Instance selection for model-based classifiers

by

Walter Dean Bennette

A thesis 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
Jo Min
Lizhi Wang
Dianne Cook
Heike Hofmann

Iowa State University

Ames, Iowa

2014

TABLE OF CONTENTS

ABSTRACT

Aspects of a classifier's training dataset can often make building a helpful and high accuracy classifier difficult. Instance selection addresses some of the issues in a dataset by selecting a subset of the data in such a way that learning from the reduced dataset leads to a better classifier. This work introduces an integer programming formulation of instance selection that relies on column generation techniques to obtain a good solution to the problem. Experimental results show that instance selection improves the usefulness of some classifiers by optimizing the training data so that that the training dataset has easier to learn boundaries between class values. Also included in this paper are two case studies from the Surveillance, Epidemiology, and End Results (SEER) database that further confirm the benefit of instance selection. Overall, results indicate that performing instance selection for a classifier is a competitive classification approach. However, it should be noted that instance selection might overfit classifiers that have already achieved a good fit to the dataset.

CHAPTER 1. INTRODUCTION

This chapter will introduce the concept of instance selection for classifier training along with its benefit and the objectives of this dissertation. At the end, an overview of the dissertation's contents is presented.

1.1 Motivation

The world is inundated with information and it is advantageous to extract as much knowledge from that information as possible. Some practices that help accomplish this task are grouped into the area of data mining, and one particularly helpful practice is known as classification.

In data mining, information is arranged into a collection of data points called instances. Each instance can describe a particular object or situation and is defined by a set of independent variables called attributes. For classification problems there is a target concept we wish to learn about and this concept is represented in each instance as a dependent variable having finitely many values called the class. The goal of classification is therefore to construct a model from instances with known class values, called the training dataset, to predict which class an instance should belong to based on its observed attribute values. In some cases the induced classification model, or classifier, is simply used to make class predictions for instances with unknown class values, while in other cases the classifier is analyzed to learn more about the target concept. The quality of the learned model is typically assessed by calculating the number of correct predictions made by the classifier when predicting the class values of a withheld set of instances called the testing dataset.

The goal of this research is to provide a method to create better classification models, or as that often implies, classification models with higher testing accuracy. Better classification models will be obtained by optimizing the training data in such a way that the training dataset has easier to learn boundaries between class values. The motivation for this is that numerous aspects of the training data can make it difficult for a classification learner to induce an accurate model. For example, two class values may completely overlap, one or more classes may have outliers, or minority class values may exist that can be ignored by the learner. A relatively recent approach for addressing such issues is to use what has been termed instance selection to create a subset of the training dataset in such a way that all or some classification learning algorithms will perform better when applied to the new and reduced training dataset.

Clearly, instance selection may be thought of as a combinatorial optimization problem where each instance is either included or not, that is, a problem with a binary decision variable, resulting in a search space of 2^n possible subsets of n instances. As n may be very large, searching through this space without structure is very difficult. Furthermore, the ultimate objective function is to maximize the test accuracy of the classifier induced on the selected instances; that is, the objective function has no closed form and can only be evaluated on an independent test dataset after a classification model has been learned. One can use a proxy of the test accuracy, e.g. the accuracy on the training data, but even then the objective function has no closed form and is difficult to evaluate. Perhaps due to those reasons past work on instance selection has focused heavily on the use of metaheuristics, primarily evolutionary algorithm, that search the space of 2^n possible instance subsets directly without attempting to place any structure on the search space or utilize any optimization theory. Thus, while numerous connections between optimization and classification have been thoroughly studied in the literature (Olafsson et al., 2008; Bradley et al., 1999), it is contended that the understanding of instance selection as an optimization problem is still in its infancy.

1.2 Research objectives

The premise of this dissertation is that the boundaries between instances with different class values in the training data can be optimized for the induction of a classifier through an integer programming (IP) formulation of instance selection. This draws on optimization theory to create a search space (formulation) that can be searched to find a better set of the training data; that is, training data with boundaries more suited to the chosen classifier, which subsequently leads to a better (high- accuracy) classification model. The overall objective of this dissertation is thus to develop optimization theory and methods for instance selection, focusing primarily on an integer programming formulation of instance selection.

A naïve integer programming formulation of instance selection is very straightforward. The decision variable could be defined as,

$$x_i = \begin{cases} 1 & \text{if instance } i \text{ is included in the final training data} \\ 0 & \text{otherwise.} \end{cases}$$

The objective function could then be to maximize the accuracy of some model (e.g., a decision tree) learned on the selected instances, and no further constraints are required. However, this objective function has no closed form and the feasible region has no structure that can be exploited to help search through this space of 2^n possible solutions (where n is the number of instances).

The contention of this dissertation is that instead of naively solving this problem using some heuristic, the formulation should be rethought to make it more solvable. One of the key novel ideas of this dissertation is thus to construct subsets of instances that appear to perform well together, that is, instances that when considered together lead to a classifier with high accuracy. A column can then be defined as $(a_{1j} \quad \cdots \quad a_{nj})^T$ where $a_{ij} = 1$ if instance i is included in the jth column (subset), and $a_{ij} = 0$ otherwise. The decision variable of the integer programming problem is then to decide which columns' instances should be used in the final training data, that is,

$$x_j = \begin{cases} 1 & \text{if column } j's \text{ instances are included in the final training data} \\ 0 & \text{otherwise.} \end{cases}$$

Additional constraints may now be relevant. For example, let J denote the set of all columns, and formulate the instance selection problem as

$$Max \qquad Z = Accuracy \qquad (1)$$

$$s.t \qquad \sum_{j \,\epsilon\, J} a_{ij} x_j \leq 1, \;\; \forall\, i \,\epsilon\, I \qquad (2)$$

$$x_j \,\epsilon\, \{0,1\}\,, \forall j \,\epsilon\, J. \qquad (3)$$

The objective is still to maximize classifier accuracy and the constraint $\sum_{j \in J} a_{ij} x_j \leq 1$ ensures that each instance is selected for the final training data at most once. One may recognize this constraint as a set packing constraint, as the set packing problem is a well-studied IP problem. However, for reasons outlined below, the pure set packing formulation may not be the best formulation of the instance selection problem.

The key to a good formulation of the instance selection problem is the ability to generate good columns, that is, to construct subsets of instances that work well together. This leads to the first explicit research question addressed in this dissertation.

<u>Research Question 1</u>: When formulating a set-packing-type IP for instance selection, how should a good initial set of columns be constructed?

Regardless of the specifics of the instance selection formulation, a key issue is to determine a set of columns (subsets) to be used in the IP. Motivated by similar approaches that have been found effective for both set covering and partitioning problems, the process is initiated by generating good initial columns J', which can then be improved (see Research Question 2 below).

<u>Research Question 2</u>: Given a set of initial columns and an IP formulation of instance selection, how can column generation be implemented to find improving columns?

In traditional column generation a linear program (LP) with a prohibitively large number of decision variables (or columns) is solved to optimality without having to consider or construct all of its possible columns. This is accomplished by considering a relaxation of the original LP that contains only a subset of the possible columns and generating any new columns needed to improve the relaxation's solution. Through an iterative process eventually no improving columns can be generated and it is known that the optimal solution to the original LP is found.

Column generation is an ideal technique to construct good columns for instance selection because of the prohibitively large number of possible columns. Even though column generation is designed specifically for solving LPs it is frequently used in integer programs by relaxing the integrality constraint of the decision variables, and this is no different when considering instance selection. However, unlike traditional column generation, instance selection requires estimation in order to find improving columns. This is because there is no closed form function to calculate the accuracy of a classifier based solely on the contents of its training data.

Therefore, to effectively implement column generation with the purpose of finding improving columns for an IP formulation of instance selection, an estimation of classifier accuracy based on the content's of its training data is required.

Research Question 3: When formulating instance selection using IP, what constraints should be placed on the selection of columns in addition to the set packing constraints?

The formulation of instance selection in equations (1) through (3) is only one of the many instance selection IPs that could be devised, and it is not without its own shortcomings. For example, adding a constraint $\sum_{j \in J} x_j \leq 1$ to (2) and (3) insures that a simple linear objective function can be defined for the instance selection integer program. While this is a plausible approach and preliminary results indicate that column generation is able to discover useful columns based on a relaxation of this IP, it also somewhat restricted. In fact, it is known that allowing more than one column can be advantageous as preliminary results also show that a greedy search can be used to select amongst the columns to combine them in a useful way. Two IP formulations of instance selection with different goals in regard to column generation are discussed in Chapter 3.

1.3 Organization of dissertation

Chapter 2 of this dissertation presents a review of the relevant literature related to data mining, instance selection, and column generation. Chapter 3 follows with two formulations of instance selection as an integer program, as well as considerations for column generation. Chapter 4 introduces results from 12 experimental datasets along with an analysis of instance selection. In Chapter 5, two case studies concerning the Surveillance, Epidemiology, and End Results (SEER) database and instance selection are discussed. Finally, conclusions and suggestions for future work are made in Chapter 6.

CHAPTER 2. LITERATURE REVIEW

Chapter 2 will introduce data mining, instance selection, and column generation. This will provide the reader with sufficient background information to follow the concepts of the dissertation and to show the current gap in the instance selection literature.

2.1 Data mining

Data mining has been found to be increasingly useful in many application areas (Han and Kamber, 2003; Witten and Frank, 2005). This can partially be contributed to increased prevalence of massive databases, which often contain a wealth of data that traditional methods of analysis fail to transform into relevant knowledge. Specifically, meaningful patterns are often hidden and unexpected, which implies that they may not be uncovered by hypothesis-driven methods. In such cases, inductive data mining methods, which learn directly from the data without an a priori hypothesis, can be used to uncover hidden patterns that can then be transformed into actionable knowledge.

All data mining starts with a set of data called the training set, which consists of instances describing the observed values of certain variables, or attributes. These instances are then used to learn a given target concept or pattern. One such approach is classification. In classification the training data is labeled, meaning that each instance is identified as belonging to one of two or more classes, and an inductive learning algorithm is used to create a model that discriminates between those class values. The model can then be used to classify any new instances according to this class variable. The primary objective is usually for the classification to be as accurate as possible.

Five useful and common classification algorithms are k-Nearest Neighbors (*k*-NN), naïve Bayes, decision tree, logistic regression, and support vector machines. Each of these classification algorithms, with the exception of *k*-NN, operates by inducing some model from the training data that can then be used to predict the class of unlabeled instances. However, each of these classifiers has an intrinsic bias that stems from assumptions made by the algorithm during its construction. This bias will cause the classifier to perform better for classification problems whose instances exhibit certain relationships amongst their attributes and true class labels. Although no classification problem may perfectly fit the assumptions made by any specific classification algorithm, it has been found that classifiers are still able to take advantage of some of the relationships and structure in the data and make good classifications. The five classifiers mentioned above will now be discussed along with their inherit biases.

K-Nearest Neighbors is an instance-based classifier. Meaning, the classification of an unlabeled instance is not based on any abstraction of the training data, but rather, is based on the training instances themselves. With *k*-NN an unlabeled instance is classified as the most prevalent class observed in the nearest *k* training instances. A positive of the *k*-NN classifier is that it is very simplistic, but this simplicity comes at the sacrifice of having to store large amounts of data and perform a large number of calculations every time that an instance needs to be classified. The bias of *k*-NN comes from the assumption that an unlabeled instance should be classified the same as its nearest neighbors. This assumption means *k*-NN will perform better for classification problems where instances belonging to different classes do not overlap and are separated by a large distance in Euclidian space.

Naïve Bayes is a statistical classifier that predicts the probability that a given instance belongs to a particular class. It is an attractive classifier because it requires only simple calculations and has been shown to have good accuracy for a wide variety of classification problems. The probability prediction of naïve Bayes operates under the assumption that the "effect of an attribute value on a given class is independent of the values of the other attributes" (Han and Kamber, 2003). This assumption allows the required calculations of the classifier to be simple, but also introduces a bias to the classifier. Specifically, the naïve Bayes classifier is biased in a way that it performs better when an instance's attribute values affect its class outcome with higher levels of independence.

Decision trees are another popular and relatively simple technique for classification. Decision tree algorithms induce a tree in a top-down manner by selecting attributes one at a time and splitting the training instances into groups according to the values of their attributes. The most important attribute is selected as the top split node, the next most important attribute is considered at the next level, and so forth. For example, in the popular C4.5 algorithm (Quinlan, 1992), attributes are chosen to maximize the information gain ratio in the split. This is an entropy measure designed to increase the average class purity of the resulting subsets of instances as a result of the sequential splits. A bias intrinsic to decision trees is of course that they perform better when an instance's attributes actually have a hierarchical structure in regard to determining class label.

Logistic regression is a classification technique closely related to linear regression. However, unlike linear regression the outcome is not a predicted numerical value, but instead is a probability prediction that an instance belongs to a specific class. This prediction is achieved by creating a linear function of the dataset's attributes. Again, as with the other techniques that create a classification scheme from the data, a bias is introduced. This time the bias is that logistic regression works better when an instance's class is actually determined by some linear function of its attribute values.

A support vector machine is a classification technique that identifies instances (or support vectors) that help define the optimal border between classes. Support vectors are typically found by transforming the original data with some non-linear mapping to a high dimension space and then implementing an optimization problem to find an optimal hyper plane that separates the instances into different classes. Support vector machines have been found to be very useful in practice as they can achieve high accuracy while avoiding overfitting. Still, support vector machines have a bias and the challenge is to find an appropriate kernel that maps the instances to a space in which they are truly separable.

Because classification algorithms have biases and because natural classification problems do not perfectly fit this bias, classifiers are quite frequently less than perfect. To combat these imperfections many researches have suggested improvement techniques. In fact, instance selection is one such class of techniques and will be discussed in great detail later. Another class of improvement techniques is ensemble classifiers. Ensemble classifiers make a prediction for an unlabeled instance by consulting multiple classifiers and using a voting scheme to make a final determination. The idea behind ensemble classifiers is that a deficiency in one classifier may be compensated for in the other classifiers, and that the majority decision has greater potential to be correct. Two ensemble classification techniques are AdaBoost and random forests.

AdaBoost is an ensemble classifier that builds sequential classifiers from a training dataset. At the beginning, a classifier is built from the training dataset with all of the instances having an equal weight. This classifier is then used to classify the original training data. Any instances that are misclassified by the classifier have their weight increased so that the classifier built next will give them more attention. This process of building a classifier and adjusting instance weights continues until a sufficient number of classifiers are constructed. An unlabeled instance can then be classified through a voting scheme where each constructed classifier gets a weighted vote. The weight of each vote corresponds to the accuracy of the classifier that cast it. AdaBoost is an effective technique to increase the accuracy of simple classifiers, but interpreting the constructed classifier is difficult because a prediction is not the result of a single classifier.

Random forest is an ensemble technique that utilizes decision trees. Similar to AdaBoost it operates by creating a large number of decision trees and uses a voting mechanism to assign a class to unlabeled instances. A single decision tree in random forest is created using a random subset of the training data’s attributes. Additionally, the training data for a single decision tree has randomness introduced through bagging, which is sampling the original training data with replacement (Brieman, 2001). It has been found that random forest can achieve quite good accuracy in comparison to other methods, but again interpreting the constructed classifier is difficult.

2.2 Instance selection

The process of instance selection was first utilized for instance based classifiers, such as k-Nearest Neighbors (k-NN), because faster and less costly classifications could be obtained by maintaining only certain necessary instances in the classifier’s dataset (Hart, 1968; Ritter et al., 1975; Wilson, 1972). That is, because instance based classifiers perform calculations for each instance of a dataset every time a new classification is to be made, a smaller dataset would require a smaller amount of memory storage and a fewer number of calculations. The goals of the first instance selection algorithms were therefore to select the minimum number of instances required to maintain the current classification accuracy of a dataset (Hart, 1968). However, it has been found that instance selection not only reduces the size of the dataset, but it also improves the dataset quality by not selecting to maintain outlying, noisy, contradictory or simply unhelpful instances (Sebban et al., 2000; Zhu and Wu, 2006; Olvera-Lopez et al., 2010).

As a consequence of instance selection's ability to improve the quality of a dataset, a developing use of instance selection ***outside*** of instance based classifiers is to select good training datasets for learning a classification model, such as with the decision tree or neural network classifiers. This area of instance selection is different than early methods used for instance based classifiers because the goal is no longer data reduction but it is rather to maximize classifier accuracy. However, a good selection of instances for an instance based classifier may not lead to the best training dataset for another type of classifier, indicating methods that incorporate the intended classifier in the learning process are warranted (Olvera-Lopez et al., 2009). In fact, it is this author's belief that using the intended classifier in instance selection procedures is beneficial because instances that obscure advantageous structure or relationships in regard to the classifier's bias can be removed. This makes learning good boundaries between class values easier for the classifier. Therefore, with the majority of instance selection algorithms being developed for instance based classifiers, effort should be given to develop instance selection algorithms that are applicable to other classification methods used in practice.

A recent review paper notes that the majority of instance selection methods for training set selection find an acceptable set of training instances utilizing random search with evolutionary algorithms (Garcia-Pedrajas, 2011). Evolutionary algorithms are a popular search technique because they are capable of taking into consideration the particular bias of a specific classification algorithm, and as such, lead to a good selection of training instances for that classifier. Most instance selection methods for training set selection are designed for neural networks and decision trees, but could be adapted for a variety of different classification algorithms that learn a classification scheme from a training dataset. The majority of the evolutionary algorithms attempt to maximize training accuracy as a proxy for testing accuracy, where training accuracy is the percentage of correct predictions made by the classifier when predicting the original training instances, and where testing accuracy is the percentage of correct predictions made when predicting instances withheld from the training process (Han and Kamber ,2006).

Training set instance selection procedures for neural networks serve to increase the generalization ability of the networks by selecting good instances on which to train. These procedures should be beneficial to neural networks because in the presence of noisy data, the networks become overfitted to the training dataset and subsequently are poor classifiers to unseen instances (Kim, 2006). Reeves and Taylor (1998) and Reeves and Bush (2001) perform instance selection for radial basis function (RBF) nets with evolutionary algorithms where the fitness of a solution is determined by the training accuracy of the RBF net, and the instances contained in the training dataset are evolved. It is found that indeed the generalization performance of RBF nets can be increased through training set instance selection. Kim (2006) also performs instance selection for increased generalization ability, but this work is focused on feed forward neural networks and financial forecasts. Here, an evolutionary algorithm determines the training instances and the structure of the network, and the fitness of a solution is evaluated by the training accuracy of the network. Again, the results indicate that the generalization performance of neural networks can be improved by selecting instances to create a higher quality training dataset.

The decision tree classifier is another popular choice for training set instance selection because unhelpful instances in the decision tree's training dataset cause the tree to unnecessarily grow its structure (Sebban et al., 2000; Oates and Jensen, 1997). This overfitting problem defeats the purpose of the decision tree by hiding or confusing the discovered knowledge in a large and un-interpretable tree structure that has poor generalization abilities. Performing instance selection often results in a smaller and more interpretable tree, an indication that the unhelpful instances have not been selected. Endou and Zhao (2002), Cano et al. (2003), and Cano et al. (2006), evolve the instances contained in a subset of the training dataset in the hopes of finding a collection of instances that adequately describe the full dataset. Endou and Zhao (2002), as well as Cano et al. (2006), evaluate the fitness of the selected training dataset through the construction and evaluation of a decision tree from the training dataset, and both of these methods are successful in reducing decision tree size, while still maintaining acceptable, if not improved, levels of accuracy. Cano et al. (2003) evaluates the fitness of the training dataset using the 1-NN classifier, and interestingly, this method is also successful in creating simple, yet accurate, decision trees.

These methods indicate that decision tree learning can indeed be improved by selecting a good set of training instances through instance selection.

One exception to the usual pattern of training set instance selection is found in a paper by Zhu and Wu (2006), where the training dataset's instances are put into groups of similar instances, and then some of the groups are greedily selected to form a reduced training dataset with the goal of improving the classification training accuracy of the naïve Bayes and decision tree classifier. This work is notably different from other methods because instead of performing an evolutionary algorithm to select a training dataset from individual instances, a greedy selection procedure is used to select a training dataset from the created groups. Results show that this method is also successful in increasing classification accuracy.

Another notable exception to the usual pattern of training set instance selection is work done by Olvera-Lopez et al. (2009). In this paper a greedy selection procedure is developed that incrementally subtracts instances from the training data to achieve a suitable subset of instances for classifier learning. This is not unlike Bennette and Olafsson (2011), with the exception that Bennette and Olafsson extend the idea by creating a large number of such subsets with the intent of recombining them in a way that results in even better classification accuracy.

