MATHMATICAL PROGRAMMING IN DATA MINING: MODELS FOR BINARY CLASSIFICATION WITH APPLICATION TO COLLUSION DETECTION IN ONLINE GAMBLING

by

Maryanne Domm

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF SYSTEMS AND INDUSTRIAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2003

THE UNIVERSITY OF ARIZONA ®
GRADUATE COLLEGE

As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Maryanne Lorraine Domm entitled Mathematical Programming in Data Mining: Models for Binary Classification with Application to Collusion Detection in Online Gambling

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

Jeffrey B. Goldberg	3/27/03 Date
Ferenc Szidarovszky	3/27/03 Date
Young Jun Son	3/27/03 Date
	Date
	Date

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copy of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Dissertation Director Jeffrey B. Goldberg — 4/16/03 Date

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the copyright holder.

SIGNED: Maryanne Da

ACKNOWLEDGMENTS

I would like to thank:

My parents and friends for their support.

Andrew for all his help with the research and writing.

Robert Maier for pointing me in the direction of SIE.

Bill Ryder and his wife for their incredible kindness during the summer of 2002.

TABLE OF CONTENTS

TABLE OF CONTENTS—*Continued*

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES—*Continued*

ABSTRACT

Data mining is a semi-automated technique to discover patterns and trends in large amounts of data and can be used to build statistical models to predict those patterns and trends. One type of prediction model is a classifier, which attempts to predict to which group a particular item belongs. An important binary classifier, the Support Vector Machine classifier, uses non-linear optimization to find a hyperplane separating the two classes of data. This classifier has been reformulated as a linear program and as a pure quadratic program.

We propose two modeling extensions to the Support Vector Machine classifier. The first, the Linearized Proximal Support Vector Machine classifier, linearizes the objective function of the pure quadratic version. This reduces the importance the classifier places on outlying data points. The second extension improves the conceptual accuracy of the model. The Integer Support Vector Machine classifier uses binary indicator variables to indicate potential misclassification errors and minimizes these errors directly. Performance of both these new classifiers was evaluated on a simple two dimensional data set as well as on several data sets commonly used in the literature and was compared to the original classifiers.

These classifiers were then used to build a model to detect collusion in online gambling. Collusion occurs when two or more players play differently against each other than against the rest of the players. Since their communication cannot be intercepted, collusion is easier for online gamblers. However, collusion can still be identified by examining the playing style of the colluding players. By analyzing the record of play from online poker, a model to predict whether a hand contains colluding players or not can be built.

We found that these new classifiers performed about as well as previous classifiers and sometimes worse and sometimes better. We also found that one form of online collusion could be detected, but not perfectly.

CHAPTER 1

INTRODUCTION

1.1 What is Data Mining?

Data mining is a technique to discover patterns and trends in data and to create a statistical model of that data. It is particularly useful for large, high dimensional data sets that are too unwieldy for traditional statistical analysis. Unsupervised learning refers to the task of describing patterns in the data while supervised learning refers to the task of building predictive models. If the quantity to be predicted in supervised learning is a numerical value, then regression analysis is performed. If, however, the quantity to be predicted can only take on a finite number of values, then classification is performed. Binary classification is undertaken when there are only two possible values for the output quantity.

An important binary classifier is the Support Vector Machine classifier (Boser, Guyon, and Vapnik 1992) (Vapnik 1996), which classifies data based on its relative location to a separating hyperplane. In order to find this separating hyperplane, a quadratic program with inequality constraints is solved to obtain the equation of the separating hyperplane. However, due to the nonlinearity of the objective function, obtaining the solution can be difficult. Kecman and Arthanari (2001) attempted to remedy this problem by replacing the quadratic term in the objective function of the Support Vector Machine classifier (SVM) by a linear term. Fung and Mangasarian (2001) also reformulated SVM to decrease solution time and created the Proximal Support Vector Machine classifier (PSVM). Their classifier has a purely quadratic objective function. In addition, they replaced the inequality constraints with equality constraints. The Karush Kuhn Tucker conditions can then be used to obtain the solution from a system of linear equations.

The Proximal Support Vector Machine classifier, while quick to solve, magnifies the importance of points far from the hyperplane. In order to remedy this, we propose the

Linearized Proximal Support Vector Machine classifier. This is a reformulation of PSVM analogous to Kecman and Arthanari's reformulation of SVM.

All Support Vector Machine classifiers try to minimize the number of potential errors the classifier will make by minimizing a sum of distances from a hyperplane. However, the actual task of classification does not place any importance on the distance from the classifying hyperplane and is only concerned with the relative location of the point to be classified. In order to model this more closely, we propose the Integer Support Vector Machine Classifier (ISVM). ISVM uses binary error indicator variables to minimize the number of potential errors the classifier can make.

The structure of this document is as follows: Chapter 2 gives the formulation for Linearized PSVM, discusses results for a simple two dimensional data set, and gives results for eleven data sets from the UCI Machine Learning Repository (Blake and Merz 1998). Chapter 3 presents similar information for a reformulation of SVM as an integer program. Chapter 4 presents an application of linear classifiers to collusion detection in online gambling. The final chapter, Chapter 5, summarizes the previous chapters and discusses future research directions.

Data mining is highly application driven, meaning the solution technique is driven by not only the data, but the question that the owners of the data want answered. Data sets come from diverse fields, including biology and health care, computer science (image processing), business and marketing, and physics. Applications from biology and health care include predicting the recurrence of cancer in patients (Mangasarian, Street, and Wolberg 1995), predicting which new chemical compounds are likely to result in new drugs (Srinivasan and King 1999), and predicting which patients are likely to develop diabetes (Smith, Everhart, Dickson, Knowler, and Johannes 1988). Applications from computer science include image classification (Schweitzer 1997), spam filtering (Androutsopoulos, Koutsias, Chandrinos, and Spyropoulos 2000), and detecting intrusion into computer systems (Han, Lu, Lu, Bo, and Yong 2002). Business and marketing uses data mining for targeting advertisements to customers to maximize their effectiveness (Ha, Bae, and Park 2002). Data

mining is also used to discover which products people are likely to buy together to improve product placement in stores (Hamuro, Katoh, Matsuda, and Yada 1998). Finally, physics uses data mining to process the massive amounts of data collected in particle collision experiments (Conway and Loomis 1995) and to examine the large amounts of data produced by N-body astrophysics simulations (Pfitzner, Salmon, and Sterling 1997). With the diversity of fields, the terminology used by practitioners can vary widely. The set of measured variables is known as the **inputs, features, predictors** or **independent variables**. Those variables that are influenced by the measured variables are known as the **outputs, responses** or **dependent variables**. The output variables may not always be included in the data set.

Input variables can be one of three types: **quantitative, qualitative** (or **categorical**), and **ordered categorical**. The values of quantitative variables can be compared and a metric is defined over the set of possible values. A qualitative or categorical variable can only take on values from a finite set and these values cannot typically be compared, i.e. (*red, green, blue*). The values of an ordered categorical variable can also only take on a finite number of values, but they can be compared. Unlike quantitative variables however, no metric is defined over the set of possible values. An example of an ordered categorical variable is one that can take on the values (*small, medium, large*).

Following the notation in Hastie et al (2001), X is an input variable or feature. If X is a vector, X_j is the j^{th} component of X. Quantitative outputs are denoted as Y and qualitative output are denoted as G. x_i is the i^{th} observation of input X and x_{ij} is the i^{th} observation of the j^{th} component of input X. A series of N observations of the input vector X is denoted $\mathbf{X} = (x_1, x_2, \ldots, x_N)$.

The data mining process consists of five steps: acquiring the data, cleaning the data, training the model, model validation, and model use. The first step, data acquisition, can be the most difficult step from a practical viewpoint. Unless the data set already exists, the collection system must be designed and tested. In addition, the storage system must be designed and implemented. Often data mining is not undertaken unless the data already

exists and is easily accessible.

The second step prepares the data for mining. There are several tasks that must take place here. First of all, the data set may be missing values and a procedure for processing those values must be determined. Options include excluding those data points with missing values and excluding that particular feature from the feature set to be mined. In the case of numerical data, a particular value (for example 0) may be assigned if appropriate. In the case of categorical data, the missing values may be treated as a separate category. Additionally, any aggregate features should be calculated. **Aggregate features** are those that are computed from other features such as averages, variances and even time information such as day of week, day of year, or month.

Training the model involves choosing an appropriate algorithm and applying it to the training data set to build a model. Algorithm choice is driven by the specifics of the problem to be solved as well as the data itself. To validate the model after it is built, its performance is evaluated on the test data set. The test set should not contain any data previously used in training the model. Typically, one large data set is split into two pieces, not necessarily of equal size. One piece is used for training the model and the remaining piece is used for testing. Once a model is created with satisfactory performance, the model is used.

One issue in creating the training and test data sets is determining how much data to use in each set. Physical limitations of computing environments, i.e. disk space, RAM and CPU speed, may severely limit the amount of data used. One strategy has been proposed by Yakowitz and Mai (1995) in the machine learning literature. Their strategy was originally devised for the task in sequential design of determining the minimum of an unknown meandering function based on a set of measurements with noise. In this sequential design problem, an experimenter obtains a number of these noisy measurements, which can be taken at any point without penalty for poor choices. The experiment will be terminated at some period in time and the experimenter neither controls this point nor has knowledge of when it is. At the end of the experiment, the experimenter must specify the point they consider to be the best on the basis of experimentation. Yakowitz and Mai propose an off-

line learning method that seeks the best trade-off between acquiring new test points and retesting previously selected points to maximize performance. This method could be used to determine experimentally when to add new points to a test or training set.

1.2 Methods Used in Data Mining

Data mining methods can be split into two groups according to the task they perform. Unsupervised learning methods are given a set of inputs and try to learn properties about the joint probability distribution function of the features. Supervised learning methods are given a set of inputs and attempt to predict the output variables.

1.2.1 Unsupervised Learning Methods

The data for an unsupervised learning task consists of N observations $(x_1, x_2, \ldots, x_N)$ of a p-dimensional random variable, $\mathbf{X}$, and try to learn properties of the joint probability distribution function $f(\mathbf{X})$ in the absence of any error information (i.e. the true output variable is not available). For low dimensional data sets (3 or fewer dimensions), techniques from statistics can be used to estimate $f(\mathbf{X})$. However, these techniques are not applicable for higher dimensional data sets ($p > 3$) because as the number of dimensions increases, the density of sample points decreases and the estimates become less reliable. In this case, most of the methods use descriptive statistics, i.e. they try to determine characteristics of those $\mathbf{X}$'s where $f(\mathbf{X})$ is large. This can be done by creating new features that are functions of old features as in Principal Components, Self-Organizing Maps and Multidimensional Scaling. Or, as in Cluster Analysis, we can try to directly find regions where $f(\mathbf{X})$ is large.

Principal Components, Principal Curves, Self-Organizing Maps, and **Multidimensional Scaling** attempt to reduce the dimensionality of the data set by determining whether the variables are functions of a smaller set of "latent" variables. Principal Components and Principal Curves (Hastie, Tibshirani, and Friedman 2001b) find series of approximations to the data. Principal components finds a series of linear approximations to the data by least

squares approximation. These linear approximations are orthogonal projections of the data onto a line or plane. The first linear approximation in the series has the highest variance and subsequent approximations decrease in variance. Principal Curves is the generalization of Principal Components to non-linear projections. Self-Organizing Maps (Hastie, Tibshirani, and Friedman 2001c) overlay a grid of integer points (called "prototypes") over the data set and bends this grid space until the prototypes approximate the data as well as possible. This procedure does not preserve relative distances between points, which in some applications is desirable. Multidimensional Scaling (Hastie, Tibshirani, and Friedman 2001d) attempts to remedy this by trying to preserve the distances between the data while finding a lower dimensional approximation of the data.

Cluster Analysis (Hastie, Tibshirani, and Friedman 2001e) assumes that the data can be represented as a mixture of simpler densities or clusters. Each individual density is assumed to represent a class in the data. In **K-means clustering** (Hastie, Tibshirani, and Friedman 2001e), K cluster centers are chosen, one for each class. Each data point is assigned to the class corresponding to the closest cluster center. The positions of the cluster centers are moved around in an iterative fashion until the points within a cluster are closer to each other than they are to points outside the cluster. Finally for binary-valued data, **Associative Rules** (Hastie, Tibshirani, and Friedman 2001f) attempt to discover joint conjunctive rules, i.e. sets of variable values, that describe regions of high density.

Model validation for unsupervised learning tasks can be somewhat problematic as no error information is contained within the data set. The model validation process must then rely on heuristics to evaluate model quality.

1.2.2 Supervised Learning

Supervised learning is somewhat easier than unsupervised learning since an outcome for each observation $\mathbf{X} = (x_1, x_2, \ldots, x_N)$, where x_i is a random p-vector, is included in the data set. In the case of a numerical outcome, Y, regression, which seeks to predict a

numerical value, is used. For a categorical outcome, G, classification is attempted.

Regression tries to make a good prediction, $\hat{\mathrm{Y}}$, of the quantitative output given a value of the input X. Least squares regression is a well-known example of this. **Linear least squares** (Hastie, Tibshirani, and Friedman 2001g) models the output, Y, as a linear combination of the components of the input p-vector $X = (X_1, X_2, \ldots X_p)$. The task is to find the coefficients $\hat{\beta}$ of X. This is done by minimizing the residual sum of squares

$$\mathrm{RSS}(\beta) = (\mathbf{y} - \beta\mathbf{X})^T (\mathbf{y} - \beta\mathbf{X}).$$

Other methods that fall into the regression category include basis expansions (splines, wavelets, etc...), kernel methods, and neural networks. **Basis expansions** (Hastie, Tibshirani, and Friedman 2001h) uses a set of m functions, $h_m(X) : \Re^p \mapsto \Re$, to transform the input variables into new variables. The model of the outputs is then a linear combination of the transformed variables. Any complete set of basis functions can be used as the transformations including splines and wavelet transformations. **Kernel methods** (Hastie, Tibshirani, and Friedman 2001i) fit a model, which can be linear or polynomial for example, to each point, x_o, using its neighbors, i.e. points "close" to it. The kernel $K_\lambda(x_o, x_i)$ is a function that determines the neighborhood of x_o by setting weights on the rest of the data points x_i. The parameter λ sets the width of the kernel.

Neural networks (Hastie, Tibshirani, and Friedman 2001j) were developed separately in the statistics and artificial intelligence fields. A neural network consists of two stages. First, new variables that are linear combinations of the inputs are created. The output is modeled as a non-linear function of the inputs. A simple neural network is shown in Figure 1.1. The variables in the hidden layer, Z, are modeled as a function of linear combinations of the inputs X

$$Z_m = \sigma\left(\alpha_{0m} + \alpha_m^T\right), \; m = 1, \ldots, M,$$

where $\sigma(v)$ is the activation function. Traditionally, the sigmoid function is used. The

output Y is then modeled as a function of a linear combination of the hidden variables

$$\hat{Y} = f(X) = g\left(\beta_0 + \beta^T Z\right).$$

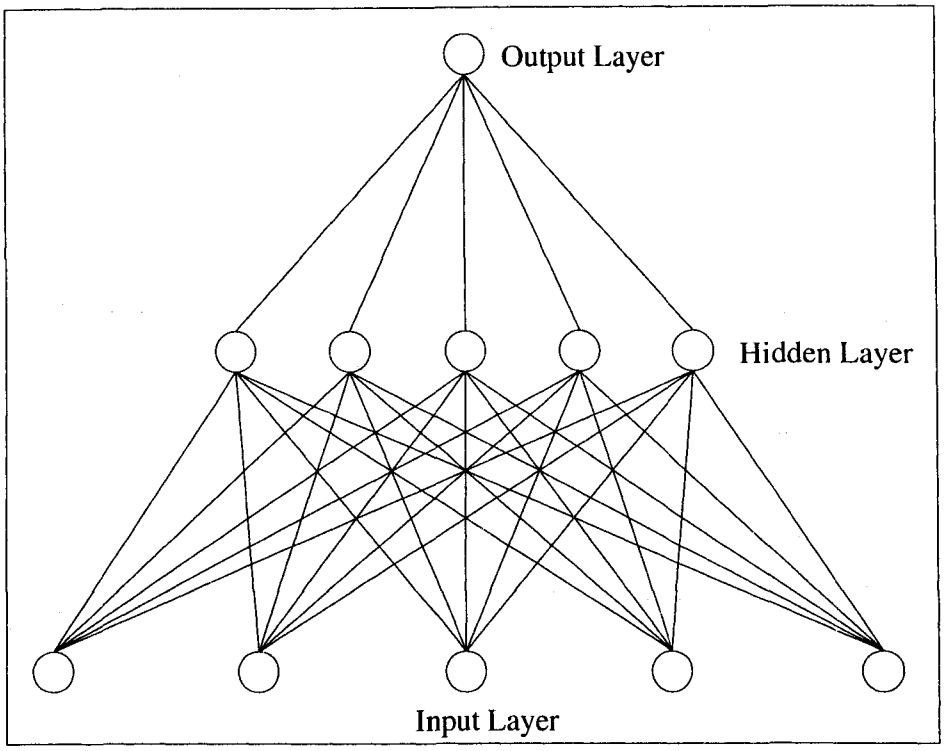

FIGURE 1.1. A graphical depiction of a neural network.

Classification tries to determine the class of a particular input X. Support vector machines, neural networks, classification trees, and k-Nearest-Neighbor classifiers are all examples of classification algorithms. **Support vector machines** (Hastie, Tibshirani, and Friedman 2001k), as explained in more detail in Section 1.4, use constrained optimization to find hyperplanes separating classes of data. These hyperplanes define regions of space for the classes. A point is classified according to the region of space it falls into. **Neural networks** (see above) may also be used for classification. Here, the neural network is generalized to the multi-output case. Each of the outputs represents a particular class and is constrained to lie in the interval $[0, 1]$. There is one output function

$g_k(\beta_{0k} + \beta_k^T Z)$, $k = 1, \ldots, K$ for each of the K outputs that is a function of a separate linear combination of the hidden variables. If the *softmax* function,

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}$$

where $T_k = \beta_0 k + \beta_k^T Z$, $k = 1, \ldots, K$, is used, the numerical value of the k^{th} output is positive and the values of the output functions sum to 1 (Hastie, Tibshirani, and Friedman 2001j). The value of the output can be interpreted as the probability that a point is in class k. **Classification trees** (Hastie, Tibshirani, and Friedman 2001l) typically split the feature space recursively into regions. Observations in a node are classified according to the majority class in that node. **K-Nearest-Neighbor** classifiers (Hastie, Tibshirani, and Friedman 2001m) classify a point based on the class of its nearest neighbors.

Model evaluation is somewhat easier for supervised learning tasks as the model prediction can be compared to the actual outcome. For regression, a loss function is defined, which is a function of the difference between the actual outcome Y and the predicted outcome $\hat{f}(X)$. Typically, squared error or absolute error is used

$$L(Y, \hat{f}(X)) = \begin{cases} \left(Y - \hat{f}(X)\right)^2 & \text{squared error} \\ \left|Y - \hat{f}(X)\right| & \text{absolute error} \end{cases}$$

The test error is the expected value of the loss function and is calculated over an independent test set. According to Tibshirani et al (2000), training error

$$\overline{err} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i)),$$

is generally a bad estimate of test error. This is because training error can be reduced by increasing model complexity, thus adapting more and more to underlying structure in the data. However, this underlying structure may be a peculiarity of the training set (e.g. noise) and may not be present in future data. Models that have adapted themselves too closely to the data are said to have overfit the data. While such a model will have low training error, it

will not generalize well and its test error will be large. Other methods are used to estimate the test error for regression tasks.

For classifiers, one simple measure of performance is accuracy. Accuracy is the fraction of correct predictions. However, Provost et al (1998) argue that comparing accuracy for classifiers may be misleading. They state that accuracy does not account for differences in misclassification costs among the classes and is also dependent on class distribution. If one class has fewer instances than other classes, accuracy will not capture a complete classification failure on that class. For a binary or two-class classifier consisting of a "positive class" and a "negative class", Provost et al propose that the Receiver-Operator Characteristic (ROC) curve be used instead to compare classifier performance. The ROC curve is a plot of the true positive rate versus false positive rate (Swets 1988). The true positive rate (TP) is the fraction of correct predictions for the positive class. The false positive (FP) rate is the fraction of incorrect predictions for the negative class.

1.3 Standard Data Sets

Patrick M. Murphy and David Aha created the UCI Machine Learning Repository (Blake and Merz 1998), a collection of (mostly) free data sets donated to the data mining community for study. The repository is currently maintained by Catherine Blake and Christopher J. Merz and contains over 100 data sets from domains as diverse as health care, civil engineering, and image processing. The data mining tasks range from binary classification to multi-class classification to prediction of continuous variables. Of particular interest to this research were the data sets for binary classifiers. The mushroom, Wisconsin breast cancer, ionosphere, liver, Pima Indians diabetes, tic-tac-toe, congressional voting records, SPECT and SPECTF, contraceptive method choice, Haberman's survival data, and the MUSK data sets were used and are described below. Additional information on the fields for each data set can be found in Appendix A.

The mushroom data set contains descriptions of 23 species in the Agaricus and Lepiota

Family. These descriptions were taken from the 1981 version of The Audubon Society Field Guide to North American Mushrooms. From these, 8124 hypothetical instances were created, 4208 of which correspond to edible mushrooms and 3916 correspond to poisonous mushrooms. There are 22 features which include various descriptions of the mushroom cap, gills and stalks. All features are categorical and the number of categories ranges from 2 to 12. J. S. Schlimmer (1987) achieved 95% accuracy on this data set using his STAGGER algorithm. STAGGER is an incremental learning technique using a connectionist scheme to represent the knowledge (essentially an neural network that evolves or changes as new training data is encountered). W. Iba et al (1988) achieved similar accuracy with their HILLARY algorithm. HILLARY is also an incremental algorithm, but only stores negative instances.

The Wisconsin breast cancer data set (wpbc) contains follow-up data for 198 different cases seen by one doctor. In 47 of these cases, the cancer recurred and in 151, cancer did not. There are 35 features in this data set, 30 of which describe cell nuclei observed in a fine needle aspirate of a breast mass. The remaining features include an ID number, a binary indicator of whether cancer has recurred, the recurrence time or disease-free time based on the value of the recurrence time, the diameter of the tumor removed, and the number of positive axillary lymph nodes observed at the time of surgery. Most of the features, excluding the ID and recurrence indicator are continuous. This data set has been studied extensively by Street, Mangasarian and Wolberg (1995) and Mangasarian, Street and Wolberg (1995).

The ionosphere data set consists of radar data targeting free electrons in the ionosphere collected by an array of 16 high-frequency antennas. A signal is considered to be good if it shows structure in the ionosphere and bad if it passes through. This set contains 351 instances and contains 34 continuous features. 225 of the signals showed structure and 126 did not. Sigillito et al (1989) used a perceptron training algorithm. Their training set consisted of the first 200 points and their test set consisted of the remaining points. Performance on the test set was at least 90%.

