2014

Ramp Loss SVM with L1-Norm Regularizaion

Eric Hess
Virginia Commonwealth University

RAMP LOSS SVM WITH L1-NORM REGULARIZATION

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

ERIC J. HESS

Masters of Science - September 2009 to August 2014

Director: Thesis Paul Brooks,

Associate Professor, Department of Mathematical Sciences

Virginia Commonwealth University

Richmond, Virginia

August, 2014

Acknowledgements

Thank you to all my friends and family that were there for me during my trials and errors in all my post graduate studies so far.

Thank you to Dr. Paul Brooks, someone whom I look up to and respect very highly. You have been patient with me even when I feel that I do not deserve it. I now have the tools that I need to commit to future ventures not only in capacity, but also quality. You are a person I will always reflect on as being influential in the most positive ways possible.

Thank you to my committee members Nihar Sheth and Dr. Qin Wang for helping me get through the last phase of my program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

RAMP LOSS SVM WITH L1-NORM REGULARIZATION

By Eric J. Hess, M.S.

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2014.

Director: Thesis Paul Brooks,
Associate Professor, Department of Mathematical Sciences

The Support Vector Machine (SVM) classification method has recently gained popularity due to the ease of implementing non-linear separating surfaces. SVM is an optimization problem with the two competing goals, minimizing misclassification on training data and maximizing a margin defined by the normal vector of a learned separating surface. We develop and implement new SVM models based on previously conceived SVM with L_1-Norm regularization with ramp loss error terms. The goal being a new SVM model that is both robust to outliers due to ramp loss, while also easy to implement in open source and off the shelf mathematical programming solvers and relatively efficient in finding solutions due to the mixed linear-integer form of the model. To show the effectiveness of the models we compare results of ramp loss SVM with L_1-Norm and L_2-Norm regularization on human organ microbial data and simulated data sets with outliers.

CHAPTER 1

INTRODUCTION

Classification, the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications [1]. There are many different classification methods that have good predictive qualities including: Decision Trees, Ruled-Based classifiers, Nearest-Neighbor classifiers, Bayesian classifiers, Artificial Neural Networks, and Support Vector Machines (SVM). Each of these classification methods have unique benefits and drawbacks; SVM, in particular, has become increasingly popular due to an ease of programming and application of many nonlinear discriminants described in section 1.4. There are several variations on Support Vector Machines; traditional SVM or C-SVM is described herein. Traditional SVM is very popular but performance is diminished with the presence of outliers for the linear classifying SVM, many solutions lack generalizability due to the trade-off between loss and margin in the objective. Brooks [2] presents a solution to this outlier problem with two mixed quadratic-integer programming (MQIP) formulations of C-SVM that give less weight to these problematic training points and therefore better generalizability as compared to traditional C-SVM. Although performance is greatly improved, these MQIPs are NP-hard and therefore many classification tasks may be time consuming or impossible with Brooks' approach. Studies of robustness on SVM with *ramp loss*, also known as *robust hinge loss* is limited as of yet. Methods for training SVM with ramp loss differ from traditional SVM in that the all training observations that fall outside of the margin of the opposite class have a loss of 2 on the objective function, while observations that fall in the margin are given a continuous measure of error between 0 and 2 depending on the distance to the correctly classified margin boundary [2].

Shen et al. [3] and Wang et al. [4] use optimization methods for SVM with the ramp loss that do not guarantee global optimality. In order to improve algorithm efficiency, Brooks [2] presents a family of facets to cut off fractional solutions for the linear kernel and introduces some heuristics to find good integer feasible solutions at nodes in the branch and cut tree for SVM with ramp loss.

Robust linear programming formulations for finding optimal hyperplanes for discrimination of linearly inseparable sets have been studied in the early 1990's prior to papers on SVM being printed in English. Bennett and Mangasarian [5] proposed a single linear programming formulation which generates a plane that minimizes an average sum of misclassified points belonging to two disjoint sets of observations in *n*-dimensional real space. Mangasarian [6] introduced a generalized SVM formulation that could be used for both quadratic and linear programming formulations. Hadzic [7] and Kecman [8] provide a formulation that utilizes the L_1-norm of the separating hyperplane for maximizing the margin which performed well in several empirical tests. Zhou, Zhang, and Jiao [9] introduces a linear programming SVM formulation in which the margin is defined as the right hand side of the constraint, $y_i\left(\mathbf{w}\cdot\mathbf{x_i}+b\right)\geq 1$, with an unknown in place of 1. This is unique because the right hand side of these constraints is representative of the margin width of the separating hyperplanes.

The classification method that is unique this paper is a SVM with Ramp Loss trained via mixed integer-linear programming. In the study of SVM with ramp loss, Brooks [2] used the L_2-Norm of the margin boundary for training the maximum margin hyperplane. Since this is the Euclidean length of the vector, $\|\mathbf{w}\|$, where as the new formulations discussed herein minimize an L_1-Norm of some measure of the primal variable, $\mathbf{w}$'s or dual variable, α's (via Mangasarian[10], Hadzic [7], and Zho [9]), Brooks' SVMs are quadratic programming problems. To create SVM models integer-linear programming problems we introduce ramp loss error terms, and in turn these problems can be solved using a branch

and bound framework with each subproblem solvable by simplex or any possibly any other popular solution algorithm for SVM. The expected benefit for the change from L_2-Norm SVM to the L_1-Norm SVM are two-fold; this means finding optimal solutions quicker than Brooks' ramp-loss SVM and providing more robust methodology for building SVM discriminants than the L_1-Norm SVM. This paper describes all variations of linear programming SVM with ramp loss in Chapter 3 to quantify the quality the qualities of each method via empirical evidence of both the Human Microbiome Project data [11] and simulated data. Results in Chapter 5 show that there are benefits to these new SVM models.

1.1 Traditional Support Vector Machines

A support vector machine (SVM) is a binary classification method which defines a hyperplane that (at best) separates all possible observations by class label developed by Vapnik [12]. This hyperplane is solved for using training data which consists of a paired observation as shown in Figure 1.1. These observations are defined by a training vector of common attributes $\mathbf{x}_i \in \mathbb{R}^n, i = 1, ..., l$ which have an associated class label y_i. It is assumed that there exists some unknown probability distribution $P(\mathbf{x}, y)$ from which these data are drawn, i.e. independently drawn and identically distributed (P is the cumulative probability distributions) which allows for approximation of actual error of the true misclassification rates with empirical error. The support vector machine learns a mapping, $\mathbf{x}_i \in \mathbb{R}^n \mapsto y_i \in \{-1, 1\}$, with the goal of maximizing a classification margin $2/\left|\left|\mathbf{w}\right|\right|$ and minimizing some measure of training error. The vector $\mathbf{w}$ is the normal vector to the separating hyperplane and is used to define the maximal margin of the SVM (i.e. the distance between two distinct hyperplanes that are equidistant from and parallel to the optimal separating surface). Testing for the parameter that controls the balance of margin and misclassification is typically done empirically by testing the SVM solution on unseen observations. The traditional formulations and derivations of, for separable and non-separable

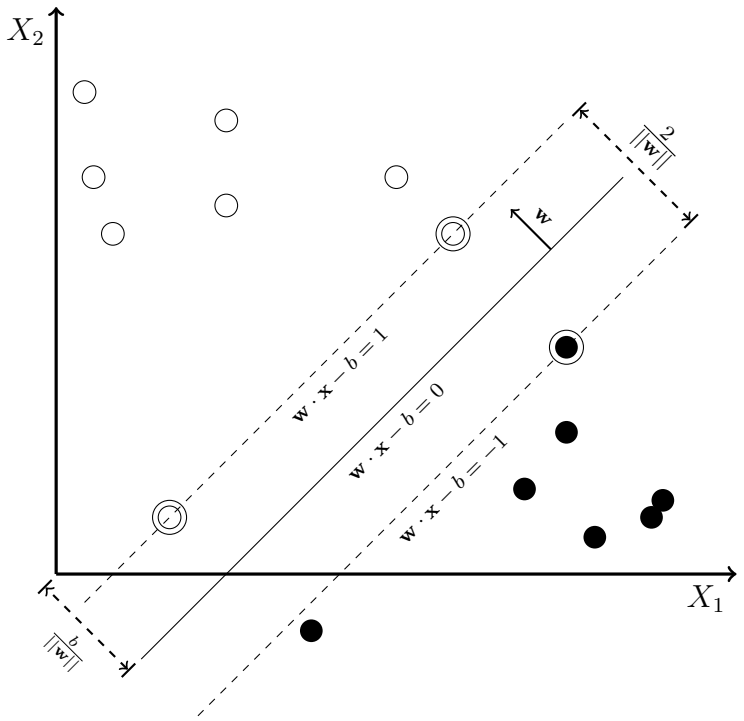

Fig. 1. Separable case of applied SVM.

SVM are briefly described in this chapter.

1.2 Separable Support Vector Machines

Given a training set of data points consisting of the tuple $(\mathbf{x}_i, y_i) \in \mathbb{R}^n$ x $\{-1, 1\}, i = 1, ..., n$, where y_i is the class label and x_i is the parameter values of the i^{th} observation, we can find planar and high-dimensional continuous surfaces that separate the classes well[13]. The random variables X and Y can be taken from a unknown conditional distribution $P(Y = h|X = X)$ [2]. Given these data points a function $f : x_i \in \mathbb{R}^n \rightarrow y_i \in \{-1, 1\}$ is a classifier for data with unknown class labels. In the case in which all training observations one possible hyperplane that can separated based on class label completely and correctly, the traditional SVM math model can be implemented. A hyperplane for discrimination purposes will be the result of implementing SVM which is defined as $\mathbf{w} \cdot \mathbf{x} + b = 0$,

in which $|b|/\,||w||$ is the perpendicular distance from the hyperplane to the origin, and $||\cdot||$ is the Euclidean norm. The support vectors are points within $1/\,||\mathbf{w}||$ units from the classification hyperplane. There are at least $n+1$ points that will be defined by SVM what will be support vectors in the case that all data are separable, of which, at least one point satisfies $\mathbf{w}\cdot\mathbf{x}+b=-1$ for the support vectors in which $y_i=-1$ and one point satisfies $\mathbf{w}\cdot\mathbf{x}+b=1$ for the support vectors in which $y_i=1$. These support vectors more specifically help define the boundaries of sets of correctly classified observations, the distance between these sets is $2/\,||w||$. It is optimal for generalization performance to maximize this margin, but we use a computationally simpler quadratic objective which we minimize, $2/\,||\mathbf{w}||^2=\frac{1}{2}\mathbf{w}\cdot\mathbf{w}$. This is the objective provided that all training observations can be classified correctly. The standard SVM formulation is (1.1).

$$\text{[Linear Separable SVM]} \qquad \begin{aligned} &\min_{\mathbf{w},b} && \tfrac{1}{2}\,||\mathbf{w}||^2\,, \\ &\text{subject to} && y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1, \quad i=1,2,...,n. \end{aligned} \tag{1.1}$$

A drawback to using this formulation is that it only allows for finding linear discriminants. An easy way to formulate a nonlinear discriminant is applied with use of the Lagrangian formulation (1.2) and Karush-Kuhn-Tucker (KKT) conditions. This trick for to produce these nonlinear discriminants is called a nonlinear mapping and is discussed in section 1.4. To create this nonlinear mapping of factors we first need to have the points in the form of dot products which conveniently so, the dual formulation has. The first step to finding the dual SVM is to solve for the Lagrangian, which have new unknowns called the Lagrange multipliers (i.e. dual variables), $\alpha_i, i=1,...,l$; there is one dual variable for each constraint in the L_P (1.2).

$$L_P=\frac{1}{2}\,||\mathbf{w}||^2-\sum_{i=1}^{l}\alpha_i\,(y_i(\mathbf{x}_i\cdot\mathbf{w}+b)-1) \tag{1.2}$$

The optimal solution the primal and the dual formulations will be the same (1.7) by the strong duality theorem. This dual formulation is obtained by taking the derivative of L_P with respect to w and b and setting each of the resulting formulations to zero, as shown in (1.3) and (1.4). These derivations became very important in solving for nonlinear discriminants.

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{1.3}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{1.4}$$

The KKT conditions also apply nonnegative constraints on all of the Lagrange Multipliers ($\alpha_i \geq 0$). The formulation that has the training vectors as dot products is the Dual Lagrangian, which can easily be found by appropriately plugging (1.3) and (1.4) into (1.2) which is the maximization problem (1.5).

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{1.5}$$

The separable SVM Problem is convex and all KKT conditions are necessary and sufficient for **w**, b, and α to be a solution to both the L_P and L_D optimization problem [14]. The dual problem is typically solved with the L_D as the objective, (1.4) as the lone set of constraints, and non-negativity on the α_i's. For many cases the dual problem can be solved more quickly than the primal problem, although this is not the most important property of the dual problem. In fact, the importance of deriving and solving the problem with respect to the dual formulation is that the attribute vectors are presented as dot products, the necessary property for nonlinear application of SVM. The discriminant can also be calculated from the resulting dual variables values.

The discriminant defined by the solution of L_P is, $f : \mathbf{w} \cdot \mathbf{x} + b = 0$. Although $\mathbf{w}$ is

missing in the dual solution, the equivalence of $\mathbf{w}$ is found using the first KKT condition, (1.3); replacing $\mathbf{w}$ with $\sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i$ gives us the discriminant in (1.6).

$$[\text{SVM Discriminant}] \quad \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i \cdot \mathbf{x} + b = 0, \tag{1.6}$$

This leaves only solving for the offset variable, b. This is typically found by solving for b with KKT condition, $\alpha_i \left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - 1 \right] = 0$, where $\mathbf{w}$ should be replaced by $\mathbf{w}_j = \sum_{i=1}^{i=n} y_i \alpha_i x_{ij}$ for each support vector. The b values for each of the support vectors, (i.e. $\alpha_i \neq 0$) are typically not unique [13] and the common practice for selecting b is to take the average of the values found.

The next section introduces the problem in which we allow inclusion of misclassified training observations.

1.3 Nonseparable Support Vector Machines

For most classification problems, it is not just sufficient to produce a classifier that performs well on trained observations, but also perform well on unseen data. Improvements in generalization performance and ease of finding solutions are the main focus for the introduction of Nonseparable SVM formulations. Creating this model involves the introduction of slack variables ξ_i for each constraint from (1.7) as $y_i \left(\mathbf{w} \cdot x_i + b \right) \geq 1 - \xi_i$ where ξ_i's are nonnegative. An example is shown in Figure 1.3. As seen in the figure, ξ is a product of the L_2-Norm of $\mathbf{w}$ (i.e. the inverse of the perpendicular distance between the parallel hyperplanes that define the SVM, otherwise known as the margin) and the distance that the misclassified point is from the hyperplane on the vector's side of the separating surface. By this interpretation of ξ, it can easily be shown that bigger margins produce smaller values for individual ξ's but more points may be associated with positive ξs.

It is possible to only modify the original constraints and not at ξ's to the objective

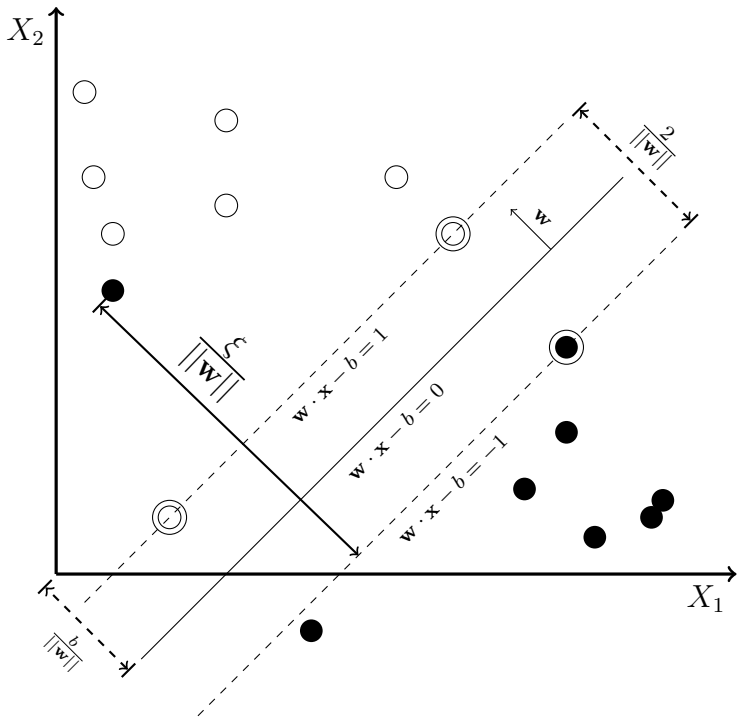

Fig. 2. Nonseparable case of applied SVM.

function which is to minimize $||\mathbf{w}||^2/2$, but this formulation will produce the null solution, i.e. $\mathbf{w} = 0$. This solution is meaningless, therefore any classification problem that produces only these null solution are most likely unclassifiable with the given factors. In an attempt to minimize the likelihood of this outcome, we add a penalty to the objective function for each misclassified observation, $f(\xi)$. The typical Nonseparable SVM formulation is (1.7).

$$\text{[Nonseparable SVM]} \quad \begin{aligned} \min_{\mathbf{w},b} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i, \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1-\xi_i, \quad i = 1,2,\cdots,n. \\ & \xi_i \geq 0, \qquad\qquad\quad i = 1,2,\cdots,n. \end{aligned} \tag{1.7}$$

The variable C is a constant specified prior to finding the solution. Selection of the value of C is typically done through empirical methods for each instance. Just like the

formulation for separable SVM, (1.2), we can find the dual using the KKT conditions in order to produce nonlinear discriminants, beginning with finding the Lagrangian of (1.7).

$$L_P = \frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(y_i(\mathbf{x}_i\cdot\mathbf{w}+b)+\xi_i-1\right) - \sum_{i=1}^{N}\mu_i\xi_i \tag{1.8}$$

The first two terms are the objective function to be minimized, the third term represents the inequality constraints associated with the slack variables, and the last term is the result of the non-negativity requirements on the values of ξ_i's. We also have the following KKT conditions to satisfy:

$$\xi_i \geq 0,\ \alpha_i \geq 0,\ \mu_i \geq 0, \tag{1.9}$$

$$\alpha_i\{y_i\left(\mathbf{w}\cdot\mathbf{x}_i+b\right)-1+\xi_i\} = 0, \tag{1.10}$$

$$\mu_i\xi_i = 0 \tag{1.11}$$

The Lagrange multiplier α_i given in (1.10) is non-zero only if the training instance resides along the planes $\mathbf{w}\cdot\mathbf{x}_i + b = \pm 1$ or if the corresponding slack variable $\xi_i > 0$. In fact, for any observation that is on the correctly classified boundary $\mathbf{w}\cdot\mathbf{x}_i + b = \pm 1$ or on the wrong side of it, the training point is deemed a support vector. Again we apply the KKT conditions to the SVM formulation by taking the partial derivatives of L_P with respect to $\mathbf{w}$, b, and ξ to zero, we get the following equations:

$$\frac{\partial L_P}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i \tag{1.12}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N}\alpha_i y_i = 0 \tag{1.13}$$

$$\frac{\partial L_P}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \mu_i = 0 \tag{1.14}$$

Substituting equations (1.12), (1.13), and (1.14) into L_P correctly produces the following dual Lagrangian:

$$\begin{aligned} L_D =& \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j + C\sum_i \xi_i \\ & - \sum_i \alpha_i \left(y_i \left(\sum_j \alpha_j y_j \mathbf{x}_i\cdot\mathbf{x}_j + b \right) - 1 + \xi_i \right) \\ & - \sum_i (C-\alpha_i)\,\xi_i, \\ =& \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j. \end{aligned} \tag{1.15}$$

which is identical to the L_D for linearly separable data. The only difference between the separable and nonseparable dual formulations is that equation (1.14) constrains α_i to a maximum of C since μ_i and α_i are both nonnegative. Therefore, the Lagrange multipliers are constrained to $0 \leq \alpha_i \leq C$. The discriminant for L_D of nonseparable SVM is the same as that from separable SVM (1.6). The offset b is also found in the same fashion it is for (1.5).

