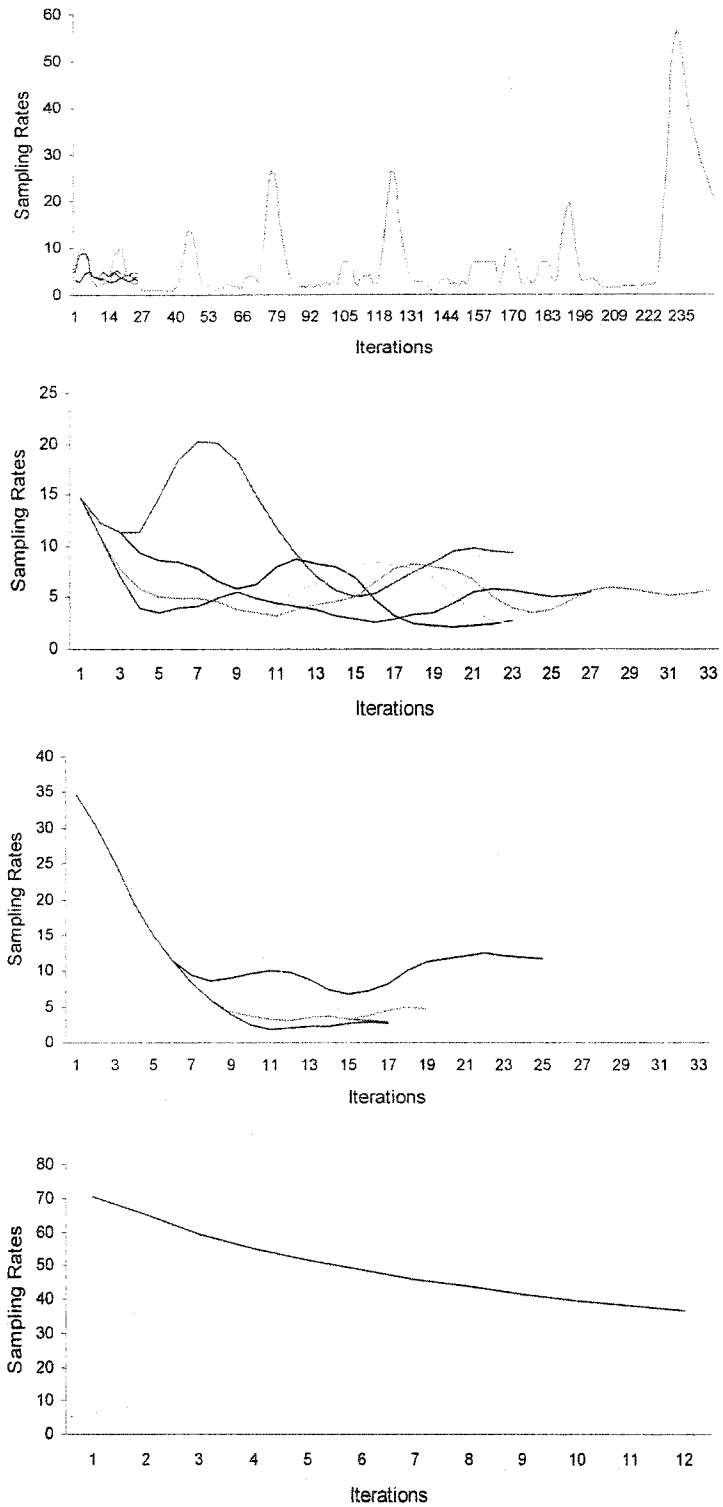


Figure 4.14 Moving averages of instance sampling rates of 'vote' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 2$, and $k = 5$.

One more interesting result is that the converged sampling rate is similar or a little smaller than the sampling rate found by the Static NP-Filter presented previously. For example, the predicted best solution of 'vote' data set found by the Static NP-Filter is 10%. The results in Figure 4.13 and Figure 4.14 except 80% initial sampling rate case show that the converged rates mostly fall in the range of 5% ~ 10%. This result should not come as a surprise because as the depth in NP increases, the size of a selected feature set gets smaller, which means fewer instances are needed naturally.

Usually the adaptiveness can provide flexibility to the changing situation. If we do not meet a worst case situation that does not happen frequently in many practical applications, then we can find a solution on the sample size with much fewer samples than the required samples in the worst case [Domingo, Gavalda and Watanabe, 2000]. However, by requiring overestimated samples in the worst case that is calculated a priori, the Static NP-Filter takes fixed amounts of instance samples in every iteration and needs equally excessive computational works for even a small size of feature sample sets. Thus, it is intuitive that the converged sample rates are mostly smaller than that of the Static NP-Filter. It has been reported in many research articles that the adaptive (dynamic) sampling approach outperfoms the static sampling approach.

The results for 'kr-vs-kp' data set also report a very similar pattern as those of 'vote' data set. The predicted best sampling rate by the Static NP-Filter is 5% for this data set. On one hand, the converged rates by the Adaptive NP-Filter are almost in a range around less than or very similar to 5% as shown in Figure 4.15 and Figure 4.16. This benefit of the Adaptive NP-Filter would help reduce the computational time while maintaining an acceptable level of accuracy.

Now we evaluate the Adaptive NP-Filter in terms of accuracy and computational time. In order to find some performances such as accuracy and computational time, similar test configurations were applied, that is Naïve Bayes classifier and 5 times repetition, and the numerical results are reported in Table 4.7 and Table 4.8 (other results are reported in Appendix B) with the averages and standard deviations.

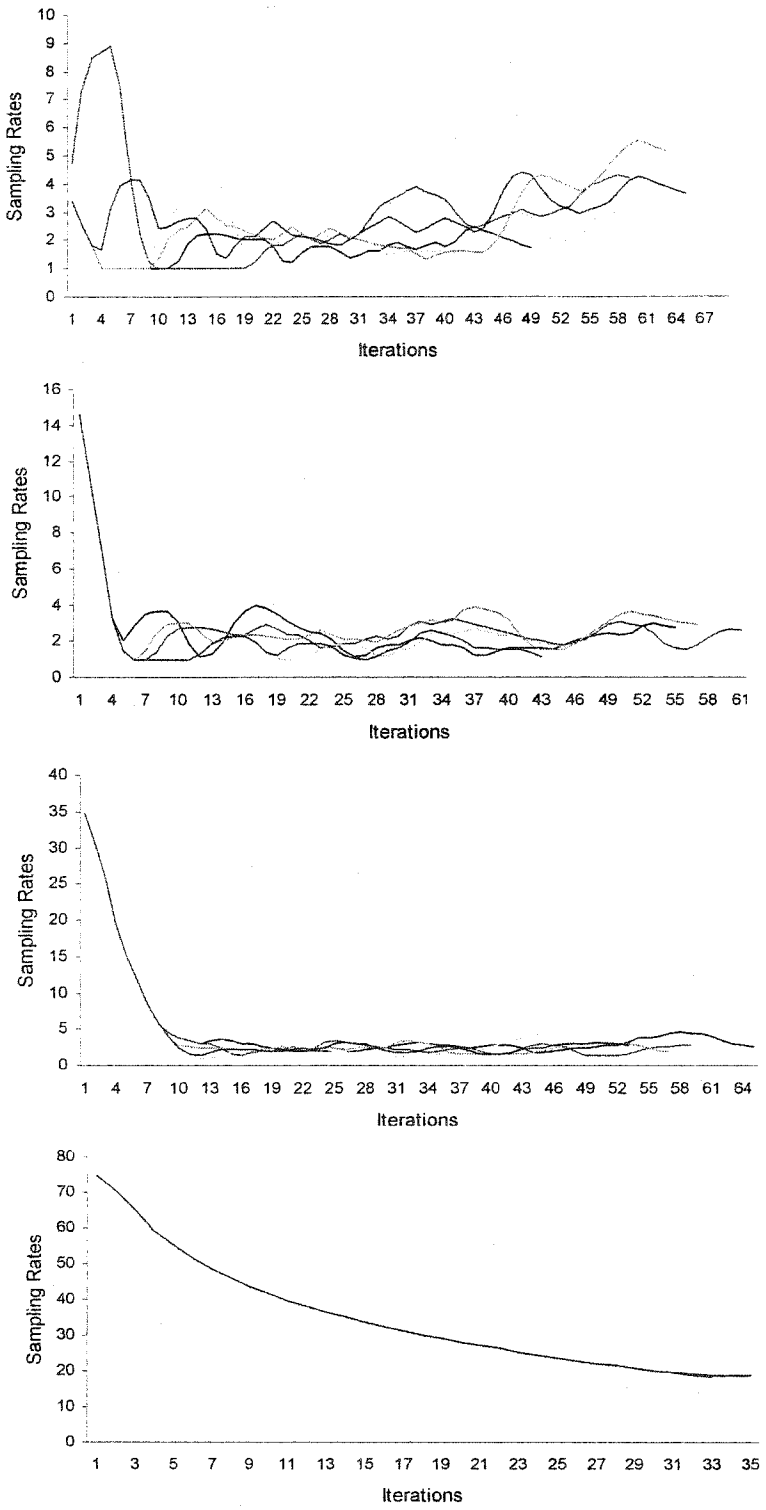


Figure 4.15 Moving averages of instance sampling rates of 'kr-vs-kp' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 2$, and $k = 5$.

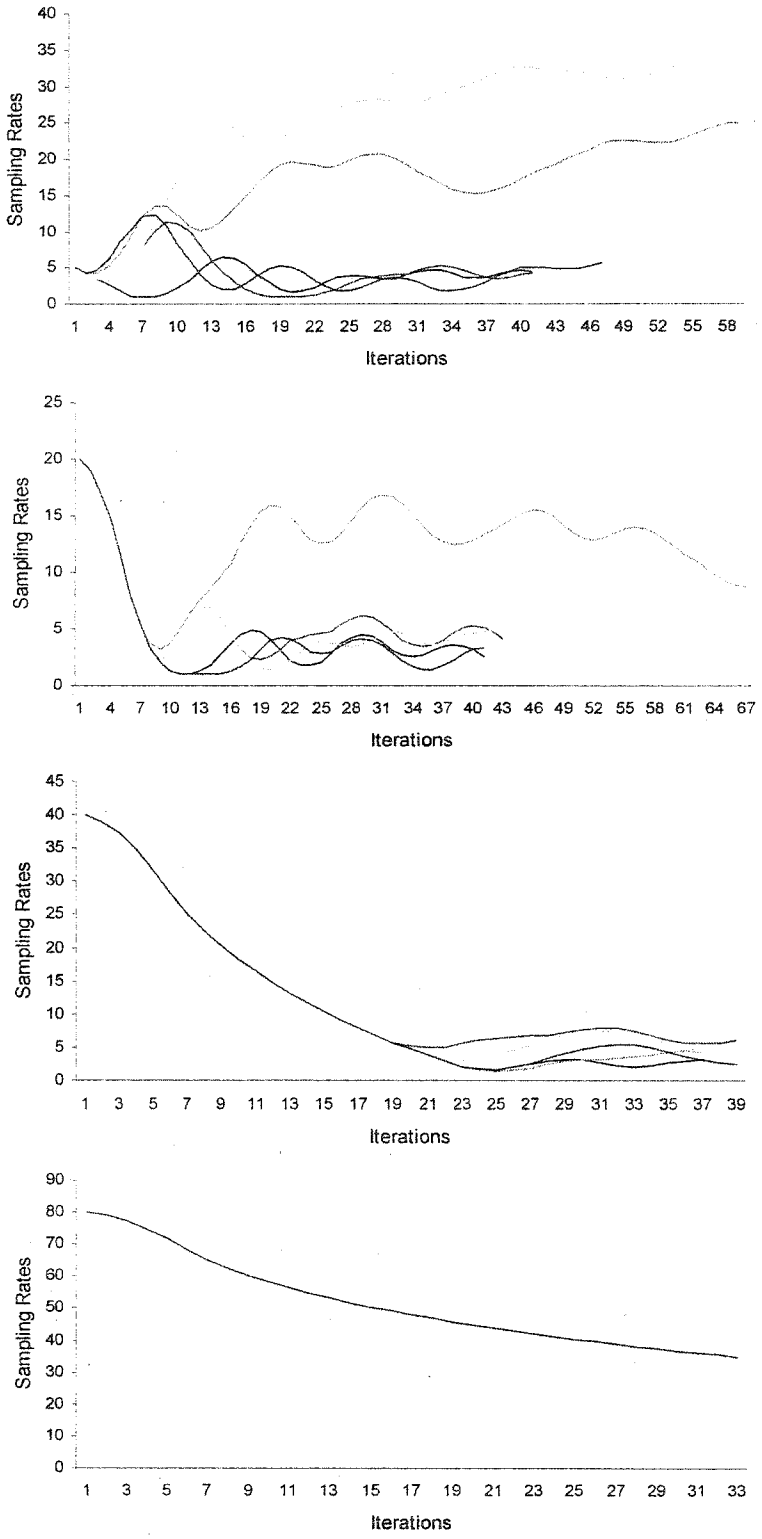


Figure 4.16 Moving averages of instance sampling rates of 'kr-vs-kp' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 4$, and $k = 5$.

Table 4.7 Numerical results of Adaptive NP-Filter with $c = 1.0$ and last $N = 5\%$.

Data set	Sampling Rates (%)	Accuracy	Speed	Backtracks	Converged Rates (%)
vote	5	92.5±1.5	1267±402	49.2±41.1	7.3±9.8
	10	93.1±0.4	4338±4368	303.4±363.1	3.7±2.1
	20	91.6±0.7	3662±3789	270.8±366.5	2.0±0.8
	40	93.1±1.1	3663±1482	240.4±147.7	1.6±0.2
	80	92.2±1.0	1067±61	0.6±0.9	16.3±0.3
audiology	5	71.3±2.1	36012±29713	129.8±124.2	29.0±6.9
	10	70.4±3.0	207481±169931	1357.4±1240.2	23.1±9.0
	20	70.8±1.6	47781±20390	167.6±88.9	35.4±8.1
	40	69.0±2.5	206110±258237	944.2±1206.5	33.8±14.0
	80	70.6±2.6	37078±20248	79.4±67.1	59.0±4.5
cancer			N/A		
kr-vs-kp	5	86.6±2.4	7122±3167	13.8±8.9	4.1±4.2
	10	84.9±4.3	7122±1306	16.8±5.7	4.7±2.3
	20	85.5±5.0	9932±6555	17.2±9.7	6.0±5.7
	40	87.2±4.7	12054±7838	15.6±10.6	5.4±5.4
	80	83.4±6.6	262008±512274	89.2±194.5	27.9±16.1
lymph	5	84.3±1.3	10790±9759	658.2±640.0	10.6±0.7
	10	83.7±0.9	13800±11248	830.0±699.8	15.6±10.6
	20	84.3±1.2	5625±4392	318.4±282.5	11.8±8.0
	40	84.9±1.4	12193±16855	701.4±1041.6	7.4±3.9
	80	83.9±0.7	1107±210	6.8±8.0	12.3±1.1

As shown in several previous figures plotting convergence pattern of instance sampling rates, when small c and last N are used, it takes longer since it causes more strict criteria for determining instance sampling rate. Regardless of values of parameters, any significant differences on accuracies are hardly found. The computational time (speed) looks to be influenced by an initial sampling rate. For example, initial sampling rate, 5% and 10% of 'vote' data set in the two tables takes least except rate 80% that should be excluded in this issue since high initial sampling rate would make rapid convergence without backtrackings. If we consider the 'audiology' data set and others, very similar pattern as the 'vote' data set is also detected. Furthermore, the finally converged sampling rate is ranged similarly but a little lower as the predicted best solution found by the Static NP-Filter. For example, the converged rate of 'vote' data set ranges from 5% to 10% approximately and mostly, 15% to 35% in 'audiology', 13% to 25% in 'cancer', 4% to 9% in 'kr-vs-kp' and 10% to 20% in

'lymph' data set. Thus, if the algorithm starts with a similar sampling rate as the converged sampling rate, it takes less time. Otherwise, it takes longer.

Table 4.8 Numerical results of Adaptive NP-Filter with $c = 1.2$ and last $N = 10\%$.

Data set	Sampling Rates (%)	Accuracy	Speed	Backtracks	Converged Rates (%)
vote	5	92.5±1.0	903±587	19.4±27.7	8.6±8.8
	10	91.7±1.2	655±77	7.2±3.5	6.9±4.9
	20	92.2±0.5	663±66	5.8±0.8	5.3±0.5
	40	91.8±1.5	869±194	6.0±4.7	9.9±8.9
	80	93.7±0.6	1490±77	0.4±0.9	33.4±9.7
audiology	5	71.9±4.1	53054±14825	194.0±137.2	15.1±13.0
	10	69.0±3.6	26173±12959	107.0±58.1	31.9±9.3
	20	71.3±1.5	44677±19774	132.0±69.1	39.5±2.8
	40	68.3±1.5	31386±10880	89.2±46.2	43.0±9.5
	80	70.9±2.6	64686±32568	111.4±106.9	77.7±10.2
cancer	5	73.2±0.9	507±153	20.8±14.2	12.9±9.7
	10	73.4±0.7	444±98	13.4±6.3	14.1±2.9
	20	73.4±0.5	866±502	47.6±47.0	25.2±23.9
	40	73.8±0.3	1007±369	55.0±36.4	22.4±12.7
	80	74.1±0.5	523±32	0.8±0.8	39.5±8.4
kr-vs-kp	5	88.0±5.0	7835±3419	5.6±1.1	6.3±4.4
	10	86.0±4.2	8556±4984	6.4±3.8	6.8±5.1
	20	86.2±4.4	13301±11386	13.8±12.2	9.3±9.4
	40	87.0±3.9	15104±3659	5.4±2.8	7.6±3.3
	80	86.1±5.5	60395±13765	2.6±3.6	44.5±13.2
lymph	5	84.6±0.9	4373±3750	319.0±359.8	20.1±3.3
	10	83.8±1.1	2448±2326	155.8±243.3	17.0±4.3
	20	84.9±1.4	3205±2952	217.6±284.8	16.7±8.8
	40	84.1±0.9	999±113	8.0±3.4	16.4±5.2
	80	84.3±0.6	1240±199	1.6±2.3	37.7±6.4

It can be surely said that the Adaptive NP-Filter finds a good solution without unnecessarily trying to larger sampling rates and time consuming.

4.5 Comparison

In this chapter, two different approaches were developed and investigated in terms of scalability of feature selection process. It is obvious that one of the most important key issues is how fast we can find a reduced subset without much sacrificing accuracy. This section

deals with performance comparison of those approaches including the Static NP-Filter based on the least amount of computational time for each approach.

Table 4.9 Comparison of three different scalability methods.