In all cases, previous instance selection techniques do not attempt to create a solvable formulation of the instance selection problem. Instead, most techniques are content to employ heuristics that simply search a prohibitively large solution space. This dissertation breaks the trend and presents an integer programming formulation of instance selection that has structure which can be exploited to find improving selections of instances.

2.3 Column generation for LPs

Column generation is a set of techniques to solve linear programs (LP) that have a huge number of variables (Desrosiers and Lubbecke, 2005; Wilhelm, 2001). In column generation the original LP is called the master problem (MP). Column generation is useful when it is too computationally expensive to solve the MP with all of the original variables or when it would be too computationally expensive to even enumerate all of the decision variables. There are several approaches to column generation, but each involves solving a reduced portion of the MP, called the reduced master problem (RMP). One approach to column generation, commonly referred to as Type I column generation, uses an auxiliary model (AM) to find a subset of the MP's useful variables and then supplies those to the RMP with the hope that the RMP will find a good feasible solution to the MP. However, because not all variables are considered, the solution to the RMP cannot be guaranteed to be optimal to the MP. Another approach to column generation, Type II column generation, also involves supplying the RMP with a subset of the MP's variables. However, in this approach, the RMP is solved and then its dual variables are supplied to a price out problem (POP). The POP generates another decision variable of the MP that will allow the RMP to improve its solution. The RMP is then resolved with the new and improved decision variable, creating a new set of dual variables. The POP can continue to interact with the RMP and generate improving variables until the optimal solution of the MP is found.

2.3.1 Type I column generation

Say there is a linear program (LP) $z_{LP} = \max_{x}\{c^T x: Ax \leq b, x \geq 0\}$, which is so large that defining all of its possible variables is computationally impractical. Defining a variable j requires defining c_j and a_j, where c_j is the objective coefficient associated with variable j

and a_j is the column of the A matrix defined by variable j. The LP with all of its variables defined will be called the master problem (MP). However, only a small number of the possible variables can actually be known or be provided to the RMP. Therefore the reduced master problem (RMP) can be defined as $z_{RMP} = \max_{x}\{\dot{c}^T\dot{x} : \dot{A}\dot{x} \leq b, \dot{x} \geq 0\}$, where $\dot{x}$ contains only the known variables and $\dot{c}$ and $\dot{A}$ contain the objective and constraint values for those variables.

In Type I column generation the RMP is provided high quality decision variables from the auxiliary model and then solved. Even though the optimal solution to the RMP is a feasible solution to the MP, it cannot be guaranteed to be the optimal solution to the MP. Therefore, it is hoped that the optimal solution to the RMP is an acceptably good feasible solution to the MP, mainly because the solution is found from what are considered good decision variables (Wilhelm, 2001). Alternatively, Type II column generation provides a method to guarantee that the optimal solution to the MP is found.

2.3.3 Type II column generation

Type II column generation exploits a feature of the Simplex algorithm for solving linear programs. Say there is a known feasible solution to $z_{LP} = \max_{x}\{c^Tx: Ax \leq b, x \geq 0\}$. For a standard form LP any feasible solution can be written in terms of the basic and non-basic variables. The Simplex algorithm moves to the optimal solution of the LP by switching variables from the basis to the non-basis in a way that improves the current feasible solution. The reduced cost is calculated for each variable belonging to the non-basis during an iteration of the Simplex algorithm, and when there are no non-basis variables with positive reduced cost, the optimal solution to the LP has been found. Type II column generation provides a method to avoid examining every variable during the Simplex algorithm and for linear programs that are so large they are infeasible to solve, this procedure can make them solvable.

As in Type I column generation, say there is a very large LP. The LP with all of its decision variables defined is the master problem (MP), and the restricted master problem (RMP) $z_{RMP} = \max\limits_{x}\left\{\dot{c}^T\dot{x} : \dot{A}\dot{x} \leq b, \dot{x} \geq 0\right\}$ is the LP with only the known decision variables defined. Now the optimal dual variables can be used in the price out problem (POP) to check if the optimal solution to the RMP is optimal to the MP. Failing to find that the solution is truly optimal, the POP can provide a new variable for the RMP. This new variable will have positive reduced cost, and by definition can improve the RMP's solution. The RMP can then be resolved, provide new dual variables to the POP, and the process repeated until no new positive reduced cost variables can be generated.

The formulation of a POP is specific to every RMP, but contains the same basic objective function and should be capable of generating all of the undefined variables belonging to the master problem. The objective of a POP is to maximize the reduced cost of a newly generated variable x^{new}. The objective of every POP should therefore be $\max\{c^{new} - \pi^* a^{new}\}$, where c^{new} is the objective coefficient of x^{new}, $\pi^*$are the optimal dual variables to the RMP, and a^{new} is the column of the $\dot{A}$ matrix to be associated with x^{new}. Provided constraints in the POP are able to dictate how the new variable affects constraints in the RMP, and given a method to calculate the objective coefficient of the new variable, Type II column generation is complete. When solving the POP if the optimal objective value is less than or equal to zero, indicating there are no positive reduced cost variables, it is known that the solution to the RMP is optimal to the MP. Otherwise the new variable should be added to the RMP and the process repeated.

CHAPTER 3. MODEL FORMULATION

This chapter includes two separate integer programming formulations of instance selection. Each formulation is designed to facilitate column generation procedures that can help find improving subsets of instances (or columns). Also discussed in this chapter are procedures to create initial sets of columns, the development of column generation for instance selection, necessary approximations for the price out problems, and a discussion of instance selection parameters.

3.1 Integer programming formulations of instance selection

The primary research objective of this dissertation is to find a structured integer programming formulation of instance selection. The motivation for doing so is to impose a structure on instance selection that will help guide a search for improved subsets of the training data. The two formulations are as follows,

$$Max \qquad Z = \sum_{j \in J} c_j x_j \qquad (4)$$

$$s.t \qquad \sum_{j \in J} a_{ij} x_j \leq 1, \ \forall\, i \in I \qquad (5)$$

$$x_j \in \{0,1\}, \forall j \in J \qquad (6),$$

and,

$$Max \qquad Z = \sum_{j \in J} c_j x_j \qquad (7)$$

$$s.t \qquad \sum_{j \in J} x_j \leq 1 \qquad (8)$$

$$x_j \in \{0,1\}, \forall j \in J \qquad (9).$$

In both formulations the decision variable x_j represents the decision to include or not include a subset of the instances (column), $S^{(j)}$, in the final training dataset. Associated with each column $S^{(j)}$, is a vector a_j. If a_{ij}, an element of a_j, has value one it indicates that the i^{th} instance of the training dataset is included in column $S^{(j)}$, otherwise it is not. This means that if the j^{th} column is included in the final training dataset, or $x_j = 1$, and the i^{th} instance is in that column, or $a_{ij} = 1$, then the i^{th} instance is included in the final dataset. The objective of each formulation is simply to maximize the accuracies of the selected columns, where c_j is the accuracy of the j^{th} column, or as that implies the accuracy of a classifier built from the instances in column $S^{(j)}$.

Unfortunately for large datasets it is computationally impractical to enumerate all of the 2^n decision variables that define *J*, where n is the number of instances in the training dataset (indexed by *I*). To resolve this issue, concepts from column generation are used to generate what are felt to be good decision variables for the instance selection problem. That is, column generation is used to generate columns that have good accuracy. After a number of good columns are generated, any number of solution procedures can then be used to obtain a suboptimal, but hopefully good, solution. A Type II column generation procedure for each of the above IP formulations encourages the generation of different but helpful types of columns.

The logic behind the first integer program, (4) through (6), is that the instances of the IP's most accurate columns should be included in the final selection of instances. This formulation acknowledges the fact that the most accurate column of a dataset is unlikely to be found, and that combining the instances of good columns may be helpful. However the constraint defined by (5) says that selected columns should be disjoint, meaning they should not have any instances in common. The result of this formulation should be a column generation procedure that strives for ever-improving and diverse columns. Meaning, column generation should generate new columns that are similar to existing columns, but that result in higher accuracy.

The second IP formulation, (7) through (9), has a constraint that says only one column can be selected by the IP, $\sum_{j \in J} x_j \leq 1$. It should be noted that this constraint also enforces the constraint defined in (5). The logic behind such a constraint is that the final training dataset should include only the instances of the most accurate column, and including any additional columns will not be helpful. This formulation would perfectly solve the instance selection problem, except, as mentioned earlier, finding the most accurate column for a dataset is unlikely. Still, this IP is attractive in a column generation procedure because any newly generated column must have accuracy higher than that of the previous best column. This may lead to the discovery of very useful columns.

Before Type II column generation can be implemented an initial set of columns needs to be generated. This can be accomplished with what can be thought of as a Type I column generation procedure. Each IP can utilize the same initial set of columns and three such methods are introduced in Section 3.2.

3.2 Constructing initial sets of columns

It is believed, and observed in preliminary experiments, that a set of initial columns for the provided IPs should be diverse and contain helpful instances. By diverse it is meant that the combination of instances included in a particular column is unique to that column, and dissimilar to combinations found in other columns. Additionally, a column is thought to include helpful instances if the accuracy of a classifier built from those instances has high accuracy. Three approaches to create initial sets of columns are introduced, each imposing varying levels of instance diversity and helpfulness.

Each approach to creating initial columns for the instance selection IPs relies on an auxiliary model akin to that needed for Type I column generation. The first two methods use greedy selection procedures that incorporate the base classifier of the instance selection problem, meaning the classifier considered for improvement, and the last method simply constructs initial columns with randomly included instances. Each method results in the creation of a column for every instance. This means that every instance is guaranteed to be included in at least one column, and the number of columns created is the same as the number of instances in the training data.

3.2.1 Backward selection

In the case of the backward selection procedure, a single column, $S^{(j)}$, initially contains all of the original training instances, and instances are removed from the column if doing so does not greatly hinder the predictive accuracy of a classifier built from that column. In this way it is thought that columns will contain helpful instances, because instances necessary to preserving classifier accuracy remain in the constructed columns. To introduce diversity to the constructed columns instances are considered for removal from the columns in a random order. Additionally, to ensure that each instance is represented in at least one column, each instance generates a column where that instance is not considered for removal. Pseudo code for the Backward Selection Algorithm is provided below. Note that "Predictive Accuracy" can be calculated by constructing the base classifier from the provided instances and evaluating it on the original training data.

Backward Selection Algorithm

Step 0:

$Given\ a\ training\ set\ T = \{\tau_1, \tau_2, \ldots, \tau_n\}\ with\ n = |T|$

$and\ a\ \beta, such\ that\ \beta\ is\ an\ acceptable\ loss\ of\ percentage\ points.$

Step 1:

```
$For\ j = 1\ to\ n$
        $S^{(j)} = \{\tau_1, \tau_2, \ldots, \tau_n\}$
        $U^{(j)} = \left\{u_1^{(j)}, u_2^{(j)}, \ldots, u_{n-1}^{(j)}\right\}\ where\ u_i^{(j)}\ is\ the\ i^{th}\ integer\ of\ a$
                                  $randomization\ of\ the\ integers$
                                  $from\ 1\ to\ n, less\ the\ integer\ j.$
```

Step 2:

```
$For\ j = 1\ to\ n$
        $For\ i = 1\ to\ n - 1$
                $If\ Predictive\ Accuracy\ \left(S^{(j)} / \left\{\tau_{u_i^{(j)}}\right\}\right) \geq Predictive\ Accuracy(T) - \beta$
                        $Then\ S^{(j)} = S^{(j)} / \left\{\tau_{u_i^{(j)}}\right\}$
```

$Else\ S^{(j)} = S^{(j)}$.

3.2.2 Forward then backward selection

In the case of forward then backward selection, columns begin empty except for a single instance, and instances that improve the accuracy of the column are added in a greedy fashion. Once all of the original training instances have been considered for addition to the column, all of the accepted instances are then considered for removal in a greedy fashion. Instances are only removed from the column if doing so does not greatly hinder the column's accuracy. As with the strictly backward selection method, a column is constructed for each instance of the original training data and instances are considered for addition to the column and subtraction from the column in a random order.

This method strives to find columns that contain only helpful instances by allowing all instances that improve the accuracy of the column to be included in the column. However, in preliminary experiments it was found that a strictly forward selection procedure seemed to include instances that were only marginally helpful. It was then found helpful to consider removing instances from the columns. This is presumably the case because some instances added later in the forward selection procedure could serve the purpose of some of the instances added early on. Below is pseudo code for the Forward then Backward Selection Algorithm, where "Predictive Accuracy" is calculated as before.

Forward then Backward Selection Algorithm

Step 0:

$Given\ a\ training\ set\ T = \{\tau_1, \tau_2, \ldots, \tau_n\}\ with\ n = |T|$

$and\ a\ \beta, such\ that\ \beta\ is\ an\ acceptable\ loss\ of\ percentage\ points.$

Step 1:

```
$For\ j = 1\ to\ n$
        $S^{(j)} = \{\tau_j\}$
        $U^{(j)} = \{u_1^{(j)}, u_2^{(j)}, \dots, u_{n-1}^{(j)}\}$ $where\ u_i^{(j)}\ is\ the\ i^{th}\ integer\ of\ a$
                                $randomization\ of\ the\ integers$
                                $from\ 1\ to\ n, less\ the\ integer\ j.$
        $V^{(j)} = \{v_1^{(j)}, v_2^{(j)}, \dots, v_{n-1}^{(j)}\}$ $where\ v_i^{(j)}\ is\ the\ i^{th}\ integer\ of\ a$
                                $randomization\ of\ the\ integers$
                                $from\ 1\ to\ n, less\ the\ integer\ j.$
```

Step 2:

```
$For\ j = 1\ to\ n$
        $For\ i = 1\ to\ n-1$
                $If\ Predictive\ Accuracy\ \left(S^{(j)} \cup \left\{\tau_{u_i^{(j)}}\right\}\right) > Predictive\ Accuracy\left(S^{(j)}\right)$
                        $Then\ S^{(j)} = S^{(j)} \cup \left\{\tau_{u_i^{(j)}}\right\}$
                $Else\ S^{(j)} = S^{(j)}$
```

Step 3:

```
$For\ j = 1\ to\ n$
        $S_1^{(j)} = S^{(j)}$
        $For\ i = 1\ to\ n-1$
                $If\ Predictive\ Accuracy\ \left(S^{(j)} / \left\{\tau_{v_i^{(j)}}\right\}\right) \geq Predictive\ Accuracy\left(S_1^{(j)}\right) - \beta$
                        $Then\ S^{(j)} = S^{(j)} / \left\{\tau_{v_i^{(j)}}\right\}$
                $Else\ S^{(j)} = S^{(j)}$ .
```

3.2.3 Random selection

In the random selection procedure no classifier information is used to determine which instances are included in a column. Meaning, this method relies on random chance for helpful instances to be included in any one column. However, by definition, this method creates diverse columns. As with the previous methods a column is generated for each instance, where that instance is guaranteed to be included in the column. The remaining training instances are of course randomly assigned to the column. One additional requirement of this method is for the user to decide how many instances should be included in a single column. This may require some background knowledge of the classification problem and classifier to make an informed decision. Pseudo code for the Random Selection Algorithm is shown below.

Random Selection Algorithm

Step 0:

```
$Given\ a\ training\ set\ T = \{\tau_1, \tau_2, \dots, \tau_n\}\ with\ n = |T|$
$and\ a\ \lambda, such\ that\ \lambda\ is\ the\ number\ of\ instances\ to\ include\ in\ each\ column.$
```

Step 1:

```
$For\ j = 1\ to\ n$
    $S^{(j)} = \{\tau_j\}$
    $U^{(j)} = \{u_1^{(j)}, u_2^{(j)}, \dots, u_{n-1}^{(j)}\}\ where\ u_i^{(j)}\ is\ the\ i^{th}\ integer\ of\ a$
                    $randomization\ of\ the\ integers$
                    $from\ 1\ to\ n, less\ the\ integer\ j.$
```

Step 2:

```
$For\ j = 1\ to\ n$
```

```
$For\ i = 1\ to\ \lambda - 1$
        $S^{(j)} = S^{(j)} \cup \tau_{u_i^{(j)}}$ .
```

3.3 Implementing column generation

Given the two integer programming formulations of instance selection and three methods to create initial columns, a simple variant of Type II column generation can now be defined. To implement Type II column generation each integer program needs to have a master problem (MP) defined, a reduced master problem (RMP) defined, and a price out problem (POP) defined. Each MP is a linear relaxation of the original integer program, and the RMP is defined as the MP having only considered a subset of the possible columns. The POPs for each integer program are similar but have slightly different objective functions to generate valid columns for their respective IPs.

3.3.1 Implementation one

Consider the formulation of instance selection defined by the IP in (4) through (6). In order to implement a Type II column generation procedure the integrality constraints of (6) need to be relaxed. In this particular case it is reasonable to relax the integrality constraints because the resultant linear program restricts the number of times an instance can be selected for the final training data. Logically, the effect of this constraint combined with the objective function is that the MP will strive to select whole columns with high accuracy. In the actual column generation procedure the intended result will be variations of existing columns that have improved accuracy. The resulting MP is,

$$Max \qquad Z = \sum_{j \in J} c_j x_j \qquad\qquad (10)$$

$$s.t \qquad \sum_{j \,\epsilon\, J} a_{ij} x_j \leq 1, \ \ \forall\, i \,\epsilon\, I \qquad (11)$$

$$x_j \geq 0, \ \ \forall j \,\epsilon\, J \qquad (12)$$

$$x_j \leq 1, \ \ \forall j \,\epsilon\, J, \qquad (13)$$

and the related RMP is,

$$Max \qquad Z \ = \sum_{j \,\in J'} c_j x_j \qquad (14)$$

$$s.t \qquad \sum_{j \,\epsilon\, J'} a_{ij} x_j \leq 1, \ \ \forall\, i \,\epsilon\, I \qquad (15)$$

$$x_j \geq 0, \ \ \forall j \,\epsilon\, J' \qquad (16)$$

$$x_j \leq 1, \ \ \forall j \,\epsilon\, J'. \qquad (17)$$

The POP is then,

$$Max \qquad Z^{POP} = \ Classifier\ Acc. - \sum_{i=1}^{n} \pi_i^* a_i \qquad (18)$$

$$s.t \qquad \sum_{i \,\epsilon\, I} a_i \leq G \qquad (19)$$

$$a_i \epsilon \{0,1\}\,,\quad \forall i \epsilon I, \qquad (20)$$

where we let $X^{(1)}$ denote the feasible region defined by (19) and (20).

In this POP the decision variables are binary. When a specific a_i assumes the value one, it represents the decision to include instance i in the new column, and a_i equal to zero represents the decision to exclude instance i from the new column. The sole constraint $\sum_{i \in I} a_i \leq G$ restricts the number of instances selected to belong to a new column to be less than or equal to some constant *G*. This constant *G* allows the POP to directly control the size of the newly generated training data subsets. The objective of this POP is to maximize reduced cost, but as stated previously, a closed form function is not known for calculating the accuracy of a classifier based on the instances it is trained from. This makes the POP unsolvable, but in Section 3.4, approximation techniques to solve the POP will be presented.

3.3.2 Implementation two

Consider the formulation of instance selection defined by the IP in (7) through (9). In order to implement a Type II column generation procedure the integrality constraints of (9) need to be relaxed. However, the integrality constraints of (9) can be relaxed and still allow for an integer solution to be found, as will now be proven.

Lemma 1. Given that x^* is optimal to $Z = \left\{ \max \sum_{j \in J} c_j x_j : \sum_{j \in J} x_j \leq 1, 0 \leq x_j \leq 1 \; \forall \, j \in J \right\}$, $\sum_{j \in J} x_j^* = 1$ when $c_j > 0 \; \forall \, j \in J$.

Proof: Assume $\sum_{j \in J} x_j^* < 1$. Then there exists i such that $x_i^* < 1$. Construct a new solution x' such that $x_i' = x_i^* + \left(1 - \sum_{j \in J} x_j^*\right)$, and $x_j' = x_j^* \; \forall \, j \neq i$. Since $1 - \sum_{j \in J} x_j^* \leq 1 - x_i^*$, then $x_i' \leq x_i^* + (1 - x_i^*)$ and it follows that $x_i' \leq 1$. Additionally, since $\sum_{j \in J} x_j^* = \sum_{j \neq i} x_j^* +$

x_i^*, then $\sum_{j\in J} x_j' = \sum_{j\in J} x_j^* + \left(1 - \sum_{j\in J} x_j^*\right)$, and again it follows that $\sum_{j\in J} x_j' \leq 1$. Finally, because $\sum_{j\in J} c_j x_j' = \sum_{j\in J} c_j x_j^* + c_i\left(1 - \sum_{j\in J} x_j^*\right)$, and since $c_i\left(1 - \sum_{j\in J} x_j^*\right) > 0$, it is true that $\sum_{j\in J} c_j x_j' > \sum_{j\in J} c_j x_j^*$. There is a contradiction, x^* is not the optimal solution. ■

Lemma 2. If $c_k < \max_i c_i$ then $x_k^* = 0$, where x^* is the optimal solution to $Z = \left\{ \max \sum_{j\in J} c_j x_j : \sum_{j\in J} x_j \leq 1, 0 \leq x_j \leq 1 \, \forall \, j \in J \right\}$, when $c_j > 0 \, \forall \, j \in J$.

