University of Nevada, Reno

A Comparative Study on Support Vector Machines

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Mathematics

by

Matthew Awere Ohemeng

Mihye Ahn, Ph.D., Thesis Advisor

December, 2017

University of Nevada, Reno

THE GRADUATE SCHOOL

We recommend that the thesis prepared under our supervision by

MATTHEW AWERE OHEMENG

entitled

A Comparative Study on Support Vector Machines

be accepted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Mihye Ahn, Ph.D., Advisor

Anna Panorska, Ph.D., Committee Member

Ilia Zaliapin, Ph.D., Committee Member

David J. Feil-Seifer, Ph.D., Committee Member

Minggen Lu, Ph.D., Graduate School Representative

David Zeh, Ph.D., Dean, Graduate School

December, 2017

Abstract

In this thesis, we study Support Vector Machines (SVMs) for binary classification. We review literature on SVMs and other classification methods. We perform simulations to compare kernel functions found in selected R packages and also investigate the variable selection property of penalized SVMs. We consider mostly linearly separable data set, mostly linearly non-separable data set, and linearly non-separable data set requiring nonlinear SVMs. In addition, traditional classification methods, including the Linear Discriminant Analysis, Quadratic Discriminant Analysis, K-Nearest Neighbors, and Logistic Regression, are also fit to the data sets and compared to the SVM models. The results of the simulation indicate that choosing a kernel function is key to obtaining a good fit to a particular data set. Moreover, in situations where nonlinear SVMs are not required (such as the linearly separable data set) fitting nonlinear SVMs to a data set might likely result in overfitting. Finally, we apply SVMs and other classification techniques to Alzheimer's disease data.

Dedication

To **my family and friends** including

Mr. Eric Ohemeng

Mrs. Regina Ohemeng

Okyere

Owoahene

Geoffrey Osei-Acheampong

Emmanuel Baiden

Henry Agbewali

Charles Amponsah

Kwame Boamah-Addo

Acknowledgements

First and foremost, I want to express my deepest gratitude to my thesis advisor, Dr. Mihye Ahn, for the conception of this thesis and for her patience, guidance, time, and support in bringing this work to its completion.

I would like to thank the members of the examining committee, Dr. Anna Panorska, Dr. Minggen Lu, Dr. Ilia Zaliapin, and Dr. David J. Feil-Seifer for their time, valuable feedback, and patience.

I would also like to thank the Faculty and Staff at the Math Department UNR from whom I have learned a lot and received support throughout my studies at UNR.

I am also grateful to the creators of R, Professor Ross Ihaka and Dr. Robert Gentleman, and the creators of RStudio J.J. Allaire and the RStudio team.

I would like to thank my parents, Eric Ohemeng and Regina Ohemeng, for their support and prayers throughout my studies at UNR.

Finally, Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG02904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012). ADNI is funded by the National Institute on Aging, the National Institute of Biomedical

Imaging and Bioengineering, and through generous contributions from the following: AbbVie, Alzheimer's Association; Alzheimer's Drug Discovery Foundation; Arcalon Biotech; BioClinica, Inc.; Biogen; Bristol-Myers Squibb Company; Eurolmmun; F. Hoffmann-La Roche Ltd and its affiliated company Genentech, Inc.; Fujirebio; GE Healthcare; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research & Development, LLC.; Johnson & Johnson Pharmaceutical Research & Development LLC.; Lumosity; Lundbeck; Merck & Co., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Neurotrack Technologies; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Takeda Pharmaceutical Company; and Transition Therapeutics. The Canadian Institutes of Health Research is providing funds to support ADNI clinical sites in Canada. Private sector contributions are facilitated by the Foundation for the National Institutes of Health (www.fnih.org). The grantee organization is the Northern California Institute for Research and Education, and the study is coordinated by the Alzheimers Therapeutic Research Institute at the University of Southern California. ADNI data are disseminated by the Laboratory for Neuro Imaging at the University of Southern California.

Contents

List of Figures

Chapter 1

Introduction

Constructing statistical models from data for inference and prediction is central in business, policy making, and health, among others. Some of the statistical tools that learn from data fall under a supervised learning category. Under this category, a statistical model is obtained from a data set referred to as a training data set which has a response variable and a set of explanatory variables (or features). The purpose of obtaining a statistical model is to find the relationship between the explanatory variables and the response variable. When this relationship is found, the estimates of the response on different inputs of explanatory variables can be obtained and inference can also be done to understand how the response is affected by the explanatory variables.

In a classification problem, the response consists of qualitative (or categorical) data and the explanatory variables can be quantitative or qualitative. The categories in the response variable are also referred to as *classes*. Classification problems are *supervised* because the data sets have a response given a set of explanatory variables. An example of classification problems involves predicting whether an email received is a spam email or not based on its content. The data set used to "train" or fit the model is referred to as the *training data* while the test data set is that which is

not part of the training data but on which the fitted model is applied to ascertain its performance. The support vector machine (SVM) is one of many classification methods. Classification techniques or methods are also known as *Classifiers.*

The Bayes Classifier is the optimal classification method when the underlying distribution of the data set is known. It is the optimal since it gives the minimum possible classification error (Hastie et al. (2001); Lin (2002); and James et al. (2013)). The Bayes Classifier assigns a new observation into a class with the highest conditional probability given the values of the explanatory variables. The underlying distribution of real data is generally unknown and this makes it impossible to compute probabilities of the Bayes Classifier. Many classification methods try to classify an observation by assuming an underlying distribution for the data set and then assigning a new observation to the class with the highest probability in hopes of obtaining classification estimates close to the Bayes Classifier. For instance, the Linear Discriminant Analysis (LDA) assumes each class follows a multivariate normal distribution. As shown by Lin (2002), the SVM tries to get estimates close to the Bayes Classifier without assuming an underlying distribution.

Fitting a model in a classification problem entails obtaining a decision boundary or discriminant function which divides the Cartesian coordinate system into regions for each class and assigns an observation into a class based on the corresponding region in which it falls. While some benchmark classification methods such as logistic regression use all of the training data to find a decision rule, the support vector machine relies on a subset of the training data known as *support vectors* to obtain its discriminant function.

In most cases, some explanatory variables are not important in determining the value of the response. Their inclusion in the model decreases the prediction accuracy of the classifier on new observations. Moreover, including such variables in the

model unnecessarily complicates the model thereby reducing model interpretability and also leads to incurring undesirable costs in obtaining such data. This necessitates the need for models which lead to subset selection of explanatory variables. Ideally, the important variables therefore end up being included in the model and the unimportant ones deleted from the model. Regularization methods when applied improve the prediction accuracy of the model. It turns out that the SVM has an "in-built" regularization to check the otherwise poor performance which would have been a result of mapping into very high dimensions. Some of the regularization techniques include the Least Absolute Shrinkage and Selection Operator (LASSO) proposed by Tibshirani (1996) and the Smoothly Clipped Absolute Deviation proposed by Fan & Li (2001). Sparse SVM using the LASSO penalty has been proposed by Wang & Shen (2007). Other regularization approaches to the SVM have also been proposed in various studies (Liu & Wu (2007); Wang et al. (2006); Hastie et al. (2004); and Zhu et al. (2004)).

SVM was developed by Vapnik (1979) who furthered work from statistical learning theory by Vapnik & Chervonenkis (1964). The more current SVM including the soft margin classifier was proposed by Cortes & Vapnik (1995). The SVM is used naturally to solve a binary classification problem for a response variable $y \in \{-1, 1\}$. It should be noted however that work on extension of the SVM to a response with more than two classes has been considered with traditional approaches, including a one-versus-one approach and a one-versus-all approach (James et al., 2013), and also multicategory SVM proposed by Lee et al. (2004). Multicategory methods with analysis of variance kernels and regularization have also been proposed by Lee et al. (2006).

In this study, simulations are done to ascertain the performance of SVMs with different kernels on varied data sets. The data sets used are the mostly linear separable

data, mostly linearly non-separable data, and linearly non-separable data requiring nonlinear SVMs. Other benchmark classifiers which are the K-Nearest Neighbor (KNN), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Logistic Regression are also compared to the SVM models. It is expected that no single model will work best for all varied data sets by the No Free Lunch Theorem (Wolpert & Macready, 1997). Still using simulations, SVMs with different regularization methods are investigated to ascertain how well the prediction accuracy is improved with different penalty functions. All the aforementioned models and varied regularization methods are also applied on Alzheimer's disease data. The data is longitudinal with six classes. Since, we are interested in binary classification, the focus will be on two groups — those who are Cognitively Normal and those diagnosed with Alzheimer's Disease. Explanatory variables which are important in detecting the presence of Alzheimer's Disease will also be investigated using regularized SVMs.

1.1 Goal of Thesis

Upon completing this thesis, we will have a good understanding of how classification methods, especially SVMs, work through the literature review. Simulations to compare kernel functions found in selected R packages will also be done.

1.2 Outline of Thesis

The thesis is organized as follows. Chapter 2 introduces the SVM as a classification method and also defines various regularization approaches which can be used on SVMs. Chapter 3 briefly introduces other classification methods and contains simulation results. In the Chapter 4, classification on Alzheimer's Disease data is done. Chapter 5 discusses and concludes our analyses.

Chapter 2

Literature Review on Support Vector Machines

2.1 Maximal Margin Classifier

Suppose that a training data set has a plot given in Figure 2.1. This is a two-dimensional plot in which each observation has two *explanatory variables (or features)*, X_1 and X_2. The training data can be classified into two classes - purple (open circles) and blue (filled circles). To fit a classification model to the data set, the qualitative classes are represented by discrete values. For binary classification using SVMs, the classes are $+1$ and -1. In our example, we assign the number 1 as the blue class and -1 as the purple class. The response variable is therefore given by $y \in \{-1, 1\}$. Our goal is to use this training data set to obtain a classifier that will classify correctly not only the training data, but also test observations which are not part of the training data.

Observing the plot, it can be stated that the data set is linearly separable. This is because a single line serving as a decision boundary can be drawn to partition the data set into two classes. In this two-dimensional setting, a line which is of the form

$A_1X_1 + A_2X_2 + A_0 = 0$, where A_0, A_1, and A_2 are unknown constants, is required to separate the data set. We let the decision function be given by the left hand side of the hyperplane; that is, $f(\mathbf{X}) = A_1X_1 + A_2X_2 + A_0$. The decision function determines how the assignment of an observation is to be done. In general, in a p dimensional space, a hyperplane of the form $A_1X_1 + A_2X_2 + \cdots + A_pX_p + A_0 = 0$ is required to partition a linearly separable data set. The hyperplane will be of dimension $p - 1$.

The rule followed when classifying an observation is referred to as a *decision rule*. The decision rule for classifying a test observation $\mathbf{X} = (X_1, X_2)^T$ is given by $sign(f(\mathbf{X}))$. The test observation will therefore be classified as having label $+1$ if $f(\mathbf{X}) > 0$ and -1 if $f(\mathbf{X}) < 0$. In a situation where $f(\mathbf{X}) = 0$, the class of the observation is ambiguous since it lies exactly on the decision boundary.

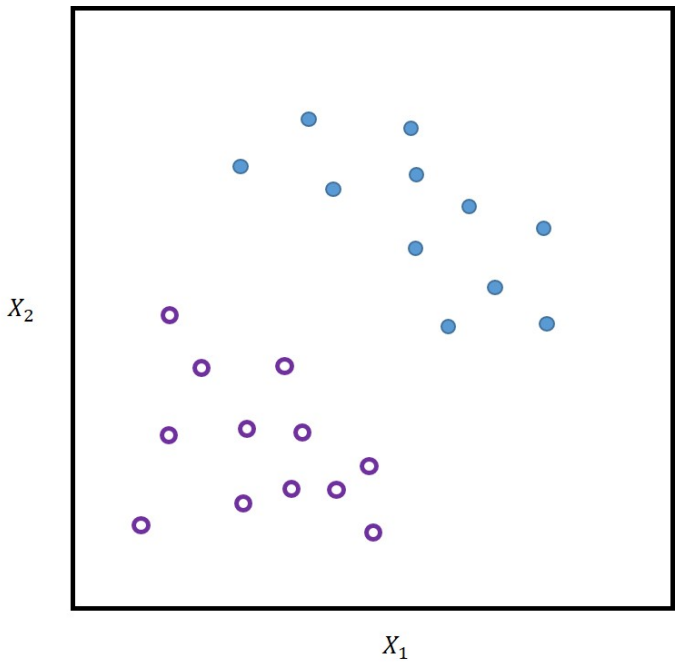

Figure 2.1: An example of linearly separable data. The blue (filled) points are in one class and the purple (open) points are in another class. The data set is linearly separable since a line can be drawn to partition the data set into their respective classes.

Although a line can partition the data set in Figure 2.1 into two classes, a natural question that arises is the best line to use as there are infinitely many lines to choose from. This is because each arbitrary line chosen can be shifted and/or rotated to obtain another line which separates the data set (James et al., 2013). The left-hand panel (A) of Figure 2.2 has five possible lines and illustrates the idea that there are a lot of possible lines which can be used to accomplish the separation goal. A good choice that comes to mind is to find an optimal line which not only separates the two classes, but does it such that it maximizes the distance of the points closest to it from both classes. This is a good choice since the further an observation is from the line, the more confidence in the correct classification of the observation.

The optimal line is obtained by first computing the distances from all the observations in the training data set to the proposed line. The smallest of those distances to the hyperplane from the perspective of the two classes represents the *margins*. The margin in the right panel (B) of Figure 2.2.

There are two margins in the binary classification problem which have the minimal distances from observations with labels $+1$ and -1 to the hyperplane, respectively. The optimal hyperplane is illustrated in the right-hand panel (B) of Figure 2.2.

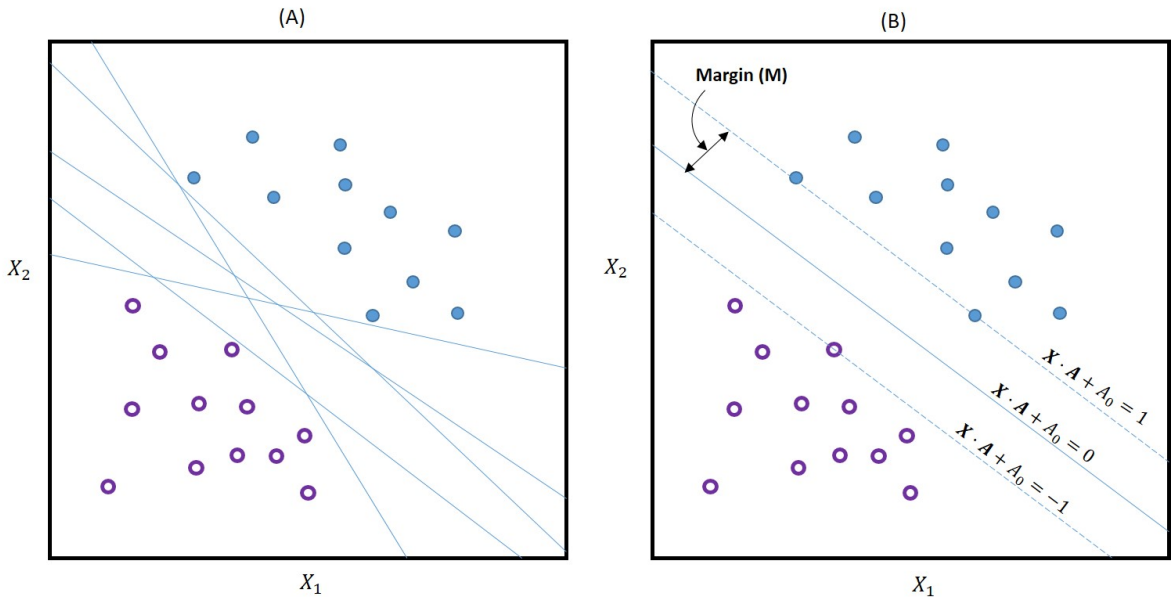

Figure 2.2: In linearly separable data, the left panel (A) shows possible separating lines and the right panel (B) shows the optimal separating line.

In the right-hand panel (B) of Figure 2.2, there is a margin to each side of the optimal line. The line lies exactly in the middle of the dashed lines which represent the margin boundaries (or gutters).

A key observation of the optimal separating line plot in Figure 2.2 is that three points from two classes lie on the dashed lines. It can be seen that these points entirely decide the position of the optimal line. If these points are moved or taken away then the optimal line changes. Points not on the dashed lines do not affect the position of the optimal line when moved or deleted as long as they are not moved onto the dashed lines or into the margin. The optimal line is therefore independent of all other except the three points on the dashed lines in Figure 2.2. These three data points are referred to as *Support Vectors*. Because the optimal line only depends on the support vectors, the maximal margin classifier and for that matter its extension, the SVM, are relatively robust to outliers unlike other classification methods such as the LDA which requires all training data points to decide its decision function.

2.1.1 Maximal Margin Classifier in a p-dimensional space

In this section, we go over the mathematical formulas for obtaining the optimal hyperplane required to separate a linearly separable data set in a p-dimensional space. The ideas for this section are from Hastie et al. (2001), Cortes & Vapnik (1995), and James et al. (2013).

Denote the number of training observations by n. For the ith observation, two classes are represented by labels $y_i \in \{-1, 1\}$, M = Margin, $\mathbf{A} = (A_1, A_2, ..., A_p)^T$, A_0 = unknown constant, and $\mathbf{X}_i = (X_{i1}, X_{i2}, X_{i3}, ..., X_{ip})^T$, where $\mathbf{A}$ is the unknown coefficient (or weight) vector of the optimal hyperplane and $\mathbf{X}_i$ represents the training data points of the ith observation.

Obtaining the maximal margin classifier has an optimization problem given as

$$\begin{aligned} &\underset{\mathbf{A}, A_0}{\text{maximize}} \quad M \\ &\text{subject to} \quad y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) \geq M, \; i = 1, \ldots, n \text{ and } \sum_{j=1}^{p} A_j^2 = 1 \end{aligned} \tag{2.1}$$

The unit vector constraint in (2.1) makes the required weight vector $\mathbf{A}$ unique since without it the weights can be scaled in infinitely many ways. Expressing this mathematically, several hyperplanes can be obtained for $k(\mathbf{X}_i \cdot \mathbf{A} + A_0) = 0$, where $k \neq 0$. The unit vector was chosen because of the nice property that computing the distances become relatively easy. This is because the distance from each training data point to the hyperplane is

$$\frac{y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0)}{\sqrt{\sum_{j=1}^{p} A_j^2}} = y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0).$$

The constraints in (2.1) can be merged by considering the following:

$$\frac{y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0)}{\|\mathbf{A}\|} \geq M \implies y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) \geq M(\|\mathbf{A}\|). \tag{2.2}$$

Since the constraint on the norm of the weight vector was introduced to make it unique, it remains valid when setting $\|\mathbf{A}\| = 1/M$ which also implies $M = 1/\|\mathbf{A}\|$. This reduces the constraint in (2.2) to $y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) \geq 1$ for all i.

The constrained optimization problem in (2.1) can hence be re-expressed as:

$$\begin{aligned} &\underset{\mathbf{A}, A_0}{\text{minimize}} \quad \frac{\|\mathbf{A}\|^2}{2} \\ &\text{subject to} \quad y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) \geq 1 \quad \text{for} \quad i = 1, \ldots, n. \end{aligned} \tag{2.3}$$

Now (2.3) is a convex optimization problem and specifically a quadratic optimization problem with inequality constraints. This problem can be solved by finding the Lagrangian primal function and its corresponding dual function. The Lagrangian primal function is obtained as

$$L(\mathbf{A}, A_0, \lambda) = \frac{\|\mathbf{A}\|^2}{2} - \sum_{i=1}^{n} \lambda_i [y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) - 1], \tag{2.4}$$

where the new variables λ_i's $(i = 1, 2, \ldots, n)$ are the Lagrangian multipliers associated with the constraint from (2.3). To optimize (2.4), we take the partial derivatives with respect to $\mathbf{A}$ and A_0, and set them to equal zero giving

$$\begin{aligned} \nabla_{\mathbf{A}} L = \mathbf{A} - \sum_{i=1}^{n} \lambda_i y_i \mathbf{X}_i = 0 \implies \mathbf{A} = \sum_{i=1}^{n} \lambda_i y_i \mathbf{X}_i \\ \nabla_{A_0} L = -\sum_{i=1}^{n} \lambda_i y_i = 0 \implies \sum_{i=1}^{n} \lambda_i y_i = 0. \end{aligned} \tag{2.5}$$

Substituting the results from (2.5) into (2.4) gives the Wolfe dual problem which

is to be maximized. The Wolfe dual is given by

$$L(\lambda, A_0) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{X}_i \cdot \mathbf{X}_j \tag{2.6}$$

in order to obtain values for λ_i's.

The Wolfe dual problem from (2.6) must also satisfy the Karush-Kuhn-Tucker (KKT) conditions which include results from (2.5) and

$$\lambda_i[y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) - 1] = 0 \text{ and } \lambda_i \geq 0. \tag{2.7}$$

From (2.7), if $\lambda_i > 0$, then $y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) - 1 = 0$. This is true for only support vectors which lie on the margin. Hence for

$$S = \{i : \mathbf{X}_i \text{ is a support vector}\},$$

the optimal hyperplane is therefore obtained as $\hat{f}(\mathbf{X}) = A_0 + \sum_{i \in S} \lambda_i y_i \mathbf{X} \cdot \mathbf{X}_i$. A new observation $\mathbf{X}_i^*$ will be classified based on the decision rule $sign(\hat{f}(\mathbf{X}_i^*))$. A_0 can be obtained by solving (2.7) with $\lambda_i > 0$ for all i which defines the support vectors.

The bigger $|\hat{f}(\mathbf{X}_i^*)|$, the further the observation $\mathbf{X}_i^*$ is from the optimal hyperplane, thus the more confidence in the correct classification of the observation. This is because for linearly separable data, we want the observations to be as far away from the decision boundary as possible. The Lagrangian dual can be solved for using software packages like MOSEK which iterates until convergence and provides platforms on several languages like R. Gill et al. (1981) showed that solving the Lagrangian dual is simpler than solving the Lagrangian primal.

The Maximal Margin Classifier works well in cases where the data set is linearly separable. However, it fails in cases where the data set is not linearly separable as

shown in the top panel (A) of Figure 2.3. It fails since it is not possible to find a single line which partitions two classes.