1.4 Nonlinear Support Vector Machines

Popularity and importance of SVM in classification is linked to the property for learning nonlinear separating surfaces easily. This is done through a method referred to as the kernel method in which all of the training points are mapped onto higher dimensional space to create a linear separating surface under these new dimensions. This mapping can then be replaced by the original dimensions as a nonlinear surface. This is done in SVM using L_D by mapping the training data on a higher dimensional space $\mathcal{H}$ using $\Phi : \mathbb{R}^k \longmapsto \mathcal{H}$. Since the dual algorithm depends on the dot products of the training points, we can map the observations on $\mathcal{H}$ by replacing $\mathbf{x}_i \cdot \mathbf{x}_j$ with $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$.

The higher dimensional mapping that is done via SVM is through the use of a kernel function, $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. We replace $\mathbf{x}_i \cdot \mathbf{x}_j$ with $K(\mathbf{x}_i, \mathbf{x}_j)$ in the dual formulation to allow for finding nonlinear separating surfaces. The kernel trick can then be described as a method for computing similarity in the transformed space using the original attribute set. An example of this transformation on two input vectors $\mathbf{u}$ and $\mathbf{v}$ in the transformed space can be written as follows:

$$\begin{aligned} \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) &= \left(u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1\right) \cdot \left(v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1\right), \\ &= u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1, \\ &= (\mathbf{u} \cdot \mathbf{v} + 1)^2. \end{aligned} \tag{1.16}$$

This is a simple example which shows that the dot product in the transformed space can be expressed in terms of a similarity function in the original space: $K(\mathbf{u}, \mathbf{v}) = \Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2$. This function is known as the kernel function and must satisfy Mercer's theorem for validity. This conditions ensures that the kernel function can always be expressed as the dot product between two input vectors in some high-dimensional space. Functions that satisfy Mercer's Theorem are called positive definite kernel functions. Examples of well known and often deployed kernels are the Polynomial (1.17), Radial (1.18), and Sigmoidal Neural Network (1.19) kernels shown below.

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p \tag{1.17}$$

$$K(\mathbf{x}, \mathbf{y}) = e^{-||\mathbf{x}-\mathbf{y}||^2/\left(2\sigma^2\right)} \tag{1.18}$$

$$K(\mathbf{x}, \mathbf{y}) = \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta) \tag{1.19}$$

These are just a few kernels that are often employed, there are more possible and some other SVM formulations can be implemented with discontinuous kernels. The nonlinear discriminant is obtained with the same transformation. That is, the original coordinate space in $\mathbf{x}$ is transformed into the new space $\Phi(\mathbf{x})$ so that a linear decision boundary can be used to separate the instances in the Hilbert space using the kernel trick. The nonlinear decision boundary obtained for SVM is then defined as shown in (1.20):

$$\begin{aligned} \sum_{i=1}^{n} y_i \alpha_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b = 0 \\ \sum_{i=1}^{n} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b = 0 \end{aligned} \tag{1.20}$$

Many nonlinear kernels can be deployed to produce various high performance classifiers for various problems giving the same problem many possible solution with modification to the kernel matrix. This is an important factor in the popularity of SVM in that many new solutions can be implemented with ease. The chief drawback of with high dimensional kernels is a discriminant of higher VC Dimension; as the transformed space gets larger the bounds on generalizability become greater [13]. The next section covers the literature describing various methodologies used for SVM with ramp loss and linear programming SVM.

CHAPTER 2

LITERATURE REVIEW

2.1 Linear Programming SVM

Linear programming formulations of Support Vector Machines are not new to the machine learning community. In fact both linear and quadratic formulations of separable SVMs were introduced in the 1960's. The documented concept creator of SVM, O.L. Mangasarian introduced a linear programming formulation for discrimination of linearly separable sets in 1965[15]. Several other formulations for linear programming SVM have been introduced; some of these formulations are discussed herein. Mangasarian [10] introduced a generalized SVM formulation that could be used for both quadratic and linear programming formulations. Hadzic [7] and Kecman [8] provide a formulation that utilizes the L_1-norm of the separating hyperplane for maximizing the margin which performed well in several empirical tests. Zhou, Zhang, and Jiao [9] introduces a linear programming SVM formulation in which the margin is defined as the right hand side of the constraint, $y_i\left(\mathbf{w}\cdot\mathbf{x_i}+b\right)\geq 1$, with an unknown in place of 1. This is unique because the right hand side of these constraints is representative of the margin width of the separating hyperplanes. Empirical result of these linear programming formulations as well as their ramp loss MIP formulation counterparts are described in Chapter 4.

2.1.1 Bennett and Mangasarian's Robust Linear Programming Discrimination of Linearly Inseparable Sets

Linear Programming Support Vector Machines are a general classification method with origins linked to the formulation developed by Kristin Bennett and O.L. Mangasarian

[5]. This formulation of classification (linear programming SVM) is used far less often than traditional C-SVM. Lack of popularity of this classification method maybe due to reduced interpretability of result due to the lack of margin in the objective function. Without the margin in the object function, outliers can have a great impact of the result. The claim in this paper is that we can create an optimal discriminant for two linearly independent sets of observations using the formulation presented in (2.1).

$$\text{[Robust LP]} \quad \min_{\mathbf{w},b} \quad \frac{1}{|m|}\sum_{i\in m}\left(y_i\mathbf{x}_i\cdot\mathbf{w}-y_ib+1\right)_+ + \frac{1}{|k|}\sum_{j\in k}\left(y_j\mathbf{x}_j\cdot\mathbf{w}-y_jb+1\right)_+ \tag{2.1}$$

In this formulation $m \in \{i = 1, 2, \cdots, n : y_i = -1\}$ and $k \in \{j = 1, 2, \cdots, n : y_j = 1\}$. The choice of weights $1/|m|$ and $1/|k|$ in (2.1) is believed to be a "natural" choice for avoiding the null solution $(\mathbf{w} = 0)$ under practical circumstances. Like traditional SVM the separating hyperplane learned is $\mathbf{w}\cdot\mathbf{x} = b$. This formulation does not take into account the objective of maximizing the margin directly, but does so indirectly with the penalties on the points violating respective class boundaries: $y_i\left(\mathbf{w}\cdot\mathbf{x}+b\right) < 1$. Sometimes a secondary one-dimensional optimization problem $(2.1c)$ in [5] can be used to improve the location of the separating surface by modifying the threshold b.

Limited performance testing was reported on the Wisconsin Breast Cancer Database using 9-dimensional real space and the Cleveland Heart Disease Database using 13-dimensional real space. In both cases, the formulation (2.1) out performed two other linear programming methodologies. No comparison between quadratic and linear programming discriminants is presented.

2.1.2 Mangasarian's Generalized SVM

Mangasarian [10] describes a generalized SVM formulation (2.2) that can be used to obtaining a linear or nonlinear separating surface (2.3). The practical importance of this formulation is in the diversity of separating surfaces that can be solved for with various functions in place of $f(\mathbf{u})$ in (2.2); several possibilities are presented in [10] including the derivation for the traditional convex quadratic formulation of SVM.

$$\begin{aligned} \min_{\mathbf{u},b,\xi} \quad & f(\mathbf{u}) + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i\sum_{j=1}^{n}\left(y_j K(\mathbf{x}_i,\mathbf{x}_j)\,u_j + b\right) \geq 1-\xi_i, \quad i=1,2,\cdots,n. \\ & \xi_i \geq 0 \qquad\qquad i=1,2,\cdots,n. \end{aligned} \tag{2.2}$$

Recall that $\mathbf{w}\cdot\mathbf{x}_i \mapsto \sum_{j=1}^{n} y_j K(\mathbf{x}_i,\mathbf{x}_j)\,\alpha_j$ from the dual of traditional SVM. By comparison sake, u_j's are then dual variables in (2.2) in the same sense as the dual variables described in (1.5) and (1.15). Therefore the separating surface produced from the solution of GSVM shown in (2.3) is the same as that for the nonlinear separable surface shown in (1.20).

$$\sum_{i=1}^{n}\left(y_i K(\mathbf{x},\mathbf{x}_i)\,u_i\right) = 0 \tag{2.3}$$

Mangasarian offers two different options for $f(\mathbf{u})$ to make a linear programming SVM from (2.2). Both of these functions are studied emperically. The first is the L1-norm of the dual variables, $\mathbf{u}$, and the second absolute value of $\sum_{j=1}^{j=n}\left(y_j K(\mathbf{x}_i,\mathbf{x}_j)\,u_j\right)$ as shown in (2.4). The formulation shown in (2.4) is used as the basis for a linear programming SVM with ramp loss formulation described in section 3.2.

In examining (2.4), the first set of constraints can be seen as equivalent to the soft margin constraints on each training observation in traditional SVM. And the second set of

constraints produces the absolute value of our general function in the objective $f(\mathbf{u})$. Unfortunately, there is limited empirical evidence that (2.4) has good classification properties, therefore the quality of this classifier is examined extensively in.

$$\begin{aligned} \min_{\mathbf{u},b,\xi,\mathbf{s}} \quad & \sum_{i=1}^{n} s_i + C\sum_{j=1}^{n} \xi_j, \\ \text{subject to} \quad & y_i \sum_{j=1}^{j=n} \left(y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) u_j + b\right) \geq 1 - \xi_i, \quad i = 1, 2, \cdots, n. \\ & s_i \geq \sum_{j=1}^{j=n} \left(y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) u_j\right) \geq -s_i, \qquad i = 1, 2, \cdots, n. \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad i = 1, 2, \cdots, n. \end{aligned} \tag{2.4}$$

2.1.3 Hadzic and Kecman's Linear Programming SVM for Classification

Hadzik [7] and Kecman [8] present formulations for LP-SVM which result in either linear or nonlinear discriminants. The first formulation, (2.5), is a derived from classical SVM with the only exception being we use of the L_1 as a regularization term instead of the L_2 Norm of the discriminant.

$$\begin{aligned} \min_{\mathbf{w}^+,\mathbf{w}^-,b,\xi} \quad & \sum_{i=1}^{n} \left(w_i^+ + w_i^- + C\xi_i\right), \\ \text{subject to} \quad & y_i \left(\mathbf{x}_i \cdot \mathbf{w}^+ - \mathbf{x}_i \cdot \mathbf{w}^- + b\right) \geq 1 - \xi_i, \quad i = 1, 2, \cdots, n. \\ & \mathbf{w}^+, \mathbf{w}^- \geq 0 \qquad\qquad i = 1, 2, \cdots, n+1. \end{aligned} \tag{2.5}$$

In this SVM formulation (2.5) the attribute vector $\mathbf{w}$ is split into positive $\mathbf{w}^+$ and negative $\mathbf{w}^-$ so that the generalization terms in the objective function are linear instead of quadratic. This new measure is referred to as the L_1-Norm because the sum of $\mathbf{w}^+$ and $\mathbf{w}^-$ equates to the absolute value of all parts of the vector $\mathbf{w}$.

There is another high performing LP SVM formulation (2.6), described in the text. This model lacks the typical trade-off parameter ξ_i used to account for misclassification and

to improve the performance of the classifier. Analysis of adding this trade-off parameter to the formulation is still a work in progress according to Kecman [8], although it is noted that anyone adding these parameters should do so with caution. Even though there is no trade-off parameter used for the problem, a gender recognition task is presented by Hadzic and Kecman as evidence of the performance of the formulation.

$$\begin{aligned} \min_{\mathbf{w}^+,\mathbf{w}^-,b} \quad & \sum_{i=1}^{n} \left(w_i^+ + w_i^-\right), \\ \text{subject to} \quad & y_i \left(\sum_{j=1}^{j=n} \left(K\left(\mathbf{x}_i, \mathbf{x}_j\right) w_j^+ - K\left(\mathbf{x}_i, \mathbf{x}_j\right) w_j^- \right) + b \right) \geq 1, \quad i = 1, 2, \cdots, n. \\ & w_i^+, w_i^- \geq 0 \qquad\qquad i = 1, 2, \cdots, n. \end{aligned} \tag{2.6}$$

The gender recognition task contained 1668 data pairs comprising 18-dimensional input vector $\mathbf{x}$ and result for male and female class labels. Kecman performed both QP-SVM and LP-SVM approaches to the data with 96.4% and 92.8% performance on test data correctly identified for these approaches respectively. This is in comparison to a Gaussian Basis Function Network which only performed at a 79% rate on images not used for training the classifier.

2.1.4 Zhou, Zhang, and Jiao's LP-SVM

Weida Zhou, Li Zhang, and Licheng Jiao [9] produce highly competitive linear programming SVM with both linear and nonlinear kernel formulations. The first formulation presented is (2.7), which replaces the metric representing the magnitude of the classifier margin in the objective function with r in the equation for the margin, $d = \frac{2r}{||\mathbf{w}||_2}$. By replacing the right hand side of the bounding constraints in traditional SVM with r (in place of 1), we allow for the substitution of terms in the objective function. Bounds are placed on $\mathbf{w}$ to have a magnitude no greater than 1 in order avoid an unbounded problem.

$$\begin{aligned} \max_{\mathbf{w},b,r} \quad & r \\ \text{subject to} \quad & y_i\left(\mathbf{w}\cdot\mathbf{x}_i+b\right)\geq r, \quad i=1,2,\cdots,n. \\ & -1\leq w_i\leq 1 \qquad i=1,2,\cdots,l. \\ & r\geq 0 \end{aligned} \tag{2.7}$$

The decision function takes the same form as that for traditional SVM, $f(\mathbf{x},\mathbf{w},b) = \mathbf{w}\cdot\mathbf{x}+b$ with result being the sign of this decision function.

For the nonseparable case with linear classifier,(2.8), the problem is modified for use of a minimization problem. To establish a minimization problem we take the negative of r and sum the slacks, ξ_i, weighted by the user parameter C. Introducing these slack variables in the first set of constraints allows for training points to lie on either side of the margin but penalize points on the wrong side accordingly.

$$\begin{aligned} \min_{\mathbf{w},b,r,\xi} \quad & -r+C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i\left(\mathbf{w}\cdot\mathbf{x}_i+b\right)\geq r-\xi_i, \quad i=1,2,\cdots,n. \\ & -1\leq w_i\leq 1 \qquad i=1,2,\cdots,l. \\ & r\geq 0 \end{aligned} \tag{2.8}$$

This objective would have the same decision function as the separable SVM. Finding the nonlinear LPSVM is similar to finding the nonlinear SVM for traditional SVM. Since we are looking for $\Phi : R^n \mapsto H$, we can map $\mathbf{x}_i\cdot\mathbf{x}_j$ to $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_i)$ with $K(\mathbf{x}_i,\mathbf{x}_j)$ if Mercer's condition holds true. Based on the KKT conditions of traditional SVM we can perform the substitution, $\mathbf{w} = \sum_{j=1}^{n}\alpha_j y_j \Phi(\mathbf{x}_j)$ in the set of classification bounding constraints. This substitution is made in (2.9), as well as the replacement of bounds on $\mathbf{w}$ with bounds on α.

$$
\begin{aligned}
\min_{\alpha,b,r,\xi} \quad & -r + C\sum_{i=1}^{n}\xi_i \\
\text{subject to} \quad & y_i\left(\sum_{j=1}^{n}\alpha_j y_j K\left(\mathbf{x}_j,\mathbf{x}_i\right)+b\right) \geq r-\xi_i, \quad i=1,2,\cdots,n. \\
& -1 \leq \alpha_i \leq 1 \qquad\qquad i=1,2,\cdots,n. \\
& r \geq 0 \\
& \xi_i \geq 0 \qquad\qquad i=1,2,\cdots,n.
\end{aligned}
\tag{2.9}
$$

Four experiments were performed on these formulations. For linear LPSVMs, separable data sets were created to show that there would be no loss in performance using the linear LPSVM in place of the QPSVM with a substantial improvement in solution speed. To test performance of the nonlinear LPSVM, the dual-spiral data, handwritten digital data, and the DS-CDMA multiuser detection data were used. Dual-spiral data is a popular test case to test classification problems; the goal of this classification problem is to separate two intertwined spirals of two-dimensional points in which each spiral containing points of a respected class.

With the use of 252 training points, both a traditional quadratic programming SVM and linear programming SVM correctly classify 628 remaining test points under the dual-spiral data problem (both with Gaussian kernel $k\left(\mathbf{x}_i,\mathbf{x}_j\right) = e^{\left(-|\mathbf{x}_i-\mathbf{x}_j|^2/2p^2\right)}$ with p=8). The training time improvement for this LPSVM and QPSVM solution of 199.7 and 4103.5 seconds is made only with compromise of approximately three times the VC dimension. A higher VC dimension generally correlates to lesser generalization performance.

The test for classification of handwritten digits was performed using "6's" and "9's" from the MNST database. Again the Gaussian kernel was implemented, this time with $p = 30$. In this test several sizes of training sets were used: 200, 400, 600, 800, and 1000 points. The point of this test is to show how an increase in the sample size will increase

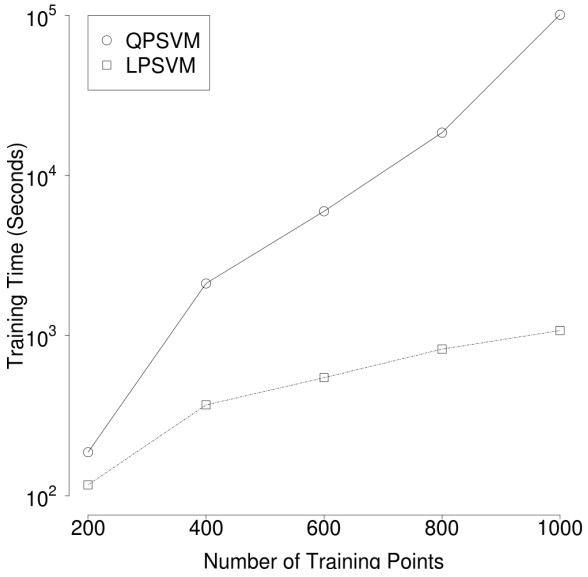

Fig. 3. Solution time of Traditional SVM compared to Zhou's LPSVM for handwritten data classification.

the solution speed of QPSVM exponentially, whereas, the solution speed for the LPSVM problem would not be impacted as greatly with additional training points. In fact, the training times for the LPSVM increase almost linearly with an increase in sample size as shown in Figure 3.

In the fourth experiment, a modern communication transmission technologies known as DS-CDMA (Direct Sequence Code Division Multiple Access) which allows for multiple users to transmit communication over the same band of frequencies is used to compare the quality of distinguishing signals with LPSVM and QPSVM. Two cases are tested, synchronous and asynchronous signals from ten and six users of the band from the respective synchronicity. The test being, is the signal being detected from the expected user or an interfering user. With varying levels of noise and both linear and radial basis function kernels being utilized, user recognition are nearly the same for both LPSVM and QPSVM formulations.

Zhou, Zhang, and Jiao's have provided an LPSVM formulation that can greatly reduce

the solution speed of the binary classification problem with experimental results as evidence of the quality of solutions as compared to traditional SVM.

2.2 Ramp Loss

One of the focuses of this paper is to find the benefits we get from applying ramp loss (robust loss) to the error terms in place of traditional hinge loss to linear programming SVM formulations. Figure 4, shows examples of the two types of losses with respect to traditional SVM. Ramp loss error terms are typically applied to learning machines in which outliers may shift the inferred formulation learned. Essentially the impact of outliers is limited due when ramp loss is applied to machine learning problems. In Support Vector Machines, this limit is typically 2 which corresponds to the margin boundary of the opposite class.