Proof: Assume there exists k such that $c_k < \max_i c_i$ and $x_k^* > 0$. Given i such that $i = arg \max_i(c_i)$ define a new solution x' where $x_i' = x_i^* + x_k^*$, $x_k' = 0$, and $x_j' = x_j^* \, \forall \, j \neq i \; or \; k$. Since $x_k^* + x_i^* \leq \sum_{j\in J} x_j^*$ and $\sum_{j\in J} x_j^* \leq 1$, then $x_i' \leq 1$. Additionally since $\sum_{j\in J} x_j' = \sum_{j\neq i \; or \; k} x_j^* + x_i^* + x_k^* = \sum_{j\in J} x_j^*$, and $\sum_{j\in J} x_j^* \leq 1$, then $\sum_{j\in J} x_j' \leq 1$. Next, because $c_i > c_k$ and $x_k^* > 0$, it is known that $\sum_{j\in J} c_j x_j^* < \sum_{j\neq k} c_j x_j^* + c_i x_k^*$. Finally, it can be shown that $\sum_{j\in J} c_j x_j' = \sum_{j\neq k} c_j x_j^* + c_i x_k^*$. There is a contradiction, x^* is not the optimal solution. ■

Theorem 1. Given that $c_j > 0 \, \forall \, j \in J$, $Z = \left\{ \max \sum_{j\in J} c_j x_j : \sum x_j \leq 1, 0 \leq x_j \leq 1 \, \forall \, j \in J \right\}$ has an optimal solution such that $x_j^* \in \{0,1\} \, \forall \, j \in J$.

Proof: Assume there is no optimal solution such that $x_j^* \in \{0,1\} \, \forall \, j \in J$. Because of Lemma 1 it is known that $\sum_{j\in J} x_j^* = 1$. Because of Lemma 2 it is known that for any j: $x_j^* > 0$ then $c_j = \max_i c_i$, say k. It can then be shown that $\sum_{j\in J} c_j x_j^* = k \sum_{j\in J} x_j^* = k$. Define a new solution x'. Choose some l such that $c_l =$ k. Then define $x_l' = 1$, and $x_j' = 0 \, \forall j \neq l$. Then $\sum_{j\in J} c_j x_j' = \sum_{j\neq l} c_j x_j' + k = k$. It is then known that $\sum_{j\in J} c_j x_j' = \sum_{j\in J} c_j x_j^*$. The new solution is also optimal. There is a contradiction. ■

The required relaxation of (7) through (9) is permissible because as Theorem 1 shows, there is always an integral optimal solution to the relaxed IP. This relaxation makes intuitive sense because it would be suboptimal to select a portion of any column that did not have the highest available accuracy. If two or more columns are tied for the highest accuracy it would be possible to select non-integer portions of each of those, but an optimal integer solution is still possible by selecting the entirety of one column. The result of this fact is that columns generated in the Type II column generation procedure will have higher accuracy than any of the existing columns. The resulting MP is,

$$Max \qquad Z = \sum_{j \in J} c_j x_j \tag{21}$$

$$s.t \qquad \sum_{j \in J} x_j \leq 1 \tag{22}$$

$$x_j \geq 0, \ \ \forall j \in J \tag{23}$$

$$x_j \leq 1, \ \ \forall j \in J, \tag{24}$$

and the associated RMP is,

$$Max \qquad Z = \sum_{j \in J'} c_j x_j \tag{25}$$

$$s.t \qquad \sum_{j \in J'} x_j \leq 1 \tag{26}$$

$$x_j \geq 0, \ \ \forall j \in J' \tag{27}$$

$$x_j \leq 1, \ \ \forall j \in J'. \qquad (28)$$

The POP is then,

$$Max \qquad Z^{POP} = \ Classifier\ Acc. - \sum_{i=1}^{n} \pi_i^* a_i - \pi_{n+1}^* \qquad (29)$$

$$s.t \qquad \sum_{i \in I} a_i \leq G \qquad (30)$$

$$a_i \in \{0,1\}\,, \ \ \forall i \in I, \qquad (31)$$

where we let $X^{(2)}$ denote the feasible region defined by (30) and (31).

The POP formulation is identical to (18) through (20) except that the objective function automatically subtracts the last dual variable. This change simply enforces constraint (26) in the RMP. As before, this POP is not solvable because no closed form function is known to evaluate the accuracy of a classifier based on the contents of its training data. Section 3.4 presents approximation methods for solving the POP.

3.4 Approximating the price out problem

The price out problems defined in (18) through (20) and (29) through (31) are not solvable because there is no known closed form function for calculating the accuracy of a classifier based solely on the contents of its training data. However, if some approximation of classifier accuracy is substituted for actual accuracy, approximations of the price out problems can be solved. The objectives become to generate a column that maximizes an estimation of reduced cost. Any generated column can then be checked for truly positive reduced cost by simply building the associated classifier and using its accuracy in the reduced cost calculation. If the column is indeed beneficial to the RMP it can be added, if it is not, a new column can be generated and considered.

Generating improving columns with an estimated POP requires a good method to predict the usefulness of a classifier based on its training data. Two methods that have been successful in preliminary tests are presented in this section. Both methods find a ranking of the instances and operate under the assumption that the more high-ranking instances contained in a training dataset the higher the accuracy of the resultant classifier. Note that the sum of the ranks of the instances included in a classifier's training data does not represent a prediction of the classifier's accuracy. Rather the sum represents an indication of the classifier's potential usefulness.

Because ranking instances does not result in a prediction of classifier accuracy, but rather a prediction of usefulness, the rank and dual variables used in the estimated POPs are scaled between zero and one. This scaling ensures a fair comparison when considering whether or not to include an instance in a new column. For example, if an instance is ranked relatively high, but has an associated dual variable that is also relatively high, it may not be a good instance to include in the generated column. Of course, it is not ideal to require that the rank of instances and the optimal dual variables be scaled, but it is necessary until some linear approximation of classifier accuracy is developed.

The two methods used to rank instances are discussed in the following subsections. The first relies on information theory and some minor pieces of empirical evidence. The second relies solely on empirical evidence. The ranking procedures allow the POP defined in (18) through (20) to be replaced by,

$$Z^{POP} = max\left\{\sum_{i=1}^{n}(r_i a_i - \pi_i'^{*} a_i) : a \in X^{(1)}\right\}, \qquad (32)$$

and the POP defined in (29) through (31) can be replaced by,

$$Z^{POP} = \ max\left\{\sum_{i=1}^{n}(r_i a_i - \pi_i'^{*} a_i) - \pi_{n+1}'^{*} : a \in X^{(2)}\right\}. \qquad (33)$$

In both (32) and (33) *r* represents a scaled ranking of the instances, achieved using the first or second method, and π'^{*} are the scaled optimal dual variables.

3.4.1 Ranking instance usefulness through information theory

The first method used to rank the usefulness of the training instances is adapted from Zhu and Wu (2006) and does so by quantifying the information lost as a result of a column of instances being removed from the training data. If a large amount of information is lost, or as that implies, a classifier suffers in accuracy and assuredness, it is assumed that the instances contained in the removed column are important to building a good classifier. The more important an instance is deemed to be, the higher its resulting rank.

To quantify the information lost through the removal of a column of instances from the training data, the log loss function is introduced. The log loss function is used to help quantify how much information is lost about a specific instance, say τ, by a classifier. This function, shown in equation (34), relies on the true distribution of an instance given by $P(y|\tau)$, and the predicted distribution of an instance given by $P_E(y|\tau)$, where *y* is a class value of τ. The actual calculation is the negative sum over all possible class values of the true probability that an instance τ belongs to class *y*, times the log value of the predicted probability that the instance τ belongs to class *y*. The result of the calculation is that the more incorrect a classifier is about the true distribution or class of an instance τ, the higher its value of L_τ. The lower bound of this calculation is zero, and there is no upper bound.

$$L_\tau = -\sum_{j=1}^{|y|} P(y_j|\tau) log\left(P_E(y_j|\tau)\right) \tag{34}$$

To utilize the amount of information lost by a classifier about a specific instance, one final calculation is introduced. The behavior of a classifier for an instance τ, say Bh_τ, relays the quantification of information loss, but also incorporates whether or not a correct classification is made for the instance. If a classifier makes the correct class prediction for τ, Bh_τ is equal to 0.1 to the power of L_τ. If the classifier makes the incorrect class prediction, Bh_τ is equal to -0.1 to the power of L_τ. This calculation is summarized in (35). The result of this calculation is that if a classifier loses little information about an instance, and it makes the correct prediction, then Bh_τ is high. If the classifier loses little information, but makes the wrong prediction, then Bh_τ is low. Intuitively, the instances that a classifier is knowledgeable about have a high value of Bh_τ. Bh_τ ranges between one and negative one.

$$Bh_\tau = \begin{cases} 0.1^{L_\tau} & if\ the\ classifier\ correctly\ classifies\ instance\ \tau \\ -0.1^{L_\tau} & otherwise \end{cases} \tag{35}$$

The idea behind the proposed ranking system is to see how the values of Bh_τ change after the removal of a column from a classifier's training data. If the removed column has *k* instances, then the average change in Bh_τ over all of the training instances is added to the rank calculation of the *k* instances. The logic is that if a column contains important instances, then the values of Bh_τ will decrease from their removal, and the average difference between the Bh_τ values will be positive. Given an initial set of columns, all columns can have the effect of their removal measured, and in the end, the instances with the highest rank are those that belong to columns that hurt classification power when removed from the training data. The Information Rank Algorithm is summarized in the pseudo code below.

Information Rank Algorithm

```
Step 0:
$Given\ a\ training\ set\ T = \{\tau_1, \tau_2, \ldots, \tau_n\},\ with\ n = |T|,$
$a\ set\ of\ columns\ S = \{S^{(1)}, S^{(2)}, \ldots, S^{(m)}\}, such\ that\ S^{(j)} \subseteq T,$
$and\ a\ rank\ for\ each\ instance\ r_\tau = 0\ \forall\ \tau \in T.$
Step 1:
$T' = \bigcup_{j=1}^{m} S^{(j)}$
Step 2:
$For\ j = 1\ to\ m$
    $T'' = T' \cap S^{(j)}$
    $\boldsymbol{For}\ \tau \in \boldsymbol{S}^{(j)}$
        $Bh_\tau^{Before} = Bh_\tau\ as\ in\ (35)\ using\ a\ classifier\ built\ from\ T'$
        $Bh_\tau^{After} = Bh_\tau\ as\ in\ (35)\ using\ a\ classifier\ built\ from\ T''$
    $Average\ Difference = \frac{\sum_{i=1}^{n} Bh_\tau^{Before} - Bh_\tau^{After}}{n}$
    $\boldsymbol{For\ \tau} \in \boldsymbol{S}^{(j)}$
        $r_\tau = r_\tau + Average\ Difference$
Step 3:
$Scale\ the\ values\ of\ r\ between\ 0\ and\ 1.$
```

3.4.2 Ranking instance usefulness through the frequency of instance occurrences

The second method used to rank the usefulness of instances is based solely on empirical evidence. Specifically, this method ranks instances according to the frequency with which they appear in columns with high accuracy. Ranking instances this way is likely successful because the high accuracy columns hold instances which are known to work well for classification. If an instance appears frequently in these columns then it probably is indeed useful for classification, receives a high rank, and is prone to being included in new training data subsets. The specific implementation of the Frequency Rank Algorithm is shown in the pseudo code below.

Frequency Rank Algorithm

```
Step 0:
$Given\ a\ training\ set\ T = \{\tau_1, \tau_2, \dots, \tau_n\},\ with\ n = |T|,$
$a\ set\ of\ columns\ S = \{S^{(1)}, S^{(2)}, \dots, S^{(m)}\}, such\ that\ S^{(j)} \subseteq T,$
$and\ a\ rank\ for\ each\ instance\ r_\tau = 0\ \forall\ t \in T.$
Step 1:
$For\ j = 1\ to\ m$
        $\boldsymbol{For\ x \in S^{(j)}}$
                $r_\tau = r_\tau + 1$
Step 3:
$Scale\ the\ values\ of\ r\ between\ 0\ and\ 1.$
```

3.5 Parameters

Instance selection as presented in this chapter requires a number of parameters to be set. These parameters are broken into two groups. First, parameters that require a choice, and second, parameters that require actual values to be chosen.

Choice Parameters:

- Which classifier to use
- Which reduced master problem (RMP) to use
- How to construct the initial columns (*J'*) of the RMP
- How to estimate reduced cost in the price out problem (POP)
- How to measure accuracy
- How to combine columns for a final selection of instances

Value Parameters:

- The number of columns from which to estimate reduced cost in the POP, *R*
- The number of instances the generated columns in the POP can include, *G*
- The number of columns from the POP to check for positive reduced cost, *K*
- The number of restarts allowed in the POP

In the following subsections are descriptions of the different parameters that require tuning or a choice for the application of instance selection. Choice parameters will have the pros and cons of each choice analyzed. Discussed with the description of each value parameter will be values that have worked in preliminary experiments, as well the potential benefit and harm of alternative values.

3.5.1 Which classifier to use

Instance selection should be performed with a specific classifier in mind. This classifier is used in the selection process and the goal is for the classifier to select instances that it can take more advantage of than the original training data. Preliminary results indicate that good classifiers for instance selection are naïve Bayes, logistic regression, and decision trees.

3.5.2 Which reduced master problem (RMP) to use

Both of the instance selection reduced master problems have promising properties for a column generation procedure, but they also both have shortcomings. The end goal of each RMP is to create high quality columns to be used in a search for the best selection of instances from which to build a classifier. However, only one RMP will be considered when creating these improved columns for the instance selection procedure. A short description of each choice, RMP1 as shown in (14) through (17), and RMP2 as shown in (25) through (28), along with its positives and negatives is provided below.

Both RMPs are linear program relaxations of similar integer programs (IP). The IPs upon which each RMP is based, have the same decision variables and objective functions. The decision variables, called columns, can have value zero, indicating the column is not selected, and value one, indicating the column is selected. A column is a subset of the original training instances, and the accuracy of a column is the accuracy of a classifier built from the column's instances as evaluated on the original training data. The shared objective function of each IP is to maximize the sum of the selected columns' accuracies, and columns must be selected such that no instance appears in more than one selected column. With no additional constraints, the IPs select high accuracy yet disjoint columns.

RMP1, the first master problem, has no constraint limiting the number of columns that can be selected in the solution, and the optimal solution to RMP1 is not optimal to the IP upon which it is based. However, the objective function of RMP1 does encourage the selection of disjoint high accuracy columns, making it useful. The price out problem of

RMP1 leads to the generation of solutions that improve upon previous columns, where an improvement is considered a similar column that results in higher accuracy, but these improvements can pertain to any column currently existing in *J'*.

It is obvious that generating improved columns could be beneficial, as they may be attractive building blocks in the search for the best subset of instances. The downside of the formulation of RMP1 is that in the column generation procedure it is possible to create columns that are degradations of previously discovered columns. This seems to clutter the search space with unhelpful columns. It is possible to generate such columns because the objective function defining RMP1 can reward the addition of small, disjoint, and low accuracy columns. This will be impossible with RMP2.

RMP2, the second reduced master problem, has an additional constraint that no more than one column may be selected in the LP solution. The optimal solution to RMP1 is also optimal to the IP upon which it is based (see Theorem One). The solution of RMP2 is to select the highest accuracy column. This means with RMP2 that any improving column found in the column generation procedure must have higher accuracy than any column previously created.

The benefit from generating higher and higher accuracy columns is obvious. Very high accuracy columns could be a good starting point in the search for the best subset of instances. The downside of RMP2 is also its strength. Since the price out problem can only generate columns with higher accuracy than the accuracy of columns previously discovered, potentially helpful columns in the search to find the best subset of instances may be omitted. For an example of such a column imagine a new column that has completely unique instances from the current best column, but also has equal accuracy. This column is not discovered or considered in the price out problem of RMP2, but could obviously be beneficial if considered in the construction of a high accuracy subset of training instances.

3.5.3 How to construct the initial columns (*J'*) of the RMP

J' is the set of initial columns used to populate the RMP in the column generation procedure. It is known that the number of iterations required to solve a column generation procedure depends on this initial population. Even though column generation for instance selection will never be run to true optimality, it is clear that the quality of generated columns will depend on the initial columns of the RMP. One glaring reason for this dependency is that the two methods provided to estimate reduced cost depend on a subset of the columns from the RMP. For the early iterations of column generation the initial *J'* is inextricably linked to the approximation of reduced cost.

Preliminary experimentation has shown that good initial columns of *J'* should be diverse and contain beneficial instances (indicated by high column accuracy). However, it is not clear exactly how diverse the columns should be and how sensitive the procedure is to the inclusion of columns with non-beneficial instances. Five ways to create initial columns are summarized below, along with the positives and negatives of each.

The Backward Selection algorithm with β equal to zero (B0) is the first method considered to create initial columns. B0 creates a column based on each training instance. Initially, a column contains all of the original training instances, but each instance (except the instance the column is created for) is considered for removal in a random order. An instance is removed from the column if it is un-helpful, meaning its removal does not change the accuracy of the column. With B0, diversity is created by the random order instances are considered for removal from the column and helpful instances are purposefully not removed from the column.

The positive of B0 columns is that some unnecessary instances are removed from each subset. The negative is that columns contain a somewhat large number of instances, so large that it is expected they still contain un-helpful instances. These un-helpful instances may make the estimates of reduced cost ineffective and may encourage the inclusion of harmful instances in the columns generated by the price out problem.

The Backward Selection algorithm with β equal to ten (B10) is the second method considered to create initial columns. These columns are created identically to the columns of B0, but now an instance is removed if doing so does not decrease the column's original accuracy by more than ten percentage points. As with B0, B10 brings in diversity through the random order instances are considered for removal from the column. With B10 some of the helpful instances are purposefully not removed from the column. However, some marginally helpful instances can be removed from the column, as controlled by the allowed loss of ten percentage points from the original column accuracy.

The positive of B10 columns is that the columns contain fewer instances than the columns of B0, possibly having removed more of the unnecessary instances. The negative, indicated by the loss of training accuracy, is that helpful and necessary instances may have also been removed from the columns.

The Forward then Backward Selection Algorithm with β equal to zero (FB) is the third method considered to create initial columns. In this procedure, again, a column is made for each instance. This time the initial column contains only the instance for which it is created. The remaining instances are then considered in a random order for inclusion in the column. Instances are included if doing so increases the accuracy of the column. After all of the instances are considered for inclusion, or the forward phase, the instances in the column are considered in a random order for removal. Instances are removed if doing so does not hurt the accuracy of the column. FB introduces diversity to the columns through the random order instances are considered for addition and subtraction, and extra effort is made to only include helpful instances.

The positive of the FB columns is that the columns have been carefully constructed to contain more helpful and fewer unnecessary instances than in previous methods. The negative is that they may not have enough diversity to help the price out problem find improving columns, as indicated in preliminary experiments.

The two final methods consider the Random Selection Algorithm to create columns that contain randomly chosen instances. As with the other methods the number of random columns is equal to the number of the original training instances. The positive of these types of columns is that they are quite diverse. The negative is that helpful instances are included in columns only by random chance. Additionally, the user must decide how many instances should be included in each column.

For the first type of random columns (RB), the number of random instances included in each column is equal to the average number of instances included in the best columns created by each of the three previous methods. The reasoning for including this number of instances is that the previous methods found good columns by including a certain number of instances, and the subset sizes are similar enough to warrant the average. It may be possible to recreate this prior success with the random inclusion of instances, the result of which would be quite diverse columns that contain helpful instances.

For the second type of random columns (RU), the number of random instances included in each column is decided by first creating a large number of randomly sized columns. Specifically, a large number of columns containing random instances are created for each possible number of instances (one instance through *n* instances, where *n* is the number of original instances). Then the best column from each size is plotted, and a user selects where it seems the column accuracy peaks or levels off. The reasoning for this is again to create diverse columns, but to do so with column sizes that seem to lead to high accuracy columns, or columns with helpful instances.

3.5.4 How to estimate reduced cost in the price out problem (POP)

The objective function of the POP is to maximize the difference between classifier accuracy and the sum of the dual variables. In order to solve the price out problem, a linear function describing the accuracy of a classifier based on its training dataset needs to be constructed. Unfortunately, no closed form, let alone linear, function exists. An estimation of classifier accuracy is used instead.