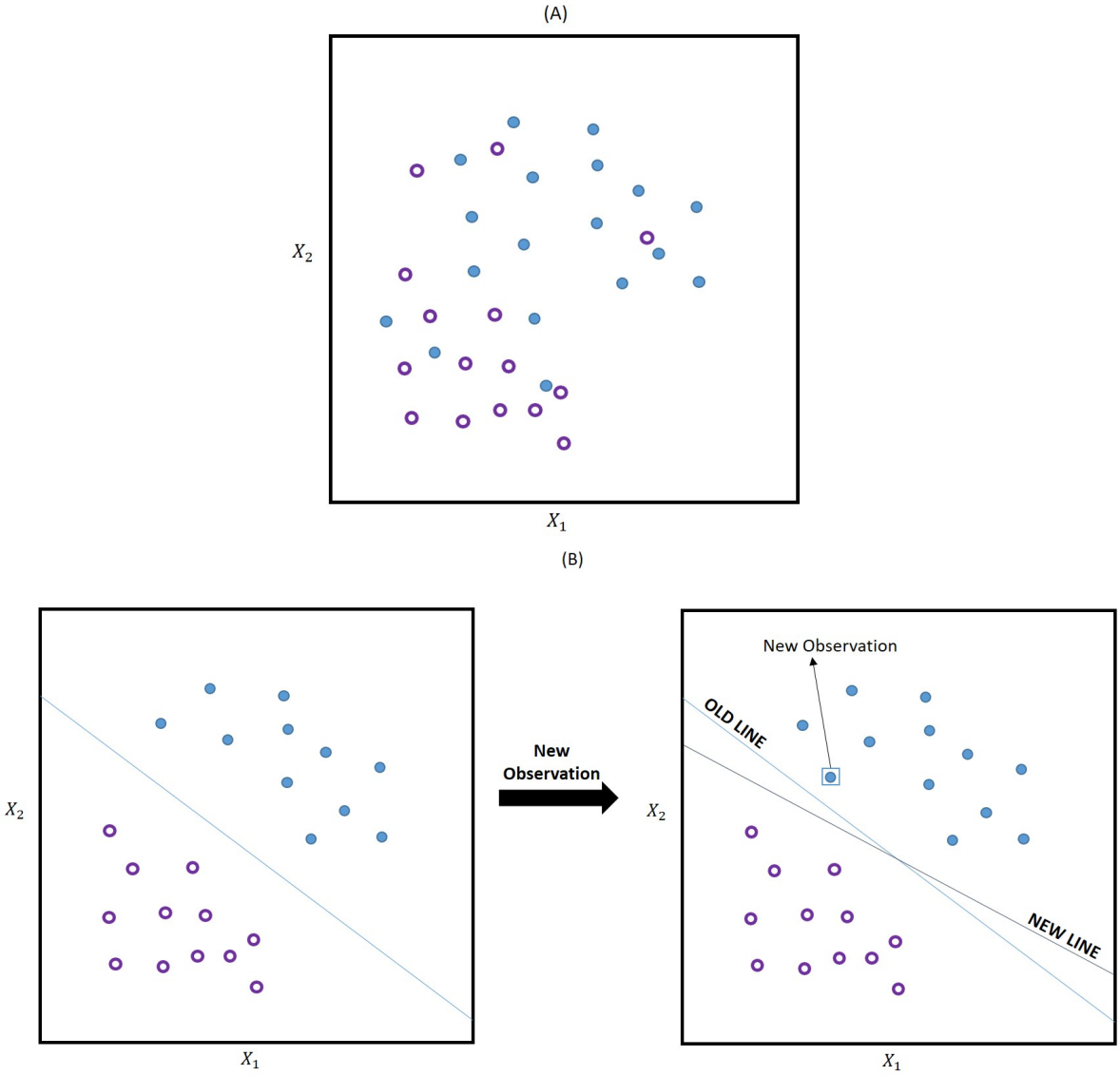

Figure 2.3: Cases where the Maximal Margin Classifier is not preferred. The blue (filled) points belong to one class and the purple (open) points belong to another class. **Top**:(A) Linearly non-separable data. **Bottom**: (B) The optimal separating line is not robust since the introduction of a new observation radically changes the position of the optimal line.

The bottom panels (B) of Figure 2.3 illustrate a situation where we have linearly separable data but the Maximal Margin Classifier is not preferred. This is because an introduction of a single observation causes a significant change in the linear decision boundary. This leads to the suspicion that the classifier may have been overfitted to the training data and hence is not robust. These two situations are among some of the reasons why an extension to the Maximal Margin Classifier is needed. The extension to the Maximal Margin Classifier is the support vector classifier (or sometimes referred to as the soft margin classifier).

2.2 Support Vector Classifier

Given the main issue of robustness of the linear decision boundary and the case of a possible linear inseparability of the data set, the concept of the support vector classifier becomes necessary.

The Support Vector Classifier is an extension of the Maximal Margin Classifier whereby separation of the data set is not done perfectly. A few points are allowed to be misclassified and this has led to this classifier been also referred to as a soft margin classifier as it is not strict or "hard" in classifying all observations perfectly.

Figure 2.4 illustrates the soft margin under the idea of the support vector classifier. For all the data points, we define the slack variables $\epsilon_i \geq 0$ for $i = 1, 2, 3, ..., n$. A slack variable ϵ_i represents the distance from the edge of the margin in the correct side to the misclassified observation. When $\epsilon_i = 0$, then the ith observation was classified appropriately. Where $0 < \epsilon_i \leq 1$, the ith observation has violated the margin. For $\epsilon_i > 1$, the ith observation is on the wrong side of the hyperplane. The total violation of the misclassified observations is given by $\sum_{i=1}^{n} \epsilon_i$.

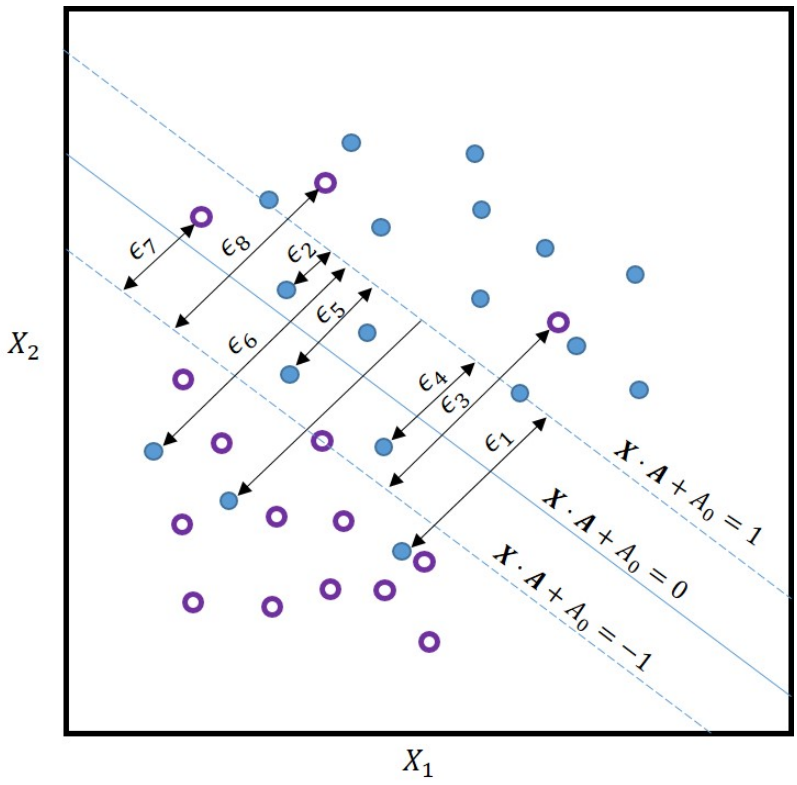

Figure 2.4: Support Vector Classifier where violations to the margin and hyperplane are allowed.

The optimization problem posed by the Support Vector Classifier as an extension of the Maximal Margin Classifier in (2.3) is given by

$$\begin{aligned} &\underset{\mathbf{A},A_0}{\text{minimize}} \quad \frac{\|\mathbf{A}\|^2}{2} \\ &\text{subject to} \quad y_i(\mathbf{A}\cdot\mathbf{X}_i + A_0) \geq 1-\epsilon_i,\ \epsilon_i \geq 0,\ i=1,\ldots,n;\ \sum_{i=1}^{n}\epsilon_i \leq G \end{aligned} \tag{2.8}$$

where G is a constant that serves as an upper bound to check the number of violations to the margin and also the data points that lie on the wrong side of the decision boundary.

In order to solve (2.8), it is rewritten in an equivalent form:

$$\begin{aligned} &\underset{\mathbf{A},A_0}{\text{minimize}} \quad \frac{\|\mathbf{A}\|^2}{2} + C\sum_{i=1}^{n}\epsilon_i \\ &\text{subject to} \quad y_i(\mathbf{A}\cdot\mathbf{X}_i + A_0) \geq 1-\epsilon_i,\ \epsilon_i \geq 0 \ \text{ for } \ i = 1,\ldots,n, \end{aligned} \tag{2.9}$$

where C is a nonnegative "cost" parameter that penalizes the minimization problem based on the size of slack variables allowed. C is generally set via cross-validation (Hastie et al., 2001). When C is set to be very big, then in order to minimize the objective function, the sum of the slack variables is expected to be small which implies little tolerance for violation to the margin. As $C \to \infty$, the problem reduces to the Maximal Margin Classifier and hence little to no tolerance for violation to the margin and hyperplane. On the other hand, as $C \to 0$ the sum of the slack variables can be made to be as big as possible which implies much more allowance for the violation of the margin and hyperplane by more data points. C manages the bias-variance trade-off of the Support Vector Classifier. In cases where C is very large, the slack variables are expected to be small which implies little or no violation to the margin. This means a narrow margin which is closely fit to the data hence a tendency to over-fit the data. In the situation where C is set to be very small, the slack variables can take on relatively bigger values, which implies a wider margin which allows more violations. The classifier obtained will likely be biased.

In the Support Vector Classifier setting, observations that lie on the wrong side of the margin and on the margin are the support vectors.

2.2.1 Obtaining the Support Vector Classifier

From (2.9), the Lagrangian (primal) function is given by

$$L(\mathbf{A}, A_0, \lambda, \mu, \epsilon) = \frac{\|\mathbf{A}\|^2}{2} + C\sum_{i=1}^{n}\epsilon_i - \sum_{i=1}^{n}\lambda_i[y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) - (1-\epsilon_i)] - \sum_{i=1}^{n}\mu_i\epsilon_i, \tag{2.10}$$

where λ_i and μ_i represent the Lagrangian multipliers associated with the constraints expressed in (2.9).

Taking the partial derivatives and setting to zero for $\mathbf{A}$, A_0, and ϵ give

$$\begin{aligned} \nabla_{\mathbf{A}} L = \mathbf{A} - \sum_{i=1}^{n}\lambda_i y_i \mathbf{X}_i = 0 \implies \mathbf{A} = \sum_{i=1}^{n}\lambda_i y_i \mathbf{X}_i \\ \nabla_{A_0} L = -\sum_{i=1}^{n}\lambda_i y_i = 0 \implies \sum_{i=1}^{n}\lambda_i y_i = 0 \\ \nabla_{\epsilon_i} L = C - \lambda_i - \mu_i = 0, \ \forall i \end{aligned} \tag{2.11}$$

Substituting results obtained in (2.11) into (2.10) gives the Lagrangian Wolfe Dual function which is required to be maximized to obtain values of the Lagrangian multipliers.

$$L(\lambda, \mu, A_0) = \frac{1}{2}\left(\sum_{i=1}^{n}\lambda_i y_i \mathbf{X}_i\right)\cdot\left(\sum_{j=1}^{n}\lambda_j y_j \mathbf{X}_j\right) + \sum_{i=1}^{n}(\lambda_i + \mu_i)\epsilon_i - \sum_{i=1}^{n}\lambda_i y_i \mathbf{X}_i \cdot \left(\sum_{j=1}^{n}\lambda_j y_j \mathbf{X}_j\right) - A_0\sum_{i=1}^{n}\lambda_i y_i + \sum_{i=1}^{n}\lambda_i(1-\epsilon_i) - \sum_{i=1}^{n}\mu_i\epsilon_i$$

$$L(\lambda, \mu, A_0) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j \mathbf{X}_i \cdot \mathbf{X}_j - \sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j \mathbf{X}_i \cdot \mathbf{X}_j + \sum_{i=1}^{n}\lambda_i$$

$$L(\lambda, \mu, A_0) = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y_i y_j \mathbf{X}_i \cdot \mathbf{X}_j \tag{2.12}$$

In order to obtain values for the λ_i's and μ_i's, (2.12) needs to be maximized. It is

key to note that (2.12) depends on the dot product of the training data points. The KKT conditions for (2.12) include the results from (2.11) and

$$\begin{aligned} \lambda_i[y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) - (1 - \epsilon_i)] &= 0 \\ \mu_i \epsilon_i &= 0 \\ y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) - (1 - \epsilon_i) \geq 0;\ \lambda_i \geq 0;\ \mu_i \geq 0 \text{ for } i &= 1, \ldots, n \end{aligned} \tag{2.13}$$

From (2.13) and (2.9), it can be inferred that $0 \leq \lambda_i \leq C$. Obtaining values for the λ_i's gives the solution which is required to obtain the Support Vector Classifier. Nonzero λ_i's require $[y_i(\mathbf{X}_i \cdot \mathbf{A} + A_0) - (1 - \epsilon_i)] = 0$. Observations which produce nonzero λ_i's are the support vectors and they include data points on the margin, data points violating the margin and misclassified points. Observations that are correctly classified do not affect the decision boundary. For a new observation $\mathbf{X}^*$ classification will be done based on the decision rule $sign(\hat{f}(\mathbf{X}^*)) = sign(\mathbf{X}^* \cdot \mathbf{A} + A_0)$.

2.3 Support Vector Machine When Kernel Functions are Introduced

In this section we describe the SVM and the need for kernel functions.

There are classification problems where using a linear boundary to separate classes is not always the best approach. The Support Vector Classifier will not perform as well as a nonlinear classifier as shown in Figure 2.5. By the Cover theorem (Cover, 1965), enlarging the feature space will lead to arbitrary separability of a data set.

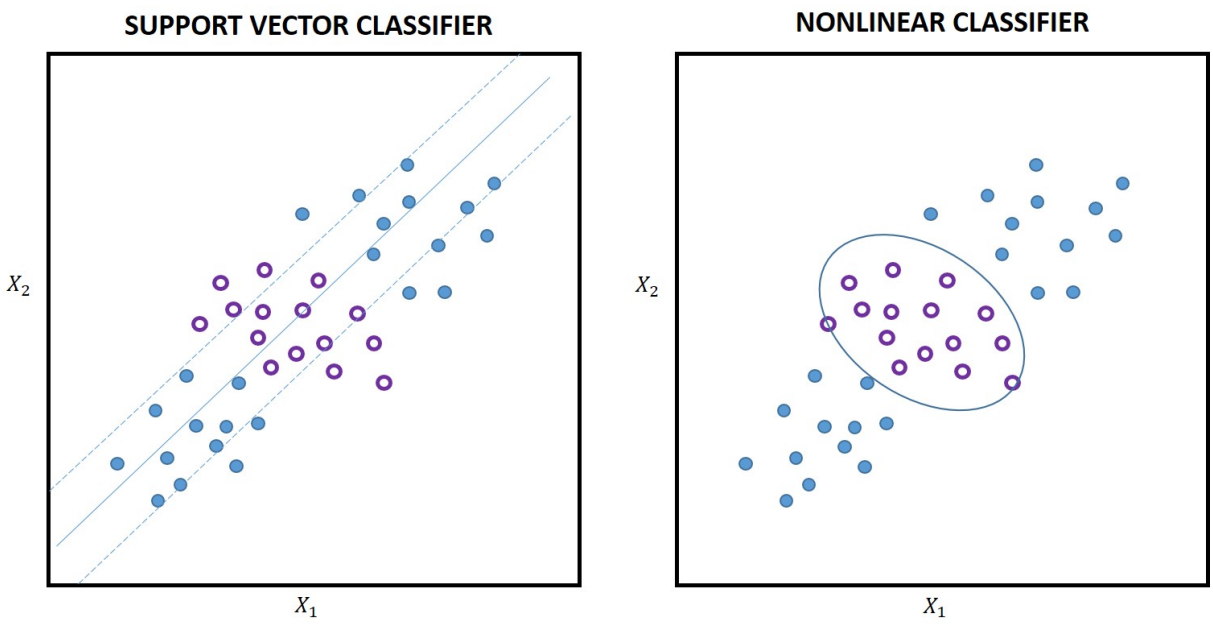

Figure 2.5: Better performance by a nonlinear decision boundary. The blue (filled) points belong to one class and the purple (open) points are observations belonging to another class. **Left**: Fitting a Support Vector Classifier which uses a linear decision boundary. **Right**: Using a nonlinear decision boundary to partition the data set into two classes. Compared to a nonlinear classifier (Right), the Support Vector Classifier (Left) performs rather poorly. This is because the nonlinear decision boundary does a much better job of separating the two classes.

The way to achieve a nonlinear decision boundary is to expand the original feature space of the linear decision boundary. In the aforementioned Support Vector Classifier, a p-dimensional feature space was used where the explanatory variables are $X_1, X_2, ..., X_p$. To achieve a nonlinear decision boundary in this case, various combinations and functions of the features are added to the original feature space. These combinations can include X_1^2, X_2^2, X_1X_2, and X_p^2 (James et al., 2013).

Suppose the feature space is enlarged by adding squares of the original explanatory variables $X_1^2, X_2^2, ..., X_p^2$, then in the original feature space, the decision boundary will be a quadratic polynomial which is nonlinear. However, in the enlarged feature space, the decision boundary will be linear — a hyperplane in p dimensions. This phenomenon is similar to going from linear regression to polynomial regression.

Since the feature space can be enlarged to a very high dimensional space (sometimes infinite space), it follows that computations become intractable. Computation intractability in high dimensional space is looked at in Section 2.3.1. To circumvent this issue of computation inefficiency in the enlarged feature space, the "kernel trick" (Schölkopf et al., 2000) using kernel functions is applied.

In enlarging the feature space, basis functions are applied on the original feature space. Suppose the basis functions for the feature expansion are $b_k(\mathbf{x})$ for $k = 1, ..., r$. These basis functions project the original p-dimensional space into an r-dimensional space. Instead of the original input vector $\mathbf{X}_i = (X_{i1}, X_{i2}, X_{i3}, ..., X_{ip})^T$ where $i \in \{1, 2, 3, ..., n\}$, the vector obtained after applying the basis function, $b(\mathbf{X}_i) = (b_1(\mathbf{X}_i), b_2(\mathbf{X}_i), ..., b_r(\mathbf{X}_i))^T$, is used. This implies that the decision function for a new observation $\mathbf{X}_i^*$, similar to that obtained for the Support Vector Classifier is given by $sign(\hat{f}(\mathbf{X}_i^*)) = sign(b(\mathbf{X}_i^*) \cdot \mathbf{A} + A_0)$.

An illustration of a non-separable data set which becomes separable after feature expansion is shown in Figure 2.6. The data set is originally made up of 4 2-dimensional observations which is not linearly separable. There are two observations for each of the two classes. Using the basis functions (X_1, X_2, X_1X_2) maps the space from 2 dimensions into 3 dimensions. In Figure 2.6, it can be seen that the mapping has now made the data set linearly separable by a hyperplane. The observations of the blue class (filled circles) are now above the separating hyperplane and the other two observations in the purple class (open circles) are now below the hyperplane.

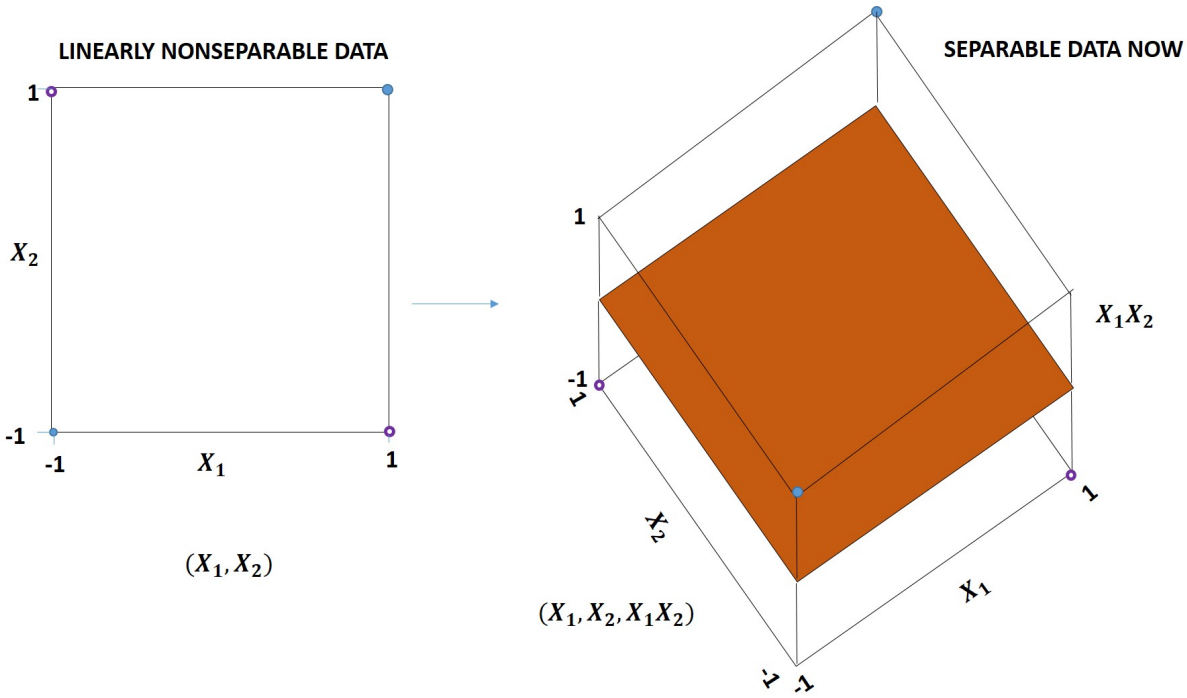

Figure 2.6: Non-separable data become separable by mapping into a higher dimension. **Left**: There are four observations — 2 in each class. The data set is not linearly separable. **Right**: The data set has now been mapped into a 3-dimensional space. This has made the data set linearly separable.