Data set	Approach	Sample Rate (%)	Accuracy	Speed	Backtracks
vote	Adaptive NP	6.9 (10) ±4.9	91.7±1.2	*655±77	7.2±3.5
	NP/Rinott	16	93.2±1.4	799±119	0.2±0.4
	Static NP	10	92.4±1.0	816±167	1.6±2.2
audilogy	Adaptive NP	31.9 (10) ±9.3	69.0±3.6	*26173±12959	107.0±58.1
	NP/Rinott	27	69.8±2.0	28872±6644	128.8±27.0
	Static NP	10	69.2±2.4	35839±14563	371.0±182.0
cancer	Adaptive NP	14.1 (10) ±2.9	73.4±0.7	444±98	13.4±6.3
	NP/Rinott	24	73.5±0.6	*434±25	2.4±3.0
	Static NP	10	72.6±1.2	486±89	7.4±3.4
kr-vs-kp	Adaptive NP	2.4 (10) ±0.8	88.3±1.2	5225±1035	10.4±2.9
	NP/Rinott	8	89.0±1.0	*5189±889	2.2±0.9
	Static NP	5	89.0±1.2	7246±809	1.8±3.0
lymph	Adaptive NP	16.4 (40) ±5.2	84.1±0.9	*999±113	8.0±3.4
	NP/Rinott	23	84.4±1.4	1053±71	2.0±2.8
	Static NP	20	84.7±1.0	1013±182	2.0±1.6

( ) of Sample Rate in Adaptive NP represents initial instance sampling rate

The Adaptive NP-Filter provides the best performance in terms of speed in three data sets, 'vote', 'audiology', and 'lymph', and the NP/Rinott-Filter takes the least time in two data sets, 'cancer' and 'kr-vs-kp' without sacrificing accuracy. On the other hand, the Static NP-Filter shows the worst performance. Thus, the results show that sampling fixed number of instances in every iteration for making scalable feature selection process can not be an excellent alternative, that is because it uses a same size of instances instead of fewer instances even in a high depth and it requires an undesirable size of samples. Even though the NP/Rinott-Filter also uses a static sampling approach, it can reduce time by controlling the number of feature sample sets applying the second sampling stage. These results imply that as the depth of the algorithm increases, a different sampling rate, that is the adaptive number of instances, should be applied. Synthetically, if we would like to perform a feature selection

process focused on the higher accuracy with a desirable speed, the NP/Rinott-Filter can be recommended. On the other hand, if we focus on the least computation time, the Adaptive NP-Filter can be effectively used.

4.6 Summary and Discussion

First we showed that the NP feature selection algorithm is appropriate for scalability of the number of instances. We have presented both analytical and heuristic approaches for scalable feature selection based on the new optimization-based feature selection methodology called the NP/Rinott-Filter and NP-Filter, and compared performances in terms of accuracy and computation time including the Static NP-Filter. Numerical results showed that the adaptive sampling approach is generally better than the other methods in terms of the computational time, and for even accuracy, the adaptive approach performed well comparing that of the NP/Rinott-Filter which performed best in terms of accuracy. Thus using dynamic sampling approach is a very effective way to deal with a large number of instances in the NP-Filter or even other general algorithms with respect to speed (computational time). The NP/Rinott-Filter showed best performances in some test problems as well with respect to accuracy and speed even though it employs a static sampling approach. Hence, the Adaptive NP-Filter and NP/Rinott-Filter can be very good alternatives for a scalable feature selection under some conditions or preferences of an experimenter.

5 MIXED TYPE OF FEATURES

5.1 Introduction

The NP feature selection method uses the information gain to determine a partitioning order of features by evaluating quality of a feature. However, if we have a data set containing continuous valued features, we have to apply an appropriate discretization method to be able to make it nominal and apply the information gain. Even though many discretization methods have been introduced, it is difficult to know where boundaries should be drawn.

In this chapter, we assume that we do not use discretization for continuous valued features, that is we use continuous values directly to evaluate the quality of features. Thus, after reviewing the previous discretization methods briefly, we employ 2 different methods in the NP-Filter, that is correlation based subset evaluator [Hall, 1998] and ReliefF feature evaluator [Kononenko, 1994], in order to deal with a mixed type of features, nominal and numerical data type. Those two methods are used to determine an order of features on partitioning evaluation. These methods will be evaluated and compared with information gain feature evaluator enabled by discretization and no feature selection (NFS) to address the effectiveness of feature selection using the NP-Filter for the selected data sets. Further, we investigate whether partitioning orders by the different feature quality evaluators may affect performances in the NP-Filter with Naïve Bayes, C4.5 and *k*-nearest neighbor learners.

Finally we perform a case study on an application that uses a mixed type of features in a data set. By applying ReliefF in the NP-Wrapper to handle numerical and nominal features in the data set gathered from an Internet auction system that facilitates reverse logistics [Ryan, Min, and Olafsson, 2001], we will construct the recommender system which can be effectively used in e-commerce systems.

5.2 Discretization and Feature Quality Evaluators

Discretization transforms values of a continuous feature into a finite number of discrete intervals based on criteria. Discretization usually performs two tasks, that are to find the number of intervals and the width or boundaries for the intervals, given the range of values of a continuous valued feature [Kurgan and Cios, 2001].

Generally, the discretization methods can be categorized one of the following areas;

- Unsupervised methods: discretizes continuous values of a feature without considering a class feature and interdependency with other features. Equal width and equal frequency discretization methods are the simplest unsupervised discretization methods. The former one just divides value range of a feature to a finite number of equal size bins where the number of bins is set by a user. The latter one divides the values of a features into a finite number of bins so that each bin contains equal number of values. Since the unsupervised methods do not consider class labels, it is more likely to lead to an undesirable classification by much possibility assigning values in different classes into same bin [Kerber, 1992].
- Supervised methods: discretizes continuous values by taking into account interdependency with a class and other features. As a statistical based method, the ChiMerge [Kerber, 1992] and StatDisc [Richeldi and Rossotto, 1995] are well known. The ChiMerge proceeds by testing whether to merge adjacent intervals based on the χ^2 statistics from one interval containing a initial real value. The StatDisc discretizes automatically from hierarchy of discretization intervals created by using a criterion measure for merging intervals until threshold of the measure is achieved. A number of entropy based methods have been proposed. One of the methods employs a recursive minimization entropy heuristic associated with Minimum Description Length as a stopping criteria to determine the number of intervals [Fayyad and Irani, 1993]. That will be used for discretization of continuous values before applying information gain in the test performed in this chapter.

Although many discretization methods have been shown to have good performances, this approach has a disadvantage in that it does not use characteristics of continuous values itself. Thus we will use two feature quality evaluators, correlation based evaluator and ReliefF as stated in the introduction for determining a partition order in the NP-Filter.

For a correlation based feature evaluator, we modify equation (3.7) which was originally developed for evaluating feature subsets [Hall, 1998]. It simply calculates the correlation between feature and class, which can be regarded as a Pearson's correlation.

ReliefF is an extended version of RELIEF developed by Kira and Rendell (1992) for estimating the quality of features that consider interdependency between features. While RELIEF can deal with discrete and continuous features, it has problems to deal with incomplete (missing values) data and multi class problems. Since ReliefF was developed to solve the problems, it will be one of the feature quality evaluators adapted to handle mixed type of features.

Searching for its two nearest neighbors, one from the same class (nearest hit) and the other from a different class (nearest miss) given an instance, RELIEF estimates qualities of features based on how well each can separate neighbor instances from different classes by having different values and have the same values for neighbor instances from the same class [Kononenko, 1994]. The RELIEF randomly selects m training instances, where m is the user-defined parameter and usually set to 10 that has been believed to perform well in many cases.

```
for i = 1 to n
    Q[A_i] = 0.0
for j = 1 to m
    randomly select an instance r
    find nearest hit h and nearest miss t
    for i = 1 to n
        Q[A_i] = Q[A_i] – diff(A_i, r, h) / m + diff(A_i, r, t) / m
```

Figure 5.1 Pseudocode of RELIEF algorithm [Kononenko, 1994].

In the pseudo code of RELIEF, for a discrete feature, *diff* (*Feature*, *Instance1*, *Instance2*) = 0, if the values are equal, otherwise it is 1, while for a continuous feature the difference is the actual difference normalized to the interval [0, 1]. A High value of $Q[A_i]$ has small amount of difference value for same class instances and large amount of difference for different class

As stated previously, ReliefF can handle incomplete and multi-class data. ReliefF finds one near miss $t(C)$ for each different class C and calculates weighted average with the prior probability of each class as follows; $Q[A_i] = Q[A_i] - diff(A_i, r, h) / m +$

$\sum_{c \neq class(r)} [P(C) \cdot diff(A_i, r, t(C))] / m$. Thus, it can estimate the capability of features to distinguish each pair of classes regardless of which two classes are closer to each other. Since the portion of dealing with incomplete data in the ReliefF is somewhat related to a feature filter, it is discussed further here.

Using correlation and ReliefF as a feature evaluator, we compare performances with information gain evaluator in perspective of the NP-Filter with several classifiers. The results are reported in the next section.

5.3 Analysis of Numerical Results

For fairness of this test, we add two more different kinds of data sets in addition to the data sets containing all nominal features in Table 3.2. The data sets can be categorized into three different domains in terms of data type of features, namely discrete, mixed (discrete and continuous) and continuous. However, all the data sets have a nominal class feature. This experiment addresses the effectiveness of a feature evaluator on performances in terms of accuracy, size, and computation time using the NP-Filter for the data sets as stated in Table 5.1, and evaluates if the evaluators perform differently according to each domain.

Table 5.1 Characteristics of the tested data domains.

Data Set	Instances	Features	Type of Features
lymph	148	18	all nominal
vote	435	16	
audiology	226	69	
cancer	286	9	
kr-vs-kp	3196	36	
anneal	898	38	6 continuous, 3 integer, others nominal
hepatitis	155	19	2 continuous, 4 integer, others nominal
credit-g	1000	20	7 continuous, others nominal
hypothyroid	3772	29	6 continuous, 1 integer, others nominal
labor	57	16	8 continuous, others nominal
vehicle	946	18	all continuous
glass	214	9	
ionosphere	351	34	
segment	2310	19	
diabetes	768	8	

Three classifiers, Naïve Bayes, C4.5, and 5-nearest neighbor are used to induce classification models with the selected features. For simplicity, the parameter k in the k-nearest neighbor learner is arbitrarily set to 5 since the number of selected features may vary for each run. Each case is repeated 5 times, and averages and standard deviations are reported.

First we consider the accuracy of the models induced after feature selection with three different feature evaluators compared to the corresponding models without feature selection, and second consider simplicity, that is a size of the models after feature selection is employed. Strictly speaking, the accuracy of the Naïve Bayes learner and size of selected features reported in Table 5.2 do not show a significant difference between feature evaluators. However, if we admit even small differences, the correlation evaluator has better performances in the discrete data set domain where 3 out of 5 data sets have higher accuracies. Here the better performance implies higher averaged accuracy and size. If a tie in average of accuracy occurs, lower standard deviation is better.

Table 5.2 Accuracy comparison of Naive Bayes with feature evaluators.

Data Set	Info. Gain		Correlation		ReliefF		NFS	
	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size
vote	*93.7±0.9	*5.8±0.8	93.3±1.3	5.8±2.4	93.0±0.7	7.0±1.0	90.1	16
audiology	70.1±2.3	27.0±4.0	*70.1±0.8	26.8±3.3	69.2±1.1	*21.2±4.4	71.2	69
cancer	73.9±0.4	5.4±0.5	*74.0±0.3	*5.4±0.5	72.8±1.6	5.4±0.9	73.4	9
kr-vs-kp	86.2±6.0	11.4±1.9	85.7±4.8	11.2±1.3	*89.0±8.3	*8.8±3.0	88.0	36
lymph	83.4±1.7	11.0±1.0	*84.5±1.9	*9.8±1.8	83.5±1.9	10.8±1.9	85.1	18
anneal	85.1±1.7	12.0±2.0	83.4±8.0	12.0±1.4	*86.2±2.8	*10.2±2.9	86.3	38
hepatitis	*85.3±1.1	10.4±1.1	85.0±0.8	*10.0±0.7	83.7±1.2	11.4±1.1	83.2	19
credit-g	73.6±0.8	*6.8±0.8	*74.8±0.8	8.0±1.0	73.5±1.7	9.0±1.7	74.8	20
hypothyroid	94.4±0.3	7.6±1.5	94.4±0.4	7.4±2.9	*94.4±0.3	*4.6±1.3	93.5	29
labor	91.2±0.0	5.8±1.5	92.3±1.0	*5.6±1.5	*93.0±0.0	7.2±2.6	96.5	16
vehicle	46.6±2.0	*9.8±2.6	*46.9±1.1	10.0±0.8	46.6±0.7	11.8±1.6	44.3	18
glass	48.6±0.0	7.0±0.0	48.6±0.0	7.0±0.0	*48.8±1.8	*5.4±1.1	45.8	9
ionosphere	88.7±1.6	*16.6±1.5	86.4±2.1	19.0±1.9	*88.7±1.4	18.0±3.2	83.2	34
segment	85.1±2.2	*6.4±1.1	*86.2±4.0	7.2±1.3	81.9±4.2	8.2±1.9	80.0	19
diabetes	*76.5±0.6	4.0±0.7	75.9±1.1	*3.8±0.8	75.9±0.9	4.6±0.9	76.2	8

Note: * implies the best performance among three evaluators.

Smaller size can be also regarded as better performance with a premise where there is significant difference in the number of selected features (size), but it does not present any

prominent pattern on results we can notice. In mixed type of data domain, the ReliefF shows better performances in 3 out of 5 data sets, and correlation and ReliefF report better performances in the mixed and continuous data type domain, which implies that capability handling a continuous value itself may affect the results.

In Table 5.2, last two columns show the results for no feature selection (NFS). First for the accuracy we note that it actually improves or is no worse when we use feature selection except just 4 data sets such as 'audiology', 'lymph', 'anneal', and 'labor'. Such improvement in accuracy may or may not occur as discussed previously, but feature dimension reduction, real contribution of feature selection resulting in simpler and easier to explain models, was accomplished.

Table 5.3 Accuracy comparison of C4.5 with feature evaluators.

Data Set	Info. Gain		Correlation		ReliefF		NFS	
	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size
vote	95.6±0.1	6.0±0.7	95.5±0.2	5.8±0.8	*95.6±0.0	*5.2±1.3	96.6	16
audiology	75.3±4.0	27.4±1.7	73.0±4.7	25.2±2.2	*76.2±1.3	*23.0±3.6	77.4	69
cancer	*74.6±1.1	5.8±1.1	74.3±0.9	5.8±1.3	73.2±2.8	*5.2±0.8	75.5	9
kr-vs-kp	*93.3±1.4	11.0±3.2	91.2±4.2	11.8±2.8	92.0±4.2	*10.4±2.5	99.5	36
lymph	*79.2±0.5	10.8±1.9	76.8±2.9	10.6±1.8	76.9±3.7	*8.6±1.9	78.4	18
anneal	96.6±2.0	*12.4±1.8	97.1±1.0	13.0±2.1	*97.7±0.4	12.8±1.1	98.6	38
hepatitis	*79.6±0.9	10.6±1.1	79.2±1.0	*9.4±0.5	78.9±1.9	11.6±0.9	78.1	19
credit-g	74.2±1.2	*6.2±1.3	*74.2±0.9	6.8±1.3	72.7±0.8	8.0±2.6	71.1	20
hypothyroid	97.0±0.4	5.8±2.2	*97.3±0.6	5.3±1.9	97.2±0.7	*4.8±1.9	99.7	29
labor	84.9±2.3	*6.2±1.5	83.9±1.5	7.4±1.1	*84.9±1.6	6.6±1.9	77.2	16
vehicle	69.1±1.8	10.0±1.0	69.8±1.6	*9.8±1.3	*71.7±2.3	11.0±2.4	73.5	18
glass	67.3±0.0	7.0±0.0	67.8±1.0	6.8±0.4	*67.8±1.3	*6.6±0.5	69.2	9
ionosphere	89.3±1.6	17.8±2.3	89.2±1.4	*16.6±1.5	*90.7±1.0	16.8±1.1	88.6	34
segment	96.6±0.3	8.2±1.3	*96.6±0.1	9.4±0.5	96.4±0.2	*4.8±0.8	97.0	19
diabetes	75.1±0.2	*3.4±0.5	75.3±0.6	4.4±0.5	*75.3±0.1	3.4±0.9	76.7	8

The accuracy results for C4.5 are reported in Table 5.3. These information gain evaluators applying entropy discretization performs better in the discrete data domain. In that the discrete domain does not need to employ discretization and C4.5 uses the information gain as a feature selection order criterion in a tree composition procedure, it is not surprising that the information gain evaluator may perform better in this domain.

Comparison results with no feature selection report that the accuracies of the model by the NP-Filter with C4.5 induction are a little degraded somewhat. However, we still have important benefit of reduction on feature dimension, and if we use NP-Wrapper, the better performance would be achieved as noted in Chapter 3.

In the mixed data type domain, the correlation and ReliefF methods perform better as expected in terms of accuracy. The ReliefF performs clearly better in 4 out of 5 data sets in the continuous data domain. For the number of selected features, even though the ReliefF evaluator in the NP-Filter performs pretty well in the discrete data domain, it does poorly in the mixed data domain.

Accuracies of 5-nearest neighbor in Table 5.4 have similar pattern as those of the previous two results. The number of selected features does not show any interesting results as similarly reported in the previous two tables.

Table 5.4 Accuracy comparison of 5-Nearest Neighbor with feature evaluators.

Data Set	Info. Gain		Correlation		ReliefF		NFS	
	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size
vote	94.3±1.4	*6.2±1.5	*94.7±1.0	7.0±2.5	94.6±0.9	7.2±0.8	94.3	16
audiology	*68.3±2.1	24.8±3.8	64.3±1.4	24.2±8.7	66.7±4.4	*20.0±1.7	61.1	69
cancer	71.3±0.0	5.4±0.5	72.7±1.5	*5.0±0.7	*74.8±1.0	5.0±1.9	72.7	9
kr-vs-kp	89.9±6.5	11.0±2.9	89.4±5.7	*11.0±2.4	*91.9±5.8	11.2±2.6	96.4	36
lymph	*83.5±0.8	10.8±1.8	81.1±1.7	*9.2±1.5	81.9±2.3	10.2±1.5	79.1	18
anneal	*97.0±0.9	12.6±0.9	96.2±1.0	15.0±4.0	95.1±1.5	*11.6±3.0	97.2	38
hepatitis	*83.6±1.1	10.2±1.3	83.0±1.9	*10.0±0.7	83.4±1.4	10.6±2.1	84.5	19
credit-g	72.3±0.6	7.8±1.3	*73.3±1.0	*5.8±1.3	71.8±1.2	8.8±2.1	72.4	20
hypothyroid	*96.4±1.5	*5.4±2.5	95.5±0.8	6.4±2.5	95.2±0.9	6.4±1.7	93.5	29
labor	88.8±1.6	7.8±1.1	89.8±2.3	6.6±1.9	*90.5±1.6	*6.6±1.1	87.7	16
vehicle	62.1±3.7	*10.0±1.4	63.0±2.3	10.8±0.8	*65.4±3.3	11.6±1.1	70.7	18
glass	*72.2±0.4	6.8±0.4	72.0±0.0	7.0±0.0	71.2±0.8	*6.4±0.5	65.4	9
ionosphere	*87.3±1.0	17.6±3.0	86.8±1.9	*16.8±1.8	86.3±1.0	18.0±1.0	85.5	34
segment	95.4±0.2	7.2±0.4	95.7±0.4	8.0±1.4	*95.8±0.2	*6.0±1.0	95.3	19
diabetes	73.2±0.4	*3.6±0.5	73.9±0.5	4.2±0.8	*74.4±0.9	4.2±1.1	73.6	8

Table 5.5 to Table 5.7 report computation time of the NP-Filter of feature evaluators induced by each classifier with no feature selection in three domain data sets. From these results we see that using no feature selection takes the least amount of time with no wonder and the results show that there are no significant differences between the evaluators in the

three domains. However it is interesting that the computation time is dependent upon each data set. For instance 'audiology' in discrete data, 'anneal' in mixed type data, and 'segment' in continuous data domain have surely the least computation time for the NP-Filter with all three classifiers when the ReliefF evaluator is applied. The correlation evaluator performs better in 'vote', 'hypothyroid', 'vehicle' and 'glass' data sets but not all cases. The information gain performs better in 'cancer' and 'ionosphere' data sets.