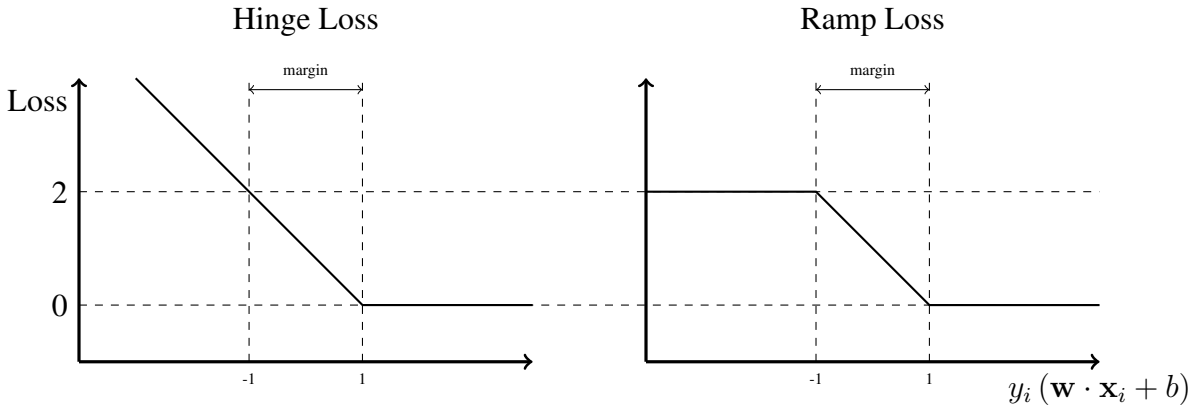

Fig. 4. Comparison of Ramp Loss and Traditional Hinge Loss used for Statistical Models

2.2.1 Wang and Vucevic's Fast Online Training Algorithm of Ramp Loss SVM

Since SVM with ramp loss creates instances that are NP-Hard in the MIQP framework, an interesting approach to reduce solution time for each instance is with an online algorithm. Zhuang Wang and Slobodan Vucetic describe a fast online algorithm OnlineSVM^{R} in [4]. Online learning involves a learning as data is presented procedure, as opposed to

all training data presented initially. This training algorithm produces optimal solutions for SVM with ramp loss on $t+1$ training data using the existing optimal solution on t previous examples. The algorithm retains the Karush-Kuhn-Tucker conditions on all previously observed examples using an SMO-style incremental learning and decremental unlearning approach under the ConCave Convex Procedure (CCCP) framework.

To produce a ramp loss function the errors have to be weighted as shown in (2.10).

$$R\left(x_i f\left(y_i, f(x_i)\right)\right) = \begin{cases} 0, & y_i f\left(x_i\right) > 1 \\ 1 - y_i f\left(x_i\right), & -1 \leq y_i\left(x_i\right) \leq 1 \\ 2, & y_i f\left(x_i\right) \leq 1 \end{cases} \tag{2.10}$$

The formulation for finding an optimal hyperplane with ramp loss SVM is shown in (2.11).

$$\min P^R(\mathbf{w}) = \min\left(\frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2 + \sum_{i=1}^{n} R\left(y_i, f\left(\mathbf{x}_i\right)\right)\right) \tag{2.11}$$

which is solved using CCCP, an algorithm involving a sequence of approximate convex problems with a convergence guarantee. This procedure can be used with the dual formulation, therefore nonlinear classifiers can still be found using CCCP. An offline CCCP procedure for solving SVM^R would involve the training of SVM from scratch for every iteration which is computationally expensive. Reducing the computational complexity for finding the solution is the reason OnlineSVM^R is proposed.

The online algorithm, OnlineSVM^R, involves the incremental learning and decremental unlearning of support vectors to guarantee the KKT conditions are satisfied with the additional of each unseen observation. In addition to this proposed algorithm, a set selection strategy is proposed to minimize kernel computations to achieve faster convergence. Additionally, for each iteration non-SVs that are far from the decision boundary are removed due to the fact that they are very unlikely to become SVs for future iterations of the

OnlineSVM^R.

Another modification to the algorithm is introduced as an online active learning approach, OnlineASVMR. The algorithm, OnlineASVM^R differs from OnlineSVM^R by only querying the examples within the ramp region of the previous iteration. The argument for including only examples that lie in the ramp region of the previous solution is that the dual variable, α_i, for this observation will be set to zero in the old solution, therefore it is unlikely that adding any examples not meeting this criteria will change the result.

Empirical testing results of OnlineSVM^R and OnlineASVM^R are compared against two online and one offline learning algorithms of SVM with hinge loss:

- IDSVM: an online SVM algorithm which guarantees optimality.
- Online Passive Aggressive (PA) algorithm: algorithm based on perceptron-style updating.
- LibSVM: offline SVM algorithm with hinge loss.

Experiments on 9 benchmark binary classification data sets were utilized. The multi-class data sets were converted to two-class sets as follows. For the digit dataset USPS we converted the original 10-class problems to binary by representing digits 1, 2, 4, 5, 7 (non-round digits) as negative class and digits 3, 6, 8, 9, 0 (round digits) as positive class. For 3-class DNA data set class 3 was separated from the other 2 classes. Class 1 in the 3-class Waveform was treated as negative and the remaining two as positive. For Cover type data the class 2 was treated as positive and the remaining 6 classes as negative. Adult, Banana, Checkerboard and Gauss were originally 2-class data sets. *NCheckerboard* is a noisy version of Checkerboard where class assignment was switched for 15% of the randomly selected examples. For both data sets, we used the noise-free Checkerboard as the test set. Attributes in all data sets were scaled to mean 0 and standard deviation 1.

For all the examples, the RBF-kernel was used, $k(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2 / 2\delta^2\right)$. Hyper-parameters C and δ^2 were selected via cross-validation for every combination. The considered values for these parameters were $C = \{0.1,\ 1,\ 5,\ 10,\ 50,\ 100,\ 500\}$ and $\delta^2 = \{M/2 - 1,\ M/20,\ M/21,\ M/22,\ M/24,\ M/26\}$, where M is the number of dimensions.

2.2.2 ψ-Learning by Shen et. al.

The development of ψ-Learning [3] is the result of an effort to retain the interpretation of large margins for separable cases, while producing improved performance for nonseparable cases by appropriately controlling the training errors in the objective. In the learning framework, the primary goal is to seek a classifier, *Sign(f)*. In this sense, traditional SVM seeks this goal by minimizing a combination of training error and the reciprocal of the margin, $\frac{1}{2}||w||^2$ (regularization). The general ψ-Learning is an unconstrained version of SVM as shown in (2.12).

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \psi_{SVM}\left(y_i f\left(x_i\right)\right). \tag{2.12}$$

where C represents the relative importance of training error and margin width as used in traditional SVM. ψ-Learning involves the solution of a machine learning formulation in which the loss function can be piece-wise linear, nonlinear, or discontinuous. If we use the piecewise linear function, $\psi_{SVM} = 0$ if $x \geq 1$ and $\psi_{SVM}(x) = 1 - x$ otherwise, the formulation is equivalent to traditional SVM. Also noted is that a loss function $\psi_{SVM}\left(y_i f\left(x_i\right)\right) = 1 - Sign\left(y_i\left(x_i\right)\right)$ is an undesirable cost function because any positive scaling transformation of f leaves its sign unchanged forcing $||w||$ of the solution to be 0. To correct for this outcome, and in turn attempting to maximize the distance for correctly classified training instances away from the decision boundary a limit on ψ is introduced in (2.13).

$$\begin{aligned} U \geq \psi(x) > 0 \qquad & \text{if} x \in (0, \tau] \\ \psi(x) = (1 - Sign(x)) \quad & \text{otherwise.} \end{aligned} \tag{2.13}$$

where $0 < \tau \leq 1$ and $U > 0$ are some constants. Empirical tests were performed using $U = 2$ which eliminates the scaling problem. Two variants of ψ are used, ψ_0 which is defined as 0 if $x \geq 1$, $1 - x$ if $0 \leq x \leq 1$, and 2 otherwise and ψ_1 which is defined as 0 if $x \geq 1$, $1 - x$ if $0 \leq x \leq 1$, and 2 otherwise. Because ψ_1 is continuous convex programming can be used to find globally optimal solutions, whereas, this is not possible with ψ_0. Although ψ-learning produces a nonconvex cost functions, it has very good learning properties (approaching that of a Bayes classifier) even when the solution cannot be proven to be optimal. For nonlinear problems, one can replace the decision function $f(x) = \mathbf{w} \cdot \mathbf{x} + b$ can be replaced by $\sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b$ with a kernel function. As long as Mercer's condition is satisfied we can replace the regularized term in the objective function of (2.12) with $||g||_K^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$. Instances can be solved via concave programming which involve decomposing the problem into convex and concave functions. A direct-search complex algorithm together with an initial guess is applicable. An initial guess may be chosen from other other machine learning tools such as traditional SVM or a stochastic search such as a genetic algorithm. Theory with [3] suggests that it is unnecessary to obtain the exact global minimizer as long as a reasonably good local minimizer can be found. The first simulation considered is 2-dimensional linear classification in which $f(\mathbf{x}) = \sum_{i=1}^{2} \mathbf{w}_i \mathbf{x}_i + b$. Random training samples are created $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}, y_i\}_{i=1}^{n}$ is generated. Fist $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}\}_{i=1}^{n}$ are sampled uniformly over the disk $\{(\mathbf{x}_1, \mathbf{x}_2) : \mathbf{x}_1^2 + \mathbf{x}_2^2 \leq 1\}$ and y_i is assigned to 1 if $\mathbf{x}_i \geq 0$ and -1 otherwise. Random observations have their class label flipped generating nonseparable cases. The test cases have flips of various counts; 0, 1, 2, and 10% of observations have flips each with three different values of C being 10, 10^3, and 10^7. Sample sizes for each of these cases are 25, 50, 100, 200, and 400. Results from

100 runs of each case show that ψ-learning outperforms SVM in terms of generalization error in all the nonseparable and separable cases, except when there are identical results fro large C in separable cases. Performance is shown to become more significant for large n and C, which up to a 425% improvement in error rates in the case 10%-flip with n=400 and $C = 10^7$.

Tests were also conducted on the Wisconsin Breast Cancer database (WBCD) which features the results from fine-needle aspirates taken from patients. The data is in theo form of nine-dimensional diagnostic characteristics with values in the range of 1 and 10, which lesser values representing normality and larger values indicating abnormality. The goal is to determine whether a sample is benign or malignant. Applying the ψ-learning produces improved performance over traditional SVM in 9 out of 10 cases. The case in which there is not a performance improvement, the results are the same for both ψ-learning and traditional SVM. Another positive result of the studies on ψ-learning is that the optimal parameter for C has a wider range than that of traditional SVM, indicating less training to for parameter tuning is needed.

2.2.3 JP Brooks' Ramp Loss SVM

Brooks' [2] presents formulations and empirical results from solving SVM with ramp loss error terms. These formulations are similar to traditional SVM formulation with the only difference being that indicator variables are utilized for training observations that are outside of the margin of the opposite class. These indicator variables z_i's are set to 1 if the training point is misclassified outside of the margin boundary. Any observations in which the indicator variable is utilized, the corresponding slack variable ξ_i's are zero and boundary constraints do not have to be satisfied. These changes from traditional SVMs gives us Brooks' SVMIP1(Ramp) in (2.14) in which the limit on error from each observation is limited to 2 as shown in the objective function.

$$
\begin{aligned}
\min_{\mathbf{w},b,\xi,\mathbf{z}} \quad & \tfrac{1}{2}\,||\mathbf{w}||^2 + C\sum_{i=1}^{n}(\xi_i + 2z_i), \\
\text{subject to} \quad & y_i\left(\mathbf{w}\cdot\mathbf{x}_i - b\right) = 1-\xi_i \text{ if } z_i = 0, \quad i = 1,2,...,n, \\
& 0 \le \xi_i \le 2, \qquad\qquad i = 1,2,...,n, \\
& z_i \in \{0,1\}.
\end{aligned}
\tag{2.14}
$$

.

Just like traditional C-SVM Brooks' LPSVM1 (ramp loss) formulation can accommodate nonlinear discriminants by replacing primal variables with dual variables, i.e. $\mathbf{w} = \sum_{i=1}^{n}(y_i\mathbf{x}_i\alpha_i)$, and replacing $\mathbf{x}_i$ with $\Phi(\mathbf{x}_i)$. Replacement of these variables, as well as application of the kernel trick, i.e. $\mathbf{x}_i\cdot\mathbf{x}_j \mapsto \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j) = K(\mathbf{x}_i,\mathbf{x}_j)$ gives us (2.15).

$$
\begin{aligned}
\min_{\alpha,b,\xi,\mathbf{z}} \quad & \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)\,\alpha_i\alpha_j + C\sum_{i=1}^{n}(\xi_i + 2z_i), \\
\text{subject to} \quad & y_i\left(\sum_{j=1}^{n}\alpha_j y_j k(\mathbf{x}_i,\mathbf{x}_i) - b\right) = 1-\xi_i \text{ if } z_i = 0, \quad i = 1,2,...,n, \\
& 0 \le \alpha_i \le C, \qquad\qquad i = 1,2,...,n, \\
& 0 \le \xi_i \le 2, \qquad\qquad i = 1,2,...,n, \\
& z_i \in \{0,1\}.
\end{aligned}
\tag{2.15}
$$

The conditional constraints can be linearized by replacing the right hand side with $1-\xi_i - Mz_i$ for all training vectors whether or not they are misclassified outside of the margins. The value for M should be large enough to not cut into the convex hull of integer feasible solutions. Keeping the M as small as possible is also important for reducing solution times. The method used for choosing a good M is not directly discussed in [2].

Brooks' presents solution methods that provide savings in computational time. These methods are utilized in sample classifications problems using ILOG CPLEX 11.1 Callable

Library with implementation of facet cuts of the convex hull of integer feasible solutions and heuristics for finding integer feasible solutions within the branch and bound framework. These facet cuts, created at each node in the branch and bound framework, constrain the values for an observations slack variables for a subset of training points. Implementation is done by finding points of one class that lie in the convex hull of a subset of points of the opposite class. For whatever points this scenario is true at least of the points in the set of points H utilized in the cuts problem is on the wrong side of the margin, hence $\sum_{i \in H} (\xi_i + z_i) \geq 1$ is a cut that can be added to the original formulation. Obtaining these cuts is done by solving $[CONV - SEP]$ in [2] at the nodes of the branch and cut tree.

Additionally, Brooks' uses a heuristic for finding initial feasible solutions. This heuristic is a three pronged approach: first a null solution is found, second a "zero error" solution is validated; and finally use of every positive-negative pair of observations to serve as the sole support vectors such that their conditional constraints hold at equality (i.e., they define the margin boundary). Findings presented show increased solution speed with this heuristic implemented on problems that have smaller C's and kernels with lower ranked kernels. The "zero error" solution is optimal for training cases used with a Gaussian Kernel applied. This is an indication that SVM with ramp loss is the same as traditional SVM when Gaussian or other high ranked kernels are applied to the training set.

The classification performance of SVM with ramp loss and hard margin loss is compared to traditional SVM on simulated and real-world data sets. Simulated data is sampled from Gaussian distributions, each using the identity matrix as the covariance matrix. The mean for group 1 is the origin, and the mean for group 2 is $(2/\sqrt{d}, 2/\sqrt{d}, \cdots, 2/\sqrt{d})$ so that the Mahalanobis distance is 2. The data sets are contaminated with outliers in two ways. In Type A data sets, outlier observations are sampled for group 1 using a Gaussian distribution with covariance matrix 0.001 times the identity matrix and mean $(10/\sqrt{d}, 10/\sqrt{d}, \cdots, 10/\sqrt{d})$, so that the Mahalanobis distance between outliers and non-

outliers is 10. In Type B data sets, outlier observations are sampled from both class distributions with exception that the covariance matrix is multiplied by 100. Outliers comprise 10% of the observations in the training set, and are not present in the validation or testing data sets. Using ramp loss function confers an advantage over hinged loss when using the linear kernel for all data sets tested. Support Vector Machines with ramp loss minimize the effect of Type A outliers clustered together. Using traditional SVM, the solution is influenced by Type A outliers by shifting the separating surface towards these outliers. Advantages to using a ramp loss error function is minimal when a higher-ranked kernel is used. SVM with a robust loss function outperforms SVM with hinge loss on 9 of 12, 3 of 12, and 5 of 12 data sets with degree-2 polynomial, degree-9 polynomial, and Gaussian kernels respectively. For Type B outliers using a robust loss function does not appear to confer an advantage over the hinge loss.

Nine real-world data sets from the UCI Machine Learning Repository are used as shown in Table 1. Result for all of these tests are shown in Figure 2 in [2]. The ramp losss performs at least as well as traditional SVM on 28 or 36 tests and the largest difference in misclassification rates is 4.6%. SVM with ramp loss has misclassification rates comparable to traditional SVM when applied to training set with no outliers. All problems were solved with a maximum solution time of 10 minutes, therefore the solutions presented for SVM ramp loss may still not be optimal. Therefore it may be optimal to produce ensemble classifiers with subsets of the validation data set to reduce computational time.

2.3 Human Microbiome Project

The Human Microbiome Project (HMP) is a multidisciplinary study on the microbial communities living inside humans [11]. This study is an expansion of the Human Genome Project for which it is now well known that our health and makeup of the human body are not only established through or genetics but this microbial community living inside each

Table 1. Real World Training Data Sets

Label	Name in UCI Repository	n	d
Adult	Adult	500	88
Australian	Statlog (Australian Credit Approval)	326	46
Breast[24]	Breast Cancer Wisconsin (Original)	341	9
Bupa	Liver Disorders	172	6
German	Statlog (German Credit Data)	500	24
Heart	Statlog (Heart)	135	19
Sonar	Connectionist Bench (Sonar, Mines vs Rocks)	104	60
WDBC[24]	Breast-Cancer Wisconsin (Diagnostic)	284	30
WPBC[24]	Breast-Cancer Wisconsin (Prognostic)	97	30

and every one of us [16]. In fact, the estimates of our microbes outnumber our somatic and germ cells by 10-fold. It could be also established as fact that the genetics of both the microbial communities and humanity have evolved in unison with each other.

CHAPTER 3

SVM WITH RAMP LOSS AND L_1-NORM REGULARIZATION

There are several Support Vector Machine formulations studied herein; Traditional Dual-SVM [13], Brooks' SVM with Ramp Loss [2], Mangesarian's Generalized SVM [10], Kecman's linear programming SVM [7], and Zhou's LPSVM [9]. Comparisons of the performance for the solutions to not only these formulations, but also the case studies against LP machines with ramp loss error terms. We are interested in comparing the speed of finding solution using an off the shelf mathematical programming solver in addition to comparing the resulting discriminants. The formulations are modified by weight of the error terms in accordance to their class relative frequency in the training sets in order increase the likelihood of optimal solutions as not null, i.e. $||\mathbf{w}|| \neq 0 \rightarrow \sum_{i=1}^{n} (\alpha_i y_i K(\mathbf{x}_i, \mathbf{x})) \neq 0$. Additionally, I talk about a L_1-norm SVM with ramp loss for variable selection.

3.1 Achieving Non-null Solutions

Null solutions can be a problematic result of SVM with unequal frequency for classes in training. To avoid this outcome, we weight the error terms in the objective function quantized by relative frequency of class label. Cases in which one class label group is larger than the other can result in null solutions often, therefore implementing a weight for making this outcome less probable is advantageous. Requirements for the weights are as follows, the loss parameter C must maintain meaning by setting the average weight equal to 1 as shown in (3.1).

$$\min_{\mathbf{w}} \quad \tfrac{1}{2} ||\mathbf{w}||^2 + C \left(\frac{n}{2n^+} \sum_{i \in I^-} \xi_i + \frac{n}{2n^-} \sum_{i \in I^+} \xi_i \right). \tag{3.1}$$

The weights of the errors according to class label should be $n/2n^+$ and $n/2n^-$ for positive and negative training observations respectively, with n, n^+, and n^- are the cardinality of all, positive and negative training entities respectively. And the I^- and I^+ are the set of negative and positive training entities respectively, i.e. $I^- \in \{1, 2, \cdots, n | y_i = -1\}$ and $I^+ \in \{1, 2, \cdots, n | y_i = 1\}$.

These same weights can be utilized for all of the SVM formulation discussed herein with the exception of traditional dual SVM because slack variables, ξ's, are not present in the objective function. The new SVM models are discussed without class size based weights on ξ's even though these weights were added to the models in the analysis of each models performance.