There are two current methods considered to estimate reduced cost.   The first is based on information theory, and the second is based on the frequency of instances appearing in the high accuracy columns of *J'*. Both result in a ranking of the instances, which can then be used as a linear function estimating the accuracy of a classifier based on its training data. The intuition is that columns built from high-ranking instances will have high classifier accuracy and columns built from low-ranking instances will have low classifier accuracy. Unfortunately, because this estimation does not correspond directly with classifier accuracy, it is necessary to scale the ranks and dual variables between zero and one. This way, when calculating reduced cost, the difference between an instance's contribution to classifier accuracy and it's dual variable is somewhat comparable. For both methods of approximation, the goal is to give helpful instances a high rank. The positive and negatives of each ranking method are discussed below.

The first method used to assign a ranking to the instances of the original training data is based on information theory. The basic idea behind this method is that the amount of classification ability lost by removing a column of instances from the training data can be quantified. Columns that when removed from the training data lead to a big loss of accuracy are said to contain important instances, and these instances are given a high ranking. This method is outlined in the Information Rank Algorithm, and it relies on having a set of initial columns. This set is a subset of columns from *J'* where the columns in *J'* are ordered from the most to least accurate, and as many columns as needed to represent *R*% of the instances from the original training data are included. Starting with the most accurate. Of course, *R* is another parameter that requires a value.

The positive of this method is that it attributes a quantification of classification ability to each instance. If this quantification is accurate, the measure has the potential to be quite informative when ranking instances. The negative of this method is that instances are never judged alone. Instead whole columns are used to measure a change in classification ability, and the difference is attributed to each instance. It is possible that some unhelpful instances will accidentally receive a high ranking by always being grouped with helpful instances, and some good instances will receive a low ranking by always being grouped with unhelpful instances.

The second method used to assign a ranking to the instances of the training data is based on the frequency with which instances appear in the high accuracy columns of *J'*. This method is outlined in Frequency Rank Algorithm and is calculated by first ordering the columns in *J'* from highest accuracy to lowest accuracy. A subset of *J'* is then created. The subset contains as many columns as required to have the union of their contents represent *R*% of the instances from the original training data. Columns are added to the subset in order of the most to the least accurate. The number of times each instance occurs in the subset of columns then corresponds to its rank. Meaning, instances that occur very frequently in the high accuracy columns are ranked high, and are thought to help create high accuracy columns.

A positive of basing the rank of instances on the frequency of instances in high accuracy columns is that the rank is based on empirical evidence from the dataset. Indeed, preliminary testing has shown that this ranking does lead to the discovery of improving columns. This means at least some of the helpful instances receive a high ranking. A negative of this method is that some truly helpful instances may not appear frequently in the initial *J'* columns, and never enter newly created columns. The opposite may also occur, where harmful instances are included in the initial *J'* by mistake, and propagate through to new columns. An additional negative is that this method seemingly requires a good set of initial columns to be successful; otherwise the ranking may be of little consequence.

3.5.5 How to measure accuracy

A measurement of accuracy is needed in several steps of the instance selection procedure. First call to mind both formulations of the reduced master problem. In both formulations the objective function is to maximize the sum of the selected columns' accuracy, where column accuracy is calculated as the accuracy of a classifier built from the column's instances evaluated on the original training data. Second call to mind the creation of *J'*, most of the procedures used to create initial columns also require some measurement of accuracy. In both cases any measurement of accuracy can be used. Two measurements are discussed here, regular accuracy and class balance accuracy. Only the more appropriate of the two will be considered for a particular application. The two measurements are discussed below along with their positives and negatives.

Regular accuracy is calculated by dividing the number of correct predictions made by a classifier by the number of total predictions. The benefit of this measurement is that it is a common way of thinking about accuracy, it is easy to calculate, and it is easy to explain. The negative of this measurement is that for imbalanced datasets (datasets where the instances belonging to one or more classes far outnumber the instances in the other classes) it can give the impression of having constructed a better model than what is actually built. For example, imagine a dataset with two classes. In the dataset 95% of the instances belong to class one, and 5% belong to class two. A very simple, and very accurate classifier simply classifies all instances as class one! Even though this model is trivial and tells us very little about the data, reporting regular accuracy can make it seem like a good model.

Class balance accuracy is one type of accuracy measurement specifically designed to handle imbalanced datasets (Mosely 2014). The benefit of class balance accuracy is that it does not reward models for favoring one class over another. This can be an ideal measurement to maximize when the desired outcome of instance selection is a model that values each class of the dataset. The downside of this measurement is that what is being measured is not immediately intuitive. A high level explanation is that class balance accuracy reports the minimum of the average recall and the average precision, where average recall and average precision summarize how the classifier performs on average for each class.

3.5.6 How to combine columns for a final selection of instances

Due to the fact that the column generation procedure does not reach the true optimal selection of instances, it has been found to be beneficial to implement some heuristic to further combine columns into a final training dataset. Only one combination procedure is considered in this dissertation, but others could be easily implemented. The final combination procedure follows a greedy algorithm where the highest training accuracy column is added to the final training dataset and the column that most improves the accuracy of final classifier is added next, and so on until no improvements can be achieved.

The positive of the greedy method is that it is very straightforward. The negative is that a slightly more involved heuristic may be able to achieve better testing accuracy while avoiding overfitting.

3.5.7 The number of columns from which to estimate reduced cost in the POP, *R*

Both of the methods used to estimate the accuracy of a classifier require a subset of *J'* to make calculations. For the method based on frequency a subset of *J'* is taken in an attempt to isolate the helpful instances of the training data. For the method that is based on information theory a subset of *J'* is taken to introduce a level of volatility for calculations. Meaning, to make sure removing a column of instances from the training data will have an affect on classifier accuracy.

In both cases, it is not clear exactly how many columns should be considered. It seems helpful to think of the number of columns as the number of columns required to represent *R*% of the original training data by the union of their contents. This reformatting is convenient when trying to make recommendations for different datasets and different methods of constructing the initial *J'*, which intrinsically result in columns containing a varying number of instances. Preliminary testing showed 25% to be a good value for *R*. A higher and lower value should also be considered.

For both methods the hope of using a lower value will be to truly isolate the helpful instances of the dataset. For the method based on information theory another benefit may be the increase in volatility. Meaning, the removal of a single column from the training data may dramatically affect the accuracy of a classifier, resulting in stronger opinions about the columns. As long as these opinions are correct, the helpful and harmful instances should be clearly identified. The downside of the small size may be that the number and variety of ranked instances will not be large enough to find improving columns in the price out problem.

Conversely, the hope of a higher value will be to obtain an opinion on a large number of instances. The benefit of this for both methods would be the increased number of instances for which an opinion of usefulness is known. The downside of including a large number of columns may be that wrong conclusions are formed about particular instances (skewed by low accuracy columns) or the sheer number of columns may mute conclusions about the instances.

3.5.8 The number of instances the generated columns in the POP can include, *G*

In the price out problem the evaluation of reduced cost is estimated as a linear function. For both techniques of estimating reduced cost, only the helpfulness of the instances is identified (do to the scaling of the instances between one and zero). The result of both of these conditions is that the objective function drives the price out problem to include as many instances in a new column as possible. The only restraints preventing the inclusion of every instance in a new column are the dual variables. Initial experimental results showed the creation of very large columns that rarely had accuracy different than that of the original classifier. Too many instances were typically included. Therefore, the ability to create very large columns is taken away from the price out problem by placing an arbitrary constraint on the number of instances allowed in a newly generated column, called *G*. Two possible values of *G* are discussed along with their positives and negatives.

The first value of *G* comes from preliminary experiments where the size of newly generated columns was constrained to be less than the size of the very best column in the initial *J'*. This seemed to be a natural size to select because any improving columns generated in the POP would be similar in size to the initial subsets. This would make comparisons between new and old columns fair because the major differences between the two would simply be the instances contained in the column. Any improvement in column accuracy could then be attributed to a better selection of instances. However, it may be possible to find even better columns by allowing *G* to be larger than the best column in the initial *J'*, as shown through the greedy selection procedure used to make the final selection of instances.

The second value of *G* comes from the greedy procedure used to combine the columns to find a final selection of instances. Preliminary results show that the greedy procedure typically includes more than one column in its selection, and as a result improves upon the accuracy of the previously best column found. This is an indication that the created columns do not yet contain enough instances. Therefore, for the second value of *G,* the size of the improving columns found in the price out problem are constrained to be less than or equal to the number of instances found in a greedy selection of the initial columns of *J'*. This value could work well because the greedy selection shows us that the columns in the initial *J'* are too small. However, this value could also work poorly because it may allow the POP to include unnecessary instances due to its larger size.

3.5.9 The number of columns from the POP to check for positive reduced cost, *K*

The optimal solution of the price out problem is a newly generated column that is estimated to have positive reduced cost, and may improve the solution of the RMP if included in *J'*. Because this is only an estimation of reduced cost, the column should be checked to see if it truly does have positive reduced cost. If it does, it certainly should be included in *J'*. This check is performed by subtracting the sum of the dual variables from the true accuracy of the generated column.

Of course, it is cumbersome to obtain a single solution from the POP and then calculate its reduced cost. This is especially true when the likelihood of finding a positive reduced cost column has been observed to be low. Instead, it is recommended that *K* columns with high estimated reduced cost be checked for truly positive reduced cost at one time. Usually, obtaining these *K* solutions is trivial because a large number of feasible but sub optimal solutions are generated in the branch and bound tree used to solve the POP. The positive reduced cost columns are then treated differently depending on the RMP.

In RMP2 all of the *K* columns that have been found to have positive reduced cost are added to *J'*. Any positive reduced cost column represents a column with greater accuracy than the previous best column. These are obviously good columns to keep. In RMP1, only the column with the greatest positive reduced cost is added to *J'*. The reason for the difference is that in RMP2, the only column that matters in regard to the calculation of reduced cost is the column with the greatest accuracy. Including columns with lower, but still positive reduced cost will not cause the most positive reduced cost column in the *K* solutions to change. The same cannot be said for RMP1. Therefore, only the highest positive reduced cost column is included so that RMP1 can take the greatest step toward improvement.

In preliminary experiments it was determined that *K* equal to 2,500 results in the discovery of improving columns for both RMP1 and RMP2. Discussed are the potential benefits and pitfalls of smaller and larger values.

For RMP1 the potential benefit of a smaller value may be that intermediate improving columns are found. Meaning, if a larger number of columns are checked, the highest positive reduced cost column found when checking a smaller number of columns may be overshadowed. This column, a potentially beneficial building block in the search for the best subset of instances, may be ignored. The negative of checking this smaller number of columns is that it is possible that not enough columns will be checked to find any truly positive reduced cost columns. When checking the smaller number of columns, the column generation procedure may not lead to the creation of any improving columns.

The intent of checking a larger number of columns for positive reduced cost is to try and ensure that enough columns are checked to find an improving column, for both RMP1 and RMP2. The negative of course is that for RMP1 the intermediate columns sought after through the lower setting will be overlooked. However, the benefit may ultimately be columns that are even better elements in the search for the best subset of instances.

3.5.10 The number of restarts allowed in the POP

A restart, simply stated, is a way to continue the search for an improving column if no improving column was found after checking *K* columns of the POP. Due to a limited amount of time available to perform the experiments in this dissertation, the number of restarts allowed in the price out problem does not vary from three, the value used in initial testing. However, the concept of a restart and the perceived benefits of using them are discussed for each RMP.

For RMP2, a restart is performed by adding the previously generated columns to the POP as cuts and then solving for *K* new columns. The cuts make re-generating a column previously checked for positive reduced cost infeasible in the POP. This is acceptable because only columns that have higher accuracy than the current best column will ever have positive reduced cost. Once verified that this is not true for a column, the column never has to be reconsidered. Figure 1 shows the solution procedure for RMP2 with restarts.

The benefit of a restart with RMP2 is an artifact of the solution procedure. Adding cuts to the POP seems to make the branch and bound tree more complex, and columns obtained after adding the cuts seem to be more diverse. This could be because the cuts force the POP to consider columns from different areas of the solution space

A restart is performed for RMP1 by checking *K* additional columns for positive reduced cost. In this formulation it is not possible to add previously checked columns as cuts because previously checked columns are not guaranteed to always have negative reduced cost. This means that the perceived benefit of adding complexity to the branch and bound tree is not present. Still, the step procedure does allow improving columns to be found that could have otherwise been overlooked. Figure 2 shows the solution procedure for RMP1 with restarts.

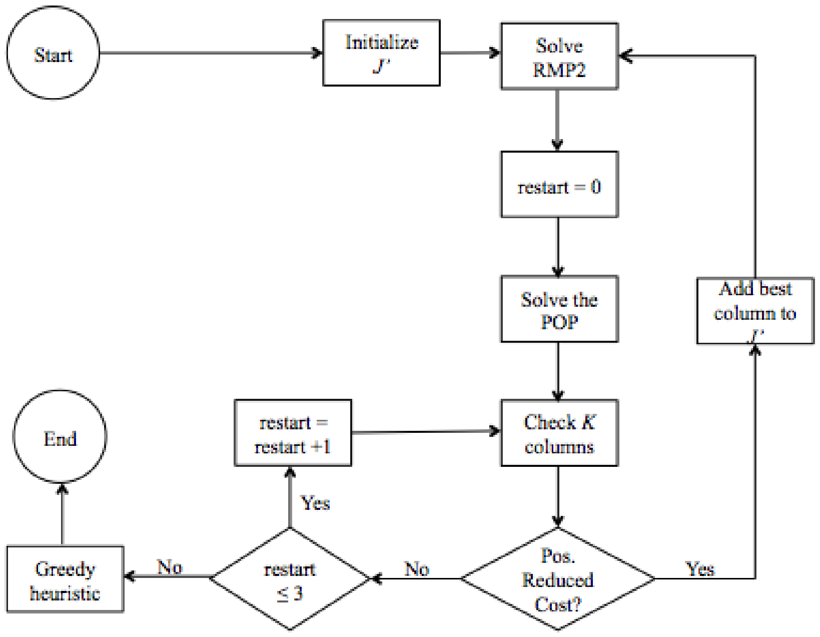

Figure 1. Solution procedure RMP2

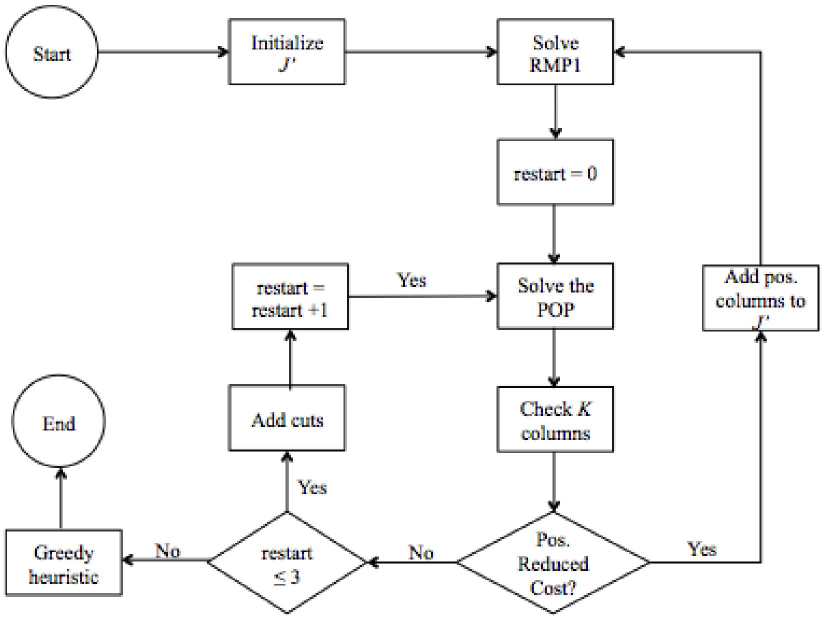

Figure 2. Solution procedure for RMP1

CHAPTER 4. EXPERIMENTAL RESULTS

This chapter presents results of instance selection applied to 12 experimental datasets. Included is an analysis of the results, focusing specifically on the reasons for successful and unsuccessful instance selection. Next is a discussion of what instance selection does to different datasets to achieve an improvement in classifier accuracy. Finally, instance selection's performance is compared to support vector machines, random forest, and AdaBoost.

4.1 Experimentation

In this section instance selection is applied to 12 experimental datasets. The datasets were selected such that instance selection could be exposed to a wide variety of classification problems. Meaning, the datasets differ in their number of instances, their number and types of attributes, their number of classes, and the difficulty of the classification task. The datasets were obtained from the UCI machine learning data repository and from "The Elements of Statistical Learning". Table 1 summarizes each of the datasets.

Name	Instances	Attributes	Numeric	Nominal	Classes
Balance	625	4	Yes	No	3
Credit	690	15	Yes	Yes	2
Diabetes	768	8	Yes	No	2
Ecoli	336	7	Yes	No	8
Glass	214	9	Yes	No	6
Horse	368	21	Yes	Yes	3
Ionosphere	351	34	Yes	No	2
LandSat*	444	36	Yes	No	6
Spect	267	22	No	Yes	2
Tic-Tac-Toe	958	9	No	Yes	2
Waveform	800	21	Yes	No	3
Wisc. Breast Cancer	699	9	Yes	No	2

Table 1. An * indicates that the dataset is a subset of the original, in that the numbers of instances are reduced.

4.1.1 Experimental design

For the experimental design, each dataset has instance selection applied with the naïve Bayes classifier (NB), logistic regression classifier (LR), and the decision tree classifier (DT). 20 replications of instance selection are performed for each dataset classifier combination. Additionally, in each replication the dataset is randomly split into a two third training and a one third testing dataset so that the testing accuracy of the classifiers can be compared before and after instance selection.

The application of instance selection to the experimental datasets follows rules of thumb that are developed in Chapter 5. Specifically, RMP2 is used, the initial *J'* is constructed with user defined random columns (see Chapter 5 for details on defining random columns), reduced cost is estimated by ranking instances based off of the frequency of their appearances in the high accuracy columns of *J'*, regular accuracy is used an accuracy measure, greedy selection is used as a final step to combine the generated columns into a final selection of instances, *R* is equal to 25%, *G* is equal to the number of instances in a greedy selection over the initial *J'*, *K* is equal to 10,000, and three restarts are allowed in the POP. Also, classifiers were constructed with the machine-learning library Weka using default parameter values, and Cplex12.5 was used to solve all mathematical models.

4.1.2 Results

Figure 3 shows the difference in testing accuracy before and after instance selection for the 12 experimental datasets. A paired t hypothesis test, testing for a difference in mean accuracy with 95% confidence was performed for each dataset classifier combination. Combinations with significant differences before and after instance selection are colored blue. Positive values on the x-axis indicate an improvement in accuracy after instance selection.