The liver data set (bupa) contains results of five blood tests which are believed to predict liver disorders caused by excessive alcohol consumption. The blood tests measure mean corpuscular volume, alkaline phosphotse, alamine aminotransferase, aspartate aminotransferase, and gamma-glutamyl transpeptidase. Each of the 345 instances contains the results from for an individual male as well as the number of drinks per day he has. The task is to predict which group an individual will fall into. The groups are labeled 1 and 2 and are assumed to indicate that an individual has a liver disorder or does not have a liver disorder. However, no indication was given as to which label corresponded to which class. 145 of the instances were in group 1 and 200 were in group 2. This data set was originally studied by R. S. Forsyth (1990) in the PC/BEAGLE User's Guide.

The Pima Indians diabetes data set contains attributes of 768 Pima Indian women at least 21 years old. 268 of these women tested positive for diabetes. All eight features are numerical and include the number of times pregnant, blood pressure, age and body mass index. The task is to predict whether a woman has diabetes or not. Smith et al (1988) originally investigated this data set using their perceptron-like adaptive learning algorithm, ADAP. They used 576 data points as their training set and the remaining points as their test set. The ADAP algorithm achieved 76% accuracy on the test set.

The tic-tac-toe data set consists of all possible configurations of the tic-tac-toe board at the end of a game. There are 9 features, one for each square and three possible states for a square ('x' owns the square, 'o' owns the square, or the square is blank). All features are categorical. The class variable indicates whether 'x' wins the game or not. This data set contains 958 end game instances, 626 of which were games where 'x' won and 332 were games where 'x' did not win. The CITRE decision tree algorithm was used on this data set by Matheus and Rendell (1989) and Matheus (1990). CITRE is an algorithm that constructs appropriate features from the original feature set using decision trees and a small amount of expert or domain knowledge. Their highest accuracy was over 90% for training sets of 500 data points. This data set was also investigated by Aha (1991). He compared results from several algorithms including IB3-CI, IB3, IB1, and CN2. IB3-CI is an extension of

IB3, an instance-based learning algorithm that only retains misclassified data points and uses a significance test to update. IB3-CI incorporates feature construction by reducing the similarity between two data points upon misclassification. IB3 is an extension of IB1 (Aha, Kibler, and Albert 1991), an instance-based learning algorithm that retains all points. CN2 (Clark and Boswell 1991) is an algorithm that generates 'if...then' rules and can compensate for noisy data. Aha (1991) used 70% of the data set as his training data and the remainder as his test set. The accuracy on the test set for six algorithms ranged from 82% for IB3 to 98% for IB1 and CN2 to 99% for IB3-CI.

The congressional voting records data set consists of the votes on 16 issues of each member of the House of Representatives of the 98th Congress from the second session of 1984. The source of this data is the Congressional Quarterly Almanac which actually lists nine types of votes: voted for, paired for, announced for, voted against, paired against, announced against, voted present, voted present to avoid conflict of interest, and did not vote. These nine vote types were simplified to three vote types: yes, no, and unknown. There were 435 total members, 168 were Republicans and 267 were Democrats. The learning task is to predict whether an individual is a Republican or a Democrat based on their votes. J. C. Schlimmer (1987) investigated this data set using his STAGGER algorithm (see the mushroom data set) and achieved at least 90% accuracy on this data set.

The SPECT and SPECTF data sets contain measured characteristics from 276 images of cardiac Single Proton Emission Computed Tomography. Each patient is diagnosed as either normal or abnormal. The SPECTF data set consists of the 44 measurements that essentially summarize the SPECT image. The SPECT data set has processed the SPECTF data further to obtain 22 binary features indicating either normal measurements or abnormal measurements. In addition, both data sets have been prepartitioned into training and test sets. The training sets consist of 80 data points equally split between the two classes. The test sets consist of 187 data points, 15 of which belong to the normal class and 172 belong to the abnormal class. This data set has been previously examined by Kurgan et al (2001), who used the CLIP3 rule generating algorithm. CLIP3 by Cios et al (1997) partitions the

training data set into groups such that noise in the data is reduced. It then generates rules from the groups using an integer program. Using CLIP3, Kurgan et al achieved an accuracy of 84% on the SPECT data set and an accuracy of 77% on SPECTF.

The contraceptive method choice data set is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. It contains ten demographic and socio-economic items from 1473 married Indonesian women who were not pregnant (or didn't know) at the time the data was collected. 844 of these women used some form of birth control and 629 of these women did not. The task is to predict which contraceptive method a woman will use, either no method, a long term method or a short term method. This data set was studied by Lim et al (2000) in their examination of 22 classification tree algorithms, nine statistical classification algorithms, and two neural network algorithms. Lim et al studied various characteristics of these algorithms including accuracy and training time. They estimated the error rate for each algorithm using a 10-fold cross-validation experiment and their lowest error rate for this data set was 0.434.

Haberman's survival data contains 306 patients that had undergone surgery for breast cancer at the University of Chicago's Billings Hospital between 1958 and 1980. 225 patients survived at least five years after surgery and 81 did not. The three features for this data set are the patient's age at the time of surgery, the year the operation was performed in, and the number of positive axillary (lymph) nodes detected. All features are numerical and the task is to predict whether a patient will survive at least five years. This data set was studied by Landwehr et al (1984) who fit a logistic regression model to the data. They used this model to predict the probability that a breast cancer patient would survive five years beyond surgery.

The MUSK data set contains 476 instances of descriptions of 92 molecules. Forty-seven of these molecules (207 instances) have been determined to be musks[1] by humans. The rest are non-musks. As described in Dietterich et al (1997), each instance has 166 features. Four of these features describe the distance of the oxygen atom in the molecule from a particular

[1] A molecule that has a musky odor.

point. The rest of the 192 features are distances along rays that describe the shape of the molecule. Dietterich et al (1997) compared the classifiers generated by several axis-parallel rectangle algorithms to the C4.5 classification tree and a neural network. Axis-parallel rectangle algorithm classifiers use a rectangular decision boundary whose sides are parallel to the axes in features space, to classify data. All points that fall within the rectangular boundary are predicted to be in one class. Dietterich et al (1997) achieved an accuracy of 92% on this data set.

1.4 Binary Classifiers

Binary Classifiers are a subset of general classifiers. The task is to determine to which of two classes an item belongs. Classification tress, neural networks and support vector machines can all be used as binary classifiers.

1.4.1 The Support Vector Machine Classifier

The Support Vector Machine Classifier (SVM) (Boser et. al. 1992, Vapnik 1996) attempt to classify data by finding a separating hyperplane between classes of data. This hyperplane is determined by finding two parallel bounding planes, one for each class of data, that are as far apart as possible. The space between the bounding planes is called the margin and those points that lie on either of the two bounding planes are called support points or support vectors.

Let M be the set of training points corresponding to class 1, N be the set corresponding to class 2, and F be the set of features. The training data is stored in matrices $\mathbf{x}^1$ $(M \times F)$ and $\mathbf{x}^2$ $(N \times F)$, one row for each data point. $w_k, k \in F$ are the hyperplane coefficients and γ is the hyperplane "intercept". Δ_i^1 is the distance from the i^{th} point in class 1 to the bounding plane for class 1 and Δ_j^2 is the distance from the j^{th} point in class 2 to the

bounding plane for class 2. The SVM formulation is shown below.

$$\begin{aligned}
\min_{(w,\gamma,\Delta^1,\Delta^2)} \quad & \nu\sum_{i\in M}\Delta_i^1 + \nu\sum_{j\in N}\Delta_j^2 + \frac{1}{2}\sum_{k\in F} w_k^2 + \frac{1}{2}\gamma^2 \\
s.t. \quad & \sum_{k\in F} x_{ik}^1 w_k - \gamma + \Delta_i^1 \geq 1, \qquad i \in M \\
& -\sum_{k\in F} x_{jk}^2 w_k + \gamma + \Delta_j^2 \geq 1, \qquad j \in N \\
& \Delta_i^1 \geq 0, \qquad i \in M \\
& \Delta_j^2 \geq 0, \qquad j \in N
\end{aligned}$$

The objective function contains two terms, an error term and a hyperplane term. The hyperplane term is the reciprocal of the margin width or distance between the bounding planes, see Figure 1.2. Thus by minimizing this term, we are maximizing the margin width (Burges 1998). The error term attempts to minimize the distance from a point to its bounding plane for those points on the wrong side of the bounding plane corresponding to its class. A point is on the wrong side of the bounding plane of its class if it lies within the margin or beyond the bounding plane of the other class. It is only in these two cases that the individual error variable Δ_i^1 or Δ_j^2 is non-zero. Since an actual misclassification error occurs when Δ_i^1 or Δ_j^2 is larger than 1, $\sum_{i\in M}\Delta_i^1 + \sum_{j\in N}\Delta_j^2$ is an upper bound on the number of misclassified points (Burges 1998). These two terms exert opposite pressures on the location of the bounding planes.

1.4.2 Goal Programming and the Support Vector Machine Classifier

An alternate paradigm for describing SVM can be found in the multi-objective optimization technique of goal programming (Ignizio and Cavalier 1994).

For classification purposes, we would like to position the bounding hyperplanes such that all data points lie outside of the bounding planes. For points x_i^1 in class 1, we have the goal

$$w^T x_i^1 \geq \gamma + 1, \; i \in M,$$

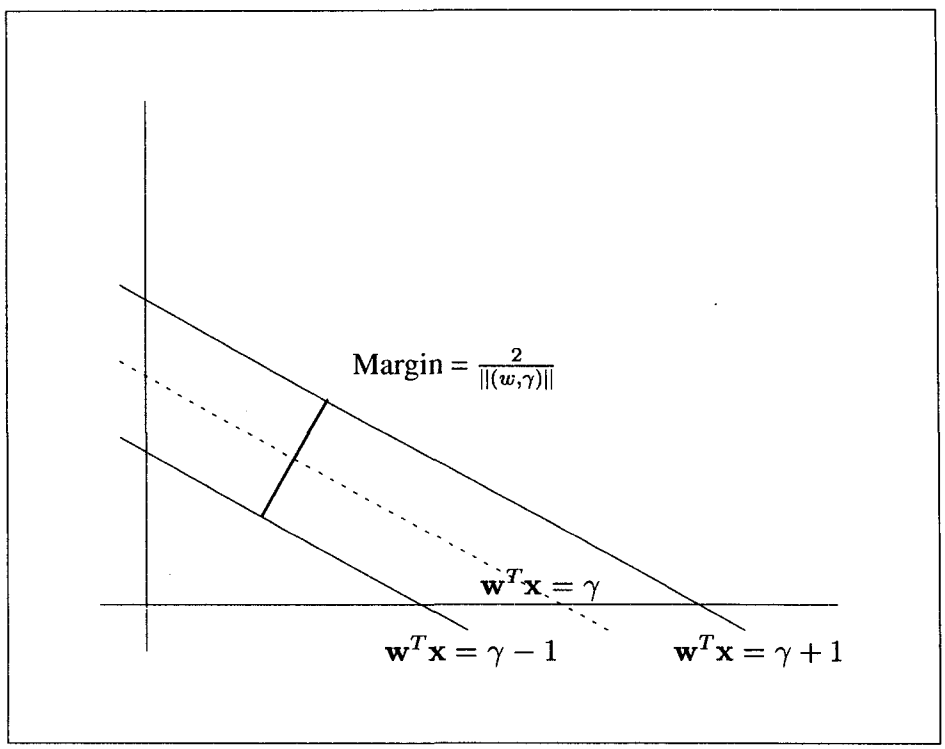

FIGURE 1.2. Illustration of the margin for SVM.

which may be rewritten as

$$w^T x_i^1 - \gamma \geq 1,\ i \in M.$$

Similarly, for points in class 2, we have the goal

$$w^T x_j^2 \leq \gamma - 1,\ j \in N,$$

which can be rewritten as

$$-w^T x_j^2 + \gamma \geq 1,\ j \in N.$$

For real data, we may not be able to satisfy all the goals. We then add slack variables that represent the positive and negative deviations from the goals, Δ^{1+}, Δ^{1-}, Δ^{2+} and Δ^{2-} respectively. Positive deviations are subtracted from the left hand sides of the goals and negative deviations are added. The goals become

$$w^T x_i^1 - \gamma + \Delta_i^{1-} - \Delta_i^{1+} = 1,\ i \in M$$

and

$$-w^T x_j^2 + \gamma + \Delta_j^{2-} - \Delta_j^{2+} = 1,\ j \in N.$$

In goal programming, a function of the unwanted deviations is minimized. In this particular problem, the unwanted deviations are the negative deviations Δ^{1-} and Δ^{2-}. In Archimedean goal programming, a weighted sum of the unwanted deviations is used. For this problem, none of the goals is more important than the other so they are weighted equally. We then get the following linear program

$$\begin{aligned}
\min \quad & \sum_{i \in M} \Delta_i^{1-} + \sum_{j \in N} \Delta_j^{2-} \\
\text{s.t.} \quad & w^T x_i^1 - \gamma + \Delta_i^{1-} - \Delta_i^{1+} = 1, && i \in M \\
& -w^T x_j^2 + \gamma + \Delta_j^{2-} - \Delta_j^{2+} = 1, && j \in N \\
& \Delta_i^{1+}, \Delta_i^{1-} \geq 0, && i \in M \\
& \Delta_j^{2+}, \Delta_j^{2-} \geq 0, && j \in N.
\end{aligned}$$

This formulation is very similar to the SVM formulation, the only real difference is the missing hyperplane term in the objective function.

1.4.3 The Linearized Support Vector Machine Classifier

Kecman and Arthanari (2001) reformulated SVM into a linear program by using the L_1 norm instead of the L_2 norm. In using the L_1 norm, it is necessary to introduce new variables representing the positive and negative parts of the components of the weight vector, $\mathbf{w}$, thus transforming the absolute value function into a linear function. Using the notation defined above, the formulation is

$$\begin{aligned}
\min_{(w,\gamma,\Delta^1,\Delta^2)} \quad & \nu\sum_{i\in M}\Delta_i^1 + \nu\sum_{j\in N}\Delta_j^2 + \sum_{k\in F}\left(w_k^+ + w_k^-\right) + \gamma^+ + \gamma^- & \\
s.t. \quad & \sum_{k\in F} x_{ik}^1\left(w_k^+ - w_k^-\right) - \gamma^+ + \gamma^- + \Delta_i^1 \geq 1, & i \in M \\
& -\sum_{k\in F} x_{jk}^2\left(w_k^+ - w_k^-\right) + \gamma^+ - \gamma^- + \Delta_j^2 \geq 1, & j \in N \\
& \Delta_i^1 \geq 0, & i \in M \\
& \Delta_j^2 \geq 0, & j \in N \\
& w_k^+, w_k^- \geq 0, & k \in F \\
& \gamma^+, \gamma^- \geq 0. &
\end{aligned}$$

1.4.4 Proximal Support Vector Machine Classifiers

Proximal Support Vector Machine Classifiers (PSVM) proposed by Fung and Mangasarian (2001) are a modification of SVMs. They start with a modification of SVM by Mangasarian and Musicant (2000, 2001) . This reformulation of SVM replaced the L_1 norm of the error variables Δ^1 and Δ^2 with the L_2 norm, thus eliminating the need for the non-negativity constraints on Δ^1 and Δ^2. This first formulation is

$$\begin{aligned}
\min_{(w,\gamma,\Delta^1,\Delta^2)} \quad & \frac{\nu}{2}\sum_{i\in M}(\Delta_i^1)^2 + \frac{\nu}{2}\sum_{j\in N}(\Delta_j^2)^2 + \frac{1}{2}\left(\sum_{k\in F} w_k^2 + \gamma^2\right) & \\
s.t. \quad & \sum_{k\in F} x_{ik}^1 w_k - \gamma + \Delta_i^1 \geq 1, & i \in M \\
& -\sum_{k\in F} x_{jk}^2 w_k + \gamma + \Delta_j^2 \geq 1, & j \in N.
\end{aligned}$$

In addition to removing a set of constraints, the objective function is now strongly convex.

In addition to minimizing the L_2 norm of the error variables Δ^1 and Δ^2, the inequality constraints are replaced by equality constraints. This has the added advantage of allowing an exact solution to be found using the Karush Kuhn Tucker (KKT) conditions. In addition, this modification changes the geometric interpretation. In SVM, the planes $\mathbf{w}^T\mathbf{x} = \gamma \pm 1$ are thought of as bounding planes, i.e. the data lies "outside" the planes. According to Fung and Mangasarian (2001), the equality constraints now cause the planes to be positioned such that the data points are clustered around them.

Using the notation defined above, the quadratic program for PSVM is

$$\begin{aligned} \min_{(w,\gamma,\Delta^1,\Delta^2)} \quad & \frac{\nu}{2}\sum_{i\in M}(\Delta_i^1)^2 + \frac{\nu}{2}\sum_{j\in N}(\Delta_j^2)^2 + \frac{1}{2}\left(\sum_{k\in F} w_k^2 + \gamma^2\right) & \\ s.t. \quad & \mathbf{x}_i^1\mathbf{w} - \gamma + \Delta_i^1 = 1, & i \in M \\ & -\mathbf{x}_j^2\mathbf{w} + \gamma + \Delta_j^2 = 1, & j \in N. \end{aligned}$$

To write the KKT conditions for this problem, we first need to write the Lagrangian. Let σ_1 and σ_2 be the Lagrange multipliers for the constraint sets. The Lagrangian is

$$\begin{aligned} L(\mathbf{w}, \gamma, \boldsymbol{\Delta^1}, \boldsymbol{\Delta^2}, \boldsymbol{\sigma_1}, \boldsymbol{\sigma_2}) \quad = \quad & \frac{\nu}{2}\left(\sum_{i\in M}(\Delta_i^1)^2 + \sum_{j\in N}(\Delta_j^2)^2\right) + \frac{1}{2}\left(\sum_{k\in F} w_k^2 + \gamma^2\right) \\ & -\sum_{i\in M}\sigma_{1i}\left(\mathbf{x}_i^1\mathbf{w} - \gamma + \Delta_i^1 - 1\right) \\ & -\sum_{j\in N}\sigma_{2j}\left(-\mathbf{x}_j^2\mathbf{w} + \gamma + \Delta_j^2 - 1\right). \end{aligned}$$

The KKT conditions are found by taking the partial derivatives of the Lagrangian and

setting those equal to zero. In addition, the original constraints are included.

$$\begin{aligned}
\mathbf{w} - \sum_{i \in M} \sigma_{1i} \mathbf{x}_i^1 + \sum_{j \in N} \sigma_{2j} \mathbf{x}_j^2 &= 0 \\
\gamma + \sum_{i \in M} \sigma_{1i} - \sum_{j \in N} \sigma_{2j} &= 0 \\
\nu \Delta_i^1 - \sigma_{1i} &= 0, \quad i \in M \\
\nu \Delta_j^2 - \sigma_{2j} &= 0, \quad j \in N \\
\mathbf{x}_i^1 \mathbf{w} - \gamma + \Delta_i^1 - 1 &= 0, \quad i \in M \\
-\mathbf{x}_j^2 \mathbf{w} + \gamma + \Delta_j^2 - 1 &= 0, \quad j \in N.
\end{aligned}$$

The first four sets of equations are solved for the Lagrange multipliers, σ_1 and σ_2,

$$\begin{aligned}
\mathbf{w} &= \sum_{i \in M} \sigma_{1i} \mathbf{x}_i^1 - \sum_{j \in N} \sigma_{2j} \mathbf{x}_j^2 \\
\gamma &= -\sum_{i \in M} \sigma_{1i} + \sum_{j \in N} \sigma_{2j} \\
\Delta_i^1 &= \frac{\sigma_{1i}}{\nu}, \qquad i \in M \\
\Delta_j^2 &= \frac{\sigma_{1j}}{\nu}, \qquad j \in N,
\end{aligned}$$

and are substituted into the last two equations which are then solved for the Lagrange multipliers.

PSVM has several advantages over SVM. First, a constraint has been removed from the formulation. Second, due to the pure convexity of the objective function, an exact solution can be written in terms of the data. This second advantage allows a solution to be found by solving a set of linear equations.

1.5 Extensions to SVM and PSVM with Application

In this dissertation, we provide two modeling extensions and a solution strategy for a novel application. The first extension presented will be a direct extension of Kecman and Arthanari's work to PSVM. The quadratic portions of PSVM's objective function are rewritten

using the absolute value function. This is then reformulated into a linear program, which is easier and faster to solve than a quadratic program and reduces the influence of outliers. A simple two dimensional data set consisting of two classes, each sampled from two bivariate Gaussian distributions, is used for initial tests. The distributions were chosen so they could be distinguished visually and so they overlapped enough to potentially confuse the classifiers. Performance is evaluated by examining accuracy, true positive rate and false positive rate. All three are calculated over the entire data set. Finally, Linearized PSVM, PSVM, and SVM are tested on eleven data sets from the UCI Machine Learning Repository (Blake and Merz 1998).

The second extension reformulates SVM as a mixed-integer linear program (ISVM). While solution time is increased over the other SVM-type classifiers, ISVM minimizes the number of errors instead of a distance. Since the correctness of the classification matters and not the magnitude of the misclassification, ISVM more closely models the classification task. The analysis performed on Linearized PSVM was also performed on ISVM.

All four classifiers, SVM, PSVM, Linearized PSVM and ISVM are then applied to collusion detection in online gambling. Since collusion is a particular behavior of gamblers, it should manifest itself in their actions during play. Given a data set where the colluders are known as well as their actions, it was hypothesized that a model could be built to classify games where collusion occurred. In this application, collusion in Texas Hold'Em, a popular poker variant was examined. The data from the Internet Relay Chat (IRC) poker channels was used to build a classifier for one particular collusion method.

CHAPTER 2

LINEARIZED PROXIMAL SUPPORT VECTOR MACHINE CLASSIFIER

2.1 Discussion

Support Vector Machine classifiers have an easy geometrical interpretation, the classifier is a plane that separates the classes of data. However, both SVM and PSVM are quadratic programs and can be difficult to solve. In the case of PSVM, the quadratic error term gives more weights to outliers. Both SVM and PSVM will tend to find non-zero weights for most features (Kecman and Arthanari 2001). By rewriting PSVM as a linear program, the influence of outliers will be reduced without an increase in solution time.

We will use the following notation. Let F be the set of features. We define M as the set of points belonging to class 1 and N as the set of points belonging to class 2. The data for class 1 is contained in matrix x and the data for class 2 is contained in matrix y. $w_k, k \in F$ and γ are the coefficients for the separating or classifying hyperplane. In addition, we need several auxiliary variables. Δ_i^1 measures the distance of the i^{th} point in class 1 from the proximal plane $w^T x = \gamma + 1$. Similarly, Δ_j^2 measures the distance of the j^{th} point in class 2 from the proximal plane $w^T y = \gamma - 1$.