3.2 Generalized SVM with Ramp Loss

Mangasarian [10] does not provide strong evidence of the strength of prediction of GSVM, but the simple idea may have properties that produce quality robust discriminates for difficult classification problems quickly. Herein we study and compare the performance of Mangasarian's LP GSVM (2.2) with the same models formulated with ramp loss error terms. The modified GSVMRL model, GSVM-RL (3.2), is this Mangasarian's LP-GSVM in (2.4) with binary variables from Brooks' SVMIP(RAMP).

$$\begin{aligned} \min_{\mathbf{u},b,\xi,\mu} \quad & f(\mathbf{u}) + \sum_{i=1}^{n} (\xi_i + 2z_i), & \\ \text{subject to} \quad & y_i \left(\sum_{j=1}^{j=n} y_j K(\mathbf{x}_i, \mathbf{x}_j) u_j + b \right) \geq 1 - \xi_i + M z_i, & i = 1, 2, \cdots, n. \\ & \xi_i \geq 0 & i = 1, 2, \cdots, n. \\ & z_i \in \{0, 1\} & i = 1, 2, \cdots, n. \end{aligned} \tag{3.2}$$

where $f(\mathbf{u})$ is typically some norm or seminorm of the dual variables[10]. The sim-

plest choice of f is the L^1-norm of the dual variables, u_i's. This is shown in (3.3) with s_i being the absolute value of u_i. This is done by replacing f in the objective with $\sum s_i$ and adding constraint 2 in (3.3) which bounds the value to s_i to the magnitude of u_i. The value of M is important because making it too small may bias the resulting discriminant towards outliers. On the other hand making M too large may mean making the branch and bound tree unmanageable. An option for controlling the size of M may be to solve the linear problem for the magnitudes of the ξ's, and adjusting M as necessary. It may be noticed that unlike traditional dual SVM, GSVM variables are not sign restricted, and therefore can produce highly nonlinear solutions; depending on the problem this trait for the problem statement may be a benefit or a detriment.

$$
\begin{aligned}
\min_{\mathbf{u},\mathbf{s},b,\xi,\mu} \quad & \sum_{i=1}^{n} s_i + \sum_{i=1}^{n} \left(\xi_i + 2z_i\right), \\
\text{subject to} \quad & y_i \left(\sum_{j=1}^{j=n} y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) u_j + b \right) \geq 1 - \xi_i + M z_i, \quad i = 1, 2, \cdots, n. \\
& s_i \geq u_i \geq -s_i, \qquad\qquad i = 1, 2, \cdots, n. \\
& \xi_i \geq 0 \qquad\qquad i = 1, 2, \cdots, n. \\
& z_i \in \{0, 1\} \qquad\qquad i = 1, 2, \cdots, n.
\end{aligned}
\tag{3.3}
$$

Another possible substitution of f is the magnitude of $\sum_{j=1}^{n} \left(y_j K\left(x_i, x_j\right) u_j\right)$ as shown in (3.4). If the kernel is linear then it is interesting to note that we are minimizing the $(\mathbf{w} \cdot \mathbf{x_i})$'s. In this way, the function that we are minimizing parallels the L^2-Norm traditional SVM more closely than (3.3).

$$
\begin{aligned}
\min_{\mathbf{u},\mathbf{s},b,\xi,\mu} \quad & \sum_{i=1}^{n} s_i + \sum_{i=1}^{n} \left(\xi_i + 2z_i\right), \\
\text{subject to} \quad & y_i \left(\sum_{j=1}^{j=n} y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right) u_j + b \right) \geq 1 - \xi_i + M z_i, \quad i = 1, 2, \cdots, n. \\
& s_i \geq \sum_{j=1}^{n} \left(y_j K\left(x_i, x_j\right) u_j\right) \geq -s_i, \qquad i = 1, 2, \cdots, n. \\
& \xi_i \geq 0 \qquad\qquad i = 1, 2, \cdots, n. \\
& z_i \in \{0, 1\} \qquad\qquad i = 1, 2, \cdots, n.
\end{aligned}
\tag{3.4}
$$

The separating surface for (3.3) and (3.4) are the same as the linear programming formulations defined in (2.3) and traditional SVM in (1.20). Predicting the class of new observations is simply defined as sign$(\sum_{i=1}^{n} \left(y_i K\left(\mathbf{x}, \mathbf{x_i}\right) u_i\right) + b)$.

3.3 V. Kecman and I. Hadzic LP SVM with Ramp Loss

Another formulation for LPSVM is discussed in Section 2.1.3 is V. Kecman and I. Hadzic's highly effective linear programming classifier [7]. The model (3.5) produces the linear discriminant that parallels (2.5). This is the model called L1-Norm RLSVM most closely resembles comparing all the models present. It is important to note the that resulting discriminant is not the same as for traditional SVM, because the regularization terms have changed; we want to minimize $|\mathbf{w}| = \sum_{i=1}^{n} \left(w_i^+ + w_i^-\right)$ instead of $||\mathbf{w}||$ in order to modify the traditional SVM problem into a linear program. In the new case the discriminant is the same as it is for traditional SVM $sign\left(\mathbf{w} \cdot \mathbf{x} + b\right)$, where $\mathbf{w}^+ - \mathbf{w}^- = \mathbf{w}$. The obvious complication to the SVM model is that there will be an additional n constraints,

$$
\begin{aligned}
\min_{\mathbf{w}^+,\mathbf{w}^-,b,\xi} \quad & \sum_{i=1}^{n} \left(w_i^+ + w_i^- + C\xi_i + 2Cz_i\right), \\
\text{subject to} \quad & y_i \left(\mathbf{x}_i \cdot \mathbf{w}^+ - \mathbf{x}_i \cdot \mathbf{w}^- + b\right) \geq 1 - \xi_i - Mz_i, \quad i = 1, 2, \cdots, n. \\
& \mathbf{w}^+, \mathbf{w}^- \geq 0. \\
& \mathbf{z} \in \{0, 1\}
\end{aligned}
\tag{3.5}
$$

This is formulation is the closet literal transition from L_2-Norm regularization of SVM to L_1-Norm with ramp loss error terms.

3.4 Zhou's SVM with Ramp Loss

Weida Zhou [9] come up with an ingenious method for producing linear programming SVM models with a slight tweak to the regularization objective. We will call this RSVM from this point forward. In traditional SVM we want to maximize the distance between two parallel planes, $2/\left|\left|\mathbf{w}\right|\right|$, in which 2 is arbitrary to what's being optimized. Zhou instead says that 2 can be replaced by a decision variable r, so that we are instead maximizing, $r/\left|\left|\mathbf{w}\right|\right|_\beta$ in which β is signifies any vector norm.

This is implemented in a SVM model by adding this new decision variable r to the objective by minimizing negative r and replacing 1 in the right hand side of the support vector constraints with r as shown in (3.6). This does make the problem unbounded with respect to $\mathbf{w}$ in the primal problem or α in the dual problem. This problem is solved by bounding theses constraints to a magnitude of 1, i.e. $-1 \leq \alpha_i \leq 1$.

$$
\begin{aligned}
\min_{\alpha,b,r,\xi} \quad & -r + C\sum_{i=1}^{n}(\xi_i + 2z_i) \\
\text{subject to} \quad & y_i\left(\sum_{j=1}^{n}\alpha_j y_j K(\mathbf{x}_j,\mathbf{x}_i) + b\right) \geq r - \xi_i - Mz_i, \quad i = 1,2,\cdots,n. \\
& -1 \leq \alpha_i \leq 1 \qquad\qquad i = 1,2,\cdots,n. \\
& r \geq 0 \\
& \xi_i \geq 0 \qquad\qquad i = 1,2,\cdots,n. \\
& z_i \in \{0,1\} \qquad\qquad i = 1,2,\cdots,n.
\end{aligned}
\tag{3.6}
$$

The formulation shown in (3.6) is the only RSVM model with ramp loss tested herein.

3.5 L_0-norm SVM

One of the main sources of discontent with SVM and many other predictor based statistical models is selection of variable that are best for improving over all accuracy and effectively minimizing noise. That is why, an SVM formulation in which a generalization of the limit on the number of parameters would be highly effective establishing which variable produce the best performing classifier. We call this formulation RLSVM++, it is shown in equation (3.7)

$$
\begin{aligned}
\min_{\mathbf{w},b,\xi,\mathbf{z},\gamma} \quad & \sum_{i=1}^{n}(\xi_i + 2z_i) \\
\text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i - Mz_i, \quad i = 1,2,\cdots,n. \\
& -M\gamma_i \leq w_i \leq M\gamma_i \qquad i = 1,2,\cdots,n. \\
& \sum_{i=1}^{n}\gamma_i \leq k \qquad\qquad i = 1,2,\cdots,n. \\
& \xi_i \geq 0 \qquad\qquad i = 1,2,\cdots,n. \\
& z_i, \gamma_i \in \{0,1\} \qquad i = 1,2,\cdots,n.
\end{aligned}
\tag{3.7}
$$

The new variable introduced in this SVM methodology, γ, is an indicator variable

which is 1 when when the corresponding coefficient is used for the resulting separating surface. And γ is bounded by the training variable k, which produces an upper limit on attributes that influence the boundary. Because of this new training variable and application of ramp loss with in the context of the problem, the trade-off of margin width and error is deemed to add unnecessary complexity to the problem. The trade-off can simply be added back into the problem set or other classification methods can be implemented on the variable selected data set. Even though RLSVM++ may be the most useful statistical model discussed in this paper, it was not tested herein.

CHAPTER 4

DATA SETS

4.1 Microbiome Data

A multinational and multidisciplinary project has been started to increase humanities understanding of the communities of microorganisms living inside the human body. [11] There are several specific sites of interest in which these tiny creature can greatly influence the health of individuals for both good and bad. The questions being asked about these microorganism vary from dietary to genetic. It may be possible for a physician to look at an individuals microbiotic community, then come up with a valid cause quickly and solution of physical alignments for long term health. In this particular study we are focusing our attention on two sites being studied, the throat and the tongue. We would like to see if we can differentiate from which the microbial communities originated, either the tongue or throat of individuals using Support Vector Machines described in Chapter 3.

This microbiotic data was compiled at Professor Patrick Schloss' Lab at the University of Michigan. Of this data set, there are 582 unique observations; 277 from the tongue and 305 from the throat. Figure 5 is a stacked bar chart that shows the diversity and similarity of the microorganism found in theses samples (it is unknown which half of the figure represents the communities on the throat and which half is from the tongue, but, but the two are separated in halves as can be seen by sorting by percentage of Corynebacterium, "dark orange", in the sample [17]. This image is the basis for considering this data set as a quality instance for evaluating our SVM models. We did some data cleaning prior to testing the data, additional duplicates were discarded and of the 604 Phylotypes, 111 were left after discarding the scarcely recognized types. The data is converted to proportions to diminish

the influence of more frequently recorded phylotypes have on the classifiers and as a way of standardizing the individual samples which is necessary because of the variance in reads between these samples.

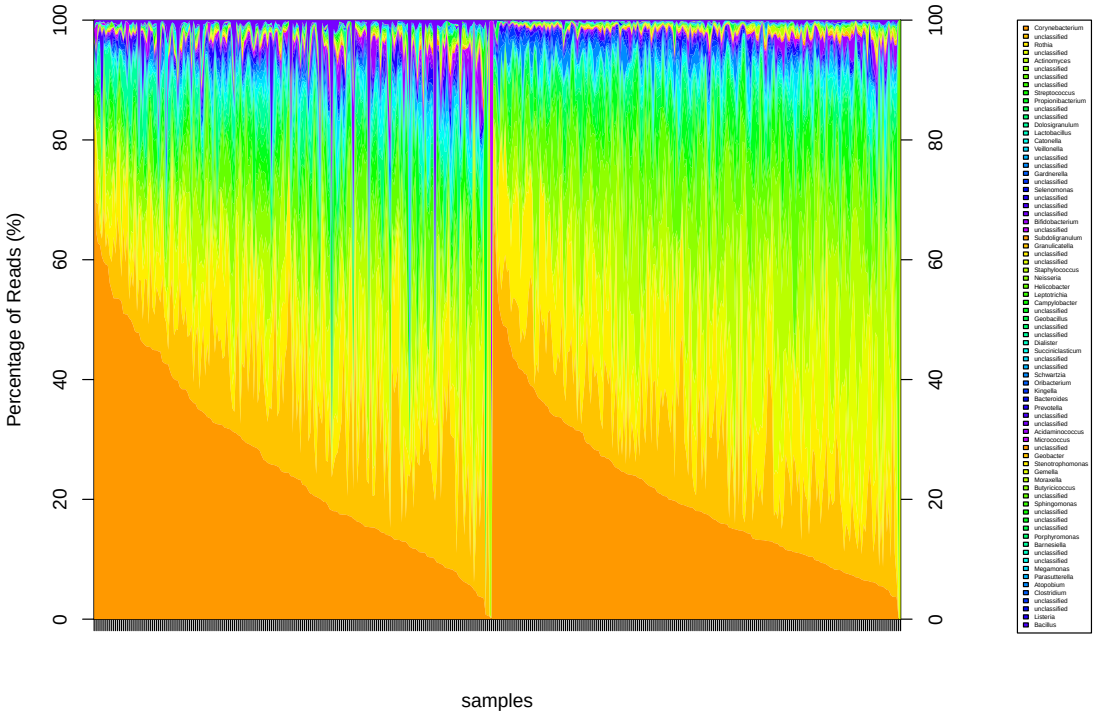

Fig. 5. Stacked Bar Chart for the percentage of reads for phylotypes taking from samples from study participant's throat and tongue.

4.2 Simulated Data

We also used some two-group simulated data sampled from Gaussian distributions with the identity matrix as the covariance matrix, which is taken from [2]. Brooks cites the *mvtnorm* package in the R programming language and environment for statistical computing as his resource for creating these samples. Data is primarily comprised of two-groups both are randomly created using a mean for group 1 as the origin and the mean for group

2 as $\left(2/\sqrt{d}, 2/\sqrt{d}, \cdots, 2/\sqrt{d}\right)$, so that the Mahalanobis distance between the two groups is 2, a classic Breiman's "twonorm" benchmark model. Noise is intentionally added to create a problem set that could alter the performance of the decision linear boundary found using a traditional SVM model. It is mentioned in [2] that the *Bayes rule* for the (non-contaminated) distributions places observations in the group for which the mean is closest, therefore the Bayes error is $P(z > 1) \approx 15.87\%$, where $z \sim N(0, 1)$. The first data sets, which we call *Type A*, noise is created with observations sampled from both class distributions with the exception that the covariance matrix is multiplied by 100. In *Type B* data sets, provide noise in the form of outlier observations randomly created using a Gaussian distribution with covariance matrix 0.001 times the identity matrix and mean of $\left(10/\sqrt{d}, 10/\sqrt{d}, \cdots, 10/\sqrt{d}\right)$, so that the Mahalanobis distance between outliers and non-outliers in 10. Outliers comprise 10% of the observations in the training set, and are not present in the testing data set. The test sets are comprised of combinations of sizes in terms of observations and variables. For each data set type we have made training and validation data set for ever combination of 60, 100, 200, 500 observations and 2, 5, 10 variables. Our first instinct is to believe that the smaller test set such as the ones with 60 observations and 2 variables will result in quicker but worse solutions, where as, the larger test sets such as the ones with 500 observations and 10 variable will result in slower but better solutions. This may be true for some formulations but not others. The next section provides a discussion of the results from all of these tests.

CHAPTER 5

RESULTS

5.1 Computing

All Support Vector Machine instances were created and solved using Gurobi 5.0 C API on a machine with an Intel Core 2 Duo CPU E8400 3.00GHz with 4 GB RAM. Training instances were each allowed 20 seconds to find the best solution within the branch and bound tree with no cuts allowed. In the validation phase, each formulation given best parameter settings was given at most fifteen minutes to find the best solution.

The user parameter **M** was set to 10 for each instance to create an environment in which most or all possible solutions are feasible. A larger value for **M** is usually recommended but under the gurobi parameter settings utilized, a large **M** can produces small non-integer solutions for some of these discrete variables **z**. Any time ξ_i is nonzero then z_i should be zero and visa versa. These small fractional solutions for **z** break this rule therefore making the model invalid. It is recommended to test for these type of constraint errors during the test phase of classification. A methodology for selecting quality **M** values has not been included in this document.

Methodologies outside of the branch and bound, and linear programming framework were not tested, but it is not outside of reason that other untested algorithms will converge on the optimal solution more quickly and more robustly (no infeasible solutions i.e. $\xi_i > 0$ and $z_i > 0$).

5.2 Microbiome Test

As it turns out, distinguishing from which body site (tongue or throat) is not as difficult as it may appear to be from the quantity of type of organisms and similarity in proportions of these organisms on these body sites as shown in Figure 5. The validation results for testing the quality each formulation are shown in Tables 2, 7, and 8. The formulation key can be found in Table 3 where formulations 1 and 2 are Brooks' formulations for SVM with Ramp loss [2]; formulations 3 and 4 are O.L. Mangasarians Generalized SVM [10] with ramp loss; formulations 5 and 13 are Kecman's linear programming SVM [7] with and without ramp loss respectively; formulations 6 and 14 are Zhou's linear programming SVM [9] with and without ramp loss respectively; formulation 10 is the traditional dual SVM [13]; formulations 11 and 12 are Mangasarian's linear generalized SVMs.

Table 2.: Results for Linear Kernel on Saliva-Throat Phylotype Data

Formulation	C	Time	Status	Accuracy	StdError	LCL	UCL
1	1000	1034.744	Best	0.937	0.0183	0.901	0.973
2	10000	14.956	Best	0.926	0.0198	0.887	0.965
3	10	153.258	Best	0.914	0.0212	0.873	0.956
4	1000	862.003	Best	0.931	0.0191	0.894	0.969
5	10	16.881	Optimal	0.891	0.0235	0.845	0.938
6	0.01	36.914	Optimal	0.903	0.0224	0.859	0.947
10	10000	0.207	Optimal	0.926	0.0198	0.887	0.965
11	10	6.273	Optimal	0.903	0.0224	0.859	0.947
12	1000	2.062	Optimal	0.903	0.0224	0.859	0.947
13	100	0.116	Optimal	0.909	0.0218	0.866	0.951

Table 2.: *(continued)*

Formulation	C	Time	Status	Accuracy	StdError	lcl	ucl
14	10	1.054	Optimal	0.886	0.0241	0.839	0.933

The validation and test set are comprised of the 582 unique observations containing proportions of the 111 microbiotic phylotypes (selected from data cleaning) taken from Professor Patrick Schloss' Lab. Of these 407 and 175 comprised the validation and test sets respectively. It is interesting to note that all classifier's 95% confidence intervals on accuracy overlap one another. This is true for each pair of formulations in this test. These 95% confidence intervals on the produced discriminant are calculated using the equation for confidence intervals on proportions. There is at least one tested methodology for calculating confidence intervals on SVM models by B. Jiang et al., [18].

The time limit for instances with linear kernels was set to 25 minutes, where as only 5 minutes were allowed for instances with Gaussian or polynomial kernels. It is noted in [2] that high dimensional kernels have properties that make the solution for the ramp loss SVM converge to the solutions for traditional hinge loss SVM. This is shown in Tables 7 and 8, in which the solution for Mangasarian's LP SVM formulation (2.2) produces the same result as the same formulation with hinge loss (3.3). Additionally it can be seen that Brooks' CSVM with ramp loss (2.15) and Mangarian's LP SVM formulation with ramp loss (3.4) both have Gaussian kernel instances with the same parameters ($C = 100$, $\gamma = 1$) that converge on the solution of the traditional SVM formulation (1.15). Although these formulations do not produce the same separating surface exactly, the surface classifies the same points correctly and incorrectly.