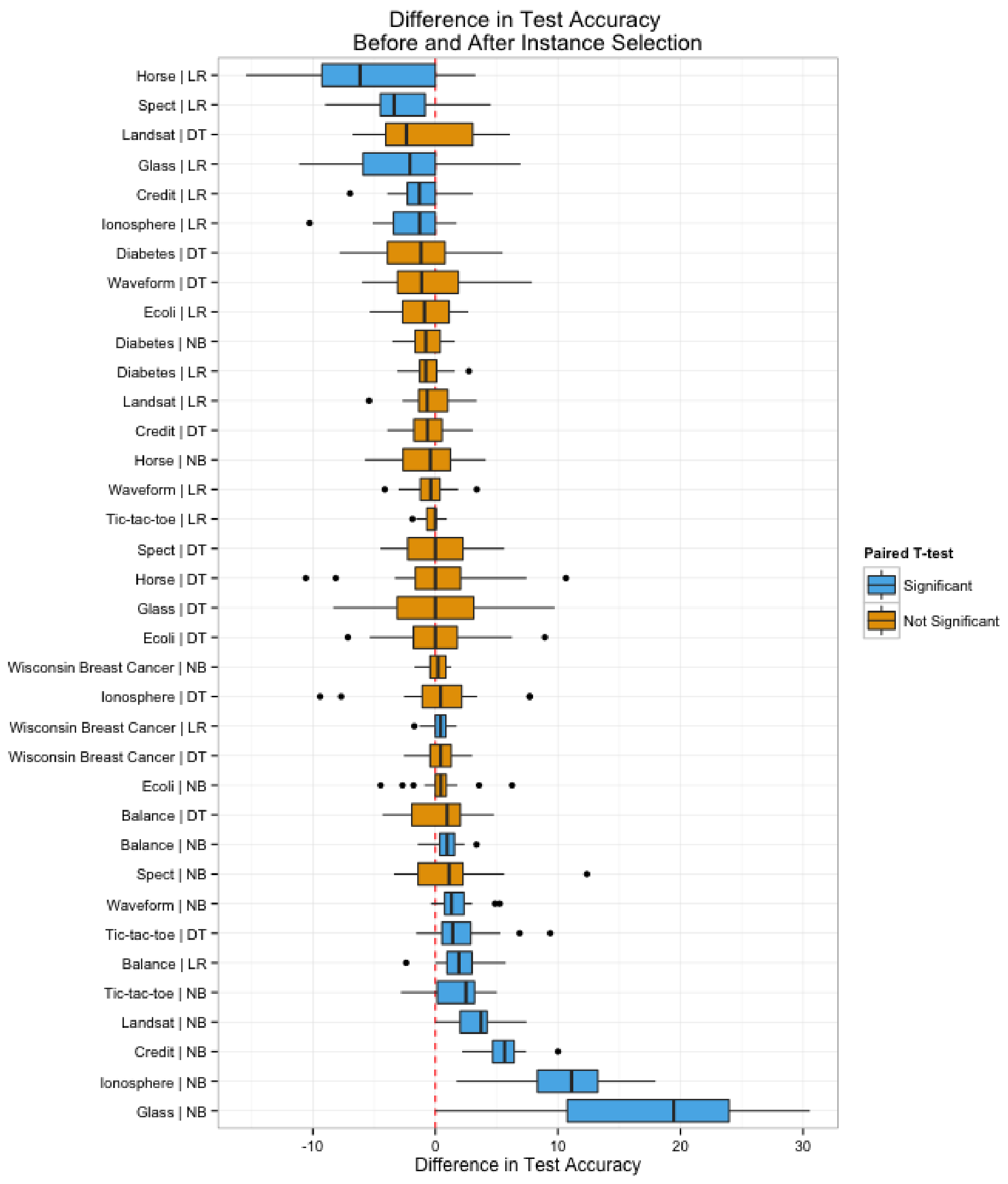

Figure 3. The difference between testing accuracy before and after instance selection. Positive values indicate an improvement from instance selection. Tables of the results can be found in Appendix A.

4.2 Analysis of results

This section will address why the success of instance selection varies for the different dataset classifier combinations. Specifics of why instance selection can lead to an increase in classifier accuracy are reserved for Section 4.3.

4.2.1 Overfitting

Instance selection manipulates a classifier’s training data in an effort to maximize training accuracy. A danger of this type of manipulation is overfitting the classifier to the data, and as that implies, losing the ability to correctly label unclassified instances. In that light, logistic regression and decision trees may not benefit from instance selection (as Figure 3 suggests) for some datasets, because they may be prone to overfitting the original data. In fact, for decision trees, overfitting is a common and well-known concern. In such situations, instance selection with the goal of maximizing training accuracy will only exacerbate the overfitting.

Consider Figure 4. This figure shows the training and test accuracies of a logistic regression classifier before and after instance selection for the Glass, Horse, and Ionosphere datasets. Recall from Figure 3 that these three dataset classifier combinations are adversely affected by instance selection. Yet, when looking at the training and test accuracy of the model built from the original training data, it is clear that logistic regression already overfits the data. It would be foolish to think that applying instance selection to maximize the training accuracy would do anything but make the overfitting worse.

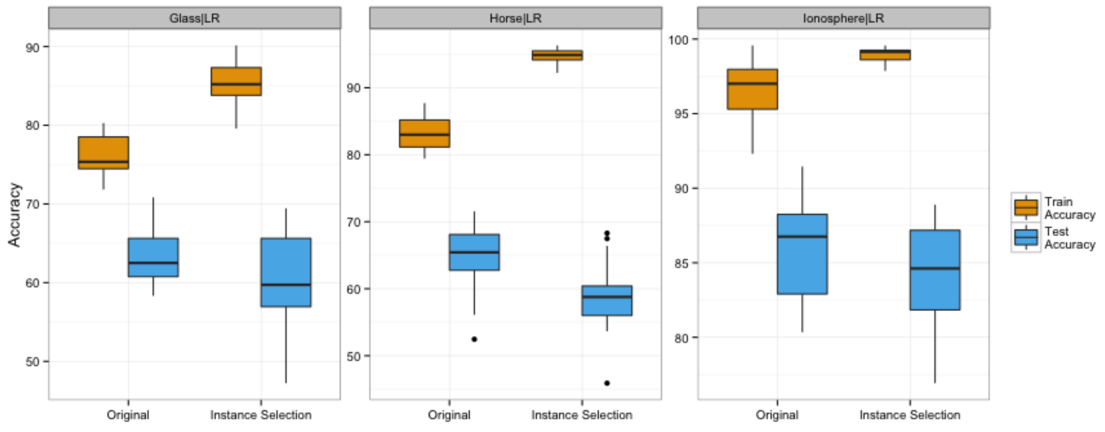

Figure 4. Logistic regression obviously overfits the original dataset.

Now, consider Figure 5. This figure shows the training and test accuracy of a naïve Bayes classifier before and after instance selection for the Balance, Credit, and Waveform datasets. Here, overfitting does not seem to be an issue, and instance selection is indeed helpful, as indicated in Figure 3.

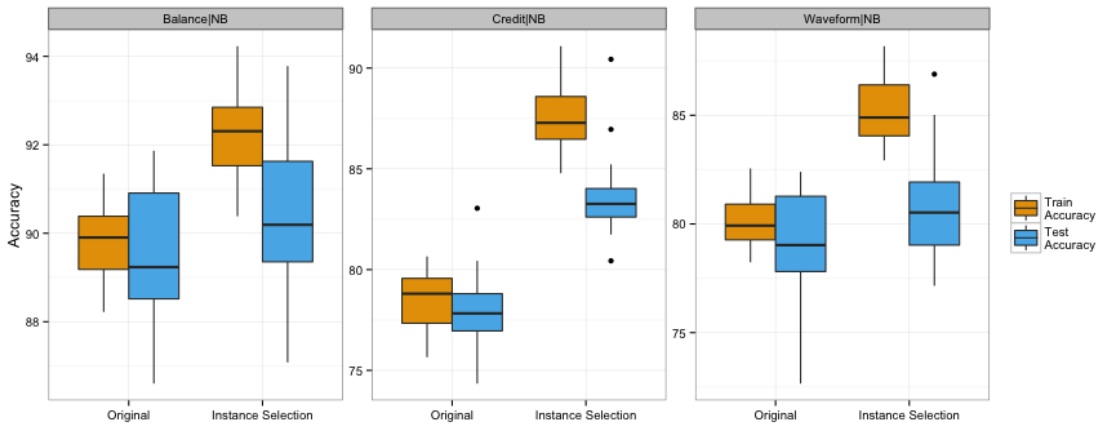

Figure 5. Naive Bayes is obviously not overfit to the original dataset.

A related observation is as follows. The Glass, Ionosphere, Landsat, and Waveform datasets, datasets with strictly numeric attributes, benefit from instance selection with the naïve Bayes classifier. For the same datasets, instance selection with the logistic regression classifier does not lead to a significant improvement. This disparity makes sense when considering the strengths and weaknesses of the two classifiers.

The naïve Bayes classifier assumes a distribution for numeric attributes to calculate prior probabilities. Because of this naïve Bayes may achieve a poor fit for datasets with strictly numeric attributes. However, instance selection could help discover beneficial structure to make up for this weakness. Conversely, logistic regression should work well for datasets with numeric attributes and it is possible that instance selection has little to contribute. A good fit, or an overfit, has probably already been achieved. This is further indicated by the lackluster performance of instance selection with the logistic regression classifier for the Glass, Ionosphere, and Landsat datasets.

A good example of differences in fit for different classifiers on the same dataset is the Glass dataset. The training and test accuracy before and after instance selection for both the logistic regression and naïve Bayes classifiers are shown in Figure 6. Here it can be seen that the original median test accuracy with the naïve Bayes classifier is very low. After instance selection the median test accuracy is comparable to the accuracy achieved by the original logistic regression classifier. However, instance selection with the logistic regression classifier does not lead to a significant improvement, perhaps because a good fit has already been achieved. Still, instance selection has enabled the naïve Bayes classifier to achieve the classification accuracy of logistic regression. What instance selection has changed to allow this increase is not immediately apparent, although it should be noted that the same increase in accuracy for the naïve Bayes classifier is achievable through attribute selection.

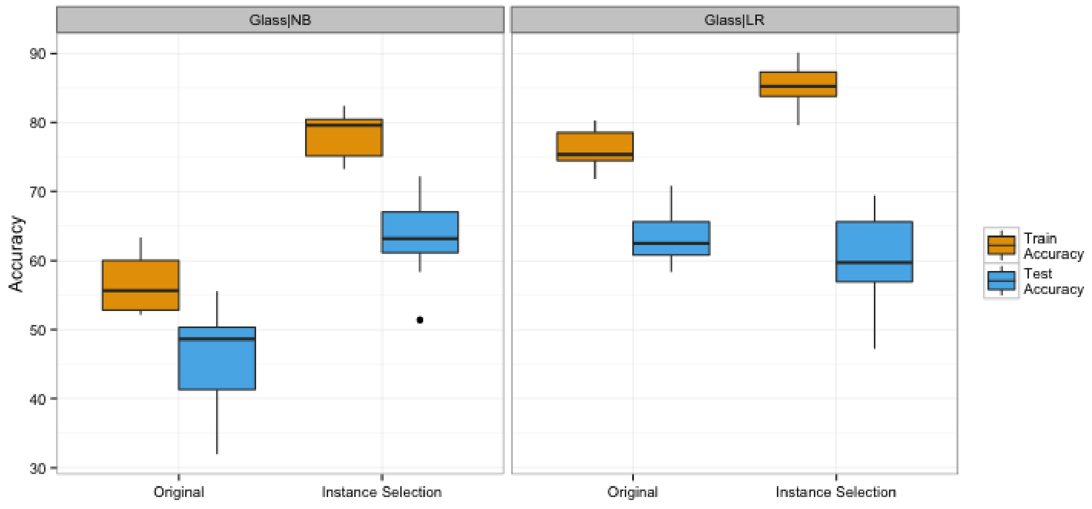

Figure 6. Instance selection on the Glass dataset using naive Bayes and logistic regression.

Overall, it is believed that when instance selection performs poorly, a structure, or bias, is imposed on the original training data that does not truly exist. Even though the intention of instance selection was to allow the classifier to take advantage of some obscured bias or relationship, it is possible that at times this discovery is exaggerated. In these cases, the structure is clearly not as strong in the true classification problem. This is evidenced by the classifier's poor performance on the testing dataset. Overfitting through instance selection can potentially be avoided by not performing instance selection with classifiers that have obviously already overfitted the original training data.

4.3 Outliers, overlapping classes, and minority classes

This section is devoted to examining exactly what instance selection is doing to improve the training accuracy of a classifier. In Chapter 1 it was noted that numerous aspects of the training data could make it difficult for a classification learner to induce an accurate model. These troubles included class values that overlap one another, one or more classes with outliers, and minority class values that are easily ignored by the learner. Depending on the dataset and classifier, it has been observed that instance selection addresses one or more of these issues in the training data. However, exactly how instance selection addresses each of these issues depends on the classification problem. This section examines each aspect more closely.

4.3.1 Outliers

Outliers in a dataset may be detrimental to a classifier because they can draw the attention of the classifier away from the bulk of the instances. Stated differently, a classifier that must consider outlying instances may miss discovering very helpful relationships that apply to the majority of the instances. As a result, the classifier accuracy may be less accurate than if it had tried to solely fit the non-outlying instances. The process of instance selection allows this possibility to be indirectly checked by considering subsets of the original training data.

In the experimental results presented in this chapter it has been observed that instance selection treats outliers differently for different dataset classifier combinations. To discover this, outliers were defined to be instances that had at least one numeric attribute value that was more than two standard deviations away from the mean attribute value of all the instances. The percentage of outliers in the training data was recorded before and after instance selection. Note, simply the count of outliers would be uninformative because instance selection reduces the size of the dataset.

Results show that instance selection with the naïve Bayes classifier for the Glass dataset leads to a significant decrease in the percentage of outliers, as determined by a paired t-test with 95% confidence. The median percentage before instance selection was 26% and after instance selection was 24%. Alternatively, for the Ionosphere dataset with the naïve Bayes classifier there was actually a significant increase in the percentage of outliers. Here the median value increased from 41% to 45%.

The reason that the treatment of outliers differs from dataset to dataset is that instance selection is treated as a wrapper. This means that feedback from the dataset is used to decide if removing an outlier is beneficial. Obviously for the Glass dataset it is beneficial to remove some outliers. However, for the Ionosphere dataset it is not. A case is discussed below that shows why the removal of outliers may be beneficial.

The goal of the Credit dataset is to classify customers as having either good or bad credit. For this illustration the data is split into a training and test dataset, where the training data contains 204 examples of good credit and 256 examples of bad credit. Here, instance selection can improve the test accuracy of the naïve Bayes classifier by two percentage points. This improvement is due to some change in the boundaries of the training data, and in this case the notable difference between the original training dataset and the selected training dataset is that instances with extreme values are removed. This is possibly the case because the classifier sacrifices the ability to classify extreme instances in exchange for obtaining better general knowledge about the dataset.

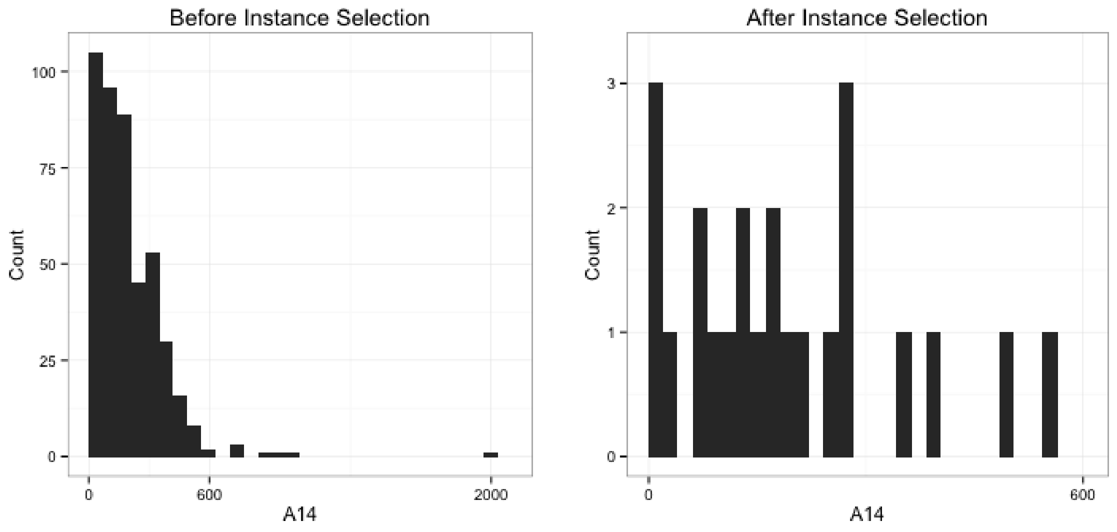

Figure 7. Variable A14 of the Credit Approval dataset before and after instance selection. Note the reduced range of the variable.

Specifically, one variable that has extreme values is A14. Figure 7 shows the ranges for this variable before and after instance selection. It is evident that some instances with extreme values are not included in the subset of selected instances. This observation will be found interesting below when considering how decision tree learning uses the variable A14. The effect of excluding extreme values (of A14 and other variables) is clearly seen in the decision trees learned on the original data versus the selected data. The decision tree learned on the original data is very complex as shown in Figure 8.

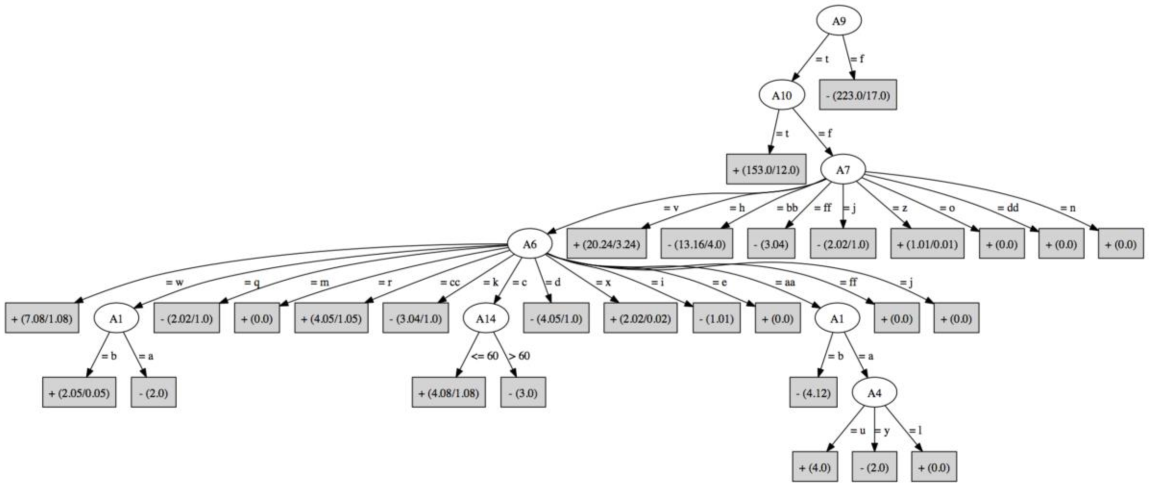

Figure 8. Decision tree learned from all of the Credit training instances.

After applying instance selection the decision tree becomes much simpler (with slightly higher accuracy), but most notably the variable A14, with its truncated ranges, is now used earlier in the decision tree. This results in a more interpretable, yet still more accurate model:

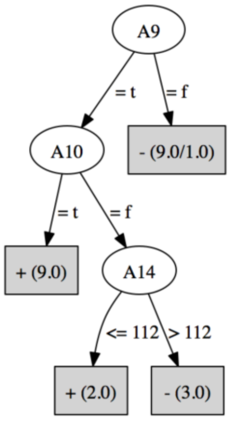

Figure 9. Decision tree learned from the selected instances of the Credit dataset.

In the original decision tree the first two branches are used to completely classify 376 instances and the remaining branches are used to split the 84 remaining instances into homogenous groups. Looking at the new decision tree in Figure 9 it can be seen that a split on variable A14 is used to replace the original tree structure and a branching to make homogenous groups for the instance selection training dataset can quite easily be found. Figure 10 illustrates the easier to separate training data found through instance selection.

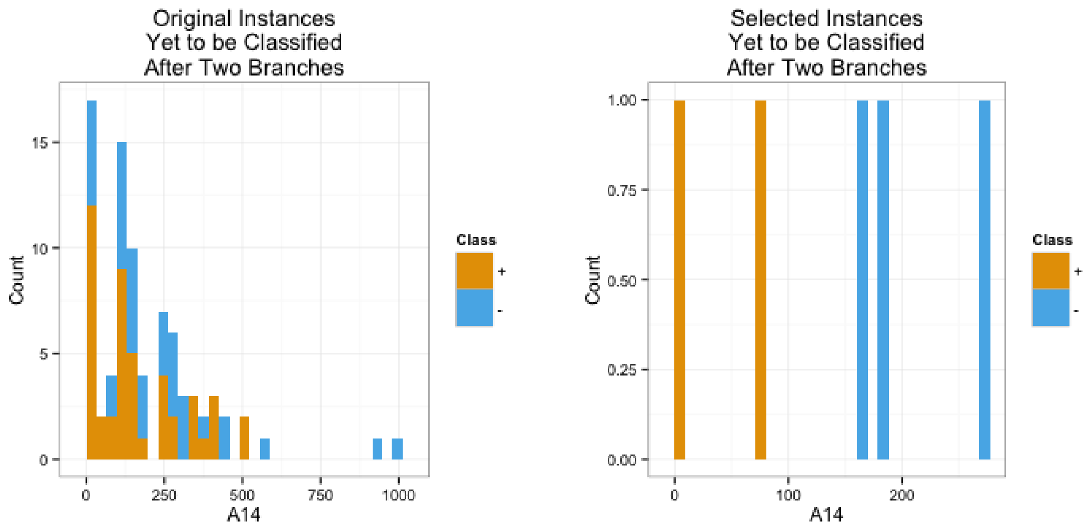

Figure 10. Note the easier to separate instances after the initial branches of the decision tree with instance selection.

In this example extreme valued instances are eliminated and as a result a clear boundary between instances with different class values is found. Classification ability is no longer sacrificed by fitting the classifier to the training dataset with extreme valued instances.

4.3.2 Overlapping classes

Classes that partially overlap, meaning classes that have some instances with similar attribute values, can make it difficult for a classifier to distinguish between one class and another. Because of this, it may sometimes be advantageous to remove similar instances and allow the classifier to learn from the data with clear boundaries between the classes. Instance selection allows for this possibility by considering subsets of the original training data. Again, instance selection simply chooses the subset with the highest accuracy, regardless if that is a subset with or without well-separated classes.