2.3.1 Kernel Functions

We showed that the p feature space has been enlarged to an r feature space (where $r > p$). The input vector is given by $b(\mathbf{X}_i)$. Obtaining the classifier remains similar to the results obtained via the Support Vector Classifier. However, the classifier is referred to as an SVM. Under the enlarged feature space, $\mathbf{A}$ and the function f, from which an unknown vector $\mathbf{X}$ will be classified, are computed by

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i y_i b(\mathbf{X}_i) \text{ and } f(\mathbf{X}) = b(\mathbf{X}) \cdot \mathbf{A} + A_0 = \sum_{i=1}^{n} \lambda_i y_i b(\mathbf{X}) \cdot b(\mathbf{X}_i) + A_0.$$

Cortes & Vapnik (1995) stated a generalization of the dot product required for $f(\mathbf{X})$ above. This generalization is based on the Hilbert-Schmidt Theory (Courant

& Hilbert, 1953). For two vectors $\mathbf{u}$ and $\mathbf{v}$, we define the dot product of the enlarged feature spaces on the vectors by

$$K(\mathbf{u}, \mathbf{v}) = b(\mathbf{u}) \cdot b(\mathbf{v}).$$

The theory states that any symmetric function $K(\mathbf{u}, \mathbf{v})$ that satisfies the condition of the Mercer's Theorem (Mercer, 1909) given by

$$\int \int K(\mathbf{u}, \mathbf{v}) g(\mathbf{u}) g(\mathbf{v}) d\mathbf{u} d\mathbf{v} > 0 \text{ for all } g \text{ such that } \int g^2(\mathbf{u}) d\mathbf{u} < \infty$$

can serve as a dot product. The function $K(\mathbf{u}, \mathbf{v})$ is referred to as a *kernel function* and as a dot product, it measures the similarity of two observations in terms of the direction of vectors. For instance, suppose that the dot product is -1. It implies that the vectors lie on opposite sides; that is, they are dissimilar.

Some choices for the kernel function which will be used in this study include the linear kernel, polynomial kernel, radial kernel, and sigmoid kernel. The forms of these kernels can be seen in Table 2.1.

Name of Kernel Function	Kernel Form ($K(\mathbf{X}_i, \mathbf{X}_j)$)
Linear	$\mathbf{X}_i \cdot \mathbf{X}_j$
Polynomial	$(a + \gamma \mathbf{X}_i \cdot \mathbf{X}_j)^{degree}$, where $a \geq 0$, $\gamma > 0$, degree > 1
Radial	$\exp(-\gamma \lvert \mathbf{X}_i - \mathbf{X}_j \rvert^2)$, where $\gamma > 0$
Sigmoid	$\tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + a)$, where $\gamma > 0$, and $a \geq 0$

Table 2.1: Kernel Functions

Kernel functions help circumvent the problem of prohibitive computations which might result from enlarging the original feature space. As an example to illustrate

the idea of the kernel trick, we use the polynomial kernel as follows. Suppose the data set in which classification is being done is in 2 dimensions. Let a pair of observations in the data set be given by $\mathbf{X}_i = (X_{i1}, X_{i2})^T$ and $\mathbf{X}_j = (X_{j1}, X_{j2})^T$. To enlarge this feature space into a 6−dimensional space (thus $r = 6$), the transformation $b(\mathbf{X}_i) = (1, \sqrt{2}X_{i1}, \sqrt{2}X_{i2}, X_{i1}^2, X_{i2}^2, \sqrt{2}X_{i1}X_{i2})^T$ is used. The decision function depends on the dot product of the new feature space, and hence the dot product is obtained as follows:

$$b(\mathbf{X}_i) \cdot b(\mathbf{X}_j) = (1, \sqrt{2}X_{i1}, \sqrt{2}X_{i2}, X_{i1}^2, X_{i2}^2, \sqrt{2}X_{i1}X_{i2}) \begin{pmatrix} 1 \\ \sqrt{2}X_{j1} \\ \sqrt{2}X_{j2} \\ X_{j1}^2 \\ X_{j2}^2 \\ \sqrt{2}X_{i1}X_{j2} \end{pmatrix} \tag{2.14}$$

$$= 1 + 2X_{i1}X_{j1} + 2X_{i2}X_{j2} + X_{i1}^2X_{j1}^2 + X_{i2}^2X_{j2}^2 + 2X_{i1}X_{i2}X_{j1}X_{j2}.$$

Using the polynomial kernel, however, the aforementioned dot product (2.14) can be computed as follows:

$$b(\mathbf{X}_i) \cdot b(\mathbf{X}_j) = (1 + \mathbf{X}_i \cdot \mathbf{X}_j)^2 = \left(1 + (X_{i1}, X_{i2}) \begin{pmatrix} X_{j1} \\ X_{j2} \end{pmatrix}\right)^2 = (1 + X_{i1}X_{j1} + X_{i2}X_{j2})^2 \tag{2.15}$$

$$= 1 + 2X_{i1}X_{j1} + 2X_{i2}X_{j2} + X_{i1}^2X_{j1}^2 + X_{i2}^2X_{j2}^2 + 2X_{i1}X_{i2}X_{j1}X_{j2},$$

which is the same as $b(\mathbf{X}_i) \cdot b(\mathbf{X}_j)$ computed in (2.14). However, the computations in (2.14) require both (1) a transformation of the two observation vectors into the 6−dimensional space and (2) computing the dot product in that space. On the other hand, (2.15) is computed in the original 2-dimensional space and then squared.

Computations in (2.15) are therefore completed much faster than using the approach in (2.14).

In sum, the decision function of SVM on a given test observation $\mathbf{X}$ is

$$sign(\hat{f}(\mathbf{X})) = sign\left(\sum_{i=1}^{n} \lambda_i y_i K(\mathbf{X}, \mathbf{X}_i) + A_0\right).$$

All unknown parameters can be obtained in the same way as obtained by the aforementioned Support Vector Classifier.

2.4 Performance Measures of a Classifier

In this section, we discuss approaches for evaluating of performance of classifiers. The performance measures considered are the training error rate, testing error rate, sensitivity, specificity, false discovery rate, and area under a receiver operating characteristic curve.

Before performing classification methods on a data set, the given data set needs to be partitioned into a training data set and a testing data set. The training data set is used to "train" or obtain the classifier. The testing data set is new to the classifier and on this data set the prediction of responses can be done to ascertain the performance of the classifier.

Type I error and Type II error which will be discussed in this section stem from hypothesis testing. In hypothesis testing, a null hypothesis is formulated and a decision on whether to reject or fail to reject the null hypothesis is made. A Type I error occurs when a true null hypothesis is incorrectly rejected and a Type II error occurs when a false null hypothesis is incorrectly retained.

2.4.1 Training and Testing Error Rates

In a classification setting, there are generally two types of errors that affect the model fit. These are the training error rate and the testing error rate. For illustration, we denote the numbers of observations in a training data set and in a test data set by n and m, respectively. For a given training data set with responses $y_i \in \{-1, 1\}$ for $i = 1, \ldots, n$, denote the classified observations be $\hat{y}_i$. Then the training error rate is computed as

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i), \text{ where } I(y_i \neq \hat{y}_i) = \begin{cases} 1 & \text{if } y_i \neq \hat{y}_i, \\ 0 & \text{if } y_i = \hat{y}_i. \end{cases}$$

In the same way, the testing error rate is computed using the testing data set. For a testing data set comprising responses $y_j \in \{-1, 1\}$ for $j = 1, 2, \ldots, m$, the classifier is used to obtain predictions for the responses denoted by $\hat{y}_j$. The testing error rate is given by

$$\frac{1}{m}\sum_{j=1}^{m} I(y_j \neq \hat{y}_j). \tag{2.16}$$

A good classifier needs (1) to classify the training data set well but also, more importantly, (2) to classify with a minimal possible error on the testing data set.

Most of the time, it is important to know not only the overall testing error rate of a classifier but also the errors in predicting the respective classes. This is to find out if the model is performing better in predicting one class as opposed to the other. This is especially useful when the two classes have unequal number of observations. For instance, suppose we wish to classify a data set which has 95 cats and 5 dogs. Suppose we obtain a classifier which classifies all the observations as cats. Then the training error rate is 5% which seems to be pretty low. However, if we consider the errors made in each class, then the classifier gives an error of 100% for the class with

dogs which indicates that the classifier did not perform as well as earlier perceived.

Such situations necessitate the need for obtaining performance measures within classes. Such performance measures include sensitivity and specificity discussed in the following section.

2.4.2 Sensitivity, Specificity, and False Discovery Rate

For a data set comprising two classes denoted as -1 and $+1$, the sensitivity is a probability that a predicted class will be $+1$ when the true class is $+1$. The specificity is a probability that the predicted class is -1 given that the true class is -1. The computation of the sensitivity and specificity is illustrated using the "confusion" matrix in Table 2.2.

		Truth		
		$+1$	-1	Total
Prediction	$+1$	True positive (a)	False positive (b)	$a+b$
	-1	False negative (c)	True negative (d)	$c+d$
	Total	$a+c$	$b+d$	n

Table 2.2: Confusion matrix showing possible outcomes when a classifier is applied on training/test data. Letters a, b, c, d denote the number of observations on each combination of predicted/true classes.

In Table 2.2, the predicted class obtained from the classifier is compared to the true class. Letters a and b represent the number of observations that were correctly classified. In a binary classification setting, two errors can be made when classifying an observation. An error is made when an observation whose true class is $+1$ is incorrectly classified as being in class -1 (the total number is shown as c in Table 2.2). Another error that can be made is when an observation whose true class is -1 is incorrectly classified as being in class $+1$ (the total number is shown as b in the

confusion matrix).

The sensitivity from the confusion matrix shown in Table 2.2 is given by $a/(a+c)$ and specificity of a classifier as $d/(b+d)$. $1-$ specificity is also referred to as a *Type I error*. The Type I error is the chance of misclassifying an observation which belongs to the $+1$ class. It can be computed from Table 2.2 as $b/(b+d)$. The proportion of the error which results from incorrectly classifying an observation which belongs to $+1$ class into the -1 class is referred to as a *Type II error*. In the confusion matrix, the Type II error is computed as $c/(a+c)$. It can therefore be said that the sensitivity is $1-$ Type II error.

Another measure of classifier performance which can be computed from the confusion matrix in Table 2.2 is the False Discovery Rate (FDR). The FDR computes the chance that an observation is in the -1 class when it has been classified as being in the $+1$ class. From the confusion matrix, the FDR is computed as $b/(a+b)$.

2.4.3 Receiver Operating Characteristic Curve

The Receiver Operating Characteristic (ROC) curve is a graphical way to compare the performance of classifiers. ROC curves are obtained by plotting the sensitivity (or True positive rate) against $1-$ specificity (or False positive rate). ROC curves are useful in a binary classification setting.

Binary classifiers set a probability threshold based on which classification into either class is done. For instance, suppose there are two classes labeled as -1 and $+1$. The threshold is set on the posterior probability given by $P(Y = +1|\mathbf{X})$, where Y is the response and $\mathbf{X}$ is an array of predictors. If the threshold is 0.5, then the classifier assigns to $+1$ if $P(Y = +1|\mathbf{X}) > 0.5$ and to the other class otherwise.

A change in this threshold can influence the sensitivity and specificity. Suppose initially the threshold is 0.5. If this threshold is changed to 0.2, then there will be

a relatively larger number of observations classified as +1. Hence it is more likely to correctly classify an observation which is in the +1 class. This increases the sensitivity. However, this will also lead to misclassification of observations belonging to the −1 class. This will lead to an increase in the False positive rate (or Type I error) which is 1− specificity. This implies that the specificity declines too.

An ROC curve plots the True positive rate against the False positive rate varying the threshold value for the posterior probability of classifying into the +1 class. The greater the area under the curve obtained on the ROC plot, the better the classifier. In the worst case, this area is 0.5 and it occurs when the classes overlap such that there is no way of distinguishing between them. Figure 2.7 illustrates an ROC curve and AUC.

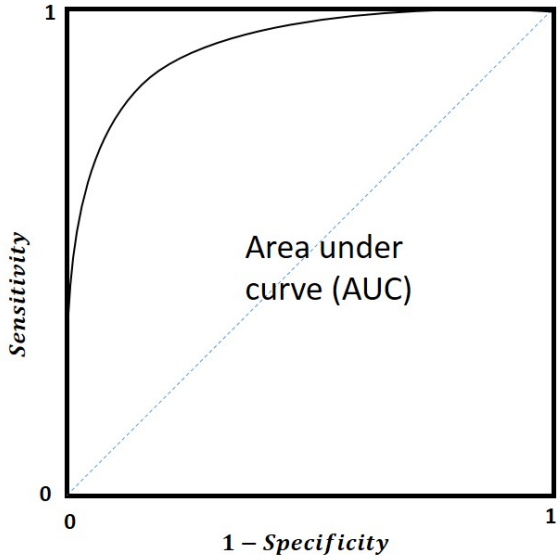

Figure 2.7: An ROC curve

2.5 Support Vector Machines on More Than Two Classes

Although the SVM is intended for a binary classification problem, the idea can be extended to a classification problem where there are more than two classes. Among many, popular approaches to solving this problem are the one-versus-one and the one-versus-all approaches (James et al., 2013).

Both approaches reduce the problem to a binary classification one. Suppose there are F classes, where $F > 2$ and a new test observation is to be classified.

In order to assign a class to the test observation, the one-versus-one approach creates $\binom{F}{2}$ SVMs on a pair of classes. A tally is obtained based on the frequency of classifying this test observation into each of the F classes. The test observation is finally classified based on the class with the highest frequency from the tally obtained.

In the one-versus-all approach, F SVMs are obtained by matching each class against the other $F-1$ classes to create a binary classification problem in each case. In the SVM setting where $\hat{f}$ represents the separating hyperplane, there is more confidence in the classification of a test observation, $\mathbf{X}$, when $|\hat{f}(\mathbf{X})|$ is large. Based on this idea, the test observation is classified into the class where $|\hat{f}(\mathbf{X})|$ is largest.

2.6 Regularization

In a linear regression setting, a continuous response variable y_i is fitted using p regressors, $X_{i1}, X_{i2}, \ldots, X_{ip}$ for $i = 1, 2, \ldots, n$. n represents the total number of observations used to fit the regression model given by

$$y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} + \epsilon_i,$$

where ϵ_i represents the random errors, β_0 is the intercept, and $\beta_1, \ldots, \beta_p$ are the regression coefficients. The regression coefficients are unknown. In order to obtain the estimates of these coefficients, the ordinary least squares (OLS) approach which the residual sum of squares (RSS) is typically used. The estimates for the regression coefficients and the intercept are obtained by minimizing the RSS given by

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j X_{ij} \right)^2 .$$

The fitted model is given by

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p.$$

The model obtained by this approach can be improved on in terms of interpretability and accuracy of prediction. This is because the OLS approach includes all the regressors as none of the regression coefficients are set to zero. In most cases, not all predictors contribute to the model and hence including them makes the model more complex than it could be. To try to mitigate this problem, it is desirable for some of the estimates of the coefficients to be zero or shrunk close to zero to make for a relatively more easily interpretable model. Another issue raised is prediction accuracy.

It is assumed that the true relationship between the regressors and the response is linear. If the true relationship is indeed linear (or approximately linear) then the linear regression model will have low bias and for $n >> p$ the fitted model will have low variance which implies its good performance on test observations. If n is greater than p but not by much, then it will result in a relatively higher variability in the fitted model and predictions made on test observations will not be as good compared to the situation where $n >> p$. Also, in a situation where $n < p$, there will not be a

unique solution for the fitted model which implies its variance will be infinite (James et al., 2013). Once more, shrinking regression coefficients to zero or close to zero improves the prediction accuracy of the model.

Regularization involves the process of shrinking regression coefficients (sometimes shrinking to zero) by imposing a constraint on the regression coefficients. In this study we looked at the Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996), Smoothly Clipped Absolute Deviation (SCAD) (Fan & Li, 2001), Ridge regression (Hoerl & Kennard, 1988), and Elastic net (Zou & Hastie, 2005).

Each of the shrinkage methods constrains or regularizes the regression coefficients using penalties added to the RSS to be minimized. A method like the LASSO can also perform a subset selection of explanatory variables, implying that it sets some of the coefficients to zero and hence removes some regressors completely from the fitted model.

The RSS can be seen as a loss function, $L(\mathbf{X}, \mathbf{y}, \beta)$, that computes the extent to which a model fits a data set. Here $\mathbf{X}$ refers to an $n \times p$ matrix where n represents the total number of observations, p the number of regressors, and $\mathbf{y}$ is an $n \times 1$ matrix. The OLS estimate of the regression coefficients are therefore given as

$$\arg \min_{\beta} L(\mathbf{X}, \mathbf{y}, \beta).$$

By regularization, the above-mentioned solution to regression coefficients using the OLS approach becomes

$$\arg \min_{\beta} L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta) \text{ where } \lambda \geq 0.$$

Constant λ is a tuning parameter which is set via cross-validation. It determines the bias-variance trade off in a model. When $\lambda = 0$, the estimate of the regression

coefficients reduces to the OLS estimates. In a situation whereby $\lambda \to \infty$, the coefficients estimates approach zeros. $P(\beta)$ represents the penalty function. Each of the regularization approaches has different penalty functions.

The regularization approach to linear regression is not exclusive to only continuous response variable. It can also be used in a classification setting. Upon expressing the SVM in terms of some loss function and a penalty, the penalty can be altered using the regularization methods, including LASSO, SCAD, ridge, and elastic net to achieve a similar objective as in the regression setting.

2.6.1 Ridge regression

Hoerl & Kennard (1988) proposed the ridge regression. Ridge regression is a form of regularization whereby the loss function is the RSS and the penalty is given by

$$P(\beta) = \sum_{j=1}^{p} \beta_j^2.$$

This penalty is also referred to as an l_2 penalty. Ridge regression shrinks the regression coefficients close to zero but typically not exactly zero as long as the regularization (or tuning) parameter λ is not approaching infinity in which case a *null* model where no regressors contribute to the model will result. Ridge regression improves the OLS estimate of the regression coefficients by sacrificing some bias in the model for a reduction in the variance and hence a better predictability of the model.

2.6.2 Least Absolute Shrinkage and Selection Operator

Tibshirani (1996) proposed the Least Absolute Shrinkage and Selection Operator (LASSO) as a method of regression shrinkage and selection. As a regularization

method, it has a penalty function which is also referred to as an l_1 penalty given by

$$P(\beta) = \sum_{j=1}^{p} |\beta_j|.$$

Unlike the penalty used in ridge regression which cannot perform variable selection, the LASSO's penalty is able to perform variable selection by setting some of the regression coefficients equal to zero. This is especially useful when some of the regressors do not contribute to explaining the response and removal of these regressors makes model interpretability better. To obtain the LASSO estimate of the regression coefficients, the intersection of the contours generated by the RSS and the constraint region is sought. Suppose $p = 2$, then the constraint region forms a rhombus which is diamond shaped with sharp points on the axes. In such a situation, the contours of the RSS are likely to touch the constraint regions at the axis, which yields a solution where some of the coefficients are set to zero.

2.6.3 Smoothly Clipped Absolute Deviation

The LASSO penalizes large values of the regression coefficients. The penalization at times is excessive since the LASSO forces some regression coefficients to equal zero. As an improvement over situations where excessive penalization of regression coefficients are not desirable, Fan & Li (2001) proposed the Smoothly clipped absolute deviation (SCAD) penalty. Unlike the LASSO and ridge regression penalties which are convex functions, the SCAD penalty function is nonconcave. The SCAD's continuous differentiable penalty function is given by

$$P'_\lambda(\theta) = \lambda \left\{ I(\theta \leq \lambda) + \frac{(a\lambda - \theta)_+}{(a-1)\lambda} I(\theta > \lambda) \right\} \text{ for some } a > 2 \text{ and } \theta > 0.$$

The two parameters a and λ can be obtained through cross validation. The SCAD penalty is also able to perform variable selection (Huang & Xie, 2007).

2.6.4 Elastic Net

Elastic Net is another regularization approach proposed by Zou & Hastie (2005). This regularization approach combines both LASSO and ridge regression penalties defined as

$$P(\beta) = (1-\alpha)\sum_{j=1}^{p}|\beta_j| + \alpha\sum_{j=1}^{p}\beta_j^2 \text{ for } 0 \leq \alpha \leq 1.$$

In situations where $\alpha = 0$ and $\alpha = 1$, the elastic net penalty reduces to that of the LASSO and ridge regression respectively. The elastic net has been shown to be useful and outperforms the LASSO especially when $p >> n$.

Among some of the problems with LASSO brought up by Zou & Hastie are that in a situation where $p > n$, it selects at most n variables which limits its feature selection approach. Also, if the correlations between predictors are very high, then the ridge regression performs better (Tibshirani, 1996).

The elastic net approach performs both variable selection and shrinkage.

2.6.5 Penalized Support Vector Machines

To obtain the soft margin classifier, the optimization problem to be solved was given by

$$\begin{aligned} \underset{\mathbf{A},A_0}{\text{minimize}} \quad & \frac{\|\mathbf{A}\|^2}{2} + C\sum_{i=1}^{n}\epsilon_i \\ \text{subject to} \quad & y_i(\mathbf{A}\cdot\mathbf{X}_i + A_0) \geq 1-\epsilon_i,\ \epsilon_i \geq 0,\ i = 1,\ldots,n. \end{aligned} \tag{2.9}$$

From the two constraints in (2.9), $\epsilon_i = \max(0, 1 - y_i(\mathbf{A}\cdot\mathbf{X}_i + A_0))$. Now, setting

$\lambda = 1/C$, (2.9) can be rewritten as

$$\underset{\mathbf{A},A_0}{\text{minimize}} \quad \frac{\lambda}{2}\|\mathbf{A}\|^2 + \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{A}\cdot\mathbf{X}_i + A_0)). \tag{2.17}$$

Problem (2.17) is in the form of a regularization approach using the l_2 penalty from ridge regression. The loss function given by $\sum_{i=1}^{n} max(0, 1 - y_i(\mathbf{A}\cdot\mathbf{X}_i + A_0))$ is referred to as the *hinge loss*, and it measures the extent to which the model fits the data. When an observation is correctly classified, then $y_i(\mathbf{A}\cdot\mathbf{X}_i + A_0) \geq 1$ hence the loss is 0. On the other hand, if an observation violates the margin or the hyperplane then a penalty is incurred which is proportional to the distance of the observation to the decision boundary. Constant λ is a nonnegative regularization parameter. It serves to control the bias-variance trade-off of the model. As $\lambda \to \infty$, this implies that the margin should be made as wide as possible. This implies that more violation to the margin is allowed and hence more support vectors to fit the model. This will result in the model with a high bias but low variance. The case where $\lambda \to 0$ corresponds to obtaining a very narrow margin. This will result in very few support vectors. Such a model will have high variance due to small number of data points used to fit the model. The bias will however be low. The fact that SVMs have a regularization feature indicates an embedded approach to tackle overfitting.