Table 5.5 Speed comparison of Naïve Bayes with feature evaluators.

Data Set	Info. Gain	Correlation	ReliefF	NFS
vote	4837±149	*4662±287	5051±176	641
audiology	45024±3406	45942±3199	*31607±5013	601
cancer	*1504±19	1536±25	1566±103	160
kr-vs-kp	321254±16262	337091±18306	*319022±19854	1002
lymph	2441±128	2477±86	*2361±184	170
anneal	74489±5341	73862±1775	*69764±4545	681
hepatitis	*3989±133	4000±65	4436±444	191
credit-g	*34159±341	35867±958	40289±6692	501
hypothyroid	438526±7751	*437735±12466	549490±109546	2163
labor	*1596±41	1646±43	1630±70	160
vehicle	60422±8229	*56741±2841	58562±6598	721
glass	*4899±8	5215±237	5039±436	241
ionosphere	*111799±2555	116321±2203	119756±7895	521
segment	416735±146156	357812±23330	*341251±2260	2143
diabetes	*11282±66	11308±147	11308±63	391

Table 5.6 Speed comparison of C4.5 with feature evaluators.

Data Set	Info. Gain	Correlation	ReliefF	NFS
vote	4848±167	*4802±106	4995±186	641
audiology	46348±2082	46092±4201	*32278±3925	1412
cancer	*1717±51	1732±66	1849±220	430
kr-vs-kp	*329669±19654	335900±14074	340483±17674	5999
lymph	2584±217	2688±191	*2369±178	381
anneal	74240±3579	76870±3168	*72514±2696	4346
hepatitis	4212±98	*4204±66	5397±1166	551
credit-g	*34673±1046	35709±857	39299±2801	2894
hypothyroid	432754±11182	*429675±8953	504808±41381	5709
labor	*1702±42	1738±34	1720±62	280
vehicle	64887±13926	*58302±4186	59057±4581	4847
glass	5802±574	*5532±42	5539±71	1021
ionosphere	*113088±1604	114985±2864	121773±14359	3565
segment	472592±234259	375860±24725	*342728±1323	12899
diabetes	11855±89	12033±103	*11809±173	1732

From the results, we could conclude that the computation time of the NP-Filter with a feature evaluator is different from each data set regardless of a data domain and classifier. Even though no feature selection takes the least time, the benefits from feature selection can compensate for loss on the computation time.

Table 5.7 Speed comparison of 5-Nearest Neighbor with feature evaluators.

Data Set	Info. Gain	Correlation	ReliefF	NFS
vote	*5313±218	5586±523	5858±156	1792
audiology	43695±5701	49096±4750	*30223±1481	1852
cancer	1758±22	*1726±56	1780±99	541
kr-vs-kp	*392868±25370	398321±25837	411211±16114	179859
lymph	2525±184	2579±124	*2389±147	330
anneal	79706±2529	84954±6919	*79634±3634	16884
hepatitis	4106±102	*4098±111	4683±497	381
credit-g	39509±1202	*37744±1558	44248±2201	11176
hypothyroid	*483557±32508	513252±29240	579391±69023	219035
labor	1670±39	*1640±62	1660±44	160
vehicle	62027±4015	*61308±5541	64496±6698	9103
glass	5059±57	*5055±4	5075±35	431
ionosphere	*114042±3452	117138±5827	119838±7773	2444
segment	415931±48566	378856±7791	*370354±5679	65454
diabetes	*12738±206	12946±327	12962±392	3245

5.4 Case Study - Recommender Systems

In e-commerce applications, recommender systems intelligently suggest products to users as well as provide information that helps users make a decision which products to be of interest, based on rules or knowledge extracted from data storing observed user behavior and experience profiles. The success of such recommender systems depends on how good quality of recommendations can be made to users. It is very important to recommend the correct products users since such a recommendation lead to have users positively respond.

In this section, the recommender system for an Internet auction system [Ryan, Min and Olafsson, 2001] to facilitate reverse logistics is provided using classification and association rules. The Internet auction system is intended to bring together a fragmented market of manufacturers, recyclers, demanufacturers and others interested in the take-back and reuse of

obsolete recyclable products. To support such an auction system, the recommender system is proposed that among other functions recommends to a user if they should participate in a particular auction. However, it is equally important that a user understands what goes into such recommendation and thus, feature selection is used to build a simple recommendation model that can be easily explained. This is a critical aspect due to the fragmentation of the market, which implies that many potential users are unfamiliar with each other and the range of available products. The new feature selection algorithm applying a nested partition method is used for reducing the number of features.

5.4.1 Recommender System for Reverse Logistics Internet Auction

This auction system simulates the growing industry of electronics recycling. The reverse logistics or returned products from consumers can be regarded as a movement process of various types of products or raw materials back from consumers for any reasons. The Internet auction system facilitates the process of reverse logistics for recycling. Three types of participants, three manufacturers, three demanufacturers, and three recyclers, participate an auction for 6 rounds. In each round, participants can bid price and quantity of items to sell or to buy [Ryan, Min and Olafsson, 2001].

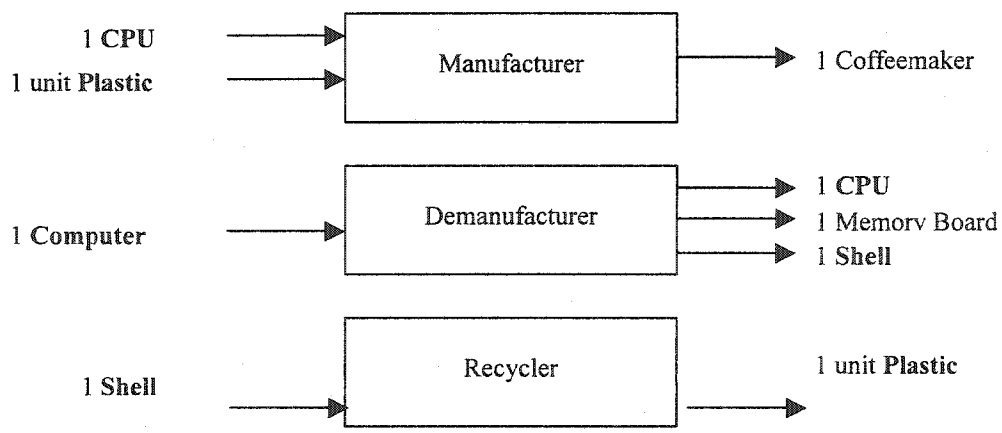


Figure 5.2 Relationships among products traded online (bold) and sold externally [Ryan, Min, and Olafsson, 2001].

As seen in Figure 5.2 and Figure 5.3, Each manufacturer sells computers, and produces and sells coffeemakers outside the auction, each of which consists of 1 CPU and 1 unit of Plastic that can be purchased from the auction. Demanufacturers buy computers from manufacturers and disassemble into CPUs, memory boards, and shells. Demanufacturers can sell memory boards outside the auction. Recyclers buy shells from demanufacturers and transforms them into plastics. However, the participants can participate in any auction they want.

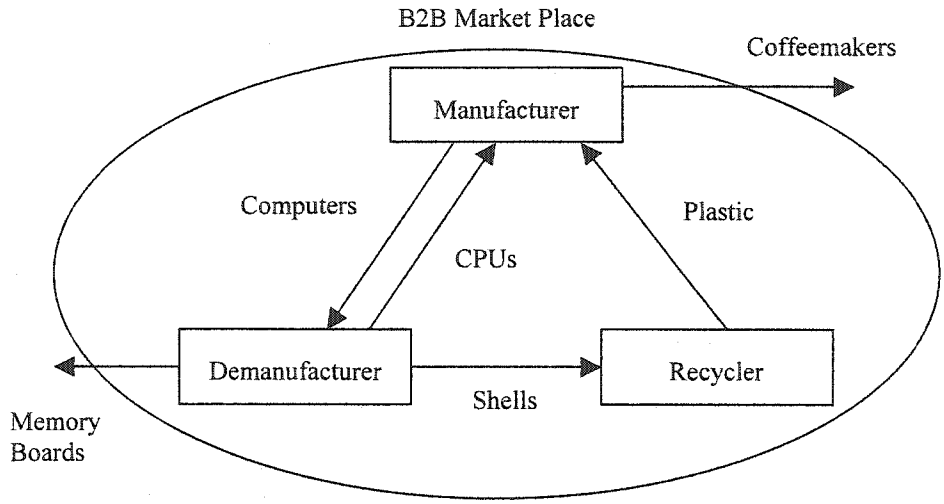


Figure 5.3 Relationships among participants in the online market place [Ryan, Min, and Olafsson, 2001].

We constructed two data sets from a database gathered in that prototyped auction system, namely auction bids and auction participation data sets. Based on the data sets, we construct recommender systems that consist of classification, association, and feature selection.

5.4.2 Recommendation for Auction Participation

In order to figure out which auctions can attract participants, the data set below is constructed by consisting of 31 features and 1 class feature. Those features state behaviors of each participant in each auction round for a specific item as well as characteristics of the auction itself and the class feature in the data is YES/NO depending on if the user participated in a particular auction or not. The detailed description of which is shown in Table 5.8. The data set contains 2592 instances, each of which states an example of an

auction including historical and current data of each participant. Using classification and association, specifically C4.5 and Apriori, we construct recommender system to generate concrete rules that can figure out which auctions have users interested. We expect to create rules that may catch out some implicit facts, not just obviously simple ones.

Table 5.8 Data set description for auction participation.

Type of Feature	Feature	Values
Product Description	Item	{Computer, CPU, Shell, Plastic}
	ItemContainsCPU	{T, F}
	ItemContainsShell	{T, F}
	ItemContainsPlastic	{T, F}
Round Information	Round	{1, 2, 3, 4, 5, 6}
Information on Last Transaction to Occur	PriceForLastComputerAuction	Numeric
	QuantityForLastComputerAuction	Numeric
	PriceForLastCPUAuction	Numeric
	QuantityForLastCPUAuction	Numeric
	PriceForLastShellAuction	Numeric
	QuantityForLastShellAuction	Numeric
	PriceForLastPlasticAuction	Numeric
	QuantityForLastPlasticAuction	Numeric
Information on Last Transaction with Participants Involvement	TimeOfParticipantsLastComputerAuction	Numeric
	PriceForParticipantsLastComputerAuction	Numeric
	QuantityForParticipantsLastComputerAuction	Numeric
	CategoryOfParticipantsLastComputerAuction	{buy,sell,none}
	TimeOfParticipantsLastCPUAuction	Numeric
	PriceForParticipantsLastCPUAuction	Numeric
	QuantityForParticipantsLastCPUAuction	Numeric
	CategoryOfParticipantsLastCPUAuction	{buy,sell,none}
	TimeOfParticipantsLastShellAuction	Numeric
	PriceForParticipantsLastShellAuction	Numeric
	QuantityForParticipantsLastShellAuction	Numeric
	CategoryOfParticipantsLastShellAuction	{buy,sell,none}
	TimeOfParticipantsLastPlasticAuction	Numeric
	PriceForParticipantsLastPlasticAuction	Numeric
	QuantityForParticipantsLastPlasticAuction	Numeric
	CategoryOfParticipantsLastPlasticAuction	Numeric
Information on Participant	ParticipantName	{M1, M2, M3, D1, D2, D3, R1, R2, R3}
	ParticipantType	{Manufacturer, Demanufacturer, Recycler}
Class	ParticpateInAuction	{yes, no}

Even though we do not apply any systematic method to a recommender system, intuitively we could recommend computers to manufacturers and demanufacturers, plastic to recyclers, and so on, since they need specific products according to their purposes. However,

it is just simple and clumsy recommendation which does not provide much contribution. Thus, we apply classification rule, C4.5, so that recommendation rules are created in the following figures.

Figure 5.4 shows top nodes of the decision tree. The top node, Participant Type, is branched into Manufacturer, Demanufacturer and Recycler. Since feature selection is not applied to generate this tree, the size is too big to be shown in one simple tree. Next several figures represent the decision tree for auction participation.

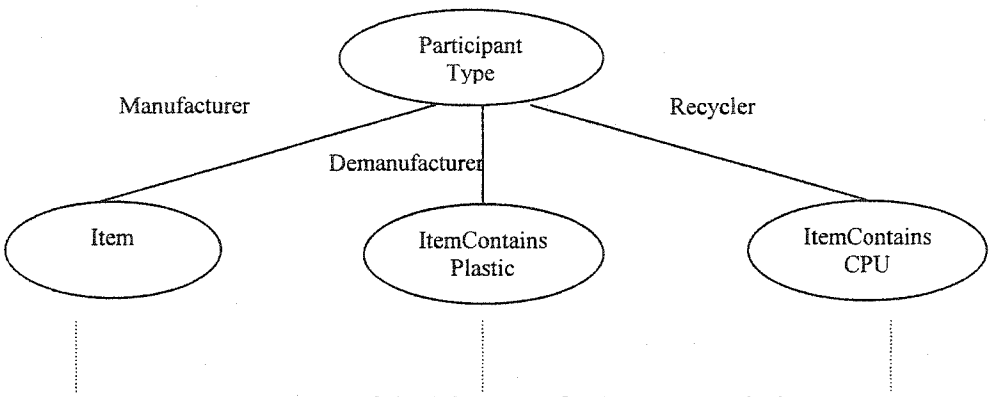


Figure 5.4 Root of decision tree for recommendation.

First let's consider auction behaviors of manufacturers and make some interesting rules (See Figure 5.5). Simply we can figure out several rules or conditions from the tree that are used to recommend the auction to manufacturers as shown in following examples.

CPU auction recommendation:

- If the item is CPU, then we *recommend* the auction.

The first rule about the CPU auction is of course obvious as stated previously and does not provide any meaningful context. Computer and Shell auctions for manufacturers are more complicated than that of CPU.

Computer auction recommendation:

- If the time since the manufacturer last participated in a computer auction is less than or equal to 1 round, then *recommend* the auction.
- Else if the time since the manufacturer last participated in a computer auction is greater than 1 round and the manufacturer bought computers in the last auction, then *recommend* the auction.

- Else if the manufacturer sold computers, then *do not recommend.*

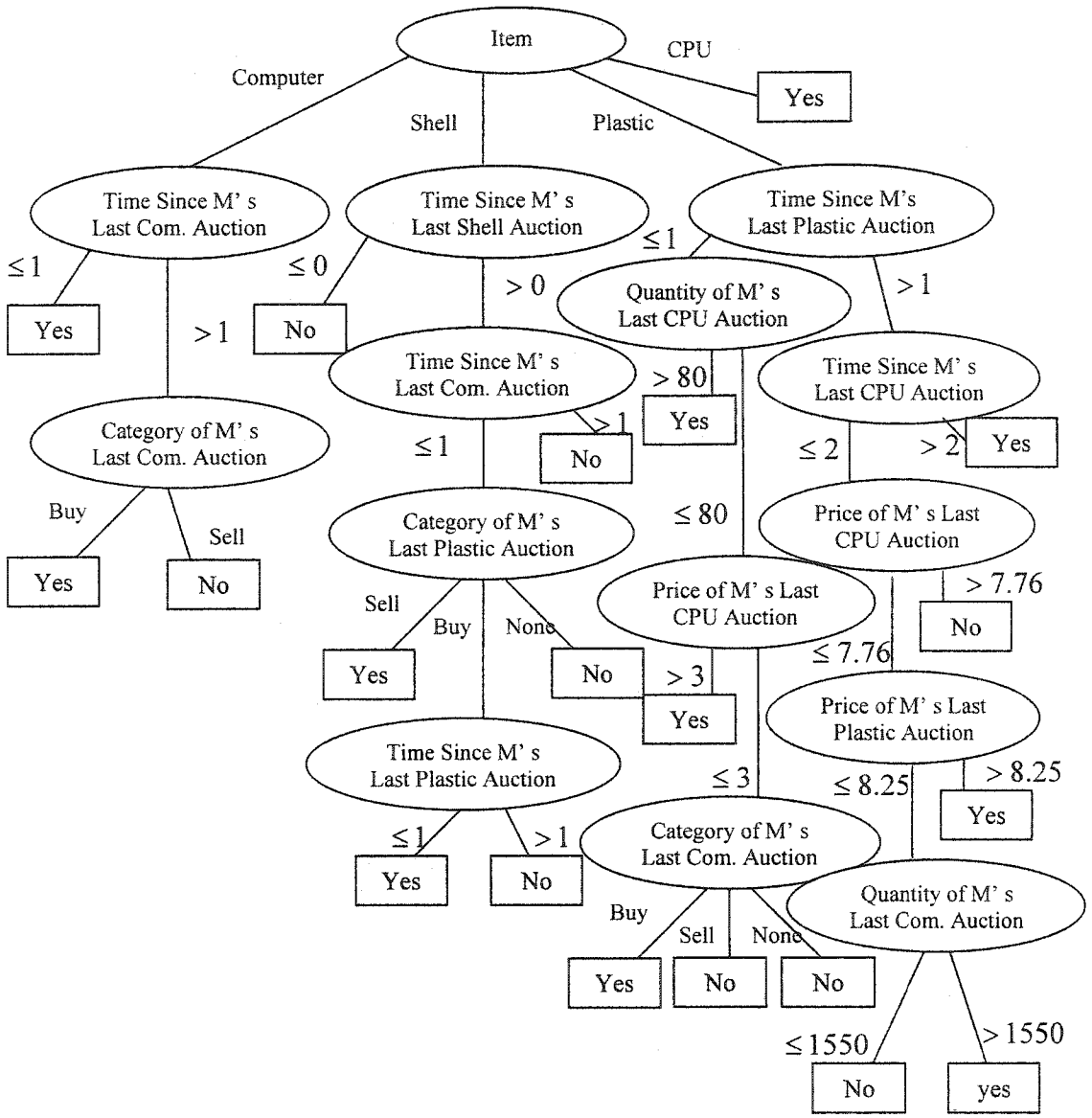


Figure 5.5 Decision tree of manufacturers' auction recommendation.

It may be interpreted as follows; if the manufacturer participated in the computer auction right before the current round or has never been participated, the manufacturer want to sell computers, which is natural. Otherwise if the manufacturer did not participate in the last round of a computer auction and has bought computers previously, the manufacturer wants to participate in the current auction. That implies that the manufacturer bought computers to

earn more money that can be used to buy CPU and Plastic in the last auction, or the manufacturer want to sell computers that have left up to the current round. That is not surprising in that everyone wants to maximize a profit. Thus, even though manufacturers sold all computers they had, they would want to make more coffeemakers to be sold continuously, which means they act as a broker. Thinking collectively, we recommend a computer auction to manufacturers inferred as they have unsold computers.

Let's look into Shell auction recommendation rules:

- If the manufacturer has never participated in a shell auction, then do *not recommend* the auction.
- Else if the time since the manufacturer last participated in a shell auction is greater than 0 and the time since the manufacturer last participated in a computer auction is greater than 1, then do *not recommend* the auction.
- Else if the manufacturer was to sell plastics in the last round, then *recommend.*
- Else, do *not recommend* the auction.