From the experimental results presented in this chapter it has been observed that instance selection treats the separation of classes differently for the different dataset classifier combinations. For datasets with all numeric attributes the treatment of overlap was quantified by the Dunn index using the R package clValid. The Dunn index is a cluster validation measure that evaluates the compactness and separation of clusters, where high values are an indication of better compactness and separation (Kovacs 2005). For the experimental datasets, instances were assigned to clusters based off of their class values, and the Dunn index was measured before and after instance selection. Of course, because instance selection removes instances, instance selection can never hurt compactness and separation, but some datasets improve in compactness and separation much more than others.

For the Balance dataset with the naïve Bayes classifier the median Dunn index before instance selection was 0.1, and after instance selection, the median Dunn index was still 0.1. Results for the Wisconsin Breast Cancer dataset with logistic regression were different, with the median Dunn index before instance selection being 0.1 and after instance selection 0.3. As with outlying instances, the importance instance selection places on increasing the separation between classes depends on the dataset and classifier. A case is now discussed that shows why the separation of classes may be beneficial for some dataset classifier combinations.

This example considers the Landsat dataset, where the classification task is to classify the pixels contained in an image. This version of the Landsat dataset is from preliminary experimentation and is actually a different subset of the full Landsat dataset that was presented in the experimental results of this chapter. Still, in this example, a logistic regression classifier is built and then improved. The classifier's testing accuracy improves from 66% for the original training dataset to 80% for the selected training dataset. To provide some insights into how such an improvement is obtained, multidimensional scaling is used to visualize the instance selection. Figure 11 shows the original training data plotted with respect to the top two principal components.

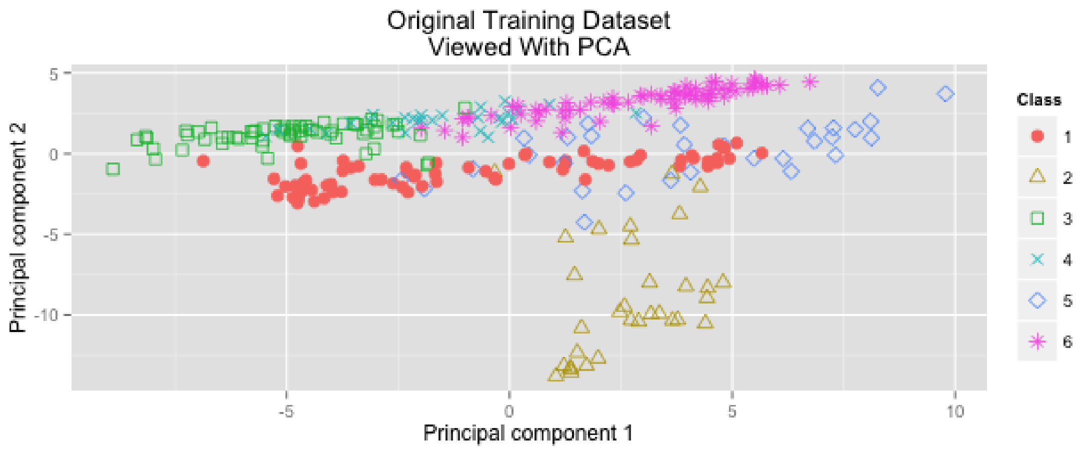

Figure 11. Training dataset visualized before instance selection. Class 4 appears difficult to separate.

It is evident that Class 4 instances are difficult to separate from the other instances as it spans the boundaries between Class 3 and Class 6. As it turns out, instance selection removed Class 4 altogether and modified the boundaries further as Figure 12 shows.

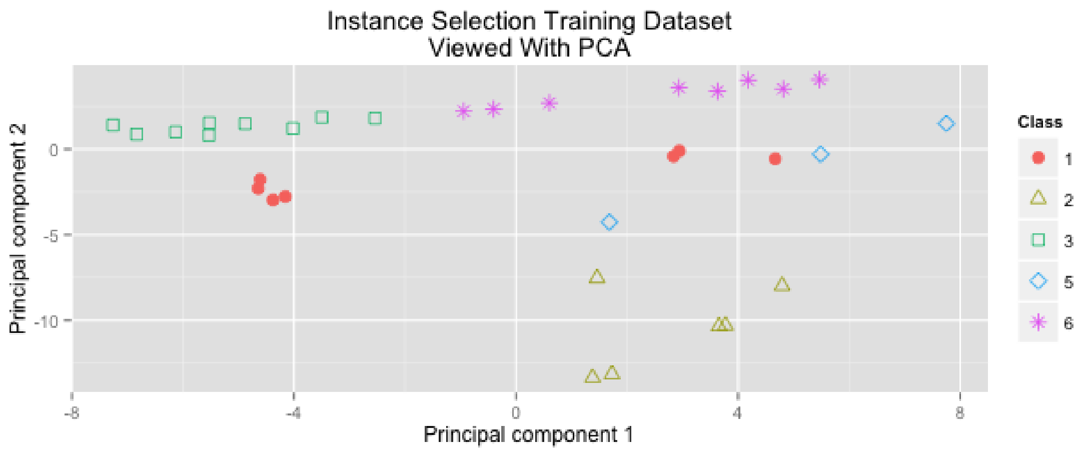

Figure 12. Training dataset visualized after instance selection. Class 4 was completely removed and the remaining classes are easy to separate.

After instance selection, Class 4 is ignored completely but there are now very clear boundaries between most of the remaining classes. (Note that a similar story was found by visualizing each pair of the four top principal components, which account for 90.9% of the dataset's variability.) Ignoring one class value to create better boundaries seems to be justifiable, as it appears almost impossible to capture its boundaries anyway. Table 2 shows the actual errors made by class values, both before and after instance selection.

	Number of Test Instances Misclassified (Test Data)						
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Total
Original Training Data	5	9	11	9	8	9	51
With Instance Selection	1	3	3	14	5	2	28

Table 2. Misclassifications for the Credit Approval dataset before and after instance selection.

It can quickly be observed that there are indeed more misclassifications made for Class 4, but it already had 9 out of 14 test instances misclassified so the change is relatively minor. The classifier built after instance selection improves the classification accuracy for all other class values and overall the errors are reduced from 51 to 28 instances. Thus, instance selection likely sacrificed the ability to classify the difficult minority Class 4 in an effort to make clearer and easier to learn boundaries for the remaining classes.

4.3.3 Minority classes

Minority classes are sometimes the important aspect of a dataset. For example, in Chapter 5 the goal of the second case study is to predict patient death as a result of some procedure. There, the interesting class is the minority. That is, the classification goal is to predict when patients do die from the procedure. However, there are other cases when overall accuracy is more important than being able to predict the minority classes. In cases like these, it may be advantageous to completely ignore the minority class and instead focus on doing a good job predicting all remaining classes. As before, instance selection can consider both alternatives by investigating subsets of the training data, and then can select the most favorable configuration.

Experimental results for the Glass and Ecoli dataset show the different ways instance selection can treat minority classes. Both of these datasets have multiple classes with one or more class values being severely underrepresented. The goals of the Glass and Ecoli dataset are to predict different types of glass and ecoli respectively. For the Glass dataset with the naïve Bayes classifier it is observed that all of the classes are preserved in the training data after every replication of instance selection. Recall from Figure 3 that instance selection does lead to a significant improvement in classifier accuracy for this dataset classifier combination. Alternatively, for the Ecoli dataset it is frequently observed that instance selection with the naïve Bayes classifier completely removes a minority class from the training data, and this does lead to an improvement of classifier accuracy. The reasoning for this improvement is likely similar to the example in Section 4.3.2**,** removing the minority class allows decision boundaries to be found that are clear enough to make up for the lost ability to classify a minority class.

4.3.4 Summary

This section has illustrated how instance selection can optimize the training data for two data examples. In the first example the extreme valued instances are removed from the training dataset and a better decision tree branching is found, indicating that there exists a clear separation of the classes. In the second example a difficult to predict class is completely removed from the training dataset and clearer boundaries between class values in the selected training dataset lead to a more accurate classifier. Instance selection appears to construct training datasets that have predictable boundaries with the intention that these clear boundaries lead to more accurate classifiers. However, it has been shown that the mechanism used to achieve the clearer boundaries depends on the dataset itself.

4.4 Comparative results

Support vector machines (SVM), random forest (RF), and AdaBoost are three classification algorithms known for their good performance on a wide variety of datasets. In this section instance selection will be compared to all three.

For each replication of instance selection (IS) a support vector machine classifier, a random forest classifier, and an AdaBoost classifier (using a decision stump as the base classifier) were constructed and evaluated. These classifiers were constructed using Weka with the default parameter settings**.** Figure 13 shows the comparative results.

Obviously no existing classifier is always going to have the best performance for every dataset, but the results from Figure 13 indicate that instance selection is a competitive classification approach. The accuracy achieved by instance selection is similar to some of best available classifiers. There are datasets where instance selection with one classifier or another appears to be the best choice, there are datasets where there is no clear best classifier, and there are datasets where one of the other methods is clearly superior.

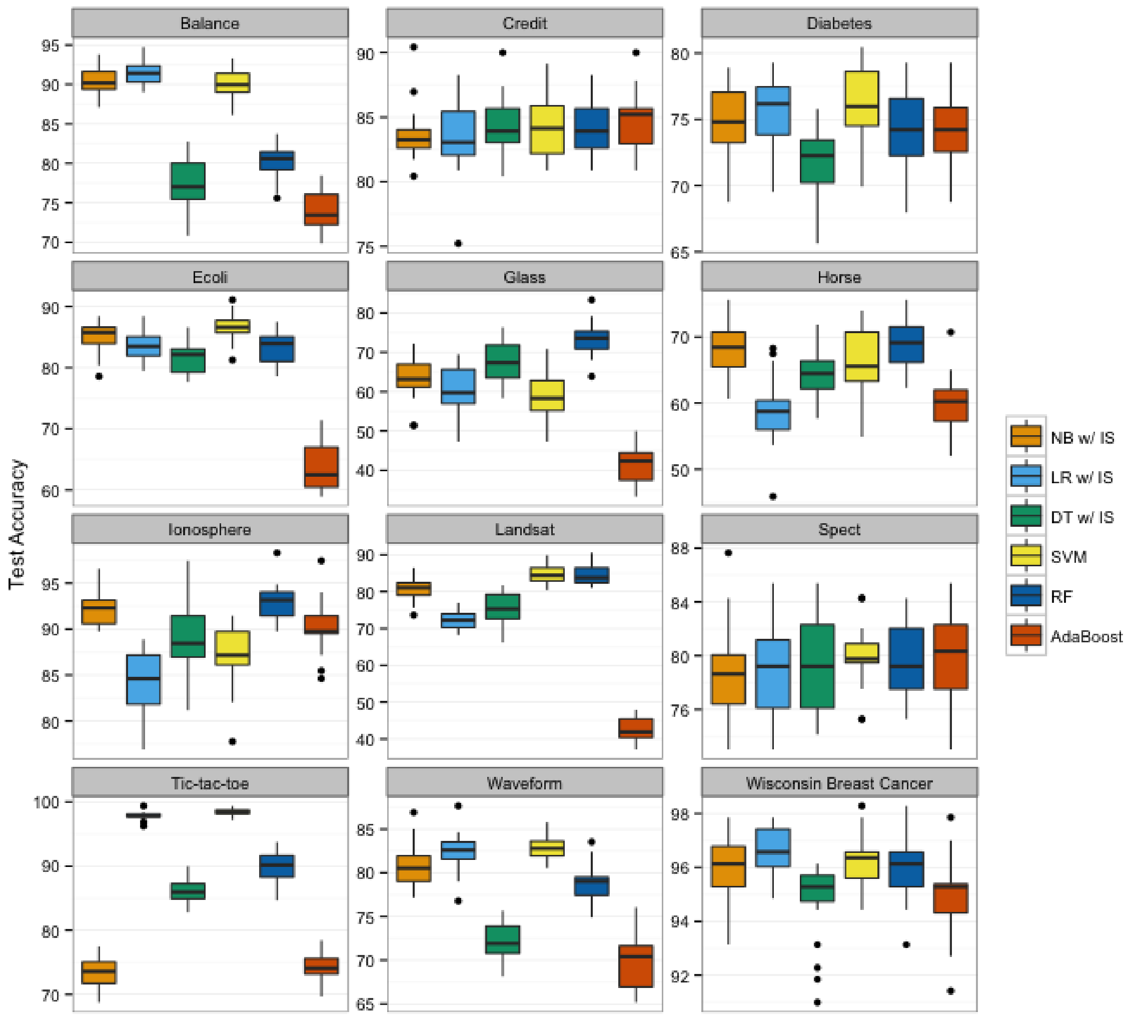

Figure 13. Comparative results of instance selection (IS), SVM, RF, and AdaBoost

It is worth noting that when selecting a classification technique it is important to remember the properties of the classifier. Consider the Glass dataset where random forest performs very well and instance selection with decision trees performs second best. Decision trees are clearly a good way to solve this classification problem. However, one benefit of instance selection with decision trees is that the result is more interpretable than random forest. That is, random forest leads to a multitude of trees and instance selection results in only one. If the goal of classification is to easily interpret the results, instance selection may become the preferred method.

Instance selection can help a classifier take advantage of structure in a dataset and improve its classification accuracy. Performing instance selection for a classifier should be considered when the base classifier is not obviously overfit to the original data, and when the desired outcome is a single and perhaps interpretable classifier.

CHAPTER 5. CASE STUDIES

The objective of this chapter is to show that integer programming techniques for instance selection can be applied to real world problems. To do so, two studies concerning cancer patients are presented and then have instance selection applied to them. The two studies analyze cancer patients in the Surveillance, Epidemiology, and End Results (SEER) database and construct classifiers to make predictions about their future or current health. This section includes a discussion about the tuning of the instance selection parameters, the application of the instance selection to the two data studies, and finally a discussion of what is learned about the parameters of instance selection.

5.1 Parameter tuning

Inherent to any classification method are a set of parameters that should be tuned to make the classifier perform well on a particular dataset. For example, with decision trees it is necessary to establish a stopping criterion for the construction of leaves, with support vector machines it is necessary to choose a cost for violating the separability constraint, and with k-nearest neighbors it is necessary to choose the number of neighbors used in classification. Instance selection is no different.

Until now, instance selection has been performed with parameter values set by rules of thumb. These rules were established because of the tuning experiments presented in this section. Specifically, tuning is performed by applying instance selection to the two case studies with every combination of the proposed parameter values. The best combination of parameters is considered to be the combination that leads to the highest training accuracy. Table 3 shows the values (or choices) proposed for each of the instance selection parameters. A complete description of the instance selection parameters can be found in Section 3.5.

Parameter	Values
Which classifier to use	Logistic regression
Which reduced master problem (RMP) to use	RMP1 or RMP2
How to construct the initial columns (*J'*) of the RMP	B0, B10, FB, RB or RU
How to estimate reduced cost in the price out problem (POP)	Information or frequency
How to measure accuracy	Case study specific
How to combine columns for a final selection of instances	Greedy selection
The number of columns from which to estimate reduced cost in the POP, *R*	10%, 25% or 50%
The number of instances the generated column in the POP can include, *G*	Size of the best column or the size of a greedy selection
The number of columns from the POP to check for positive reduced cost, *K*	500, 2,500 or 10,000
The number of restarts allowed in the POP	3

Table 3. Possible parameter values for instance selection tuning.

5.2 Case studies

The Surveillance, Epidemiology, and End Results (SEER) database is a program of the National Cancer Institute that collects and organizes information concerning cancer patients. The collected data concerns demographic information of the patients, the types of cancers the patients have, their prescribed treatments, and the subsequent outcomes of the treatments. The SEER database works with 19 different registries and has patient records from 1973 through today. In total, the SEER database covers approximately 28% of the United State's population.

The SEER database is vast and it allows researchers to analyze trends in healthcare as well as to identify best practices in cancer treatment. One interesting area of research using the SEER database are studies that utilize logistic regression classifiers to make predictions. In this sub section, two case studies from the SEER database that concern prediction using logistic regression are considered. It will be shown that instance selection can be used to help improve the predictive accuracy and/or usefulness of these classifiers.

The first case study of this dissertation redevelops the classifier by Bhattacharyya and Fried (2002) in "Nodal Metastasis in Major salivary Gland Cancer" and attempts to improve its predictive accuracy through instance selection. The goal of this classifier is to predict whether a patient identified as having major salivary gland cancer also has metastasis in regional lymph nodes. This prediction is based off of non-invasive tests, and the argument is that if such a prediction can be made then a patient can either be spared or prescribed a more invasive procedure. The second case study of this dissertation redevelops the classifier by Jeldres et. al (2009) in, "A Population-based Assessment of Perioperative Mortality After Nephroureterectomy for Upper-tract Urothelial Carcinoma", and again attempts to improve the classifier's predictive accuracy through instance selection. Here the goal is to predict which patients will not survive a Nephroureterectomy, based on demographic information about the patient and the histology of their tumor.

Discussed for each case study are the training dataset before and after instance selection, the logistic regression classifier before and after instance selection, and any insights from the instance selection process.

5.2.1 Nodal metastasis in major salivary gland cancer

Cancer of the major salivary glands is a relatively rare form of cancer that affects the parotid gland, the sublingual gland, and the submandibular gland. Due in part to its rarity, predictive factors of the disease and the quantification of patient survival are still in question. The population study by Bhattacharyya and Fried (2002), “Nodal Metastasis in Major Salivary Gland Cancer”, attempts to further the understanding of this cancer by building a multivariate logistic regression classifier based on preoperative information to predict the presence of nodal metastasis. Such a prediction is helpful because identifying nodal metastasis is necessary for the effective treatment of major salivary gland cancer, but checking for the condition involves an invasive procedure.

It has been established in this dissertation that instance selection can be used to select a subset of instances that when learned from, can result in a better classifier than if the classifier had been built from the entire dataset. For this study it is hoped that instance selection can identify a group of patients that allow multivariate logistic regression to clearly identify when a patient is positive for nodal metastasis. Any improvement in such a classification would certainly be welcoming to those that suffer from major salivary cancer, and would definitely contribute to the understanding of factors that can predict nodal metastasis from the disease. Following is a discussion of the study’s training dataset before and after instance selection, the logistic regression models before and after instance selection, and a discussion of what is learned from performing instance selection.

5.2.1.1 The application of instance selection

The study's dataset consists of patients diagnosed with major salivary gland cancer between 1988 and 1998 that have at least one regional lymph node sampled. Attributes included in the multivariate logistic regression are: the age at diagnosis, gender, histopathology, extraglandular involvement, grade, and tumor size. The outcome to be predicted is if any regional lymph nodes are positive for metastasis. Of the original 1,423 patients, 39% are found to be positive for nodal metastasis. For experimental purposes the dataset is split into two thirds training data and one third testing data, where the testing dataset is reserved for evaluating final classifiers only.

As discussed in Section 5.1 instance selection will be applied to the training datasets with extensive parameter tuning. The set of parameters that result in the selection of instances with the highest training accuracy will be used to construct the final dataset. The parameter values used in the tuning procedure can be found in Section 5.1, with the exception that regular accuracy will be used for evaluating classifier accuracy (because the training data is relatively balanced), and that 100 instances are included in the user defined random columns.

The number of instances included in the user defined random columns was determined by plotting the best accuracy of 1,000 random subsets for every possible column length, and determining where the rise in accuracy levels off. Figure 14 shows that the rise in accuracy levels off around 100 instances for this classification problem. It is interesting to note that the size of the initial columns created with the other methods include much less than 100 instances on average, but that their accuracy is similar to that of the random subsets.

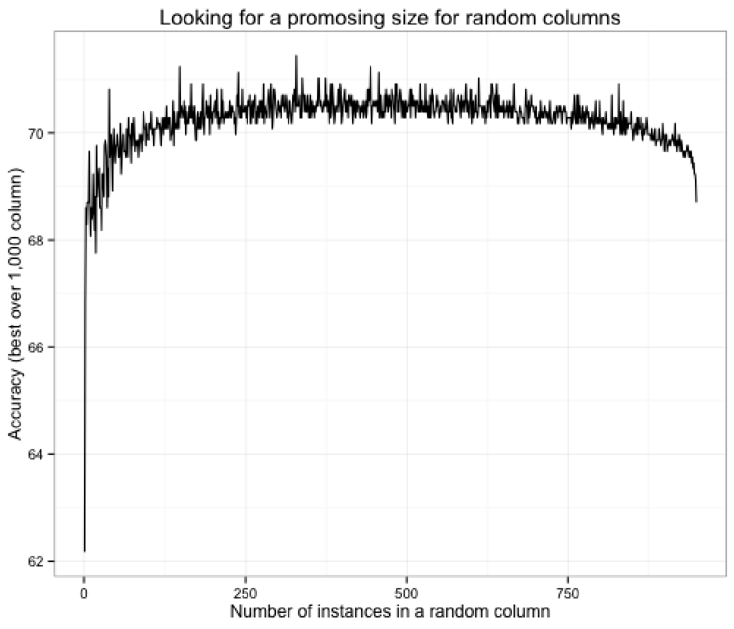

Figure 14. The rise in accuracy seems to level out when 100 random instances are included in a column.