The penalties from the SCAD, elastic net, and LASSO are all applied on (2.17) to get different forms of SVMs which might perform better in some situations than the standard SVM. Zhu et al. (2004) made this claim using the penalty from the LASSO.

2.7 Cross Validation

In this section, we discuss cross validation which is a central theme in statistical learning. Examples of this method were obtained from James et al. (2013).

When a model is fit to a data set, of interest is not only the training error but also the test error. In most cases, test data set is not readily available. An approach to estimate the test error rate is by using a re-sampling method. Re-sampling methods obtain samples from a training data set and refit the model to get additional information regarding the model which is fit to the data. Cross validation is a re-sampling method which helps in estimating the test error rate. Cross validation techniques include the leave-one-out cross validation (LOOCV) and the k-fold cross validation.

In cross validation, suppose there are n training data points. The data set is divided into the training set and the validation set. The training set is used to fit the model and the validation set is used to compute the test error. The process is repeated a couple of times (depending on the cross validation technique) by re-sampling the validation set and the training set. In the end, the test error of the model is obtained as the average of all test errors obtain from re-sampling.

For LOOCV, the validation set consists of one observation at a time and the rest of the $n-1$ are used to train the model. The process is repeated n times, each time using a different observation as the validation set. In the classification setting, the errors are computed using the approach from (2.16). The estimate for the test error using the LOOCV is therefore given by

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i),$$

where y_i represents the observation in the validation set of the ith re-sample and $\hat{y}_i$

represents the estimate of y_i using the fitted model. The LOOCV approach is similar to finding influential points by using PRESS residuals. In order to obtain the PRESS residual for the ith observation, the observation is deleted and a new regression model is fit. The ith residual can then be obtained as $e_{(i)} = y_i - \hat{y}_{(i)}$ where $e_{(i)}$ is the ith PRESS residual, y_i is the ith observation, $\hat{y}_{(i)}$ is the fitted value of the ith response based on all observations except the ith one. Another re-sampling approach that operates in a similar ways as the LOOCV is the Jackknife (see Quenouille (1949) and Quenouille (1956) for details) re-sampling which helps in estimating bias and variance.

For k-fold cross validation, the training data set is randomly divided into k approximately equally sized subgroups. Similar to the LOOCV, each model is fit using $k-1$ subgroups and one of the subgroups becomes the validation set. Another subgroups is chosen as the validation set and the model fitted using the other $k-1$ subgroups. The process goes on k times making sure that each subgroup becomes the validation set at some point. The test error rate is computed using the approach from (2.16) in each case of re-sampling. Denote the test error rate from the ith validation by $Error_i$. The overall test error rate is computed as $\frac{1}{k}\sum_{i=1}^{k} Error_i$.

Chapter 3

Simulation Study

3.1 Alternative Classification Methods

Common classification methods considered in this study include the Bayes Classifier, Logistic Regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and the $K-$Nearest Neighbors (KNN). Comparison of performance of these classifiers on different kinds of data sets is the purpose of this thesis. We start this section with brief descriptions of the classifiers listed above. We follow with the description of the simulations and results.

3.1.1 The Bayes Classifier

The Bayes classifier assigns an observation to a class with the highest probability given some predictors. Suppose the class labels are given by $y = \{1, 2, ..., j\}$ with $\mathbf{X}$ as a vector of predictors. Then, the Bayes classifier assigns a test observation into a class based on the decision rule $\max_{k \in \{1,2,...,j\}} Pr(Y = k|\mathbf{X})$. In a two-class setting where $y \in \{1, 0\}$, the Bayes classifier assigns an observation to the class 1 if $Pr(Y = 1|\mathbf{X}) > 0.5$. The *Bayes error rate*, which is the test error rate of the Bayes classifier,

is the minimal possible hence serves as a benchmark for all classifiers (Hastie et al., 2001). Practically, it is not possible to make predictions using the Bayes classifier since the underlying distribution of a data set is unknown and in most cases has to be estimated. The Bayes error rate is always greater than zero since the classes of the true population always overlap (James et al., 2013). Some classification techniques try to estimate the Bayes classifier by approximating the underlying distribution. The Bayes classifier is hence the optimal classifier.

3.1.2 Multiple Logistic Regression

In a binary classification setting, suppose $Y \in \{1, 0\}$ and the vector of predictors is given by $\mathbf{X} = (X_1, X_2, ..., X_p)$. The multiple logistic regression model is defined as

$$\log\left(\frac{Pr(Y=1|\mathbf{X})}{1-Pr(Y=1|\mathbf{X})}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p, \tag{3.1}$$

where $\beta_0, \beta_1, ..., \beta_p$ are parameters which need to be estimated via maximum likelihood estimation. $\log(\cdot)$ on the left hand side of (3.1) is referred to as the *logit* or $log-odds$. The prediction into class 1 is obtained based on the estimated conditional probability given by

$$\hat{Pr}(Y=1|\mathbf{X}) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p)}.$$

Usually in a binary classification setting, it will be assigned into class 1 if $\hat{Pr}(Y = 1|\mathbf{X}) > 0.5$. This threshold can however be varied depending on how conservative or otherwise the classification into class 1 is to be done.

3.1.3 Linear Discriminant Analysis

Suppose classification into K classes ($Y \in \{1, 2, 3, ..., K\}$) is to be done given explanatory variables $\mathbf{X} = (X_1, X_2, ..., X_p)$ and n training observations. The LDA approximates the Bayes classifier's discriminant function which is given by

$$\underset{j\in\{1,2,...,K\}}{arg\ max}\, Pr(Y = j|\mathbf{X} = \mathbf{x}) = \arg\max \frac{Pr(Y = j) \cdot Pr(\mathbf{X} = \mathbf{x}|Y = j)}{\sum_{j=1}^{K} Pr(Y = j) \cdot Pr(\mathbf{X} = \mathbf{x}|Y = j)},$$

where $Pr(Y = j)$ is the prior probability of the jth class and $Pr(\mathbf{X} = \mathbf{x}|Y = j)$ is the posterior probability of class j. LDA approximates the prior probability of a class computed as the number of observations in the class divided by the total number of observations. LDA also assumes that the posterior probability given p features/explanatory variables follows a multivariate normal distribution. Each class j, where $j \in \{1, 2, ..., K\}$, has a $p \times 1$ mean vector $\hat{\mu}_j$ and a $p \times p$ covariance matrix $\hat{\Sigma}$ common to all classes. The covariance matrix, the prior probability, and the means of the classes are computed as

$$\hat{\Sigma} = \frac{1}{n-K} \sum_{j=1}^{K} \sum_{Y_i=j} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T; \quad \hat{\mu}_j = \frac{1}{n_j} \sum_{Y_i=j} x_i; \text{ and } \quad \hat{Pr}(Y = j) = \hat{\pi}_j = \frac{n_j}{n} \tag{3.2}$$

The decision rule for LDA is to assign to a class such that

$$\underset{j=1,2,3,...,K}{arg\ \max}\ \mathbf{x}^T \Sigma^{-1} \mu_j - \frac{1}{2} \mu_j^T \Sigma^{-1} \mu_j + \log(\pi_j).$$

3.1.4 Quadratic Discriminant Analysis

QDA is similar to the LDA in all aspects except for the assumption that all classes have the same covariance matrix. QDA assumes that each jth class has a different covariance matrix, Σ_j, for each $j \in \{1, 2, 3, ..., K\}$. The prior probabilities and mean

vectors are computed using the same approach as (3.2) and the decision function is given by

$$\underset{j=1,2,3,...,K}{arg\ \max} -\frac{1}{2}\mathbf{x}^T\Sigma_j^{-1}\mathbf{x} + \mathbf{x}^T\Sigma_j^{-1}\mu_j - \frac{1}{2}\mu_j^T\Sigma_j^{-1}\mu_j - \frac{1}{2}\log|\Sigma_j| + \log\pi_j.$$

3.1.5 $K-$Nearest Neighbors

Suppose the response variable $Y \in \{1, 2, 3, .., M\}$ and there are p explanatory variables, $\mathbf{X} = (X_1, X_2, ..., X_p)$. In order to assign a test observation to a class j (thus $Y = j$), the $K-$ Nearest Neighbors (KNN) classification method first obtains the K closest neighbors/observations and then assigns the observation to the class with the highest probability among the K nearest neighbors. Given that δ_0 contains all K observations nearest to the test observation then the decision rule of assigning an observation into a particular class is given as

$$\underset{j\in\{1,2,...,M\}}{arg\ max}\ Pr(Y = j|\mathbf{X}) \text{ where } Pr(Y = j|\mathbf{X}) = \frac{1}{K}\sum_{i\in\delta_0} I(y_i = j).$$

K is a positive integer and is chosen via cross-validation. It is also important that given the number of responses K is chosen such that there are no ties. For instance, in a two class setting the possible choices for K should not include even integers as this can result in a tie whereby a test observation's neighbors are equally split into the two classes. This will result in an equal probability for classification into the two classes which is indecisive.

The choice of K is very important. When $K = 1$, the decision boundary that results will be very flexible and this is a characteristics of a low bias but high variance classifier. As K increases, the flexibility reduces and that implies a relatively high bias but low variance decision boundary. A very obvious limitation in using the KNN

classifier is that when the training data consists of data points where a particular class is dominant numerically, the likelihood of classifying into that class is higher compared to the other classes and this is likely to lead to misclassification. Figure 3.1 shows a case where 5 nearest neighbors are checked. Computing the probabilities of the two classes included as neighbors of the test observation, the purple class has a probability of 2/5, while the blue class has a probability of 3/5. The blue class has the highest probability and hence the test observation is assigned to the blue class.

Practically, a decision boundary is obtained and this divides the space into regions for the respective classes. If an observation falls in a region, then it is assigned to the corresponding class.

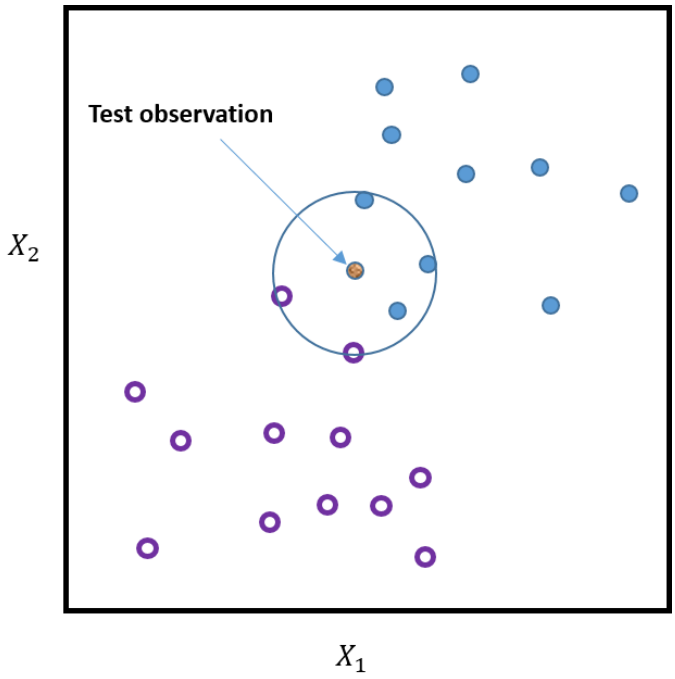

Figure 3.1: 5 Nearest Neighbors. The test observation is classified into the blue class since among the neighbors it has the highest probability which is 3/5.

3.2 Simulation Setting

In simulation studies, three kinds of data sets are generated based on the simulation setting by Liu & Wu (2007). The data sets include those which are mostly linearly separable, another which are mostly linearly nonseparable, and nonseparable data sets which require nonlinear SVM. The binary classification case is considered and hence the response for the data sets is given by $y \in \{-1, 1\}$. R packages which focuses on using different kernels for the SVM used in this study are $e1071$ developed by Dimitriadou et al. (2005), *svmpath* developed by Hastie (2004) and *kernlab* developed by Karatzoglou et al. (2004).

The $e1071$ package was considered because it is the first to implement SVM in R. The *kernlab* has many more kernel functions than the $e1071$ and since kernel functions are of interest in this study, this package was hence considered. The *svmpath* package can obtain the solution path for any cost parameter (Karatzoglou et al., 2005). The other packages for penalized SVMs were considered since they provide options for the penalty functions which are of interest in this study.

Kernels considered by $e1071$ are linear, polynomial, radial, and sigmoid. For *svmpath* we consider the polynomial and radial kernels. Available kernels using the *kernlab* package are radial, polynomial, vanilla, Laplace, Bessel, ANOVA, and spline. The performance of SVMs producing sparse solutions is also sought using $sparseSVM$ by Yi & Zeng (2016) and $penalizedSVM$ by Becker et al. (2009). The ability of penalized SVMs to select important variables is also ascertained. The LASSO and Elastic Net are considered when using the $sparseSVM$ package. For the $penalizedSVM$, the SCAD, LASSO, and Elastic SCAD are considered.

Other classifiers fit to these data sets are the KNN, LDA, QDA, and Logistic Regression using the R packages *caret* for KNN and $MASS$ for LDA and QDA. The *glm* function is used to fit the Logistic Regression model.

The data sets which are mostly linearly separable and that which are mostly linearly nonseparable are generated as follows. 100 random samples are generated for each of the two data types. Each sample has a size of 200 and has ten dimensional variables in all defined by $(x_1, x_2, ..., x_{10})^T$. $(x_1, x_2)^T$ are the variables which decide the response and $(x_3, x_4, ..., x_{10})^T$ are the noise added to the data set. Data for the noise variables are generated independently from $N(0, 9\sigma^2)$. $(x_1, x_2)^T$ are generated by first randomly generating y based on $P(Y = 1) = 0.5$ and then completing the process by generating the data from a two-dimensional mixed normal distribution given by Liu & Wu (2007) as $0.5N((1.5y, 0)^T, \sigma^2 I_2) + 0.5N((0, 1.5y)^T, \sigma^2 I_2)$, where I_2 is a 2×2 identity matrix. Now, setting $\sigma = 0.4$ gives random samples which are mostly linearly separable and also setting $\sigma = 0.8$ gives data sets that are mostly linearly nonseparable.

The nonseparable data sets requiring nonlinear SVM were generated as follow. 100 random samples of size 200 were generated based on 12 explanatory variables. For explanatory variables $(x_1, x_2, x_3, x_4, x_1^2, x_2^2, x_3^2, x_4^2, x_1^3, x_2^3, x_3^3, x_4^3)^T$ where $x_i \sim$ Uniform$[-2, 2]$ for $i = 1, 2, 3, 4$. The three features $(x_1, x_1^3, x_2)^T$ are used in computing a new vector $f(\mathbf{x}) = x_1 - x_1^3 - x_2$ making x_1, x_1^3, and x_2 the important variables which decide the response y. The other variables represent the noise in the data set. $y = 1$ if $f(\mathbf{x}) > 0.5$ and $y = -1$ otherwise. Finally, to make the data set nonseparable, y is recomputed by setting $y = -y$ with probability 0.1 for $yf(\mathbf{x}) < 0.5$.

The following tables show the available kernels for the three R packages.

Table 3.1: Kernels in e1071

kernel	Kernel form
linear	$u'v$
polynomial	$(\gamma u'v + \text{coef0})^{\text{degree}}$
radial	$\exp(-\gamma\|u - v\|^2)$
sigmoid	$\tanh(\gamma u'v + \text{coef0})$

Table 3.2: Kernels in svmpath

kernel
Polynomial
Radial

Table 3.3: Kernels in Kernlab

kernel	Kernel form
Radial Basis	$\exp(-\sigma\|\|x - x'\|\|^2)$
Polynomial	$(\text{scale} \cdot < x, x' > + \text{offset})^{\text{degree}}$
Linear	$< x, x' >$
Hyperbolic tangent	$\tanh(\text{scale} \cdot < x, x' > + \text{offset})$
Laplacian	$\exp(-\sigma\|\|x - x'\|\|)$
Bessel	$\frac{\text{Bessel}^n_{(\nu+1)}(\sigma\|\|x-x'\|\|)}{(\|\|x-x'\|\|)^{-n(\nu+1)}}$
ANOVA RBF	$\left(\sum_{k=1}^n \exp(-\sigma(x^k - x'^k)^2)\right)^d$
Spline	$\prod_{d=1}^D 1 + x_i x_j + x_i x_j min(x_i, x_j) - \frac{x_i, x_j}{2} min(x_i, x_j)^2 + \frac{min(x_i, x_j)^3}{3}$

The penalties available in the sparseSVM are the Elastic Net and LASSO and those of the penalizedSVM are the SCAD, LASSO, Elastic SCAD, and Elastic Net.

The different combinations considered in the various packages are given in the tables that follow. Default values which can be found in parenthesis as well as tuned values are considered for the various parameters.

Table 3.4: Combinations used in e1071

kernel	degree (3)	gamma $(1/p)$	coef0 (0)	cost (1)
linear	X	X	X	O
polynomial	2,3,4,5	O	O	O
radial	X	O	X	O
sigmoid	X	O	O	O

* a default value is in parenthesis; X = not available; O = Options (tuned value or default)

Table 3.5: Combinations used in kernlab

kernel	C (1)	sigma	degree	scale (1)	offset (1)	order (1)
rbfbot	O	O	X	X	X	X
polydot	O	X	1,2,3,4,5	O	O	X
vanilladot	O	X	X	X	X	X
tanhdot	O	X	X	O	O	X
laplacedot	O	O	X	X	X	X
besseldot	O	O	1,2	X	X	1,2
anovadot	O	O	1,2	X	X	X
splinedot	O	X	X	X	X	X

* a default value is in parenthesis; X = not available; O = Options (tuned value or default)

Table 3.6: Combinations used in svmpath

kernel	gamma $(1/p)$	degree
poly.kernel	X	1,2,3,4,5
radial.kernel	O	X

* a default value is in parenthesis; X = not available; O = Options (tuned value or default)

Table 3.7: Combinations used in sparseSVM

alpha (1)	gamma (0.1)
O	O

* a default value is in parenthesis; O = Options (tuned value or default)

Table 3.8: Available penalties using penalizedSVM

method
Scad
1norm
scad+L2

3.3 Simulation Results

3.3.1 Results for Linearly Separable Data

Results for e1071

In simulation studies, different combinations of parameters (default and tuned) were considered. The results shown in Table 3.9 indicate that various kernel functions produced similar results. The model yielding the best results with the best combination of error measures used the linear kernel. It has to be said that the difference in performance of this kernel function from most of the kernel functions is not wide. This kernel tunes the cost parameter which is the only parameter for the kernel. The worst performing kernel is the polynomial kernel with poor performance considering several combinations of parameters. Classification error, sensitivity, specificity, and FDR were the worst for the polynomial kernel in general.

Results for kernlab

From Table 3.10, it can be seen that there is not much difference in the performance measures. However, by the slightest margin, the Laplacian kernel function performs best in kernlab with the best possible combination of error measures. Other kernels perform closely to the Laplacian kernel. These kernels include the radial basis, Laplacian, Bessel, and ANOVA kernels. Of note is the poor performance of the Spline kernel function which recorded relatively unfavorable performance measures. The entire results are seen in Table 3.10 that follows.

Table 3.9: Separable Case. R package e1071 is used. (standard deviation is in parenthesis)

Model No.	Kernel	degree	gamma	coef0	cost	Classification error	Sensitivity	Specificity	FDR	AUC
1	linear	X	X	X	tune	0.006 (0.002)	0.994 (0.003)	0.994 (0.003)	0.006 (0.003)	0.994 (0.002)
2		X	X	X	1	0.009 (0.002)	0.992 (0.004)	0.991 (0.004)	0.009 (0.004)	0.991 (0.002)
3	polynomial	2	1/p	tune	tune	0.007 (0.002)	0.993 (0.003)	0.993 (0.003)	0.007 (0.003)	0.993 (0.002)
4		2	tune	0	tune	0.501 (0.009)	0.500 (0.337)	0.499 (0.335)	0.491 (0.062)	0.499 (0.009)
5		2	tune	tune	1	0.008 (0.005)	0.992 (0.005)	0.991 (0.006)	0.009 (0.006)	0.992 (0.005)
6		2	1/p	0	tune	0.501 (0.010)	0.484 (0.324)	0.515 (0.324)	0.501 (0.046)	0.500 (0.010)
7		2	1/p	tune	1	0.013 (0.005)	0.988 (0.006)	0.987 (0.007)	0.013 (0.007)	0.987 (0.005)
8		2	tune	0	1	0.502 (0.009)	0.509 (0.337)	0.487 (0.337)	0.504 (0.022)	0.498 (0.008)
9		2	1/p	0	1	0.503 (0.010)	0.493 (0.300)	0.502 (0.299)	0.497 (0.055)	0.497 (0.010)
10		3	1/p	tune	tune	0.007 (0.003)	0.993 (0.004)	0.992 (0.004)	0.008 (0.004)	0.993 (0.003)
11		3	tune	0	tune	0.013 (0.004)	0.987 (0.005)	0.986 (0.006)	0.014 (0.006)	0.987 (0.004)
12		3	tune	tune	1	0.009 (0.004)	0.991 (0.006)	0.991 (0.005)	0.009 (0.005)	0.991 (0.004)
13		3	1/p	0	tune	0.013 (0.004)	0.987 (0.006)	0.986 (0.006)	0.014 (0.005)	0.987 (0.004)
14		3	1/p	tune	1	0.013 (0.004)	0.987 (0.006)	0.987 (0.006)	0.013 (0.006)	0.987 (0.004)
15		3	tune	0	1	0.012 (0.002)	0.988 (0.004)	0.988 (0.004)	0.012 (0.004)	0.988 (0.002)
16		3	1/p	0	1	0.012 (0.002)	0.988 (0.004)	0.987 (0.004)	0.012 (0.004)	0.988 (0.002)
17		4	1/p	tune	tune	0.008 (0.002)	0.992 (0.004)	0.992 (0.004)	0.008 (0.004)	0.992 (0.002)
18		4	tune	0	tune	0.499 (0.007)	0.528 (0.386)	0.475 (0.386)	0.491 (0.024)	0.501 (0.006)
19		4	tune	tune	1	0.008 (0.005)	0.991 (0.007)	0.992 (0.005)	0.008 (0.005)	0.992 (0.005)
20		4	1/p	0	tune	0.500 (0.007)	0.489 (0.373)	0.512 (0.374)	0.499 (0.058)	0.501 (0.006)
21		4	1/p	tune	1	0.015 (0.006)	0.986 (0.009)	0.985 (0.008)	0.015 (0.008)	0.985 (0.006)
22		4	tune	0	1	0.500 (0.006)	0.488 (0.417)	0.513 (0.417)	0.484 (0.071)	0.500 (0.005)
23		4	1/p	0	1	0.500 (0.008)	0.493 (0.339)	0.507 (0.338)	0.497 (0.026)	0.500 (0.008)
24		5	1/p	tune	tune	0.010 (0.003)	0.990 (0.004)	0.990 (0.004)	0.010 (0.004)	0.990 (0.003)
25		5	tune	0	tune	0.032 (0.012)	0.968 (0.015)	0.968 (0.018)	0.032 (0.017)	0.968 (0.012)
26		5	tune	tune	1	0.008 (0.004)	0.992 (0.004)	0.991 (0.005)	0.009 (0.005)	0.992 (0.004)
27		5	1/p	0	tune	0.032 (0.013)	0.970 (0.014)	0.967 (0.021)	0.032 (0.019)	0.968 (0.013)
28		5	1/p	tune	1	0.014 (0.006)	0.986 (0.009)	0.985 (0.008)	0.015 (0.008)	0.986 (0.006)
29		5	tune	0	1	0.023 (0.006)	0.978 (0.010)	0.975 (0.012)	0.025 (0.011)	0.977 (0.006)
30		5	1/p	0	1	0.023 (0.005)	0.979 (0.008)	0.976 (0.010)	0.024 (0.010)	0.977 (0.005)
31		3	1/p	0	1	0.012 (0.002)	0.988 (0.004)	0.987 (0.004)	0.012 (0.004)	0.988 (0.002)
32	radial	X	tune	X	tune	0.007 (0.003)	0.993 (0.003)	0.992 (0.004)	0.008 (0.004)	0.993 (0.003)