One of the goals for a new formulation is that that be faster if not significantly faster than Brooks' SVMIP1 (2.14) and SVMIP2 (2.15). Speed to the optimal or best solution is

are significantly different for each formulation. The only MIP formulation that were able find optimal solution for the linear kernel are Kecman's formulation with ramp loss (formulation 5) at under 17 seconds versus the next best Zhou's SVM with ramp loss (formulation 6) at just under 37 seconds. Optimality for resulting linear separating surfaces did not impact classification performance as the best performer was Brooks' SVMIP1 (2.14) with an accuracy of 9.37% in a maximum time for finding a best incumbent solution of 1034 seconds. Mangasarian's linear SVM models with ramp loss (formulation 3 and 4) prove substantially more likely to find better incumbent solutions the longer the model runs. The best performing models for these formulations 3 and 4 were above 91% with a reasonable assumption that the finally incumbent solution are in fact the optimal solutions, then we can use this as evidence that ramp loss will improve these SVMs.

The improvement in efficiency should not mean we sacrifice the performance improvement over the SVMs with hinge loss as the regularization term in the objective. This increase in performance is not found in the test to classify body site from microorganism data, with that performance increase being due to a shift and rotation of the discriminant away from outliers decreasing the magnitude of an effect on the result. The best comparison we have in models are Kecman's linear programming SVM formulation (13) in equation (2.5) and same model with ramp loss error terms (5) in equation (3.5)(formulation 5 and 13 can only produce linear discriminants). Of the 111 phylotypes modeled only 22 and 19 are found in the solutions to formulations 13 and 5 respectively (i.e. $w_i^+ - w_i^- \neq 0$), with 13 of these phylotypes in both. Instead of the shift or rotation that we would expect to see for the new models, we see that these models can select a new set of variable to be important to the model. This may seem inconsequential, but an outlier may only be an outlier due to one or very few variables relative to an abundance of variables. And because of this we may see an increase in performance when we add ramp loss to SVM models due to a form of unanticipated variable selection by the model. This is not the case for the SVMs with

ramp loss applied to the simulated data with outliers of Type B because these outliers are applied in all directions. This we see in the following section.

5.3 Simulated Data

Training on the simulated data is described in Appendix B with results from parameter tuning each model are in Table 9 and 10. The validation results are in Table 11 and Table 12, which are left out of this section due to the size of the tables. We are able to compare the 2-dimensional simulated data results in scatter plots shown in Figures 6, 7, 8, 9, and 10. As shown in [2], these 2-dimensional data sets are where SVM with ramp loss shine in comparison to traditional SVM models. With type A outliers Brooks' outperforms standard dual SVM for 3 of the 4 2-dimensional test sets, with the one exception being the largest test set with 500 observations. There are only three other test sets that the traditional dual SVM model outperforms Brooks' models. Brooks' ramp loss SVM models do much better than traditional SVM when considering the type B outliers. In fact, for all but three of the type B outlier training sets, running the dual SVM model results in the null solution, with the other three surfaces being shifted far from the optimal separating boundary. Brooks' SVMIP1 and SVMIP2 find solutions that are all in between 80% and 85% accuracy with only 3 of the solutions proven to be optimal.

Figures 7 and 8 display the result from Mangasarian's generalized SVM formulations with and without ramp loss. Type A outliers seemed to make little difference between the formulations with and without ramp loss. The results are mixed because both formulations with and without ramp loss had instances in which the null solution was the best one found. When the L1-Norm of the dual variables, $|\mathbf{u}|$, is used as the regularization term the results for the model with hinge loss is null for 60 observations, as well as, the one with 100 observations when we applied ramp loss error. Other than these null solutions and three additional, the outliers play little role in the solution for instances with Type A outliers.

Only six instance (which ramp loss was applied and Type A outliers added) conducted left an accuracy under 80%, leaving a total of 37 test achieving accuracies above 80% on the data set with Type A outliers. The Type B outliers again have a significant influence on the result of non robust SVM. The tests conducted on 2-dimensional data shown in Figure 7 provide evidence to suggest that the separating surfaces from Mangasiarian's LPSVM do not shift over towards the outliers but instead rotate about the center of the main cluster of data. The separating surface does this on four of the eight 2D test conducted, where as, when ramp loss is applied the separation boundary appears to be consistently near optimal. Again, null solutions are scattered throughout these results. There were seven null solutions for Mangasiarians LPSVM with hinge loss and five with ramp loss. Of the 24 test on the Mangasiarian's formulation with ramp loss and Type B outliers 19 were above 80% accuracy. Only 8 of the 24 tests conducted on the Mangasiarian's LPSVM performed better that 80%.

Performance of Kecman's linear programming SVM, equation (2.5), can be seen in Figure 9. As discussed in the previous section on the classification of body sites based on phylotypes sampled, Kecman's formulation has a way of finding optimal solutions that leave out certain variables as a sort of quasi variable selection. This is shown in the scatter plots with 100 and 500 observations and Type B outliers in the data. In both of these figures and for both cases in which ramp loss and hinge loss is used as the error term in objective, the second variable (y on the graph) does not influence the boundary of the separating surface. This model may need to be studied for this benefit specifically, even though this data is constructed in a way in which removing the effects of any variable will harm the performance. In comparing the results on Kecman SVM models, adding ramp loss seems to make the model more influence by the outliers. Figure 9 shows the boundary of the separating surface rotated and widened towards these outliers in all the plots with Type A outliers added. This may be the one case in which not only does ramp loss not benefit

the SVM but instead harms the performance. None of the Kecman SVMs with ramp loss achieved a performance of above 80% and six out of the twelve were above a threshold of 70%. One of the test conducted with Kecman's linear programming SVM achieved above 80% accuracy.

When we look at the performance of Zhou's SVM model, we see very strange results as shown in Figure 9. The objective is to minimize the reciprocal of boundary variable r and the error, but this results in a value of zero, with zero error and zero for r. There may be an issue with either the computation of this model or an error in the model itself. This happens in 18 of the models that do not include ramp loss error and 19 of the models that do. This even occurs when we can see that some of the observations are obviously misclassified in Figure 9, as with the green separating surface in the top left figure which is the boundary from the ramp loss model should perfectly split the data. This is obviously not the case. More work needs to be put in to find out if this is just a natural result of the model or if it is a miscalculation. Although the performance testing is not optimal for both the Kecamn and Zhou LPSVM formulations with ramp loss, it appears that the formulations do solve much more rapidly than the ramp loss counter parts.

We begin to see the gains that can be made in time commitment in both testing and validation of SVM with ramp loss as we look at Figure 11. This figures shows that for each and every simulated test the ramp loss models for both Kecman (5) and Zhou (6) are dramatically faster than Brooks' SVMIP1 and SVMIP2. For each test these formulations were solved in under one second, where as, the other formulations time to best solution was up to 10,500 seconds.

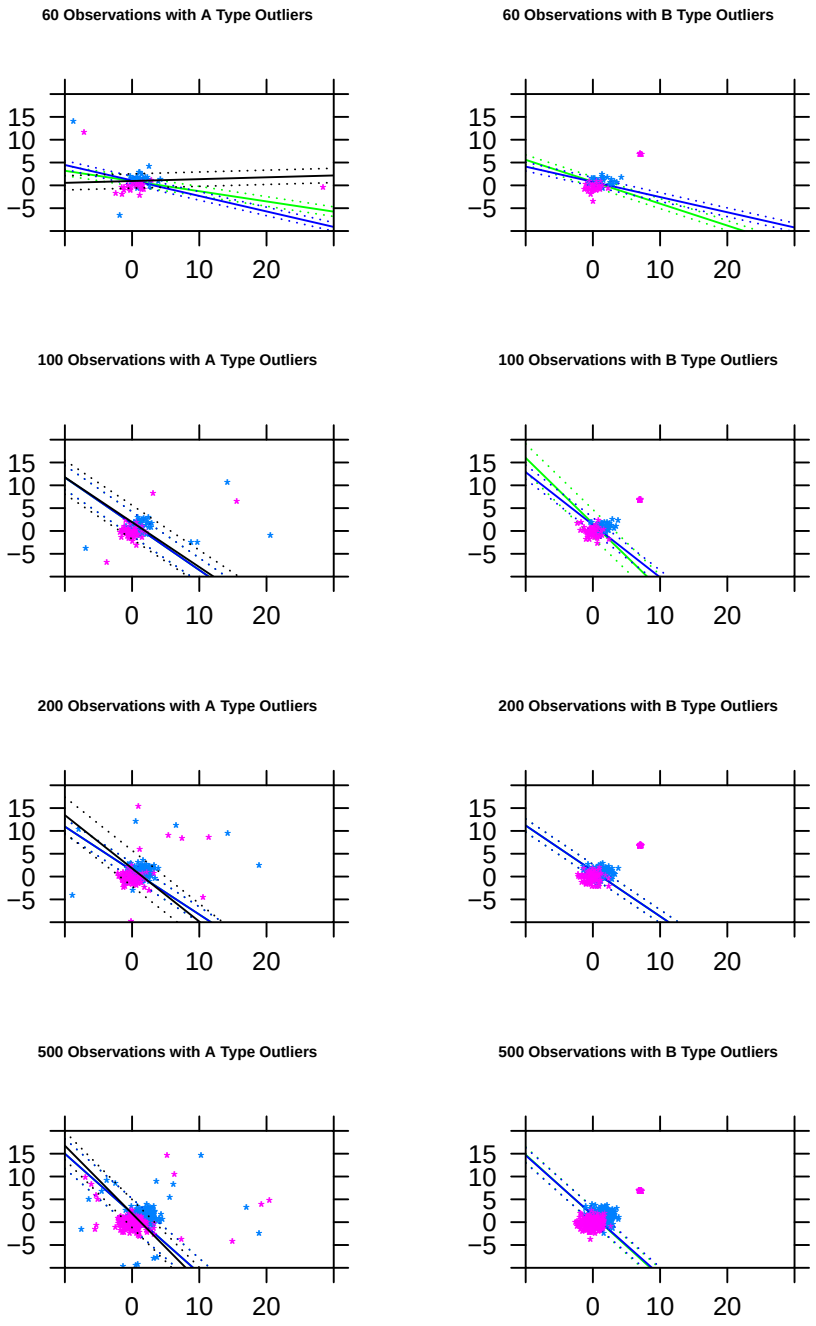

Fig. 6. The separating surfaces from three different models. The green line represents the boundary from the Brooks' SVMIP1 (2.14) surface, blue from the Brooks' SVMIP2 (2.15) surface, and black from the dual SVM model.

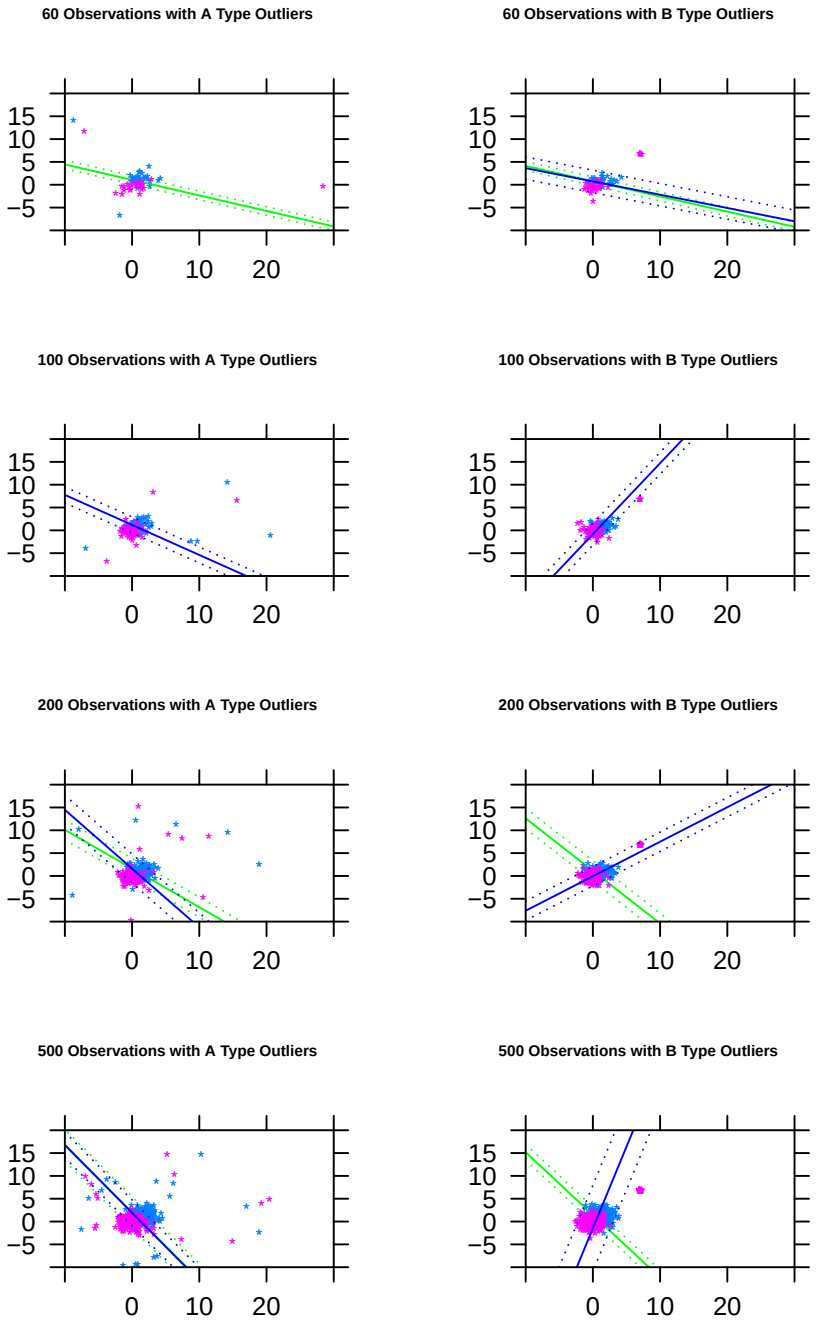

Fig. 7. The separating surfaces from two different models. The green line represents the boundary from Mangasarian's linear SVM formulation (2.4) regularized by $|\mathbf{u}|$ in blue and equivalent formulation but with ramp loss in green.

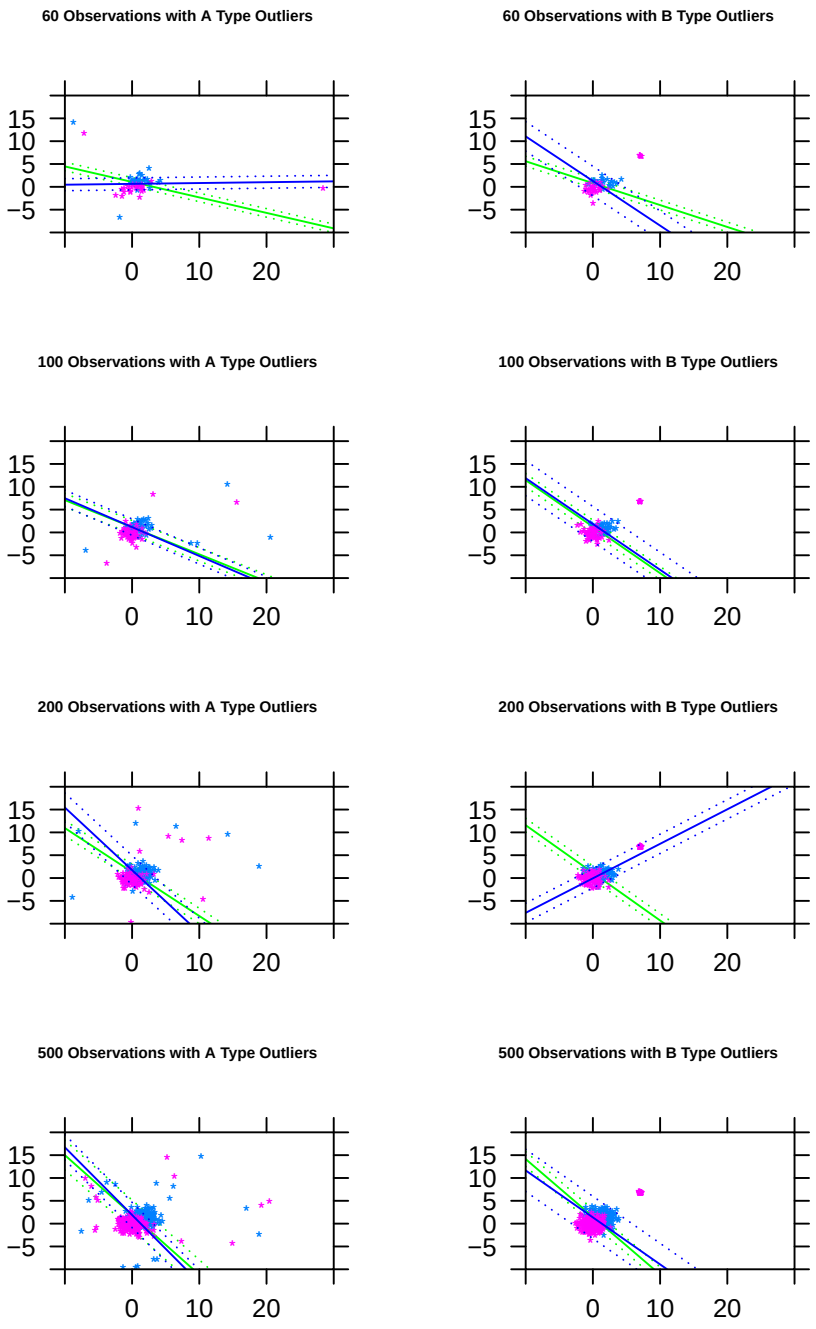

Fig. 8. The separating surfaces from two different models. The green line represents the boundary from Mangasarian's linear formulation (2.4) regularized by $\left|\sum_{j=0}^{n}\sum_{i=0}^{n}\left(y_i K\left(\mathbf{x}_i,\mathbf{x}_j\right)y_i\right)\right|$ in blue and equivalent formulation but with ramp loss in green.

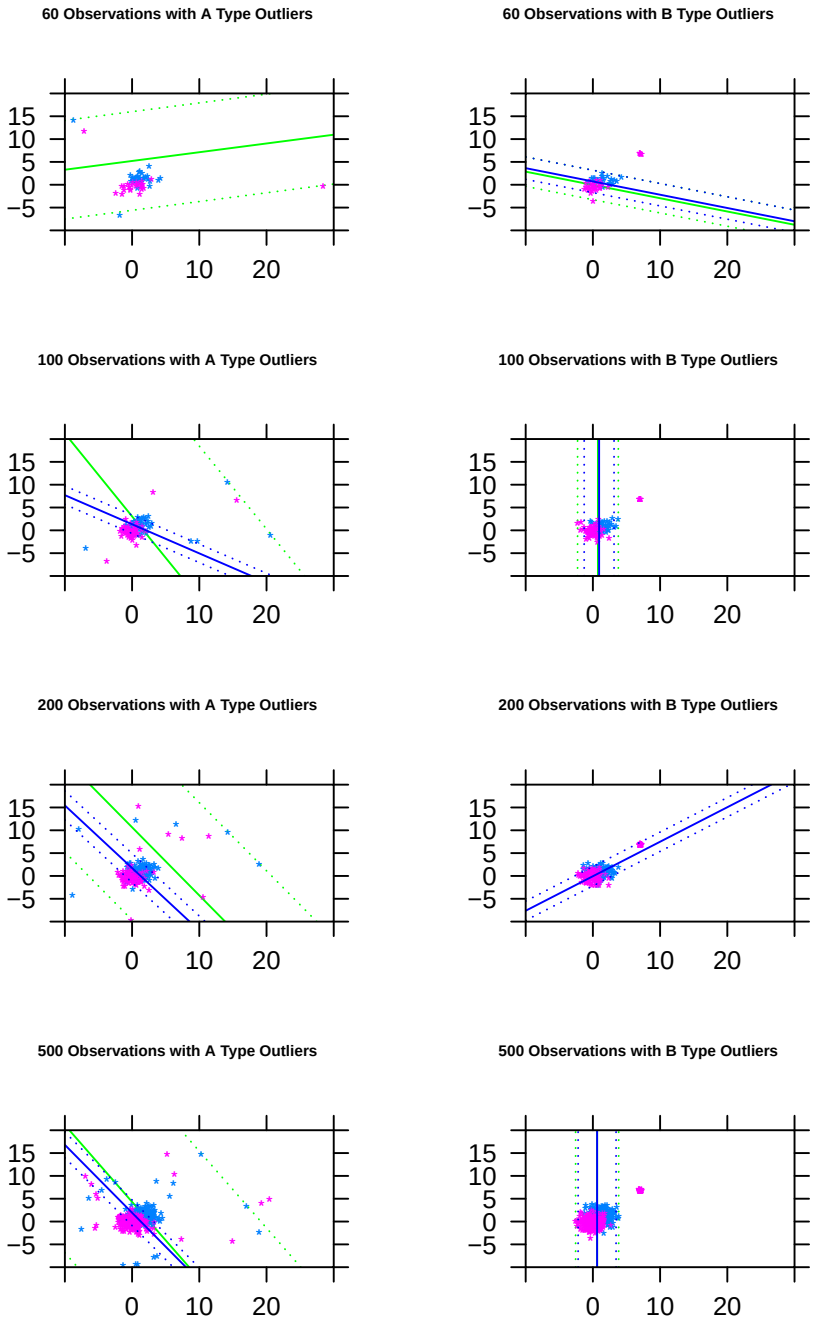

Fig. 9. The separating surfaces from two different models. The green line represents the boundary from Kecman's linear formulation (2.5) in green and equivalent formulation but with ramp loss in green.