It is inferred that manufacturers are interested in shell auctions when they want to act as a broker. For example if manufacturers have sold plastics, actually a behavior of a broker, shell auctions should be recommended.

When it comes to Plastic auction recommendation, the following selected rules might be created without pointing out concrete values of price and quantity:

- If the time since the manufacturer last participated in a plastic auction is less than or equal to 1,
  - If the manufacturer traded large amount of CPU (greater than 80) last, then *recommend* the auction.
  - Else if the manufacturer traded small amount of CPU last with higher price, then *recommend* the auction.
- Else if the manufacturer did not participated in a last plastic auction (> 1),
  - If the time since the manufacturer last participated in a CPU auction is greater than 2, then *recommend* the auction.
  - Else if the time since the manufacturer last participated in a CPU auction with higher price is less than or equal to 2, then do *not recommend* the auction.

Since these rules refer to values of price and quantity of items, it is hard to infer meaningful context. However in a word it seems that if manufacturers bought CPU in previous rounds, they need to buy plastics to produce coffeemakers. Thus we recommend the plastic auction. In addition, if they did not participate in plastic or CPU auction for a while, the auction should be recommended since they have to make coffeemakers continuously.

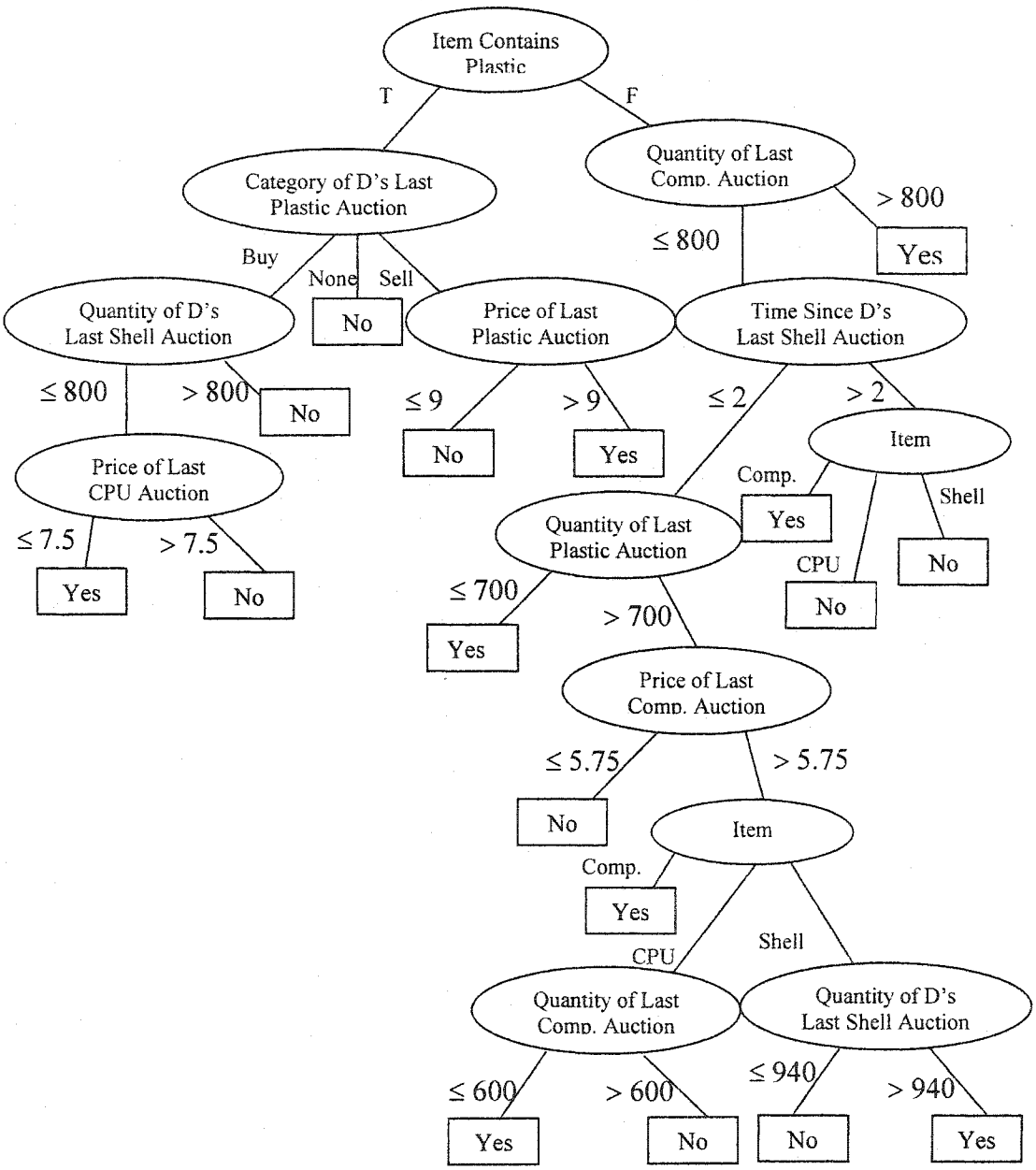


Figure 5.6 Decision tree of demanufacturers' auction recommendation.

Now let's look at the decision tree for demanufacturers (See Figure 5.6). The top node of demanufaturers splits according as an item is plastic or not. Intuitively if the item is plastic, the demanufacturer wants to act as a broker since a plastic is not a part of memory boards. If the item is plastic, the recommendation rules for demanufacturers are as follows;

- If the demanufacturer sold plastics last with higher price, then *recommend* the auction.
- Else if the demanufacturer bought plastics last and traded large amount of shells last, then do *not recommend* the auction.
    - Else if the demanufacturer traded small amount of shells last and CPUs were traded with low price last, then *recommend* the auction.
    - Else, do *not recommend* the auction.

Since demanufacturers behave as a broker, they would be interested in auctions that they can make much profit, which is directly reflected in the first rule. The rest of rules imply that if they did not earn money from shell and CPU auctions, they would turn their interest to the plastic auctions to make money through the broker's behaviors.

If an item does not contain plastic, the item is one of computer, shell, and CPU. In this branch, the demanufacturers show a pattern where they concentrate on their own role, that is, buying computers and selling CPU and shell. For example, if computers were traded largely in last time, the auction is recommended. Even though the small amount of computers were traded last, the computer auction is recommended if the demanufacturer did not participated in shell auctions recently. Furthermore, if the demanufacturer participated in shell auctions recently but small amount of plastic was traded, the auction is recommended, which indicates that demanufacturers are not interested in plastics. Other subtrees can be interpreted very similarly with regard that for items not containing plastic, it seems that demanufacturers do not follow the role of a broker.

Briefly, for plastic items demanufacturers show general characteristics for a broker. On the other hand, for non plastic items they act their own basic role, that is buying computers and selling shell and CPU. However this interpretation could be different from an analyst. For more precise analysis, larger and more concrete data sets should be needed.

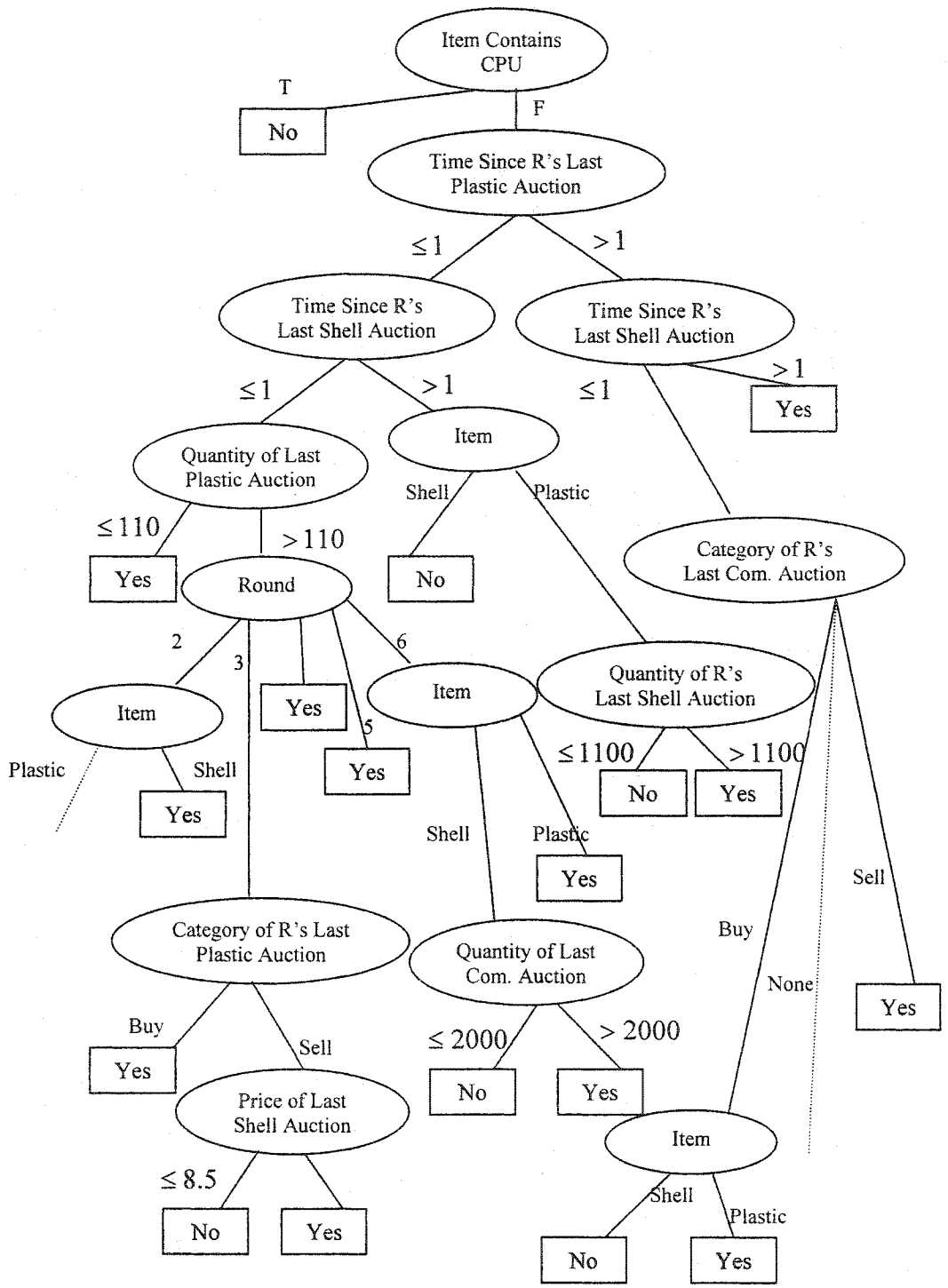


Figure 5.7 Decision tree of recyclers' auction recommendation.

For recyclers, what kind of an auction should be recommended? In a word, recyclers do not act behaviors of a broker in that the first node splits depending on whether or not the item contains CPU. The items that contain CPU are computer and CPU itself. Thus from the tree, it is easily found that the auction for items containing CPU is not recommended to recyclers, which implies that recyclers do not participate in auctions as a broker only. If items do not contain CPU, that is shell or plastic, the following rules can be selectively made as follows;

- If the time since the recycler last participated in a plastic auction is greater than 1 and the time since the recycler last participated in a shell auction is also greater than 1 (did not participated in plastic and shell auction last time), then *recommend* the auction.
- Else if the recycler participated in a plastic auction last time but did not participated in a shell auction last time, then the auction is *recommended* if the item is plastic and the recycler bought large amount of shells last time. Otherwise it is *not recommended* if the item is shell.
- Else if the recycler participated in a plastic auction last time and also participated in a shell auction last time but small amount of plastic was traded, then *recommend* the auction.

The other rules depend on the auction round, price, quantity, and some conditions. First let's consider the first rule above. It is not surprising in that if the recycler did not participate in plastic and shell auction in the last round, business transactions for those parts would be definitely needed to continuously make profit. The second rule implies that recyclers would not like to buy shells anymore and be interested in selling inventory of plastics. The third rule would be also understood in accordance with the second one.

We investigated recommendation rules based on decision tree induced classification algorithm, C4.5, using a full size of data set with 32 features and 2592 instances. Generally, most learning algorithms are not scalable with the increasing number of features. Thus it is critical to reduce the feature dimension for faster induction of classification. Even though the number, 32, in this prototyped data set is quite manageable, real data sets would have huge amount of data. In order to reduce the feature dimension, the NP-Wrapper is used with C4.5 classification learning algorithm. Hence, we have much smaller size of data set with 10 selected features but even accuracy improvement, 84.7% versus 85.5%.

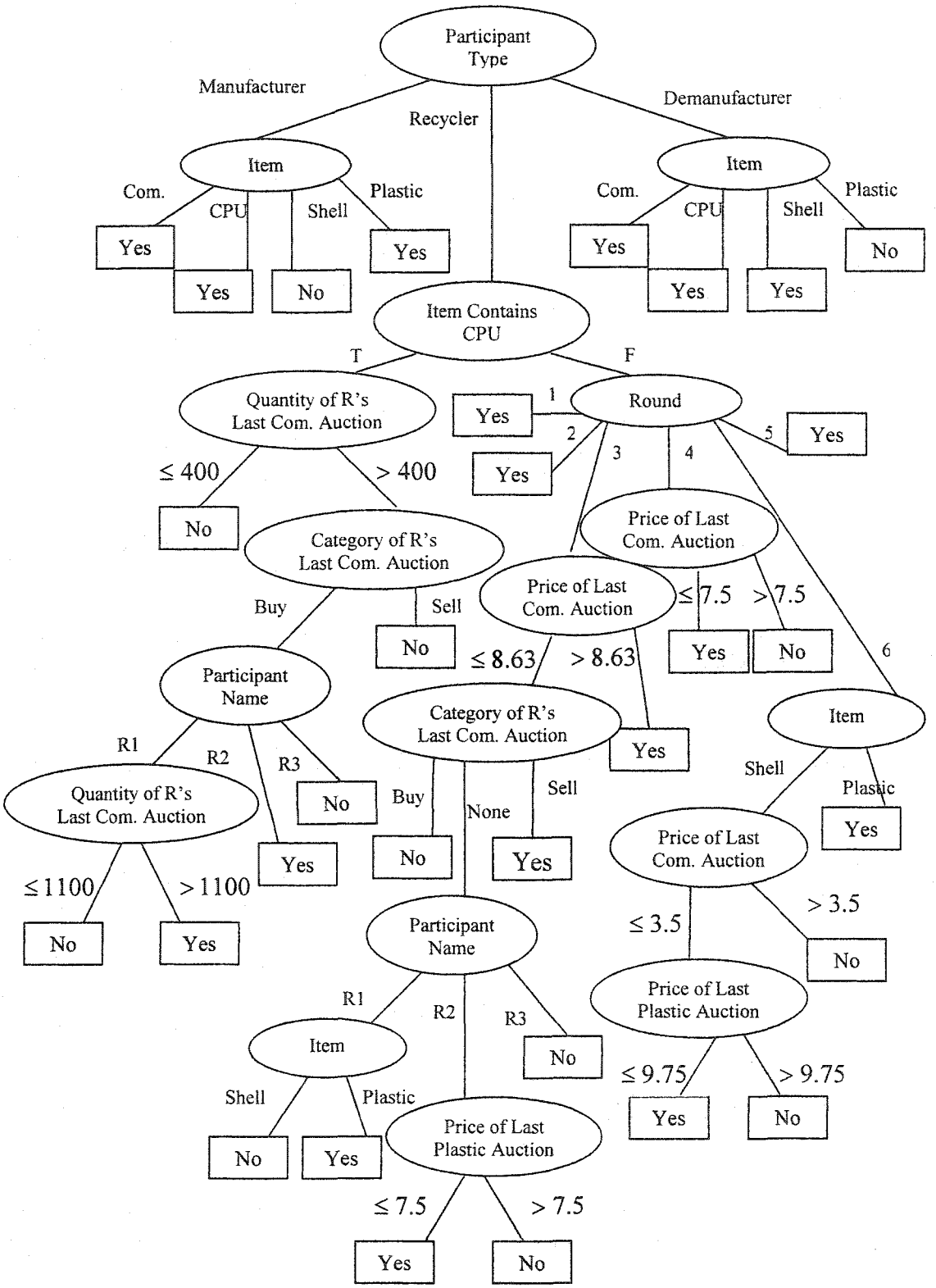


Figure 5.8 Decision tree reduced by feature selection.

The selected 10 features are as follows;

- Item
- ItemContainsCPU
- ItemContainsShell
- Round
- PriceForLastComputerAuction
- PriceForLastPlasticAuction
- QuantityForParticipantsLastComputerAuction
- CategoryOfParticipantsLastComputerAuction
- ParticipantName
- ParticipantType

The induced decision tree based on the reduced data set is shown in Figure 5.8. Comparing time to build the model, 1.94 seconds for full data set is significantly reduced to 0.64 seconds for reduced data set. Of course, the size of the tree also becomes much smaller, which means that the small size of decision tree can be more easily interpreted and provide clear recommendation.

One major difference with the full size decision tree is that shell auctions are not recommended to manufacturers and plastic auctions are not recommended to demanufacturers. This fact would imply that recommendation rules derived from the full size tree where manufacturers and demanufacturers act brokering behaviors for the non-direct trade product (shell for manufacturers and plastic for demanufacturers) cannot be always applied, that is, those rules do not clearly represent auction behaviors of manufacturers and demanufacturers. The role of them as a broker is not a general action pattern for the auction participation. Of course, they might act as a broker for the direct trade items (e.g. computers for manufacturers). However, it seems more reasonable that manufacturers and demanufacturers might not be interested in brokering behaviors in auctions.

The decision tree for recyclers has very similar structure of the full size decision tree. For example, recyclers do not participate in the auction for items containing CPU, which is exactly same as one from the full size tree. And other subtrees show that they are primarily interested in shell and plastic auctions.

We just investigated recommendation rules using a classification rule, C4.5. Now we apply association rule called Apriori to create a recommender. Given a set of instances each of which contains some number of items from a given collection, association rules produce dependency rules which will predict occurrence of an item based on occurrences of other items. The following association rules are selectively chosen among ones derived from the reduced data set.

- Price for last computer auction = (-inf-1.26] and Price for last plastic auction = (-inf-1.4] ==> Quantity for participants last computer auction = (-inf-220]
- Price for last computer auction = (-inf-1.26] ==> Quantity for participants in last computer auction = (-inf-220]
- Price for last computer auction = (-inf-1.26] and Quantity for participants in last computer auction = (-inf-220] ==> Price for last plastic auction = (-inf-1.4]
- Item containing CPU = True and Participant type = Recycler ==> Participate in auction = no
- Price for last computer auction = (-inf-1.26] ==> Quantity for participants in last computer auction = (-inf-220]
- Price for last computer auction = (-inf-1.26] ==> Price for last plastic auction = (-inf-1.4]

Let's look into the first rule above. If computers and plastics were traded with cheap prices in the last time, participants traded small amount of computers in last time. It is natural that the price of computers affects the quantity of computers of participants. However, we can infer that the price of plastic also affects the quantity, even though the exact reason is not explicitly known. The third and last rule could provide a clue in that the rule, 'if computers were traded with cheap price in the last time, then plastics were traded also with cheap price', can be explained as participants (probably manufacturers) who did not make much money from the computer auction are not affordable to pay for high-priced plastics. The fourth rule provides exactly same result as one from decision tree in that recyclers do not participate in auctions for item containing CPU, which supports the rule that recyclers do not act as only a broker.