Table 3.9 Continued

Model No.	Kernel	degree	gamma	coef0	cost	Classification error	Sensitivity	Specificity	FDR	AUC
33		X	tune	X	1	0.007 (0.003)	0.993 (0.003)	0.993 (0.004)	0.007 (0.004)	0.993 (0.003)
34		X	1/p	X	tune	0.011 (0.002)	0.989 (0.004)	0.989 (0.004)	0.011 (0.004)	0.989 (0.002)
35		X	1/p	X	1	0.012 (0.002)	0.988 (0.004)	0.989 (0.004)	0.011 (0.004)	0.988 (0.002)
36	sigmoid	X	1/p	tune	tune	0.008 (0.004)	0.993 (0.004)	0.992 (0.005)	0.008 (0.005)	0.992 (0.004)
37		X	tune	0	tune	0.007 (0.002)	0.993 (0.003)	0.993 (0.003)	0.007 (0.003)	0.993 (0.002)
38		X	tune	tune	1	0.007 (0.003)	0.993 (0.004)	0.992 (0.005)	0.008 (0.005)	0.993 (0.003)
39		X	1/p	0	tune	0.007 (0.003)	0.993 (0.004)	0.993 (0.004)	0.007 (0.004)	0.993 (0.003)
40		X	1/p	tune	1	0.009 (0.003)	0.991 (0.004)	0.991 (0.004)	0.009 (0.004)	0.991 (0.003)
41		X	tune	0	1	0.007 (0.002)	0.993 (0.003)	0.993 (0.003)	0.007 (0.003)	0.993 (0.002)
42		X	1/p	0	1	0.008 (0.002)	0.992 (0.003)	0.992 (0.003)	0.008 (0.003)	0.992 (0.002)

Table 3.10: Separable Case. R package kernlab is used. (standard deviation is in parenthesis)

Model No.	Kernel	C	sigma	degree	scale	offset	order	Classification error	Sensitivity	Specificity	FDR	AUC
1	rbfbot	1	auto	X	X	X	X	0.009 (0.002)	0.991 (0.003)	0.991 (0.003)	0.009 (0.003)	1.000 (0.000)
2		tune	tune	X	X	X	X	0.008 (0.003)	0.993 (0.003)	0.992 (0.006)	0.008 (0.006)	1.000 (0.000)
3	polydot	1	X	1	1	1	X	0.009 (0.002)	0.992 (0.004)	0.991 (0.004)	0.009 (0.004)	1.000 (0.000)
4		1	X	2	1	1	X	0.044 (0.012)	0.955 (0.018)	0.957 (0.016)	0.043 (0.015)	0.990 (0.005)
5		1	X	3	1	1	X	0.029 (0.006)	0.970 (0.011)	0.971 (0.009)	0.029 (0.009)	0.994 (0.002)
6		1	X	4	1	1	X	0.063 (0.015)	0.937 (0.023)	0.937 (0.022)	0.062 (0.020)	0.974 (0.007)
7		1	X	5	1	1	X	0.065 (0.022)	0.932 (0.042)	0.939 (0.038)	0.060 (0.032)	0.978 (0.006)
8	vanilladot	1	X	X	X	X	X	0.009 (0.002)	0.992 (0.004)	0.991 (0.004)	0.009 (0.004)	1.000 (0.000)
9		tune	X	X	X	X	X	0.300 (0.218)	0.687 (0.435)	0.712 (0.427)	0.201 (0.228)	0.999 (0.000)
10	tanhdot	1	X	X	1	1	X	0.078 (0.008)	0.921 (0.021)	0.922 (0.021)	0.077 (0.017)	0.978 (0.004)
11	laplacedot	1	auto	X	X	X	X	0.007 (0.001)	0.993 (0.003)	0.994 (0.003)	0.006 (0.003)	1.000 (0.000)
12		tune	tune	X	X	X	X	0.007 (0.002)	0.993 (0.003)	0.992 (0.004)	0.008 (0.004)	1.000 (0.000)
13	besseldot	1	1	1	X	X	1	0.075 (0.012)	0.924 (0.025)	0.926 (0.028)	0.073 (0.024)	0.981 (0.006)
14		tune	tune	1	X	X	1	0.007 (0.002)	0.994 (0.003)	0.993 (0.004)	0.007 (0.004)	1.000 (0.000)
15		tune	tune	1	X	X	2	0.007 (0.002)	0.993 (0.004)	0.993 (0.004)	0.007 (0.004)	1.000 (0.000)
16		tune	tune	2	X	X	1	0.007 (0.003)	0.993 (0.003)	0.992 (0.005)	0.008 (0.005)	1.000 (0.000)
17		tune	tune	2	X	X	2	0.007 (0.002)	0.993 (0.003)	0.993 (0.004)	0.007 (0.004)	1.000 (0.000)
18	anovadot	1	1	1	X	X	X	0.013 (0.003)	0.987 (0.004)	0.987 (0.005)	0.013 (0.005)	0.999 (0.000)
19		tune	tune	1	X	X	X	0.008 (0.004)	0.992 (0.005)	0.992 (0.005)	0.008 (0.005)	1.000 (0.000)
20		tune	tune	2	X	X	X	0.052 (0.056)	0.953 (0.083)	0.943 (0.098)	0.048 (0.067)	0.997 (0.005)
21	splinedot	1	X	X	X	X	X	0.214 (0.034)	0.775 (0.043)	0.797 (0.053)	0.206 (0.043)	0.833 (0.034)
22		tune	X	X	X	X	X	0.196 (0.037)	0.794 (0.045)	0.814 (0.057)	0.187 (0.048)	0.848 (0.038)

Results for sparseSVM

For the sparseSVM package, of interest is not only the error measures but also the ability of the model to perform variable selection. In the separable case, the important variables are x_1 and x_2. A model that sets the weights of the other variables/features to equal zero but gives estimates for the weights of x_1 and x_2 are therefore correct models.

It must be stated that this package yielded some errors when computations were being done and hence those computations did not yield any results. This is particularly the case for the LASSO penalty with a default tuning parameter $\lambda = 0.1$ which has 54 models not fitted out of the 100 expected models. Complete results were obtained for the other models. To ascertain the best model for variable selection, not only was the number of correct models considered but also the mean of correct zeros too. The LASSO penalty with default lambda yielded the highest mean in terms of obtaining the model with the correct zeros. The LASSO with a tuned lambda however had the highest number of correct models which stood at 22 models.

Considering the error measures too, the Elastic net with tuned lambda, the Elastic net with default lambda and the LASSO with default lambda obtained quite close results. The best model with the best combination of error measures is the Elastic Net with default lambda which performs slightly better than the others in terms of a better FDR. Table 3.11 shows the complete list of simulation considering the error measures while Table 3.12 shows the results for the different penalty functions.

Table 3.11: Separable Case. R package sparseSVM is used. (standard deviation is in parenthesis)

Model No.	alpha	gamma	Classification error	Sensitivity	Specificity	FDR	AUC
1	1	0.1	0.006 (0.003)	0.993 (0.004)	0.994 (0.003)	0.006 (0.003)	1.000 (0.000)
2	1	tune	0.021 (0.085)	0.984 (0.099)	0.974 (0.140)	0.016 (0.070)	0.985 (0.086)
3	tune	0.1	0.006 (0.002)	0.994 (0.004)	0.994 (0.003)	0.005 (0.003)	1.000 (0.000)
4	tune	tune	0.006 (0.002)	0.994 (0.003)	0.994 (0.003)	0.006 (0.003)	1.000 (0.000)

Table 3.12: Separable Case. Result of variable selection. R package sparseSVM is used. (standard deviation is in parenthesis) CZ: Number of correct zeros, IZ: Number of incorrect zeros, CM: Number of correct models, NF: Number of models not fitted due to convergence failure

Model No.	alpha	gamma	mean(CZ)	mean(IZ)	median(CZ)	median(IZ)	CM	NF
1	1	0.1	4.28	0.00	4.00	0.00	9	54
2	1	tune	3.96	0.06	3.00	0.00	22	0
3	tune	0.1	2.02	0.00	1.00	0.00	3	0
4	tune	tune	1.90	0.00	1.00	0.00	3	0

Results for penalizedSVM

Unlike in the sparseSVM where some models were not fitted, the penalizedSVM obtained complete results for all the models and hence recorded zeros for the number of models not fitted. The scad penalty yielded the best variable selection with a complete total of 91 correct models. Here, correct models imply the ability of the penalty to set the coefficients of x_j for $j = 3, 4, 5, ..., 10$ to equal zero since the only important variables are x_1 and x_2. The LASSO penalty performed quite poorly in terms of its ability to select the correct variables. It was only able to obtain 5 correct models out of the expected 100.

The error measures were also considered to ascertain the best performing model in terms a good combination of error measures. The performance of the two penalties SCAD and Elastic SCAD gave the same results except for a small difference the standard deviation of the FDR, specificity, and classification error. By and large, the errors obtained do not show a significant difference between these two competing models however factoring in the ability of the penalty to perform variable selection, the overall best model is the one with the SCAD penalty. Table 3.13 shows the results for the penalized SVMs considering the various error measures while Table 3.14 shows the results for variable selection.

Table 3.13: Separable Case. R package penalizedSVM is used. (standard deviation is in parenthesis)

Model No.	method	Classification error	Sensitivity	Specificity	FDR
1	scad	0.005 (0.002)	0.995 (0.002)	0.995 (0.003)	0.005 (0.003)
2	1norm	0.012 (0.013)	0.987 (0.021)	0.990 (0.011)	0.010 (0.011)
3	scad+L2	0.005 (0.001)	0.995 (0.002)	0.995 (0.002)	0.005 (0.002)

Table 3.14: Separable Case. Result of variable selection. R package penalizedSVM is used. (standard deviation is in parenthesis) CZ: Number of correct zeros, IZ: Number of incorrect zeros, CM: Number of correct models, NF: Number of models not fitted due to convergence failure

Model No.	method	mean(CZ)	mean(IZ)	median(CZ)	median(IZ)	CM	NF
1	scad	7.63	0.00	8.00	0.00	91	0
2	1norm	4.86	0.00	5.00	0.00	5	0
3	scad+L2	7.37	0.00	8.00	0.00	83	0

Results for svmpath

The svmpath package has just two kernels — the polynomial and radial basis kernels. The best kernel is the polynomial kernel of degree 1. The model with the worst combination of error measures is the polynomial kernel of degree 5. Table 3.15 shows the results for all considered cases.

Results for Other Classifiers

The results when other classifiers were fitted to the data set indicate a very high performance by the two classifiers which assume their classes to follow a normal distribution which are LDA and QDA. It can however be seen from Table 3.16 that the performance measures are close and the definition of best or worst may just be due to random fluctuations during computations. These two classifiers have the lowest error among all the classifiers including the SVMs. The KNN performs worst given the error measure combinations obtained. Table 3.16 shows the results for all four classifiers.

Table 3.15: Separable Case. R package svmpath is used. (standard deviation is in parenthesis)

Model No.	Kernel	gamma	degree	Classification error	Sensitivity	Specificity	FDR
1	poly.kernel	X	1	0.009 (0.004)	0.991 (0.007)	0.991 (0.007)	0.008 (0.006)
2		X	2	0.030 (0.009)	0.970 (0.019)	0.971 (0.020)	0.028 (0.019)
3		X	3	0.021 (0.009)	0.978 (0.018)	0.980 (0.019)	0.020 (0.018)
4		X	4	0.056 (0.017)	0.943 (0.034)	0.946 (0.032)	0.053 (0.028)
5		X	5	0.078 (0.041)	0.920 (0.065)	0.924 (0.066)	0.072 (0.056)
6	radial.kernel	$1/p$	X	0.014 (0.005)	0.986 (0.011)	0.987 (0.010)	0.013 (0.010)
7		tune	X	0.011 (0.004)	0.989 (0.007)	0.989 (0.008)	0.011 (0.008)

Table 3.16: Separable Case. Other classifiers are used. (standard deviation is in parenthesis)

Model No.	Classification error	Sensitivity	Specificity	FDR	AUC
LDA	0.005 (0.001)	0.995 (0.001)	0.995 (0.001)	0.005 (0.001)	1.000 (0.000)
QDA	0.006 (0.001)	0.994 (0.003)	0.994 (0.002)	0.006 (0.002)	1.000 (0.000)
Logistic Reg	0.013 (0.006)	0.988 (0.007)	0.987 (0.012)	0.013 (0.011)	0.987 (0.006)
KNN	0.041 (0.010)	0.960 (0.023)	0.959 (0.023)	0.040 (0.021)	0.994 (0.003)

3.3.2 Results for Linearly Nonseparable Data

Results for e1071

Given the different parameter combinations for the e1071, the results show very close performance measures. Although the kernel functions produce similar results, the best model resulting in the best combination of error measures is the linear kernel with a default cost parameter. The worst performing kernel was the polynomial kernel with degree 2, default gamma, default coef0, and tuned cost parameter. The polynomial kernel function had relatively poor performance measures of sensitivity, specificity, FDR and AUC. Table 3.17 shows the results for all considered cases.

Results for svmpath

Considering all the error measures, there are similar results for the two kernel functions. The polynomial kernel with degree 1 represents the linear kernel in the other two R packages. This linear kernel is also referred to as the Support Vector Classifier. Obtaining quite good results is the radial kernel which has the overall highest sensitivity and lowest classification error. Table 3.18 contains all the considered cases.

Results for kernlab

Overall, the results obtained for most of the considered models are similar. The polynomial and linear kernels produced identical results in terms of the best combination of error measures. The Spline kernel function with default cost recorded the worst in all performance measures indicating that this kernel was not suitable for the data set. Table 3.19 shows the results for all the various combinations.

Table 3.17: Nonseparable Case. R package e1071 is used. (standard deviation is in parenthesis)

Model No.	Kernel	degree	gamma	coef0	cost	Classification error	Sensitivity	Specificity	FDR	AUC
1	linear	X	X	X	tune	0.106 (0.007)	0.893 (0.021)	0.895 (0.021)	0.105 (0.017)	0.894 (0.007)
2		X	X	X	1	0.106 (0.007)	0.892 (0.022)	0.896 (0.021)	0.104 (0.017)	0.894 (0.007)
3	polynomial	2	1/p	tune	tune	0.108 (0.009)	0.892 (0.022)	0.893 (0.022)	0.107 (0.018)	0.892 (0.009)
4		2	tune	0	tune	0.501 (0.006)	0.538 (0.350)	0.460 (0.350)	0.502 (0.029)	0.499 (0.005)
5		2	tune	tune	1	0.107 (0.008)	0.892 (0.020)	0.894 (0.019)	0.106 (0.016)	0.893 (0.008)
6		2	1/p	0	tune	0.502 (0.006)	0.517 (0.329)	0.480 (0.329)	0.498 (0.034)	0.498 (0.006)
7		2	1/p	tune	1	0.118 (0.011)	0.883 (0.026)	0.882 (0.028)	0.117 (0.022)	0.882 (0.011)
8		2	tune	0	1	0.501 (0.006)	0.519 (0.362)	0.479 (0.361)	0.497 (0.054)	0.499 (0.006)
9		2	1/p	0	1	0.502 (0.006)	0.492 (0.309)	0.505 (0.310)	0.507 (0.060)	0.498 (0.006)
10		3	1/p	tune	tune	0.109 (0.010)	0.890 (0.021)	0.893 (0.022)	0.107 (0.018)	0.891 (0.010)
11		3	tune	0	tune	0.129 (0.013)	0.870 (0.029)	0.871 (0.027)	0.129 (0.022)	0.871 (0.013)
12		3	tune	tune	1	0.109 (0.009)	0.890 (0.024)	0.892 (0.023)	0.108 (0.019)	0.891 (0.009)
13		3	1/p	0	tune	0.129 (0.014)	0.873 (0.032)	0.868 (0.034)	0.130 (0.026)	0.871 (0.014)
14		3	1/p	tune	1	0.135 (0.013)	0.863 (0.029)	0.868 (0.028)	0.132 (0.022)	0.865 (0.013)
15		3	tune	0	1	0.128 (0.007)	0.870 (0.028)	0.875 (0.025)	0.125 (0.019)	0.872 (0.007)
16		3	1/p	0	1	0.128 (0.007)	0.872 (0.027)	0.873 (0.027)	0.127 (0.020)	0.872 (0.007)
17		4	1/p	tune	tune	0.111 (0.008)	0.888 (0.023)	0.889 (0.022)	0.111 (0.018)	0.889 (0.008)
18		4	tune	0	tune	0.500 (0.005)	0.530 (0.379)	0.471 (0.379)	0.499 (0.011)	0.501 (0.004)
19		4	tune	tune	1	0.109 (0.009)	0.890 (0.024)	0.891 (0.024)	0.108 (0.019)	0.891 (0.009)
20		4	1/p	0	tune	0.501 (0.005)	0.525 (0.360)	0.475 (0.359)	0.493 (0.056)	0.500 (0.005)
21		4	1/p	tune	1	0.153 (0.015)	0.847 (0.031)	0.848 (0.035)	0.152 (0.027)	0.847 (0.015)
22		4	tune	0	1	0.500 (0.004)	0.521 (0.428)	0.479 (0.427)	0.507 (0.091)	0.500 (0.003)
23		4	1/p	0	1	0.499 (0.005)	0.520 (0.337)	0.481 (0.336)	0.499 (0.012)	0.501 (0.005)
24		5	1/p	tune	tune	0.117 (0.011)	0.883 (0.023)	0.883 (0.027)	0.116 (0.022)	0.883 (0.011)
25		5	tune	0	tune	0.174 (0.029)	0.827 (0.070)	0.825 (0.077)	0.169 (0.052)	0.826 (0.028)
26		5	tune	tune	1	0.109 (0.009)	0.890 (0.023)	0.891 (0.023)	0.109 (0.019)	0.891 (0.009)
27		5	1/p	0	tune	0.174 (0.031)	0.827 (0.068)	0.825 (0.081)	0.169 (0.054)	0.826 (0.031)
28		5	1/p	tune	1	0.158 (0.017)	0.841 (0.036)	0.842 (0.032)	0.158 (0.024)	0.842 (0.017)
29		5	tune	0	1	0.152 (0.018)	0.846 (0.063)	0.849 (0.058)	0.148 (0.040)	0.848 (0.018)
30		5	1/p	0	1	0.151 (0.017)	0.847 (0.059)	0.850 (0.056)	0.147 (0.038)	0.849 (0.017)
31		3	1/p	0	1	0.128 (0.007)	0.872 (0.027)	0.873 (0.027)	0.127 (0.020)	0.872 (0.007)
32	radial	X	tune	X	tune	0.107 (0.009)	0.892 (0.022)	0.894 (0.020)	0.106 (0.017)	0.893 (0.009)

Table 3.17 Continued

Model No.	Kernel	degree	gamma	coef0	cost	Classification error	Sensitivity	Specificity	FDR	AUC
33		X	tune	X	1	0.107 (0.009)	0.892 (0.021)	0.894 (0.021)	0.106 (0.018)	0.893 (0.009)
34		X	1/p	X	tune	0.118 (0.009)	0.884 (0.023)	0.880 (0.026)	0.119 (0.020)	0.882 (0.009)
35		X	1/p	X	1	0.120 (0.007)	0.879 (0.023)	0.881 (0.024)	0.119 (0.019)	0.880 (0.007)
36	sigmoid	X	1/p	tune	tune	0.112 (0.012)	0.888 (0.027)	0.889 (0.019)	0.111 (0.016)	0.889 (0.012)
37		X	tune	0	tune	0.106 (0.007)	0.893 (0.020)	0.894 (0.019)	0.105 (0.016)	0.894 (0.007)
38		X	tune	tune	1	0.109 (0.008)	0.891 (0.021)	0.890 (0.020)	0.109 (0.016)	0.891 (0.008)
39		X	1/p	0	tune	0.108 (0.009)	0.892 (0.022)	0.892 (0.021)	0.108 (0.017)	0.892 (0.009)
40		X	1/p	tune	1	0.112 (0.010)	0.888 (0.024)	0.889 (0.025)	0.111 (0.021)	0.888 (0.010)
41		X	tune	0	1	0.106 (0.006)	0.894 (0.019)	0.894 (0.019)	0.106 (0.015)	0.894 (0.006)
42		X	1/p	0	1	0.107 (0.006)	0.891 (0.022)	0.894 (0.022)	0.106 (0.018)	0.893 (0.006)

Table 3.18: Nonseparable Case. R package svmpath is used. (standard deviation is in parenthesis)

Model No.	Kernel	gamma	degree	Classification error	Sensitivity	Specificity	FDR
1	poly.kernel	X	1	0.113(0.012)	0.888(0.034)	0.887(0.034)	0.112(0.027)
2		X	2	0.159(0.014)	0.840(0.057)	0.841(0.058)	0.155(0.040)
3		X	3	0.141(0.017)	0.860(0.052)	0.859(0.057)	0.137(0.042)
4		X	4	0.195(0.030)	0.805(0.074)	0.805(0.085)	0.188(0.057)
5		X	5	0.208(0.035)	0.798(0.112)	0.786(0.121)	0.197(0.074)
6	radial.kernel	1/p	X	0.122(0.010)	0.881(0.039)	0.876(0.042)	0.122(0.032)
7		tune	X	0.112(0.010)	0.892(0.034)	0.884(0.034)	0.114(0.027)