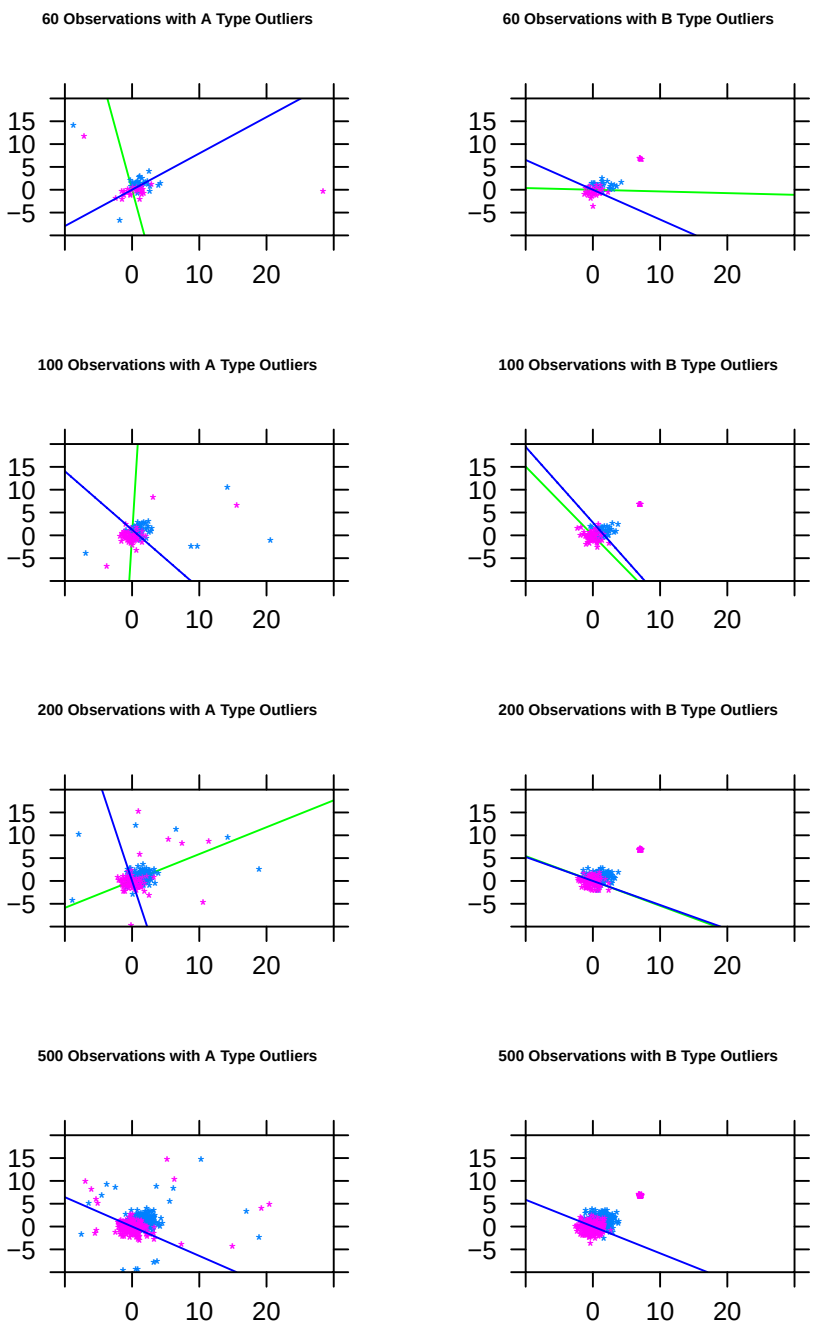

Fig. 10. The separating surfaces from two different models. The green line represents the boundary from Zhou's linear formulation (2.9) in green and equivalent formulation but with ramp loss in green.

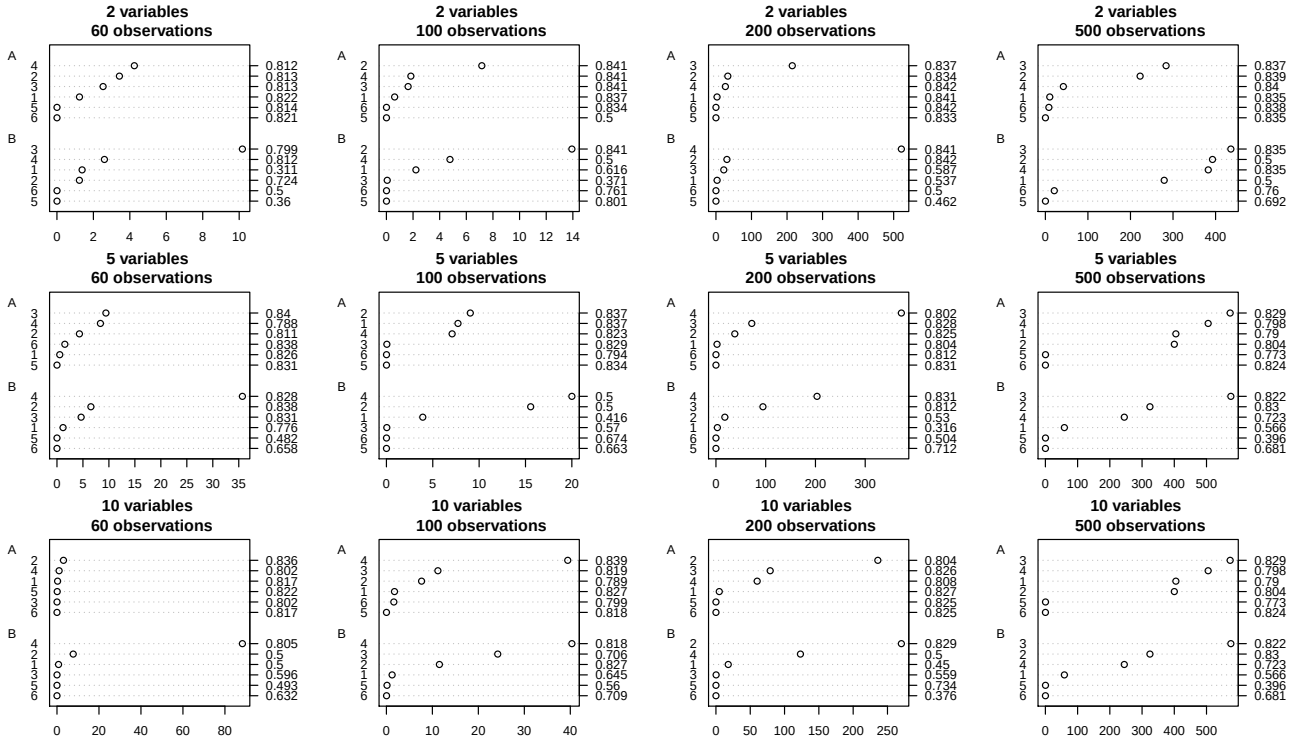

Fig. 11. Dot plots of the time in seconds to find the best solution to each mixed-integer classification problem. The plots are grouped by outlier class and sorted from longest to shorts time for finding the best solution.

Appendix A

CROSS VALIDATION RESULTS ON MICROBIOME DATA

Table 3 list the results from parameter tuning of the support vector machines from 582 observations of microbiome data described in Chapter 4. This data waw split up at random into testing and validation sets first in which 30% of the data (175 observations) left for testing, leaving 407 observations for validation and parameter tuning. The performance from the 10-fold cross-validation are shown in follow tables: Table 4 for linear kernel or primal problems starting on 56; Table 5 for the Gaussian kernel problems starting on 57; and Table 6 for the Polynomial kernel problems starting on 57. Each of these tables has a column with a formulation number on it; each of these numbers correspond to a Support Vector Machine listed in Table 3.

Table 3. Microbiome Throat vs. Tongue Paramater Tuning Results

Support Vector Machine	Equation Reference	Cross Reference	Paramaters						
			Linear	Gaussian		Polynomial			
			C	C	γ	C	α	β	π
Brooks' Ramp Loss SVM w/ Primal Variables	(2.14)	1	1000	N/A					
			90.9%						
Brooks' Ramp Loss SVM w/ Dual Variables	(2.15)	2	10000	100	1	0.1	1	5	6
			90.2%	90.2%		90.4%			
Ramp Loss Generalized SVM Regularized by $\|\mathbf{u}\|$	(3.3)	3	10	100	10	1	5	1	6
			86.7%	82.8%		79.9%			
Ramp Loss Generalized SVM Regularized by $\sum_{j=1}^{n}\left(\sum_{i=1}^{n}\left(y_i K\left(\mathbf{x}_i, \mathbf{x}_j\right) u_i\right)\right)$	(3.4)	4	1000	100	1	1	5	100	2
			89.9%	88.5%		89.9%			
Ramp Loss Kecman SVM	(3.5)	5	10	N/A					
			89.2%						
Ramp Loss Zhou SVM	(3.6)	6	0.01	0.01	1	0.01	1	0	2
			88.5%	87.7%		83.5%			
Traditional Dual SVM	(1.5)	10	10000	100	1	0.01	0.1	10	6
			90.2%	88.7%		90.4%			
Generalized SVM Regularized by $\|\mathbf{u}\|$	(2.4)	11	10	1	10	1	5	1	6
			84.5%	82.8%		79.9%			
Generalized SVM Regularized by $\sum_{j=1}^{n}\left(\sum_{i=1}^{n}\left(y_i K\left(\mathbf{x}_i, \mathbf{x}_j\right) u_i\right)\right)$	(2.4)	12	1000	100	1	10	5	10	2
			90.2%	88.7%		91.2%			
Kecman SVM	(2.5)	13	100	N/A					
			91.2%						
Zhou SVM	(2.9)	14	10	0.01	0.1	0.1	1	0	6
			84.5%	88.0%		84.8%			

Table 4.: Cross-Validation Results for Linear Kernel on Saliva-Throat Phylotype Data

Formulation	C	truepos	falsepos	trueneg	falseneg	total	Accuracy	StdError	LCL	UCL
1	10000	178	20	191	18	407	0.907	0.0144	0.878	0.935
2	10000	21	1	16	3	41	0.902	0.0463	0.812	0.993
3	10	131	21	152	22	326	0.868	0.0187	0.831	0.905
4	1000	178	18	193	18	407	0.912	0.0141	0.884	0.939
5	10	183	40	171	13	407	0.870	0.0167	0.837	0.902
6	0.01	178	25	186	18	407	0.894	0.0152	0.864	0.924
10	100	178	18	193	18	407	0.912	0.0141	0.884	0.939
11	10	170	31	180	26	407	0.860	0.0172	0.826	0.894
12	1000	180	20	191	16	407	0.912	0.0141	0.884	0.939
13	100	181	17	194	15	407	0.921	0.0133	0.895	0.948
14	0.01	178	25	186	18	407	0.894	0.0152	0.864	0.924

Table 5.: Cross-Validation Results for Gaussian Kernel on Saliva-Throat Phylotype Data

Formulation	C	Gamma	truepos	falsepos	trueneg	falseneg	total	Accuracy	StdError	LCL	UCL
2	100	1	183	21	190	13	407	0.916	0.0137	0.890	0.943
3	10	10	167	34	177	29	407	0.845	0.0179	0.810	0.880
4	100	10	172	22	189	24	407	0.887	0.0157	0.856	0.918
6	0.01	0.1	178	26	185	18	407	0.892	0.0154	0.862	0.922
10	100	1	180	17	194	16	407	0.919	0.0135	0.892	0.945
11	10	10	167	32	179	29	407	0.850	0.0177	0.815	0.885
12	100	1	178	19	192	18	407	0.909	0.0142	0.881	0.937
14	0.01	0.1	178	26	185	18	407	0.892	0.0154	0.862	0.922

Table 6.: Results for Polynomial Kernel on Saliva-Throat Phylotype Data

Formulation	C	Alpha	Beta	Pi	truepos	falsepos	trueneg	falseneg	total	Accuracy	StdError	LCL	UCL
2	10	5	5	2	183	19	192	13	407	0.921	0.0133	0.895	0.948

Table 6.: *(continued)*

Formulation	C	Alpha	Beta	Pi	truepos	falsepos	trueneg	falseneg	total	Accuracy	StdError	LCL	UCL
3	10	5	1	6	169	40	171	27	407	0.835	0.0184	0.799	0.871
4	0.1	1	100	2	34	3	42	3	82	0.927	0.0288	0.870	0.983
6	0.01	1	0	2	164	14	197	32	407	0.887	0.0157	0.856	0.918
10	10	5	5	2	183	19	192	13	407	0.921	0.0133	0.895	0.948
11	10	5	1	6	169	40	171	27	407	0.835	0.0184	0.799	0.871
12	10	10	1	2	185	20	191	11	407	0.924	0.0131	0.898	0.950
14	0.01	1	0	2	164	14	197	32	407	0.887	0.0157	0.856	0.918

Table 7.: Results for Gaussian Kernel on Saliva-Throat Phylotype

Formulation	C	Gamma	Time	Status	Accuracy	StdError	LCL	UCL
2	100	1	256.607	Best	0.909	0.0218	0.866	0.951
3	1	10	8.680	Best	0.886	0.0241	0.839	0.933
4	100	1	83.557	Best	0.909	0.0218	0.866	0.951

Table 7.: *(continued)*

Formulation	C	Gamma	Time	Status	Accuracy	StdError	LCL	UCL
6	0.01	1	97.185	Best	0.903	0.0224	0.859	0.947
10	100	1	0.818	Optimal	0.909	0.0218	0.866	0.951
11	1	10	7.074	Optimal	0.886	0.0241	0.839	0.933
12	100	1	2.371	Optimal	0.931	0.0191	0.894	0.969
14	0.01	0.1	0.457	Optimal	0.903	0.0224	0.859	0.947

Table 8.: Results for Polynomial Kernel on Saliva-Throat Phylotype

Formulation	C	Alpha	Beta	Time	Status	Accuracy	StdError	LCL	UCL
2	0.1	1	5	146.553	Best	0.914	0.0212	0.873	0.956
3	1	5	1	225.838	Best	0.891	0.0235	0.845	0.938
4	1	5	100	92.622	Best	0.931	0.0191	0.894	0.969
6	0.1	10	0	1.923	Optimal	0.869	0.0255	0.819	0.919
10	0.01	0.1	10	0.807	Optimal	0.926	0.0198	0.887	0.965

Table 8.: *(continued)*

Formulation	C	Alpha	Beta	Time	Status	Accuracy	StdError	lcl	ucl
11	1	5	1	7.137	Optimal	0.891	0.0235	0.845	0.938
12	10	5	10	2.377	Optimal	0.920	0.0205	0.880	0.960
14	0.1	1	0	1.645	Optimal	0.869	0.0255	0.819	0.919

CHAPTER 6

RESULTS ON SIMULATED DATA

The cross-validation results for simulated data described in Chapter 4 are shown in Tables 9 and 10 for type A and B outlier infused training data test sets. Descriptions of the distribution of these two types of data sets can be found in Chapter 4 as well. The test set consists of 50,000 observations distributed randomly of each class, only without any outlier observations. The results of the classifier training indicate that the Type B outliers do influence the classic SVM model towards the null solution, therefore some form of robustness in the model. Some of the training was performed twice inadvertently, that explains the rows in Tables 9 and 10 that list 100,000 as the number of tested observations.

The accuracies listed Tables 9 and 10 are the overall accuracy of 5 runs per training data type for each variable and observation count setting. The formulation number can be found in Appendix A in Table 3, where Formulations 1 and 2 represent Brooks' ramp loss SVM, 3 through 6 represent linearized SVM with ramp loss making each a MIP, 10 being the standard dual SVM formulation, and 11 through 14 are linearized SVM.

The last two tables, Tables 11 and 12, in this appendix are the validation results for classification performance for simulation data with Type A and B outliers respectively. These tables differ from Tables 9 and 10 in that they only contain the overall accuracy of the resulting discriminant, as well as time to completion and status of the program session. The status has two results indicated by a number, these are Gurobi status codes, which are 2 for an "optimal solution" and 9 for "time limit elapsed".