After performing the parameter tuning for instance selection it was found that the best training dataset in regard to training accuracy was achieved by using RMP2, creating initial columns with a user defined number of random instances, estimating reduced cost with information theory, setting *R* equal to 10%, *G* equal to the number of instances found in the best initial column, and *K* equal to 10,000. Results from this data study and the next will be generalized into rules of thumb in Section 5.3.

Table 3 shows a break down of the training dataset before and after instance selection. The major difference in the training dataset after instance selection is that the proportion of male to female patients is closer to even, as is the split between having and not having extraglandular involvement. The other observable aspects of the data remain relatively unchanged. Interestingly, if the original training dataset is split into two classification tasks, the first being to predict nodal metastasis for men, and the second nodal metastatis for women, a compelling argument can be found for increasing the relative proportion of women in the training data.

When building a logistic regression classifier to predict nodal metastasis solely for men, the training accuracy is 63%, whereas when doing the same for women, the training accuracy is 75%. It makes intuitive sense that the logistic regression classifier may benefit from devoting more attention to the easier task, predicting nodal metastasis in women. The same observation can be made when considering the adjustment made to the proportions found in extraglandular involvement.

		Original (N = 949)		Instance Selection (N = 100)	
Variable		Mean	Range	Mean	Range
Age at diagnosis		57	5-94	58	5-88
Size		30	1-170	32	10-125
Variable	Level	No.	%	No.	%
Gender					
	Male	544	57%	52	52%
	Female	405	43%	48	48%
Histopathology					
	Acinar cell carcinoma	128	13%	13	13%
	Adenocarcinoma	125	13%	16	16%
	Adenoid cystic carcinoma	132	14%	16	16%
	Adenosquamous carcinoma	15	2%	1	1%
	Mucoepidermoid carcinoma	229	24%	18	18%
	Other	154	16%	17	17%
	Sarcoma	1	0%	0	0%
	Squamous cell carcinoma	165	17%	19	19%
Extraglandular					
	Yes	383	57%	47	47%
	No	545	40%	51	51%
	Unknown	21	2%	2	2%
Grade					
	Grade 1	81	9%	7	7%
	Grade 2	215	23%	24	24%
	Grade 3	236	25%	22	22%
	Grade 4	90	9%	9	9%
	Unknown	327	34%	38	38%
Nodal Metastasis					
	No	590	62%	61	61%
	Yes	359	38%	39	39%

Table 4. Training data before and after instance selection. Notice that the proportion of females increases after instance selection, as does the proportion of cases with no extraglandular involvement.

5.2.1.2 Multivariate logistic regression before and after instance selection

The multivariate logistic regression classifier built from the original training data had 68.7% training accuracy and 69.6% testing accuracy. The multivariate logistic regression classifier built from the training data after instance selection had 71.9% training accuracy and 70.0% testing accuracy. The confusion matrix from the testing data for each classifier is shown in Table 5 and Table 6. Here “No” represents patients that were not found to have distant metastasis, and “Yes” represents the patients that were. Notice that the classifier built through instance selection, shown in Table 6**,** makes more false positives than false negatives (a desirable property), unlike the classifier built without instance selection. Based on its higher testing accuracy and favorable types of error, the classifier built with instance selection is an improvement over the original classifier.

		Predicted	
		No	Yes
Actual	No	224	56
	Yes	88	106

Table 5. Confusion matrix for testing data with multivariate logistic regression classifier before instance selection.

		Predicted	
		No	Yes
Actual	No	220	60
	Yes	82	112

Table 6. Confusion matrix for testing data with multivariate logistic regression classifier after instance selection.

5.2.1.3 Instance selection insights

The major insight gained from performing instance selection is not directly related to the developed model. Instead, it is related to the selected instances. Noticing that the proportion of men to women changed in the selected training data led to the discovery that a logistic regression classifier can more accurately predict distant metastasis for women. A medical professional may decide that it makes sense to develop a model for both men and women. Multiple models could inspire more confidence in predictions for women. The same can be said for patients that exhibit extraglandular involvement.

5.2.2 A population-based assessment of perioperative mortality after Nephroureterectomy for Upper-tract Urothelial Carcinoma

Upper-tract Urothelial Carcinoma (UTUC) occurs mainly in the urinary bladder, ureters or renal pelvis. The standard of care to treat this disease is to perform a nephroureterctomy (NU), which is the surgical removal of the kidney, ureter, and may include the excision of the bladder cuff. The purpose of an NU is to eliminate the carcinoma and to prevent its spread (Jeldres 2009).

It is said in the population study by Jeldres et al. (2009), "A Population based assessment of perioperative mortality after Nephroureterctomy for Upper-tract Urothelial Carcinoma" that no model currently exists to predict individual or average perioperative mortality associated with NU. Even with perioperative mortality associated with NU being low, it is argued that such a model would be beneficial as a decision-making aide for at risk patients. Therefore, the SEER database is used to develop a multivariate logistic regression classifier to predict the 90-day survivability of patients who have UTUC treated with NU.

As with the study of subsection 5.2.1, it is hoped that instance selection can identify a group of patients that allow multivariate logistic regression to clearly identify when a patient is at risk of perioperative mortality associated with NU. As the original classifier for this problem simply predicts that all patients will survive (an artifact of the imbalanced data), an ability to truly identify patients at risk from the surgery would certainly be welcoming to those that suffer from UTUC. Following is a discussion of the study's training dataset before and after instance selection, the logistic regression models before and after instance selection, and a discussion of what is learned from performing instance selection.

5.2.2.1 The application of instance selection

The study's dataset considers patients diagnosed with UTUC and treated with NU between the years 2004 and 2010. The study's population includes patients over the age of 18 with available disease stage and no known distant metastases. The attributes considered in the model are the patient's age, gender, race, whether they had NU with or without the bladder cuff excision, as well as their tumor's grade, location, T stage, and N stage. The target or class attribute was death within 90 days of surgery for any reason. Of the original 2,328 patients, 9% are found to have died within 90 days of NU. For experimental purposes the dataset is split into two thirds training data and one third testing data, where the testing dataset is reserved for evaluating final classifiers only.

As discussed in Section 5.1 instance selection will be applied to the training dataset with extensive parameter tuning. The set of parameters that result in the selection of instances with the highest training accuracy will be selected to construct the final dataset. The parameter values used in the tuning procedure can be found in Section 5.1, with the exception that class balance accuracy will be used for evaluating classifier accuracy (because the training data is unbalanced and the original classifier ignores the minority class), and that 75 instances are included in the user defined random columns. It is important to note that class balance accuracy is chosen to evaluate instance selection for this problem because the desired outcome is a classifier that values each class equally.

The number of instances included in the user defined random columns was determined by plotting the best accuracy of 1,000 random subsets for every possible column length, and determining where the rise in class balance accuracy peaks or levels off. Figure 15 shows that the rise in accuracy peaks around 75 instances for this classification problem.

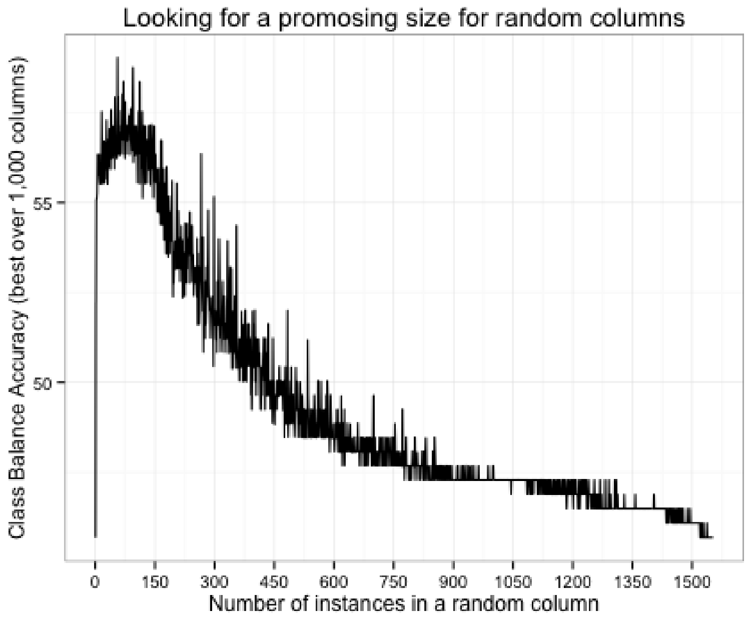

Figure 15. Class balance accuracy seems to peak when 75 random instances are included in a column.

After performing the parameter tuning for instance selection it was found that the best training dataset in regard to training class balance accuracy was achieved by using RMP1, creating initial columns with backward selection allowing a loss of ten percentage points, estimating reduced cost with information theory, setting *R* equal to 25%, *G* equal to the number of instances found in the best initial column, and *K* equal to 2,500. Results from this data study and the previous will be generalized into rules of thumb in Section 5.3.

Table 7 shows a break down of the training dataset before and after instance selection. Here it is observed that the proportion of male to female grows farther apart after instance selection. However, unlike in Section 5.2.1.1, no good guess is easily found for the difference. Additionally, the range of ages is reduced dramatically after instance selection. The new range of age at diagnosis covers 74% of the cases in the original training data. Perhaps the range is reduced to allow the classifier to focus its classification ability on the majority of cases, and to ignore cases with ages in the extremes.

		Original (N = 1552)		Instance Selection (N = 14)	
Attribute		Mean	Range	Mean	Range
Age at diagnosis		71.7	22 - 99	71.7	65 - 96
Attribute	Level	No.	%	No.	%
Gender					
	Male	901	41.9%	10	71.4%
	Female	651	58.1%	4	28.6%
Race					
	Caucasian	1352	87.1%	13	92.9%
	Other	200	12.9%	1	7.1%
Primary tumor location					
	Renal pelvis	1076	69.3%	10	71.4%
	Ureteral	476	30.7%	4	28.6%
Type of surgery					
	NU with cuff	1135	73.1%	10	71.4%
	NU without cuff	417	26.9%	4	28.6%
Grade					
	Grade 1	126	8.1%	1	7.1%
	Grade 2	366	23.6%	4	28.6%
	Grade 3	423	27.3%	3	21.4%
	Grade 4	637	41.0%	6	42.9%
T stage					
	Ta	389	25.1%	5	35.7%
	Tis	66	4.3%	0	0.0%
	T1	314	20.2%	1	7.1%
	T2	221	14.2%	1	7.1%
	T3	462	29.8%	6	42.9%
	T4	100	6.4%	1	7.1%
N stage					
	NO	1395	89.9%	10	71.4%
	N+	123	7.9%	2	14.3%
	Nx	34	2.2%	2	14.3%
Survived					
	Yes	1418	91.4%	12	85.7%
	No	134	8.6%	2	14.3%

Table 7. The training dataset before and after instance selection. Notice the range of age at diagnosis is reduced.

5.2.1.2 Multivariate logistic regression before and after instance selection

The multivariate logistic regression classifier built from the original training data had 45.7% training class balance accuracy and 45.2% testing class balance accuracy. The multivariate logistic regression classifier built from the training data after instance selection had 61.1% training class balance accuracy and 57.5% testing class balance accuracy. However, if only regular accuracy is considered it can be seen that testing accuracy reduces from 90.4% to 87.7% after instance selection. Still, it is argued that the classifier achieved through instance selection is more useful than the classifier achieved without.

Consider the confusion matrices from the testing data shown in Table 8 and Table 9. Here the "No" class represents a patient that did not survive an NU, and "Yes" represents patients that did. Notice that the confusion matrix for the classifier built before instance selection, shown in Table 8**,** simply predicts that every patient will survive an NU. This logistic regression classifier is uninformative about the data. However, consider the classifier built after instance selection. It is actually able to predict the death of some patients. If the goal of this classification was to actually identify patients at risk from an NU, the classifier built after instance selection is much more informative than the original classifier.

		Predicted	
		No	Yes
Actual	No	0	74
	Yes	0	702

Table 8. Confusion matrix for testing data with multivariate logistic regression classifier before instance selection.

		Predicted	
		No	Yes
Actual	No	18	56
	Yes	59	643

Table 9. Confusion matrix for testing data with multivariate logistic regression classifier after instance selection.

Table 10 holds the odds ratios associated with the classifiers built before and after instance selection. It is believed that because the selected training data contains so few patients that are at risk from an NU, the odds ratios have rather extreme values after instance selection. Also, some variables identified as important by the odds ratio stay the same before and after instance selection, for example, patients with T stage equal to T4, but some do change, for example age at diagnosis.

Variable	Odds ratio of "No" before instance selection	Odds ratio of "No" after instance selection
Age at diagnosis	1.0	17.4
Sex	1.3	2.5×10^{4}
Race	1.1	1.2×10^{9}
Primary tumor location	1.1	3.6×10^{8}
Type of surgery	1.3	2.4×10^{6}
Grade	1.3	0.0
T stage = Ta	1.3	1.2×10^{2}
T stage = Tis	0.4	1
T stage = T1	0.7	0
T stage = T2	0.8	0
T stage = T3	1.1	9
T stage = T4	2.4	5.9×10^{26}
N stage = NO	0.8	0
N stage = N+	1.1	0.2
N stage = NX	1.8	2.0×10^{11}

Table 10. The odds ratios from each classifier

5.2.2.3 Instance selection insights

Insights can be derived from applying instance selection to the NU survival problem. Specifically, a medical professional may be able to learn something about patients at risk from NU by analyzing the odds ratios shown in Table 10**.** Also, because the selected training dataset contains only two instances of patients not surviving the NU procedure, and fourteen patients total, it is possible that looking directly at these instances may reveal additional insight. After all, these fourteen instances are used as a substitute to represent the entire training data. It may be appropriate to consider the twelve surviving patients as ideal patients for NU, and the two non-surviving patients as non-ideal patients for NU. An example insight from the selected instances, as shown in Table 11**,** is that patients with T stage equal to T4 may be at high risk from NU. This is backed up by the odds ratios in Table 10.

Age	Gender	Race	Primary Tumor Location	Type of Surgery	Grade	Tstage	Nstage	Survived
87	Male	Cauc.	Ureteral	NU w cuff	2	Ta	N0	No
74	Male	Cauc.	Ureteral	NU w/out cuff	4	T4	N+	No
96	Female	Cauc.	Ren. pelvis	NU w cuff	4	T3	N0	Yes
65	Female	Cauc.	Ren. pelvis	NU w/out cuff	3	T3	N+	Yes
77	Female	Cauc.	Ren. pelvis	NU w/out cuff	4	T3	N0	Yes
69	Female	Other	Ren. pelvis	NU w cuff	3	Ta	N0	Yes
69	Male	Cauc.	Ren. pelvis	NU w cuff	2	T3	N0	Yes
66	Male	Cauc.	Ren. pelvis	NU w cuff	2	T3	NX	Yes
67	Male	Cauc.	Ren. pelvis	NU w cuff	3	Ta	N0	Yes
85	Male	Cauc.	Ren. pelvis	NU w cuff	4	T2	N0	Yes
80	Male	Cauc.	Ren. pelvis	NU w cuff	4	Ta	N0	Yes
80	Male	Cauc.	Ren. pelvis	NU w/out cuff	4	T3	NX	Yes
83	Male	Cauc.	Ureteral	NU w cuff	1	T1	N0	Yes
77	Male	Cauc.	Ureteral	NU w cuff	2	Ta	N0	Yes

Table 11. The selected training data for predicting death within 90 days of an NU.

5.3 Rules of thumb for instance selection parameters

To implement instance selection as described in this dissertation, parameter values need to be chosen. Section 5.1 described an experimental design to test a variety of values for each parameter and the two case studies implemented this design. From these results, rules of thumb for selecting parameter values are derived. Separate rules will be developed for using regular accuracy and class balance accuracy. The settings for class balance accuracy should be used only when the desired outcome of instance selection is a classifier that values each class equally.

Remember, different classification problems and goals will benefit from parameter values chosen specifically for that task. These rules of thumb are presented as good starting positions only.

Parameter values for using regular accuracy (RA) are discussed first. These values are derived from the experimental design implemented for the case study on nodal metastatis in major salivary gland cancer. Figure 16 shows that major differences in final training accuracy are noticed when considering how the initial columns of *J’* are constructed. Clear differences are not noticed in regard to any other single parameter.

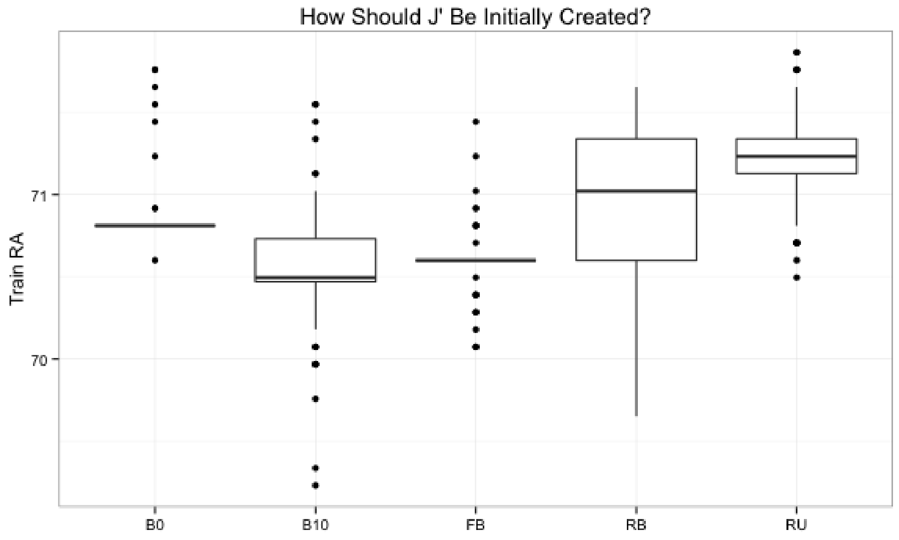

Figure 16. Training accuracy in regard to the initial construction of *J'*. RU seems to be the best choice for balanced data.

Now, if only the cases that consider user defined random columns (RU) as the initial *J’* are analyzed, other choices for parameter values become clearer. Figure 16 and Figure 17 are used to find that initial columns should be created with RU, RMP2 should be used, instances should be ranked according to the frequency of their appearances, *R* should equal 25%, *G* should equal the number of instances selected in a greedy selection procedure over the initial *J'*, *K* should equal 10,000, and regular accuracy should be used to evaluate classifier accuracies.

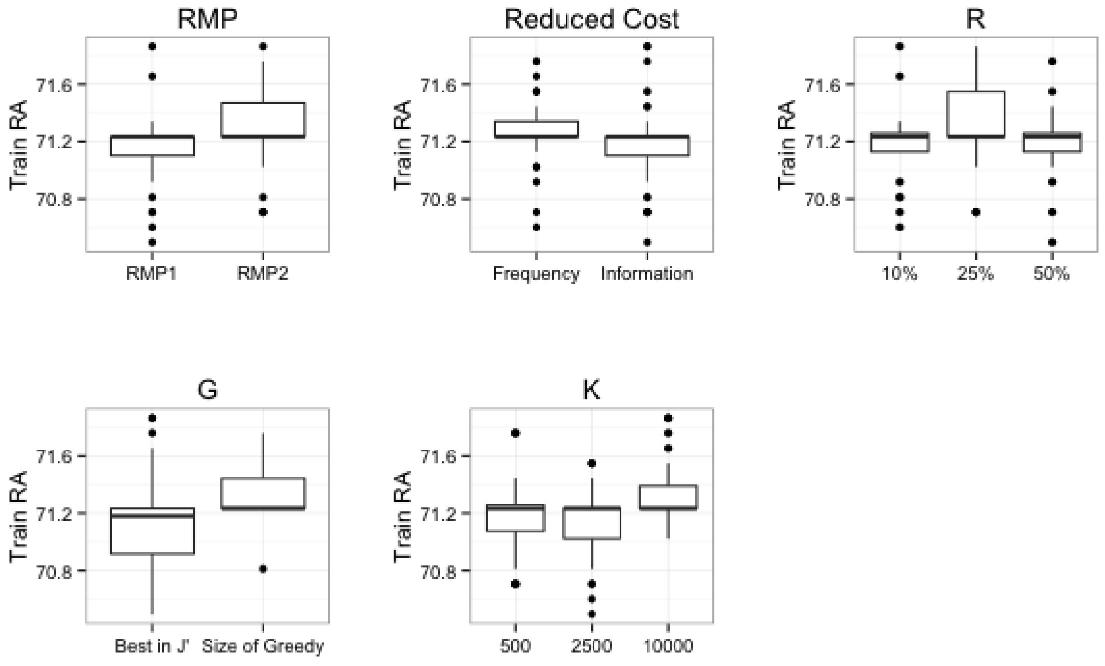

Figure 17. Boxplots to help pick rules of thumb when using regular accuracy.