Table 3.19: Nonseparable Case. R package kernlab is used. (standard deviation is in parenthesis)

Model No.	Kernel	C	sigma	degree	scale	offset	order	Classification error	Sensitivity	Specificity	FDR	AUC
1	rbfbot	1	auto	X	X	X	X	0.114 (0.007)	0.886 (0.024)	0.886 (0.024)	0.113 (0.019)	0.956 (0.004)
2		tune	tune	X	X	X	X	0.112 (0.010)	0.886 (0.030)	0.891 (0.029)	0.109 (0.024)	0.959 (0.007)
3	polydot	1	X	1	1	1	X	0.106 (0.007)	0.892 (0.020)	0.896 (0.019)	0.104 (0.016)	0.962 (0.004)
4		1	X	2	1	1	X	0.200 (0.020)	0.800 (0.036)	0.799 (0.036)	0.200 (0.027)	0.875 (0.022)
5		1	X	3	1	1	X	0.185 (0.018)	0.815 (0.029)	0.816 (0.031)	0.184 (0.024)	0.885 (0.019)
6		1	X	4	1	1	X	0.216 (0.022)	0.783 (0.041)	0.785 (0.040)	0.215 (0.029)	0.846 (0.024)
7		1	X	5	1	1	X	0.212 (0.025)	0.782 (0.055)	0.793 (0.055)	0.207 (0.037)	0.856 (0.024)
8	vanilladot	1	X	X	X	X	X	0.106 (0.007)	0.892 (0.020)	0.896 (0.019)	0.104 (0.016)	0.962 (0.004)
9		tune	X	X	X	X	X	0.260 (0.169)	0.705 (0.365)	0.774 (0.315)	0.182 (0.155)	0.954 (0.009)
10	tanhdot	1	X	X	1	1	X	0.198 (0.012)	0.795 (0.039)	0.809 (0.032)	0.193 (0.021)	0.887 (0.011)
11	laplacedot	1	auto	X	X	X	X	0.108 (0.007)	0.892 (0.030)	0.892 (0.030)	0.107 (0.024)	0.961 (0.004)
12		tune	tune	X	X	X	X	0.109 (0.007)	0.890 (0.027)	0.892 (0.026)	0.108 (0.021)	0.960 (0.004)
13	besseldot	1	1	1	X	X	1	0.189 (0.013)	0.812 (0.045)	0.811 (0.046)	0.187 (0.031)	0.899 (0.012)
14		tune	tune	1	X	X	1	0.107 (0.008)	0.890 (0.028)	0.896 (0.025)	0.104 (0.020)	0.962 (0.004)
15		tune	tune	1	X	X	2	0.108 (0.009)	0.891 (0.029)	0.893 (0.027)	0.107 (0.022)	0.961 (0.006)
16		tune	tune	2	X	X	1	0.109 (0.009)	0.890 (0.029)	0.893 (0.026)	0.107 (0.021)	0.961 (0.005)
17		tune	tune	2	X	X	2	0.108 (0.007)	0.890 (0.027)	0.894 (0.026)	0.105 (0.021)	0.961 (0.005)
18	anovadot	1	1	1	X	X	X	0.140 (0.011)	0.857 (0.028)	0.863 (0.026)	0.137 (0.020)	0.936 (0.008)
19		tune	tune	1	X	X	X	0.109 (0.010)	0.889 (0.025)	0.892 (0.025)	0.107 (0.021)	0.960 (0.007)
20		tune	tune	2	X	X	X	0.282 (0.101)	0.726 (0.266)	0.709 (0.275)	0.228 (0.131)	0.881 (0.039)
21	splinedot	1	X	X	X	X	X	0.326 (0.039)	0.663 (0.057)	0.685 (0.055)	0.322 (0.042)	0.705 (0.046)
22		tune	X	X	X	X	X	0.317 (0.038)	0.670 (0.055)	0.697 (0.055)	0.311 (0.043)	0.715 (0.046)

Results for sparseSVM

The four models available under the sparseSVM package produced similar results. However, the model with the best combination of errors is the LASSO with default lambda value. The correct model for the linearly nonseparable case is the model which sets weights for $x_3, x_4, ..., x_{10}$ to be zero but gives a value to those of x_1 and x_2. A problem encountered while using this package was that in trying to ascertain the correct models, some data sets could not be fitted. In this case, there were 2 models not fitted for the LASSO penalty. In terms of variable selection, the LASSO penalty with a tuned lambda parameter obtained the best results being able to select 50 correct models. These results are shown in Table 3.21 that follows.

In terms of error measures, the results are identical. The LASSO with default lambda, although by a slight margin, produced the best performance measures. It should also be noted that this model also performed well in variable selection. In all, considering variable selection and performance measures, the LASSO penalty with default lambda is the best model overall. Table 3.20 that follows shows the results for all considered cases.

Table 3.20: Nonseparable Case. R package sparseSVM is used. (standard deviation is in parenthesis)

Model No.	alpha	gamma	Classification error	Sensitivity	Specificity	FDR	AUC
1	1	0.1	0.098 (0.006)	0.900 (0.023)	0.905 (0.020)	0.095 (0.017)	0.967 (0.003)
2	1	tune	0.102 (0.041)	0.891 (0.093)	0.906 (0.024)	0.095 (0.018)	0.963 (0.047)
3	tune	0.1	0.102 (0.008)	0.892 (0.038)	0.904 (0.035)	0.095 (0.028)	0.967 (0.003)
4	tune	tune	0.102 (0.008)	0.892 (0.039)	0.905 (0.035)	0.095 (0.028)	0.967 (0.003)

Table 3.21: Nonseparable Case. Result of variable selection. R package sparseSVM is used. (standard deviation is in parenthesis) CZ: Number of correct zeros, IZ: Number of incorrect zeros, CM: Number of correct models, NF: Number of models not fitted due to convergence failure

Model No.	alpha	gamma	mean(CZ)	mean(IZ)	median(CZ)	median(IZ)	CM	NF
1	1	0.1	5.68	0.00	7.00	0.00	43	2
2	1	tune	5.92	0.02	8.00	0.00	50	0
3	tune	0.1	4.76	0.00	6.00	0.00	32	0
4	tune	tune	4.95	0.00	6.00	0.00	31	0

Results for penalizedSVM

In terms of variable selection, the SCAD penalty performed best being able to select 52 correct models. As defined earlier, a correct model has nonzero weights for only x_1 and x_2. The LASSO penalty could not obtain any correct model. In terms of error measures, the SCAD penalty also had the best combination of error measures. Overall, SCAD performed better than the other penalty options available in the penalizedSVM package. Tables 3.22 and 3.23 that follow show the results for considered models in terms of error measures and variable selection respectively.

Table 3.22: Nonseparable Case. R package penalizedSVM is used. (standard deviation is in parenthesis)

Model No.	method	Classification error	Sensitivity	Specificity	FDR
1	scad	0.098 (0.007)	0.900 (0.020)	0.903 (0.018)	0.097 (0.015)
2	1norm	0.103 (0.006)	0.897 (0.021)	0.897 (0.021)	0.103 (0.017)
3	scad+L2	0.100 (0.008)	0.898 (0.027)	0.902 (0.026)	0.097 (0.021)

Table 3.23: Nonseparable Case. Result of variable selection. R package penalizedSVM is used. (standard deviation is in parenthesis) CZ: Number of correct zeros, IZ: Number of incorrect zeros, CM: Number of correct models, NF: Number of models not fitted due to convergence failure

Model No.	method	mean(CZ)	mean(IZ)	median(CZ)	median(IZ)	CM	NF
1	scad	6.47	0.00	8.00	0.00	52	0
2	1norm	2.75	0.00	3.00	0.00	0	0
3	scad+L2	6.04	0.00	7.00	0.00	43	0

Results for Other Classifiers

From Table 3.24, it can be seen that similar results were obtained. By the smallest of margins, the LDA recorded the best combination of performance measures while the KNN recorded the worst. Table 3.24 that follows shows the results for all these classifiers.

Table 3.24: Nonseparable Case. Other classifiers are used. (standard deviation is in parenthesis)

Model No.	Classification error	Sensitivity	Specificity	FDR	AUC
LDA	0.101 (0.004)	0.898 (0.016)	0.900 (0.016)	0.101 (0.013)	0.965 (0.003)
QDA	0.117 (0.007)	0.882 (0.021)	0.883 (0.022)	0.116 (0.018)	0.953 (0.005)
Logistic Reg	0.105 (0.005)	0.894 (0.019)	0.896 (0.018)	0.104 (0.015)	0.895 (0.005)
KNN	0.282 (0.021)	0.718 (0.094)	0.718 (0.099)	0.274 (0.051)	0.798 (0.026)

3.3.3 Results for Nonseparable Data Requiring Nonlinear SVM

Results for e1071

Once more, the differences are generally not very pronounced, however, the model with the best combination of performance measures is the polynomial kernel with degree 2, a default gamma and cost, and finally a tuned coef0. The polynomial kernel with degree 5 recorded the worst performance measure combinations. The performance measures of all the models are shown in Table 3.25.

Results for svmpath

The radial kernel with tuned gamma parameter produced the best combination of error measures. This confirms what we expected since the data set is supposed to be fit with nonlinear SVM. Table 3.26 shows the results for all considered cases.

Results for kernlab

The Sigmoid kernel is the worst performing given its combination of error measures,

while the best performing is the polynomial kernel (model 3 in Table 3.27) .

Table 3.25: NonLinear SVM Case. R package e1071 is used. (standard deviation is in parenthesis)

Model No.	Kernel	degree	gamma	coef0	cost	Classification error	Sensitivity	Specificity	FDR	AUC
1	linear	X	X	X	tune	0.063 (0.009)	0.935 (0.021)	0.939 (0.017)	0.061 (0.015)	0.937 (0.009)
2		X	X	X	1	0.063 (0.009)	0.934 (0.021)	0.940 (0.016)	0.060 (0.015)	0.937 (0.009)
3	polynomial	2	1/p	tune	tune	0.058 (0.013)	0.939 (0.025)	0.945 (0.019)	0.054 (0.017)	0.942 (0.013)
4		2	tune	0	tune	0.151 (0.019)	0.845 (0.045)	0.853 (0.048)	0.146 (0.036)	0.849 (0.019)
5		2	tune	tune	1	0.057 (0.010)	0.941 (0.020)	0.946 (0.019)	0.054 (0.017)	0.943 (0.010)
6		2	1/p	0	tune	0.153 (0.019)	0.846 (0.045)	0.848 (0.047)	0.150 (0.035)	0.847 (0.018)
7		2	1/p	tune	1	0.056 (0.011)	0.943 (0.020)	0.946 (0.018)	0.054 (0.016)	0.944 (0.011)
8		2	tune	0	1	0.189 (0.017)	0.803 (0.070)	0.819 (0.071)	0.178 (0.047)	0.811 (0.017)
9		2	1/p	0	1	0.197 (0.016)	0.792 (0.076)	0.814 (0.075)	0.183 (0.049)	0.803 (0.016)
10		3	1/p	tune	tune	0.059 (0.013)	0.940 (0.022)	0.943 (0.019)	0.057 (0.018)	0.941 (0.013)
11		3	tune	0	tune	0.124 (0.012)	0.872 (0.038)	0.881 (0.038)	0.118 (0.030)	0.876 (0.012)
12		3	tune	tune	1	0.059 (0.013)	0.940 (0.022)	0.942 (0.020)	0.057 (0.018)	0.941 (0.013)
13		3	1/p	0	tune	0.123 (0.013)	0.874 (0.038)	0.880 (0.041)	0.119 (0.031)	0.877 (0.013)
14		3	1/p	tune	1	0.070 (0.014)	0.928 (0.027)	0.931 (0.022)	0.068 (0.020)	0.930 (0.014)
15		3	tune	0	1	0.129 (0.013)	0.867 (0.047)	0.876 (0.049)	0.122 (0.037)	0.871 (0.013)
16		3	1/p	0	1	0.134 (0.013)	0.859 (0.051)	0.874 (0.054)	0.124 (0.041)	0.866 (0.013)
17		4	1/p	tune	tune	0.068 (0.014)	0.930 (0.027)	0.933 (0.022)	0.066 (0.020)	0.932 (0.014)
18		4	tune	0	tune	0.179 (0.027)	0.810 (0.082)	0.832 (0.080)	0.164 (0.052)	0.821 (0.027)
19		4	tune	tune	1	0.059 (0.013)	0.938 (0.025)	0.943 (0.020)	0.056 (0.019)	0.941 (0.013)
20		4	1/p	0	tune	0.178 (0.030)	0.807 (0.088)	0.836 (0.083)	0.161 (0.054)	0.822 (0.030)
21		4	1/p	tune	1	0.075 (0.017)	0.922 (0.029)	0.928 (0.022)	0.072 (0.020)	0.925 (0.017)
22		4	tune	0	1	0.180 (0.033)	0.806 (0.101)	0.835 (0.111)	0.156 (0.074)	0.821 (0.033)
23		4	1/p	0	1	0.188 (0.035)	0.795 (0.117)	0.829 (0.120)	0.160 (0.079)	0.812 (0.035)
24		5	1/p	tune	tune	0.072 (0.013)	0.926 (0.024)	0.931 (0.021)	0.069 (0.019)	0.928 (0.013)
25		5	tune	0	tune	0.169 (0.046)	0.821 (0.114)	0.842 (0.109)	0.150 (0.067)	0.831 (0.046)
26		5	tune	tune	1	0.060 (0.013)	0.938 (0.023)	0.942 (0.020)	0.057 (0.018)	0.940 (0.013)
27		5	1/p	0	tune	0.168 (0.044)	0.815 (0.118)	0.848 (0.098)	0.146 (0.062)	0.832 (0.044)
28		5	1/p	tune	1	0.074 (0.017)	0.923 (0.027)	0.929 (0.022)	0.071 (0.021)	0.926 (0.017)
29		5	tune	0	1	0.186 (0.039)	0.799 (0.124)	0.828 (0.126)	0.158 (0.082)	0.814 (0.038)
30		5	1/p	0	1	0.199 (0.038)	0.788 (0.142)	0.814 (0.138)	0.168 (0.088)	0.801 (0.038)
31		3	1/p	0	1	0.134 (0.013)	0.859 (0.051)	0.874 (0.054)	0.124 (0.041)	0.866 (0.013)
32	radial	X	tune	X	tune	0.068 (0.012)	0.930 (0.024)	0.934 (0.021)	0.066 (0.019)	0.932 (0.012)

Table 3.25 Continued

Model No.	Kernel	degree	gamma	coef0	cost	Classification error	Sensitivity	Specificity	FDR	AUC
33		X	tune	X	1	0.098 (0.012)	0.900 (0.026)	0.905 (0.025)	0.095 (0.022)	0.902 (0.012)
34		X	1/p	X	tune	0.091 (0.011)	0.907 (0.028)	0.912 (0.024)	0.088 (0.021)	0.909 (0.011)
35		X	1/p	X	1	0.093 (0.008)	0.904 (0.024)	0.910 (0.024)	0.090 (0.021)	0.907 (0.008)
36	sigmoid	X	1/p	tune	tune	0.132 (0.013)	0.866 (0.027)	0.870 (0.029)	0.130 (0.023)	0.868 (0.013)
37		X	tune	0	tune	0.059 (0.011)	0.939 (0.021)	0.943 (0.018)	0.057 (0.016)	0.941 (0.011)
38		X	tune	tune	1	0.135 (0.009)	0.860 (0.027)	0.870 (0.029)	0.130 (0.023)	0.865 (0.009)
39		X	1/p	0	tune	0.129 (0.011)	0.868 (0.027)	0.875 (0.027)	0.125 (0.022)	0.871 (0.011)
40		X	1/p	tune	1	0.126 (0.009)	0.871 (0.026)	0.878 (0.026)	0.122 (0.021)	0.874 (0.009)
41		X	tune	0	1	0.132 (0.009)	0.865 (0.024)	0.871 (0.026)	0.129 (0.021)	0.868 (0.009)
42		X	1/p	0	1	0.123 (0.008)	0.873 (0.024)	0.881 (0.024)	0.119 (0.019)	0.877 (0.008)

Table 3.26: NonLinear SVM Case. R package svmpath is used. (standard deviation is in parenthesis)

Model No.	Kernel	gamma	degree	Classification error	Sensitivity	Specificity	FDR
1	poly.kernel	X	1	0.073(0.026)	0.926(0.038)	0.928(0.044)	0.071(0.039)
2		X	2	0.096(0.013)	0.902(0.038)	0.907(0.041)	0.091(0.034)
3		X	3	0.118(0.016)	0.874(0.048)	0.890(0.046)	0.109(0.036)
4		X	4	0.142(0.067)	0.850(0.080)	0.867(0.082)	0.132(0.077)
5		X	5	0.204(0.037)	0.785(0.110)	0.806(0.115)	0.184(0.073)
6	radial.kernel	1/p	X	0.094(0.014)	0.903(0.036)	0.909(0.041)	0.090(0.032)
7		tune	X	0.069(0.015)	0.930(0.038)	0.932(0.035)	0.066(0.030)

Table 3.27: Basis Case. R package kernlab is used. (standard deviation is in parenthesis)

Model No.	Kernel	C	sigma	degree	scale	offset	order	Classification error	Sensitivity	Specificity	FDR	AUC
1	rbfbot	1	auto	X	X	X	X	0.094 (0.008)	0.902 (0.031)	0.910 (0.032)	0.090 (0.027)	0.973 (0.004)
2		tune	tune	X	X	X	X	0.083 (0.011)	0.914 (0.026)	0.919 (0.028)	0.080 (0.024)	0.978 (0.006)
3	polydot	1	X	1	1	1	X	0.063 (0.009)	0.934 (0.021)	0.940 (0.018)	0.060 (0.016)	0.988 (0.003)
4		1	X	2	1	1	X	0.100 (0.014)	0.899 (0.024)	0.901 (0.023)	0.099 (0.021)	0.962 (0.011)
5		1	X	3	1	1	X	0.129 (0.018)	0.872 (0.030)	0.870 (0.027)	0.129 (0.023)	0.931 (0.017)
6		1	X	4	1	1	X	0.135 (0.018)	0.865 (0.031)	0.865 (0.028)	0.134 (0.023)	0.921 (0.019)
7		1	X	5	1	1	X	0.135 (0.017)	0.865 (0.031)	0.865 (0.029)	0.134 (0.023)	0.916 (0.017)
8	vanilladot	1	X	X	X	X	X	0.063 (0.009)	0.934 (0.021)	0.940 (0.018)	0.060 (0.016)	0.988 (0.003)
9		tune	X	X	X	X	X	0.089 (0.011)	0.908 (0.028)	0.914 (0.025)	0.086 (0.021)	0.975 (0.006)
10	tanhdot	1	X	X	1	1	X	0.276 (0.018)	0.717 (0.043)	0.730 (0.041)	0.272 (0.024)	0.802 (0.019)
11	laplacedot	1	auto	X	X	X	X	0.110 (0.009)	0.884 (0.044)	0.895 (0.046)	0.103 (0.036)	0.967 (0.005)
12		tune	tune	X	X	X	X	0.087 (0.011)	0.911 (0.027)	0.915 (0.025)	0.084 (0.022)	0.977 (0.006)
13	besseldot	1	1	1	X	X	1	0.178 (0.019)	0.818 (0.049)	0.826 (0.047)	0.173 (0.034)	0.904 (0.019)
14		tune	tune	1	X	X	1	0.077 (0.011)	0.919 (0.026)	0.926 (0.024)	0.074 (0.021)	0.982 (0.005)
15		tune	tune	1	X	X	2	0.079 (0.011)	0.918 (0.026)	0.924 (0.026)	0.076 (0.023)	0.981 (0.005)
16		tune	tune	2	X	X	1	0.080 (0.011)	0.919 (0.027)	0.922 (0.027)	0.077 (0.023)	0.981 (0.005)
17		tune	tune	2	X	X	2	0.081 (0.013)	0.916 (0.027)	0.923 (0.026)	0.077 (0.023)	0.980 (0.007)
18	anovadot	1	1	1	X	X	X	0.065 (0.008)	0.932 (0.019)	0.938 (0.019)	0.062 (0.017)	0.985 (0.004)
19		tune	tune	1	X	X	X	0.066 (0.010)	0.932 (0.022)	0.937 (0.020)	0.063 (0.018)	0.985 (0.005)
20		tune	tune	2	X	X	X	0.197 (0.090)	0.807 (0.212)	0.799 (0.212)	0.158 (0.123)	0.949 (0.016)
21	splinedot	1	X	X	X	X	X	0.216 (0.056)	0.789 (0.063)	0.779 (0.066)	0.218 (0.059)	0.813 (0.065)
22		tune	X	X	X	X	X	0.178 (0.055)	0.835 (0.063)	0.808 (0.063)	0.186 (0.057)	0.851 (0.070)

Results for sparseSVM

The error measures for the different models are similar. The best however is the LASSO which has slightly better error measures. The package could not produce fits for two data sets due to an error (or bug). Three of the models could not identify the correct models. Here, the correct model implies the one which has nonzero coefficients for only the important variables x_1, x_2, and x_1^2. Incorrect models are those that have nonzero coefficients for $x_3, x_4, \ldots, x_{10}$ and/or zero coefficients for the important variables. Elastic net with tuned lambda is the only one that is able to correctly select 3 models. The variable selection methods obtained using the sparseSVM package are shown in Table 3.28 that follows. The error measures for the different combinations considered using the sparseSVM package are also shown in Table 3.29.

Table 3.28: Basis Case. Result of variable selection. R package sparseSVM is used. (standard deviation is in parenthesis) CZ: Number of correct zeros, IZ: Number of incorrect zeros, CM: Number of correct models, NF: Number of models not fitted due to convergence failure

Model No.	alpha	gamma	mean(CZ)	mean(IZ)	median(CZ)	median(IZ)	CM	NF
1	1	0.1	3.71	0.19	3.00	0.00	0	2
2	1	tune	3.48	0.13	3.00	0.00	0	0
3	tune	0.1	2.98	0.13	2.00	0.00	0	0
4	tune	tune	3.22	0.11	3.00	0.00	3	0

Table 3.29: Basis Case. R package sparseSVM is used. (standard deviation is in parenthesis)

Model No.	alpha	gamma	Classification error	Sensitivity	Specificity	FDR	AUC
1	1	0.1	0.056 (0.015)	0.941 (0.025)	0.948 (0.022)	0.051 (0.020)	0.990 (0.005)
2	1	tune	0.054 (0.013)	0.943 (0.023)	0.950 (0.018)	0.050 (0.017)	0.991 (0.004)
3	tune	0.1	0.058 (0.014)	0.938 (0.028)	0.947 (0.019)	0.053 (0.017)	0.990 (0.004)
4	tune	tune	0.057 (0.014)	0.939 (0.025)	0.947 (0.018)	0.053 (0.017)	0.990 (0.004)

Results for penalizedSVM

SCAD and Elastic SCAD penalties were able to identify 6 correct models which has nonzero coefficients for only the important variables x_1, x_2, and x_1^2. SCAD penalty has the best combination of error measures and given its performance in variable selection, it is the overall best performing model for the data set. Tables 3.30 and 3.31 show the results for variable selection and error measures respectively.