The first step in writing PSVM as a linear program involves replacing the replacing the L_2 norm by the L_1 norm. Using the notation above, we obtain

$$\begin{aligned} \min \quad & \mu \sum_{i \in M} |\Delta_i^1| + \nu \sum_{j \in N} |\Delta_j^2| + \beta \left(\sum_{k \in F} |w_k| + |\gamma| \right) \\ \text{s.t.} \quad & \sum_{k \in F} x_{ik} w_k - \gamma + \Delta_i^1 = 1, \qquad i \in M \\ & -\sum_{k \in F} y_{jk} w_k + \gamma + \Delta_j^2 = 1, \qquad j \in N, \end{aligned}$$

where μ, ν, and β are parameters to control the relative importance of the errors and the

width between the proximal planes.

However, this is still not a linear program due to the absolute value functions. We replace each free variable (**w**, γ, Δ^1, and Δ^2) with two variables representing the positive and negative parts of that variable, i.e.

$$w_k = w_k^+ - w_k^-.$$

The absolute value is replaced by the sum of the positive and negative parts

$$|w_k| = w_l^+ + w_k^-.$$

We then get

$$\begin{aligned}
\min \quad & \mu \sum_{i \in M} \left(\Delta_i^{1+} + \Delta_i^{1-}\right) + \nu \sum_{j \in N} \left(\Delta_j^{2+} + \Delta_j^{2-}\right) + \beta \left(\sum_{k \in F} \left(w_k^+ + w_k^-\right) + \gamma^+ + \gamma^- \right) \\
\text{s.t.} \quad & \sum_{k \in F} x_{ik} \left(w_k^+ - w_i^-\right) - \gamma^+ + \gamma^- + \Delta_i^{1+} - \Delta_i^{1-}, = 1 && i \in M \\
& -\sum_{k \in F} y_{jk} \left(w_k^+ - w_k^-\right) + \gamma^+ - \gamma^- + \Delta_j^{2+} - \Delta_j^{2-} = 1, && j \in N \\
& w_k^+, w_k^- \geq 0, && k \in F \\
& \gamma^+, \gamma^- \geq 0 \\
& \Delta_i^{1+}, \Delta_i^{1-} \geq 0, && i \in M \\
& \Delta_j^{2+}, \Delta_j^{2-} \geq 0, && j \in N.
\end{aligned}$$

A potential difficulty with this model and all SVM-type models is a scaling mismatch between the error terms and the hyperplane terms. The standard technique of normalizing the terms to the same scale cannot be used here since the scale of the terms cannot be known a priori. However, the objective function parameters can be used to compensate for this difficulty. As default values for the parameters, we suggest setting μ, ν, and β equal to 1 or to $1/3$. If an satisfactory solution is not obtained, then the parameters can be changed to attempt to find a better solution. Parameter setting is discussed for a simple example in Section 2.2.1.

2.2 Visualization Examples

The feature space in most data mining applications is quite large and extremely difficult to visualize. For ease in visualization, a non-separable two dimensional data set was created to compare Linearized PSVM to SVM and PSVM. This data set consists of 500 points per class. Each class was generated from a bivariate normal distribution. The parameters for each class are shown in table 2.1. These parameters were chosen such that the classes

Class	μ_x	μ_y	σ_x	σ_y	c
1	2	2	3	3	-0.75
2	5	5	3	3	-0.75

TABLE 2.1. Parameters used to generate the classes in the 2-D data set.

were reasonably compact and easy to distinguish visually, but also overlapped slightly. The overlapping points are an attempt to confuse the algorithms.

The solutions to the various optimization problems were obtained as follows. SVM, PSVM, and Linearized PSVM were translated into AMPL (Fourer, Gay, and Kernighan 2002), an algebraic language for modeling linear and non-linear optimization problems. These models were solved using various solvers provided by the NEOS Optimization server (Czyzyk, Mesnier, and Moré 1998) (Gropp and Moré 1997) (Dolan 2001). SVM and PSVM were submitted to the MINOS solver (version 5.5) (Murtagh and Saunders 1998) and Linearized PSVM was submitted to the FortMP solver (version 3.2e) (FortMP 1999). All three were submitted to NEOS via its email interface.

The NEOS Optimization server is a distributed computing environment. Problem requests submitted to NEOS are received by a central server which parses the requests and determines which machine to send the problem to for solving based on the machine load and available software. The user does not have control over which machine receives a job. In order to perform a meaningful comparison of computing times, all three problems should be solved on the same machine. Since this is not possible with NEOS, a comparison of computing times was not performed. In practice, results for all three classifiers were

returned by NEOS almost instantaneously.

For each model, the entire data set was used as the training set. The hyperplanes found by SVM, PSVM and Linearized PSVM are shown in table 2.2.

Algorithm	Hyperplane
SVM	$y = 6.9254 - 1.0206x$
PSVM	$y = 7.3354 - 1.0696x$
Linearized PSVM	$y = 7.0652 - 1.0166x$

TABLE 2.2. Hyperplanes found by SVM, PSVM, and Linearized PSVM for a non-separable 2-D data set.

A plot of the data set and the classifying hyperplane and bounding or proximal hyperplanes along with a plot of the data lying in the margin for SVM can be found in Figures 2.1 and 2.2. The plots for PSVM can be found in Figures 2.3 and 2.4. Finally, the same plots for Linearized PSVM can be found in Figures 2.5 and 2.6.

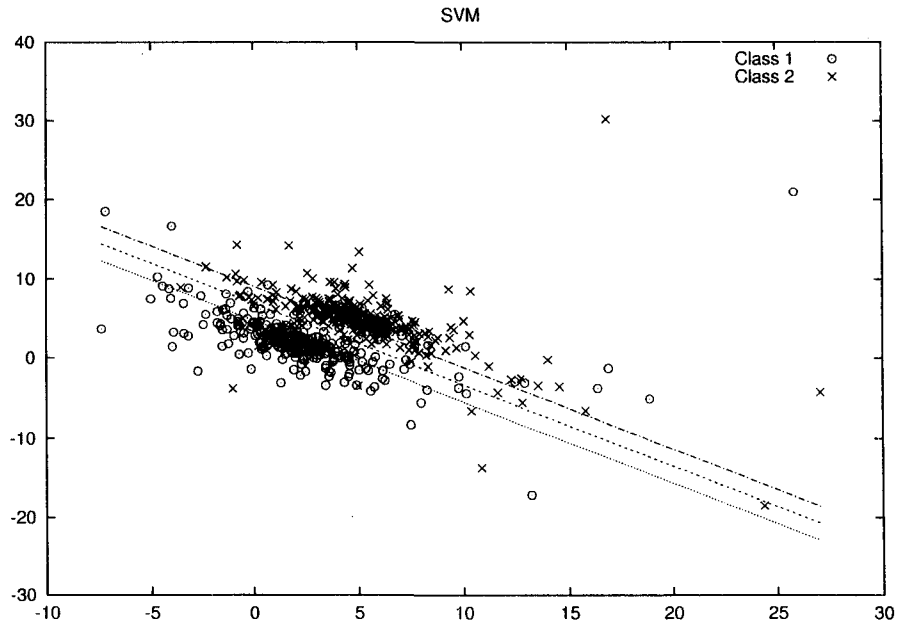

FIGURE 2.1. Classifying hyperplane and bounding hyperplanes for SVM

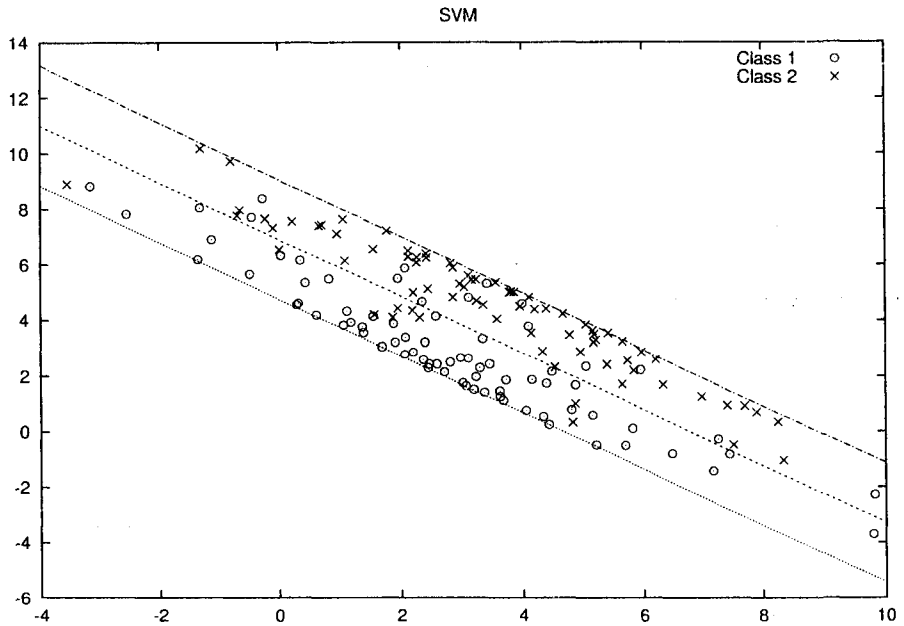

FIGURE 2.2. Expanded view of the data lying in the margin for SVM

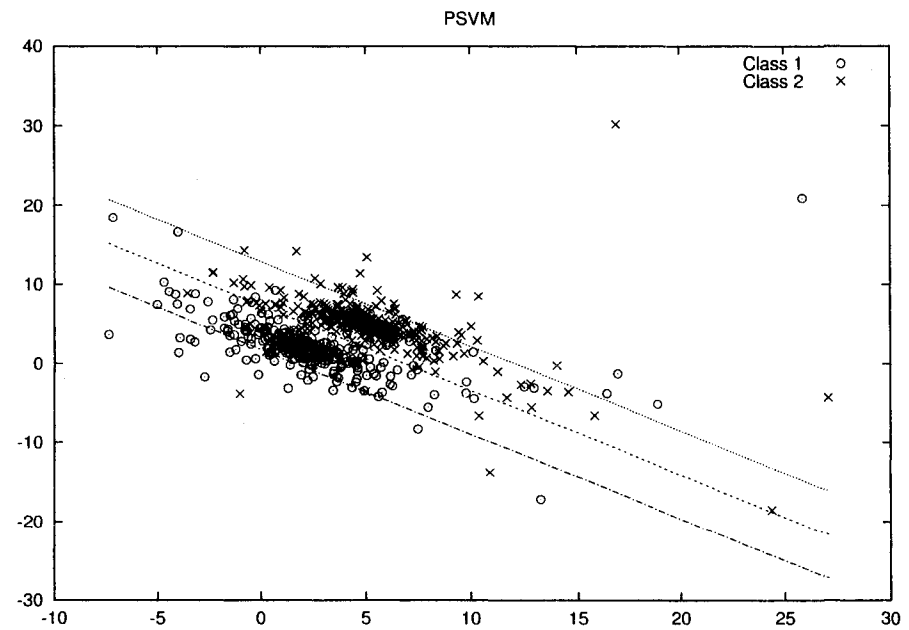

FIGURE 2.3. Classifying hyperplane and bounding hyperplanes for PSVM

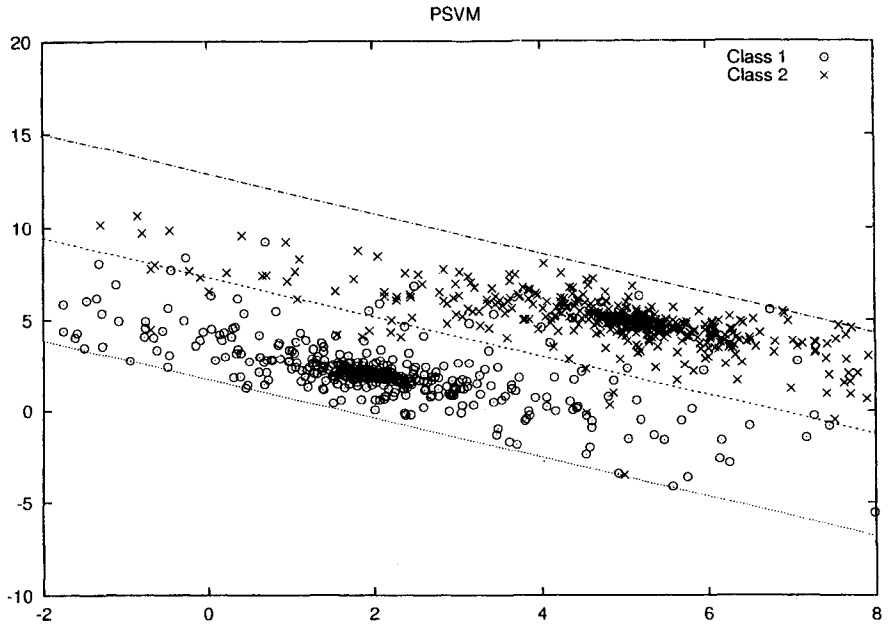

FIGURE 2.4. Expanded view of the data lying in the margin for PSVM

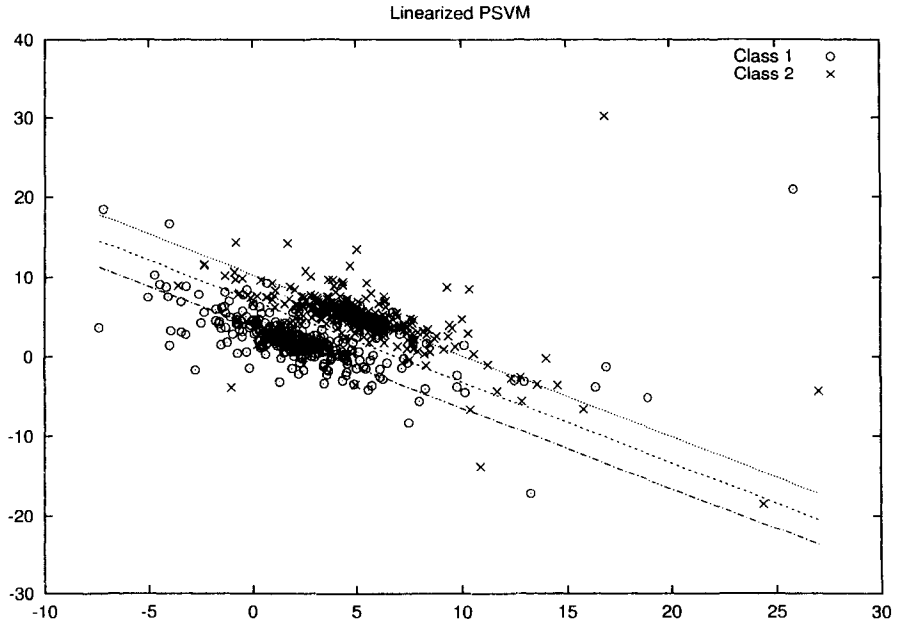

FIGURE 2.5. Classifying hyperplane and bounding hyperplanes for Linearized PSVM

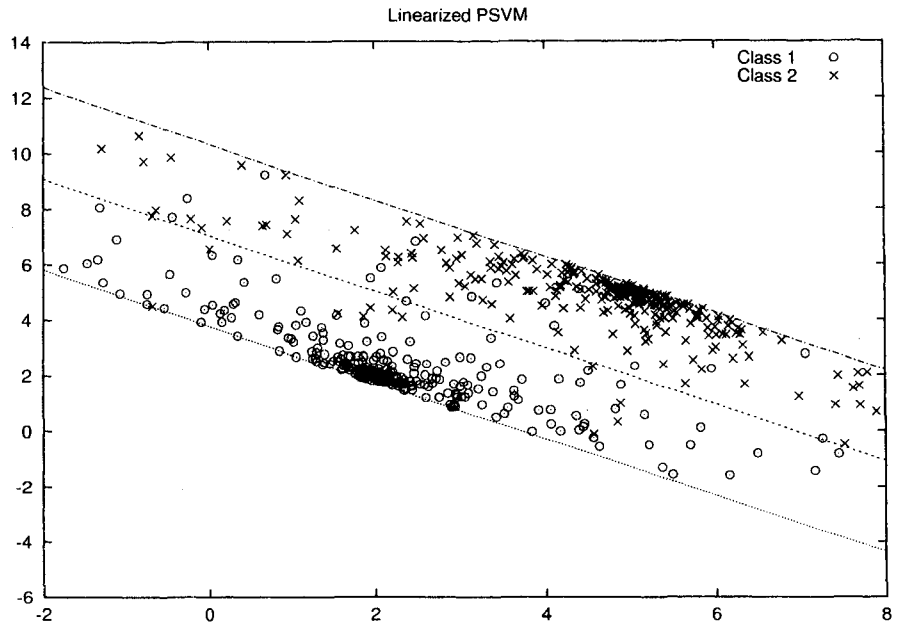

FIGURE 2.6. Expanded view of the data lying in the margin for Linearized PSVM

According to Fung and Mangasarian (2001), PSVM is supposed to cluster the data around the proximal planes. At least with this data set, it does not do this. In addition, the linear formulation of PSVM does cluster the data around the proximal planes. The use of the L_2 norm in PSVM weights outliers more heavily than the L_1 norm in Linearized PSVM. This puts additional pressure on the bounding planes and tends to pull them apart. However, this discussion is irrelevant if the classifiers perform poorly.

In order to obtain a prediction for a data point $\mathbf{x}$, we compute the quantity $p = \mathbf{w}^T\mathbf{x} - \gamma$. If $p > 0$ then the classifier predicts that $\mathbf{x}$ is in class 1. If $p < 0$ then the classifier predicts that $\mathbf{x}$ is in class 2. Finally if $p = 0$, then the classifier cannot make a prediction.

The performance of all three classifiers was examined on the training set, which in this case was the entire data set, using accuracy, true positive rate and false positive rate as defined in Chapter 1. While a Receiver-Operator Characteristic curve may be more enlightening (Provost, Fawcett, and Kohavi 1998), calculating the true positive rate and

false positive rate over the entire data set yields information about classifier performance in a more compact fashion. For analysis purposes, the "positive" class was considered to be class 1 and the "negative" class was considered to be class 2. The results for each classifier on this 2-D data set are shown in table 2.3. The SVM classifier had the highest accuracy and lowest false positive rate. PSVM had the highest true positive rate, however Linearized PSVM had a higher accuracy and lower false positive rate. As we can see, all

Model	Accuracy	True Positive Rate	False Positive Rate
SVM	0.952	0.94	0.036
PSVM	0.95	0.944	0.044
Linearized PSVM	0.951	0.94	0.038

TABLE 2.3. Training set performance for SVM, PSVM and Linearized PSVM for a non-separable 2-D data set.

three classifiers perform similarly on this data set.

2.2.1 Parameter Selection for the 2-D Case

An important consideration in any support vector machine classifier is selecting the weighting parameters (μ, ν, β) in the objective function. In Linearized PSVM, there are three parameters although only two are necessary. However, working with three parameters may be more intuitive. As formulated, Linearized PSVM has 3 terms, one error term for each class and one hyperplane/margin term. Each weighting parameter can be thought of as a cost parameter and can be set to reflect the relative importance of each term. Without any additional information about the data, the parameters can all be set to 1. This gives equal weight to the error terms and the margin term. If it is more important to get one of the classes correct, then either μ or ν may be increased. Additional choices include increasing the weighting on the hyperplane term and even weighting the individual data points differently, reflecting their importance or confidence in their correctness[1]. These additional

[1]This actually increases the number of parameters beyond three.

parameters increase the flexibility of the model. Finally, the parameters may be used to compensate for differences in scale between the error terms and the hyperplane term.

For the non-separable two dimensional data set, the effect of weighting the margin term differently than the error terms was examined. The error terms were weighted equally. The value for μ ranged from 0 to 0.5 and β was set so that the sum of all three parameters was 1. Figure 2.7 shows the effect of changing the parameters on the resulting classifying hyperplanes. Most of the parameter changes appeared to have no effect on the resulting

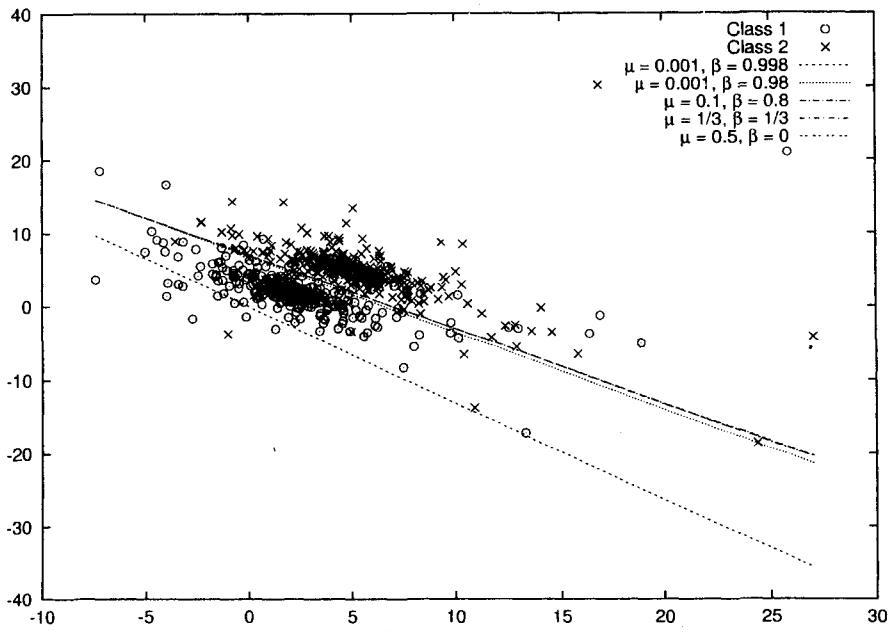

FIGURE 2.7. Varying objective function parameters for Linearized PSVM

hyperplanes. The effect on performance is shown in table 2.4. The performance is mostly unaffected by varying the parameters. A similar study was performed on PSVM. PSVM has a single parameter, ν and was set to range between 0 and 1000. Figure 2.8 shows the effect of changing ν on the hyperplanes. The effect on performance is shown in table 2.5. Both Linearized PSVM and PSVM were relatively insensitive to objective function parameter changes on this data set. The data has most of its mass in 2 regions. Since we

μ	β	Accuracy	True Positive	False Positive
0	1	0	0	1
0.001	0.998	0.509	0.02	0.002
0.01	0.98	0.949	0.932	0.034
0.1	0.8	0.951	0.94	0.038
0.2	0.6	0.951	0.94	0.038
$0.\bar{3}$	$0.\bar{3}$	0.951	0.94	0.038
0.4	0.2	0.951	0.94	0.038
0.5	0	0.951	0.94	0.038

TABLE 2.4. Performance results from sensitivity analysis of Linearized PSVM to changes in objective function parameters.