5.4.3 Recommendation for Auction Bid

In addition to identifying which auctions are of interest, it may be also of interest to predict the probability of a bid success/failure, and even predict a likely settlement price. Through similar ways performed in the previous section, we explore what auctions should be recommended using classification algorithm, C4.5. To do that the following data set is constructed considering participant information and basic bid statistics on previous auctions with three class features (See Table 5.9). The data set consists of 30 features including class features and 2592 instances reflecting results of previous auction bids.

This data set has three class features, settlement price, quantity traded and bid success. Specifically, the bid success features is a calculated one, awarded quantity / traded quantity, which implies that 1 represents acquiring all products, fractional value does partial success, and 0 means failure on bids. In case that the traded quantity is 0, it is classified to 'no trade' since it has a different meaning from failure (none). When this data set is used in classification process, one of three features is included in the data set.

As expected this data set consists of almost numeric features, the induced decision tree is too complicated to be interpreted. Thus we applied NP-Wrapper with C4.5 classification algorithm and ReliefF feature evaluator so that the following 12 features were selected. In this case, the bid success feature was used as a class feature.

- AverageSettlementPrice
- TotalQuantityTraded
- HighestAskBid
- LowestSellBid
- HighestSellBid
- LastSettlementPrice
- LastQuantity
- LastNumberOfAskBids
- LastLowestAskBid
- LastLowestSellBid
- Category
- BidPrice

Table 5.9 Data set description for auction bid (2592 instances, 27 features + 3 class).

Type of Feature	Feature	Values
Participant Information	Item	{Computer, CPU, Shell, Plastic}
	Participant Name	{M1, M2, M3, D1, D2, D3, R1, R2, R3}
	Participant Type	{Manufacturer, Demanufacturer, Recycler}
Accumulative Item	Average Settlement Price	Numeric
	Total Quantity	Numeric
	Overall number of ask bids	Numeric
	Overall number of sell bids	Numeric
	Overall lowest ask bid	Numeric
	Overall highest ask bid	Numeric
	Overall lowest sell bid	Numeric
	Overall highest sell bid	Numeric
Last Auction for Item	Last Settlement Price	Numeric
	Last Quantity	Numeric
	Last Number of ask bids	Numeric
	Last Number of sell bids	Numeric
	Last Lowest ask bid	Numeric
	Last Highest ask bid	Numeric
	Last Lowest sell bid	Numeric
	Last Highest sell bid	Numeric
Participants Last Auction for Item	Last Bid Category	{buy, sell, none}
	Last Bid/Ask Price	Numeric
	Last Requested Quantity	Numeric
	Last Awarded Quantity	Numeric
Participants Current Auction	Category	{buy, sell, none}
	Bid/Ask Price	Numeric
	Requested Quantity	Numeric
	Awarded Quantity	Numeric
Class Variables	Settlement Price	Numeric
	Quantity Traded	Numeric
	Bid Success	{none, partial, all, notrade}

The accuracy is improved 77.4% of the full size decision tree to 81.5%. The time to build a model is also reduced significantly, 3.84 seconds versus 1.95 seconds. The Figure 5.9 – 5.11 show the decision tree induced from the reduced data set.

Let's look into the decision tree one after another from the top node. The tree begins splitting the node, Bid Price, by determining whether or not a participant makes a bid in an auction. If the bid price is 0, it would result in either none or no trade since the participant did not make a bid in the auction. Thus we exclude the part for the bid price less than or equal to 0 in our analysis to create recommendations. The next node is Total Traded Quantity up to the current round auction. Figure 5.9 shows the part that the total traded quantity is less than or equal to 100.

Synthetically, the decision tree shows which conditions can be classified to one of bid results such as all, partial, none, or no trade. Let's consider the following recommendation rules selectively chosen.

- If the total traded quantity is small and the average settlement price is low, then the bid results in no trade.
- Else if the average settlement price is high and the last settlement price is also high, then the bid results in no trade.

The rule above shows some cases that the bid results in no trade, which means that nobody did not succeed in that auction without considering whether or not participants made bids. The first rule is natural since the auction that no one wins has no settlement price and no traded quantity. However the second rule is a little more interesting than the first one since it gives a notion that no one participates in the auctions titled a high price.

Let's further consider rules starting from the next node to the last settlement price, Highest Ask Price.

- If the highest ask bid is low but the current bid price is high, then the bid results in no trade.
    - If the ask bid price is low, then the bid results in none.
    - Else if the ask bid price is in average range and the average settlement price is low, then the bid results in all.
        - If the average settlement price is high and the bid price is much higher than the average settlement price, then bid results all
        - Else, then the bid results in none.
- Else if the highest ask bid is high but the bid price is much lower than the highest ask price, then the bid results in none.
    - If the bid price is high but lower than the highest ask price, and the average settlement price is low, then the bid results in partial.
        - If the average settlement price is as high as the highest ask price and the bid price is also high, then the bid results in all.

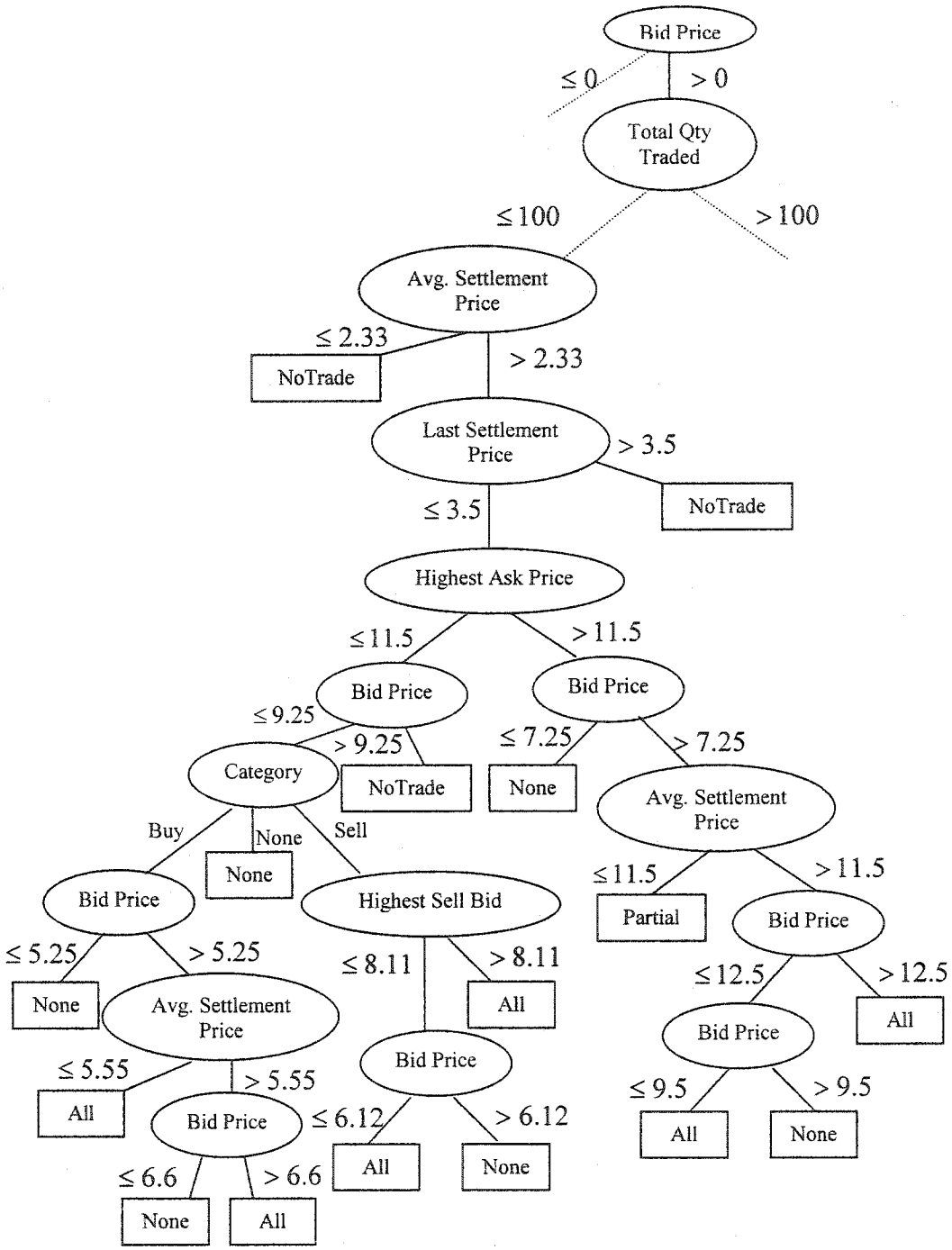


Figure 5.9 Decision tree for auction bid recommendation where the total traded quantity is less than or equal to 100.

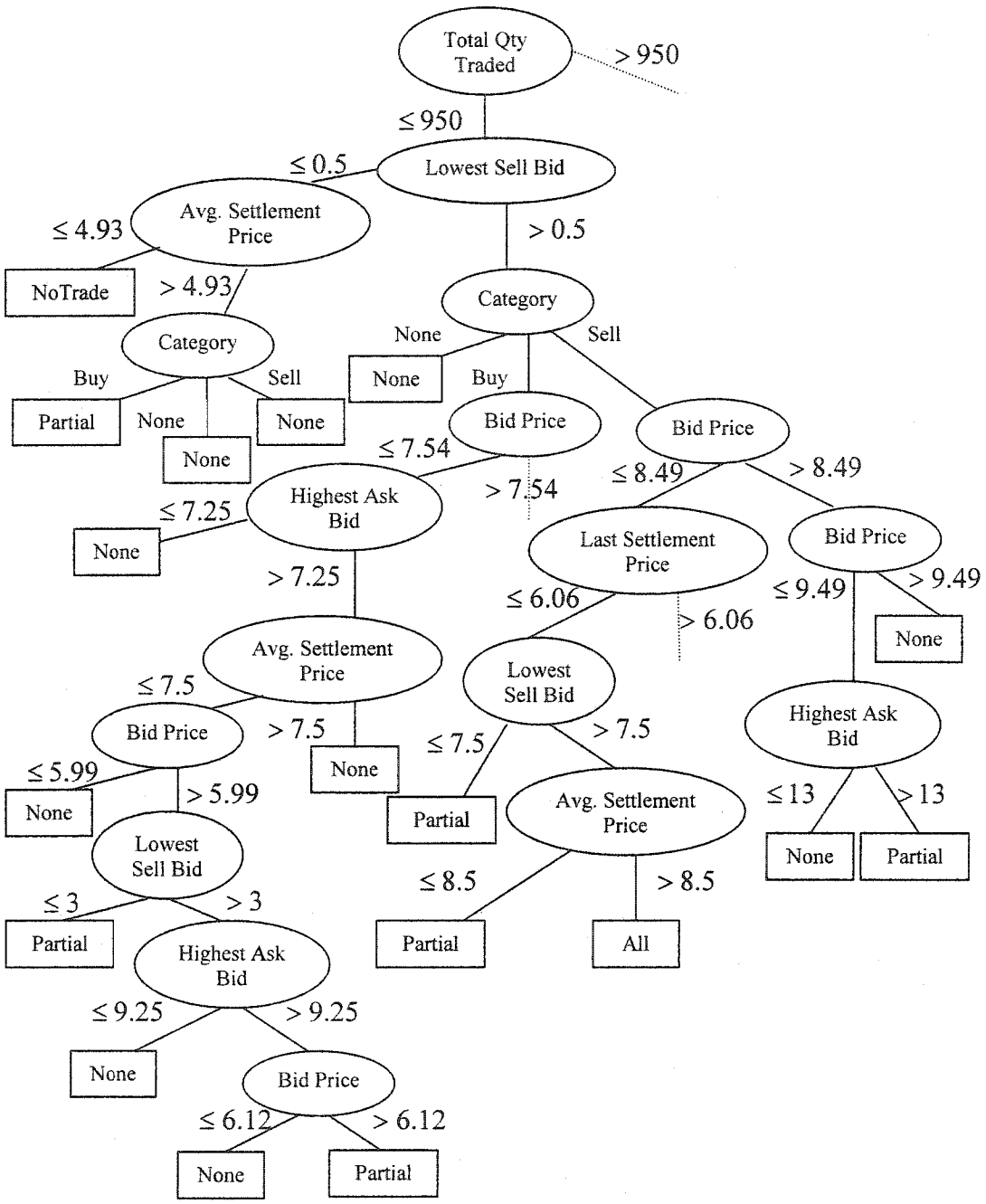


Figure 5.10 Decision tree for auction bid recommendation where the total traded quantity is greater than 100 but less than or equal to 950.

Since describing a value of price in rules is meaningless, just simple terms, namely high and low were used. Those rules might be intuitively guessed but can be used to make recommendations to help users bid more confidently.

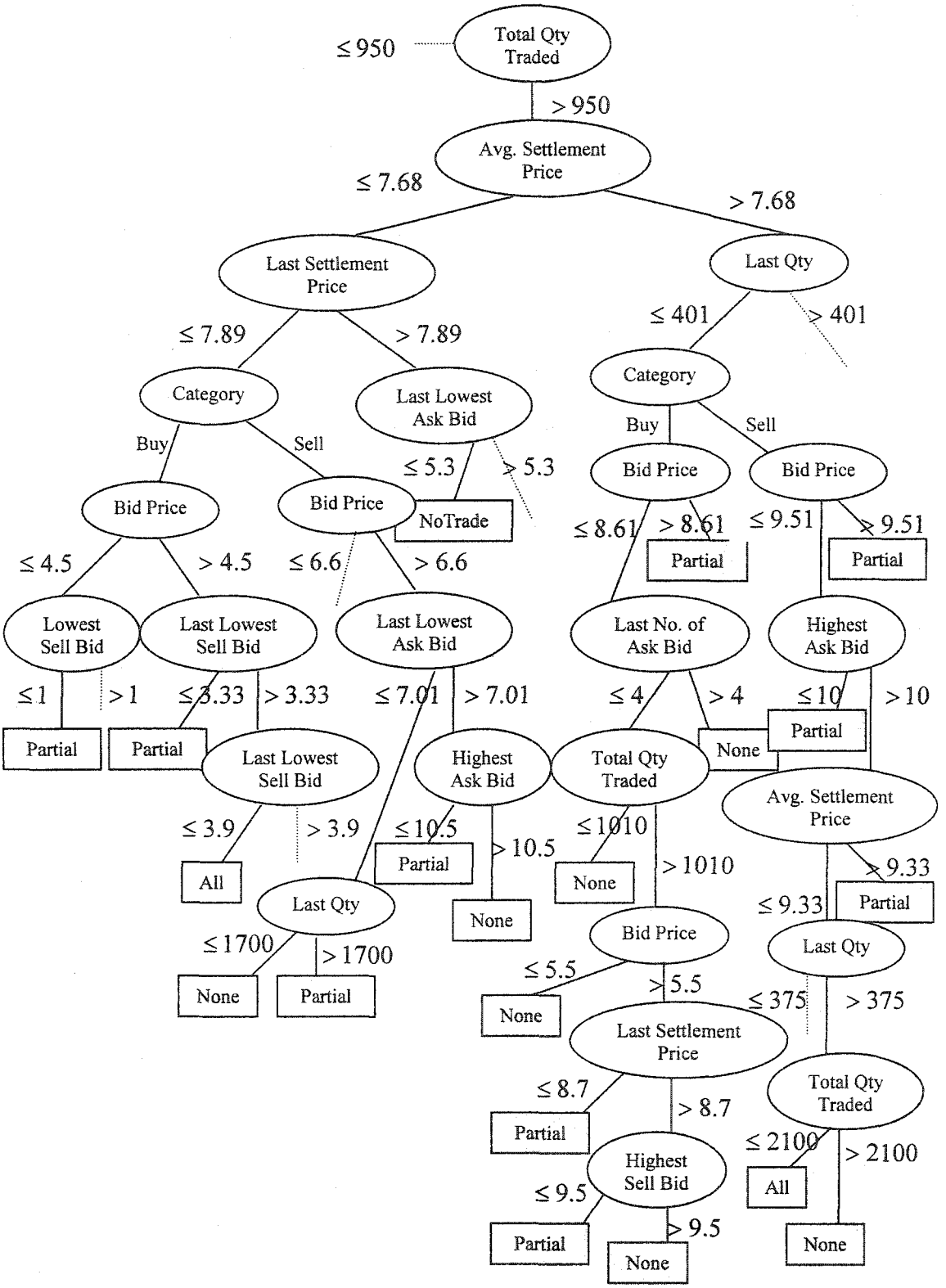


Figure 5.11 Decision tree for auction bid recommendation where the total traded quantity is greater than 950.

The first rule could be interpreted as follows; if the current bid is for a sell auction but the bid price is higher than the highest ask bid price, the auction would have no winner. The second rule is actually straightforward. Participants would not want to sell their products to ones to buy them with cheap price. The other rules can also be interpreted in terms of similar context of situation.

We explore some interesting facts from Figure 5.10. The general pattern of these rules can be very similarly interpreted in that the bid results depend on the relationship between the bid price and historical bid prices. In the situation where the total traded quantity is in a middle range (more than 100 and less than or equal to 950 in Figure 5.10), if the lowest sell bid price is very low and the average settlement price is low, then the bid results in no trade. That implies that auctions with very low lowest bid price and average settlement price can not attract participants so that it does not make any trade. However, at that point, if the average settlement price is a little high, then the buy bid would make partial win. Otherwise the sell bid would result in none. The rules of decision tree in Figure 5.11 can be explained in the same context. These rules reflect transactional history for auction bids and predict results of current bids. Thus, it is believed that the recommender shown previously can provide a dynamic guidance whenever they make bids so that it helps participants win auctions.

5.5 Summary and Discussion

In this chapter, we briefly reviewed discretization, which is needed to evaluate a continuous valued feature or apply some learning algorithms and basic concept of the ReliefF evaluator required for determining a partitioning order in the NP frame.

In terms of accuracy and size of selected features, the numerical results showed that in three data domains there are no significant differences between feature evaluators in the NP-Filter, and when compared with no feature selection, the NP-Filter with the evaluators, accuracies improves or are not worse in most cases and we achieved significant reduction on feature dimension. In terms of accuracy, the information gain evaluator with entropy discretization performs a little better in the discrete data domain. On the other hand, the ReliefF presents better performances in the continuous data domain. In the mixed type data domain, any evaluator did not show conspicuously better results. When it comes to

computation time of the NP-Filter applying the evaluators with three classifiers, it is dependent on the data set itself regardless of any specific data domain.

Further in order to verify that the NP feature selection method can really handle the mixed type of data set, we performed a case study by constructing a recommender system based on decision tree induced by classification and association rules. We show that it can provide auction users with a good decision support tool for auction participation and bids.

Furthermore, we stated that scalability can be achieved by reducing the number of features to fast respond to users' demands, which is a critical issue in an online auction system. Even though it takes time for feature selection process, it would not make a problem since it can be batch-processed in off-line status.

As a case study of the NP feature selection method which can handle mixed type of features, we constructed recommender systems for auction participation and auction bids using classification and association rules with their interpretations. It would be meaningful that this research provided how feature selection and learning algorithm can make contribution on an online auction system. However since the recommendation rules were derived using data gathered from a prototyped system having some limitation, for example, sealed bid double auction system and limited number of rounds, products and users. Thus, it is hard to apply these results to a real auction system by the stated limits.