Table 3.30: Nonlinear SVM Case. Result of variable selection. R package penalizedSVM is used. (standard deviation is in parenthesis) CZ: Number of correct zeros, IZ: Number of incorrect zeros, CM: Number of correct models, NF: Number of models not fitted due to convergence failure

Model No.	method	mean(CZ)	mean(IZ)	median(CZ)	median(IZ)	CM	NF
1	scad	5.88	0.06	6.00	0.00	6	0
2	1norm	4.47	0.01	5.00	0.00	1	0
3	scad+L2	6.55	0.23	7.00	0.00	6	0

Table 3.31: Nonlinear SVM Case. R package penalizedSVM is used. (standard deviation is in parenthesis)

Model No.	method	Classification error	Sensitivity	Specificity	FDR
1	scad	0.048 (0.012)	0.950 (0.016)	0.954 (0.017)	0.045 (0.016)
2	1norm	0.055 (0.019)	0.944 (0.027)	0.947 (0.028)	0.053 (0.025)
3	scad+L2	0.061 (0.014)	0.936 (0.025)	0.942 (0.021)	0.058 (0.019)

Results for Other Classifiers

The Logistic Regression model outperforms the other three classifiers LDA, QDA, and KNN. The outcome of all the classifiers are shown in Table 3.32 that follows.

Table 3.32: Basis Case. Other classifiers are used. (standard deviation is in parenthesis)

Model No.	Classification error	Sensitivity	Specificity	FDR	AUC
LDA	0.091 (0.011)	0.907 (0.020)	0.911 (0.023)	0.088 (0.020)	0.977 (0.006)
QDA	0.087 (0.012)	0.911 (0.021)	0.915 (0.019)	0.085 (0.017)	0.971 (0.008)
Logistic Reg	0.059 (0.013)	0.939 (0.023)	0.944 (0.021)	0.056 (0.019)	0.941 (0.013)
KNN	0.155 (0.015)	0.838 (0.054)	0.852 (0.057)	0.146 (0.042)	0.935 (0.010)

3.3.4 Summary of Results

In sum, choosing kernel functions is key to the performance of a classifier. We learn from the simulations that using a much more complicated model than is required can negatively affect the performance of SVMs. This can be seen in the linearly separable data set case where fitting nonlinear kernel functions did not result in better performance measures compared to those obtained for the Support Vector Classifier which is less complicated. Moreover, considering penalized SVMs, some models could not be fitted when using the sparseSVM package due to a bug. Such a problem was not encountered at all when using the penalizedSVM package hence in that respect, the penalizedSVM is very useful. When using the sparseSVM package, the most useful kernel function for the various data sets in general is the LASSO while that of the penalizedSVM package is the SCAD penalty function.

Another observation made is that the LDA and QDA performed very well for the linearly separable and nonseparable data sets. This might be due to the way in which those data sets were generated. The LDA and QDA assume the classes to follow a multivariate normal distribution and this is the case in the simulations. This argument is upheld by the results seen in the nonseparable data set requiring nonlinear SVMs where the Logistic Regression model rather performs better than the LDA and QDA. The data set in this case was not generated using a multivariate normal distribution.

Chapter 4

Real Data Analysis

4.1 Fitting the Classification Methods to Alzheimer's Disease data

Data used in the preparation of this study was obtained from the Alzheimers Disease Neuroimaging Initiative (ADNI) database (http://adni.loni.usc.edu) (Jack et al., 2008). The ADNI was launched in 2003 as a public-private partnership, led by Principal Investigator Michael W. Weiner, MD. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimers disease (AD). The ADNI unites research who try and study the progression of Alzheimer's disease with the aim of obtaining a cure which is nonexistent at the moment.

Alzheimer's Disease as defined by the Alzheimer's Association is a type of dementia that causes problems with memory, thinking and behavior. The data from the ADNI's website has six (6) defined stages which are Normal Aging/Cognitively

Normal (CN), Significant Memory Concern (SMC), Early Mild Cognitive Impairment (EMCI), Mild Cognitive Impairment (MCI), Late Mild Cognitive Impairment (LMCI), and Alzheimer's disease (AD). Since we are interested in binary classification in this study, the response is set using the two stages CN and AD.

The data set is longitudinal and there are three phases in which the data for the ADNI study were collected - ADNI1, ADNI GO, and ADNI2. It was collected through clinical and imaging assessments obtained from participants who were recruited across North America.

The data obtained initially had 13,017 observations. We only chose the baseline data excluding subsequent visits which was ascertained from knowing the Participant ID. Also, we deleted missing values, SMC, MCI, EMCI, and LMCI. The resulting data had 681 observations with 11 features in addition to 3 dummy variables from Marital status making 14 variables in all. The response is either CN or AD. Of the 681 subjects, 440 were recorded to be Cognitively Normal (CN) and 241 having Alzheimer's disease on their first visit. The feature variables used include Age, Gender, Education, Marital status (1=Divorced, 2=Married, 3=Never Married, 4= Widowed), APOE4 gene, volume of ventricles, volume of hippocampus, volume of WholeBrain, volume of Entorhinal cortex, volume of Fusiform gyrus, volume of Middle temporal lobe, and Intracranial volume.

Some of the explanatory variables (or features) are described as follows. The Apolipoprotein E (APOE) is a class of proteins that have to do with the metabolism of fats in the body. The ventricular system is made up of four interconnected cavities in the brain. The first part of the brain to be damaged as a result of Alzheimer's disease is the entorhinal cortex. The Fusiform gyrus and middle temporal lobe are regions of the brain associated with recognition of known faces.

The data set was resampled 25 times into training and testing data sets as done

by Liu & Wu (2007). Each split has 80% training data and 20% testing data.

Of interest is not only a binary classification but also variable selection to ascertain the significance of the aforementioned variables to Alzheimer disease detection.

4.1.1 Results for e1071

The results obtained for the $e0171$ show similar results. The linear kernel with tuned cost parameter performed very well in classifying the data. It had the best classification error, sensitivity, and AUC. Considering specificity and FDR, the polynomial kernel with degree 5, tuned gamma, default coef0 and default cost parameter produced the best results. By and large, the sigmoid kernel performed worst in classifying AD and CN patients. Table 4.1 shows the output for the various models considered.

4.1.2 Results for svmpath

Although the results are similar, the radial basis kernel with tuned gamma recorded the best performance measures in terms of classification error and sensitivity. Polynomial kernels with degrees 2 and 5 had the worst performance measures in terms of classification error, sensitivity, specificity, and FDR. Table 4.2 gives the results for all considered models.

4.1.3 Results for kernlab

From Table 4.3, the best performing kernels are the polynomial kernel, vanilla, and Laplacian. The best kernels based on classification error and sensitivity are the polynomial and vanilla kernels. The Laplacian kernel recorded the best performance measures in terms of specificity and FDR. The worst performing models on the other hand are the ANOVA kernel and the Bessel kernel. The ANOVA kernel performs

especially poorly based on the classification error, specificity, and FDR. The Bessel kernel also performs poorly considering sensitivity and the AUC.

Table 4.1: Real data analysis. R package e1071 is used. (standard deviation is in parenthesis)

Model No.	Kernel	degree	gamma	coef0	cost	Classification error	Sensitivity	Specificity	FDR	AUC
1	linear	X	X	X	tune	0.093 (0.024)	0.848 (0.059)	0.942 (0.025)	0.108 (0.043)	0.895 (0.029)
2		X	X	X	1	0.093 (0.022)	0.841 (0.054)	0.946 (0.022)	0.102 (0.040)	0.893 (0.027)
3	polynomial	2	1/p	tune	tune	0.096 (0.020)	0.839 (0.055)	0.942 (0.020)	0.109 (0.035)	0.890 (0.027)
4		2	tune	0	tune	0.190 (0.031)	0.555 (0.079)	0.954 (0.024)	0.124 (0.056)	0.755 (0.038)
5		2	tune	tune	1	0.095 (0.021)	0.837 (0.056)	0.943 (0.022)	0.107 (0.037)	0.890 (0.027)
6		2	1/p	0	tune	0.187 (0.035)	0.564 (0.084)	0.955 (0.023)	0.121 (0.050)	0.760 (0.041)
7		2	1/p	tune	1	0.093 (0.022)	0.837 (0.055)	0.947 (0.019)	0.101 (0.035)	0.892 (0.028)
8		2	tune	0	1	0.187 (0.024)	0.590 (0.070)	0.939 (0.024)	0.151 (0.043)	0.765 (0.030)
9		2	1/p	0	1	0.197 (0.026)	0.576 (0.062)	0.932 (0.030)	0.169 (0.060)	0.754 (0.030)
10		3	1/p	tune	tune	0.092 (0.016)	0.846 (0.046)	0.943 (0.021)	0.106 (0.037)	0.895 (0.021)
11		3	tune	0	tune	0.117 (0.019)	0.771 (0.052)	0.947 (0.020)	0.108 (0.038)	0.859 (0.024)
12		3	tune	tune	1	0.094 (0.020)	0.841 (0.059)	0.944 (0.024)	0.104 (0.041)	0.892 (0.027)
13		3	1/p	0	tune	0.115 (0.018)	0.777 (0.053)	0.947 (0.021)	0.107 (0.039)	0.862 (0.023)
14		3	1/p	tune	1	0.102 (0.017)	0.813 (0.048)	0.947 (0.024)	0.102 (0.043)	0.880 (0.020)
15		3	tune	0	1	0.109 (0.018)	0.792 (0.058)	0.948 (0.020)	0.102 (0.035)	0.870 (0.024)
16		3	1/p	0	1	0.114 (0.020)	0.785 (0.057)	0.945 (0.019)	0.111 (0.037)	0.865 (0.026)
17		4	1/p	tune	tune	0.096 (0.019)	0.841 (0.058)	0.941 (0.020)	0.110 (0.035)	0.891 (0.026)
18		4	tune	0	tune	0.196 (0.038)	0.557 (0.084)	0.945 (0.028)	0.147 (0.068)	0.751 (0.044)
19		4	tune	tune	1	0.095 (0.019)	0.838 (0.058)	0.943 (0.022)	0.106 (0.036)	0.890 (0.025)
20		4	1/p	0	tune	0.194 (0.030)	0.556 (0.071)	0.947 (0.024)	0.142 (0.054)	0.752 (0.035)
21		4	1/p	tune	1	0.108 (0.018)	0.797 (0.049)	0.946 (0.023)	0.105 (0.038)	0.872 (0.021)
22		4	tune	0	1	0.181 (0.031)	0.586 (0.075)	0.951 (0.022)	0.125 (0.050)	0.769 (0.037)
23		4	1/p	0	1	0.181 (0.031)	0.594 (0.069)	0.946 (0.027)	0.134 (0.059)	0.770 (0.034)
24		5	1/p	tune	tune	0.099 (0.018)	0.828 (0.047)	0.943 (0.019)	0.107 (0.033)	0.886 (0.022)
25		5	tune	0	tune	0.153 (0.028)	0.661 (0.067)	0.953 (0.018)	0.111 (0.040)	0.807 (0.035)
26		5	tune	tune	1	0.094 (0.018)	0.834 (0.051)	0.947 (0.023)	0.101 (0.041)	0.890 (0.024)
27		5	1/p	0	tune	0.159 (0.022)	0.653 (0.059)	0.948 (0.017)	0.124 (0.035)	0.800 (0.028)
28		5	1/p	tune	1	0.133 (0.019)	0.712 (0.051)	0.955 (0.018)	0.099 (0.039)	0.834 (0.025)
29		5	tune	0	1	0.138 (0.024)	0.697 (0.059)	0.956 (0.021)	0.100 (0.043)	0.826 (0.029)
30		5	1/p	0	1	0.140 (0.022)	0.697 (0.055)	0.952 (0.019)	0.108 (0.042)	0.825 (0.027)
31		3	1/p	0	1	0.114 (0.020)	0.785 (0.057)	0.945 (0.019)	0.111 (0.037)	0.865 (0.026)
32	radial	X	tune	X	tune	0.095 (0.019)	0.842 (0.056)	0.941 (0.023)	0.110 (0.039)	0.891 (0.025)

Table 4.1 Continued

Model No.	Kernel	degree	gamma	coef0	cost	Classification error	Sensitivity	Specificity	FDR	AUC
33		X	tune	X	1	0.097 (0.018)	0.837 (0.058)	0.940 (0.020)	0.111 (0.035)	0.889 (0.025)
34		X	1/p	X	tune	0.100 (0.018)	0.831 (0.052)	0.940 (0.021)	0.113 (0.034)	0.886 (0.024)
35		X	1/p	X	1	0.094 (0.021)	0.839 (0.056)	0.945 (0.023)	0.104 (0.040)	0.892 (0.027)
36	sigmoid	X	1/p	tune	tune	0.106 (0.020)	0.830 (0.059)	0.931 (0.025)	0.127 (0.044)	0.881 (0.026)
37		X	tune	0	tune	0.094 (0.021)	0.845 (0.059)	0.941 (0.025)	0.109 (0.042)	0.893 (0.027)
38		X	tune	tune	1	0.101 (0.023)	0.834 (0.058)	0.936 (0.026)	0.119 (0.044)	0.885 (0.028)
39		X	1/p	0	tune	0.105 (0.021)	0.830 (0.064)	0.932 (0.027)	0.125 (0.046)	0.881 (0.028)
40		X	1/p	tune	1	0.155 (0.021)	0.746 (0.058)	0.903 (0.035)	0.185 (0.059)	0.825 (0.025)
41		X	tune	0	1	0.102 (0.021)	0.832 (0.059)	0.936 (0.026)	0.118 (0.045)	0.884 (0.027)
42		X	1/p	0	1	0.151 (0.024)	0.748 (0.060)	0.907 (0.035)	0.178 (0.060)	0.828 (0.028)

Table 4.2: Real data analysis. R package svmpath is used. (standard deviation is in parenthesis)

Model No.	Kernel	gamma	degree	Classification error	Sensitivity	Specificity	FDR
1	poly.kernel	X	1	0.096 (0.026)	0.838 (0.066)	0.943 (0.020)	0.108 (0.038)
2		X	2	0.137 (0.100)	0.780 (0.116)	0.909 (0.182)	0.124 (0.121)
3		X	3	0.103 (0.024)	0.788 (0.068)	0.960 (0.019)	0.082 (0.036)
4		X	4	0.120 (0.025)	0.748 (0.079)	0.955 (0.027)	0.093 (0.048)
5		X	5	0.169 (0.028)	0.604 (0.090)	0.961 (0.030)	0.095 (0.063)
6	radial.kernel	1/p	X	0.099 (0.025)	0.834 (0.066)	0.940 (0.022)	0.112 (0.035)
7		tune	X	0.095 (0.025)	0.838 (0.061)	0.945 (0.021)	0.105 (0.041)

Table 4.3: Real data analysis. R package kernlab is used. (standard deviation is in parenthesis)

Model No.	Kernel	C	sigma	degree	scale	offset	order	Classification error	Sensitivity	Specificity	FDR	AUC
1	rbfbot	1	auto	X	X	X	X	0.096 (0.021)	0.821 (0.058)	0.951 (0.020)	0.095 (0.037)	0.957 (0.013)
2		tune	tune	X	X	X	X	0.098 (0.020)	0.813 (0.061)	0.952 (0.023)	0.093 (0.041)	0.957 (0.014)
3	polydot	1	X	1	1	1	X	0.093 (0.021)	0.847 (0.053)	0.942 (0.023)	0.107 (0.040)	0.953 (0.021)
4		1	X	2	1	1	X	0.133 (0.027)	0.801 (0.062)	0.905 (0.026)	0.173 (0.044)	0.922 (0.025)
5		1	X	3	1	1	X	0.159 (0.028)	0.775 (0.057)	0.879 (0.035)	0.217 (0.060)	0.892 (0.029)
6		1	X	4	1	1	X	0.168 (0.024)	0.735 (0.071)	0.886 (0.036)	0.213 (0.052)	0.867 (0.029)
7		1	X	5	1	1	X	0.152 (0.022)	0.745 (0.052)	0.907 (0.031)	0.180 (0.053)	0.892 (0.024)
8	vanilladot	1	X	X	X	X	X	0.093 (0.021)	0.847 (0.053)	0.942 (0.023)	0.107 (0.040)	0.953 (0.021)
9		tune	X	X	X	X	X	0.131 (0.050)	0.717 (0.162)	0.957 (0.026)	0.092 (0.045)	0.952 (0.015)
10	tanhdot	1	X	X	1	1	X	0.262 (0.037)	0.621 (0.078)	0.805 (0.054)	0.353 (0.070)	0.810 (0.030)
11	laplacedot	1	auto	X	X	X	X	0.100 (0.022)	0.798 (0.061)	0.958 (0.018)	0.084 (0.034)	0.956 (0.015)
12		tune	tune	X	X	X	X	0.100 (0.022)	0.812 (0.058)	0.950 (0.021)	0.099 (0.040)	0.958 (0.015)
13	besseldot	1	1	1	X	X	1	0.363 (0.033)	0.463 (0.067)	0.736 (0.050)	0.501 (0.053)	0.585 (0.041)
14		tune	tune	1	X	X	1	0.096 (0.020)	0.832 (0.057)	0.946 (0.021)	0.103 (0.037)	0.956 (0.018)
15		tune	tune	1	X	X	2	0.095 (0.020)	0.833 (0.055)	0.946 (0.023)	0.102 (0.040)	0.956 (0.017)
16		tune	tune	2	X	X	1	0.097 (0.018)	0.825 (0.056)	0.948 (0.018)	0.101 (0.032)	0.957 (0.016)
17		tune	tune	2	X	X	2	0.097 (0.018)	0.830 (0.057)	0.946 (0.023)	0.102 (0.041)	0.956 (0.017)
18	anovadot	1	1	1	X	X	X	0.109 (0.017)	0.823 (0.052)	0.930 (0.023)	0.130 (0.038)	0.955 (0.014)
19		tune	tune	1	X	X	X	0.097 (0.019)	0.833 (0.057)	0.944 (0.021)	0.107 (0.038)	0.957 (0.017)
20		tune	tune	2	X	X	X	0.488 (0.158)	0.724 (0.365)	0.403 (0.413)	0.557 (0.124)	0.758 (0.054)
21	splinedot	1	X	X	X	X	X	0.327 (0.038)	0.545 (0.083)	0.745 (0.058)	0.450 (0.064)	0.695 (0.046)
22		tune	X	X	X	X	X	0.325 (0.039)	0.550 (0.083)	0.746 (0.058)	0.447 (0.066)	0.697 (0.047)

4.1.4 Results for sparseSVM

The Elastic net with a default gamma obtained the best error measures overall however the LASSO with a default gamma yielded a higher sensitivity. Figures 4.1 to 4.4 show the frequency at which the different variables are chosen as being significant in deciding whether a person has AD or is CN. The variables that were consistently chosen are the ICV, Middle temporal lobe, Entorhinal cortex, Hippocampus volume, ventricles volume, APOE4, marital status (if married), education level, and age. Although the other variables such as gender, marital status (if never married), marital status (if divorced), and Fusiform gyrus are not selected all the time, overall, their selection frequencies are quite high - at least 19 out of the possible 25. It is plausible that all the variables contribute to detecting the presence or absence of Alzheimer's Disease.

Table 4.4: Real data analysis. R package sparseSVM is used. (standard deviation is in parenthesis)

Model No.	alpha	gamma	Classification error	Sensitivity	Specificity	FDR	AUC
1	1	0.1	0.095 (0.023)	0.839 (0.059)	0.944 (0.024)	0.105 (0.041)	0.953 (0.021)
2	1	tune	0.095 (0.023)	0.836 (0.058)	0.944 (0.027)	0.104 (0.047)	0.952 (0.021)
3	tune	0.1	0.094 (0.023)	0.838 (0.059)	0.945 (0.024)	0.103 (0.042)	0.953 (0.020)
4	tune	tune	0.096 (0.024)	0.832 (0.063)	0.945 (0.024)	0.103 (0.044)	0.952 (0.021)

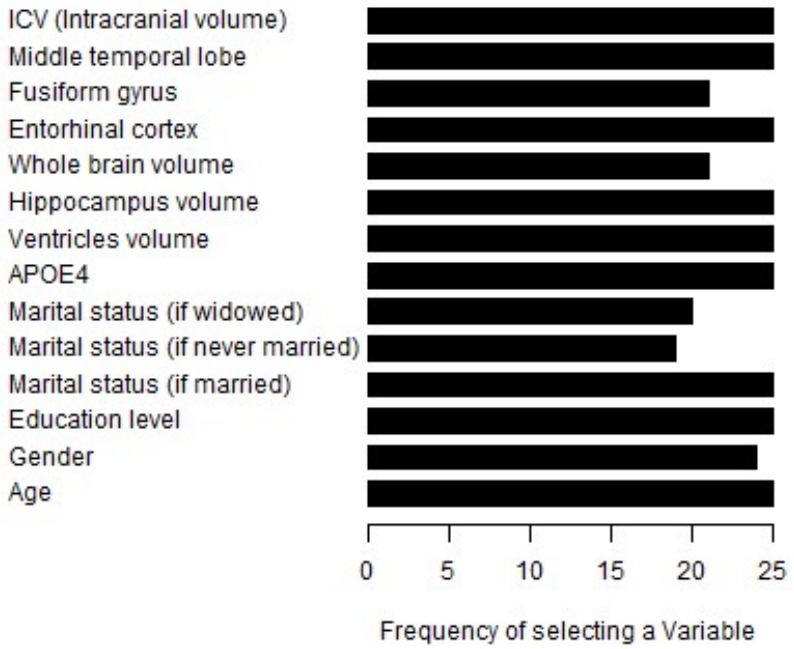

Figure 4.1: LASSO, gamma=0.1 (variable selection).