ν	Accuracy	True Positive	False Positive
0	0.444	0.852	0.964
0.2	0.951	0.944	0.042
0.4	0.95	0.944	0.044
0.6	0.95	0.944	0.044
0.8	0.95	0.944	0.044
1.0	0.95	0.944	0.044
10.0	0.952	0.944	0.04
100	0.95	0.944	0.044
1000	0.95	0.944	0.044

TABLE 2.5. Performance results from sensitivity analysis of PSVM to changes in objective function parameters.

are clustering the proximal planes around the data, shifting points from one side to another will not change the planes much, hence the lines will not change much at all. It is only when the coefficient on the error term goes to zero, which allows clustering on the points to become unimportant that the performance is greatly affected.

2.3 Performance on Standard Data Sets

SVM, PSVM, and Linearized PSVM were tested on several data sets from the UCI Machine Learning Repository (Blake and Merz 1998). The mushroom, Wisconsin breast cancer, ionosphere, liver, Pima-Indians diabetes, tic-tac-toe, congressional voting records, SPECT

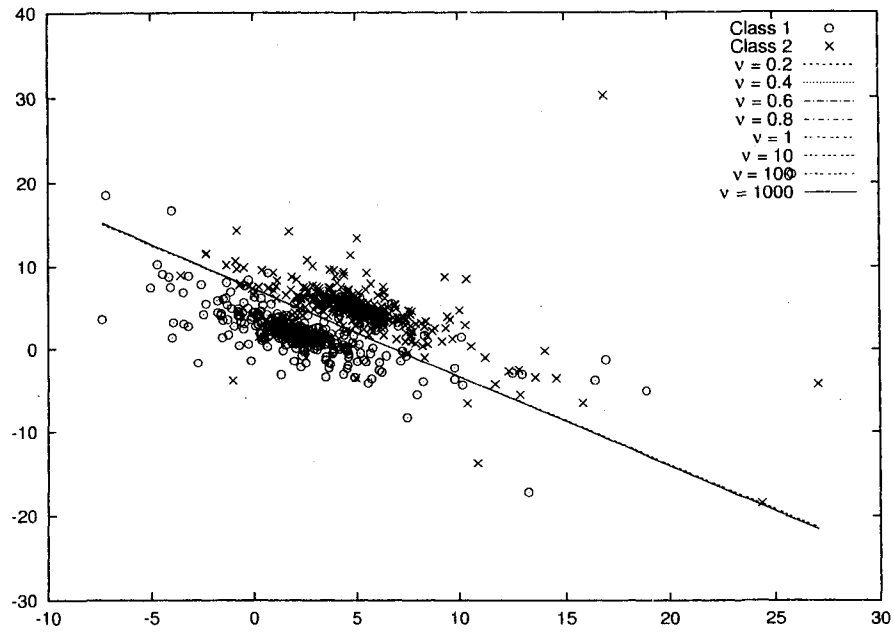

FIGURE 2.8. Varying objective function parameters for PSVM

and SPECTF, contraceptive method choice, Haberman's survival data, and the MUSK data sets were used. For each data set, the classifiers were trained on the entire set. As with the 2-D data set, each optimization problem was translated into AMPL and the submitted to the NEOS Optimization server for solving. After a solution was obtained, the performance of the classifiers on the training sets were compared as in the 2-D case above.

The task for the mushroom data set was to classify whether a mushroom was edible, "e", or poisonous, "p". The edible class was considered to be the "positive" class for performance purposes. As shown in table 2.6, SVM and PSVM were able to classify all instances of the training set correctly with an accuracy and true positive rate of 1. Linearized PSVM had a slightly lower accuracy as it was unable to classify four instances in the poisonous class. In the Linearized PSVM model, only 18 out of the 117 features had non-zero weights. PSVM had 72 non-zero weights and while SVM had 113 non-zero weights. Since Linearized PSVM chose far fewer non-zero weights than SVM or PSVM and the data is

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	1	1	0
PSVM	1	1	0
Linearized PSVM	0.99951	1	0

TABLE 2.6. Training set performance of SVM, PSVM, and Linearized PSVM on the mushroom data set.

categorical, it is more likely that a data point would have zeros where the Linearized PSVM model has non-zero weights. Thus Linearized PSVM would be more likely to be unable to classify points in this data set.

The classifiers did not perform as well on the Wisconsin breast cancer data set (wpbc) as well as they did on the mushroom data set. The "recurrent" class or class where the cancer returned was considered to be the positive class. As shown in table 2.7, PSVM had the highest accuracy. Linearized PSVM and SVM had identical performance on this data set and tend to classify everything in the "non-recurrent" class. Since all three classifiers

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.76289	0	0
PSVM	0.79381	0.19565	.02027
Linearized PSVM	0.76289	0	0

TABLE 2.7. Training set performance of SVM, PSVM, and Linearized PSVM on the Wisconsin breast cancer data set.

performed poorly, the data set itself was examined. While a scatterplot matrix with the entire feature set was unreadable, each individual plot from full scatterplot matrix was created individually and a representative sample is included in the scatterplot matrix in Figure 2.9. We can see from the scatterplot matrix that the two classes overlap significantly, and are not linearly separable.

The ionosphere data set consisted of two classes. Class 'g' are the "good" signals or those that show structure in the ionosphere while class 'b' are the "bad" signals or those that do not show structure in the ionosphere. Good signals were considered to be the

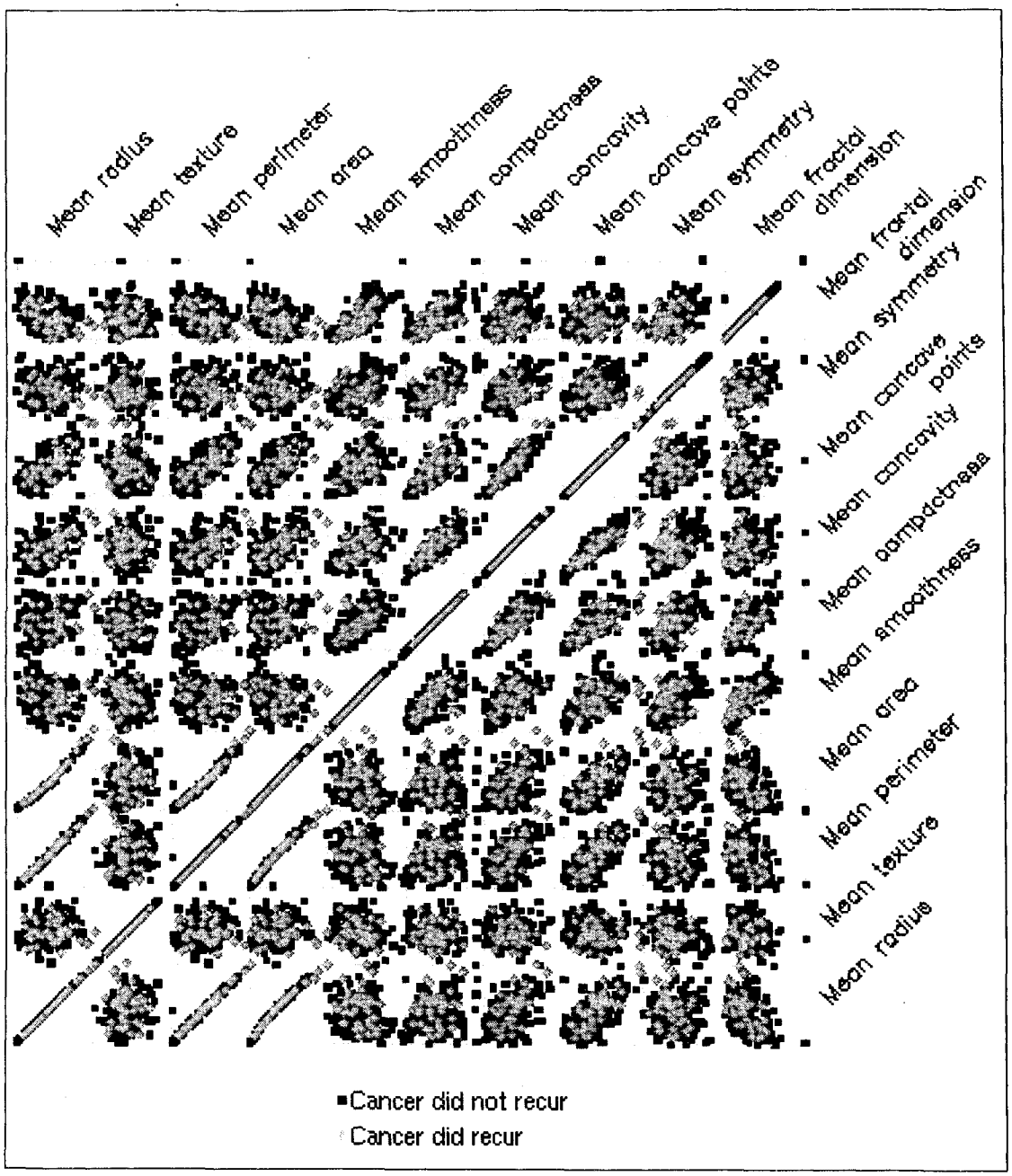

FIGURE 2.9. Partial scatterplot matrix for the Wisconsin breast cancer data set.

positive class for analysis purposes. As shown in table 2.8, SVM had the highest accuracy and lowest false positive rate. Linearized PSVM had the highest true positive rate with parameters $\mu = \nu = 0.4$ and $\beta = 0.2$.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.92308	0.97778	0.1746
PSVM	0.89174	0.98222	0.269841
Linearized PSVM	0.89458	0.99556	0.28571

TABLE 2.8. Training set performance of SVM, PSVM, and Linearized PSVM on the ionosphere data set.

The Liver Disorders data set (bupa) did not provide any information as to which class (1 or 2) was the desired class. For performance analysis, class 1 was considered to be the positive class. The training set performance can be seen in table 2.9. SVM had the highest accuracy as well as the lowest false positive rate. Linearized PSVM had the by far lowest false positive rate, but at the expense of accuracy and true positive rate. As can be seen

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.71594	0.55172	0.16500
PSVM	0.69565	0.50345	0.16500
Linearized PSVM	0.60580	0.12414	0.04500

TABLE 2.9. Training set performance of SVM, PSVM, and Linearized PSVM on the Liver Disorders data set.

from a scatterplot matrix of this data set, see Figure 2.10, the two classes appear to almost completely overlap. Thus, any linear classifier will not do well and a non-linear classifier should be used. It is unclear why Linearized PSVM does much worse than the other two classifiers, however, it may be related to the L_1 norm in the hyperplane term in the objective function.

The class label 0 from the Pima Indians Diabetes data set indicated that the individual represented tested negative for diabetes while the class label 1 indicated that the individual tested positive for diabetes. Class 0 was considered to be the positive class for analysis

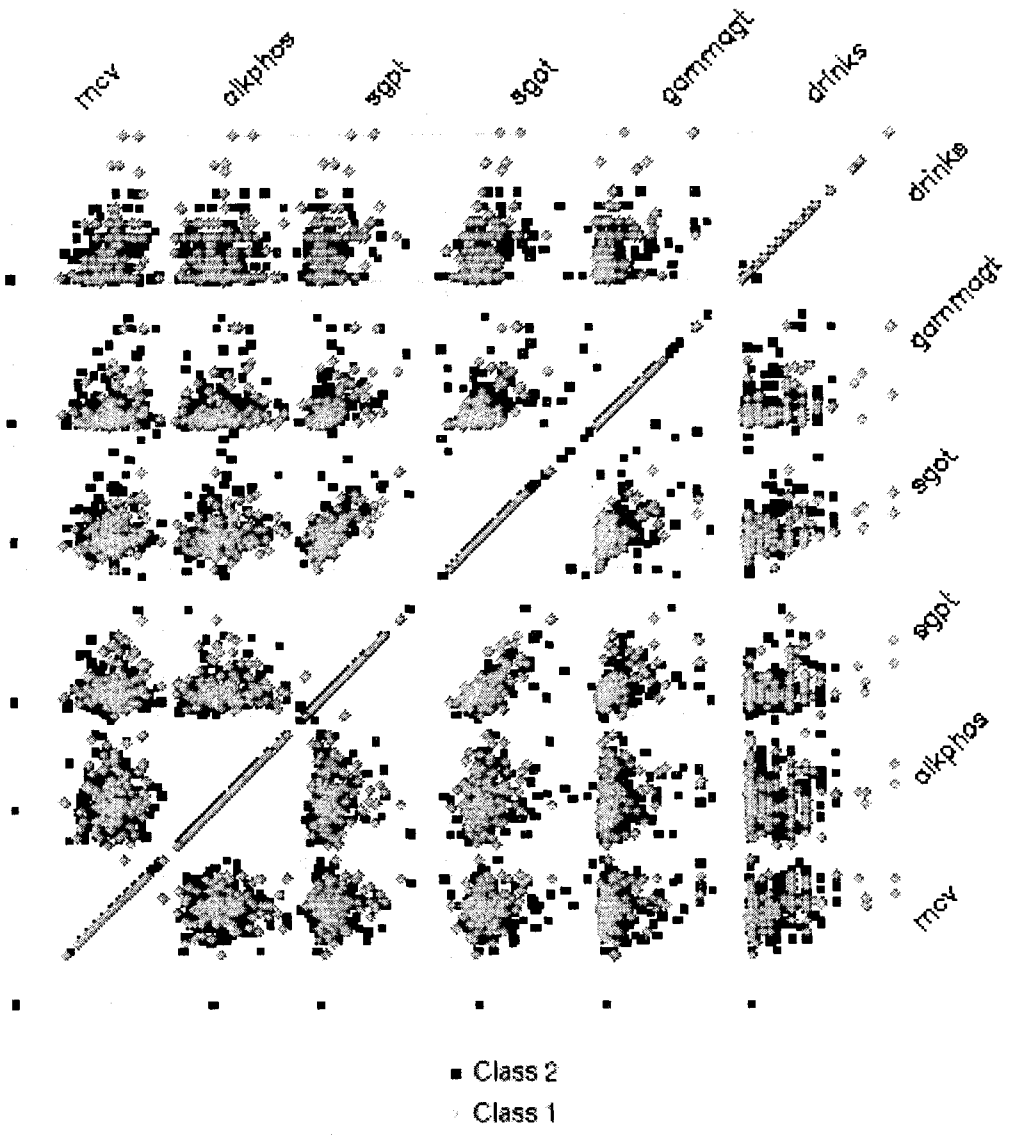

FIGURE 2.10. Scatterplot matrix for the liver disorders data set.

purposes. The training set performance for each algorithm on this data set can be seen in table 2.10. PSVM had the highest accuracy, while SVM had the lowest false positive rate. Linearized PSVM had the highest true positive rate.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.77344	0.88400	0.43284
PSVM	0.77865	0.89600	0.44030
Linearized PSVM	0.77734	0.90400	0.45896

TABLE 2.10. Training set performance of SVM, PSVM, and Linearized PSVM on the Pima Indians Diabetes data set.

The task for the tic-tac-toe data set was to predict those games where "x" would win, indicated by the "positive class". The "negative" class indicates games where "x" did not win (i.e. "o" won or the game was a tie. The performance for each algorithm can be seen in table 2.11. All three classifiers had identical performance even though they found different classifying hyperplanes.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.98330	1	0.04819
PSVM	0.98330	1	0.04819
Linearized PSVM	0.98330	1	0.04819

TABLE 2.11. Training set performance of SVM, PSVM, and Linearized PSVM on the tic-tac-toe data set.

The task for the SPECT and SPECTF data sets was to predict which cardiac SPECT images were abnormal and which were not. Class label 1 corresponded to abnormal SPECT images and was taken as the positive class for analysis purposes. These data sets were already partitioned into training and test sets and thus a standard data mining experiment could be run, i.e. the training data was used to create the model and the test data was used to evaluate performance. On the SPECT data set, all three classifiers had identical false positive rates and SVM had the highest accuracy and true positive rate. As can be seen in table 2.12, Linearized PSVM and PSVM had a similar performance. On the SPECTF data

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.73262	0.72674	0.2
PSVM	0.70053	0.69186	0.2
Linearized PSVM	0.68984	0.68023	0.2

TABLE 2.12. Test set performance of SVM, PSVM, and Linearized PSVM on the SPECT data set.

set, the performance of SVM improved over that on the SPECT data set. Both accuracy and true positive rate increased and the false positive rate decreased. As can be seen in table 2.13, the accuracy for PSVM and Linearized PSVM decreased.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.80297	0.78972	0.14545
PSVM	0.69145	0.65420	0.16364
Linearized PSVM	0.68030	0.64486	0.18182

TABLE 2.13. Test set performance of SVM, PSVM, and Linearized PSVM on the SPECTF data set.

The task for the Congressional Voting Records data set was to predict whether a representative was a democrat or a republican based on their votes on sixteen measures. Democrats were taken as the positive class for analysis purposes. As can be seen in table 2.14, SVM had the highest accuracy and the highest true positive rate. PSVM and Linearized PSVM had an identical true positive rate.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.97931	0.98127	0.023809
PSVM	0.96092	0.94756	0.017857
Linearized PSVM	0.95632	0.94756	0.029762

TABLE 2.14. Training set performance of SVM, PSVM, and Linearized PSVM on the Congressional Voting Records data set.

The task for the MUSK data set was to predict whether a molecule was a musk or not. The class labeled as "1." corresponded to the musk class and was taken to be the positive

class for analysis purposes. SVM had the best performance and was able to classify all instances correctly. As can be seen in table 2.15, Linearized PSVM outperformed PSVM.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	1	1	0
PSVM	0.88865	0.87923	0.10409
Linearized PSVM	0.90756	0.89372	0.081784

TABLE 2.15. Training set performance of SVM, PSVM, and Linearized PSVM on the MUSK data set.

The task for the Contraceptive Method Choice data set was to predict which woman would use contraception. The class of women who used contraception was taken as the positive class. As can be seen in table 2.16, SVM had the highest accuracy. PSVM had the lowest false positive rate while Linearized PSVM had the highest true positive rate. However, all the models had rather high false positive rates and thus are not very good. A scatterplot matrix was created for this data set, see Figure 2.11. Both classes overlap

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.74406	0.90047	0.46582
PSVM	0.73523	0.86967	0.44515
Linearized PSVM	0.68500	0.93839	0.65501

TABLE 2.16. Training set performance of SVM, PSVM, and Linearized PSVM on the Contraceptive Method Choice data set.

significantly thus a linear classifier will not do well on this data set. It is unclear why Linearized PSVM does much worse on this data set.

The task for the Haberman surgery survival data set was to predict which patients would survive at least five years after breast cancer surgery. The class of patients that survived was taken to be the positive class for performance analysis. As can be seen in table 2.17, PSVM had the highest accuracy and Linearized PSVM had the highest true positive rate. However, all models tended to predict that a patient survived. In addition, Linearized PSVM was not able to find a reasonable solution with its default parameters and needed the parameter for

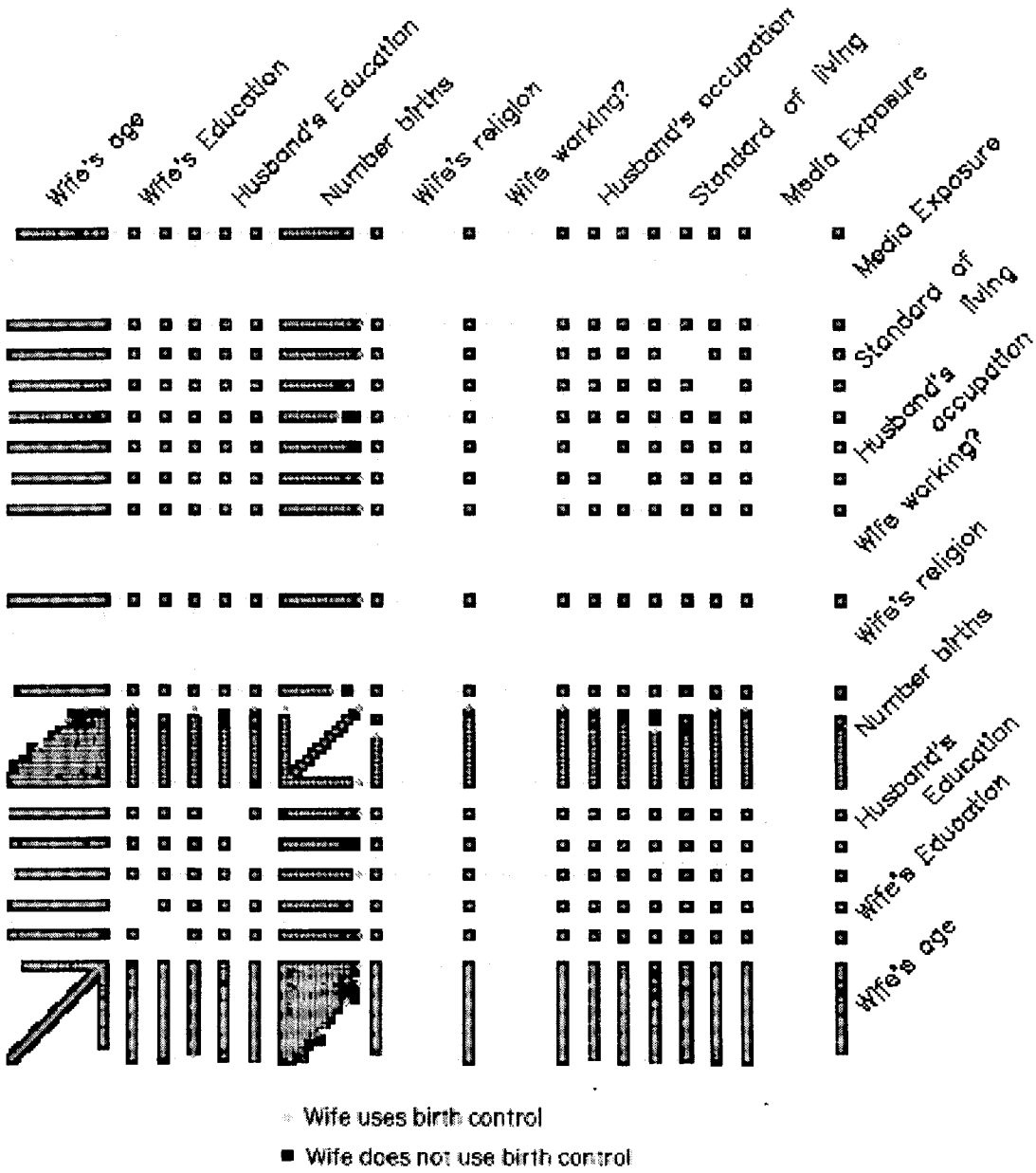

FIGURE 2.11. Scatterplot matrix for the contraceptive method choice data set.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.73203	0.98667	0.97531
PSVM	0.74183	0.96	0.86420
Linearized PSVM	0.73529	1	1

TABLE 2.17. Training set performance of SVM, PSVM, and Linearized PSVM on the Haberman surgery survival data set.

the hyperplane term in the objective function increased relative to the other two terms in order to find a solution. Since this data set had few features, it could be examined visually. A scatterplot matrix was created for this data set, see Figure 2.12. As can be seen from the scatterplot matrix, both classes appear to completely overlap. Thus, a linear classifier is not an appropriate model for this data set.