6 CONCLUSION

We have developed a new scalable optimization based feature selection methods that can be implemented as both a filter and a wrapper. The methods provide significant contributions in that the NP based feature selection algorithm has an optimization framework even presenting a scalable structure and can effectively be used to create learning models that are easily interpreted as a preprocessing step prior applying learning algorithms. The major contributions of this dissertation are as follows:

- Optimization based feature selection

  It is shown that the new approach with an optimization framework can guarantee an optimal solution given a certain distance of the optimum with a given probability after a finite time stopping criterion is satisfied. The numerical results show that the new method performs quite well on several comparison test problems. Through numerical results, we showed why intelligent partitioning is important with respect to accuracy and speed and that using random sampling on instances can be a potential way for handling large number of instances in the NP-Filter. Finally, due to the partitioning scheme of the NP, adding simply new features at the maximum depth is faster than starting over.

- Systematic approaches to scalable feature selection

  First we have showed that the generic NP feature selection algorithm is scalable for the number of instances. To address this, we have presented two systematic ways for scalable feature selection. First, we used an analytical approach to find an optimal solution for the instance sampling rate based on the NP/Rinott using relationships between a variance of performances and the proportion of instance. Second, we developed an adaptive sampling algorithm called Adaptive NP-Filter based on the NP-Filter that the number of instance samples can be dynamically changed. Numerical results on the comparison tests of the three approaches reported that the adaptive sampling approach in the Adaptive NP-Filter is a very

scalable feature selection method with acceptable accuracy level and the NP/Rinott-Filter can be as well a good feature selection method according to some conditions and preferences of experimenters.

- Mixed type of features and recommender systems

  We have briefly reviewed discretization and feature quality evaluators in terms of their ability to deal with continuous variables. In terms of accuracy and size of selected features, we presented numerical results to report the benefits of feature selection when compared with no feature selection and evaluate the NP-Filter with the evaluators on nominal, mixed type, and continuous data domains. In terms of accuracy, the information gain evaluator with entropy discretization performs a little better in the discrete data domain while the ReliefF presents better performances in the continuous data domain. The computation time of the NP-Filter applying the evaluators was dependent on the each data set regardless of any specific data domain. Furthermore, as an application of feature selection we performed a case study by constructing a recommender system based on decision tree induced by classification and association rules. We created rules that can provide auction users with a decision support for auction participation and bids. In addition, we addressed the benefits of feature selection in recommender systems.

All of the research problems considered in this dissertation address important elements in designing scalable feature selection and benefits of feature selection with application to recommender systems under the situation where information we need increases exponentially so that huge amount of data needs to be efficiently handled.

Some of the future research directions include: other than scalability of instance dimension, research on scalability of feature dimension needs to be made in more systematic approach. In the Adaptive NP algorithm we used a step size of instance sampling rate that is decreased in inverse proportion to the depth. However, analytical approach is needed to calculate more intelligent step size. When it comes to recommender systems, rather than

recommendations focused on the accuracy, new research focus should be directed to how to satisfy demands of users in view of human computer interface, that is a problem on recommendation display. Here the key issue is how to create recommendations that users can easily and quickly understand.

APPENDIX A RELATIONSHIP BETWEEN PERFORMANCE VARIANCES AND INSTANCE SAMPLING RATES

The numerical results were reported according to the NP-Filter (Naïve Bayes classifier)'s 5 times run for each data set.

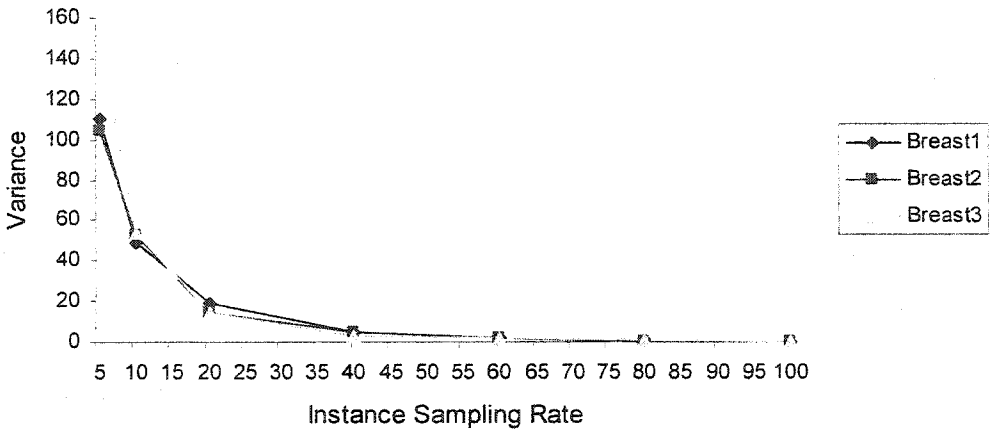


Figure A.1 Performance variances for instance sampling rates of three different modified data sets with 7 features and 1 class feature of 'cancer' data set.

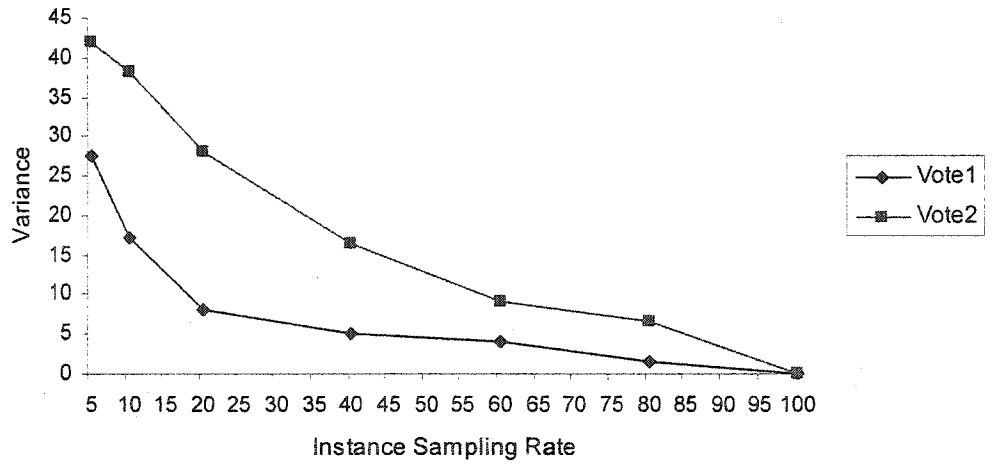


Figure A.2 Performance variances for instance sampling rates of two different modified data sets with 7 features and 1 class feature of 'vote' data set.

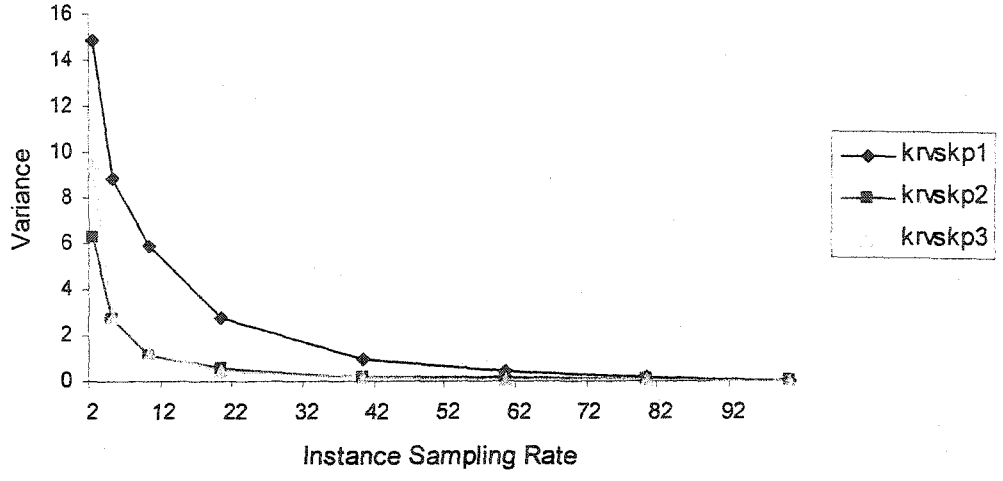


Figure A.3 Performance variances for instance sampling rates of three different modified data sets with 7 features and 1 class feature of 'kr-vs-kp' data set.

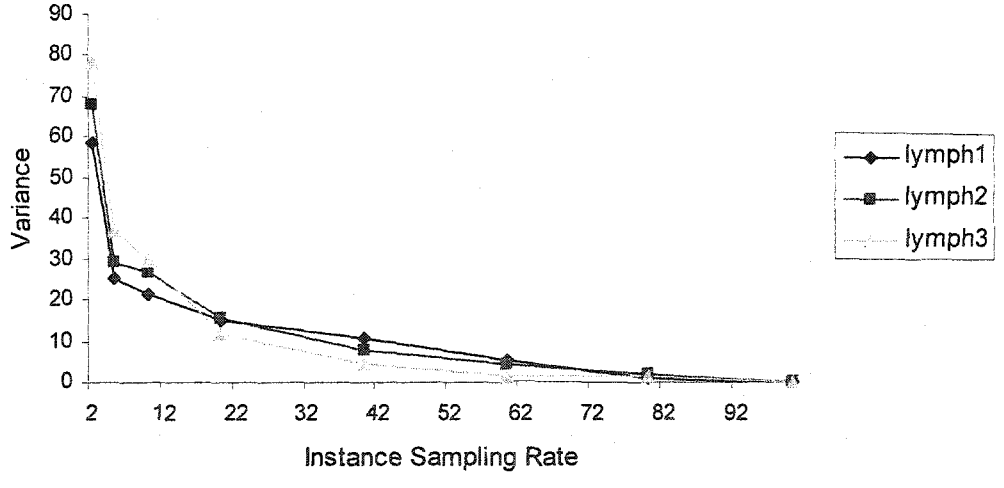


Figure A.4 Performance variances for instance sampling rates of three different modified data sets with 7 features and 1 class feature of 'lymph' data set.

APPENDIX B ADAPTIVE NP-FILTER EVALUATION

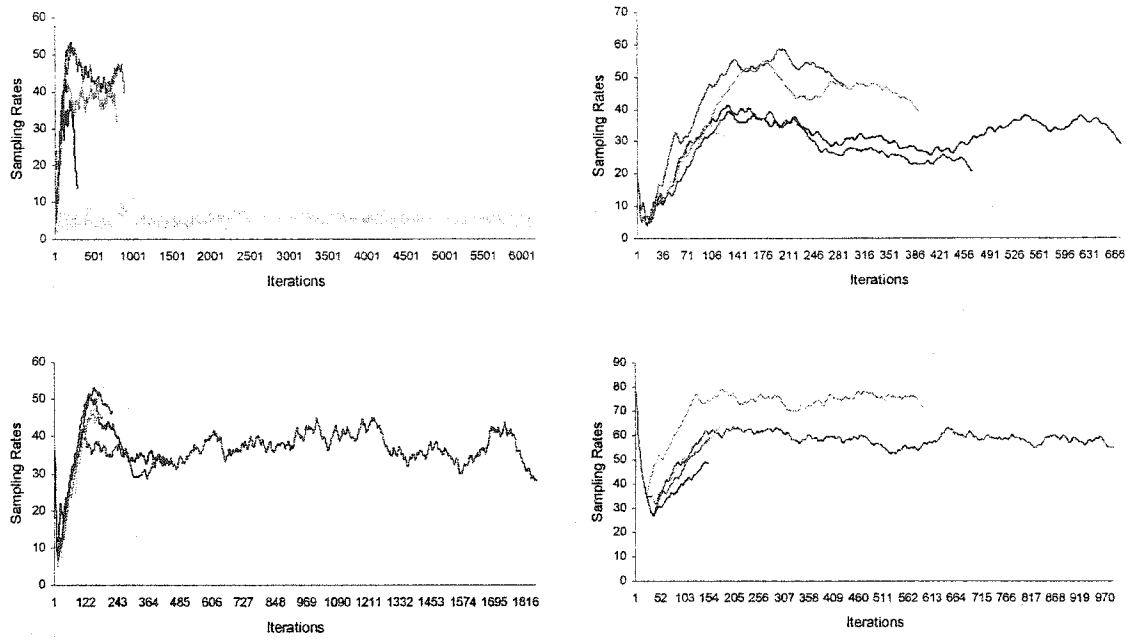


Figure B.1 Moving averages of instance sampling rates of 'audiology' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 3$, and $k = 5$.

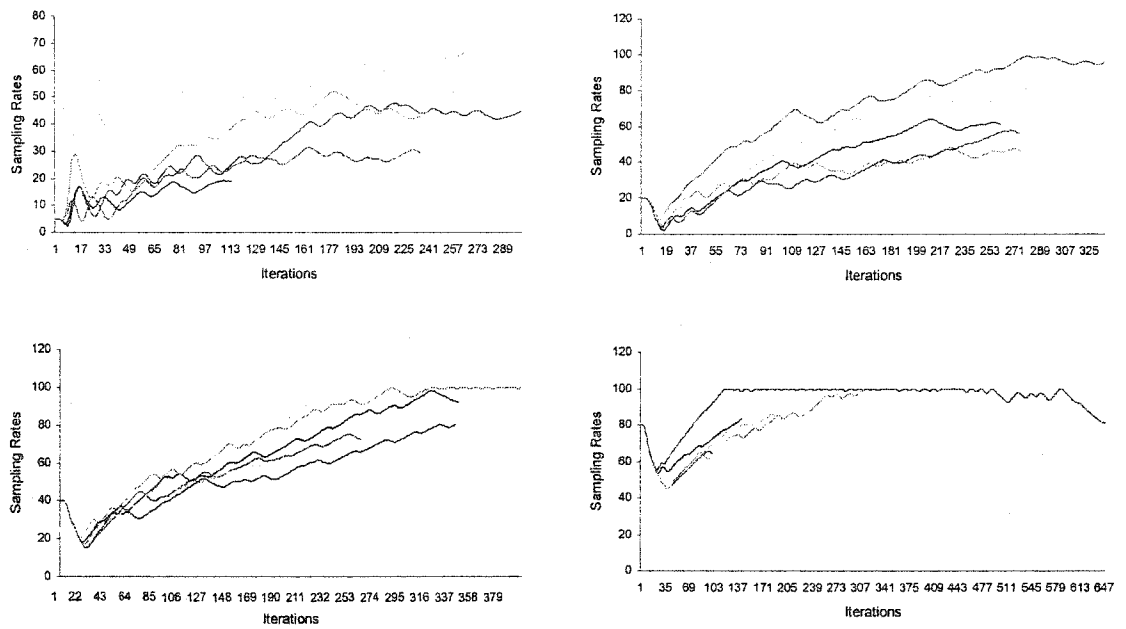


Figure B.2 Moving averages of instance sampling rates of 'audiology' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 7$, and $k = 5$.

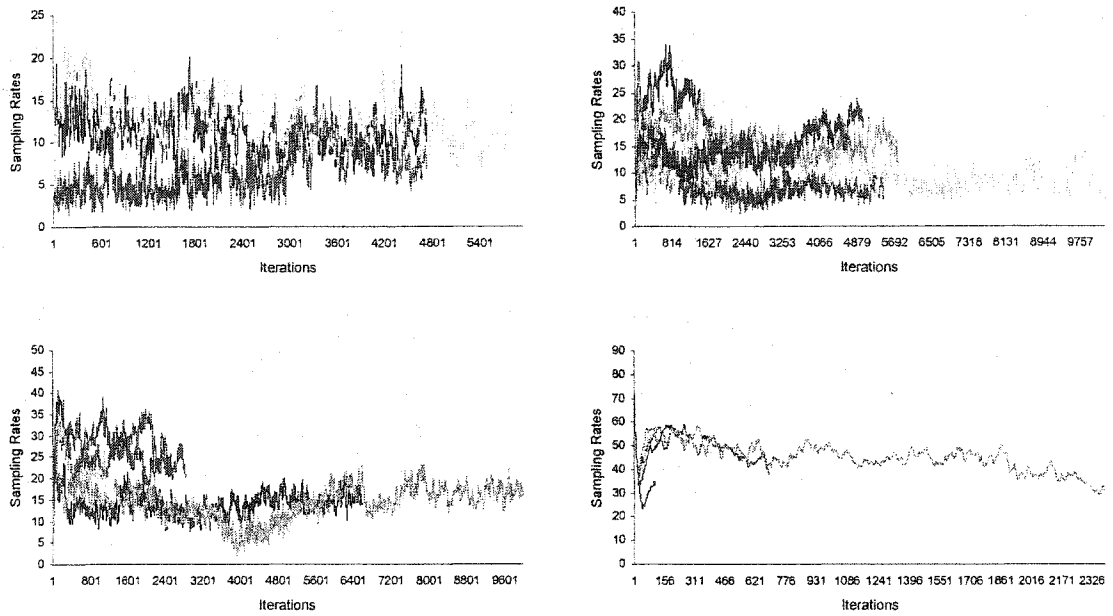


Figure B.3 Moving averages of instance sampling rates of 'audiology' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 3$, and $k = 5$.

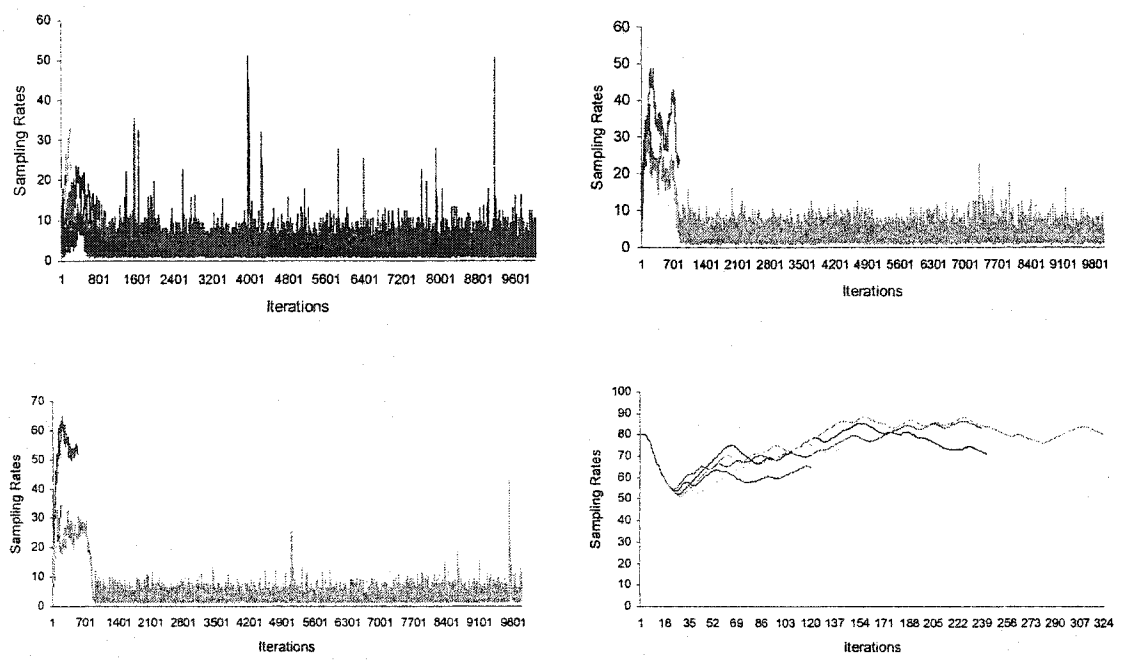


Figure B.4 Moving averages of instance sampling rates of 'audiology' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 7$, and $k = 5$.