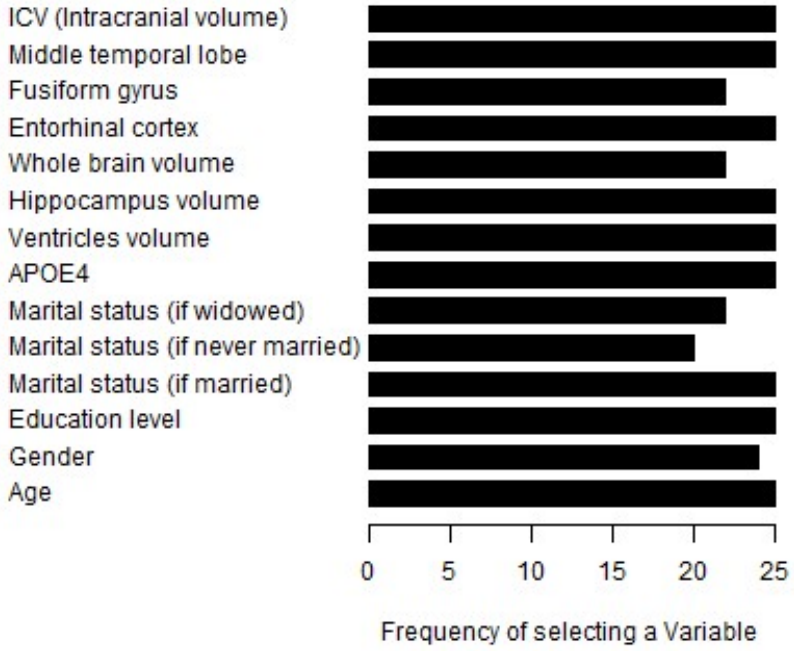

Figure 4.2: LASSO, gamma=tune (variable selection).

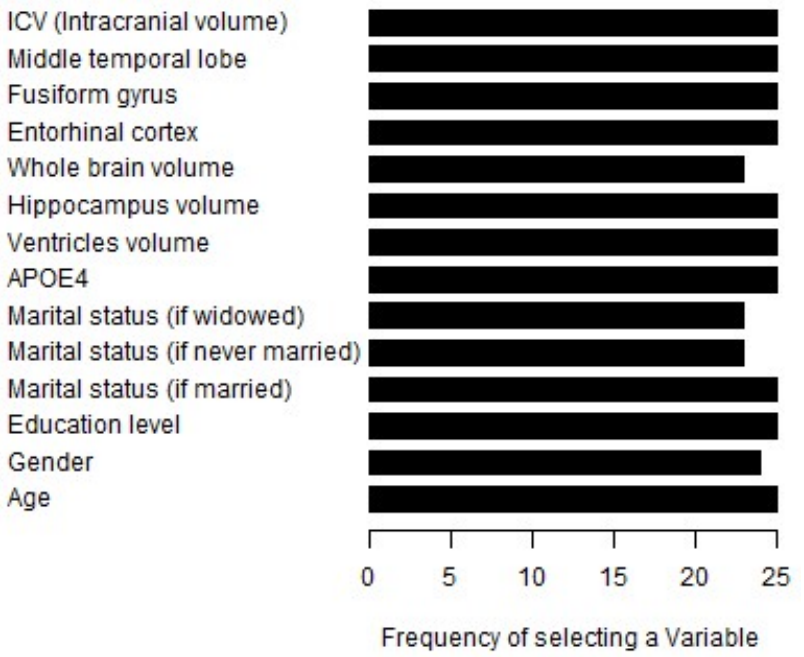

Figure 4.3: Elastic Net, gamma=0.1 (variable selection).

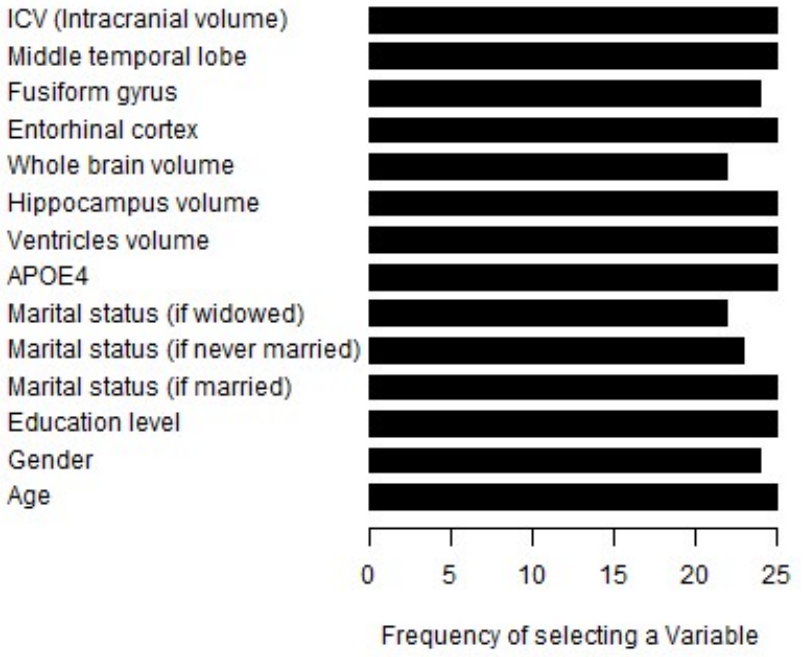

Figure 4.4: Elastic Net, gamma=tune (variable selection).

4.1.5 Results for penalizedSVM

Table 4.5 shows the three models considered using the penalizedSVM package. Taking into account the classification error, specificity and FDR, the LASSO (or 1norm) penalty function recorded the best performance measures. The SCAD penalty has the highest sensitivity. The frequency of the variables selected are also shown in Figures 4.5, 4.6, and 4.7. From the plots, the consistently selected variables include ICV, Middle temporal lobe, Entorhinal cortex, Hippocampus volume, ventricles volume, APOE4, marital status (if married), education level, and age.

Table 4.5: Real data analysis. R package penalizedSVM is used. (standard deviation is in parenthesis)

Model No.	method	Classification error	Sensitivity	Specificity	FDR
1	scad	0.093 (0.021)	0.866 (0.051)	0.930 (0.024)	0.124 (0.041)
2	1norm	0.091 (0.022)	0.847 (0.051)	0.945 (0.025)	0.101 (0.043)
3	scad+L2	0.093 (0.023)	0.853 (0.054)	0.939 (0.026)	0.112 (0.045)

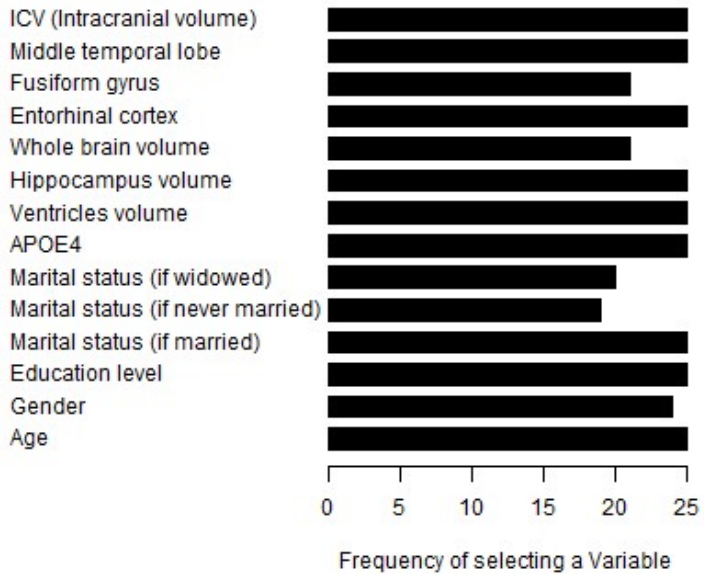

Figure 4.5: SCAD (variable selection).

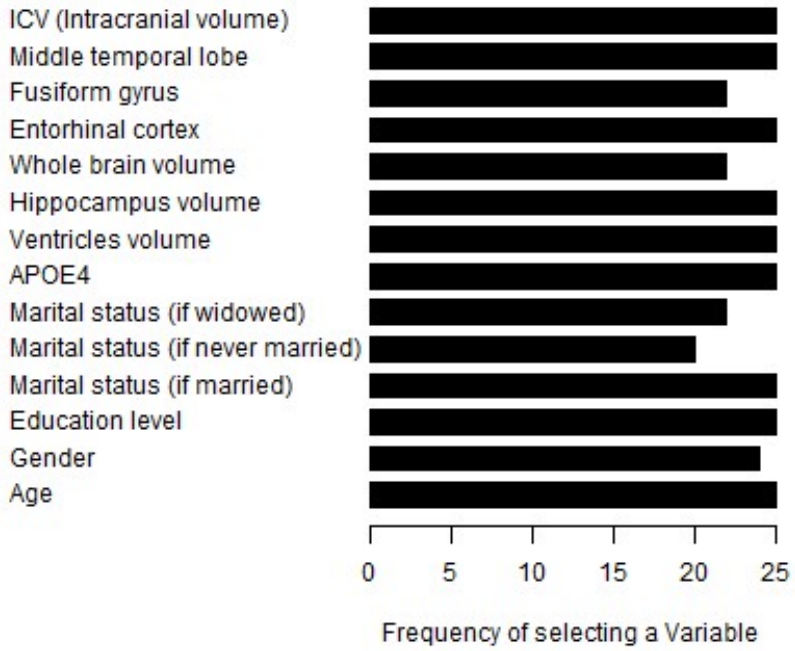

Figure 4.6: LASSO (variable selection).

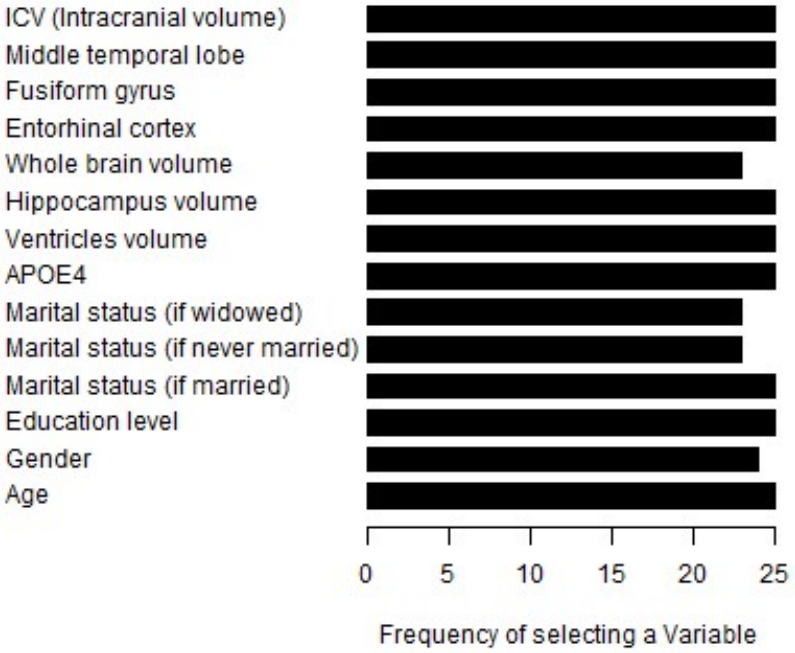

Figure 4.7: SCAD + L2 (variable selection).

4.1.6 Results for Other Classifiers

The LDA obtained the best error measures among the other classifiers. It must however be noted that the QDA and KNN also perform well in the area of sensitivity and specificity respectively. Table 4.6 shows the complete results for the classifiers.

Table 4.6: Real data analysis. Other classifiers are used. (standard deviation is in parenthesis)

Model No.	Classification error	Sensitivity	Specificity	FDR	AUC
LDA	0.088 (0.022)	0.852 (0.054)	0.947 (0.027)	0.097 (0.046)	0.954 (0.021)
QDA	0.112 (0.025)	0.871 (0.049)	0.898 (0.032)	0.170 (0.049)	0.939 (0.023)
Logistic Reg	0.090 (0.021)	0.854 (0.051)	0.943 (0.027)	0.104 (0.047)	0.899 (0.025)
KNN	0.110 (0.025)	0.782 (0.065)	0.952 (0.020)	0.098 (0.038)	0.943 (0.019)

4.1.7 Summary of Results

The frequency in which an explanatory variable is selected by the penalized SVM approach indicates its importance in classifying a Cognitively Normal person and an Alzheimer's Disease (AD) patient. The chosen data set for this study indicates that some factors are cardinal in detecting an AD patient. These factors are ICV, Middle temporal lobe, Entorhinal cortex, Hippocampus volume, ventricles volume, APOE4, marital status (if married), education level, and age. The other factors, gender, marital status (if never married), marital status (if divorced), and Fusiform gyrus, were sometimes removed from the model. However, their selection frequencies are quite high — at least 19 out of the possible 25. It is plausible that all the variables contribute to detecting the presence or absence of Alzheimer's Disease.

Since the observations in the two classes are unequally balanced, emphasis was placed more on the sensitivity, specificity, and AUC (see 2.4 for details). The performance measures of the different classifiers are similar. However, the SCAD penalty function from the penalizedSVM package produced the best result for sensitivity while the Laplacian kernel function from the kernlab package produced the best specificity result. Taking into account the other classification methods aside SVMs, the QDA recorded the best sensitivity measure while the LDA had the best performance measures of specificity and AUC. Overall, QDA gave the best sensitivity measure while the Laplacian kernel function gave the best specificity measure. It is however crucial to note that the performance measures did not differ by much.

Chapter 5

Conclusion and Remarks

In sum, this study focused on the SVM which is an extension of the simple and intuitive classifier referred to as the Maximal Margin Classifier. For a linearly separable data set, this Maximal Margin Classifier finds a decision boundary which perfectly separates the two classes such that the closest observations to the separating hyperplane are as far as possible. This is equivalent to maximizing the margin which is the distance of the closest observation to the hyperplane.

In cases where the data set is not linearly separable, there is no solution to the Maximal Margin Classifier. This necessitates the use of the Support Vector Classifier which allows violations to the margin. The Support Vector Classifier is also sometimes desired although the data set is linearly separable. This occurs in situations where the margin is narrow or when the decision boundary is not robust to new observations.

For some linearly nonseparable data, the Support Vector Classifier performs poorly compared to a nonlinear classifier. This nonlinear classifier is obtained by mapping into higher dimensions. When the data set is mapped into higher dimensions, linear separability is obtained and in that space the decision boundary is a hyperplane. However, in the original space, that same decision boundary is nonlin-

ear. Problems regarding computation intractability when mapping into higher space is done are addressed by introducing kernel functions. A kernel function is a generalization of the dot product. Kernel functions use the "kernel trick" to circumvent this computation intractability posed by mapping into higher dimensions. The Support Vector Classifier uses the dot product to obtain its solution. When the dot product is replaced with a kernel function, the SVM is created.

For statistical models, when more explanatory variables are added it seems to affect the predictive accuracy of the model. Regularization is a way to address this situation. A reasonable concern that comes up is whether the predictive accuracy of the SVM is affected by mapping into higher dimensions or adding explanatory variables. The SVM has an embedded regularization technique to curb this possible problem. The penalty required for regularization in the SVM is the Ridge regression penalty which does not perform variable selection. Altering this penalty using the LASSO, SCAD, and Elastic Net also has the potential of improving the prediction accuracy and also producing sparse solutions which will make for a less complicated model.

In the simulation setting, mostly linearly separable data sets, mostly linearly nonseparable data sets, and linearly nonseparable data sets requring nonlinear SVMs were generated. Different SVM models together with some benchmark classifiers like the KNN, LDA, QDA, and Logistic regression were also fitted on the data sets. This investigation was motivated from the no free lunch theorem which suggests that no one classifier performs best on all kinds of data sets.

In all, five packages were considered in the simulation study. Different combinations of tuned and default values of the various parameters available in the packages were fit to the data sets. The packages are the e1071, svmpath, kernlab, penalizedSVM, and sparseSVM.

For the mostly linearly separable data, the linear kernel, SCAD penalized SVM, LDA, and QDA performed very well. The good performance of the LDA and QDA can be associated with the fact its underlying assumption of the data set following a multivariate normal distribution is satisfied based on the way the simulated data sets were generated. Since these assumptions are satisfied, the LDA and QDA perform better than most of the more complex SVMs. The fact that the linear kernel performed well also seems to support the argument that adding more features when it is irrelevant to do so can unnecessarily affect the predictive accuracy of the model negatively even if by a small margin. Overall, the SCAD penalty used under the penalizedSVM package and LDA produced the best classification results (exactly the same results). In such a case, when we are interested in variable selection then the penalized SVM will be the best in classifying the data set.

Moreover, for the mostly linearly non-separable data set, the best performing kernel functions included the linear, Bessel, and Sigmoid. Considering just the kernel functions, the linear kernel overall performed best however, the LDA once more reproduced better results than the different kernel functions. The penalized SVM using the LASSO penalty however gave the best results than all the other classifiers.

In addition, for the linearly non-separable data set requiring nonlinear SVMs, the best performing models were the SVM with SCAD penalty, SVM with LASSO penalty, the polynomial kernel with degree 2, and the Logistic regression model. The SVM with SCAD penalty was the overall best performing model however, it was not always able to select the important variables. The frequency at which it selected the correct model is quite low.

In terms of using different kernel functions, the *kernlab* package proved very useful. For the penalized SVMs, the $penalizedSVM$ package was very useful in terms of fitting models to any kind of data set unlike the $sparseSVM$ package which

had bugs and hence could not fit models for all the data sets. The sparseSVM package was useful in selecting the important variables in the simulations.

Next, the models described were fitted to Alzheimer's disease data obtained from the Alzheimer's Disease Neuroimaging Initiative's website. Of interest was binary classification hence different stages of Alzheimer's disease found in the data set were deleted as well as observations with incomplete data. Furthermore, only baseline (the first visit) data were included in the analysis. The models used in the simulation were also fitted on the data set. After performing variable selection procedures on the data, it came to light that some of the factors are very key in figuring out Alzheimer's disease since they are always selected to contribute to the model. The factors are ICV, Middle temporal lobe, Entorhinal cortex, Hippocampus volume, ventricles volume, APOE4, marital status (if married), education level, and age. The other factors were sometimes missing from the model but not much of the time. In sum, it seems that all the variables contribute to detecting whether or not a person has Alzheimer's or not. This in part attributable to the fact that ADNI focuses as much as possible on collecting data on the most likely important variables. This made the variable selection factor of the SVM not be extremely pronounced.

The Alzheimer's Disease data is longitudinal and a restriction of the study was to use only the baseline data excluding subsequent visits. Looking forward, it will be great to analyze the data set using longitudinal SVMs.

References

Becker, N., Werft, W., Toedt, G., Lichter, P., & Benner, A. (2009). penalizedSVM: an R-package for feature selection SVM classification. *Bioinformatics*, *25*(13), 1711–1712.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Courant, R., & Hilbert, D. (1953). Methods of Mathematical Physics, Vol. I. *Interscience, New York*, 343–350.

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*(3), 326–334.

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., & Weingessel, A. (2005). Misc functions of the department of statistics (e1071), tu wien. *R package version*, 1–5.

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, *96*(456), 1348–1360.

Gill, P. E., Murray, W., & Wright, M. H. (1981). *Practical optimization*. Academic press.

Hastie, T. (2004). svmpath: The *SVM* Path algorithm. *R package, Version 0.9. URL http://CRAN. R-project. org*.

Hastie, T., Rosset, S., Tibshirani, R., & Zhu, J. (2004). The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, *5*(Oct), 1391–1415.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer, New York.

Hoerl, A., & Kennard, R. (1988). *Ridge regression, in encyclopedia of statistical sciences, vol. 8.* Wiley, New York.

Huang, J., & Xie, H. (2007). Asymptotic oracle properties of scad-penalized least squares estimators. In *Asymptotics: Particles, processes and inverse problems* (pp. 149–166). Institute of Mathematical Statistics.

Jack, C. R. J., Bernstein, M. A., Fox, N. C., Thompson, P., Alexander, G., Harvey, D., ... W., W. M. (2008). *The Alzheimers Disease Neuroimaging Initiative (ADNI): MRI methods.* (J Magn Reson Imaging 27, 685-691)

James, G., Witten, D., Hastie, T., & Tibsharani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer, New York.

Karatzoglou, A., Meyer, D., & Hornik, K. (2005). *Support vector machines in R.* Department of Statistics and Mathematics, WU Vienna University of Economics and Business.

Karatzoglou, A., Smola, A., Hornik, K., & Zeileis, A. (2004). kernlab - an S4 package for kernel methods in R. *Journal of statistical software*, *11*(9), 1–20.

Lee, Y., Kim, Y., Lee, S., & Koo, J.-Y. (2006). Structured multicategory support vector machines with analysis of variance decomposition. *Biometrika*, *93*(3), 555–571.

Lee, Y., Lin, Y., & Wahba, G. (2004). Multicategory support vector machines: theory and application to the classification of microarray data and satellite radiance data. *Journal of American Statistical Association*, *99*(465), 67–81.

Lin, Y. (2002). Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, *6*(3), 259–275.

Liu, Y., & Wu, Y. (2007). Variable selection via a combination of the l_0 and l_1 penalties. *Journal of Computational and Graphical Statistics*, *16*(4), 782–798.

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, *209*, 415–446.

Quenouille, M. H. (1949). Problems in plane sampling. *The Annals of Mathematical Statistics*, *3*, 355–375.

Quenouille, M. H. (1956). Notes on bias in estimation. *Biometrika*, *43*(3/4), 353–360.

Schölkopf, B., Smola, A. J., Williamson, R. C., & Bartlett, P. L. (2000). New support vector algorithms. *Neural computation*, *12*(5), 1207–1245.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Vapnik, V. (1979). *Estimation of Dependences Based on Empirical Data [in Russian]*. Nauka, Moscow.

Vapnik, V., & Chervonenkis, A. (1964). A note on a class of perceptrons. *Automation and Remote Control*, *25*.

Wang, L., & Shen, X. (2007). On l_1-norm multiclass support vector machines: methodology and theory. *Journal of the American Statistical Association*, *102*(478), 583–594.

Wang, L., Zhu, J., & Zou, H. (2006). The doubly regularized support vector machine. *Statistica Sinica*, *16*, 589–615.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67–82.

Yi, C., & Zeng, Y. (2016). sparsesvm: Solution paths of sparse linear support vector machine with lasso or elastic-net regularization [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=sparseSVM (R package version 1.1-2)

Zhu, J., Rosset, S., Tibshirani, R., & Hastie, T. J. (2004). 1-norm support vector machines. In *Advances in neural information processing systems* (pp. 49–56).

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320.