For all these data sets, no single classifier dominated the others. On some data sets, all classifiers had significant difficulty. These data sets do not appear to be linearly separable. The different results from the classifiers coupled with the additional information provided by the true positive rate and false positive rate illustrated this more clearly than the accuracy from a single classifier. For the contraceptive method choice data set, the accuracy for PSVM and SVM was reasonable. The high true positive rate around 0.9 for both classifiers is quite good, but the false positive rate around 0.45 is a little suspicious. However, when the results from Linearized PSVM are included, the high false positive rate looks more suspicious and the data itself seems to warrant further investigation. The contraceptive method choice data set had nine features and could be visualized using a scatterplot matrix. Most applications will have more features and will not be easily visualized. The inclusion of additional hyperplane classifiers increases the likelihood that data sets that are not linearly separable will be identified.

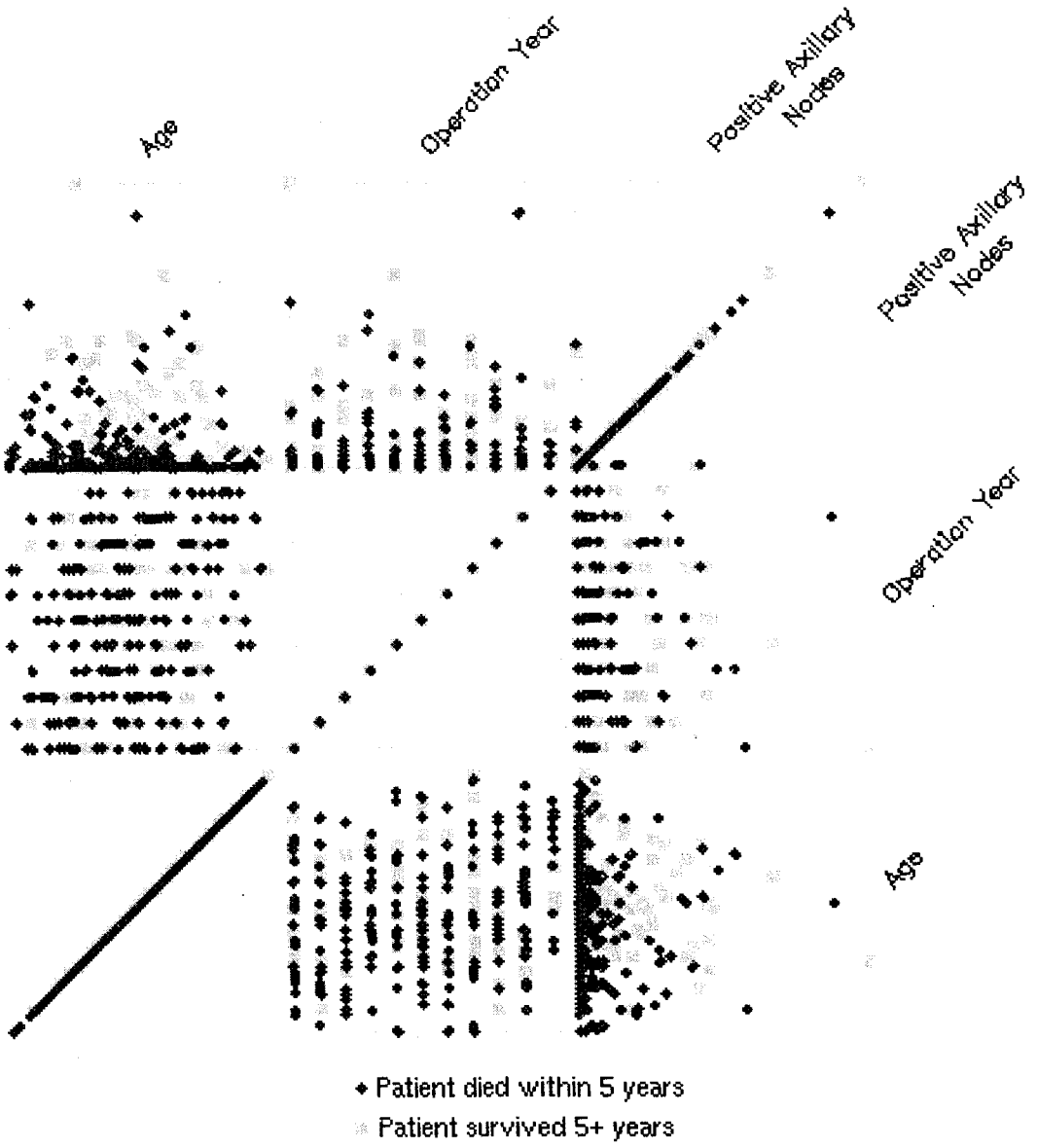

FIGURE 2.12. A scatterplot matrix for the Haberman survival data.

2.4 Conclusion

We have introduced a linearized version of the Proximal Support Vector Machine classifier, Linearized PSVM. This linear reformulation does not weight outlying points as heavily as the original quadratic PSVM. While the Karush Kuhn Tucker conditions can no longer be used to obtain the solution, the solution time has not been drastically increased.

We have also introduced new weighting parameters to control the model of the separating hyperplane. A simple example of weighting the error terms differently than the hyperplane term was examined for a simple 2-D data set. While the weighting parameters did not have much of an effect for the results of the 2-D case, accuracy was increased for several of the data sets by increasing or decreasing the weights of the error terms relative to the hyperplane term.

To compare the performance of Linearized PSVM to SVM and PSVM, models for several common data sets were created and the common performance measures of accuracy, true positive rate and false positive rate were examined. While no single classifier dominated the others, the results were similar enough to provide insight into the nature of the data. Data sets that are not linearly separable are far more common than linearly separable data sets. In those cases where the data is not linearly separable, all three classifiers perform differently enough so that these data sets are more easily identified without the aid of visualization.

CHAPTER 3

INTEGER SUPPORT VECTOR MACHINE

3.1 Discussion

The objective function for the Support Vector Machine classifier has two terms, an error term and margin or hyperplane term. The error term is a function of the distances from the bounding planes (in the case of SVM) or the proximal planes (in the case of PSVM and Linearized PSVM). When the performance of classifiers is evaluated, the distance from the classifying hyperplane is not important, only the relative location of the point to this hyperplane. To model this more accurately, new error variables can be introduced that indicate where the points lie relative to the bounding planes. The error term in the objective function is reformulated in terms of these error indicator variables and become a count of the potential errors the classifier can make. In this formulation, SVM becomes a mixed integer linear program.

We will use the following notation in the Integer SVM (ISVM) formulation. The set F is the set of features. We define M as the set of points belonging to class 1 and N as the set of points belonging to class 2. The data for class 1 is contained in matrix x and the data for class 2 is contained in matrix y. $w_k, k \in F$ and γ are the coefficients for the separating or classifying hyperplane. In addition, we need several auxiliary variables. Δ_i^1 measures the distance of the i^{th} point in class 1 from the proximal plane $w^T x = \gamma + 1$. Similarly, Δ_j^2 measures the distance of the j^{th} point in class 2 from the proximal plane $w^T y = \gamma - 1$.

Similar to SVM, we wish to find a separating hyperplane between two classes of data, class 1 and class 2. This hyperplane is midway between two bounding hyperplanes, one for each class.

The objective function in SVM consists of two terms. An error term and a margin term. This margin term has the effect of pushing the bounding hyperplanes apart. The error term

is defined by the distance from a bounding hyperplane. For ISVM, we will introduce error indicator variables and replace the distance-based error term by one that minimizes the number of errors.

Using the notation defined above, we note that distances from the bounding hyperplanes Δ_i^1 and Δ_j^2 are positive and negative in the regions indicated in Figure 3.1. Since the

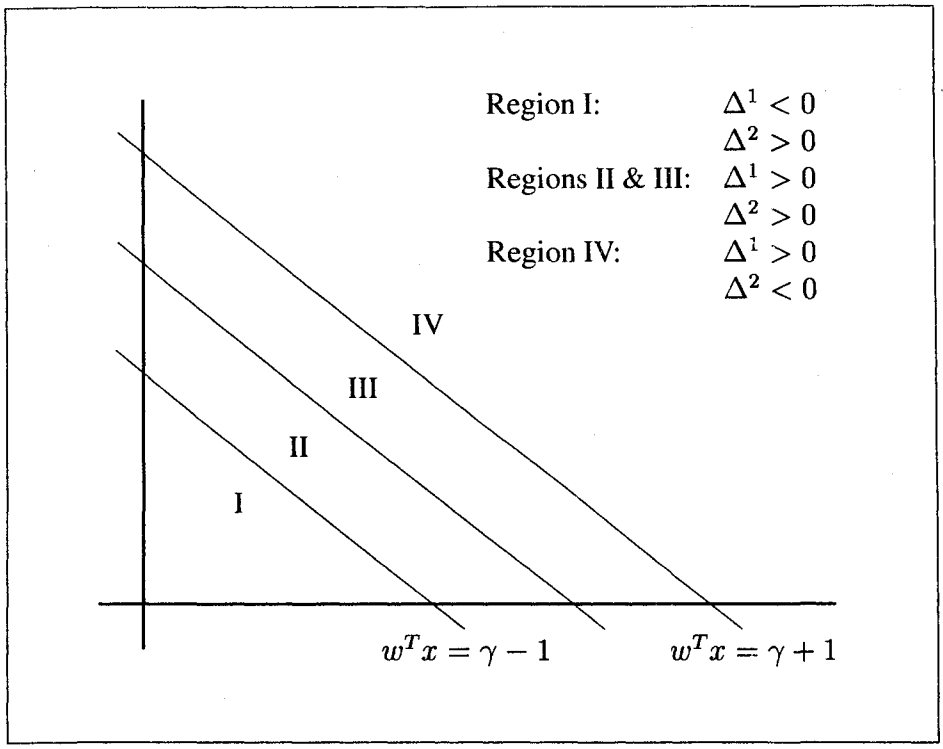

FIGURE 3.1. Hyperplanes with sign of Delta indicated.

classifier is more likely to make errors if Δ_i^1 and Δ_j^2 are positive, we define error indicator variables as follows

$$\delta_i = \begin{cases} 1, & \text{if } \Delta_i^1 > 0 \\ 0, & \text{otherwise} \end{cases}, i \in M,$$

$$\varepsilon_j = \begin{cases} 1, & \text{if } \Delta_j^2 > 0 \\ 0, & \text{otherwise} \end{cases}, j \in N.$$

These are represented by the following standard if-then constraints

$$L\delta_i \geq \Delta_i^1$$

and

$$L\varepsilon_j \geq \Delta_j^2$$

where L is a large constant.

Following Chapter 2, we have chosen the L_1 norm for the hyperplane coefficients to simplify the formulation. Again, we split the free variables into their positive parts and negative parts, i.e.

$$w_k = w_k^+ - w_k^-.$$

In addition, each term in the objective function receives its own parameter (μ, ν, and β respectively). The formulation is thus a mixed-integer linear program:

$$\begin{aligned}
\min \quad & \mu \sum_{i \in M} \delta_i + \nu \sum_{j \in N} \varepsilon_j + \beta \left(\sum_{k \in F} \left(w_k^+ + w_k^- \right) + \gamma^+ + \gamma^- \right) & \\
s.t. \quad & \sum_{k \in F} x_{ik} \left(w_k^+ - w_k^- \right) - \gamma^+ + \gamma^- + \Delta_i^1 = 1, & i \in M \\
& -\sum_{k \in F} y_{jk} \left(w_k^+ - w_k^- \right) + \gamma^+ - \gamma^- + \Delta_j^2 = 1, & j \in N \\
& L\delta_i \geq \Delta_i^1, & i \in M \\
& L\varepsilon_j \geq \Delta_j^2, & j \in N \\
& w_k^+, w_k^- \geq 0, & k \in F \\
& \delta_i \in \{0,1\}, & i \in M \\
& \varepsilon_j \in \{0,1\}, & j \in N \\
& \gamma^+, \gamma^- \geq 0 &
\end{aligned}$$

The scaling problem noted for Linearized PSVM can potentially be a larger problem for ISVM with its combination of integer and continuous objective function terms. Again, the objective function parameters can be used to compensate for this and parameter selection is discussed further for a simple 2-D example in Section 3.2.2. Parameter setting can be

easier conceptually for ISVM since the meaning of the error indicator variables is clearer. By reviewing the values of these variables, it is relatively easy to determine how to set the parameters. If there are many error indicator variables set to 1, the error costs can be increased relative to the hyperplane cost. A similar heuristic can be used for the case where many error indicator variables for a particular class are set to 1.

3.2 Visualization Examples

Most data mining applications involve high-dimensional data sets, which are difficult to visualize. To compare ISVM to SVM and PSVM, a non-separable two-dimensional data set was created consisting of 250 points per class. Each class in this data set was generated from a bivariate normal distribution with the same parameters as were used in Chapter 2.

Following the procedure in Chapter 2, ISVM was translated into AMPL (Fourer, Gay, and Kernighan 2002). The models were solved using the NEOS Optimization server (Czyzyk, Mesnier, and Moré 1998) (Gropp and Moré 1997) (Dolan 2001). ISVM was submitted via email to the XPRESS-MP solver (version 12.13) (XPRESS 1999).

The hyperplanes found by training the classifiers on the entire data set is shown in table 3.1. A plot of the data set and the classifying hyperplane and bounding or proximal

Algorithm	Hyperplane
SVM	$y = 6.9254 - 1.0206x$
PSVM	$y = 7.3354 - 1.0696x$
ISVM	$y = 6.7869 - 1.0876x$

TABLE 3.1. Hyperplanes found by SVM, PSVM, and ISVM for a non-separable 2-D data set.

hyperplanes as well as a plot of the points in the margin for SVM can be found in Figures 3.2 and 3.3. Figures 3.4 - 3.7 show the same plots for PSVM and ISVM respectively. It is interesting to note that even though ISVM has equality constraints for the hyperplanes like PSVM, it finds bounding hyperplanes like SVM instead of PSVM's proximal hyperplanes.

This is a result of the formulation of the error indicator variables. In SVM, the errors are only positive for those points that are inside the margin or beyond the hyperplane for the opposite class. In ISVM, the error indicator variables are defined similarly, resulting in similar behavior.

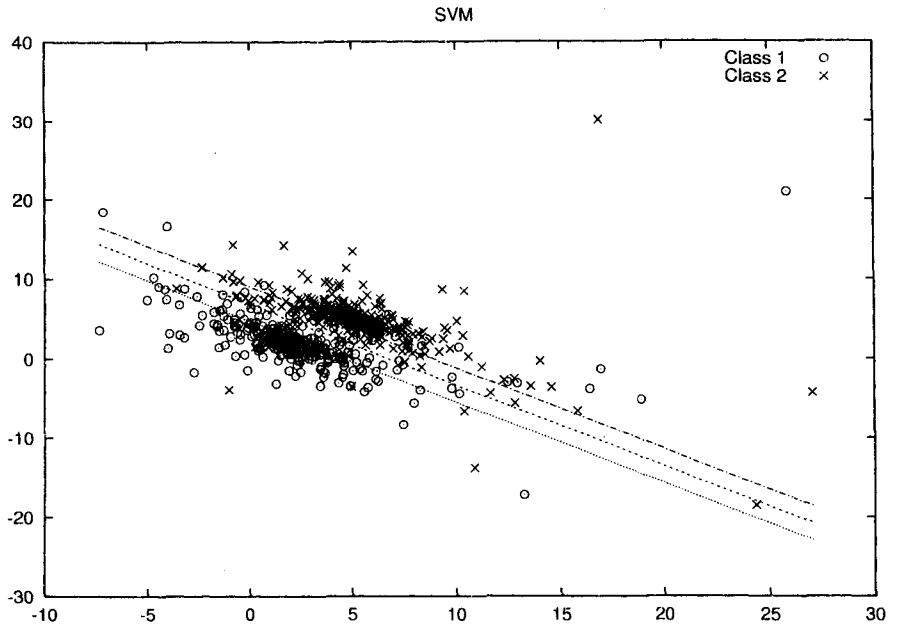

FIGURE 3.2. Classifying hyperplane and bounding hyperplanes for SVM

Classifier performance was compared by examining accuracy, true positive rate, and false positive rate (see Chapter 1) with class 1 taken to be the positive class. The results for each classifier on this 2-D data set are show in table 3.2. ISVM had the highest accuracy

Model	Accuracy	True Positive Rate	False Positive Rate
SVM	0.952	0.94	0.036
PSVM	0.95	0.944	0.044
ISVM	0.956	0.932	0.02

TABLE 3.2. Performance of SVM, PSVM, and ISVM on the 2-D data set.

and the lowest false positive rate while PSVM had the highest true positive rate.