Table 9.: 10-fold Cross-Validation results on Simulated Data of Type A

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
1	2	60	0.01	195636	36115	213885	54364	500000	0.819
2	2	60	0.01	195636	36115	213885	54364	500000	0.819
3	2	60	100	125470	43154	106846	24530	300000	0.774
4	2	60	0.1	124201	32510	117490	25799	300000	0.806
5	2	60	0.1	199935	197071	52929	50065	500000	0.506
6	2	60	1	113742	91762	158238	136258	500000	0.544
10	2	60	100	248818	103804	196196	51182	600000	0.742
11	2	60	10	203167	73678	176322	46833	500000	0.759
12	2	60	0.01	208187	71082	178918	41813	500000	0.774
13	2	60	0.1	206248	66493	183507	43752	500000	0.780
14	2	60	1	232420	277364	222636	267580	1000000	0.455
1	5	60	0.01	202043	39728	210272	47957	500000	0.825
2	5	60	0.01	202043	39728	210272	47957	500000	0.825
3	5	60	10	41433	10349	39651	8567	100000	0.811

Table 9.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
4	5	60	1	204452	55353	194647	45548	500000	0.798
5	5	60	10	132484	59469	190531	117516	500000	0.646
6	5	60	10	116511	103627	146373	133489	500000	0.526
10	5	60	0.01	198306	45584	204416	51694	500000	0.805
11	5	60	100	414404	126750	373250	85596	1000000	0.788
12	5	60	0.01	207389	56840	193160	42611	500000	0.801
13	5	60	10	207860	62545	187455	42140	500000	0.791
14	5	60	1	234194	163138	336862	265806	1000000	0.571
1	10	60	0.1	206930	52215	197785	43070	500000	0.809
2	10	60	0.1	206930	52215	197785	43070	500000	0.809
3	10	60	100	38923	14396	35604	11077	100000	0.745
4	10	60	0.1	202622	62642	187358	47378	500000	0.780
5	10	60	0.1	189251	54275	195725	60749	500000	0.770
6	10	60	0.1	151308	94368	155632	98692	500000	0.614
10	10	60	0.1	176705	54545	195455	73295	500000	0.744

Table 9.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
11	10	60	100	365998	125370	374630	134002	1000000	0.741
12	10	60	0.1	191816	67042	182958	58184	500000	0.750
13	10	60	1	377712	131566	368434	122288	1000000	0.746
14	10	60	0.1	135433	101949	148051	114567	500000	0.567
1	2	100	0.01	210902	43365	206635	39098	500000	0.835
2	2	100	1	414326	79824	420176	85674	1000000	0.835
3	2	100	0.01	150000	150000	100000	100000	500000	0.500
4	2	100	0.1	210248	41338	208662	39752	500000	0.838
5	2	100	0.1	167074	104479	145521	82926	500000	0.625
6	2	100	1	119017	93188	156812	130983	500000	0.552
10	2	100	0.1	211090	45091	204909	38910	500000	0.832
11	2	100	100	406182	72678	427322	93818	1000000	0.834
12	2	100	0.1	204842	37476	212524	45158	500000	0.835
13	2	100	10	205089	37870	212130	44911	500000	0.834
14	2	100	0.01	123086	56883	193117	126914	500000	0.632

Table 9.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
1	5	100	0.01	206244	43306	206694	43756	500000	0.826
2	5	100	0.1	208684	46391	203609	41316	500000	0.825
3	5	100	100	82826	20760	79240	17174	200000	0.810
4	5	100	1	163481	41398	158602	36519	400000	0.805
5	5	100	10	132753	80521	169479	117247	500000	0.604
6	5	100	1	121019	113107	136893	128981	500000	0.516
10	5	100	0.01	201306	44262	205738	48694	500000	0.814
11	5	100	100	411960	95632	404368	88040	1000000	0.816
12	5	100	0.1	206574	47423	202577	43426	500000	0.818
13	5	100	1	412080	95928	404072	87920	1000000	0.816
14	5	100	0.01	102087	98482	151518	147913	500000	0.507
1	10	100	0.1	203211	48229	201771	46789	500000	0.810
2	10	100	10	41007	9352	40648	8993	100000	0.817
3	10	100	100	196259	57089	192911	53741	500000	0.778
4	10	100	100	200798	58328	191672	49202	500000	0.785

Table 9.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
5	10	100	10	198029	93158	156842	51971	500000	0.710
6	10	100	0.1	122794	70018	179982	127206	500000	0.606
10	10	100	0.01	202595	54302	195698	47405	500000	0.797
11	10	100	100	397916	112252	387748	102084	1000000	0.786
12	10	100	0.1	196877	53975	196025	53123	500000	0.786
13	10	100	1	398112	112126	387874	101888	1000000	0.786
14	10	100	0.01	145957	109863	140137	104043	500000	0.572
1	2	200	0.1	207637	37809	212191	42363	500000	0.840
2	2	200	0.1	207011	37331	212669	42989	500000	0.839
3	2	200	10	40379	9333	40667	9621	100000	0.810
4	2	200	10	211420	41597	208403	38580	500000	0.840
5	2	200	0.1	95950	53196	196804	154050	500000	0.586
6	2	200	1	121349	146963	103037	128651	500000	0.449
10	2	200	0.01	40787	6691	43309	9213	100000	0.841
11	2	200	100	412100	78128	421872	87900	1000000	0.834

Table 9.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
12	2	200	0.1	205349	38479	211521	44651	500000	0.834
13	2	200	1	411092	77346	422654	88908	1000000	0.834
14	2	200	0.1	121860	111024	138976	128140	500000	0.522
1	5	200	0.01	204564	41775	208225	45436	500000	0.826
2	5	200	0.01	211810	52309	197691	38190	500000	0.819
3	5	200	100	199111	42120	207880	50889	500000	0.814
4	5	200	100	204287	45854	204146	45713	500000	0.817
5	5	200	0.1	129821	43834	206166	120179	500000	0.672
6	5	200	0.1	122735	95191	154809	127265	500000	0.555
10	5	200	0.01	87303	21184	78816	12697	200000	0.831
11	5	200	100	409918	95480	404520	90082	1000000	0.814
12	5	200	0.01	206616	46401	203599	43384	500000	0.820
13	5	200	100	410486	95916	404084	89514	1000000	0.815
14	5	200	1	245256	273000	227000	254744	1000000	0.472
1	10	200	0.01	208286	47722	202278	41714	500000	0.821

Table 9.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
2	10	200	0.1	203962	49176	200824	46038	500000	0.810
3	10	200	100	235955	59834	240166	64045	600000	0.794
4	10	200	0.1	207463	53398	196602	42537	500000	0.808
5	10	200	1	188018	133457	116543	61982	500000	0.609
6	10	200	0.1	123234	74679	175321	126766	500000	0.597
10	10	200	0.01	164945	43317	156683	35055	400000	0.804
11	10	200	100	398976	108078	391922	101024	1000000	0.791
12	10	200	0.01	203022	55018	194982	46978	500000	0.796
13	10	200	0.1	201177	55625	194375	48823	500000	0.791
14	10	200	0.01	172489	70551	179449	77511	500000	0.704
1	2	500	0.01	212883	46271	203729	37117	500000	0.833
2	2	500	0.1	210965	44882	205118	39035	500000	0.832
3	2	500	10	151566	111305	138695	98434	500000	0.581
4	2	500	100	202656	42763	207237	47344	500000	0.820
5	2	500	0.1	205499	193888	56112	44501	500000	0.523

Table 9.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
6	2	500	0.1	123260	140839	109161	126740	500000	0.465
10	2	500	0.1	214230	45336	204664	35770	500000	0.838
11	2	500	100	419188	80070	419930	80812	1000000	0.839
12	2	500	1	419900	80746	419254	80100	1000000	0.839
13	2	500	1	420162	81004	418996	79838	1000000	0.839
14	2	500	0.1	123638	106232	143768	126362	500000	0.535
1	5	500	0.01	204854	43027	206973	45146	500000	0.824
2	5	500	10	158709	31207	168793	41291	400000	0.819
3	5	500	100	242013	91662	308338	157987	800000	0.688
4	5	500	10	189236	43243	206757	60764	500000	0.792
5	5	500	0.1	191388	100772	149228	58612	500000	0.681
6	5	500	1	125263	95038	154962	124737	500000	0.560
10	5	500	0.1	205884	41929	208071	44116	500000	0.828
11	5	500	100	408108	78208	421792	91892	1000000	0.830
12	5	500	0.01	204154	39229	210771	45846	500000	0.830

Table 9.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
13	5	500	10	204156	39294	210706	45844	500000	0.830
14	5	500	0.1	124333	96848	153152	125667	500000	0.555
1	10	500	0.01	207238	44562	205438	42762	500000	0.825
2	10	500	10	160431	39588	160412	39569	400000	0.802
3	10	500	100	298194	76433	323567	101806	800000	0.777
4	10	500	0.01	200689	45429	204571	49311	500000	0.811
5	10	500	0.1	92603	14012	235988	157397	500000	0.657
6	10	500	0.1	124546	96763	153237	125454	500000	0.556
10	10	500	0.01	195212	38310	211690	54788	500000	0.814
11	10	500	100	402556	90730	409270	97444	1000000	0.812
12	10	500	0.01	201289	45179	204821	48711	500000	0.812
13	10	500	1	402500	91318	408682	97500	1000000	0.811
14	10	500	0.1	123963	110248	139752	126037	500000	0.527

Table 10.: 10-fold Cross-Validation results on Simulated Data of Type B

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
1	2	60	0.1	198940	33293	216707	51060	500000	0.831
2	2	60	0.1	198940	33293	216707	51060	500000	0.831
3	2	60	10	35064	5110	44890	14936	100000	0.800
4	2	60	0.1	119311	19300	130700	30689	300000	0.833
5	2	60	10	213987	196953	53047	36013	500000	0.534
6	2	60	10	120513	124335	125665	129487	500000	0.492
10	2	60	0.1	215424	221488	28512	34576	500000	0.488
11	2	60	100	384832	148246	351754	115168	1000000	0.737
12	2	60	0.1	191978	69449	180551	58022	500000	0.745
13	2	60	10	193238	72820	177180	56762	500000	0.741
14	2	60	0.1	115300	104695	145305	134700	500000	0.521
1	5	60	0.1	201317	37289	212711	48683	500000	0.828
2	5	60	0.1	201317	37289	212711	48683	500000	0.828
3	5	60	10	37738	7614	42386	12262	100000	0.801

Table 10.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
4	5	60	0.01	82988	14699	85301	17012	200000	0.841
5	5	60	100	128361	130158	119842	121639	500000	0.496
6	5	60	0.1	121642	119572	130428	128358	500000	0.504
10	5	60	0.01	184861	186263	13737	15139	400000	0.496
11	5	60	100	287704	273042	226958	212296	1000000	0.515
12	5	60	0.1	147366	113109	136891	102634	500000	0.569
13	5	60	10	143890	119787	130213	106110	500000	0.548
14	5	60	0.1	117026	103217	146783	132974	500000	0.528
1	10	60	0.1	193573	38381	211619	56427	500000	0.810
2	10	60	0.1	193573	38381	211619	56427	500000	0.810
3	10	60	0.01	150000	150000	50000	50000	400000	0.500
4	10	60	0.01	80756	13628	86372	19244	200000	0.836
5	10	60	0.1	185160	155276	94724	64840	500000	0.560
6	10	60	1	120083	130659	119341	129917	500000	0.479
10	10	60	0.01	28146	28981	21019	21854	100000	0.492

Table 10.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
11	10	60	0.1	0	0	250000	250000	500000	0.500
12	10	60	0.01	187635	120028	129972	62365	500000	0.635
13	10	60	0.1	177184	134314	115686	72816	500000	0.586
14	10	60	0.1	121538	143681	106319	128462	500000	0.456
1	2	100	0.01	197819	30707	219293	52181	500000	0.834
2	2	100	0.01	197323	30323	219677	52677	500000	0.834
3	2	100	0.01	250000	250000	0	0	500000	0.500
4	2	100	0.01	124762	22650	127350	25238	300000	0.840
5	2	100	10	159488	109187	140813	90512	500000	0.601
6	2	100	10	121519	91549	158451	128481	500000	0.560
10	2	100	0.01	250000	250000	0	0	500000	0.500
11	2	100	100	363494	135360	364640	136506	1000000	0.728
12	2	100	0.01	203840	40333	209667	46160	500000	0.827
13	2	100	0.1	193808	57194	192806	56192	500000	0.773
14	2	100	0.01	206150	44528	205472	43850	500000	0.823

Table 10.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
1	5	100	0.01	196616	31467	218533	53384	500000	0.830
2	5	100	0.1	202045	38335	211665	47955	500000	0.827
3	5	100	0.01	50000	50000	0	0	100000	0.500
4	5	100	0.1	206815	45150	204850	43185	500000	0.823
5	5	100	0.1	156187	99097	150903	93813	500000	0.614
6	5	100	0.1	120774	99500	150500	129226	500000	0.543
10	5	100	0.01	216780	196368	53632	33220	500000	0.541
11	5	100	100	294030	263956	236044	205970	1000000	0.530
12	5	100	0.01	125075	52740	197260	124925	500000	0.645
13	5	100	0.1	157803	76586	173414	92197	500000	0.662
14	5	100	0.01	206615	52161	197839	43385	500000	0.809
1	10	100	0.1	198521	42148	207852	51479	500000	0.813
2	10	100	0.1	195826	38949	211051	54174	500000	0.814
3	10	100	100	139400	113392	136608	110600	500000	0.552
4	10	100	0.01	37680	4717	45283	12320	100000	0.830

Table 10.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
5	10	100	10	115376	114183	135817	134624	500000	0.502
6	10	100	1	121985	96055	153945	128015	500000	0.552
10	10	100	0.1	137657	146128	103872	112343	500000	0.483
11	10	100	0.1	0	0	250000	250000	500000	0.500
12	10	100	0.01	126669	63207	186793	123331	500000	0.627
13	10	100	0.01	150000	150000	100000	100000	500000	0.500
14	10	100	0.01	155240	39214	210786	94760	500000	0.732
1	2	200	100	416926	77908	422092	83074	1000000	0.839
2	2	200	1	417268	77610	422390	82732	1000000	0.840
3	2	200	10	56783	25277	74723	43217	200000	0.658
4	2	200	0.1	42612	8491	41509	7388	100000	0.841
5	2	200	0.1	166495	167279	82721	83505	500000	0.498
6	2	200	0.1	123382	83889	166111	126618	500000	0.579
10	2	200	0.01	50000	50000	0	0	100000	0.500
11	2	200	100	328078	188670	311330	171922	1000000	0.639

Table 10.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
12	2	200	0.1	172621	89140	160860	77379	500000	0.667
13	2	200	1	338766	173986	326014	161234	1000000	0.665
14	2	200	0.1	122957	116430	133570	127043	500000	0.513
1	5	200	0.1	198353	32753	217247	51647	500000	0.831
2	5	200	0.1	190650	29231	220769	59350	500000	0.823
3	5	200	100	249713	88224	261776	100287	700000	0.731
4	5	200	10	201068	35678	214322	48932	500000	0.831
5	5	200	0.1	184025	121575	128425	65975	500000	0.625
6	5	200	0.1	122860	139065	110935	127140	500000	0.468
10	5	200	0.01	50000	50000	0	0	100000	0.500
11	5	200	100	297394	183420	316580	202606	1000000	0.614
12	5	200	0.01	165337	29769	220231	84663	500000	0.771
13	5	200	0.1	160111	66587	183413	89889	500000	0.687
14	5	200	0.01	122293	108235	141765	127707	500000	0.528
1	10	200	0.01	194581	30402	219598	55419	500000	0.828

Table 10.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
2	10	200	0.1	198128	38544	211456	51872	500000	0.819
3	10	200	100	141712	135862	164138	158288	600000	0.510
4	10	200	0.1	118123	23443	176557	81877	400000	0.737
5	10	200	0.1	128066	119943	130057	121934	500000	0.516
6	10	200	10	123567	103889	146111	126433	500000	0.539
10	10	200	0.01	156132	164228	35772	43868	400000	0.480
11	10	200	100	289940	273274	226726	210060	1000000	0.517
12	10	200	0.01	175801	75557	174443	74199	500000	0.700
13	10	200	0.1	144683	119227	130773	105317	500000	0.551
14	10	200	1	245876	257944	242056	254124	1000000	0.488
1	2	500	0.1	204804	35157	214843	45196	500000	0.839
2	2	500	1	406141	116337	383663	93859	1000000	0.790
3	2	500	100	26438	15545	34455	23562	100000	0.609
4	2	500	100	186544	26318	223682	63456	500000	0.820
5	2	500	0.1	166076	102368	147632	83924	500000	0.627

Table 10.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
6	2	500	0.1	124314	82262	167738	125686	500000	0.584
10	2	500	0.01	200000	200000	0	0	400000	0.500
11	2	500	100	350252	158020	341980	149748	1000000	0.692
12	2	500	0.01	205381	39409	210591	44619	500000	0.832
13	2	500	0.1	195481	61248	188752	54519	500000	0.768
14	2	500	0.1	123706	119890	130110	126294	500000	0.508
1	5	500	1	407443	71213	428787	92557	1000000	0.836
2	5	500	0.1	166440	67216	182784	83560	500000	0.698
3	5	500	100	202943	163579	186421	147057	700000	0.556
4	5	500	10	151426	65109	184891	98574	500000	0.673
5	5	500	0.1	147486	103851	146149	102514	500000	0.587
6	5	500	1	124190	124547	125453	125810	500000	0.499
10	5	500	1	324460	328769	21231	25540	700000	0.494
11	5	500	100	307566	209506	290494	192434	1000000	0.598
12	5	500	0.01	182830	81089	168911	67170	500000	0.703

Table 10.: *(continued)*

Formulation	Variables	Observations	C	Truepos	Falsepos	Trueneg	Falseneg	Total	Accuracy
13	5	500	0.1	169063	90550	159450	80937	500000	0.657
14	5	500	0.01	124275	89235	160765	125725	500000	0.570
1	10	500	0.01	203239	36081	213919	46761	500000	0.834
2	10	500	0.1	174105	83507	166493	75895	500000	0.681
3	10	500	100	247528	193108	206892	152472	800000	0.568
4	10	500	0.1	187367	78149	171851	62633	500000	0.718
5	10	500	0.1	180954	124088	125912	69046	500000	0.614
6	10	500	1	124441	67289	182711	125559	500000	0.614
10	10	500	0.01	250000	250000	0	0	500000	0.500
11	10	500	100	339648	191756	308244	160352	1000000	0.648
12	10	500	0.01	193914	71873	178127	56086	500000	0.744
13	10	500	0.1	179919	89173	160827	70081	500000	0.681
14	10	500	1	248830	211180	288820	251170	1000000	0.538

Table 11.: Test Results on Simulated Data of Type A

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
2	60	1	0.1	4.91e+01	2	0.799	0.797	0.802
2	60	2	10	1.08e+01	2	0.813	0.811	0.816
2	60	3	100	1.98e+01	2	0.814	0.812	0.816
2	60	4	0.1	1.42e+01	2	0.813	0.811	0.816
2	60	5	0.1	9.17e-02	2	0.500	0.497	0.503
2	60	6	1	3.90e-01	2	0.360	0.357	0.363
2	60	10	0.1	2.15e-03	2	0.744	0.741	0.747
2	60	11	0.1	3.18e-03	2	0.500	0.497	0.503
2	60	12	100	4.74e-03	2	0.756	0.754	0.759
2	60	13	0.01	9.99e-04	2	0.500	0.497	0.503
2	60	14	10	1.35e-03	2	0.688	0.685	0.691
2	100	1	0.01	6.05e+02	9	0.841	0.838	0.843
2	100	2	0.01	6.00e+02	9	0.841	0.838	0.843
2	100	3	0.01	6.00e+02	9	0.500	0.497	0.503
2	100	4	0.01	6.00e+02	9	0.834	0.832	0.837

Table 11.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
2	100	5	0.1	1.14e-01	2	0.801	0.798	0.803
2	100	6	10	5.84e-01	2	0.371	0.368	0.374
2	100	10	0.01	3.09e-03	2	0.825	0.823	0.827
2	100	11	100	7.59e-03	2	0.837	0.834	0.839
2	100	12	0.01	1.20e-02	2	0.836	0.833	0.838
2	100	13	0.1	2.24e-03	2	0.834	0.832	0.836
2	100	14	0.01	4.42e-03	2	0.835	0.832	0.837
2	200	1	100	6.00e+02	9	0.842	0.840	0.844
2	200	2	1	6.00e+02	9	0.842	0.840	0.844
2	200	3	10	6.00e+02	9	0.834	0.832	0.836
2	200	4	0.1	6.00e+02	9	0.842	0.840	0.844
2	200	5	0.1	1.48e-01	2	0.500	0.497	0.503
2	200	6	1	1.09e+00	2	0.587	0.584	0.590
2	200	10	0.01	7.44e-03	2	0.837	0.835	0.839
2	200	11	100	1.08e-01	2	0.840	0.838	0.843

Table 11.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
2	200	12	0.1	9.40e-02	2	0.839	0.837	0.841
2	200	13	1	7.33e-03	2	0.839	0.837	0.841
2	200	14	0.1	4.98e-03	2	0.630	0.627	0.633
2	500	1	0.1	6.00e+02	9	0.835	0.833	0.837
2	500	2	1	6.00e+02	9	0.835	0.833	0.837
2	500	3	100	6.00e+02	9	0.835	0.832	0.837
2	500	4	100	6.00e+02	9	0.835	0.833	0.837
2	500	5	0.1	7.52e-01	2	0.692	0.689	0.695
2	500	6	0.01	2.63e+02	2	0.500	0.497	0.503
2	500	10	0.01	2.95e-02	2	0.837	0.834	0.839
2	500	11	100	5.11e-01	2	0.836	0.834	0.839
2	500	12	0.01	7.08e-01	2	0.837	0.834	0.839
2	500	13	0.1	2.82e-02	2	0.836	0.834	0.839
2	500	14	0.1	1.36e-01	2	0.586	0.583	0.589
5	60	1	0.1	3.65e+01	2	0.831	0.829	0.833

Table 11.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
5	60	2	0.1	7.63e+01	2	0.831	0.829	0.833
5	60	3	10	6.00e+02	9	0.788	0.785	0.791
5	60	4	0.01	3.45e+02	2	0.811	0.809	0.813
5	60	5	10	1.14e-01	2	0.658	0.655	0.661
5	60	6	0.1	3.22e+00	2	0.828	0.825	0.830
5	60	10	0.01	2.24e-03	2	0.828	0.826	0.830
5	60	11	100	4.45e-03	2	0.828	0.826	0.830
5	60	12	0.1	5.00e-03	2	0.828	0.826	0.830
5	60	13	10	1.21e-03	2	0.828	0.826	0.830
5	60	14	0.1	1.91e-03	2	0.305	0.302	0.308
5	100	1	0.01	6.00e+02	9	0.829	0.826	0.831
5	100	2	0.1	6.00e+02	9	0.823	0.821	0.825
5	100	3	0.01	6.00e+02	9	0.500	0.497	0.503
5	100	4	0.1	6.00e+02	9	0.794	0.791	0.796
5	100	5	0.1	1.13e-01	2	0.674	0.671	0.677