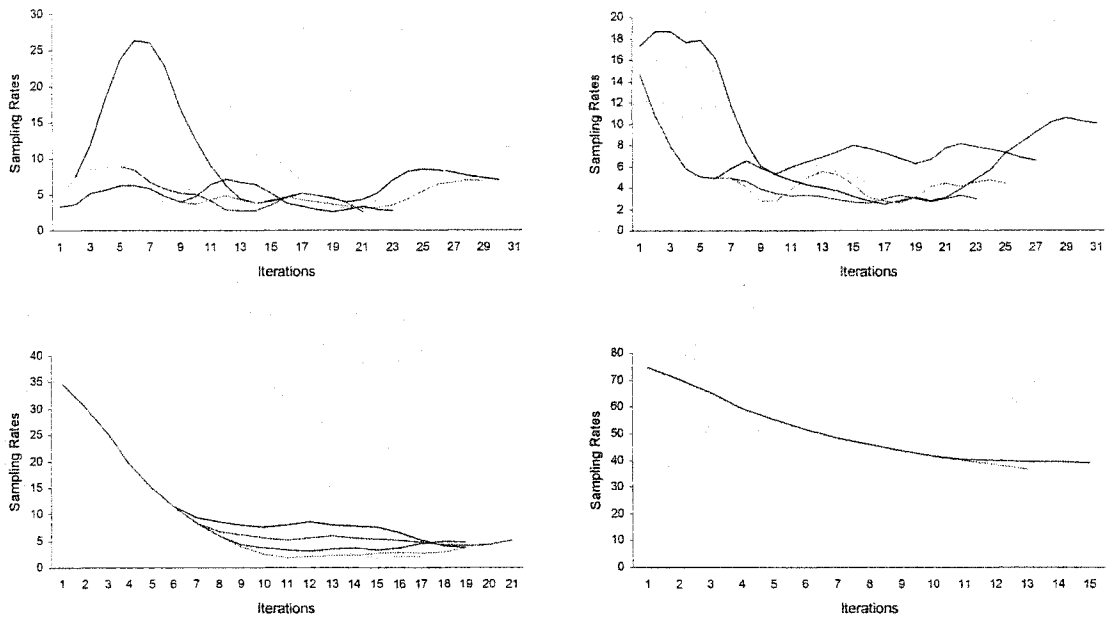


Figure B.5 Moving averages of instance sampling rates of 'vote' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 2$, and $k = 5$.

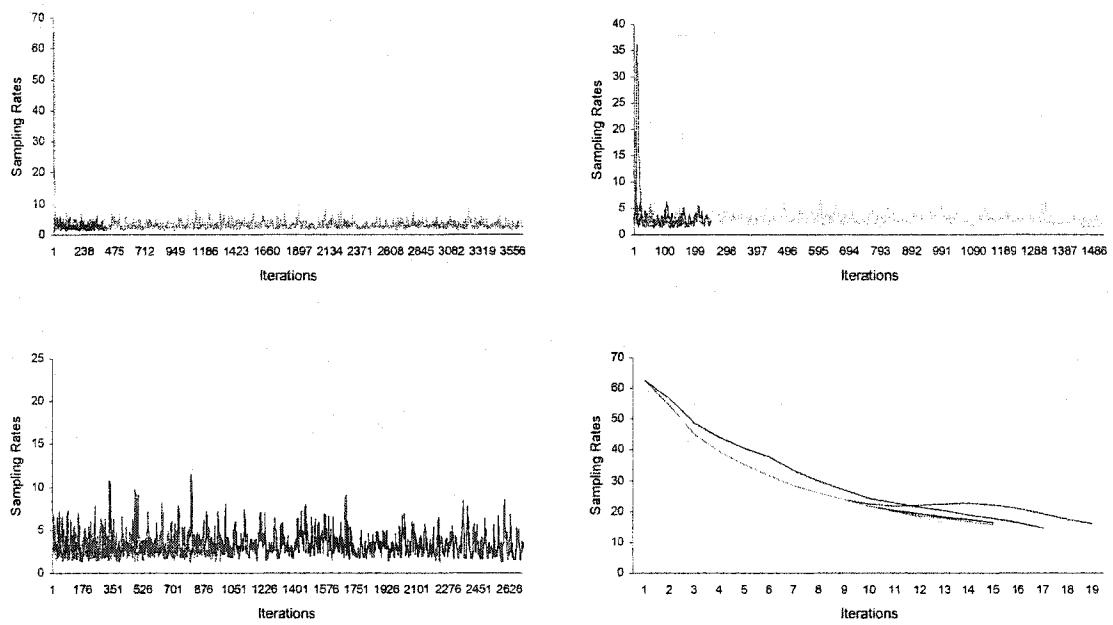


Figure B.6 Moving averages of instance sampling rates of 'vote' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 1$, and $k = 5$.

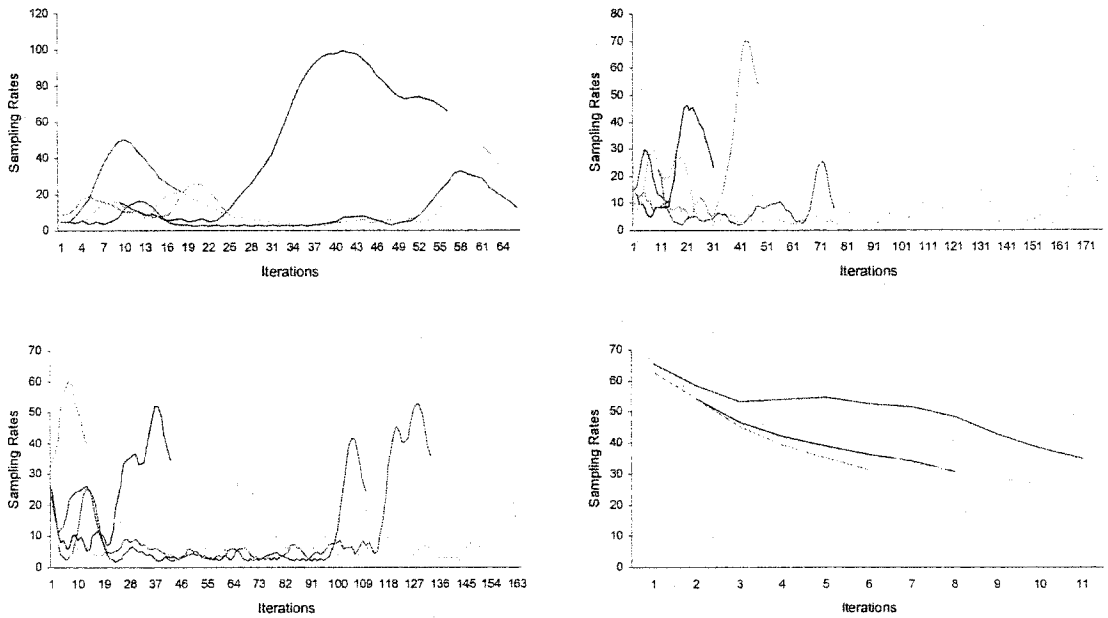


Figure B.7 Moving averages of instance sampling rates of 'cancer' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 1$, and $k = 5$.

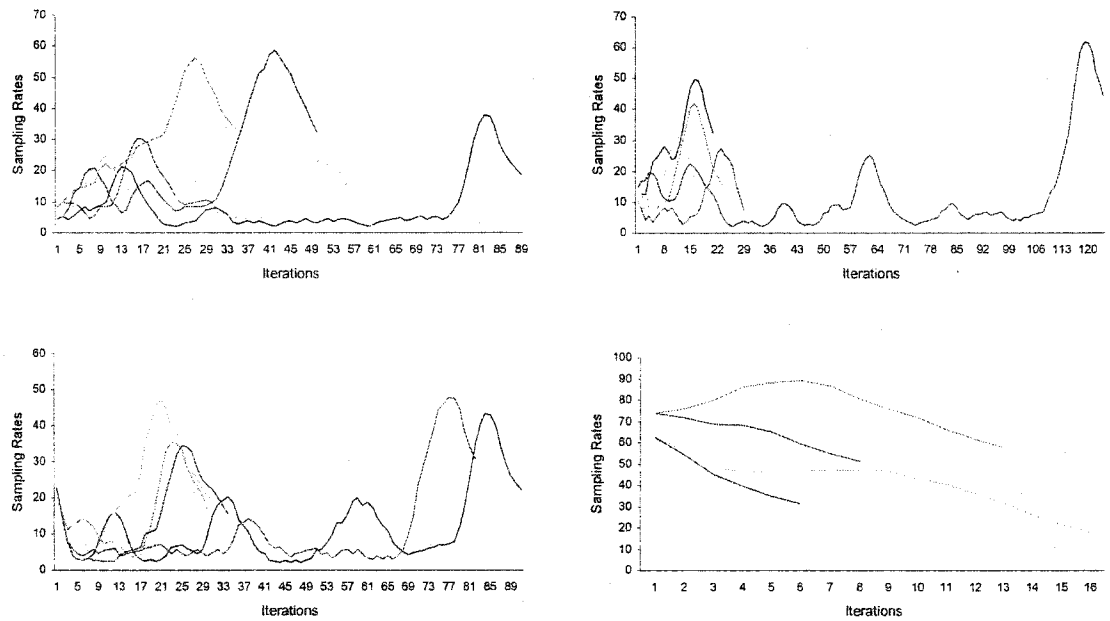


Figure B.8 Moving averages of instance sampling rates of 'cancer' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 1$, and $k = 5$.

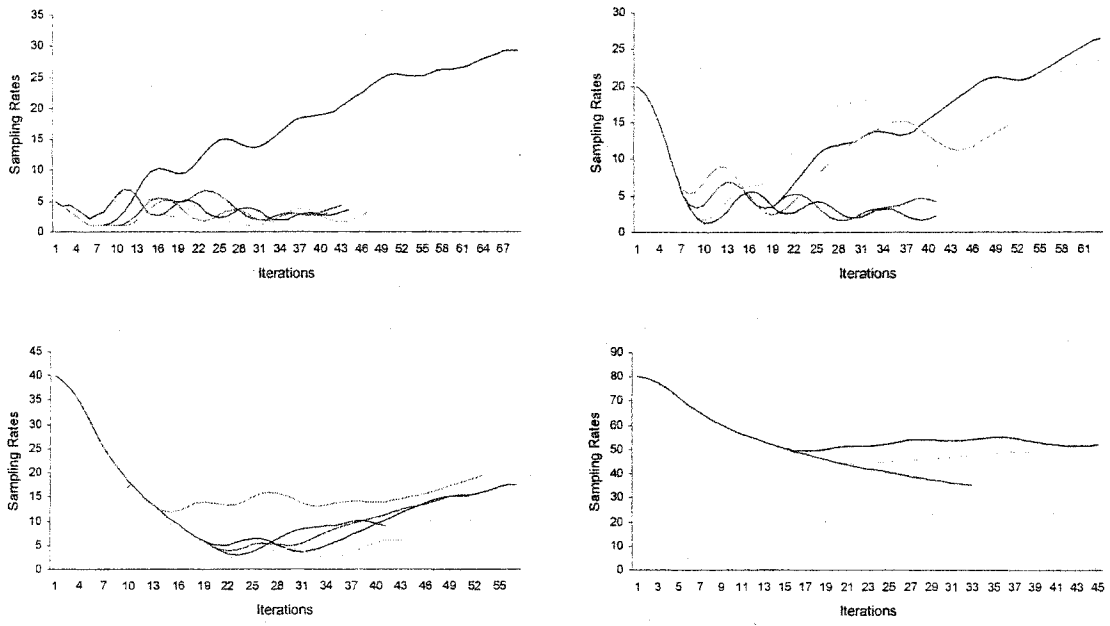


Figure B.9 Moving averages of instance sampling rates of 'kr-vs-kp' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 4$, and $k = 5$.

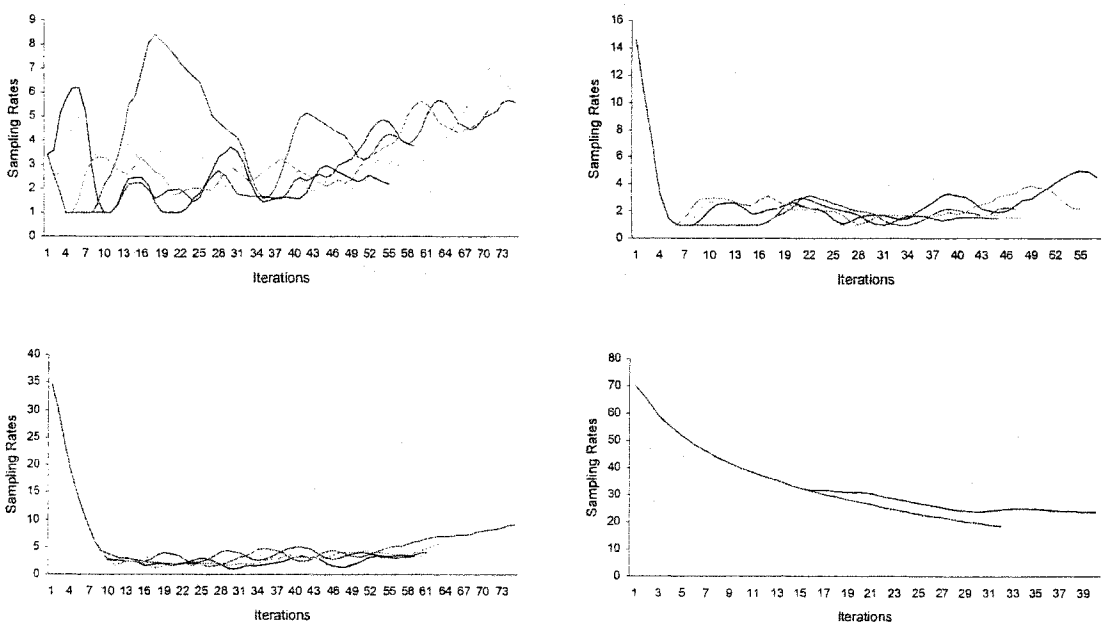


Figure B.10 Moving averages of instance sampling rates of 'kr-vs-kp' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 2$, and $k = 5$.

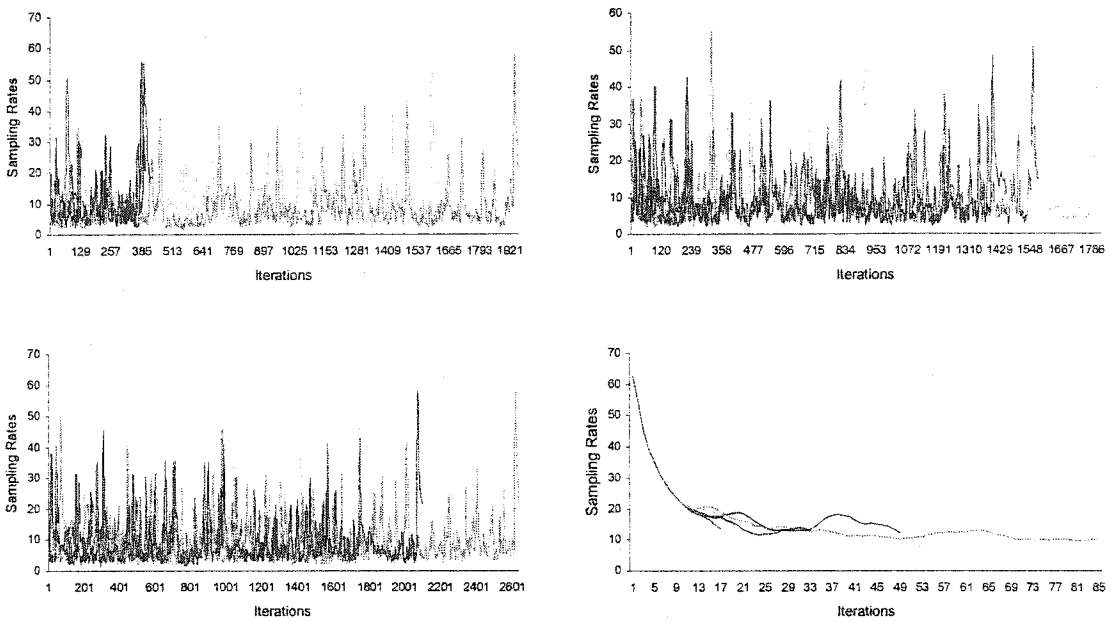


Figure B.11 Moving averages of instance sampling rates of 'lymph' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 1$, and $k = 5$.

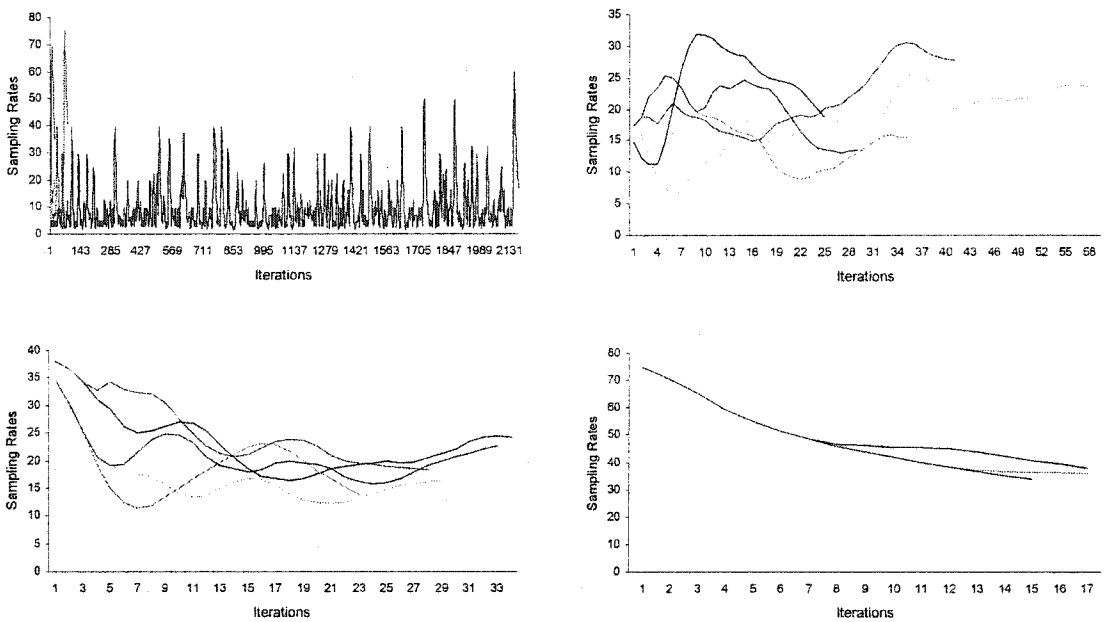


Figure B.12 Moving averages of instance sampling rates of 'lymph' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.0$, last $N = 2$, and $k = 5$.