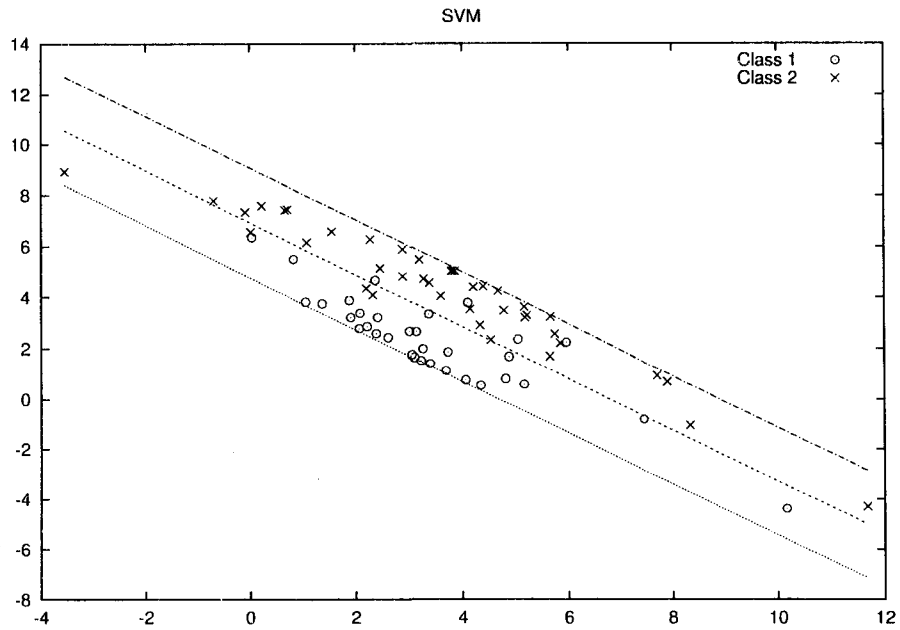

FIGURE 3.3. Expanded view of the data lying in the margin for SVM

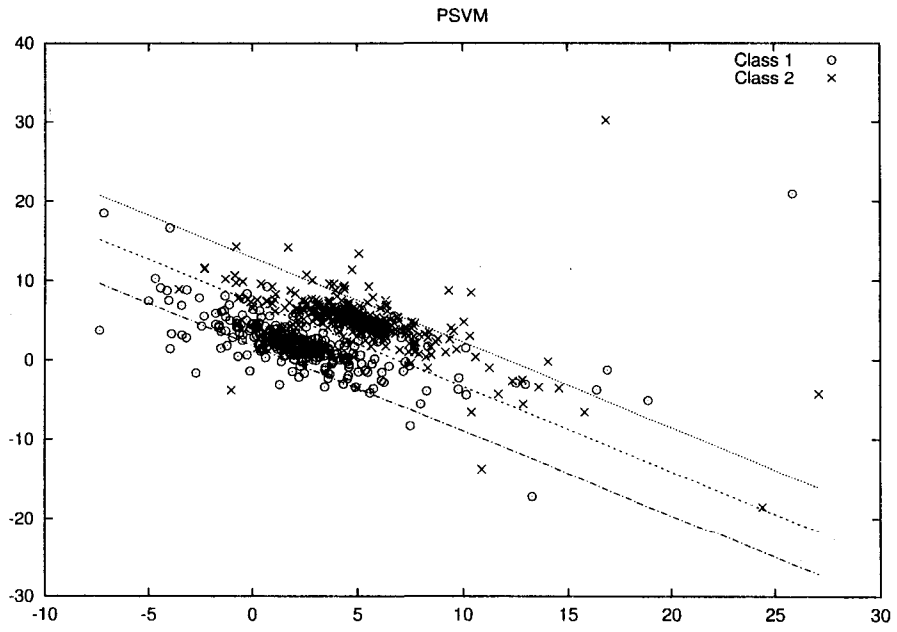

FIGURE 3.4. Classifying hyperplane and bounding hyperplanes for PSVM

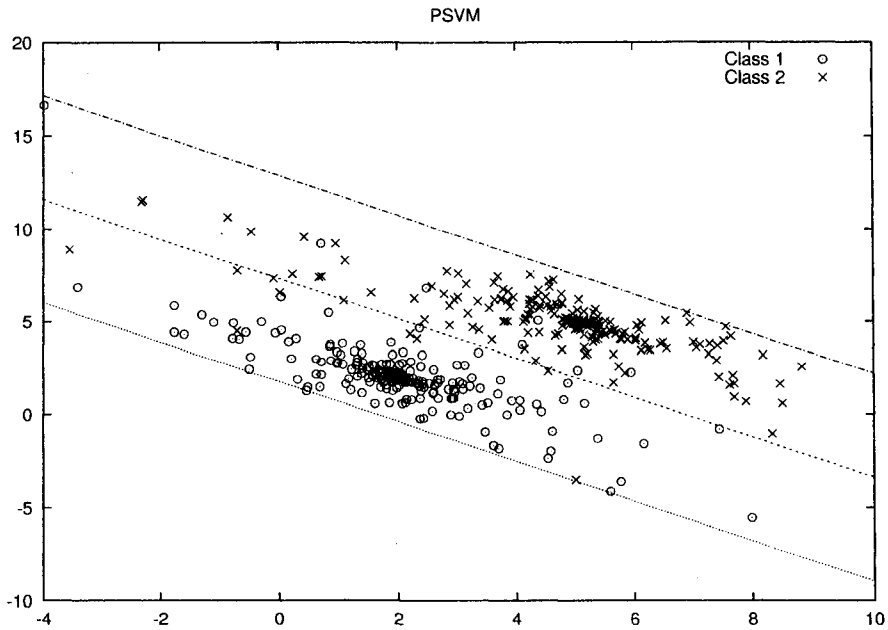

FIGURE 3.5. Expanded view of the data lying in the margin for PSVM

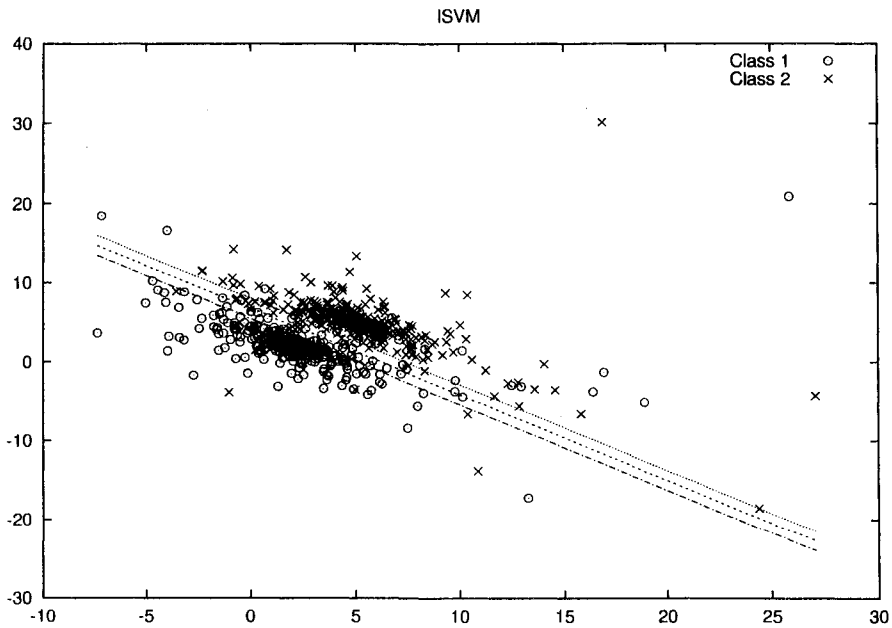

FIGURE 3.6. Classifying hyperplane and bounding hyperplanes for ISVM

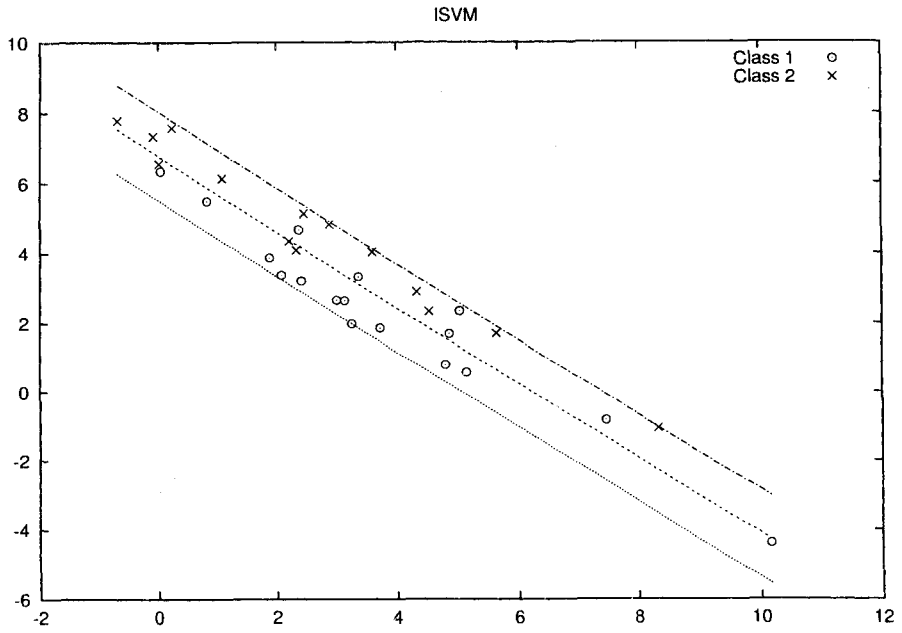

FIGURE 3.7. Expanded view of the data lying in the margin for ISVM

3.2.1 Solution Time

ISVM, being a mixed-integer linear program, will most likely take longer to solve than Linearized PSVM, PSVM and SVM. Most mixed-integer solvers use tree-based search (i.e. branch and bound) techniques to find the optimal solution. This can be slower than the simplex method used to solve linear programs and the reduced gradient algorithm used by the MINOS solver (Murtagh and Saunders 1998) to solve non-linear programs. Unfortunately, a comparison of solution times among the three algorithms is not possible due to the use of the NEOS solver. In order to perform a meaningful comparison of solution times, each problem should be solved on the same machine. As described in Chapter 2, the user does not have control over which machine the problem is sent to. Since different problems cannot be run on the same machine, a comparison of solution times is not meaningful. In practice however, the NEOS server returned results for ISVM slower than for Linearized PSVM, PSVM, or SVM, where results were returned almost immediately.

3.2.2 Parameter Selection

The objective function parameters for ISVM (μ, ν, β) perform the same function as those in Linearized PSVM. Again, these parameters may be "tuned" to better reflect expert knowledge about the data. The error parameters, μ and ν, may be thought of as misclassification costs. If it is more important to be correct on one class or another, these costs can be set to reflect that. Another approach could be to set the costs to reflect confidence in the data set. In the case where data is merged from different sources, some sources may not be as reliable as other sources. When this is known a priori, the misclassification costs can be set to reflect confidence in the data.

In the course of performing the experiments on the standard data sets (see Section 3.3), it was discovered that occasionally ISVM would have trouble finding a reasonable solution. All components of w would be set to zero, γ would be set to 0 or 1, and either δ or ε equal to 2. These types of models do not perform well. It is believed that the cause is a mismatch in scale between the integer and continuous parts of the objective function. It was discovered that if the misclassification costs were increased relative to β, ISVM could be forced off this pathological solution to find a better classifier.

The parameter setting experiment carried out on Linearized PSVM was also completed for ISVM. However, in this case, half of the data points of the original non-separable data set were excluded due to memory and disk space restrictions on the XPRESS solver provided by NEOS. Again, the error terms for each class were weighted equally. The values for μ and ν ranged from 0 to 0.5 and β was set so the sum of all three costs was 1. The effect of varying the parameters on the classifying hyperplanes can be seen in Figure 3.8. The effect on performance is shown in table 3.3. The experiment was also carried out for Linearized PSVM on the reduced data set. The effect of varying the parameters on the classifying hyperplane is shown in Figure . The effect on performance can be seen in table 3.4.

For the reduced data set, ISVM appears to be more sensitive than Linearized PSVM to

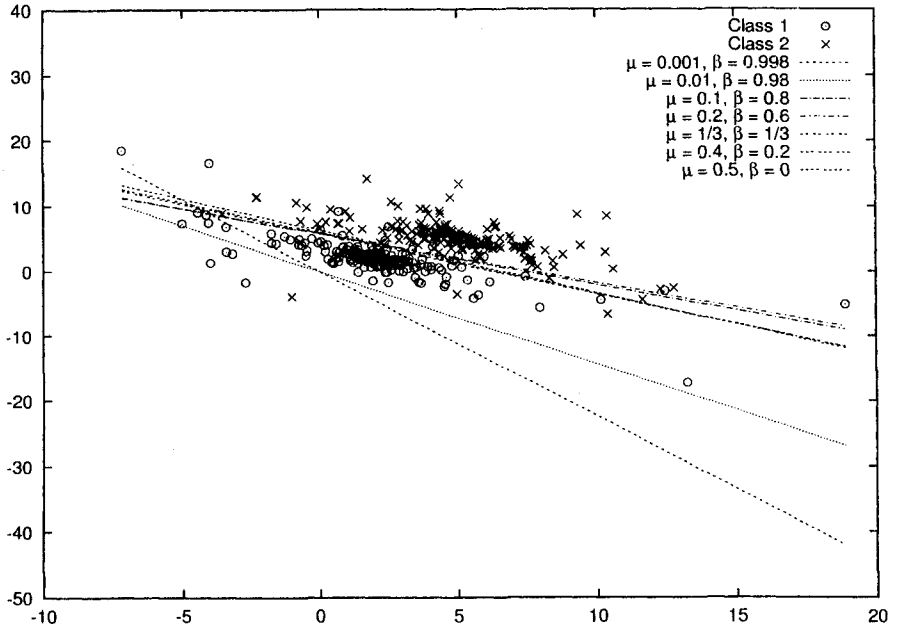

FIGURE 3.8. The Effect of Changing Objective Function Parameters on the Classifying Hyperplanes for ISVM

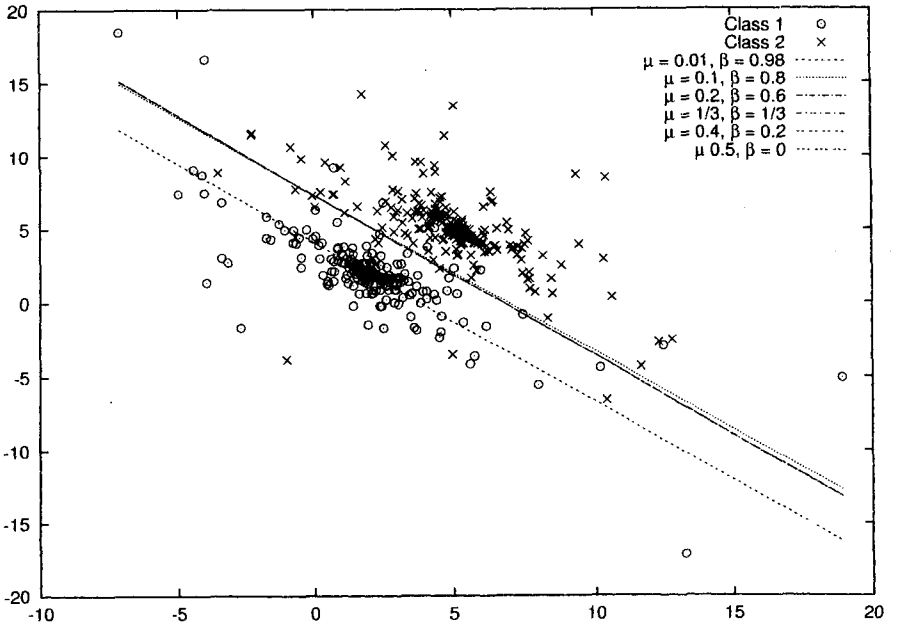

FIGURE 3.9. The Effect of Changing Objective Function Parameters on the Classifying Hyperplanes for Linearized PSVM

μ	β	Accuracy	True Positive Rate	False Positive Rate
0	1	0	0	0
0.001	0.998	0.516	0.036	0.004
0.01	0.98	0.506	0.016	0.004
0.1	0.8	0.958	0.94	0.024
0.2	0.6	0.954	0.94	0.032
$0.\bar{3}$	$0.\bar{3}$	0.956	0.932	0.02
0.4	0.2	0.956	0.932	0.02
0.5	0	0.962	0.948	0.024

TABLE 3.3. Performance results from sensitivity analysis of ISVM to changes in objective function parameters.

μ	β	Accuracy	True Positive Rate	False Positive Rate
0	1	0	0	0
0.001	0.998	0	0	0
0.01	0.98	0.708	.428	.012
0.1	0.8	0.954	0.952	0.044
0.2	0.6	0.954	0.956	0.048
$0.\bar{3}$	$0.\bar{3}$	0.954	0.956	0.048
0.4	0.2	0.954	0.956	0.048
0.5	0	0.954	0.956	0.048

TABLE 3.4. Performance results from sensitivity analysis of Linearized PSVM to changes in objective function parameters.

changes in objective function parameters. The objective function terms in SVM have been described as supplying opposing forces on the bounding/separating hyperplanes. However, this is not completely correct. The error term actually supplies both forces, otherwise setting the coefficient of the hyperplane term to zero would result in the zero solution just like setting the error coefficient to zero does. The increased sensitivity in ISVM arises from the integer error terms. In order to obtain a change in performance, the hyperplane must shift so that at least one point ends up on the other side of the classifying hyperplane. If the hyperplane shifts but no points end up on the other side of the classifying hyperplane, then from a performance viewpoint we have multiple equivalent solutions. Suppose we have one point very near the classifying hyperplane and we cause a shift in the classifying

hyperplane so that this point ends up just on the other side of the classifying hyperplane. For Linearized PSVM, the distance from this point to its proximal hyperplane does not change much. Thus the objective function value does not change much and the objective function coefficients have less of an effect. For ISVM, the error indicator variable for this point will change from 0 to 1 or from 1 to 0. This is a much larger change in the objective function value and the objective function parameters have a larger effect.

3.3 Performance on Standard Data Sets

SVM, PSVM, and ISVM were tested on several data sets from the UCI Machine Learning Repository (Blake and Merz 1998). The same data sets used in Chapter 2 are also used here. For each data set, the classifier was trained on the entire set and performance was evaluated on the training set unless otherwise noted.

The mushroom data set consists of two classes, "edible" and "poisonous". For analysis purposes, the "edible" class was taken as the positive class. As shown in table 3.5, SVM, PSVM, and ISVM were each able to classify all instances of the training set correctly.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	1	1	0
PSVM	1	1	0
ISVM	1	1	0

TABLE 3.5. Training set performance of SVM, PSVM, and ISVM on the mushroom data set.

The Wisconsin breast cancer data set (wpbc) was more difficult for the classifiers. For analysis purposes, the "recurrent" class was considered to be the positive class. All classifiers tended to place the instances in the "non-recurrent" class. However, ISVM performs better on this difficult data set. Since ISVM minimizes an error count and not an error distance, it may do better with difficult data sets. The performance results for all three classifiers can be seen in table 3.6.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.76289	0	0
PSVM	0.79381	0.19565	0.02027
ISVM	0.79897	0.36956	0.06757

TABLE 3.6. Training set performance of SVM, PSVM, and ISVM on the Wisconsin breast cancer data set.

The task of the ionosphere data set is to predict whether a particular radar return indicates structure in the ionosphere. The positive class consisted of those signals that show structure. As can be seen in table 3.7, SVM had the highest accuracy while ISVM had the lowest false negative rate. The PSVM classifier had the highest true positive rate.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.92308	0.97778	0.17460
PSVM	0.89174	0.98222	0.26984
ISVM	0.91168	0.94667	0.15079

TABLE 3.7. Training set performance of SVM, PSVM, and ISVM on the ionosphere data set.

The Liver Disorders data set (bupa) was another difficult data set for the classifiers as the classes overlap significantly, see Figure 2.10. As seen in table 3.8, ISVM had the highest accuracy and the lowest false positive rate. SVM had the highest true positive rate.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.71594	0.55172	0.16500
PSVM	0.69565	0.50345	0.16500
ISVM	0.72174	0.54483	0.15000

TABLE 3.8. Training set performance of SVM, PSVM, and ISVM on the Liver Disorders data set.

The task for the Pima Indians diabetes data set is to predict whether a woman has diabetes or not. As shown in table 3.9, PSVM had the highest accuracy as well as the

highest true positive rate. ISVM had its best performance using parameters $\mu = \nu = 0.4$ and $\beta = 0.2$. This data set was particularly difficult for ISVM, however, none of the

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.77344	0.88400	0.43284
PSVM	0.77865	0.89600	0.44030
ISVM	0.65234	0.974	0.94776

TABLE 3.9. Training set performance of SVM, PSVM, and ISVM on the Pima Indians diabetes data set.

classifiers performed particularly well. The data set was then visualized using a scatterplot matrix, see Figure 3.10. It appears to be not linearly separable.

The tic-tac-toe data set consists of all possible configurations of the tic-tac-toe board at the end of a game. The "positive" class contains games where player "x" won and the negative class contains games where "o" won or there was a tie. Table 3.10 shows the performance of the classifiers on this data set. All three classifiers were able to classify all

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.98330	1	0.04819
PSVM	0.98330	1	0.04819
ISVM	0.98747	1	0.03614

TABLE 3.10. Training set performance of SVM, PSVM, and ISVM on the tic-tac-toe data set.

of the "positive" class instances correctly. ISVM was able to classify more of the "negative" class correctly with parameters $\mu = \nu = 0.5$ and $\beta = 0$. Using default parameters, ISVM does not perform nearly as well. For categorical data sets, the data is transformed so all points lie on the unit hypercube. If one class happens to occupy one of these corners, then the data set is linearly separable (any separating hyperplane merely needs to slice off this corner). The tic-tac-toe data set is likely almost linearly separable, i.e. one or a few corners are inhabited mostly by the "positive" class with just a few of the "negative" class points interspersed.

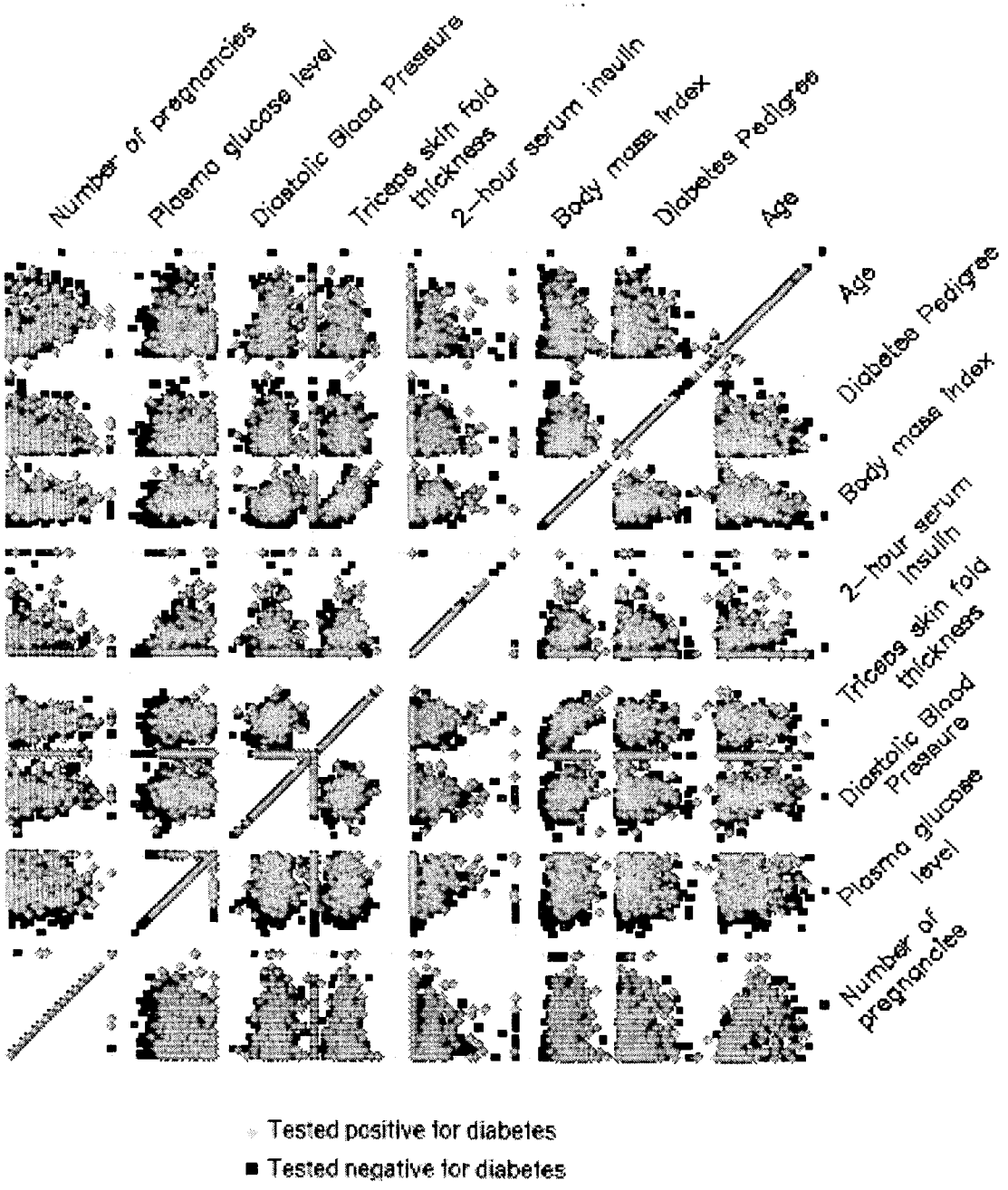

FIGURE 3.10. Scatterplot matrix for the Pima Indians diabetes data set.

The SPECT and SPECTF data sets consisted of features constructed from cardiac SPECT images. For analysis purposes, the class of abnormal images was taken to be the positive class. In addition, the SPECT and SPECTF data sets came prepartitioned into training and test sets. Thus performance was evaluated on the test data. For the SPECT data set, ISVM had the highest accuracy as well as the highest true positive rate. Unfortunately it also had the highest false positive rate, as can be seen in table 3.11. On the

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.73262	0.72674	0.2
PSVM	0.70053	0.69186	0.2
ISVM	0.75936	0.76744	0.33333

TABLE 3.11. Test set performance of SVM, PSVM, and ISVM on the SPECT data set.

SPECTF data set, whose features were continuous, SVM increased its performance across the board. As can be seen in table 3.12, ISVM matched SVM's false positive rate, but decreased its true positive rate.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.80297	0.78972	0.14545
PSVM	0.69145	0.65420	0.16364
ISVM	0.77695	0.75701	0.14545

TABLE 3.12. Test set performance of SVM, PSVM, and ISVM on the SPECTF data set.

In predicting the difference between Democrats and Republicans based on their votes, SVM and PSVM did quite well. The class of Democratic representatives were taken to be the positive class. As can be seen in table 3.13, SVM had the highest accuracy and highest true positive rate. PSVM and ISVM had identical performance on all performance measures.

The MUSK data set contained descriptions of molecules that were either musks or non-musks. The musk class was taken to be the positive class for analysis purposes. As can be

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.97931	0.98127	0.023809
PSVM	0.96092	0.94756	0.017857
ISVM	0.96092	0.94756	0.017857

TABLE 3.13. Training set performance of SVM, PSVM, and ISVM on the Congressional Voting Records data set.

seen in table 3.14, both ISVM and SVM performed very well on this data set and classified everything correctly.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	1	1	0
PSVM	0.88865	0.87923	0.10409
ISVM	1	1	0

TABLE 3.14. Training set performance of SVM, PSVM, and ISVM on the MUSK data set.

The Contraceptive Method Choice data set contained socio-economic information on married women. The original data set contained three classes; those who used no form of birth control, those who used a long-term method of birth control and those who used a short-term method of birth control. To study the performance of binary classifiers on this data set, the last two classes were re-labeled as those who use birth control. As can be seen in table 3.15, SVM had the highest accuracy and highest true positive rate while PSVM had the lowest false positive rate. ISVM had its best performance with parameters $\mu = \nu = 0.5$

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.74406	0.90047	0.46582
PSVM	0.73523	0.86967	0.44515
ISVM	0.69178	0.86730	0.54372

TABLE 3.15. Training set performance of SVM, PSVM, and ISVM on the Contraceptive method Choice data set.

and $\beta = 0$, but still did not perform very well compared to either PSVM or SVM. As shown

in Figure 2.11, this data set is not linearly separable and a non-linear model would be more appropriate.