Table 11.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
5	100	6	0.1	9.76e-01	2	0.416	0.413	0.419
5	100	10	0.01	3.10e-03	2	0.820	0.817	0.822
5	100	11	100	9.63e-03	2	0.814	0.812	0.817
5	100	12	0.01	1.64e-02	2	0.824	0.822	0.827
5	100	13	0.1	3.47e-03	2	0.803	0.800	0.805
5	100	14	0.01	5.05e-03	2	0.500	0.497	0.503
5	200	1	0.1	6.00e+02	9	0.812	0.809	0.814
5	200	2	0.1	6.00e+02	9	0.812	0.809	0.814
5	200	3	100	6.00e+02	9	0.804	0.802	0.807
5	200	4	10	6.00e+02	9	0.802	0.800	0.805
5	200	5	0.1	2.01e-01	2	0.504	0.501	0.507
5	200	6	1	1.85e+00	2	0.316	0.313	0.319
5	200	10	0.01	7.47e-03	2	0.785	0.782	0.788
5	200	11	100	1.19e-01	2	0.761	0.759	0.764
5	200	12	0.01	1.30e-01	2	0.781	0.778	0.783

Table 11.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
5	200	13	0.1	7.28e-03	2	0.769	0.767	0.772
5	200	14	0.01	1.72e-02	2	0.637	0.634	0.640
5	500	1	1	6.00e+02	9	0.814	0.811	0.816
5	500	2	0.1	6.00e+02	9	0.814	0.812	0.816
5	500	3	100	6.00e+02	9	0.814	0.812	0.816
5	500	4	10	6.00e+02	9	0.814	0.811	0.816
5	500	5	0.1	7.51e-01	2	0.696	0.693	0.699
5	500	6	1	4.63e+00	2	0.632	0.629	0.635
5	500	10	1	3.34e-02	2	0.824	0.822	0.827
5	500	11	100	5.33e-01	2	0.829	0.827	0.831
5	500	12	0.01	7.06e-01	2	0.829	0.826	0.831
5	500	13	0.1	2.74e-02	2	0.829	0.827	0.831
5	500	14	1	1.49e-01	2	0.536	0.533	0.540
10	60	1	0.1	2.39e+00	2	0.817	0.815	0.819
10	60	2	0.1	1.03e+01	2	0.817	0.815	0.819

Table 11.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
10	60	3	0.01	6.00e+02	9	0.500	0.497	0.503
10	60	4	0.01	2.83e+02	2	0.822	0.820	0.825
10	60	5	0.1	2.09e-01	2	0.805	0.803	0.807
10	60	6	10	1.00e+00	2	0.632	0.629	0.635
10	60	10	0.01	2.67e-03	2	0.783	0.780	0.785
10	60	11	0.1	5.55e-03	2	0.500	0.497	0.503
10	60	12	0.01	8.14e-03	2	0.747	0.744	0.749
10	60	13	0.1	1.83e-03	2	0.733	0.730	0.735
10	60	14	0.1	2.21e-03	2	0.404	0.401	0.407
10	100	1	0.1	6.00e+02	9	0.818	0.816	0.821
10	100	2	0.1	6.00e+02	9	0.818	0.816	0.821
10	100	3	100	6.00e+02	9	0.799	0.796	0.801
10	100	4	0.01	6.00e+02	9	0.819	0.816	0.821
10	100	5	1	1.94e-01	2	0.709	0.707	0.712
10	100	6	0.01	3.33e+00	2	0.706	0.703	0.709

Table 11.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
10	100	10	0.1	1.18e-02	2	0.773	0.770	0.776
10	100	11	0.1	1.26e-02	2	0.500	0.497	0.503
10	100	12	0.01	2.52e-02	2	0.666	0.663	0.669
10	100	13	0.01	1.98e-03	2	0.500	0.497	0.503
10	100	14	0.01	4.04e-03	2	0.512	0.508	0.515
10	200	1	0.01	6.00e+02	9	0.829	0.827	0.831
10	200	2	0.1	6.00e+02	9	0.826	0.824	0.828
10	200	3	100	6.00e+02	9	0.827	0.824	0.829
10	200	4	0.1	6.00e+02	9	0.825	0.823	0.828
10	200	5	0.1	3.68e-01	2	0.734	0.732	0.737
10	200	6	100	3.12e+00	2	0.376	0.373	0.379
10	200	10	0.01	8.82e-03	2	0.832	0.830	0.834
10	200	11	100	1.10e-01	2	0.829	0.827	0.832
10	200	12	0.01	1.24e-01	2	0.830	0.828	0.833
10	200	13	0.1	8.40e-03	2	0.828	0.826	0.830

Table 11.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
10	200	14	1	9.63e-03	2	0.376	0.373	0.379
10	500	1	0.01	6.00e+02	9	0.804	0.801	0.806
10	500	2	0.1	6.00e+02	9	0.773	0.770	0.775
10	500	3	100	6.00e+02	9	0.798	0.795	0.800
10	500	4	0.1	6.00e+02	9	0.790	0.788	0.793
10	500	5	0.1	1.57e+00	2	0.723	0.720	0.725
10	500	6	10	7.85e+00	2	0.681	0.679	0.684
10	500	10	0.01	3.89e-02	2	0.809	0.807	0.812
10	500	11	100	5.42e-01	2	0.809	0.807	0.812
10	500	12	0.01	8.71e-01	2	0.811	0.808	0.813
10	500	13	0.1	3.78e-02	2	0.804	0.802	0.807
10	500	14	1	1.38e-01	2	0.681	0.679	0.684

Table 12.: Test Results on Simulated Data of Type B

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
2	60	1	0.1	2.22e+02	2	0.821	0.819	0.824
2	60	2	10	4.21e+01	2	0.812	0.810	0.815
2	60	3	100	1.03e+02	2	0.812	0.810	0.815
2	60	4	0.1	1.72e+02	2	0.822	0.820	0.824
2	60	5	1	1.06e-01	2	0.724	0.721	0.727
2	60	6	0.1	4.10e-01	2	0.311	0.309	0.314
2	60	10	0.1	9.44e-03	2	0.500	0.497	0.503
2	60	11	100	4.03e-03	2	0.806	0.804	0.809
2	60	12	0.1	5.37e-03	2	0.842	0.839	0.844
2	60	13	10	1.15e-03	2	0.806	0.804	0.809
2	60	14	0.1	1.42e-03	2	0.311	0.308	0.314
2	100	1	0.01	6.00e+02	9	0.837	0.835	0.840
2	100	2	0.1	6.00e+02	9	0.841	0.838	0.843
2	100	3	0.01	6.00e+02	9	0.500	0.497	0.503
2	100	4	0.1	6.00e+02	9	0.841	0.839	0.843

Table 12.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
2	100	5	0.1	1.02e-01	2	0.761	0.758	0.763
2	100	6	0.1	5.83e-01	2	0.616	0.613	0.619
2	100	10	0.01	2.65e-03	2	0.500	0.497	0.503
2	100	11	100	9.46e-03	2	0.584	0.581	0.587
2	100	12	0.01	1.36e-02	2	0.831	0.828	0.833
2	100	13	0.1	2.34e-03	2	0.756	0.753	0.759
2	100	14	0.01	4.29e-03	2	0.807	0.805	0.810
2	200	1	100	6.00e+02	9	0.841	0.839	0.843
2	200	2	1	6.00e+02	9	0.841	0.839	0.843
2	200	3	10	6.00e+02	9	0.833	0.830	0.835
2	200	4	0.1	6.00e+02	9	0.837	0.835	0.839
2	200	5	0.1	1.60e-01	2	0.462	0.459	0.465
2	200	6	1	1.09e+00	2	0.537	0.533	0.540
2	200	10	0.01	6.81e-03	2	0.500	0.497	0.503
2	200	11	100	1.12e-01	2	0.449	0.446	0.452

Table 12.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
2	200	12	0.1	9.46e-02	2	0.449	0.446	0.452
2	200	13	1	9.41e-03	2	0.449	0.446	0.452
2	200	14	0.1	8.00e-03	2	0.564	0.561	0.567
2	500	1	0.1	6.00e+02	9	0.838	0.835	0.840
2	500	2	1	6.00e+02	9	0.839	0.837	0.842
2	500	3	100	6.00e+02	9	0.837	0.835	0.840
2	500	4	100	6.00e+02	9	0.840	0.838	0.842
2	500	5	0.1	6.78e-01	2	0.760	0.757	0.763
2	500	6	0.01	2.41e+01	2	0.500	0.497	0.503
2	500	10	0.01	2.84e-02	2	0.500	0.497	0.503
2	500	11	100	5.28e-01	2	0.689	0.686	0.692
2	500	12	0.01	6.11e-01	2	0.842	0.840	0.844
2	500	13	0.1	3.26e-02	2	0.760	0.757	0.762
2	500	14	0.1	1.45e-01	2	0.568	0.565	0.571
5	60	1	0.1	3.36e+01	2	0.838	0.836	0.841

Table 12.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
5	60	2	0.1	7.69e+01	2	0.838	0.836	0.841
5	60	3	10	6.00e+02	9	0.826	0.824	0.828
5	60	4	0.01	6.00e+02	9	0.840	0.837	0.842
5	60	5	10	1.07e-01	2	0.776	0.774	0.779
5	60	6	0.1	6.12e-01	2	0.482	0.479	0.485
5	60	10	0.01	2.55e-03	2	0.500	0.497	0.503
5	60	11	100	5.81e-03	2	0.612	0.609	0.615
5	60	12	0.1	6.48e-03	2	0.807	0.805	0.810
5	60	13	10	1.70e-03	2	0.771	0.769	0.774
5	60	14	0.1	2.25e-03	2	0.677	0.674	0.680
5	100	1	0.01	6.00e+02	9	0.834	0.832	0.836
5	100	2	0.1	6.00e+02	9	0.837	0.835	0.839
5	100	3	0.01	6.00e+02	9	0.500	0.497	0.503
5	100	4	0.1	6.00e+02	9	0.837	0.834	0.839
5	100	5	0.1	1.12e-01	2	0.663	0.660	0.666

Table 12.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
5	100	6	0.1	9.80e-01	2	0.570	0.567	0.573
5	100	10	0.01	2.65e-03	2	0.592	0.589	0.595
5	100	11	100	1.35e-02	2	0.450	0.446	0.453
5	100	12	0.01	1.58e-02	2	0.500	0.497	0.503
5	100	13	0.1	3.18e-03	2	0.663	0.660	0.666
5	100	14	0.01	4.09e-03	2	0.793	0.790	0.795
5	200	1	0.1	6.00e+02	9	0.831	0.828	0.833
5	200	2	0.1	6.00e+02	9	0.831	0.828	0.833
5	200	3	100	6.00e+02	9	0.825	0.823	0.828
5	200	4	10	6.00e+02	9	0.828	0.826	0.830
5	200	5	0.1	1.68e-01	2	0.712	0.710	0.715
5	200	6	1	1.85e+00	2	0.530	0.527	0.533
5	200	10	0.01	7.45e-03	2	0.500	0.497	0.503
5	200	11	100	1.14e-01	2	0.598	0.595	0.601
5	200	12	0.01	1.21e-01	2	0.831	0.829	0.833

Table 12.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
5	200	13	0.1	8.60e-03	2	0.722	0.719	0.724
5	200	14	0.01	1.64e-02	2	0.722	0.719	0.724
5	500	1	1	6.00e+02	9	0.840	0.838	0.843
5	500	2	0.1	6.00e+02	9	0.840	0.838	0.842
5	500	3	100	6.00e+02	9	0.835	0.833	0.837
5	500	4	10	6.00e+02	9	0.840	0.838	0.842
5	500	5	0.1	1.30e+00	2	0.767	0.764	0.770
5	500	6	1	4.63e+00	2	0.295	0.293	0.298
5	500	10	1	3.01e-02	2	0.500	0.497	0.503
5	500	11	100	5.36e-01	2	0.774	0.771	0.776
5	500	12	0.01	7.06e-01	2	0.826	0.823	0.828
5	500	13	0.1	3.56e-02	2	0.794	0.792	0.797
5	500	14	0.01	1.48e-01	2	0.535	0.532	0.538
10	60	1	0.1	3.53e+01	2	0.802	0.800	0.804
10	60	2	0.1	8.02e+01	2	0.802	0.800	0.804

Table 12.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
10	60	3	0.01	6.00e+02	9	0.500	0.497	0.503
10	60	4	0.01	6.00e+02	9	0.836	0.834	0.838
10	60	5	0.1	1.10e-01	2	0.596	0.593	0.599
10	60	6	10	9.93e-01	2	0.493	0.490	0.496
10	60	10	0.01	2.42e-03	2	0.605	0.602	0.608
10	60	11	0.1	5.89e-03	2	0.500	0.497	0.503
10	60	12	0.01	8.28e-03	2	0.500	0.497	0.503
10	60	13	0.1	1.42e-03	2	0.641	0.638	0.644
10	60	14	0.1	2.57e-03	2	0.521	0.518	0.525
10	100	1	0.1	6.00e+02	9	0.827	0.825	0.829
10	100	2	0.1	6.00e+02	9	0.827	0.825	0.829
10	100	3	100	6.00e+02	9	0.789	0.786	0.792
10	100	4	0.01	6.00e+02	9	0.839	0.837	0.841
10	100	5	1	3.13e-01	2	0.645	0.642	0.648
10	100	6	0.01	1.62e+00	2	0.560	0.556	0.563

Table 12.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
10	100	10	0.1	1.15e-02	2	0.573	0.570	0.576
10	100	11	0.1	1.16e-02	2	0.500	0.497	0.503
10	100	12	0.01	2.81e-02	2	0.841	0.838	0.843
10	100	13	0.01	1.90e-03	2	0.500	0.497	0.503
10	100	14	0.01	4.81e-03	2	0.823	0.820	0.825
10	200	1	0.01	6.00e+02	9	0.825	0.823	0.828
10	200	2	0.1	6.00e+02	9	0.804	0.802	0.807
10	200	3	0.01	6.00e+02	9	0.500	0.497	0.503
10	200	4	0.1	6.00e+02	9	0.808	0.806	0.811
10	200	5	0.1	2.66e-01	2	0.450	0.447	0.453
10	200	6	100	3.12e+00	2	0.559	0.556	0.562
10	200	10	0.01	1.01e-02	2	0.484	0.481	0.487
10	200	11	100	1.10e-01	2	0.467	0.464	0.470
10	200	12	0.01	1.23e-01	2	0.677	0.674	0.680
10	200	13	0.1	1.06e-02	2	0.446	0.443	0.449

Table 12.: *(continued)*

Attributes	Observations	Formulation	C	Time	Status	Accuracy	LCL	UCL
10	200	14	1	1.04e-02	2	0.528	0.525	0.531
10	500	1	0.01	6.00e+02	9	0.830	0.828	0.833
10	500	2	0.1	6.00e+02	9	0.824	0.822	0.827
10	500	3	100	6.00e+02	9	0.829	0.826	0.831
10	500	4	0.1	6.00e+02	9	0.822	0.820	0.825
10	500	5	0.1	1.43e+00	2	0.566	0.563	0.569
10	500	6	10	7.85e+00	2	0.396	0.393	0.399
10	500	10	0.01	3.49e-02	2	0.500	0.497	0.503
10	500	11	100	5.56e-01	2	0.500	0.497	0.503
10	500	12	0.01	8.57e-01	2	0.580	0.577	0.583
10	500	13	0.1	4.24e-02	2	0.594	0.591	0.597
10	500	14	1	1.35e-01	2	0.517	0.514	0.520

REFERENCES

[1] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN: 0321321367.

[2] J. Paul Brooks. "Support Vector Machines with the Ramp Loss and the Hard Margin Loss". In: *Oper. Res.* 59 (2 2011), pp. 467–479. ISSN: 0030-364X. DOI: http://dx.doi.org/10.1287/opre.1100.0854. URL: http://dx.doi.org/10.1287/opre.1100.0854.

[3] Xiaotong Shen et al. "On ψ-Learning". In: *Journal of the American Statistical Association* 98 (2003), pp. 724–734. URL: http://ideas.repec.org/a/bes/jnlasa/v98y2003p724-734.html.

[4] Zhuang Wang and Slobodan Vucetic. "Fast Online Training of Ramp Loss Support Vector Machines". In: *Data Mining, IEEE International Conference on* 0 (2009), pp. 569–577. ISSN: 1550-4786. DOI: http://doi.ieeecomputersociety.org/10.1109/ICDM.2009.53.

[5] Kristin P. Bennett and O. L. Mangasarian. "Robust linear programming discrimination of two linearly inseparable sets". In: *Optimization Methods and Software* 1.1 (1992), pp. 23–34. DOI: 10.1080/10556789208805504. eprint: http://www.tandfonline.com/doi/pdf/10.1080/10556789208805504. URL: http://www.tandfonline.com/doi/abs/10.1080/10556789208805504.

[6] O. L. Mangasarian and David R. Musicant. "Lagrangian support vector machines". In: *J. Mach. Learn. Res.* 1 (2001), pp. 161–177. ISSN: 1532-4435. DOI: http://

dx.doi.org/10.1162/15324430152748218. URL: http://dx.doi.org/10.1162/15324430152748218.

[7] I Hadzic and V Kecman. "Support vector machines trained by linear programming: theory and application in image compression and data classification". In: *Proceedings of the 5th Seminar on Neural Network Applications in Electrical Engineering*. 2000, pp. 18–23.

[8] Vojislav Kecman and Tiru Arthanari. "Comparisons of QP and LP Based Learning from Empirical Data". In: *Proceedings of the 14th International conference on Industrial and engineering applications of artificial intelligence and expert systems: engineering of intelligent systems*. IEA/AIE '01. London, UK, UK: Springer-Verlag, 2001, pp. 326–332. ISBN: 3-540-42219-6. URL: http://dl.acm.org/citation.cfm?id=646863.707964.

[9] Licheng Jiao Weida Zhou Li Zhang. "Linear Programming Support Vector Machines". In: *Pattern Recognition* 35 (12 2002), pp. 2927–2936.

[10] O. L. Mangasarian. "Generalized Support Vector Machines". In: *Advances in Large Margin Classifiers*. MIT Press, 1998, pp. 135–146.

[11] The Human Microbiome Project Consortium. "A Framework for Human Microbiome Research". In: *Nature* 486 (2012), pp. 215–221.

[12] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. COLT '92. Pittsburgh, Pennsylvania, United States: ACM, 1992, pp. 144–152. ISBN: 0-89791-497-X. DOI: http://doi.acm.org/10.1145/130385.130401. URL: http://doi.acm.org/10.1145/130385.130401.

[13] Christopher J. C. Burges. “A Tutorial on Support Vector Machines for Pattern Recognition”. In: *Data Min. Knowl. Discov.* 2 (2 1998), pp. 121–167. ISSN: 1384-5810. DOI: 10.1023/A:1009715923555. URL: http://dl.acm.org/citation.cfm?id=593419.593463.

[14] Corinna Cortes and Vladimir Vapnik. “Support-Vector Networks”. In: *Mach. Learn.* 20 (3 1995), pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: http://dl.acm.org/citation.cfm?id=218919.218929.

[15] O.L. Mangasarian. “Linear and nonlinear separation of patterns by linear programming”. In: *Operations Research* 13 (1965), pp. 444–452.

[16] Peter J Turnbaugh et al. “The human microbiome project: exploring the microbial part of ourselves in a changing world”. In: *Nature* 449.7164 (2007), p. 804.

[17] The Human Microbiome Consortium. “Structure, function and diversity of the healthy human microbiome”. In: *Nature* 486.7164 (2012), pp. 207–214.

[18] Bo Jiang et al. *Estimating the confidence interval for prediction errors of support vector machine classifiers*. 2008.