It should be noted that the majority of these parameter values for regular accuracy make intuitive sense. For example, because the initial columns are constructed with a user defined number of random instances it is logical that the best instance ranking method would be based on frequency. This is because it is known that the initial columns of *J'* will be very diverse, and it seems intuitive that instances appearing frequently in the good random columns will be helpful. Additionally, it makes sense that newly generated columns should be allowed to grow to a size that has been found helpful in an initial greedy selection over *J'*. However, it is not intuitive why RMP2 seems to be the preferred RMP, but perhaps it is favored because the goal of RMP2 is to find columns that strictly improve upon the highest accuracy column of *J'* and these columns are useful in the final greedy selection procedure.

Now, parameter values for using class balance accuracy (CBA) are discussed. Again, these rules of thumb should be used when the dataset is imbalanced and the desired outcome is a classifier that values each class. These values are derived from the experimental design implemented for the case study on NU for UTUC. Figure 18 shows that some difference in final training accuracy is noticed when considering how the initial columns of *J'* are constructed. Clear differences are not noticed in regard to any other single parameter.

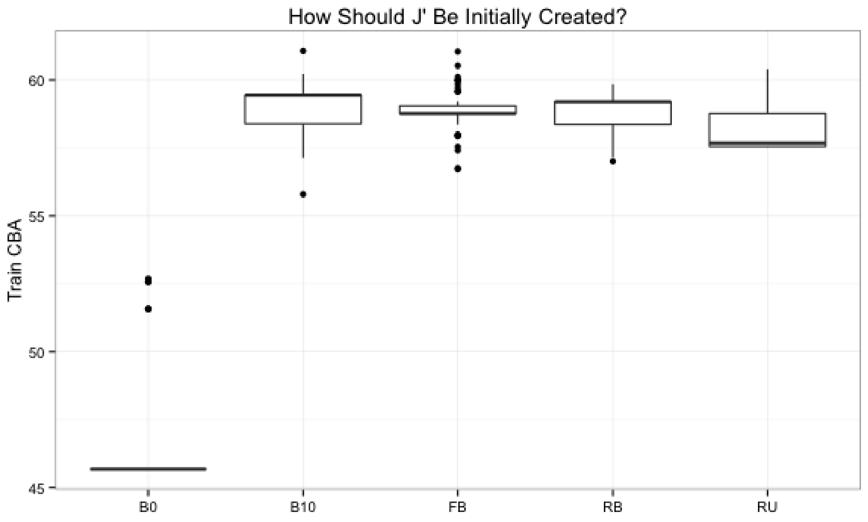

Figure 18. Training accuracy in regard to the initial *J'* creation procedure. B10 seems to be the best method for imbalanced data.

If only the cases that use an initial *J'* created through the backward greedy selection procedure with an allowed loss of ten percentage points (B10) are analyzed, other parameter choices become clearer. Figure 18 and Figure 19 are used to find that initial columns should be created with B10, RMP1 should be used, instances should be ranked with the information measure, *R* should equal 25%, *G* should equal the number of instances selected in a greedy selection procedure over the initial *J'*, *K* should equal 500, and class balance accuracy should be used to evaluate classifier accuracies. Obviously, the rules of thumb for class balance accuracy are different from the regular accuracy rules, specifically in the way the initial *J'* is created, the RMP used, how the instances are ranked, and what *K* should equal.

It is believed that the best initial columns for class balance accuracy are created with B10 because the procedure results in columns with very few instances. In fact, most columns contain only a single instance, which seems to negate the premise that it is desirable to make decision variables that are collections of good instances. However, if columns of individual instances were added to all the generated columns before the final greedy selection, the best initial columns would ***not*** have B10. It is believed that for the imbalanced data problem it is desirable to have small columns that can be used to bolster the accuracy of other large but good columns. The other generated columns appear to be too large to allow a useful combination to be found in the final greedy selection procedure. Still, for the outlined experiment, initial columns created through B10 are the best choice.

After selecting B10 for the initial columns, the remaining parameter choices make intuitive sense. First, because RMP1 is chosen, it is logical that *K* should be a small value so that improving columns that are only slight improvements will be discovered. That is, because RMP1 only includes the highest reduced cost column generated in the POP, fewer columns should be checked to allow moderately improved columns to be added to *J'* and to be used in the final greedy search. Second, information is likely used to rank instances because the frequency measure may favor instances from the minority class. Meaning, because only the high accuracy columns are included in the frequency calculation and because these columns contain minority instances out of necessity, and because there are by definition a limited number of minority instances, minority instances receive a disproportionate high rank. Relying on information instead allows minority instances to be assessed for their actual usefulness.

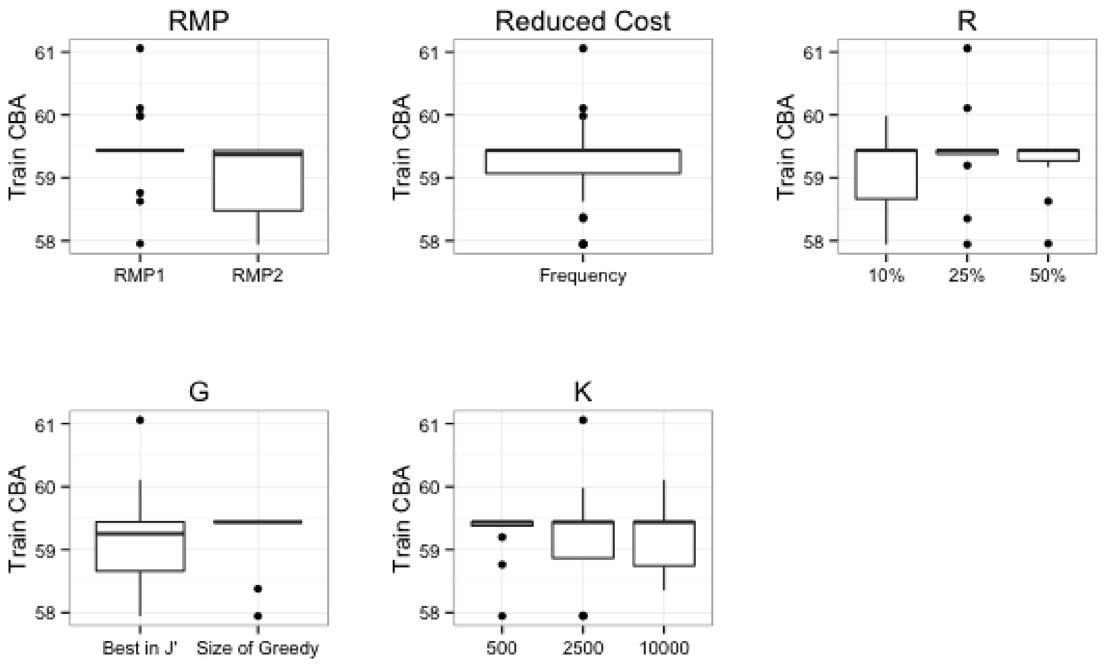

Figure 19. Boxplots to help determine rules of thumb for class balance accuracy.

CHAPTER 6. CONCLUSIONS

The motivation of this research has been that there are aspects of a classifier's training data that can impede its effectiveness. For model-based classifiers these troubling aspects are not always identifiable through a simple analysis of the data. Therefore, this work has proposed that instance selection be used to optimize the training dataset to the classifier's advantage. Specifically, instance selection is reformulated as an integer program and draws upon optimization theory to obtain a good selection of instances from which to learn.

Experimental results show that instance selection can increase the accuracy of classifiers by optimizing the training data in such a way that the training dataset has easier to learn boundaries between class values. In part, it was shown that instance selection might remove instances from classes that partially overlap, instances with outlying attribute values, and instances from minority classes that could potentially be ignored. However, exactly what is removed from a set of training data is dependent on both the classification problem and the intended classifier. Additionally, results show that a classifier built after instance selection can be a competitive classification technique, but instance selection may also be prone to overfitting classifiers that have already obtained a good fit to the data.

Also included in this dissertation were two case studies that concern predicting the current or future health of patients in the SEER database. For both cases it was shown that instance selection could improve the classifier constructed from the original training data. In fact, for the second case study where the goal was to predict mortality after a Nephroureterectomy, instance selection lead to a classifier that was actually informed about the classification problem. Meaning, unlike the original classifier, it did not simply predict that every patient would survive the procedure.

In short, instance selection can help improve the classification ability of model-based classifiers by allowing the classifier to choose a training dataset that considers its particular strengths and weaknesses.

6.1 Future work

The focus of this research has been instance selection to increase a model-based classifier's accuracy. However, in the process of completing this research, interesting areas for future investigation have been identified.

Overfitting – Experimental results in Chapter 4 suggests that instance selection can lead to overfitting. At least part of the overfitting can be contributed to the greedy selection procedure used to combine columns into a final selection of instances. It may be beneficial to devise a more sophisticated combination procedure to specifically address overfitting.

Imbalanced data – In Chapter 5 it was shown that using instance selection to maximize class balance accuracy lead to the discovery of a more useful model for imbalanced data. It may be interesting to further investigate the benefit of instance selection for imbalanced datasets where the ability to predict each class is important. Specifically, it would be interesting to investigate what instances are important in imbalanced datasets.

Model simplicity – Many of the techniques used to improve classifier accuracy inherently lead to classifiers that are more complex. For example, ensemble classifiers rely on multiple classifiers to make a single prediction, which makes interpretation difficult. With instance selection the result is a single classifier, and it has been noticed for decision trees and logistic regression that this classifier is sometimes simpler than the original. Meaning, for logistic regression the number of non-zero coefficients has decreased, and for decision trees the number of branches has decreased. In fact, the simplification of decision trees through instance selection was the focus of Bennette and Olafsson (2011), but it may be worth revisiting this area to investigate the tradeoffs between interpretability and accuracy in instance selection.

APPENDIX A. TABLED RESULTS

Naïve Bayes						
	Mean Train Accuracy			Mean Test Accuracy		
Dataset	Before IS	After IS	CI of Difference	Before IS	After IS	CI of Difference
Balance	89.8	92.2	[-2.9, -2.0]	89.5	90.5	[-1.5, -0.5]
Credit	78.4	87.5	[-10.0, -8.2]	78.0	83.7	[-6.5, -4.9]
Diabetes	77.0	80.3	[-3.7, -2.8]	75.4	74.8	[-0.0, 1.3]
Ecoli	88.7	92.2	[-4.1, -2.9]	84.5	84.9	[-1.5, 0.6]
Glass	56.7	78.4	[-23.8, -19.5]	46.2	63.5	[-21.3, -13.2]
Horse	72.2	79.1	[-8.3, -5.6]	69.0	68.3	[-0.8, 2.1]
Ionosphere	84.4	95.7	[-12.3, -10.4]	81.6	92.3	[-12.7, -8.7]
Landsat	79.0	86.4	[-8.7, -6.3]	76.9	80.4	[-4.5, -2.6]
Spect	79.6	87.7	[-8.9, -7.3]	77.6	79.0	[-3.1, 0.4]
Tic-tac-toe	72.0	76.8	[-5.6, -4.1]	71.5	73.3	[-2.9, -0.7]
Waveform	80.1	85.3	[-5.8, -4.6]	79.0	80.7	[-2.4, -1.0]
Wisconsin Breast Cancer	96.1	97.9	[-1.9, -1.5]	95.9	96.1	[-0.6, 0.2]

Table 12. Confidence intervals are constructed using a paired t-test with 95% confidence. Null hypothesis is that the mean difference before and after instance selection (IS) is zero.

Logistic Regression						
	Mean Train Accuracy			Mean Test Accuracy		
Dataset	Before IS	After IS	CI of Difference	Before IS	After IS	CI of Difference
Balance	91.2	93.8	[-3.0, -2.2]	89.3	91.5	[-3.0, -1.3]
Credit	89.3	90.9	[-2.2, -0.8]	84.7	83.5	[0.2, 2.2]
Diabetes	78.4	80.6	[-2.6, -1.8]	76.2	75.6	[-0.1, 1.2]
Ecoli	90.0	92.9	[-3.3, -2.3]	84.7	83.7	[-0.1, 2.2]
Glass	76.2	85.1	[-10.2, -7.6]	63.5	60.6	[0.7, 5.1]
Horse	83.2	94.7	[-12.5, -10.5]	64.3	58.8	[2.8, 8.3]
Ionosphere	96.8	98.9	[-2.9, -1.3]	86.0	84.0	[0.7, 3.4]
Landsat	100.0	100.0	[-0.0, 0.0]	72.5	72.4	[-0.9, 1.2]
Spect	87.6	91.9	[-4.7, -4.0]	81.2	78.7	[1.1, 4.0]
Tic-tac-toe	98.4	98.8	[-0.5, -0.3]	98.1	97.8	[-0.1, 0.6]
Waveform	86.8	89.1	[-2.6, -2.0]	83.0	82.4	[-0.3, 1.3]
Wisconsin Breast Cancer	97.0	97.9	[-1.0, -0.8]	96.2	96.6	[-0.8, -0.0]

Table 13. Confidence intervals are constructed using a paired t-test with 95% confidence. Null hypothesis is that the mean difference before and after instance selection (IS) is zero.

Decision Tree						
	Mean Train Accuracy			Mean Test Accuracy		
Dataset	Before IS	After IS	CI of Difference	Before IS	After IS	CI of Difference
Balance	89.8	91.2	[-1.9, -0.9]	77.4	77.7	[-1.5, 1.1]
Credit	90.2	92.6	[-3.4, -1.4]	84.8	84.4	[-0.4, 1.3]
Diabetes	86.1	90.6	[-6.1, -2.9]	72.7	71.8	[-0.8, 2.6]
Ecoli	92.1	94.3	[-3.0, -1.5]	81.4	81.6	[-2.0, 1.5]
Glass	91.4	95.0	[-4.4, -2.7]	67.8	67.8	[-2.3, 2.1]
Horse	76.2	83.6	[-9.1, -5.9]	64.3	64.3	[-2.2, 2,2]
Ionosphere	98.1	98.9	[-1.4, -0.3]	88.5	88.8	[-2.2, 1.7]
Landsat	95.7	97.1	[-2.0, -0.7]	76.1	75.2	[-1.0, 2.8]
Spect	87.8	90.5	[-3.3, -2.0]	79.2	79.2	[-1.5, 1.4]
Tic-tac-toe	92.3	94.8	[-3.3, -1.8]	84.0	86.1	[-3.4, -0.9]
Waveform	96.1	98.1	[-3.0, -1.0]	72.8	72.2	[-1.1, 2.3]
Wisconsin Breast Cancer	97.7	98.4	[-1.1, -0.5]	94.4	94.7	[-1.0, 0.3]

Table 14. Confidence intervals are constructed using a paired t-test with 95% confidence. Null hypothesis is that the mean difference before and after instance selection (IS) is zero.

BIBLIOGRAPHY

Bennette (2011). Instance selection for simplified decision trees through the generation and selection of instance candidate subsets. Master thesis, Iowa State University.

Bhattacharyya, Fried (2002). Nodal Metastatis in Major Salivary Gland Cancer, Arch Otolaryngol Head Neck Surg, 128, 904-908.

Bradley, U. Fayyad, O. Mangasarian. (1999). Mathematical Programming for Data Mining: Formulations and Challenges, INFORMS Journal on Computing, 11(3), 217-238.

Breiman (2001). Random Forests. Machine Learning. 45(1):5-32.

Cano, Herrera, Lozano. (2003). Using Evolutionary Algorithms as Instance Selection for Data Reduction in KDD: An Experimental Study, IEEE Transactions on Evolutionary Computation, 7(6), 561-575.

Cano, Herrera, Lozano. (2006). Evolutionary Stratified Training Set Selection for Extracting Classification Rules with Trade off Precision-Interpretability, Data & Knowledge Engineering, 60, 90-108.

Desrosiers, Lubbecke (2005). A Primer in Column Generation: in Column Generation, Desaulniers, Desrosiers, Solomon (Eds), Springer, Boston, MA, 1-32.

Endou, Zhao. (2002). Generation of Comprehensible Decision Trees Through Evolution of Training Data, in proceedings of the 2002 Congress on Evolutionary Computation, 1221-1225.

Garcia-Pedrajas. (2011). Evolutionary computation for training set selection, WIREs Data Mining and Knowledge Discovery, 1, 512-523.

Guy Brock, Vasyl Pihur, Susmita Datta and Somnath Datta (2011). clValid: Validation of Clustering Results. R package version 0.6-4. http://CRAN.R-project.org/package=clValid

H. Wickham. ggplot2: elegant graphics for data analysis. Springer New York, 2009.

Han, Kamber. (2006). Data Mining Concepts and Techniques, Morgan Kaufmann, San Francisco, CA.

Hart. (1968). The Condensed Nearest Neighbor Rule, IEEE Transactions on Information Theory (Corresp.), IT-14, 515-516.

Hastie, Trevor, et al. (2009). The elements of statistical learning. Vol. 2. No. 1. New York: Springer, 2009.

Jeldres, Sun, Isbarn, Lughezzani, Budaus, Alasker, Sharlat, Lattouf, Widmner, Pharand, Arjane, Graefen, Montorsi, Perrotte, Karakiewicz (2009). A Population-based Assessment of Perioperative Mortality After Nephroureterectomy for Upper-tract Urothelial Carcinoma, Urology, 75(2), 315-320.

Kim. (2006). Artificial neural networks with evolutionary instance selection for financial forecasting, Expert Syst Appl, 30, 519-526.

Kovács, Ferenc, Legány, and Babos (2005). Cluster validity measurement techniques. The

6th international symposium of Hungarian researchers on computational intelligence.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

Mosely (2014). A balanced approach to the multi-class imbalance problem, PhD thesis, Iowa State University.

Oates, Jensen. (1997). The Effects of Training Set Size on Decision Tree Complexity, in proceedings of the Fourteenth International Conference on Machine Learning.

Olafsson, Li, Wu. (2008). Operations research and data mining, European Journal of Operational Research, 187(3), 1429-1448.

Olvera-Lopez, Carrasco-Ochoa, Martinez-Trinidad, Kittler. (2010). A review of instance selection methods, Artif Intell Rev, 34, 133–143.

Olvera-Lopez, Martinez-Trinidad, J. Carrasco-Ochoa, J. Kittler, Prototype selection based on sequential search, Intelligent Data Analysis, 13(2009) 599-631.

Quinlan. (1992). C4.5 Programs for Machine Learning, Morgan Kaufmann, San Francisco, CA.

R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/

Reeves, Bush. (2001). Using genetic algorithms for training data selection in RBF networks, in: Instance Selection and Construction for Data Mining, Liu and Motoda (Eds), Kluwer, Norwell, MA, 339–356.

Reeves, Taylor. (1998). Selection of training sets for neural networks by a genetic algorithm, Parallel Problem Solving from Nature- PSSN V, 633-642.

Ritter, Woodruff, Lowry, Isenhour. (1975). An Algorithm for a Selective Nearest Neighbor Decision Rule, IEEE Transactions on Information Theory, 21(6), 665-669.

Sebban, Nock, Chauchat, Rakotomalala. (2000). Impact of learning set quality and size on decision tree performances, International Journal of Computers, Systems and Signals, 1(1), 85-105.

SEER Research Data 1973-2010 -- ASCII Text Data: Surveillance, Epidemiology, and End Results (SEER) Program (www.seer.cancer.gov) Research Data (1973-2010), National Cancer Institute, DCCPS, Surveillance Research Program, Surveillance Systems Branch, released April 2013, based on the November 2012 submission.

UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, Irvine, CA, University of California, School of Information and Computer Science 2010.

Wilhelm. (2001). A Technical Review of Column Generation in Integer Programming, Optimization and Engineering, 2(2), 159-200.

Wilson. (1972). Asymptotic Properties of Nearest Neighbor Rules Using Edited Data, IEEE Transactions on Systems, Man, and Cybernetics, 2(3), 408-421.

Witten, Frank. (2005). Data Mining; Practical Machine Learning Tools and Techniques, Elsevier, San Francisco, CA.

Zhu, Wu. (2006). Scalable Instance Selection and Ranking, The 18th International Conference on Pattern Recognition, 352-355.