The Haberman surgery survival was another difficult data set for the classifiers. The class of patients who survived was taken to be the positive class for analysis purposes. As can be seen in table 3.16, ISVM had the highest accuracy and the lowest false positive rate, but at the expense of the true positive rate. In addition, it must be noted that the mixed-integer solver was unable complete its search due to memory constraints. While the XPRESS solver returned a solution, there is no guarantee that it is a global optimum and a better solution may exist. However, as shown by Figure 2.12, this data set is highly non-linearly separable and a non-linear model would be more appropriate.

Algorithm	Accuracy	True Positive Rate	False Positive Rate
SVM	0.73203	0.98667	0.97531
PSVM	0.74183	0.96	0.86420
ISVM	0.77778	0.93778	0.66667

TABLE 3.16. Training set performance of SVM, PSVM, and ISVM on the Haberman surgery survival data set.

Out of the 11 data sets studied, ISVM outperformed or tied SVM and PSVM on 7 data sets. However, on some data sets, all classifiers had difficulty. When these data sets were visualized, the classes appear to overlap significantly. As with Linearized PSVM, examining results from ISVM also allows these non-separable data sets to be identified more easily. In addition, if a linear classifier must be used, ISVM can find better classifiers for these difficult data sets.

3.4 Conclusion

ISVM, a mixed-integer formulation of SVM, was introduced. This formulation replaces the distance based error measurement of SVM, PSVM and Linearized PSVM with error indicator variables. This change more closely models the classification task as the actual

distances from the classifying hyperplane is not important, only the relative location. In this formulation, the number of potential errors are now directly minimized. However, the resulting formulation is a mixed-integer linear program and solutions cannot be found as quickly.

New tuning parameters were incorporated to control the model of the separating hyperplane. These tuning parameters were more important for ISVM as changing the parameters allowed a non-pathological solution to be found.

The performance of ISVM on several common data sets was compared to SVM and PSVM using the performance measures of accuracy, true positive rate and false positive rate. As with Linearized PSVM in Chapter 2, no single classifier dominated, however, ISVM performed better on some non-separable data sets. The differing results provide insight into the nature of the data. Viewing the results from all these classifiers allows data that is not linearly separable to be identified without visualization.

CHAPTER 4

COLLUSION DETECTION IN IRC POKER

4.1 Introduction

4.1.1 Poker

Poker has many variants, however they all have a few things in common. The basic idea in poker is that the players bet into a community pot during a hand. At the end of the hand, the player holding the best hand (or set of cards) wins the money in the pot. In a hand, cards are dealt and the players place bets in a sequence until all bets are called. This process is called a round of betting or simply, a round. While the number of betting rounds in a hand varies, the actions available to a player are as follows:

fold This action forfeits any claim to the pot. The player's cards are returned to the dealer and all previous bets are lost.

check A check is a bet of zero. Unlike the fold, the claim to the pot is not relinquished.

bet or raise This bet is larger than all previous bets (and greater than zero) in this round. All players that follow must match this bet or exceed it in order to retain a claim to the pot. When the size of the previous bets is larger than zero, this action is called a raise. If the size of the previous bets is zero, this action is called a bet.

call This action is a bet larger than zero and equal to a preceding bet. This action maintains a player's claim to the pot.

In many poker games, the betting proceeds clockwise from the first player to the left of the dealer until each player has either called or folded.

In the first betting round, some or all of the players are required to put money in the pot. If all players place money into the pot, this is called an ante. In other poker variants,

a couple of players put initial bets into the pot, called blind bets. Both blind bets and antes serve to get nonzero bets started and prevent all players from checking.

The number of betting rounds varies among the different varieties of poker and can range from one betting round to five or more. In those variants with multiple betting rounds, additional cards enter into the game. In some variants, the players discard cards from their hand and receive replacement cards from the deck. In others, community cards are dealt or flipped over in a certain pattern. Some variants even use dice (for example, Pai Gow). See http://www.pokermike.com/poker for some of the variants.

In all variants, after all players have called or folded, the players show their cards and the winner is determined. This is called a *show down.* The hand ranking used is the standard poker hand ranking system. If two players have the same hand, the player with the highest ranked cards wins the pot[1]. If two players have hands with the same rank cards, they typically split the pot. The hand ranks are shown below from highest to lowest:

Royal Flush A royal flush consists of an ace, king, queen, jack, and ten all of the same suit.

Straight Flush A straight flush is any five cards of the same suit that are in sequence. For example 10♠ 9♠ 8♠ 7♠ 6♠, and Q♡ J♡ 10♡ 9♡ 8♡ are both straight flushes, however, the second straight flush is the better hand since its top card, Q♡, is higher than the top card of the first flush, 10♠.

Four of a Kind A hand containing four of a kind has four cards all of the same rank and one card of another rank. For example,8♣ 8◇ 8♠ 8♡ 2♠ and 6◇ 6♠ 6♡ 6♣ K♡ are both four of a kind. However, the first hand is better than the second hand.

Full House A full house consists of three cards of one rank and two cards of another. For example, 10♡ 10♠ 10◇ 2♡ 2◇ and 9♣ 9◇ 9♠ A♡ A♠ are both full houses. The rank of the triple determines which full house is better, so the full house with 10's

[1] However, there are variants where the lowest ranked cards win the pot.

and 2's ('tens on twos' or 'tens full of twos') beats the full house with 9's and A's ('nines on aces' or 'nines full of aces'). A tie would be broken by the pair.

Flush A flush contains five cards of the same suit, e.g. Q♠ 2♠ 8♠ A♠ 7♠. The highest card determines which flush is better and comparison continues down all five cards until the tie is broken.

Straight A straight consists of any five cards in sequential order. For example, 6♡ 7♣ 8♠ 9♠ 10♣ is a straight. The ace may be counted high or low, but not at the same time, for example,. 3♠ 2♣ A♡ K◇ Q◇ is not a straight. The straight with the highest top card is the higher-valued straight.

Three of a kind Three of a kind contains three cards of one rank and two cards of two other ranks, A♣ A♠ A◇ 10♣ 7♡ is a three of a kind. When comparing two hands with three of a kind, the triple determines which hand wins. If the triples are equal, then the remaining cards in the hand are compared.

Two Pairs A hand with two pairs contains two sets of two cards with the same rank, e.g. J♠ J♣ 2◇ 2♣ 7♡. When comparing hands, first the highest ranking pairs are compared, then the lower pairs, and finally the odd cards.

Pair A hand with two cards of the same rank and the remaining cards of differing ranks, 2♡ 2♣ 7♠ K♡ 10◇ contains a pair of twos. When comparing hands, first the rank of the pairs is compared then the rank of the remaining cards is compared.

High Card For hands with five cards that do not match any of the definitions above, the card with the highest rank is compared first and then the remaining cards are compared in order.

4.1.2 Texas Hold'Em

Texas Hold'Em is one popular variation of poker. It is the variation used in The World Series of Poker, played in Las Vegas, Nevada once a year. In this tournament, anyone can play as long as they pay the entrance fee of $10,000. The appeal of such a tournament is that an amateur player can win against professional players. Even outside of a tournament, poker has a certain appeal as it is possible for a poor player to win a big pot. This is especially true of Texas Hold'Em, as it has been said that any two cards can win.

To start a hand in Texas Hold'Em, the initial bets are handled as follows. The first two players to the left of the dealer are required to bet before any cards are dealt. These bets are *blind* bets. The first blind bet is called the *small blind* and is a fraction of the betting increment, usually one half. The second blind bet is called the *big blind* and is the full betting increment. The actions available to subsequent players are fold, call, or raise.

In Texas Hold'Em, there is a maximum of four betting rounds to a hand. These are called the *preflop*, the *flop*, the *turn*, and the *river*. In the first round, the *preflop*, each player is dealt two cards, called the *hole cards*. These two cards are not shown to any other player unless a *show down* is requested at the end of the hand. The players then act in turn, either folding, calling or raising until all bets have been called. In the next round, called the *flop*, three cards are dealt face up. The players use these community cards and the two in their hand to make a five card poker hand. Again, the players act in turn until all bets are called. In this round, the possible actions start at checking (essentially a 'pass') and betting. After a player bets for the first time, the subsequent players may fold, call or raise. This continues until all bets have been called. At the third round, called the *turn*, a single card is dealt face up. Now the players have six cards (four community cards and two *hole cards*) with which to create the best poker hand. The betting, or action, is played just like the previous round. In the last possible round, called the *river*, a final card is dealt face up, making five community cards. Again, the action continues as in the previous two rounds. It is important to note that not every hand makes it to the *river*. If there is one player

remaining in any round, that player wins the pot. This may even happen on the *preflop*. On the other hand, if there are two or more players left at the end of the *river*, a *show down* is called. The players then show their hands and the dealer (in casinos, a professional dealer employed by the casino) decides who the winner is. If the players are tied, they split the pot.

While games exist where there is no limit on the amount of money that a player can bet in a particular round, the most widely played games are those where there is a limit on betting amounts. One of the more common forms of limits are structured limits, which specify the betting increments. A player may only bet or raise a certain increment. Typically, the increments change depending on the round of play. The game then is identified by its increments. For example, \$10-\$20 Texas Hold'Em means that for the first two rounds, the *preflop* and the *flop*, bets and raises are in increments of \$10. For the *turn* and the *river*, bets and raises are in increments of \$20. The various limits are split into categories, low-limit, middle limit and high limit games. Games with limits below \$10-\$20 are called low-limit games. Games with limits between \$10-\$20 and \$30-\$60 are middle limit games. High limit games are those with limits of at least \$50-\$100.

In games played in casinos, the casino provides the dealer to reduce cheating opportunities (and arguments). A dealer button is passed around the table to signify where to start dealing the cards from. This also signifies who acts first in the betting rounds and who must put in the *small blind* and *big blind* bets. As mentioned earlier, the first player to the left of the dealer puts in the *small blind*. This position is also called the *small blind*. The second player to the left of the dealer puts in one full bet, called the *big blind*. This position is also called the *big blind*. While the rest of the positions do not have formal names, they are often classified into *early*, *middle*, and *late* positions (see Figure 4.1). The third, fourth, and sometimes fifth players to the left of the dealer are in *early* position. These players are among the first to act after the cards are dealt. The sixth, seventh and again sometimes the fifth players to the left of the dealer are in *middle* position. The eighth and ninth players last position. These players are among the last to act after the cards are dealt. The final

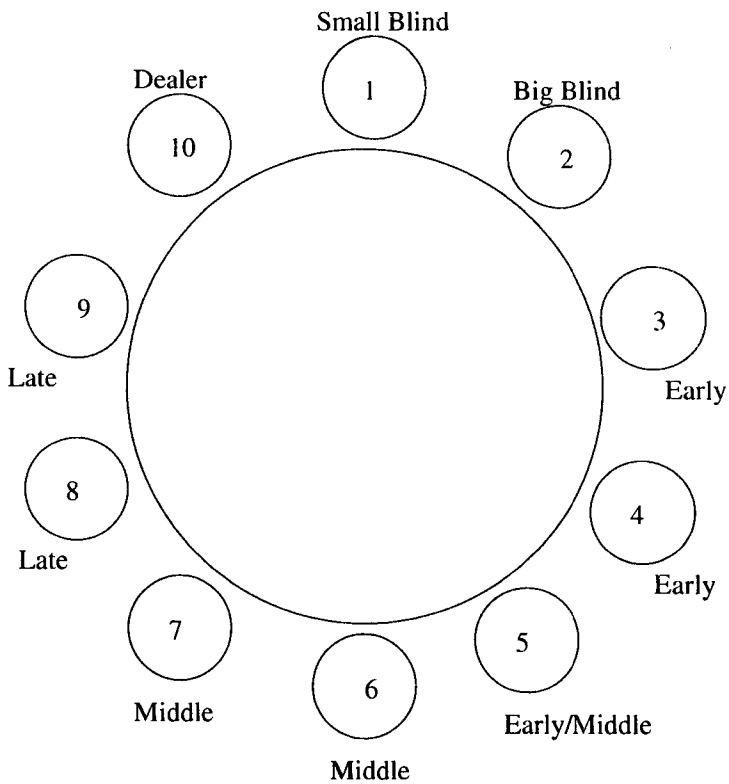

FIGURE 4.1. Diagram of positions for Texas Hold'Em.

position is the dealer himself. This player, also called the button, is the last to act after the cards are dealt. From a strategic point of view, it is ideal to act last in the round as more information about opponents' hands is available.

4.1.3 Collusion

Collusion occurs when two or more players play differently against each other than they do against the rest of the table. When this happens without an explicit agreement between the players, it is called implicit collusion. Explicit collusion (or what most people mean when they talk about collusion) occurs when there is an agreement between the players. They typically split the winnings privately.

One method of collusion occurs when players share which hole cards they are holding with each other. The strongest hand plays and the rest fold. Another method involves one player knowingly playing a hand more weakly than the hand merits, i.e. a player calls or folds when he should have raised or a player folds when they should have called. A third method of collusion occurs when two players raise and re-raise each other to try to drive all the other players out of the pot. Finally, if one member has a strong hand, the colluding players can try to keep the other players in the game and get more money out of them.

4.1.4 Online Poker

The growth of the Internet has fueled an increase in online gambling. A number of websites host online poker play for real money. Typically, a player has to download client software to play and set up an account with the site. To play for real money, the player must deposit money into the account, which they can then use to buy chips. Money is deposited via either an e-cash system (e.g. NETeller or FirePay) or a credit card. When a player decides to "cashout", the money is either credited to the e-cash account or to the credit card.

In addition to play for money, the poker websites also offer play for free. Free poker

play is also available[2] on Internet Relay Chat (IRC). Long before instant messaging became popular, there was "real time" chatting on IRC. Each chat room or channel was hosted by a server and messages were relayed back and forth between all participants in the group. In addition, commands existed to navigate between chat rooms and get (limited) info on others who were online. The IRC poker channels differed from regular IRC channels in that poker was played. Players create an account on one of the IRC poker channels and can play poker for "IRC bucks". Each account typically starts with 1000 "IRC bucks". The actual play was controlled by IRCbot, a computer poker dealing program written by Todd Mummert . IRCbot handled the poker play, player waiting list management, and could even run tournaments. IRCbot responds to messages sent directly to it and parsed all public messages looking for commands it understood. For example, to fold a hand, one would send the message "fold" either privately to the dealer or publicly to the entire chatroom. For those who did not like the text-based poker play, several graphical IRC poker clients exist for both Windows and Unix computer systems.

4.1.5 Online Collusion Detection

Colluding in online poker play may be somewhat easier than in off-line play. Non-colluding players in the game can observe body language and/or money exchanges between colluders in the parking lot in addition to observing play during the game. If two players are sharing cards, this may be difficult to detect from playing styles alone. When colluders are trying to drive everyone else out of a hand, they will raise and re-raise each other. In Texas Hold'Em, the number of raises in a round is usually capped at three raises total. With this type of collusion, the pot size is large with a lot of raising by the same small set of players, and the raises are often capped. When colluders are trying to keep players in a game, pot sizes can also be large, but there will be more players in later stages of the game. However, instead of lots of raises, there will be more calling and raising is not likely to be capped.

[2] irc.poker.net is still off-line as of February 12, 2003

Players that purposefully play a hand weaker than it merits are difficult to detect unless all the cards are shown at the end.

In online poker play, detecting collusion becomes more difficult. Colluders do not even need to be in the same state to collude. Colluders can share information and make decisions about their actions over the phone or over instant messaging protocols such as AOL Instant Messaging or Jabber. It is even possible for a player to collude with themselves by creating multiple accounts. Since the communication between colluders cannot be intercepted, the only way to detect collusion is by observing play. A quick search of rec.gambling.poker and the poker forums hosted by Two Plus Two Publishing show many posts by players asking if they have been the victims of collusion (2 + 2 Publishing 2000).

Most online poker websites take collusion and cheating very seriously. In addition to examining IP addresses and forbidding multiple accounts, they employ unspecified statistical techniques on the data from each hand which flag suspicious hands. These are then analyzed by professional poker players. When collusion is detected, colluders are banned from play.

Collusion involves players playing a hand differently against certain opponents than they would normally. Thus, collusion play should be detectable by examining a hand history or series of hand histories. We will use data mining techniques to build a classifier to classify poker hands as containing collusion and not containing collusion.

4.2 Data Source

4.2.1 IRC Poker Data

Until the IRC poker server went off-line, monthly hand histories from the IRC poker channels were available via the IRC Poker Database[3]. The hand histories from the 10-20 Texas Hold'Em channel for months between April 1995 and August 2001 were downloaded. The

[3]Only a partial excerpt now exists at http://www.geocities.com/mjmaurer/ircpoker

following months were not available for download: January 2001, January 1998, September - December 1997 and June 1997.

Each month of data consists of several files, the hdb file, hroster file, and pdb files. The hdb file contains the general hand information and is a space (tab) delimited ASCII file. The descriptions of the columns are shown in table 4.1 and an example entry can be found in Appendix B.

Column	Descriptions
1	hand timestamp (number of seconds from Jan. 1, 1970)
2	hand set number (reported by dealer program)
3	hand number (reported by dealer program)
4	number of players dealt cards
5	number of players and pot size at beginning of the flop (separated by a /)
6	number of players and pot size at beginning of the turn (separated by a /)
7	number of players and pot size at beginning of the river (separated by a /)
8	pot size at the end of the hand
9-13	board cards (either 0, 3, 4, 5, blank if missing)

TABLE 4.1. Column definitions for the hdb file from the IRC hand history data.

The hroster file contains the IRC nicknames of the players in each hand. This file is also space delimited. The first column is the hand timestamp. The second column is the number of players dealt cards. The remaining columns are the IRC nicknames of the players in the hand. An example entry can be found in Appendix B.

Each account or IRC nickname has its own pdb file. This space delimited ASCII file contains all the individual player information for all hands a particular player has participated in. The columns in this file are organized as shown in table 4.2. An example entry from a pdb file can be found in Appendix B.

Each single bet a player makes is encoded into a single character. Since a player may make several bets during a betting round, the action for a betting round is encoded into a

Column	Description
1	IRC nickname for this player
2	hand timestamp
3	number of players dealt cards
4	position for this hand (starting at position 1, the small blind)
5	preflop betting action
6	flop betting action
7	turn betting action
8	river betting action
9	bankroll at the start of this hand
10	total amount bet during this hand
11	amount won by this player
12 - 13	hole cards held by this player, if revealed

TABLE 4.2. Column definitions for the pdb files from the IRC poker hand history data.

string of characters. The action encodings and their meanings are shown in table 4.3.

Action	Meaning
-	no action taken, player is out of the hand
Q	players quits the hand (IRC peculiarity)
K	player is kicked from the hand (another IRC peculiarity)
B	blind bet (small blind or big blind)
f	fold
k	check
c	call
b	bet
r	raise
A	player is all-in

TABLE 4.3. Action encodings from player data from the IRC poker hand history data.

4.2.2 Class Determination

Supervised learning tasks require the outputs to be known a priori. In the case of a classification task, the class of all points (both testing and training) must be determined. Determining the class of a particular data point proved more difficult than acquiring the data.

A search of postings to the USENET news group rec.gambling.poker revealed a number of accusations of collusion. While some accusations gave a specific date for the incident, others were not as clear. However, all accusations reported the IRC nicknames involved. Using the date of the postings as a guide, the IRC poker archive was examined to try to find these (alleged) incidents of collusion. Since the accusations involved multiple IRC nicknames, the hroster files were examined to try to find those hands containing the accused IRC nicknames. Unfortunately, no hands corresponding to the accusations could be found in the archive. This is due to a number of causes. First, some months were not available in the archive. Based on the date of the postings, it is likely that some of these incidents occurred during those unavailable months. Second, most of the posts did not specify which channel the incident occurred on. It is possible that the incident did not occur on the 10-20 Texas Hold'Em channel.

Since the veracity of the accusations could not be personally verified, it would be incorrect to assume that the accused nicknames were colluders. In addition, colluding players do not always collude in every game. One collusion accusation was rather interesting, however. This accusation involved one player using multiple dummy accounts to build up his regular account. While the IRC nicknames mentioned in the accusation were not found, similar sets were found. In the original accusation, the colluder used an easily identifiable pattern for his dummy IRC nicknames. All the dummy accounts consisted of two lowercase letters followed by a single digit. Accounts with this pattern were found in May through July 2001. It was decided to used this accusation as the basis of the data set. A model will then be built to detect collusion where one player is colluding with themselves,

4.3 Data Mining IRC Data

4.3.1 Data Preparation

The first step in preparing the data was to collect the hands for the colluding class and non-colluding class. Using the roster information (from the hroster file), the hands were

partitioned into two sets: those that contained colluders (or those who have played with colluders) and those that did not. The set of hands containing collusion was partitioned further into hands that contained IRC nicknames that fit the pattern of the dummy accounts and those that did not. The hands containing IRC nicknames corresponding to the pattern established for the dummy accounts comprised the class of hands containing collusion. The class of hands not containing collusions were hands randomly taken from the set of hands that did not contain colluders or those who played with colluders.

The training set was extracted from the May 2001 hand histories and only used hands where 10 or fewer players were dealt cards. The class of hands with collusion consisted of 212 hands. 240 hands were selected from the hands that did not contain collusion. The test set was extracted from the June 2001 hand histories. There were 16 hands identified as containing collusion and 244 hands selected from the class of hands that did not contain collusion.

After the hands were labeled, the relevant features were extracted. Features were constructed using both the general hand data contained in the hdb file and the individual player data contained in the pdb files to use as much of the data as possible. From the hdb files, the number of players dealt cards, number of players that see the flop, turn and river, the pot size at the beginning of the flop and the board cards were used as features. Since the model is trying to detect the transfer of IRC bucks from many small accounts to one account, this behavior might manifest itself in the number of players that remain in each round as well as pot size. From the pdb files, the action for each round, total amount bet, bankroll at the beginning of the round, the amount won, and the hole cards were used. A visual examination of the hand histories for a few of the dummy players revealed that they all tended to raise and then call in each round. They also do not win any hands. Since each IRC poker account starts with 1000 IRC bucks, the bankroll for these accounts is never greater than the starting amount. These features were processed further so their values are more compatible with the SVM algorithms. Additional processing of these features is explained below.

The cards, both hole and board cards were encoded as integers between 1 and 52. The spades were given values between 40 and 52. Hearts lay between 27 and 39. Diamonds were between 14 and 26 and Clubs comprised the remaining integers. If a card was missing (i.e. there was no showdown or the hand ended before the river) it was encoded as 0. Each card was give its own feature, i.e. there were five board card features (unsorted) and two hole card features (also unsorted). In addition, each position has its own hole cards, thus there were a total of 20 hole card features.