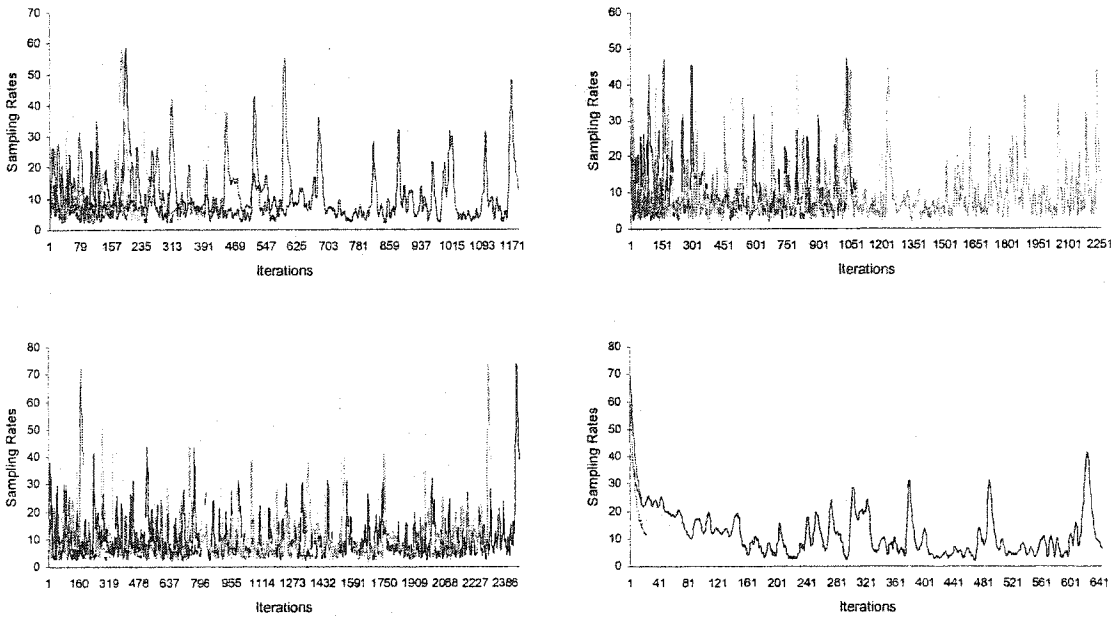


Figure B.13 Moving averages of instance sampling rates of 'lymph' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 1$, and $k = 5$.

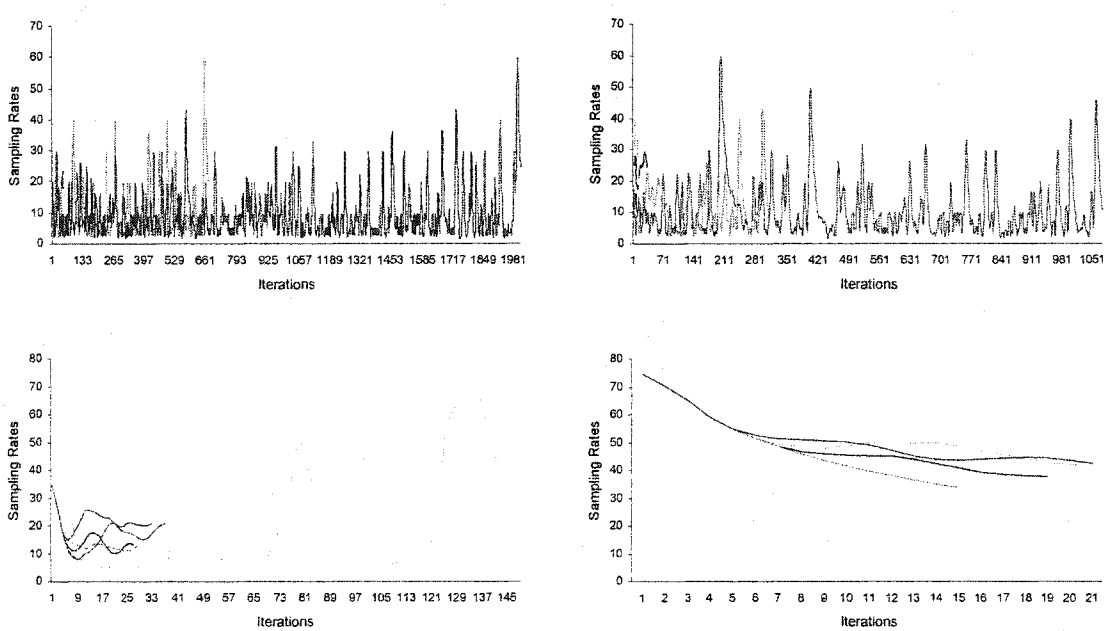


Figure B.14 Moving averages of instance sampling rates of 'lymph' data set with 4 different initial sampling rates, 5, 20, 40, 80, $c = 1.2$, last $N = 2$, and $k = 5$.

Table B.1 Numerical results of Adaptive NP-Filter with $c = 1.0$ and last $N = 10\%$.

Data set	Sampling Rates (%)	Accuracy	Speed	Backtracks	Converged Rates (%)
vote	5	92.4±2.2	700±188	9.8±4.0	9.1±6.1
	10	91.2±1.2	762±90	8.6±2.6	7.0±4.7
	20	92.6±0.8	669±46	6.4±2.1	5.4±2.0
	40	93.0±1.7	859±125	5.8±2.2	9.4±7.3
	80	93.6±0.7	1474±17	0.0±0.0	36.5±0.0
audiology	5	69.9±2.6	53206±18499	138.8±26.5	70.8±13.6
	10	72.6±2.0	44006±33919	89.4±58.2	54.5±6.5
	20	72.7±1.6	49587±31218	98.4±57.7	52.5±26.5
	40	71.0±2.8	75714±20107	123.2±20.8	85.7±10.8
	80	69.6±1.0	229611±392499	434.4±802.3	90.2±10.8
cancer	5	73.0±1.6	779±354	31.4±19.5	25.0±12.0
	10	73.6±0.8	630±292	26.0±21.0	17.5±10.2
	20	73.7±0.6	502±12.7	13.2±7.4	27.1±14.4
	40	73.7±0.4	795±488	34.2±37.9	26.0±13.3
	80	73.6±0.5	524±37	3.8±3.9	29.4±2.1
kr-vs-kp	5	89.9±2.3	7350±2296	5.8±2.9	5.3±3.9
	10	86.8±2.9	13222±15670	10.4±13.8	8.0±9.6
	20	88.5±5.5	9243±2440	6.2±2.2	8.1±4.7
	40	90.7±1.5	13312±806	3.6±1.1	6.2±3.3
	80	84.1±7.2	51772±5016	0.2±0.4	35.9±2.3
lymph	5	83.9±1.1	7754±4051	604.0±341.8	20.1±12.6
	10	84.3±0.7	3391±5420	226.0±484.7	18.5±4.7
	20	83.9±1.8	2996±4202	171.6±360.2	16.0±16.2
	40	84.9±1.4	1091±127	5.0±3.7	20.6±9.5
	80	84.5±1.3	1101±101	1.6±1.7	38.3±4.9

Table B.2 Numerical results of Adaptive NP-Filter with $c = 1.2$ and last $N = 5\%$.

Data set	Sampling Rates (%)	Accuracy	Speed	Backtracks	Converged Rates (%)
vote	5	91.9±1.6	4088±4836	281.4±419.9	7.0±9.7
	10	92.0±1.2	1934±177	111.8±28.6	1.8±0.3
	20	92.4±1.2	3515±2490	222.6±193.3	6.1±10.1
	40	91.8±1.1	2786±1500	161.2±108.6	1.8±0.4
	80	92.3±0.6	1032±68	0.0±0.0	16.5±0.0
audiology	5	71.5±1.6	137866±76940	1356.2±767.9	14.1±7.0
	10	70.4±2.3	79718±47661	755.6±470.5	14.0±4.0
	20	71.3±2.8	201865±137215	1492.8±911.8	17.2±5.1
	40	69.5±1.4	211606±119611	2076.6±1585.6	12.3±4.6
	80	69.8±1.9	80335±54251	274.6±223.2	39.0±9.6
cancer			N/A		
kr-vs-kp	5	87.1±3.2	15583±19838	19.2±18.4	4.2±2.6
	10	88.3±1.2	5225±1035	10.4±2.9	2.4±0.8
	20	86.3±2.8	14207±18690	14.4±13.4	7.8±12.0
	40	90.0±2.6	9439±2266	12.8±3.0	3.9±2.2
	80	85.4±5.7	34906±8611	2.4±5.4	21.1±6.5
lymph	5	83.9±0.9	4050±4653	205.0±284.8	14.9±6.7
	10	84.6±1.2	8911±10493	513.6±641.4	16.0±7.2
	20	84.2±2.2	6367±4015	343.6±239.7	17.6±10.5
	40	84.6±1.2	14629±7557	867.2±508.2	13.4±8.6
	80	84.3±0.9	5776±7818	285.4±485.8	11.2±1.5

REFERENCES

Aha, D.W., and Bankert, R. L., 1996, "A comparative evaluation of sequential feature selection algorithms", In D. Fisher J.-H. Lenz (Eds.), *Artificial Intelligence and Statistics V*, Springer-Verlag, New York.

Almuallim, H., and Dietterich, T., 1994, "Learning boolean concepts in the presence of many irrelevant features", *Artificial Intelligence*, 69 (1-2), pp 279-305.

Basu, C., Hirch, H. and Cohen, W., 1998, "Recommendation as classification: using social and content based information for recommendation", in *Proceedings of the National Conference on Artificial Intelligence.*

Blake, C.L. and Merz, C.J., 1998, *UCI Repository of machine learning databases* <http://www.ics.uci.edu/mlearn/MLRepository.html>, University of California, Irvine, CA (Date Accessed: October 31, 2003).

Bradley, P. S., Fayyad, U.M., and Mangasarian, O.L., 1999, "Mathematical programming for data mining: formulations and challenges", *INFORMS Journal on Computing*, 11

Bradley, P.S., Mangasarian, O.L., and Street, W.N., 1998, "Feature selection via mathematical programming", *INFORMS Journal on Computing*, 10(2), 209-217.

Brassard, G., and Bratley, P., 1996, *Fundamentals of Algorithms*, Prentice Hall, New Jersey.

Breese, J., Heckerman, D. and Kadie, C., 1998, "Empirical Analysis of Predictive Algorithms for collaborative filtering", in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence.*

Caruana, R., and Freitag, D., 1994, "Greedy feature selection", in *Proceedings of the Eleventh International Conference on Machine Learning*}, 28-36. New Brunswick, NJ: Morgan Kaufmann.

Catlett, J., 1991, "Megainduction: A test flight", in *Proceedings of the Eighth International Workshop on Machine Learning*, pp 596-599. Morgan Kaufmann.

Dash, M., and Liu, H., 1997, "Feature selection for classification", *Intelligent Data Analysis*, 1(3).

Doak, J., 1992, "An evaluation of feature selection methods and their applications to computer security", *Technical Report CSE 92-18*, Davis, CA University of California, Department of Computer Science.

Domingo, C. Gavalda R., and Watanabe, R., 2000, "Adaptive sampling methods for scaling up knowledge discovery algorithms", *Journal of Knowledge Discovery and Data Mining*.

Fayyad, U. M. and Irani, K. B., 1993, "Multi-interval discretization of continuous features as preprocessing for machine learning" in *Proceedings of 13th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, pp. 1022-1027.

Ginsberg, M. L., 1993, *Essentials of artificial intelligence*, Morgan Kaufmann.

Good, N., Shafer, J. B., Konstan, J. A., Borchers, A. and Sarwar, B., 1999, "Combining collaborative filtering with personal agents for better recommendations", in *Proceedings of the National Conference on Artificial Intelligence*.

Guttman, R., Maes, P. and Moukas, A., 1998, "Agent-mediated electronic commerce: a survey", *Knowledge Engineering Review*, 13 (2), pp 143-152.

John, G., Kohavi, R., and Pfleger, K., 1994, "Irrelevant features and the subset selection problem", in *Proceedings of the Eleventh International Conference on Machine Learning*, pp 121-129, New Brunswick, NJ. Morgan Kaufmann.

John, G. and Langley, P., 1996, "Static versus dynamic sampling for data mining", in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 367-370.

Hall, M.A., 1998, "Correlation-based feature selection for discrete and numeric class machine learning", in *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford University, CA. Morgan Kaufmann.

Hill, W., Stead, L., Rosenstein, M., and Furnas, G., 1995, "Recommending and Evaluating Choices in a Virtual Community of Use", In *Proceedings of CHI '95.*

Karypis, G., 2000, "Evaluation of item-based top-N recommendation algorithms", *Technical Report #00-046*, Department of Computer Science, University of Minnesota.

Kerber, R., 1992, "Chimerge: discretization of numeric features", in *Proceedings of National Conference on Artificial Intelligence*, MIT Press, pp. 123-128.

Kim, Y.S., Street, W.N, and Menczer, F., 2000, "Feature selection in unsupervised learning via evolutionary search", in *Proceedings of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*

Kira, K. and Rendell, L., 1992, "A practical approach to feature selection", in *Proceedings of International Conference on Machine Learning, ICML-92*, pp. 249-256.

Kivinen, J. and Mannila, H., 1994, "The power of sampling in knowledge discovery", in *ACM Symposium on Principles of Database Theory*, 77-85.

Kohavi, R., 1994, "Feature subset selection as search with probabilistic estimates", in *AAAI Fall Symposium on Relevance.*

Kohavi, R. and Frasca, B., 1994, "Useful feature subsets and rough set reducts", in *Third International Workshop on Rough Sets and Soft Computing.*

Kohavi, R. and John, G. H., 1997, "Wrappers for feature subset selection", *Artificial Intelligence*, Vol. 97, No. 1-2, pp 273-324.

Koller, D. and Sahami, M., 1996, "Toward optimal feature selection", in Machine Learning: Proceedings of the Thirteenth International Conference. Morgan Kaufmann.

Koller, D. and Sahami, M., 1997, "Hierarchically classifying documents using very few words", in Proceedings of the International Conference on Machine Learning, pp 170-178.

Kononenko, I., 1994, "Estimating features: analysis and extensions of RELIEF", in *Proc. European Conf. on Machine Learning* (Catania, April 1994).

Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L. and Riedl, J., 1997, "GroupLens: applying collaborative filtering to usenet news", *Communications of the ACM, 40(3)*, pp. 77-87.

Kurgan, L., and Cios, K. J., 2001, "Discretization algorithm that uses class-feature interdependence maximization", in *Proceedings of the 2001 International Conference on Artificial Intelligence (IC-AI 2001)*, pp. 980-987, Las Vegas, Nevada.

Lin, W., Alverez, S. A. and Ruiz, C., 2000, "Collaborative recommendation via adaptive association rule mining", in *Proceedings of ACM WEBKDD 2000.*

Liu, H. and Motoda, H., 1998, *Feature Extraction, Construction and Selection: A Data Mining Perspective*, Kluwer Academic Publishers.

Liu, H., and Setiono, R., 1996a, "Feature selection and classification - a probabilistic wrapper approach", in *Proceedings of the 9th International Conference on Industrial and Engineering Applications of AI and ES.*

Liu, H., and Setiono, R., 1996b, "A probabilistic approach to feature selection - a filter solution", in *Proceedings of the Thirteenth International Conference on Machine Learning*, Morgan Kaufmann.

Liu, H., and Setiono, R., 1998, "Some issues on scalable feature selection", *Expert Systems with Application*, 15, 333-339.

Mitchell, T., 1997, *Machine Learning*, McGraw Hill.

Modrzejewski, M., 1993, "Feature selection using rough sets theory" in P.B. Brazdil (ed.), *Proceedings of the European Conference on Machine Learning*, 213-226.

Narendra, P.M., and Fukunaga, K., 1977, "A branch and bound algorithm for feature subset selection", *IEEE Transactions on Computers*, 26(9), 917-922.

Ólafsson, S., 2003, "Two-stage nested partitions method for stochastic optimization," *Methodology and Computing in Applied Probability*, to appear.

Olafsson, S. and Yang, J., 2003, "Intelligent partitioning for feature selection", *INFORMS Journal on Computing*, (accepted).

Olafsson, S. and Yang, J., 2001, "Scalable optimization-based feature selection", *Proceedings of Workshop on Discrete Mathematics and Data Mining* (2nd SIAM Conference on Data Mining), Arlington, VA.

Provost, F., Jensen, D. and Oates, T., 1999, "Efficient progressive sampling", in *Proceedings of the fifth International Conference on Knowledge Discovery and Data Mining*, 23-32.

Provost, F. and Kolluri, V., 1999, "A survey of methods for scaling up inductive algorithms", *Data Mining and Knowledge Discovery 3*: 131-169.

Quinlan, J. R., 1986, "Induction of decision trees", *Machine Learning*, 1.

Quinlan, J. R., 1993, *C4.5: Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann.

Raman, B. and Ioerger, T. R., 2002, "Instance based filter for feature selection", *Journal of Machine Learning Research* 1-23.

Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., and Rield, J., 1994, "GroupLens: an open architecture for collaborative filtering of netnews". in *Proceedings of CSCW '94*, Chapel Hill, NC.

Richeldi, M., and Rossotto, M., 1995, "Class-driven statistical discretization of continuous features (extended abstract)", in *N. Navrac and S. Wrobel*, eds., *Machine Learning: ECML-95 (Proc. European Conference on Machine Learning, 1995)*.

Rinott, Y., 1978, "On two-stage selection procedures and related probability-in-equalities", *Communications in Statistics*, A7, 799-811.

Ryan, S.M., Min, K.J. and Olafsson, S., 2001, "Experimental study of scalability enhancements for reverse logistics E-Commerce", in Prabu, Kumara and Kamath (eds.) *Scalable Enterprise System – An Introduction to Recent Advances*.

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2000), "Application of dimensionality reduction in recommender system- A Case Study", in *ACM WebKDD 2000 Workshop*.

Shapire, R. E., 1990, "The strength of weak learnability", *Machine Learning*, Vol. 5, pp 97-227.

Shardanand, U., and Maes, P., 1995, "Social information filtering: algorithms for automating 'Word of Mouth'", in *Proceedings of CHI '95*. Denver, CO.

Shi, L. and Olafsson, S., 2000, "Nested partitions method for global optimization", *Operations Research*, 48, 390-407.

Skalak, D., 1994, "Prototype and feature selection by sampling and random mutation hill-climbing algorithms", in *Proceedings of the Eleventh International Conference on Machine Learning*, pp 293-301, New Brunswick, NJ. Morgan Kaufmann.

Terveen, L. and Hill, W., 2001, "Beyond recommender systems: helping people help each other", in J. M. Carroll (Ed.), *Human Computer Interaction in the New Millennium*, ACM Press, New York, pp 487-509.

Toivonen, H., 1996, "Sampling large databases for association rules", in *Proceedings of the 22nd International Conference on Very Large Databases*, 134-145.

Vucetic, S., and Obradovic, Z., 2000, "A Regression based approach for scaling-up personalized recommender systems in e-Commerce", in *Proceedings of ACM WEBKDD 2000*.

Weiss, G. M. and Provost, F., 2001, "The effect of class distribution on classifier learning: an empirical study", Technical Report ML-TR-44, Department of Computer Science, Rutgers University August 2, 2001.

Yang, J. and Honavar, V., 1998, "Feature subset selection using a genetic algorithm" In H. Motada and H. Liu (eds), *Feature Selection, Construction, and Subset Selection: A Data Mining Perspective*, Kluwer, New York.