The total amount bet, bankroll at the beginning of the hand and the amount won were expanded into ten features, one for each position. The action for each round was also expanded similarly by position. In addition, each individual action received its own feature. Since the number of bets a player makes in a round can vary, the entire data set was examined and the maximum number of actions for each round was determined. This maximum was then used as the number of features to create for the action in each round and position. For example, if the maximum number of actions for the preflop was three, there would be three features for position 1, three for position 2, and etc. for a total of 30 features. As explained above, each action is encoded by a single letter. A standard technique for categorical variables is to create binary dummy variables, one for each category value. For each data point, the dummy variable corresponding to the category value is set to one and the rest of the dummy variables are set to zero. This technique can drastically increase the number of features. To avoid further feature explosion, these actions were encoded into integers as shown in table 4.4. Missing values were encoded as zero.

Character	-	Q	K	B	f	k	c	b	r	A
Integer	0	1	2	3	4	5	6	7	8	9

TABLE 4.4. Integer encodings for player actions.

The IRC nicknames were deliberately not used as features. We wanted to build a model based on the characteristics of play. By excluding IRC nicknames from the feature space, we prevent the model from focusing on these nicknames exclusively as predictors. This

increases the model's ability to generalize to new players colluding in this style.

Finally, the day of the week and hour of the day were extracted from the timestamp. The timestamp is actually the number of seconds since January 1, 1970. It was suspected that hour of day and day of week might have some predictive ability. In order for a player to collude with themselves, he needs to get as many of the dummy accounts playing at the same time as he can. This is easier to do when the channel is empty or nearly empty, which is more likely to occur at night. It may be easier on certain days of the week as well.

The final feature set contained 222 features. This is larger feature set than most of the previously used data set and it is possible that many features will have no predictive ability at all.

4.3.2 Model Choice

The data consists of two classes, hands in which collusion has occurred and hands in which collusion has not. This is a binary classification task and the binary classifiers discussed previously (SVM, PSVM, Linearized PSVM and ISVM) were used.

4.4 Results

Each algorithm was trained using the training data set and performance was evaluated using the performance measures from previous chapters. The class of colluders was taken to be the positive class for performance analysis. As can be seen in table 4.5, all classifiers did very well on the training set. SVM had the highest accuracy and true positive rate as well as lowest false positive rate. ISVM had the second highest accuracy and true false positive rate. PSVM and ISVM had identical false positive rates. It should be noted that ISVM was unable to complete its search due to memory and disk space restrictions on NEOS. Since the models perform well on the training data, performance was evaluated on the test data. Again, the class of colluders was taken as the positive class for performance analysis, which can be seen in table 4.6. Here, PSVM had the highest accuracy and true positive

Model	Accuracy	True Positive Rate	False Positive Rate
SVM	0.99115	0.99528	0.0125
PSVM	0.92478	0.86792	0.025
Linearized PSVM	0.89159	0.83491	0.058333
ISVM	0.98009	0.98585	0.025

TABLE 4.5. Training set results for all classifiers.

Model	Accuracy	True Positive Rate	False Positive Rate
SVM	0.82692	0.75	0.16803
PSVM	0.91538	0.8125	0.077869
Linearized PSVM	0.87692	0.8125	0.11885
ISVM	0.77692	0.8125	0.22541

TABLE 4.6. Test set performance for all classifiers.

rate as well as the lowest false positive rate. Comparing training and test set performance, PSVM and Linearized PSVM had a small decrease in performance while SVM and ISVM experienced a significant decrease in performance. However, while performance decreased on the test set, all classifiers were able to detect most hands containing players colluding with themselves using several dummy accounts.

This experiment has shown that is it possible to detect collusion using SVM-type classifiers. However, the results would need improvement before use in a real collusion detection system such as those employed by online casinos. While the accuracy is reasonably high, the classifiers miss at best almost 19% of hands containing collusion and falsely identify at best almost 8% of hands without collusion as containing collusion. At worst, they miss 25% of hands containing collusion and falsely identify 23% hands not containing collusion as containing collusion. These error rates are a little too high for a practical system. Performance could probably be improved by using expanded training and test data sets.

4.5 Conclusion

The problem of online collusion detection is an important one for operators of online gambling sites to solve as allowing collusion destroys confidence in the game. We have shown that it is possible to detect hands containing one person colluding with themselves using binary SVM-type classifiers.

The likelihood of the type of collusion detected in this experiment has been mostly eliminated by online casinos by their policies of allowing one account per player. However, it is still possible for other types of collusion to occur. One important collusion type is known as trapping, which is where the two colluding players 'trap' a third player between them. The two players force the third to bet more money in order to stay in the hand. It may be possible to detect this type of collusion on a hand by hand basis using similar features as those used in this experiment. While bankroll may not be a good predictor in this case, the trapping behavior may still be observed in individual actions. However, it may be more fruitful to change the design of the experiment. Instead of classifying hands as containing collusion or not, a model would be built to classify players as colluders or non-colluders. Playing style history, partner history and other calculated information could included as features and may capture the trapping behavior better.

In any case, a collusion detection system built using data mining methodologies should not be used solely to determine if a player should be banned. These systems will probably never achieve 100% correctness and thus will make mistakes. Instead, these systems should be used to identify hands that are likely to contain collusion. These hands can then be examined in more detail by poker experts to determine whether or not collusion actually occurred. Then the online casino can take appropriate action.

CHAPTER 5
CONCLUSION

In this dissertation, two new binary classifiers have been proposed. The first binary classifier, Linearized PSVM, is an extension of Kecman and Arthanari's (2001) work on linearizing SVM to PSVM. Linearized PSVM replaces the quadratic functions in the objective function with absolute value functions and linearizes those. This reformulation is not as sensitive to outlying points as the original PSVM. While the ability to obtain a solution via the Karush Kuhn Tucker conditions has been lost, the amount of time to obtain a solution has not been drastically increased. In addition, Linearized PSVM was reformulated with additional objective function parameters which allow the classifier to incorporate knowledge about the relative importance of the classes and even the relative confidence in the individual data points in the training data set. While Linearized PSVM did not dominate the other classifiers in performance on most of the standard data mining datasets, its performance was comparable to SVM and PSVM. On those datasets Linearized PSVM performed significantly worse, it increased suspicion that the data set itself was not linearly separable, which was shown by visualizing the data set.

The second binary classifier, ISVM, is a reformulation of the linearized support vector machine as an integer program. In ISVM, the distance-based error terms are replaced by binary error-indicator variables. Thus, ISVM minimizes the number of errors instead of a error based on distance and is a better conceptual model of the classification task. However, this comes at a cost of computational speed. ISVM also outperformed SVM and PSVM on several difficult data sets. The increased performance on difficult data sets such as the Wisconsin breast cancer data and the liver disorders data set (if a SVM-type classifier must be used) may warrant the increased solution time. In addition, ISVM is also reformulated to include more objective function parameters. On some data sets, these parameters were

instrumental in helping ISVM find a non-pathological solution.

In examining the performance of these classifiers on data sets commonly used in the literature, no classifier dominated the others in performance. ISVM outperformed SVM and PSVM on more data sets than Linearized PSVM. However, on data sets that are not linearly separable, ISVM and Linearized PSVM can return results different enough from SVM and PSVM to illuminate the non-separability of the data sets without visualizing the data. This is useful when the feature space of the data set has a large number of dimensions.

The binary classifiers were applied to the problem of collusion detection in online gambling. Collusion occurs when players play differently against another player or group of players than they ordinarily would. Explicit collusion occurs when the players have an agreement in place to play in a particular fashion and then split the winnings afterwards. Several behaviors of collusion have been identified by the poker playing community, including trapping other players in order to force extra bets out of them. Online gambling, with its increased anonymity, increases the difficulty of detecting collusion. In addition, collusion may take on new forms, such as one person colluding with themselves by using multiple accounts simultaneously. The hand history data from the IRC poker channels was used to build a model to classify hands as either containing this type of collusion or not containing this type of collusion. It was demonstrated that it is possible to use binary classification techniques to detect one person colluding with themselves in online gambling. However, test set performance must be increased before the classifier can be used in a real system.

5.1 Future Research

Future research for Linearized PSVM and ISVM includes investigating the use of large-scale optimization techniques to increase practical problem size. Most SVM-type classifiers have difficulty with high dimensional problems especially when the size of the training set is large. Since data mining is typically used on large data sets, scaling the classifiers to

large problems would an important contribution.

An additional research question for Linearized PSVM involves detecting a scaling mismatch between objective function terms and setting the parameters to compensate. By inspecting the values of the binary error indicator variables in ISVM, it is possible to determine the cause of poor performance. However, Linearized PSVM has continuous error variables and both positive and negative values should be present. The effect of a scaling mismatch on the error variables is unclear.

Future research into the application of support vector machine classifiers in collusion detection in online gambling include extending the models to incorporate other types of collusion, especially trapping. Another direction for research is to examine collusion in other forms of online gambling.

APPENDIX A

UCI MACHINE LEARNING REPOSITORY DATA EXAMPLES

Feature information for the ionosphere data set:

Fields 1-34	17 pairs of values corresponding to the real and imaginary parts of the processed pulse numbers of the signal.
Field 35	class label, either "good" or "bad".

Feature information for the liver disorders data set:

Feature	**Description**
mcv	mean corpuscular volume, numeric
alkphos	alkaline phospotase, numeric
sgpt	alamine aminotransferase, numeric
sgot	aspartate aminotransferase, numeric
gammagt	gamma-glutamyl transpeptidase
drinks	number of half-pint equivalents of alcoholic beverages drunk per day
class	values = 1, 2. no information given on class meaning

Feature information for the Pima Indians diabetes data set:

Number of times pregnant
Plasma glucose concentration a 2 hours in an oral glucose tolerance test
Diastolic blood pressure (mm Hg)
Triceps skin fold thickness (mm)
2-Hour serum insulin (mu U/ml)
Body mass index (weight in kg/(height in m)^2)
Diabetes pedigree function
Age (years)
Class variable, 1 = tested positive for diabetes, 0 = did not test positive for diabetes

Feature information for the mushroom data set:

Feature	**List of Values**
class	edible=e, poisonous=p
cap-shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
cap-surface	fibrous=f, grooves=g, scaly=y, smooth=s
cap-color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
bruises?	bruises=t, no=f
odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
gill-attachment	attached=a, descending=d, free=f, notched=n
gill-spacing	close=c, crowded=w, distant=d
gill-size	broad=b, narrow=n
gill-color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
stalk-shape	enlarging=e, tapering=t
stalk-root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
stalk-surface-above-ring	ibrous=f, scaly=y, silky=k, smooth=s
stalk-surface-below-ring	ibrous=f, scaly=y, silky=k, smooth=s
stalk-color-above-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
stalk-color-below-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
veil-type	partial=p, universal=u
veil-color	brown=n, orange=o, white=w, yellow=y
ring-number	none=n, one=o, two=t
ring-type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
spore-print-color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

Feature information for the Wisconsin breast cancer data set:

Feature	**Description**
1. ID number	patient number, integer
2. Class	R = cancer recurred, N = cancer did not recur
3. Time	recurrence time if cancer recurred, disease-free time if cancer did not recur
4. mean radius	mean of mean distances from center of each cell nucleus to points on the perimeter
5. mean texture	mean of standard deviation of gray-scale values for each cell nucleus
6. mean perimeter	
7. mean area	
8. mean smoothness	mean of local variation in radius lengths for each cell nucleus
9. mean compactness	mean of perimeter^2 / area - 1.0
10. mean concavity	mean of severity of concave portions of the contour
11. mean concave points	number of concave portions of the contour
12. mean symmetry	
13. mean fractal dimension	mean of "coastline approximation" - 1
14 - 23.	standard error of fields 4 - 13
24 - 33.	mean of three largest values of fields 4 - 13
34. tumor size	diameter of excised tumor in cm
35. lymph node status	number of positive axillary lymph nodes observed during surgery

Feature information for the tic-tac-toe data set:

(x = player x has taken the square, o = player o has taken the square, b = the square is blank)

top-left-square	x,o,b
top-middle-square	x,o,b
top-right-square	x,o,b
middle-left-square	x,o,b
middle-middle-square	x,o,b
middle-right-square	x,o,b
bottom-left-square	x,o,b
bottom-middle-square	x,o,b
bottom-right-square	x,o,b
Class	positive (x won), negative (o won or tie)

Feature information for the Congressional voting records data set:

Feature	**List of Values**
Class Name	(democrat, republican)
handicapped-infants	(y,n,?)
water-project-cost-sharing	(y,n,?)
adoption-of-the-budget-resolution	(y,n,?)
physician-fee-freeze	(y,n,?)
el-salvador-aid	(y,n,?)
religious-groups-in-schools	(y,n,?)
anti-satellite-test-ban	(y,n,?)
aid-to-nicaraguan-contras	(y,n,?)
mx-missile	(y,n,?)
immigration	(y,n,?)
synfuels-corporation-cutback	(y,n,?)
education-spending	(y,n,?)
superfund-right-to-sue	(y,n,?)
crime	(y,n,?)
duty-free-exports	(y,n,?)
export-administration-act-south-africa	(y,n,?)

Feature information for the contraceptive method choice data set:

Feature	**List of Values**
Wife's age	(numerical)
Wife's education	1=low, 2, 3, 4=high
Husband's education	1=low, 2, 3, 4=high
Number of children ever born	(numerical)
Wife's religion	0=Non-Islam, 1=Islam
Wife's now working?	0=Yes, 1=No
Husband's occupation	1, 2, 3, 4
Standard-of-living index	1=low, 2, 3, 4=high
Media exposure	0=Good, 1=Not good
Class	1=No-use, 2=use

Feature information for the MUSK data set:

Feature	**Description**
f1 through f162	Distances along rays measured in hundredths of Angstroms (see Dietterich et al. 1997)
f163	Distance of the oxygen atom in the molecule to a designated point in 3-space.
f164	X-displacement from the designated point.
f165	Y-displacement from the designated point.
f166	Z-displacement from the designated point.
class	0. = non-musk, 1. = musk

APPENDIX B

IRC POKER DATABASE EXAMPLES

Sample line from an IRC Poker Database hdb file:

```
870420768 1 20744 5 4/40 3/160 2/200 1/200 4h 9s 5c 5h 9h
```

Sample line from an IRC Poker Database hroster file:

```
870420768  5 Pyramid donguy fingaz grumpy mysti
```

Sample line from an IRC Poker Data pdb file:

```
Pyramid 870420768 5 1 Bc kf - - 4190 10 0
```

REFERENCES

2 + 2 Publishing (2000). The 2 + 2 forums. http://www.twoplustwo.com. Last accessed on March, 15 2003.

Aha, D. W. (1991). Incremental constructive induction: An instance-based approach. In *Eighth International Workshop on Machine Learning*, pp. 117–121. Morgan Kaufmann.

Aha, D. W., D. Kibler, and M. Albert (1991). Instance-based learning algorithms. *Machine Learning 6*, 37–66.

Androutsopoulos, I., J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos (2000). An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. *SIGIR Forum 34*, 160–7.

Blake, C. and C. Merz (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Boser, B. E., I. M. Guyon, and V. N. Vapnik (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152. ACM Press.

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery 2*(2), 121–167.

Cios, K. J., D. K. Wedding, and N. Liu (1997). CLIP3: cover learning using integer programming. *Kybernetes 26*(4–5), 513–536.

Clark, P. and R. Boswell (1991). Rule induction with CN2: Some recent improvements. In *Proceedings Fifth European Working Session on Learning*, Berlin, pp. 151–163. Springer.

Conway, J. S. and C. Loomis (1995). Using an analog neural network to trigger on tau leptons at cdf. *International Journal of Modern Physics C Physics and Computers 6*(4), 549–54.

Czyzyk, J., M. Mesnier, and J. Moré (1998). The NEOS server. *IEEE Journal on Computational Science and Engineering 5*, 68 – 75.

Dietterich, T. G., R. H. Lathrop, and T. Lozano-Perez (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence 89*(1-2), 31–71.

Dolan, E. (2001, May). The NEOS server 4.0 administrative guide. Technical memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory.

Forsyth, R. S. (1990?). PC/BEAGLE user's guide. Author's address: Pathway Research Ltd., 59 Cranbrook Rd., Bristol BS6 7BS, United Kingdom.

FortMP (1995–1999). *FortMP Manual, Release 3*. Middlesex, United Kingdom: OptiRisk Systems Ltd. http://www.optirisk-systems.com, last accessed February 7, 2003.

Fourer, R., D. Gay, and B. Kernighan (2002). *AMPL: A Modeling Language for Mathematical Programming* (2nd ed.). Duxbury Press / Brooks/Cole Publishing Company.

Fung, G. and O. L. Mangasarian (2001). Proximal support vector machine classifiers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 77–86. ACM Press.

Gropp, W. and J. Moré (1997). Optimization environments and the NEOS server. In M. Buhmann and A. Iserles (Eds.), *Approximation Theory and Optimization*, pp. 167 – 182. Cambridge University Press.

Ha, S. H., S. M. Bae, and S. C. Park (2002). Customer's time-variant purchase behavior and corresponding marketing strategies: an online retailer's case. *Computers and Industrial Engineering 43*(4), 801–820.

Hamuro, Y., N. Katoh, Y. Matsuda, and K. Yada (1998). Mining pharmacy data helps to make profits. *Data Mining and Knowledge Discovery 2*(4), 391–398.

Han, H., X. L. Lu, J. Lu, C. Bo, and R. L. Yong (2002). Data mining aided signature discovery in network-based intrusion detection system. *ACM SIGOPS Operating Systems Review 36*(4), 7–13.

Hastie, T., R. Tibshirani, and J. Friedman (2001a). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Series in Statistics. New York: Springer Verlag.

Hastie, T., R. Tibshirani, and J. Friedman (2001b). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 485–493. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001c). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 480–485. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001d). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 502–503. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001e). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 453–480. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001f). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 439–453. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001g). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 11–14. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001h). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 115–154. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001i). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 165–190. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001j). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 347–367. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001k). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 371–389. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001l). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 267–279. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001m). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 415–427. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Hastie, T., R. Tibshirani, and J. Friedman (2001n). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, pp. 193–221. In *Springer Series in Statistics* Hastie, Tibshirani, and Friedman (2001a).

Iba, W., J. Wogulis, and P. Langley (1988). Trading off simplicity and coverage in incremental concept learning. In *Procedings of the 5th International Conference on Machine Learning*, pp. 73 – 79. Morgan Kaufmann, Publishers.

Ignizio, J. P. and T. M. Cavalier (1994). *Linear Programming*, pp. 285–291,515–27,535–37. Prentice Hall International Series in Industrial and Systems Engineering. Prentice Hall.

Kecman, V. and T. Arthanari (2001, June). Comparisons of QP and LP based learning from empirical data. In L. Monostori, J. Váncza, and M. Ali (Eds.), *Engineering of Intelligent Systems, 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Lecture Notes in Computer Science, Lecture Notes in Artificial Inteligence, pp. 326–332. Springer-Verlag, New York.

Kurgan, L. A., K. J. Cios, R. Tadeusiewicz, M. Ogiela, and L. S. Goodenday (2001). Knowledge discovery approach to automated cardiac SPECT diagnosis. *Artificial Intelligence in Medicine 23*(2), 149–169.

Landwehr, J. M., D. Pregibon, and A. C. Shoemaker (1984). Graphical models for assessing logistic regression models (with discussion). *Journal of the American Statistical Association 79*, 61–83.

Lim, T., W. Loh, and Y. Shih (2000). A comparison of prediction accuracy, complexity and training time of thirty-three old and new classification algorithms. *Machine Learning 40*, 203–228.

Mangasarian, O. L. and D. R. Musicant (2000, April). Active support vector machine classification. Technical Report 00-04, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/00-04.ps.

Mangasarian, O. L. and D. R. Musicant (2001). Lagrangian support vector machines. *Journal of Machine Learning Research 1*, 161–177.

Mangasarian, O. L., W. N. Street, and W. H. Wolberg (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Reserach 43*(4), 570 – 577.

Matheus, C. J. (1990). Adding domain knowledge to SBL through feature construction. In *Eighth National Conference on Artificial Intelligence*, pp. 803–808. AAAI Press.

Matheus, C. J. and L. A. Rendell (1989). Constructive induction on decision trees. In *Eleventh International Joint Conference on Artificial Intelligence*, pp. 645–650. Morgan Kaufmann.

Mummert, T. (1998). An IRC poker dealing program. http://www-2.cs.cmu.edu/afs/cs/user/mummert/public/www/ircbot.html. Last Accessed: February 12, 2003.

Murtagh, B. A. and M. A. Saunders (1983–1998). Minos 5.5 user's guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Department of Operations research, Stanford University, Stanford, CA 94305. http://www.sbsi-sol-optimize.com/manuals/Minoslast accessed February 6, 2003.

Pfitzner, D. W., J. K. Salmon, and T. Sterling (1997). Halo world: Tools for parallel cluster finding in astrophysical n-body simulation. *Data Mining and Knowledge Discovery 1*(4), 419–438.

Provost, F., T. Fawcett, and R. Kohavi (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 445–453. Morgan Kaufmann, San Francisco, CA.

Schlimmer, J. (1987). *Concept Acquisition Through Representational Adjustment (Technical Report 87-19)*. Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine.

Schweitzer, H. (1997). A distributed algorithm for content based indexing of images by projections on ritz primary images. *Data Mining and Knowledge Discovery 1*(4), 375–390.

Sigillito, V., S. Wing, L. Hutton, and K. Baker (1989). Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Technical Digest 10, Johns Hopkins University.

Smith, J. W., J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes (1988). Using the ADAP learning llgorithm to forecast the onset of diabetes mellitus. In *Symposium on Computer Applications and Medical Care*, pp. 261 – 265. IEEE Computer Society Press.

Srinivasan, A. and R. D. King (1999). Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery 3*(1), 37–57.

Street, W. N., O. L. Mangasarian, and W. H. Wolberg (1995). An inductive learning approach to prognostic prediction. In A. Prieditis and S. Russell (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 522 – 530. Morgan Kaufmann.

Swets, J. A. (1988). Measuring the accuracy of diagnostic systems. *Science, New Series 240*(4857), 1285–1293.

Vapnik, V. (1996). *The Nature of Statistical Learning Theory*. New York: Springer-Verlag.

XPRESS (1999). *Using XPRESS-MP in AMPL*. Northants, United Kingdom: Dash Associates. ftp://ftp.mcs.anl.gov/pub/neos/solver_options/AMPLXP.htm, last accessed February 7, 2003.

Yakowitz, S. and J. Mai (1995). Methods and theory for off-line machine learning. *IEEE Transactions on Automatic Control 40*(1), 161 – 165.